PaceMatch Connect

Source Code Documentation

Generated: 12/3/2025, 6:43:23 PM

Total Files: 172

Table of Contents

45. src/components/ui/badge.tsx
46. src/components/ui/breadcrumb.tsx
47. src/components/ui/button.tsx
48. src/components/ui/calendar.tsx
49. src/components/ui/card.tsx
50. src/components/ui/carousel.tsx
51. src/components/ui/chart.tsx
52. src/components/ui/checkbox.tsx
53. src/components/ui/collapsible.tsx
54. src/components/ui/command.tsx
55. src/components/ui/context-menu.tsx
56. src/components/ui/dialog.tsx
57. src/components/ui/drawer.tsx
58. src/components/ui/dropdown-menu.tsx
59. src/components/ui/form.tsx
60. src/components/ui/hover-card.tsx
61. src/components/ui/input-otp.tsx
62. src/components/ui/input.tsx
63. src/components/ui/label.tsx
64. src/components/ui/menubar.tsx
65. src/components/ui/navigation-menu.tsx
66. src/components/ui/pagination.tsx
67. src/components/ui/popover.tsx
68. src/components/ui/progress.tsx
69. src/components/ui/radio-group.tsx
70. src/components/ui/resizable.tsx
71. src/components/ui/scroll-area.tsx
72. src/components/ui/select.tsx
73. src/components/ui/separator.tsx
74. src/components/ui/sheet.tsx
75. src/components/ui/sidebar.tsx
76. src/components/ui/skeleton.tsx
77. src/components/ui/slider.tsx
78. src/components/ui/sonner.tsx
79. src/components/ui/switch.tsx
80. src/components/ui/table.tsx
81. src/components/ui/tabs.tsx
82. src/components/ui/textarea.tsx
83. src/components/ui/toast.tsx
84. src/components/ui/toaster.tsx
85. src/components/ui/toggle-group.tsx
86. src/components/ui/toggle.tsx
87. src/components/ui/tooltip.tsx
88. src/components/ui/use-toast.ts
89. src/contexts/NotificationContext.tsx
90. src/contexts/UserContext.tsx
91. src/hooks/use-mobile.tsx

92. src/hooks/use-toast.ts
93. src/hooks/useAdmin.ts
94. src/hooks/useAuth.ts
95. src/hooks/useLocation.ts
96. src/hooks/useMatching.ts
97. src/hooks/useMovementDetection.ts
98. src/hooks/useNearbyUsers.ts
99. src/hooks/useVenueCheckIns.ts
100. src/index.css
101. src/lib/avatars.ts
102. src/lib/dummyData.ts
103. src/lib/messageStorage.ts
104. src/lib/mockData.ts
105. src/lib/socialStorage.ts
106. src/lib/utils.ts
107. src/main.tsx
108. src/pages/AdminAnalytics.tsx
109. src/pages/AdminComments.tsx
110. src/pages/AdminDashboard.tsx
111. src/pages/AdminEvents.tsx
112. src/pages/AdminLogin.tsx
113. src/pages/AdminModeration.tsx
114. src/pages/AdminSettings.tsx
115. src/pages/AdminSetup.tsx
116. src/pages/AdminUsers.tsx
117. src/pages/AdminVenues.tsx
118. src/pages/Chat.tsx
119. src/pages/EditProfile.tsx
120. src/pages/Events.tsx
121. src/pages/Friends.tsx
122. src/pages/Index.tsx
123. src/pages/LoginScreen.tsx
124. src/pages/MapScreen.tsx
125. src/pages/Messages.tsx
126. src/pages/MyEvents.tsx
127. src/pages/NotFound.tsx
128. src/pages/PasswordReset.tsx
129. src/pages/ProfileSetup.tsx
130. src/pages/ProfileView.tsx
131. src/pages/Settings.tsx
132. src/pages/WorkoutHistory.tsx
133. src/services/adminService.ts
134. src/services/authService.ts
135. src/services/challengeService.ts
136. src/services/checkInService.ts
137. src/services/emailService.ts
138. src/services/emailServiceSimple.ts

139. src/services/encounteredUsersService.ts
140. src/services/eventService.ts
141. src/services/feedService.ts
142. src/services/firebase.ts
143. src/services/friendService.ts
144. src/services/locationService.ts
145. src/services/locationSharingService.ts
146. src/services/matchingService.ts
147. src/services/messageService.ts
148. src/services/notificationService.ts
149. src/services/pokeService.ts
150. src/services/userService.ts
151. src/services/venuePreferenceService.ts
152. src/services/venueRequestService.ts
153. src/services/venueService.ts
154. src/services/weatherService.ts
155. src/services/workoutService.ts
156. src/styles/animations.css
157. src/utils/anonymousName.ts
158. src/utils/distance.ts
159. src/utils/getLocationForWeather.ts
160. src/utils/mapIcons.ts
161. src/utils/markerOverlap.ts
162. src/utils/navigation.ts
163. src/utils/platform.ts
164. src/utils/profilePicture.ts
165. src/utils/safeArea.ts
166. src/utils/workoutState.ts
167. src/vite-env.d.ts
168. tailwind.config.ts
169. tsconfig.app.json
170. tsconfig.json
171. tsconfig.node.json
172. vite.config.ts

File: capacitor.config.ts

Lines: 78

```
 1 | import { CapacitorConfig } from '@capacitor/cli';
 2 |
 3 | const config: CapacitorConfig = {
 4 |   appId: 'com.pacematch.app',
 5 |   appName: 'PaceMatch',
 6 |   webDir: 'dist',
 7 |   // Remove server config for production builds
 8 |   // Uncomment below for live reload during development
 9 |   // server: {
10 |   //   url: 'https://pacematch-gps.web.app',
11 |   //   cleartext: true
12 |   // },
13 |   android: {
14 |     allowMixedContent: true,
15 |     // Enable location permissions
16 |     permissions: [
17 |       'ACCESS_FINE_LOCATION',
18 |       'ACCESS_COARSE_LOCATION',
19 |       'ACCESS_BACKGROUND_LOCATION',
20 |       'INTERNET',
21 |       'CAMERA',
22 |       'READ_EXTERNAL_STORAGE',
23 |       'WRITE_EXTERNAL_STORAGE'
24 |     ],
25 |     // Android build configuration
26 |     buildOptions: {
27 |       keystorePath: undefined, // Set path to your keystore for production builds
28 |       keystoreAlias: undefined, // Set your keystore alias
29 |       releaseType: undefined // 'AAB' or 'APK'
30 |     }
31 |   },
32 |   ios: {
33 |     // Enable location permissions for iOS
34 |     permissions: [
35 |       'LOCATION_WHEN_IN_USE',
36 |       'LOCATION_ALWAYS',
37 |       'CAMERA',
38 |       'PHOTO_LIBRARY'
39 |     ],
40 |     // iOS build configuration
41 |     scheme: 'pacematch',
42 |     // Add your bundle identifier if different
43 |     // bundleIdentifier: 'com.pacematch.app'
44 |   },
45 |   plugins: {
46 |     SplashScreen: {
47 |       launchShowDuration: 2000,
48 |       launchAutoHide: true,
49 |       backgroundColor: '#ffffff',
50 |       androidSplashResourceName: 'splash',
51 |       androidScaleType: 'CENTER_CROP',
52 |       showSpinner: false,
53 |       iosSpinnerStyle: 'small',
54 |       spinnerColor: '#999999'
55 |     },
56 |     StatusBar: {
57 |       style: 'dark',
58 |       backgroundColor: '#ffffff'
59 |     },
60 |     Geolocation: {
61 |       // Request location permissions
62 |       permissions: {
63 |         location: {
64 |           usage: 'always' // or 'whenInUse' for foreground only
65 |         }
66 |       }
```

```
    },
    GoogleAuth: {
      scopes: ['profile', 'email'],
      serverClientId: '891545961086-
cs7aq62rgshps172e0fiidcnh2lseiFr.apps.googleusercontent.com', // Web Client ID - Use...
      forceCodeForRefreshToken: true
    }
  }
};

export default config;

```

File: components.json

Lines: 21

```
1 | {
2 |   "$schema": "https://ui.shadcn.com/schema.json",
3 |   "style": "default",
4 |   "rsc": false,
5 |   "tsx": true,
6 |   "tailwind": {
7 |     "config": "tailwind.config.ts",
8 |     "css": "src/index.css",
9 |     "baseColor": "slate",
10 |     "cssVariables": true,
11 |     "prefix": ""
12 |   },
13 |   "aliases": {
14 |     "components": "@/components",
15 |     "utils": "@/lib/utils",
16 |     "ui": "@/components/ui",
17 |     "lib": "@/lib",
18 |     "hooks": "@/hooks"
19 |   }
20 | }
21 |
```

File: eslint.config.js

Lines: 27

```
 1 | import js from "@eslint/js";
 2 | import globals from "globals";
 3 | import reactHooks from "eslint-plugin-react-hooks";
 4 | import reactRefresh from "eslint-plugin-react-refresh";
 5 | import tseslint from "typescript-eslint";
 6 |
 7 | export default tseslint.config(
 8 |   { ignores: ["dist"] },
 9 |   {
10 |     extends: [js.configs.recommended, ...tseslint.configs.recommended],
11 |     files: ["**/*.{ts,tsx}"],
12 |     languageOptions: {
13 |       ecmaVersion: 2020,
14 |       globals: globals.browser,
15 |     },
16 |     plugins: {
17 |       "react-hooks": reactHooks,
18 |       "react-refresh": reactRefresh,
19 |     },
20 |     rules: {
21 |       ...reactHooks.configs.recommended.rules,
22 |       "react-refresh/only-export-components": ["warn", { allowConstantExport: true }],
23 |       "@typescript-eslint/no-unused-vars": "off",
24 |     },
25 |   },
26 | );
27 |
```

File: index.html

Lines: 26

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no, viewport-fit=cover" />
    <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline' 'unsafe-eval' https://apis.google.com" />
    <title>PaceMatch - Connect with Active People Nearby</title>
    <meta name="description" content="PaceMatch is a location-based social matching app for runners, cyclists, and active individuals in your area." />
    <meta name="author" content="PaceMatch" />

    <meta property="og:title" content="PaceMatch - Connect with Active People Nearby" />
    <meta property="og:description" content="Find and connect with nearby runners, cyclists, and active individuals in your area." />
    <meta property="og:type" content="website" />
    <meta property="og:image" content="https://lovable.dev/opengraph-image-p98pqg.png" />

    <meta name="twitter:card" content="summary_large_image" />
    <meta name="twitter:site" content="@PaceMatch" />
    <meta name="twitter:image" content="https://lovable.dev/opengraph-image-p98pqg.png" />
  </head>

  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>

```

File: package.json

Lines: 112

```
  1 | {
  2 |   "name": "vite_react_shadcn_ts",
  3 |   "private": true,
  4 |   "version": "0.0.0",
  5 |   "type": "module",
  6 |   "scripts": {
  7 |     "dev": "vite",
  8 |     "build": "vite build",
  9 |     "build:dev": "vite build --mode development",
 10 |     "lint": "eslint .",
 11 |     "preview": "vite preview",
 12 |     "build:android": "npm run build && npx cap sync android",
 13 |     "open:android": "npx cap open android",
 14 |     "sync:android": "npx cap sync android",
 15 |     "build:ios": "npm run build && npx cap sync ios",
 16 |     "open:ios": "npx cap open ios",
 17 |     "sync:ios": "npx cap sync ios"
 18 |   },
 19 |   "dependencies": {
 20 |     "@capacitor/android": "^7.4.4",
 21 |     "@capacitor/browser": "^7.0.2",
 22 |     "@capacitor/camera": "^7.0.2",
 23 |     "@capacitor/cli": "^7.4.4",
 24 |     "@capacitor/core": "^7.4.4",
 25 |     "@capacitor/filesystem": "^7.1.5",
 26 |     "@capacitor/geolocation": "^7.1.6",
 27 |     "@capacitor/ios": "^7.4.4",
 28 |     "@capacitor/preferences": "^7.0.2",
 29 |     "@capacitor/splash-screen": "^7.0.3",
 30 |     "@capacitor/status-bar": "^7.0.3",
 31 |     "@codetrix-studio/capacitor-google-auth": "^3.4.0-rc.4",
 32 |     "@emailjs/browser": "^4.4.1",
 33 |     "@emotion/react": "^11.14.0",
 34 |     "@emotion/styled": "^11.14.1",
 35 |     "@hookform/resolvers": "^3.10.0",
 36 |     "@mui/icons-material": "^7.3.5",
 37 |     "@mui/material": "^7.3.5",
 38 |     "@radix-ui/react-accordion": "^1.2.11",
 39 |     "@radix-ui/react-alert-dialog": "^1.1.14",
 40 |     "@radix-ui/react-aspect-ratio": "^1.1.7",
 41 |     "@radix-ui/react-avatar": "^1.1.10",
 42 |     "@radix-ui/react-checkbox": "^1.3.2",
 43 |     "@radix-ui/react-collapsible": "^1.1.11",
 44 |     "@radix-ui/react-context-menu": "^2.2.15",
 45 |     "@radix-ui/react-dialog": "^1.1.14",
 46 |     "@radix-ui/react-dropdown-menu": "^2.1.15",
 47 |     "@radix-ui/react-hover-card": "^1.1.14",
 48 |     "@radix-ui/react-label": "^2.1.7",
 49 |     "@radix-ui/react-menubar": "^1.1.15",
 50 |     "@radix-ui/react-navigation-menu": "^1.2.13",
 51 |     "@radix-ui/react-popover": "^1.1.14",
 52 |     "@radix-ui/react-progress": "^1.1.7",
 53 |     "@radix-ui/react-radio-group": "^1.3.7",
 54 |     "@radix-ui/react-scroll-area": "^1.2.9",
 55 |     "@radix-ui/react-select": "^2.2.5",
 56 |     "@radix-ui/react-separator": "^1.1.7",
 57 |     "@radix-ui/react-slider": "^1.3.5",
 58 |     "@radix-ui/react-slot": "^1.2.3",
 59 |     "@radix-ui/react-switch": "^1.2.5",
 60 |     "@radix-ui/react-tabs": "^1.1.12",
 61 |     "@radix-ui/react-toast": "^1.2.14",
 62 |     "@radix-ui/react-toggle": "^1.1.9",
 63 |     "@radix-ui/react-toggle-group": "^1.1.10",
 64 |     "@radix-ui/react-tooltip": "^1.2.7",
 65 |     "@react-google-maps/api": "^2.20.7",
 66 |     "@tanstack/react-query": "^5.83.0",
```

```
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
    "cmdk": "^1.1.1",
    "date-fns": "^3.6.0",
    "embla-carousel-react": "^8.6.0",
    "firebase": "^12.6.0",
    "framer-motion": "^12.23.24",
    "geolib": "^3.3.4",
    "input-otp": "^1.4.2",
    "lucide-react": "^0.462.0",
    "next-themes": "^0.3.0",
    "react": "^18.3.1",
    "react-day-picker": "^8.10.1",
    "react-dom": "^18.3.1",
    "react-hook-form": "^7.61.1",
    "react-resizable-panels": "^2.1.9",
    "react-router-dom": "^6.30.1",
    "recharts": "^2.15.4",
    "sonner": "^1.7.4",
    "tailwind-merge": "^2.6.0",
    "tailwindcss-animate": "^1.0.7",
    "vaul": "^0.9.9",
    "zod": "^3.25.76"
  },
  "devDependencies": {
    "@eslint/js": "^9.32.0",
    "@tailwindcss/typography": "^0.5.16",
    "@types/node": "^22.16.5",
    "@types/react": "^18.3.23",
    "@types/react-dom": "^18.3.7",
    "@vitejs/plugin-react-swc": "^3.11.0",
    "autoprefixer": "^10.4.21",
    "eslint": "^9.32.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "globals": "^15.15.0",
    "lovable-tagger": "^1.1.11",
    "pdfkit": "^0.17.2",
    "postcss": "^8.5.6",
    "tailwindcss": "^3.4.17",
    "typescript": "^5.8.3",
    "typescript-eslint": "^8.38.0",
    "vite": "^5.4.19"
  }
}
```

File: postcss.config.js

Lines: 7

```
1 | export default {
2 |   plugins: {
3 |     tailwindcss: {},
4 |     autoprefixer: {},
5 |   },
6 | };
7 |
```

File: src/App.css

Lines: 43

```
#root {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
  text-align: center;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}
.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: no-preference) {
  a:nth-of-type(2) .logo {
    animation: logo-spin infinite 20s linear;
  }
}

.card {
  padding: 2em;
}

.read-the-docs {
  color: #888;
}

```

File: src/App.tsx

Lines: 398

```
  1 | import { Toaster } from "@/components/ui/toaster";
  2 | import { Toaster as Sonner } from "@/components/ui/sonner";
  3 | import { TooltipProvider } from "@/components/ui/tooltip";
  4 | import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
  5 | import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom";
  6 | import { useState, useEffect } from "react";
  7 | import { NotificationProvider } from "@/contexts/NotificationContext";
  8 | import { UserProvider } from "@/contexts/UserContext";
  9 | import { NotificationSystem } from "@/components/NotificationSystem";
 10 | import { useNotificationContext } from "@/contexts/NotificationContext";
 11 | import LoginScreen from "./pages/LoginScreen";
 12 | import ProfileSetup from "./pages/ProfileSetup";
 13 | import MapScreen from "./pages/MapScreen";
 14 | import Events from "./pages/Events";
 15 | import MyEvents from "./pages/MyEvents";
 16 | import WorkoutHistory from "./pages/WorkoutHistory";
 17 | import Messages from "./pages/Messages";
 18 | import Chat from "./pages/Chat";
 19 | import Settings from "./pages/Settings";
 20 | import Friends from "./pages/Friends";
 21 | import EditProfile from "./pages/EditProfile";
 22 | import Index from "./pages/Index";
 23 | import NotFound from "./pages/NotFound";
 24 | import AdminLogin from "./pages/AdminLogin";
 25 | import AdminDashboard from "./pages/AdminDashboard";
 26 | import AdminUsers from "./pages/AdminUsers";
 27 | import AdminAnalytics from "./pages/AdminAnalytics";
 28 | import AdminModeration from "./pages/AdminModeration";
 29 | import AdminEvents from "./pages/AdminEvents";
 30 | import AdminComments from "./pages/AdminComments";
 31 | import AdminSettings from "./pages/AdminSettings";
 32 | import AdminVenues from "./pages/AdminVenues";
 33 | import AdminSetup from "./pages/AdminSetup";
 34 | import PasswordReset from "./pages/PasswordReset";
 35 | import { AdminProtectedRoute } from "./components/AdminProtectedRoute";
 36 | import { useAuth } from "./hooks/useAuth";
 37 | import { auth } from "./services/firebase";
 38 | import { checkAdminStatus } from "./services/adminService";
 39 | import { handleRedirectResult } from "./services/authService";
 40 |
 41 | const queryClient = new QueryClient();
 42 |
 43 | // Protected route component
 44 | const ProtectedRoute = ({ children }: { children: React.ReactNode }) => {
 45 |   const { user, loading } = useAuth();
 46 |   const [checkingStatus, setCheckingStatus] = useState(false);
 47 |   const [userStatus, setUserStatus] = useState<"active" | "suspended" | "banned" | null>(null);
 48 |
 49 |   useEffect(() => {
 50 |     const checkUserStatus = async () => {
 51 |       if (loading || !user) {
 52 |         setCheckingStatus(false);
 53 |         return;
 54 |       }
 55 |
 56 |       try {
 57 |         setCheckingStatus(true);
 58 |         const { getUserData } = await import("./services/authService");
 59 |         const userData = await getUserData(user.uid);
 60 |
 61 |         if (userData) {
 62 |           const status = userData.status || "active";
 63 |           setUserStatus(status);
 64 |
 65 |           // Check if suspension has expired
 66 |           if (status === "suspended" && userData.suspendedUntil) {
```

```
            if (userData.suspendedUntil < Date.now()) {
              // Suspension expired, but we'll let the user through
              // Admin can unsuspend them
              setUserStatus("active");
            }
          }
        } else {
          setUserStatus("active");
        }
      } catch (error) {
        console.error("Error checking user status:", error);
        setUserStatus("active"); // Default to active on error
      } finally {
        setCheckingStatus(false);
      }
    };

    checkUserStatus();
  }, [user, loading]);

  // Show loading spinner while checking auth state or user status
  if (loading || checkingStatus) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  // Check both hook state and Firebase auth directly as fallback
  // This prevents race conditions where hook hasn't updated yet but user is authenticated
  const isAuthenticated = user || auth.currentUser;

  if (!isAuthenticated) {
    return <Navigate to="/login" replace />;
  }

  // Block banned users
  if (userStatus === "banned") {
    return (
      <div className="min-h-screen flex items-center justify-center p-6">
        <div className="text-center space-y-4 max-w-md">
          <h1 className="text-2xl font-bold text-destructive">Account Banned</h1>
          <p className="text-muted-foreground">
            Your account has been permanently banned. If you believe this is an error, please
contact support. </p>
        </div>
      </div>
    );
  }

  // Block suspended users (unless suspension expired)
  if (userStatus === "suspended") {
    return (
      <div className="min-h-screen flex items-center justify-center p-6">
        <div className="text-center space-y-4 max-w-md">
          <h1 className="text-2xl font-bold text-warning">Account Suspended</h1>
          <p className="text-muted-foreground">
            Your account has been temporarily suspended. Please contact support for more
information.     </p>
        </div>
      </div>
    );
  }

  return <>{children}</>;
};

// Admin route handler - redirects to login or dashboard
const AdminRoute = () => {
  const { user, loading } = useAuth();
```

```
138 |   const [checking, setChecking] = useState(true);
139 |
140 |   useEffect(() => {
141 |     const checkAdmin = async () => {
142 |       if (loading) return;
143 |
144 |       if (!user || !user.email) {
145 |         setChecking(false);
146 |         return;
147 |       }
148 |
149 |       try {
150 |         const isAdmin = await checkAdminStatus(user.email);
151 |         if (isAdmin) {
152 |           window.location.href = "/admin/dashboard";
153 |         } else {
154 |           window.location.href = "/admin/login";
155 |         }
156 |       } catch (error) {
157 |         console.error("Error checking admin status:", error);
158 |         window.location.href = "/admin/login";
159 |       } finally {
160 |         setChecking(false);
161 |       }
162 |     };
163 |
164 |     checkAdmin();
165 |   }, [user, loading]);
166 |
167 |   if (checking || loading) {
168 |     return (
169 |       <div className="min-h-screen flex items-center justify-center">
170 |         <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
171 |       </div>
172 |     );
173 |   }
174 |
175 |   return <Navigate to="/admin/login" replace />;
176 | };
177 |
178 | const AppContent = () => {
179 |   const { notifications, dismissNotification, handleNotificationTap } = useNotificationContext();
180 |   const { user } = useAuth();
181 |
182 |   // Handle Google sign-in redirect result (prevents redirect loop)
183 |   useEffect(() => {
184 |     const checkRedirectResult = async () => {
185 |       try {
186 |         const result = await handleRedirectResult();
187 |         if (result) {
188 |           console.log("'  Google sign-in redirect handled successfully");
189 |           // User will be automatically redirected by ProtectedRoute or LoginScreen
190 |         }
191 |       } catch (error) {
192 |         console.error("Error handling redirect result:", error);
193 |         // Don't show error to user - they can try again
194 |       }
195 |     };
196 |
197 |     // Only check redirect result once on mount
198 |     checkRedirectResult();
199 |   }, []); // Empty dependency array - only run once on mount
200 |
201 |   return (
202 |     <>
203 |       <NotificationSystem
204 |         notifications={notifications}
205 |         onDismiss={dismissNotification}
206 |         onTap={handleNotificationTap}
207 |       />
208 |       <Routes>
```

```
209 |         <Route path="/login" element={<LoginScreen />} />
210 |         <Route path="/reset-password" element={<PasswordReset />} />
211 |         {/* Admin Routes */}
212 |         <Route path="/admin" element={<AdminRoute />} />
213 |         <Route path="/admin/setup" element={<AdminSetup />} />
214 |         <Route path="/admin/login" element={<AdminLogin />} />
215 |         <Route
216 |           path="/admin/dashboard"
217 |           element={
218 |             <AdminProtectedRoute>
219 |               <AdminDashboard />
220 |             </AdminProtectedRoute>
221 |           }
222 |         />
223 |         <Route
224 |           path="/admin/users"
225 |           element={
226 |             <AdminProtectedRoute>
227 |               <AdminUsers />
228 |             </AdminProtectedRoute>
229 |           }
230 |         />
231 |         <Route
232 |           path="/admin/analytics"
233 |           element={
234 |             <AdminProtectedRoute>
235 |               <AdminAnalytics />
236 |             </AdminProtectedRoute>
237 |           }
238 |         />
239 |         <Route
240 |           path="/admin/moderation"
241 |           element={
242 |             <AdminProtectedRoute>
243 |               <AdminModeration />
244 |             </AdminProtectedRoute>
245 |           }
246 |         />
247 |         <Route
248 |           path="/admin/events"
249 |           element={
250 |             <AdminProtectedRoute>
251 |               <AdminEvents />
252 |             </AdminProtectedRoute>
253 |           }
254 |         />
255 |         <Route
256 |           path="/admin/comments"
257 |           element={
258 |             <AdminProtectedRoute>
259 |               <AdminComments />
260 |             </AdminProtectedRoute>
261 |           }
262 |         />
263 |         <Route
264 |           path="/admin/settings"
265 |           element={
266 |             <AdminProtectedRoute>
267 |               <AdminSettings />
268 |             </AdminProtectedRoute>
269 |           }
270 |         />
271 |         <Route
272 |           path="/admin/venues"
273 |           element={
274 |             <AdminProtectedRoute>
275 |               <AdminVenues />
276 |             </AdminProtectedRoute>
277 |           }
278 |         />
279 |         <Route
```

```
          path="/profile-setup"
          element={
            <ProtectedRoute>
              <ProfileSetup />
            </ProtectedRoute>
          }
        />
        <Route
          path="/"
          element={
            <ProtectedRoute>
              <Index />
            </ProtectedRoute>
          }
        />
        <Route
          path="/map"
          element={
            <ProtectedRoute>
              <MapScreen />
            </ProtectedRoute>
          }
        />
        <Route
          path="/events"
          element={
            <ProtectedRoute>
              <Events />
            </ProtectedRoute>
          }
        />
        <Route
          path="/my-events"
          element={
            <ProtectedRoute>
              <MyEvents />
            </ProtectedRoute>
          }
        />
        <Route
          path="/workout-history"
          element={
            <ProtectedRoute>
              <WorkoutHistory />
            </ProtectedRoute>
          }
        />
        <Route
          path="/messages"
          element={
            <ProtectedRoute>
              <Messages />
            </ProtectedRoute>
          }
        />
        <Route
          path="/chat"
          element={
            <ProtectedRoute>
              <Chat />
            </ProtectedRoute>
          }
        />
        <Route
          path="/friends"
          element={
            <ProtectedRoute>
              <Friends />
            </ProtectedRoute>
          }
        />
        <Route
```

```
        <Route
          path="/edit-profile"
          element={
            <ProtectedRoute>
              <EditProfile />
            </ProtectedRoute>
          }
        />
        <Route
          path="/settings"
          element={
            <ProtectedRoute>
              <Settings />
            </ProtectedRoute>
          }
        />
        {/* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "*" ROUTE */}
        <Route path="*" element={<NotFound />} />
      </Routes>
    </>
  );
};

const App = () => {
  return (
    <QueryClientProvider client={queryClient}>
      <TooltipProvider>
        <Toaster />
        <Sonner />
        <BrowserRouter
          future={{
            v7_startTransition: true,
            v7_relativeSplatPath: true,
          }}
        >
          <UserProvider>
            <NotificationProvider>
              <AppContent />
            </NotificationProvider>
          </UserProvider>
        </BrowserRouter>
      </TooltipProvider>
    </QueryClientProvider>
  );
};

export default App;

```

File: src/components/AdminProtectedRoute.tsx

Lines: 39

```
// Admin protected route component
import { Navigate } from "react-router-dom";
import { useAuth } from "@/hooks/useAuth";
import { useAdmin } from "@/hooks/useAdmin";

interface AdminProtectedRouteProps {
  children: React.ReactNode;
}

/**
 * Protected route that requires both authentication and admin status
 */
export const AdminProtectedRoute = ({ children }: AdminProtectedRouteProps) => {
  const { user, loading: authLoading } = useAuth();
  const { isAdmin, loading: adminLoading } = useAdmin();

  // Show loading spinner while checking auth and admin status
  if (authLoading || adminLoading) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  // If not authenticated, redirect to login
  if (!user) {
    return <Navigate to="/login" replace />;
  }

  // If not admin, redirect to admin login
  if (!isAdmin) {
    return <Navigate to="/admin/login" replace />;
  }

  return <>{children}</>;
};

```

File: src/components/BottomNavigation.tsx

Lines: 164

```tsx
import { motion } from "framer-motion";
import { useNavigate, useLocation } from "react-router-dom";
import { useState, useEffect } from "react";
import HomeIcon from "@mui/icons-material/Home";
import MyLocationIcon from "@mui/icons-material/MyLocation";
import EventIcon from "@mui/icons-material/Event";
import MailIcon from "@mui/icons-material/Mail";
import PersonIcon from "@mui/icons-material/Person";
import { useNotificationContext } from "@/contexts/NotificationContext";
import { Capacitor } from "@capacitor/core";

const BottomNavigation = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { unreadMessageCount } = useNotificationContext();
  const [safeAreaBottom, setSafeAreaBottom] = useState(0);

  const navItems = [
    { path: "/", icon: HomeIcon, label: "Feed" },
    { path: "/events", icon: EventIcon, label: "Events" },
    { path: "/map", icon: MyLocationIcon, label: "Beacon" },
    { path: "/messages", icon: MailIcon, label: "Messages" },
    { path: "/settings", icon: PersonIcon, label: "Profile" },
  ];

  const isActive = (path: string) => location.pathname === path;

  // Calculate safe area bottom inset
  useEffect(() => {
    const updateSafeArea = () => {
      if (Capacitor.isNativePlatform()) {
        // Try to get from CSS env() first
        const root = document.documentElement;
        const computedStyle = getComputedStyle(root);
        const envBottom = computedStyle.getPropertyValue('env(safe-area-inset-bottom)');

        if (envBottom) {
          // Parse the value (e.g., "48px" -> 48)
          const value = parseFloat(envBottom);
          if (!isNaN(value)) {
            setSafeAreaBottom(value);
            return;
          }
        }

        // Fallback: Android typically has ~48px navigation bar
        // iOS varies but typically 0-34px depending on device
        setSafeAreaBottom(48); // Default Android navigation bar height
      } else {
        setSafeAreaBottom(0);
      }
    };

    updateSafeArea();
    window.addEventListener('resize', updateSafeArea);
    window.addEventListener('orientationchange', updateSafeArea);

    // Also try after a short delay to ensure CSS is applied
    const timeout = setTimeout(updateSafeArea, 100);

    return () => {
      window.removeEventListener('resize', updateSafeArea);
      window.removeEventListener('orientationchange', updateSafeArea);
      clearTimeout(timeout);
    };
  }, []);
```

```
  67 |
  68 |   return (
  69 |     <div
  70 |       className="fixed bottom-0 left-0 right-0 z-50 bg-card/95 backdrop-blur-md border-t border-
border shadow-elevation-...
  71 |       style={{
  72 |         paddingBottom: `max(env(safe-area-inset-bottom, 0px), ${safeAreaBottom}px)`,
  73 |       }}
  74 |     >
  75 |       <div className="flex items-center justify-around px-4 py-3 max-w-2xl mx-auto">
  76 |         {navItems.map((item) => {
  77 |           const Icon = item.icon;
  78 |           const active = isActive(item.path);
  79 |           const isMessages = item.path === "/messages";
  80 |           const hasUnreadMessages = isMessages && unreadMessageCount > 0;
  81 |
  82 |           // Message icon color - yellow when unread
  83 |           const messageIconColor = hasUnreadMessages
  84 |             ? "text-yellow-400"
  85 |             : active
  86 |             ? "text-primary"
  87 |             : "text-muted-foreground";
  88 |
  89 |           return (
  90 |             <motion.button
  91 |               key={item.path}
  92 |               whileTap={{ scale: 0.9 }}
  93 |               onClick={() => navigate(item.path)}
  94 |               className="flex flex-col items-center gap-1.5 min-w-[60px] touch-target relative"
  95 |             >
  96 |               <motion.div
  97 |                 animate={{
  98 |                   scale: active ? 1.15 : 1,
  99 |                   y: active ? -3 : 0,
 100 |                 }}
 101 |                 transition={{ type: "spring", stiffness: 400, damping: 17 }}
 102 |                 className="relative"
 103 |               >
 104 |                 {/* Icon container with gradient background when active */}
 105 |                 <div
 106 |                   className={`relative p-2.5 rounded-xl transition-all duration-300 ${
 107 |                     active
 108 |                       ? "bg-gradient-to-br from-primary via-primary to-success"
 109 |                       : "bg-transparent"
 110 |                   }`}
 111 |                 >
 112 |                   <Icon
 113 |                     style={{ fontSize: active ? 28 : 26 }}
 114 |                     className={
 115 |                       isMessages
 116 |                         ? hasUnreadMessages
 117 |                           ? "text-yellow-400"
 118 |                           : active
 119 |                           ? "text-white"
 120 |                           : "text-muted-foreground"
 121 |                         : active
 122 |                         ? "text-white drop-shadow-sm"
 123 |                         : "text-muted-foreground"
 124 |                     }
 125 |                   />
 126 |                 </div>
 127 |
 128 |                 {/* Red dot badge for unread messages */}
 129 |                 {hasUnreadMessages && (
 130 |                   <motion.div
 131 |                     initial={{ scale: 0 }}
 132 |                     animate={{ scale: 1 }}
 133 |                     exit={{ scale: 0 }}
 134 |                     transition={{ type: "spring", stiffness: 500, damping: 25 }}
 135 |                     className="absolute -top-1 -right-1 w-3 h-3 bg-destructive rounded-full
border-2 border-background/s...
 136 |                   />
 137 |                 )}
```

```
138 |               </motion.div>
139 |               <span
140 |                 className={`text-xs font-semibold transition-all duration-300 ${
141 |                   active
142 |                     ? "text-primary"
143 |                     : "text-muted-foreground"
144 |                 }`}
145 |               >
146 |                 {item.label}
147 |               </span>
148 |               {active && (
149 |                 <motion.div
150 |                   layoutId="activeTab"
151 |                   className="absolute -bottom-1 left-1/2 -translate-x-1/2 w-8 h-1 rounded-full
bg-gradient-to-r from-pri…
152 |                   transition={{ type: "spring", stiffness: 400, damping: 30 }}
153 |                 />
154 |               )}
155 |             </motion.button>
156 |           );
157 |         })}
158 |       </div>
159 |     </div>
160 |   );
161 | };
162 |
163 | export default BottomNavigation;
164 |
```

File: src/components/CommentDrawer.tsx

Lines: 197

```tsx
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Card } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Textarea } from "@/components/ui/textarea";
import { Avatar } from "@mui/material";
import { WorkoutPost as FirebaseWorkoutPost, Comment, addComment as addFirebaseComment, deleteComment as deleteFirebaseComment } from "@/lib/firebase";
import CloseIcon from "@mui/icons-material/Close";
import SendIcon from "@mui/icons-material/Send";
import DeleteIcon from "@mui/icons-material/Delete";
import { formatDistanceToNow } from "date-fns";
import { toast } from "sonner";

interface CommentDrawerProps {
  isOpen: boolean;
  post: FirebaseWorkoutPost | null;
  onClose: () => void;
  currentUserId: string;
  currentUsername: string;
  currentAvatar: string;
}

export const CommentDrawer = ({
  isOpen,
  post,
  onClose,
  currentUserId,
  currentUsername,
  currentAvatar,
}: CommentDrawerProps) => {
  const [commentText, setCommentText] = useState("");
  const [comments, setComments] = useState<Comment[]>([]);

  // Load comments from post when drawer opens or post changes
  useEffect(() => {
    if (post && isOpen) {
      setComments(post.comments || []);
    }
  }, [post, isOpen]);

  const handleAddComment = async () => {
    if (!commentText.trim() || !post) return;

    try {
      await addFirebaseComment(post.id, {
        userId: currentUserId,
        username: currentUsername,
        avatar: currentAvatar,
        text: commentText.trim(),
      });

      // Optimistically update UI - Firebase listener will update it properly
      const newComment: Comment = {
        id: `temp-${Date.now()}`,
        userId: currentUserId,
        username: currentUsername,
        avatar: currentAvatar,
        text: commentText.trim(),
        timestamp: Date.now(),
      };

      setComments([...comments, newComment]);
      setCommentText("");
      toast.success("Comment added!");
    } catch (error) {
      console.error("Error adding comment:", error);
```

```
      toast.error("Failed to add comment");
    }
  };

  const handleDeleteComment = async (commentId: string) => {
    if (!post) return;

    try {
      await deleteFirebaseComment(post.id, commentId, currentUserId);
      setComments(comments.filter(c => c.id !== commentId));
      toast.success("Comment deleted");
    } catch (error: any) {
      console.error("Error deleting comment:", error);
      toast.error(error.message || "Failed to delete comment");
    }
  };

  if (!isOpen || !post) return null;

  return (
    <AnimatePresence>
      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        exit={{ opacity: 0 }}
        className="fixed inset-0 z-50 bg-background/80 backdrop-blur-sm"
        onClick={onClose}
      >
        <motion.div
          initial={{ y: "100%" }}
          animate={{ y: 0 }}
          exit={{ y: "100%" }}
          transition={{ type: "spring", damping: 30, stiffness: 300 }}
          className="fixed bottom-0 left-0 right-0 max-h-[80vh] bg-card rounded-t-3xl shadow-
elevation-4"
          onClick={(e) => e.stopPropagation()}
        >
          {/* Handle bar */}
          <div className="flex justify-center pt-3 pb-2">
            <div className="w-12 h-1 bg-border rounded-full" />
          </div>

          {/* Header */}
          <div className="flex items-center justify-between px-6 py-3 border-b border-border">
            <h2 className="text-lg font-bold">Comments</h2>
            <button
              onClick={onClose}
              className="p-2 hover:bg-accent rounded-full transition-colors"
            >
              <CloseIcon />
            </button>
          </div>

          {/* Comments list */}
          <div className="overflow-y-auto max-h-[50vh] px-6 py-4 space-y-4">
            {comments.length === 0 ? (
              <div className="text-center py-8 text-muted-foreground">
                <p className="text-sm">No comments yet</p>
                <p className="text-xs mt-1">Be the first to comment!</p>
              </div>
            ) : (
              comments.map((comment) => (
                <motion.div
                  key={comment.id}
                  initial={{ opacity: 0, y: 10 }}
                  animate={{ opacity: 1, y: 0 }}
                  className="flex gap-3"
                >
                  <Avatar src={comment.avatar} alt={comment.username} sx={{ width: 36, height:
36 }} />
                  <div className="flex-1">
                    <div className="bg-muted/50 rounded-2xl px-4 py-2">
                      <p className="font-semibold text-sm">{comment.username}</p>
```

```
                      <p className="text-sm mt-1">{comment.text}</p>
                    </div>
                    <div className="flex items-center gap-3 mt-1 px-2">
                      <p className="text-xs text-muted-foreground">
                        {formatDistanceToNow(comment.timestamp, { addSuffix: true })}
                      </p>
                      {comment.userId === currentUserId && (
                        <button
                          onClick={() => handleDeleteComment(comment.id)}
                          className="text-xs text-destructive hover:text-destructive/80 flex
items-center gap-1"
                        >
                          <DeleteIcon style={{ fontSize: 14 }} />
                          Delete
                        </button>
                      )}
                    </div>
                  </div>
                </motion.div>
              ))
            )}
          </div>

          {/* Input */}
          <div className="px-6 py-4 border-t border-border bg-background">
            <div className="flex gap-3">
              <Avatar src={currentAvatar} alt={currentUsername} sx={{ width: 40, height: 40 }} />
              <div className="flex-1 flex gap-2">
                <Textarea
                  value={commentText}
                  onChange={(e) => setCommentText(e.target.value)}
                  placeholder="Add a comment..."
                  className="resize-none min-h-[44px] max-h-[120px]"
                  maxLength={280}
                  onKeyDown={(e) => {
                    if (e.key === "Enter" && !e.shiftKey) {
                      e.preventDefault();
                      handleAddComment();
                    }
                  }}
                />
                <Button
                  onClick={handleAddComment}
                  disabled={!commentText.trim()}
                  size="icon"
                  className="flex-shrink-0"
                >
                  <SendIcon style={{ fontSize: 20 }} />
                </Button>
              </div>
            </div>
            <p className="text-xs text-muted-foreground text-right mt-1">
              {commentText.length}/280
            </p>
          </div>
        </motion.div>
      </motion.div>
    </AnimatePresence>
  );
};

```

File: src/components/CreateEventModal.tsx

Lines: 1387

```
  1 | import { useState, useCallback, useRef, useEffect } from "react";
  2 | import { motion, AnimatePresence } from "framer-motion";
  3 | import { Button } from "@/components/ui/button";
  4 | import { Input } from "@/components/ui/input";
  5 | import { Label } from "@/components/ui/label";
  6 | import { Textarea } from "@/components/ui/textarea";
  7 | import { Card } from "@/components/ui/card";
  8 | import {
  9 |   Drawer,
 10 |   DrawerContent,
 11 |   DrawerHeader,
 12 |   DrawerTitle,
 13 |   DrawerDescription,
 14 | } from "@/components/ui/drawer";
 15 | import { GoogleMap, Marker, Autocomplete, useJsApiLoader } from "@react-google-maps/api";
 16 | import { Geolocation } from "@capacitor/geolocation";
 17 | import { isNativePlatform } from "@/utils/platform";
 18 | import CloseIcon from "@mui/icons-material/Close";
 19 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
 20 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
 21 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
 22 | import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
 23 | import EventIcon from "@mui/icons-material/Event";
 24 | import LocationOnIcon from "@mui/icons-material/LocationOn";
 25 | import PeopleIcon from "@mui/icons-material/People";
 26 | import DescriptionIcon from "@mui/icons-material/Description";
 27 | import MapIcon from "@mui/icons-material/Map";
 28 | import SearchIcon from "@mui/icons-material/Search";
 29 | import AccessTimeIcon from "@mui/icons-material/AccessTime";
 30 | import { toast } from "sonner";
 31 | import { z } from "zod";
 32 | import type { Event as FirebaseEvent } from "@/services/eventService";
 33 |
 34 | const libraries: ("places")[] = ["places"];
 35 | const defaultMapCenter = { lat: 14.5995, lng: 120.9842 };
 36 |
 37 | // Validation schema
 38 | const createEventSchema = z.object({
 39 |   title: z
 40 |     .string()
 41 |     .trim()
 42 |     .min(3, { message: "Title must be at least 3 characters" })
 43 |     .max(100, { message: "Title must be less than 100 characters" }),
 44 |   description: z
 45 |     .string()
 46 |     .trim()
 47 |     .min(10, { message: "Description must be at least 10 characters" })
 48 |     .max(500, { message: "Description must be less than 500 characters" }),
 49 |   activityType: z.enum(["running", "cycling", "walking", "others"], {
 50 |     required_error: "Please select an activity type",
 51 |   }),
 52 |   date: z
 53 |     .string()
 54 |     .min(1, { message: "Date is required" })
 55 |     .refine((date) => new Date(date) >= new Date(new Date().setHours(0, 0, 0, 0)), {
 56 |       message: "Date must be today or in the future",
 57 |     }),
 58 |   time: z.string().min(1, { message: "Time is required" }),
 59 |   location: z
 60 |     .string()
 61 |     .trim()
 62 |     .min(3, { message: "Location must be at least 3 characters" })
 63 |     .max(200, { message: "Location must be less than 200 characters" }),
 64 |   lat: z.number().optional(),
 65 |   lng: z.number().optional(),
 66 |   maxParticipants: z
```

```
    .number()
    .int()
    .min(2, { message: "Must allow at least 2 participants" })
    .max(1000, { message: "Cannot exceed 1000 participants" })
    .optional(),
});

export type CreateEventFormData = z.infer<typeof createEventSchema>;

interface CreateEventModalProps {
  mode?: "create" | "edit";
  eventToEdit?: FirebaseEvent | null;
  onClose: () => void;
  onCreateEvent?: (eventData: CreateEventFormData) => Promise<void> | void;
  onUpdateEvent?: (eventId: string, eventData: CreateEventFormData) => Promise<void> | void;
  initialLocation?: { lat: number; lng: number } | null;
}

export const CreateEventModal = ({
  mode = "create",
  eventToEdit,
  onClose,
  onCreateEvent,
  onUpdateEvent,
  initialLocation,
}: CreateEventModalProps) => {
  const isEditMode = mode === "edit" && !!eventToEdit;
  const determineInitialMapCenter = () => ({
    lat: eventToEdit?.lat || initialLocation?.lat || defaultMapCenter.lat,
    lng: eventToEdit?.lng || initialLocation?.lng || defaultMapCenter.lng,
  });
  const [formData, setFormData] = useState<Partial<CreateEventFormData>>(() => {
    if (isEditMode && eventToEdit) {
      return {
        title: eventToEdit.title,
        description: eventToEdit.description,
        activityType: (eventToEdit.type as CreateEventFormData["activityType"]) || "running",
        date: eventToEdit.date,
        time: eventToEdit.time,
        location: eventToEdit.location,
        maxParticipants: eventToEdit.maxParticipants,
        lat: eventToEdit.lat,
        lng: eventToEdit.lng,
      };
    }

    return {
      activityType: "running",
      maxParticipants: undefined,
      lat: initialLocation?.lat,
      lng: initialLocation?.lng,
    };
  });
  const [errors, setErrors] = useState<Partial<Record<keyof CreateEventFormData, string>>>({});
  const [isSubmitting, setIsSubmitting] = useState(false);

  // Map picker state
  const [showMapPicker, setShowMapPicker] = useState(false);
  const [selectedLocation, setSelectedLocation] = useState<{ lat: number; lng: number; address?:
string } | null(null);
>(determineInitialMapCenter);
  const [mapPickerCenter, setMapPickerCenter] = useState<{ lat: number; lng: number }
  const [liveMapCenter, setLiveMapCenter] = useState<{ lat: number; lng: number } |
null>(determineInitialMapCenter);
  const [mapPickerZoom, setMapPickerZoom] = useState(13);
  const [mapPickerRef, setMapPickerRef] = useState<google.maps.Map | null>(null);
  const [userLocation, setUserLocation] = useState<{ lat: number; lng: number } | null>(null);
  const [isLoadingUserLocation, setIsLoadingUserLocation] = useState(false);
  const autocompleteRef = useRef<google.maps.places.Autocomplete | null>(null);
  const geocoderRef = useRef<google.maps.Geocoder | null>(null);

  useEffect(() => {
    if (isEditMode && eventToEdit) {
      setFormData({
```

```
        title: eventToEdit.title,
        description: eventToEdit.description,
        activityType: (eventToEdit.type as CreateEventFormData["activityType"]) || "running",
        date: eventToEdit.date,
        time: eventToEdit.time,
        location: eventToEdit.location,
        maxParticipants: eventToEdit.maxParticipants,
        lat: eventToEdit.lat,
        lng: eventToEdit.lng,
      });
      setSelectedLocation(null);
      const updatedCenter = {
        lat: eventToEdit.lat || initialLocation?.lat || defaultMapCenter.lat,
        lng: eventToEdit.lng || initialLocation?.lng || defaultMapCenter.lng,
      };
      setMapPickerCenter(updatedCenter);
      setLiveMapCenter(updatedCenter);
    }
  }, [isEditMode, eventToEdit, initialLocation]);

  // Google Maps API loader - Use same ID as Events page to avoid loader conflict
  const googleMapsApiKey = import.meta.env.VITE_GOOGLE_MAPS_API_KEY;
  const { isLoaded: isMapLoaded, loadError: mapLoadError } = useJsApiLoader({
    id: 'google-map-script', // Same ID as Events.tsx to share the loader instance
    googleMapsApiKey: googleMapsApiKey || '',
    libraries: libraries
  });

  // Get user location when component mounts or map picker opens
  useEffect(() => {
    const getUserLocation = async () => {
      setIsLoadingUserLocation(true);

      try {
        let loc: { lat: number; lng: number } | null = null;

        if (isNativePlatform()) {
          // Use Capacitor Geolocation for native apps (iOS/Android)
          try {
            // Request permissions first
            const permissionStatus = await Geolocation.requestPermissions();

            if (permissionStatus.location === 'granted') {
              const position = await Geolocation.getCurrentPosition({
                enableHighAccuracy: true,
                timeout: 10000
              });

              loc = {
                lat: position.coords.latitude,
                lng: position.coords.longitude,
              };
            } else {
              console.warn("Location permission denied on native platform");
            }
          } catch (error) {
            console.warn("Could not get user location (native):", error);
          }
        } else {
          // Use browser geolocation for web
          if (navigator.geolocation) {
            await new Promise<void>((resolve, reject) => {
              navigator.geolocation.getCurrentPosition(
                (position) => {
                  loc = {
                    lat: position.coords.latitude,
                    lng: position.coords.longitude,
                  };
                  resolve();
                },
                (error) => {
```

```
209 |                   console.warn("Could not get user location (web):", error);
210 |                   reject(error);
211 |                 }
212 |               );
213 |             });
214 |           }
215 |         }
216 |
217 |         if (loc) {
218 |           setUserLocation(loc);
219 |
220 |           // If map picker is open, center on user location
221 |           if (showMapPicker) {
222 |             setMapPickerCenter(loc);
223 |             setLiveMapCenter(loc);
224 |             setMapPickerZoom(15);
225 |
226 |             // Also pan the map if ref exists
227 |             if (mapPickerRef) {
228 |               mapPickerRef.panTo(loc);
229 |               mapPickerRef.setZoom(15);
230 |             }
231 |           }
232 |         }
233 |       } catch (error) {
234 |         console.warn("Error getting user location:", error);
235 |       } finally {
236 |         setIsLoadingUserLocation(false);
237 |       }
238 |     };
239 |
240 |     getUserLocation();
241 |   }, [showMapPicker, mapPickerRef]);
242 |
243 |   useEffect(() => {
244 |     if (
245 |       isMapLoaded &&
246 |       !geocoderRef.current &&
247 |       typeof window !== "undefined" &&
248 |       window.google?.maps?.Geocoder
249 |     ) {
250 |       geocoderRef.current = new window.google.maps.Geocoder();
251 |     }
252 |   }, [isMapLoaded]);
253 |
254 |   useEffect(() => {
255 |     setLiveMapCenter(mapPickerCenter);
256 |   }, [mapPickerCenter]);
257 |
258 |   // Center map on user location when it becomes available (handles async loading)
259 |   useEffect(() => {
260 |     if (showMapPicker && userLocation && mapPickerRef && !selectedLocation) {
261 |       // Only auto-center if no location is selected yet
262 |       mapPickerRef.panTo({ lat: userLocation.lat, lng: userLocation.lng });
263 |       mapPickerRef.setZoom(15);
264 |       setMapPickerCenter(userLocation);
265 |       setLiveMapCenter(userLocation);
266 |       setMapPickerZoom(15);
267 |     }
268 |   }, [userLocation, showMapPicker, mapPickerRef, selectedLocation]);
269 |
270 |   // Detect mobile viewport - use initial check only, avoid re-renders during typing
271 |   const isMobile = typeof window !== 'undefined' && window.innerWidth < 640;
272 |
273 |   const activities = [
274 |     { id: "running", label: "Running", icon: DirectionsRunIcon, color: "success" },
275 |     { id: "cycling", label: "Cycling", icon: DirectionsBikeIcon, color: "primary" },
276 |     { id: "walking", label: "Walking", icon: DirectionsWalkIcon, color: "warning" },
277 |     { id: "others", label: "Others", icon: FitnessCenterIcon, color: "secondary" },
278 |   ] as const;
279 |
```

```
280 |   const handleInputChange = useCallback((field: keyof CreateEventFormData, value: string |
number) => {
281 |     setFormData((prev) => ({ ...prev, [field]: value }));
282 |     // Clear error for this field only if it exists
283 |       setErrors((prev) => {
284 |       if (prev[field]) {
285 |         const newErrors = { ...prev };
286 |         delete newErrors[field];
287 |         return newErrors;
288 |       }
289 |       return prev; // Return same object reference if no change
290 |       });
291 |   }, []);
292 |
293 |   // Handle opening map picker
294 |   const handleOpenMapPicker = useCallback(() => {
295 |     setShowMapPicker(true);
296 |
297 |     if (formData.lat && formData.lng) {
298 |       const existingSelection = {
299 |         lat: formData.lat,
300 |         lng: formData.lng,
301 |         address: formData.location,
302 |       };
303 |       setSelectedLocation(existingSelection);
304 |       const coords = { lat: formData.lat, lng: formData.lng };
305 |       setMapPickerCenter(coords);
306 |       setLiveMapCenter(coords);
307 |       setMapPickerZoom(15);
308 |       return;
309 |     }
310 |
311 |     setSelectedLocation(null);
312 |     // Center map on user location if available, otherwise use initialLocation or default
313 |     if (userLocation) {
314 |       setMapPickerCenter(userLocation);
315 |       setLiveMapCenter(userLocation);
316 |       setMapPickerZoom(14);
317 |     } else if (initialLocation) {
318 |       setMapPickerCenter(initialLocation);
319 |       setLiveMapCenter(initialLocation);
320 |       setMapPickerZoom(14);
321 |     }
322 |   }, [userLocation, initialLocation, formData.lat, formData.lng, formData.location]);
323 |
324 |   // Handle closing map picker
325 |   const handleCloseMapPicker = useCallback(() => {
326 |     setShowMapPicker(false);
327 |     setSelectedLocation(null);
328 |   }, []);
329 |
330 |   /**
331 |    * Reverse geocodes coordinates to provide a friendlier address label.
332 |    */
333 |   const resolveAddressFromCoords = useCallback(
334 |     ({ lat, lng }: { lat: number; lng: number }) =>
335 |       new Promise<string | undefined>((resolve) => {
336 |         if (!geocoderRef.current) {
337 |           resolve(undefined);
338 |           return;
339 |         }
340 |
341 |         geocoderRef.current.geocode(
342 |           { location: { lat, lng } },
343 |           (results, status) => {
344 |             if (status === "OK" && results && results.length > 0) {
345 |               resolve(results[0].formatted_address);
346 |             } else {
347 |               console.warn("Reverse geocoding failed:", status);
348 |               resolve(undefined);
349 |             }
350 |           }
```

```
351 |         );
352 |       }),
353 |     []
354 |   );
355 |
356 |   /**
357 |    * Updates local state with the provided coordinates and enriches them
358 |    * with a resolved address when the Maps API returns one.
359 |    */
360 |   const updateLocationFromCoordinates = useCallback(
361 |     ({ lat, lng, address }: { lat: number; lng: number; address?: string }) => {
362 |       setSelectedLocation(address ? { lat, lng, address } : { lat, lng });
363 |
364 |       if (!address) {
365 |         resolveAddressFromCoords({ lat, lng }).then((resolvedAddress) => {
366 |           if (!resolvedAddress) return;
367 |           setSelectedLocation((prev) => {
368 |             if (!prev || prev.lat !== lat || prev.lng !== lng) {
369 |               return prev;
370 |             }
371 |             return { lat, lng, address: resolvedAddress };
372 |           });
373 |         });
374 |       }
375 |
376 |       // Always update React state to ensure sync even if component re-renders
377 |       setMapPickerCenter({ lat, lng });
378 |       setLiveMapCenter({ lat, lng });
379 |       setMapPickerZoom(15);
380 |
381 |       // Also pan the map if ref exists (for immediate visual feedback)
382 |       if (mapPickerRef) {
383 |         mapPickerRef.panTo({ lat, lng });
384 |         mapPickerRef.setZoom(15);
385 |       }
386 |     },
387 |     [resolveAddressFromCoords, mapPickerRef]
388 |   );
389 |
390 |   // Handle map click to select location
391 |   const handleMapClick = useCallback(
392 |     (e: google.maps.MapMouseEvent) => {
393 |       if (!e.latLng) return;
394 |       const lat = e.latLng.lat();
395 |       const lng = e.latLng.lng();
396 |       updateLocationFromCoordinates({ lat, lng });
397 |     },
398 |     [updateLocationFromCoordinates]
399 |   );
400 |
401 |   /**
402 |    * Snapshot the current visible center so the user can drop a pin there.
403 |    */
404 |   const handleMapIdle = useCallback(() => {
405 |     if (!mapPickerRef) return;
406 |     const center = mapPickerRef.getCenter();
407 |     if (!center) return;
408 |     setLiveMapCenter({
409 |       lat: center.lat(),
410 |       lng: center.lng(),
411 |     });
412 |   }, [mapPickerRef]);
413 |
414 |   const handleCenterPinDrop = useCallback(() => {
415 |     if (!liveMapCenter) return;
416 |     updateLocationFromCoordinates(liveMapCenter);
417 |   }, [liveMapCenter, updateLocationFromCoordinates]);
418 |
419 |   // Handle place selection from Autocomplete
420 |   const handlePlaceSelect = useCallback(() => {
421 |     if (!autocompleteRef.current) return;
```

```
422 |     const place = autocompleteRef.current.getPlace();
423 |     if (!place.geometry?.location) return;
424 |
425 |     updateLocationFromCoordinates({
426 |       lat: place.geometry.location.lat(),
427 |       lng: place.geometry.location.lng(),
428 |       address: place.formatted_address || place.name || "",
429 |     });
430 |   }, [updateLocationFromCoordinates]);
431 |
432 |   // Handle confirming selected location
433 |   const handleConfirmLocation = useCallback(() => {
434 |     if (selectedLocation) {
435 |       handleInputChange("lat", selectedLocation.lat);
436 |       handleInputChange("lng", selectedLocation.lng);
437 |       // Update location field with address if available, otherwise coordinates
438 |       if (selectedLocation.address) {
439 |         handleInputChange("location", selectedLocation.address);
440 |       } else {
441 |         handleInputChange("location", `${selectedLocation.lat.toFixed(6)},
${selectedLocation.lng.toFixed(6)}`);
442 |       }
443 |       handleCloseMapPicker();
444 |       toast.success("Location selected");
445 |     }
446 |   }, [selectedLocation, handleInputChange, handleCloseMapPicker]);
447 |
448 |   const handleSubmit = async (e: React.FormEvent) => {
449 |     e.preventDefault();
450 |     setIsSubmitting(true);
451 |     setErrors({});
452 |
453 |     try {
454 |       // Decide which location string to validate with
455 |       const locationValue = !isEditMode && initialLocation
456 |         ? `${initialLocation.lat.toFixed(6)}, ${initialLocation.lng.toFixed(6)}`
457 |         : formData.location || eventToEdit?.location || "";
458 |
459 |       // Validate form data
460 |       const validatedData = createEventSchema.parse({
461 |         ...formData,
462 |         location: locationValue || "",
463 |         maxParticipants: formData.maxParticipants
464 |           ? Number(formData.maxParticipants)
465 |           : undefined,
466 |       });
467 |
468 |       const resolvedLat = isEditMode
469 |         ? formData.lat ?? eventToEdit?.lat
470 |         : initialLocation?.lat ?? formData.lat;
471 |       const resolvedLng = isEditMode
472 |         ? formData.lng ?? eventToEdit?.lng
473 |         : initialLocation?.lng ?? formData.lng;
474 |
475 |       // Include selected location coordinates (prioritize context-specific fallbacks)
476 |       const eventDataWithLocation = {
477 |         ...validatedData,
478 |         lat: resolvedLat,
479 |         lng: resolvedLng,
480 |       };
481 |
482 |       if (isEditMode) {
483 |         if (!eventToEdit?.id) {
484 |           throw new Error("Missing event identifier");
485 |         }
486 |         if (!onUpdateEvent) {
487 |           toast.error("Update handler not available");
488 |           return;
489 |         }
490 |         await Promise.resolve(onUpdateEvent(eventToEdit.id, eventDataWithLocation));
491 |         toast.success("Event updated successfully!");
492 |       } else {
```

```
493 |         if (!onCreateEvent) {
494 |           toast.error("Create handler not available");
495 |           return;
496 |         }
497 |         await Promise.resolve(onCreateEvent(eventDataWithLocation));
498 |         toast.success("Event created successfully!");
499 |       }
500 |
501 |       onClose();
502 |     } catch (error) {
503 |       if (error instanceof z.ZodError) {
504 |         const fieldErrors: Partial<Record<keyof CreateEventFormData, string>> = {};
505 |         error.errors.forEach((err) => {
506 |           if (err.path[0]) {
507 |             fieldErrors[err.path[0] as keyof CreateEventFormData] = err.message;
508 |           }
509 |         });
510 |         setErrors(fieldErrors);
511 |         toast.error("Please fix the form errors");
512 |       } else {
513 |         toast.error(isEditMode ? "Failed to update event" : "Failed to create event");
514 |       }
515 |     } finally {
516 |       setIsSubmitting(false);
517 |     }
518 |   };
519 |
520 |   const getTodayDate = () => {
521 |     const today = new Date();
522 |     return today.toISOString().split("T")[0];
523 |   };
524 |
525 |   // Map Picker Component
526 |   const renderMapPicker = () => {
527 |     if (!showMapPicker) return null;
528 |
529 |     return (
530 |       <AnimatePresence>
531 |         {isMobile ? (
532 |           // Mobile: Full-screen Drawer
533 |           <Drawer open={showMapPicker} onOpenChange={(open) => !open && handleCloseMapPicker()}>
534 |             <DrawerContent className="max-h-[100vh] h-[100vh] p-0">
535 |               <DrawerHeader className="border-b border-border p-4">
536 |                 <div className="flex items-center justify-between">
537 |                   <div>
538 |                     <DrawerTitle className="text-lg">Select Location</DrawerTitle>
539 |                     <DrawerDescription className="text-xs">
540 |                       Search, click, or use the center crosshair to drop a pin
541 |                     </DrawerDescription>
542 |                   </div>
543 |                   <Button
544 |                     variant="ghost"
545 |                     size="icon"
546 |                     onClick={handleCloseMapPicker}
547 |                     className="h-8 w-8"
548 |                   >
549 |                     <CloseIcon style={{ fontSize: 20 }} />
550 |                   </Button>
551 |                 </div>
552 |
553 |                 {/* Search Bar */}
554 |                 {isMapLoaded && (
555 |                   <div className="mt-3">
556 |                     <div className="relative">
557 |                       <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2
text-muted-foreground" s...  <Autocomplete
559 |                         onLoad={(autocomplete) => {
560 |                           autocompleteRef.current = autocomplete;
561 |                         }}
562 |                         onPlaceChanged={handlePlaceSelect}
563 |                         options={{
```

```
                          fields: ["geometry", "formatted_address", "name"],
                        }}
                      >
                        <Input
                          placeholder="Search for a place..."
                          className="pl-10 h-11"
                        />
                      </Autocomplete>
                    </div>
                  </div>
                )}
              </DrawerHeader>

              <div className="flex-1 relative">
                {mapLoadError ? (
                  <div className="w-full h-full flex items-center justify-center bg-muted">
                    <div className="text-center p-4">
                      <p className="text-destructive font-semibold">Error loading map</p>
                      <p className="text-sm text-muted-foreground">Please check your Google Maps
API key</p>
                    </div>
                  </div>
                ) : !isMapLoaded ? (
                  <div className="w-full h-full flex items-center justify-center bg-muted">
                    <div className="text-center">
                      <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-
primary mx-auto mb-4"></div> <p className="text-muted-foreground">Loading map...</p>
                    </div>
                  </div>
                ) : (
                  <GoogleMap
                    mapContainerStyle={{ width: "100%", height: "100%" }}
                    center={mapPickerCenter}
                    zoom={mapPickerZoom}
                    onLoad={(map) => {
                      setMapPickerRef(map);
                      if (
                        typeof window !== "undefined" &&
                        window.google?.maps?.Geocoder &&
                        !geocoderRef.current
                      ) {
                        geocoderRef.current = new window.google.maps.Geocoder();
                      }
                      const currentCenter = map.getCenter();
                      if (currentCenter) {
                        setLiveMapCenter({
                          lat: currentCenter.lat(),
                          lng: currentCenter.lng(),
                        });
                      }
                      if (userLocation) {
                        map.panTo({ lat: userLocation.lat, lng: userLocation.lng });
                        map.setZoom(15);
                      }
                    }}
                    onClick={handleMapClick}
                    onIdle={handleMapIdle}
                    options={{
                      disableDefaultUI: true,
                      zoomControl: true,
                      streetViewControl: false,
                      mapTypeControl: false,
                      fullscreenControl: false,
                      gestureHandling: "greedy",
                      draggableCursor: "crosshair",
                    }}
                  >
                    {/* User Location Marker */}
                    {userLocation && (
                      <Marker
                        position={{ lat: userLocation.lat, lng: userLocation.lng }}
                        icon={{
```

```
                          url: "https://maps.google.com/mapfiles/ms/icons/blue-dot.png",
                          scaledSize: isMapLoaded && window.google ? new
window.google.maps.Size(32, 32) : undefined,
                        title="Your location"
                      />
                    )}

                    {/* Selected Location Marker */}
                    {selectedLocation && (
                      <Marker
                        position={{ lat: selectedLocation.lat, lng: selectedLocation.lng }}
                        icon={{
                          url: "https://maps.google.com/mapfiles/ms/icons/red-dot.png",
                          scaledSize: isMapLoaded && window.google ? new
window.google.maps.Size(48, 48) : undefined,
                        title="Selected location"
                      />
                    )}
                  </GoogleMap>
                )}

                {isMapLoaded && !mapLoadError && (
                  <>
                    {/* Center crosshair */}
                    <div className="pointer-events-none absolute inset-0 flex items-center
justify-center z-10">
                      <div className="relative h-14 w-14 flex items-center justify-center">
                        <span className="absolute left-1/2 top-0 h-14 w-[2px] -translate-x-1/2
bg-primary/70"></span>
                        <span className="absolute top-1/2 left-0 w-14 h-[2px] -translate-y-1/2
bg-primary/70"></span>
                        <div className="w-6 h-6 rounded-full border-2 border-primary/80 bg-
background/60 backdrop-blur-s</div>
                      </div>
                    </div>

                    {/* Loading user location indicator */}
                    {isLoadingUserLocation && (
                      <div className="absolute top-4 left-1/2 -translate-x-1/2 z-20 bg-
background/90 backdrop-blur-sm px-4 py-2 rounded-full shadow-lg border border-
primary">
                        <div className="flex items-center gap-2">
                          <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-
primary"></div>
                          <p className="text-sm text-muted-foreground">Getting your location...</
p>
                        </div>
                      </div>
                    )}
                  </>
                )}

                {/* Selected Location Info and Actions */}
                <div className="absolute bottom-0 left-0 right-0 bg-card/95 backdrop-blur-sm
border-t border-border p-4">
                  {selectedLocation ? (
                    <div className="space-y-3">
                      <div>
                        <p className="text-sm font-semibold">Selected Location</p>
                        <p className="text-xs text-muted-foreground">
                          {selectedLocation.address || `${selectedLocation.lat.toFixed(6)},
${selectedLocation.lng.toFix...</p>
                      </div>
                      <div className="flex gap-2">
                        <Button
                          variant="outline"
                          onClick={handleCloseMapPicker}
                          className="flex-1 h-11"
                        >
                          Cancel
                        </Button>
                        <Button
                          onClick={handleConfirmLocation}
                          className="flex-1 h-11"
                        >
                          Use This Location
                        </Button>
                      </div>
                    </div>
                  ) : (
```

```
                    <div className="space-y-3 text-center">
                      <p className="text-sm text-muted-foreground">
                        Click anywhere on the map or align the crosshair, then drop the pin.
                      </p>
                      <Button
                        variant="secondary"
                        onClick={handleCenterPinDrop}
                        className="w-full h-10"
                        disabled={!isMapLoaded}
                      >
                        Drop Pin Here
                      </Button>
                      {liveMapCenter && (
                        <p className="text-xs text-muted-foreground">
                          Map center: {liveMapCenter.lat.toFixed(5)},
{liveMapCenter.lng.toFixed(5)} </p>
                      )}
                    </div>
                  )}
                </div>
              </div>
            </DrawerContent>
          </Drawer>
        ) : (
          // Desktop: Modal Overlay
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            onClick={handleCloseMapPicker}
            className="fixed inset-0 bg-background/90 backdrop-blur-sm z-[60] flex items-center
justify-center p-4"
          >
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              exit={{ opacity: 0 }}
              transition={{ duration: 0.2 }}
              onClick={(e) => e.stopPropagation()}
              className="w-full max-w-4xl bg-card rounded-2xl shadow-elevation-4 border-2 border-
border overflow-hidden"
              style={{ height: "80vh" }}
            >
              {/* Header */}
              <div className="p-4 border-b border-border">
                <div className="flex items-center justify-between mb-3">
                  <div>
                    <h3 className="text-xl font-bold">Select Location</h3>
                    <p className="text-sm text-muted-foreground">
                      Search, click, or use the center crosshair to drop a pin
                    </p>
                  </div>
                  <Button
                    variant="ghost"
                    size="icon"
                    onClick={handleCloseMapPicker}
                    className="h-8 w-8"
                  >
                    <CloseIcon style={{ fontSize: 20 }} />
                  </Button>
                </div>

                {/* Search Bar */}
                {isMapLoaded && (
                  <div className="relative">
                    <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2
text-muted-foreground" sty.<Autocomplete
                      onLoad={(autocomplete) => {
                        autocompleteRef.current = autocomplete;
                      }}
                      onPlaceChanged={handlePlaceSelect}
                      options={{
                        fields: ["geometry", "formatted_address", "name"],
```

```
                      }}
                    >
                      <Input
                        placeholder="Search for a place..."
                        className="pl-10 h-11"
                      />
                    </Autocomplete>
                  </div>
                )}
              </div>

              {/* Map */}
              <div className="relative" style={{ height: "calc(80vh - 180px)" }}>
                {mapLoadError ? (
                  <div className="w-full h-full flex items-center justify-center bg-muted">
                    <div className="text-center p-4">
                      <p className="text-destructive font-semibold">Error loading map</p>
                      <p className="text-sm text-muted-foreground">Please check your Google Maps API key</p>
                    </div>
                  </div>
                ) : !isMapLoaded ? (
                  <div className="w-full h-full flex items-center justify-center bg-muted">
                    <div className="text-center">
                      <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary mx-auto mb-4"></div> <p className="text-muted-foreground">Loading map...</p>
                    </div>
                  </div>
                ) : (
                  <GoogleMap
                    mapContainerStyle={{ width: "100%", height: "100%" }}
                    center={mapPickerCenter}
                    zoom={mapPickerZoom}
                    onLoad={(map) => {
                      setMapPickerRef(map);
                      if (
                        typeof window !== "undefined" &&
                        window.google?.maps?.Geocoder &&
                        !geocoderRef.current
                      ) {
                        geocoderRef.current = new window.google.maps.Geocoder();
                      }
                      const currentCenter = map.getCenter();
                      if (currentCenter) {
                        setLiveMapCenter({
                          lat: currentCenter.lat(),
                          lng: currentCenter.lng(),
                        });
                      }
                      if (userLocation) {
                        map.panTo({ lat: userLocation.lat, lng: userLocation.lng });
                        map.setZoom(15);
                      }
                    }}
                    onClick={handleMapClick}
                    onIdle={handleMapIdle}
                    options={{
                      disableDefaultUI: true,
                      zoomControl: true,
                      streetViewControl: false,
                      mapTypeControl: false,
                      fullscreenControl: false,
                      gestureHandling: "greedy",
                      draggableCursor: "crosshair",
                    }}
                  >
                    {/* User Location Marker */}
                    {userLocation && (
                      <Marker
                        position={{ lat: userLocation.lat, lng: userLocation.lng }}
                        icon={{
                          url: "https://maps.google.com/mapfiles/ms/icons/blue-dot.png",
```

```
                          scaledSize: isMapLoaded && window.google ? new
window.google.maps.Size(32, 32) : undefined,
                        }
                        title="Your location"
                      />
                    )}

                    {/* Selected Location Marker */}
                    {selectedLocation && (
                      <Marker
                        position={{ lat: selectedLocation.lat, lng: selectedLocation.lng }}
                        icon={{
                          url: "https://maps.google.com/mapfiles/ms/icons/red-dot.png",
                          scaledSize: isMapLoaded && window.google ? new
window.google.maps.Size(48, 48) : undefined,
                        }
                        title="Selected location"
                      />
                    )}
                  </GoogleMap>
                )}
                {isMapLoaded && !mapLoadError && (
                  <>
                    {/* Center crosshair */}
                    <div className="pointer-events-none absolute inset-0 flex items-center
justify-center z-10">
                      <div className="relative h-14 w-14 flex items-center justify-center">
                        <span className="absolute left-1/2 top-0 h-14 w-[2px] -translate-x-1/2
bg-primary/70"></span>
                        <span className="absolute top-1/2 left-0 w-14 h-[2px] -translate-y-1/2
bg-primary/70"></span>
                        <div className="w-6 h-6 rounded-full border-2 border-primary/80 bg-
background/60 backdrop-blur-s</div>
                      </div>
                    </div>

                    {/* Loading user location indicator */}
                    {isLoadingUserLocation && (
                      <div className="absolute top-4 left-1/2 -translate-x-1/2 z-20 bg-
background/90 backdrop-blur-sm px-
                        <div className="flex items-center gap-2">
                          <div
primary"></div>
                          <p>
                          <p>Getting your location...</
                        </div>
                      </div>
                    )}
                  </>
                )}
              </div>

              {/* Footer with Selected Location Info and Actions */}
              <div className="p-4 border-t border-border bg-card">
                {selectedLocation ? (
                  <div className="space-y-3">
                    <div>
                      <p className="text-sm font-semibold">Selected Location</p>
                      <p className="text-xs text-muted-foreground">
                        {selectedLocation.address || `${selectedLocation.lat.toFixed(6)},
${selectedLocation.lng.toFixed(6)}`}
                      </p>
                    </div>
                    <div className="flex gap-3">
                      <Button
                        variant="outline"
                        onClick={handleCloseMapPicker}
                        className="flex-1 h-11"
                      >
                        Cancel
                      </Button>
                      <Button
                        onClick={handleConfirmLocation}
                        className="flex-1 h-11"
                      >
                        Use This Location
                      </Button>
                    </div>
                  </div>
                ) : (
                  <div className="space-y-3 text-center">
```

```
                    <p className="text-sm text-muted-foreground">
                      Click anywhere on the map or align the crosshair, then drop the pin.
                    </p>
                    <Button
                      variant="secondary"
                      onClick={handleCenterPinDrop}
                      className="w-full sm:w-auto h-10"
                      disabled={!isMapLoaded}
                    >
                      Drop Pin Here
                    </Button>
                    {liveMapCenter && (
                      <p className="text-xs text-muted-foreground">
                        Map center: {liveMapCenter.lat.toFixed(5)},
{liveMapCenter.lng.toFixed(5)}
                      </p>
                    )}
                  </div>
                )}
              </div>
            </motion.div>
          )}
        </AnimatePresence>
      );
    };

    const submitButtonLabel = isEditMode
      ? isSubmitting
        ? "Saving..."
        : "Save Changes"
      : isSubmitting
      ? "Creating..."
      : "Create Event";

    const modalTitle = isEditMode ? "Edit Event" : "Create New Event";
    const modalDescription = isEditMode
      ? "Adjust the details for your existing event"
      : "Share your activity with the community";

    // Shared form content - extracted to reduce re-renders
    const renderFormContent = () => (
      <>
        {/* Event Title */}
        <div className="space-y-2">
          <Label htmlFor="title" className="text-sm sm:text-base font-semibold flex items-center gap-2">
            <div className="p-1.5 bg-primary/10 rounded-lg">
              <EventIcon style={{ fontSize: 18 }} className="text-primary" />
            </div>
            Event Title *
          </Label>
          <Input
            id="title"
            placeholder="e.g., Morning Run in Central Park"
            value={formData.title || ""}
            onChange={(e) => handleInputChange("title", e.target.value)}
            className={`h-12 sm:h-14 bg-gradient-to-br from-background to-muted/20 border-2 focus:border-primary focus:ring-
                errors.title ? "border-destructive" : "border-border/50"
            }`}
            maxLength={100}
          />
          {errors.title && (
            <p className="text-xs sm:text-sm text-destructive">{errors.title}</p>
          )}
        </div>

        {/* Description */}
        <div className="space-y-2">
          <Label htmlFor="description" className="text-sm sm:text-base font-semibold flex items-center gap-2">
            <div className="p-1.5 bg-success/10 rounded-lg">
              <DescriptionIcon style={{ fontSize: 18 }} className="text-success" />
            </div>
```

```
          Description *
        </Label>
        <Textarea
          id="description"
          placeholder="Describe your event, what to expect, any special requirements..."
          value={formData.description || ""}
          onChange={(e) => handleInputChange("description", e.target.value)}
          className={`min-h-[100px] sm:min-h-[120px] resize-none bg-gradient-to-br from-background to-muted/20 border-2 ${errors.description ? "border-destructive" : "border-border/50"}`}
          maxLength={500}
        />
        <div className="flex justify-between items-center">
          {errors.description ? (
            <p className="text-xs sm:text-sm text-destructive">{errors.description}</p>
          ) : (
            <p className="text-xs text-muted-foreground">
              {formData.description?.length || 0} / 500 characters
            </p>
          )}
        </div>
      </div>

      {/* Activity Type */}
      <div className="space-y-3">
        <Label className="text-sm sm:text-base font-semibold flex items-center gap-2">
          <FitnessCenterIcon style={{ fontSize: 18 }} className="text-primary" />
          Activity Type *
        </Label>
        <div className="grid grid-cols-2 sm:grid-cols-4 gap-3 sm:gap-4">
          {activities.map((activity) => {
            const Icon = activity.icon;
            const isSelected = formData.activityType === activity.id;
            return (
              <motion.button
                key={activity.id}
                type="button"
                whileHover={{ scale: 1.05, y: -2 }}
                whileTap={{ scale: 0.95 }}
                onClick={() => handleInputChange("activityType", activity.id)}
                className={`
                  relative flex flex-col items-center justify-center p-4 sm:p-5 rounded-2xl border-2 transition-all dura...
                  ${
                    isSelected
                      ? activity.color === "success"
                        ? "border-success bg-gradient-to-br from-success/20 to-success/5 shadow-lg shadow-success/20"
                        : activity.color === "primary"
                        ? "border-primary bg-gradient-to-br from-primary/20 to-primary/5 shadow-lg shadow-primary/20"
                        : activity.color === "warning"
                        ? "border-warning bg-gradient-to-br from-warning/20 to-warning/5 shadow-lg shadow-warning/20"
                        : "border-secondary bg-gradient-to-br from-secondary/20 to-secondary/5 shadow-lg shadow-secondar... : "border-border/50 bg-gradient-to-br from-card to-muted/30 hover:border-primary/50 hover:shadow-m...
                  }
                `}
              >
                {isSelected && (
                  <motion.div
                    initial={{ scale: 0 }}
                    animate={{ scale: 1 }}
                    className="absolute inset-0 bg-gradient-to-br from-white/10 to-transparent rounded-2xl"
                  />
                )}
                <div className="relative">
                  <Icon
                    className={
                      isSelected
                        ? activity.color === "success"
                          ? "text-success drop-shadow-sm"
                          : activity.color === "primary"
                          ? "text-primary drop-shadow-sm"
                          : activity.color === "warning"
```

```
                          ? "text-warning drop-shadow-sm"
                          : "text-secondary drop-shadow-sm"
                        : "text-muted-foreground"
                    }
                    style={{ fontSize: isMobile ? 28 : 36 }}
                  />
                </div>
                <span
                  className={`text-xs sm:text-sm mt-2 sm:mt-3 font-bold relative ${
                    isSelected
                      ? activity.color === "success"
                        ? "text-success"
                        : activity.color === "primary"
                        ? "text-primary"
                        : activity.color === "warning"
                        ? "text-warning"
                        : "text-secondary"
                      : "text-muted-foreground"
                  }`}
                >
                  {activity.label}
                </span>
                {isSelected && (
                  <motion.div
                    initial={{ scale: 0 }}
                    animate={{ scale: 1 }}
                    className="absolute top-2 right-2 w-5 h-5 bg-primary rounded-full flex items-center justify-center"
                  >
                    <span className="text-white text-xs">' </span>
                  </motion.div>
                )}
              </motion.button>
            );
          })}
        </div>
        {errors.activityType && (
          <p className="text-xs sm:text-sm text-destructive">{errors.activityType}</p>
        )}
      </div>

      {/* Date & Time */}
      <div className="grid grid-cols-1 sm:grid-cols-2 gap-4 sm:gap-5">
        <div className="space-y-2">
          <Label htmlFor="date" className="text-sm sm:text-base font-semibold flex items-center gap-2">
            <div className="p-1.5 bg-primary/10 rounded-lg">
              <EventIcon style={{ fontSize: 18 }} className="text-primary" />
            </div>
            Date *
          </Label>
          <Input
            id="date"
            type="date"
            min={getTodayDate()}
            value={formData.date || ""}
            onChange={(e) => handleInputChange("date", e.target.value)}
            className={`h-12 sm:h-14 bg-gradient-to-br from-background to-muted/20 border-2 focus:border-primary focus:r
            errors.date ? "border-destructive" : "border-border/50"
            }`}
          />
          {errors.date && (
            <p className="text-xs sm:text-sm text-destructive">{errors.date}</p>
          )}
        </div>

        <div className="space-y-2">
          <Label htmlFor="time" className="text-sm sm:text-base font-semibold flex items-center gap-2">
            <div className="p-1.5 bg-success/10 rounded-lg">
              <AccessTimeIcon style={{ fontSize: 18 }} className="text-success" />
            </div>
            Time *
          </Label>
```

```
1132 |           <Input
1133 |             id="time"
1134 |             type="time"
1135 |             value={formData.time || ""}
1136 |             onChange={(e) => handleInputChange("time", e.target.value)}
1137 |             className={`h-12 sm:h-14 bg-gradient-to-br from-background to-muted/20 border-2
focus:border-success ${errors.time
1138 |               ? "border-destructive" : "border-border/50"
1139 |             }`}
1140 |           />
1141 |           {errors.time && (
1142 |             <p className="text-xs sm:text-sm text-destructive">{errors.time}</p>
1143 |           )}
1144 |         </div>
1145 |       </div>
1146 |
1147 |       {/* Location */}
1148 |       <div className="space-y-2">
1149 |         <Label htmlFor="location" className="text-sm sm:text-base font-semibold flex items-
center gap-2">
1150 |           <div className="p-1.5 bg-warning/10 rounded-lg">
1151 |             <LocationOnIcon style={{ fontSize: 18 }} className="text-warning" />
1152 |           </div>
1153 |           Location {initialLocation ? "" : "*"}
1154 |         </Label>
1155 |         <div className="flex gap-3">
1156 |           <Input
1157 |             id="location"
1158 |             placeholder={initialLocation ? `Location: ${initialLocation.lat.toFixed(6)},
${initialLocation.lng.toFixed(6...
1159 |               value={
1160 |                 formData.location ||
1161 |                 (initialLocation
1162 |                   ? `Lat: ${initialLocation.lat.toFixed(6)}, Lng: ${initialLocation.lng.toFixed(6)}
`
1163 |                   : formData.lat && formData.lng
1164 |                   ? `Lat: ${formData.lat.toFixed(6)}, Lng: ${formData.lng.toFixed(6)}`
1165 |                   : "")
1166 |             }
1167 |             readOnly
1168 |             disabled
1169 |             className={`flex-1 h-12 sm:h-14 bg-gradient-to-br from-muted/30 to-muted/10 cursor-
not-allowed border-2 ${
1170 |               errors.location ? "border-destructive" : "border-border/50"
1171 |             }`}
1172 |             title="Use the pin button to pick a location"
1173 |             maxLength={200}
1174 |           />
1175 |           {!initialLocation && (
1176 |             <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
1177 |               <Button
1178 |                 type="button"
1179 |                 variant="outline"
1180 |                 onClick={handleOpenMapPicker}
1181 |                 className="h-12 sm:h-14 px-4 sm:px-6 border-2 border-warning text-warning bg-
warning/10 hover:bg-warning
1182 |                 title="Pin location on map"
1183 |               >
1184 |                 <MapIcon style={{ fontSize: 22 }} />
1185 |                 <span className="hidden sm:inline ml-2">Pin on Map</span>
1186 |               </Button>
1187 |             </motion.div>
1188 |           )}
1189 |         </div>
1190 |         {initialLocation && (
1191 |           <p className="text-xs text-muted-foreground">
1192 |             Ø=ÜÍ Location selected from map: {initialLocation.lat.toFixed(6)},
{initialLocation.lng.toFixed(6)}
1193 |           </p>
1194 |         )}
1195 |         {!initialLocation && formData.lat && formData.lng && (
1196 |           <p className="text-xs text-muted-foreground">
1197 |             Ø=ÜÍ Location coordinates: {formData.lat.toFixed(6)}, {formData.lng.toFixed(6)}
1198 |           </p>
1199 |         )}
1200 |         {!initialLocation && !formData.lat && (
1201 |           <p className="text-xs text-muted-foreground">
1202 |             Click "Pin on Map" to select a location, or enter a location name
```

```
          </p>
        )}
        {errors.location && (
          <p className="text-xs sm:text-sm text-destructive">{errors.location}</p>
        )}
      </div>

      {/* Max Participants */}
      <div className="space-y-2">
        <Label htmlFor="maxParticipants" className="text-sm sm:text-base font-semibold flex items-center gap-2">
          <div className="p-1.5 bg-secondary/10 rounded-lg">
            <PeopleIcon style={{ fontSize: 18 }} className="text-secondary" />
          </div>
          Maximum Participants (Optional)
        </Label>
        <Input
          id="maxParticipants"
          type="number"
          placeholder="Leave empty for unlimited"
          min="2"
          max="1000"
          value={formData.maxParticipants || ""}
          onChange={(e) =>
            handleInputChange(
              "maxParticipants",
              e.target.value ? parseInt(e.target.value) : ""
            )
          }
          className={`h-12 sm:h-14 bg-gradient-to-br from-background to-muted/20 border-2 focus:border-secondary focus:r
            errors.maxParticipants ? "border-destructive" : "border-border/50"
          }`}
        />
        {errors.maxParticipants && (
          <p className="text-xs sm:text-sm text-destructive">{errors.maxParticipants}</p>
        )}
        <p className="text-xs text-muted-foreground">
          Set a limit on how many people can join this event
        </p>
      </div>

      {/* Action Buttons */}
      <div className="flex flex-col sm:flex-row gap-4 pt-6 border-t border-border/50">
        <Button
          type="button"
          variant="outline"
          onClick={onClose}
          disabled={isSubmitting}
          className="w-full sm:flex-1 h-12 sm:h-14 border-2 hover:bg-destructive/10 hover:border-destructive/50 transiti
        >
          Cancel
        </Button>
        <motion.div
          whileHover={{ scale: 1.02 }}
          whileTap={{ scale: 0.98 }}
          className="w-full sm:flex-1"
        >
          <Button
            type="submit"
            disabled={isSubmitting}
            className="w-full h-12 sm:h-14 font-bold text-base bg-gradient-to-r from-primary via-primary to-success hove
          >
            {isSubmitting ? (
              <span className="flex items-center gap-2">
                <div className="w-5 h-5 border-2 border-white border-t-transparent rounded-full animate-spin" />
                Creating...
              </span>
            ) : (
              <span className="flex items-center gap-2">
                <EventIcon style={{ fontSize: 20 }} />
                {isEditMode ? "Update Event" : "Create Event"}
              </span>
```

```
            )}
          </Button>
        </motion.div>
      </div>
    </>
  );

  // Mobile: Use Drawer (Bottom Sheet)
  if (isMobile) {
    return (
      <>
        {renderMapPicker()}
        {!showMapPicker && (
          <Drawer open={true} onOpenChange={(open) => !open && onClose()}>
            <DrawerContent className="max-h-[90vh]">
              <DrawerHeader className="border-b border-border pb-4">
                <div className="flex items-center justify-between">
                  <div className="flex items-center gap-3">
                    <div className="p-2 bg-primary/10 rounded-lg">
                      <EventIcon className="text-primary" style={{ fontSize: 24 }} />
                    </div>
                    <div>
                      <DrawerTitle className="text-xl">{modalTitle}</DrawerTitle>
                      <DrawerDescription className="text-xs">
                        {modalDescription}
                      </DrawerDescription>
                    </div>
                  </div>
                  <Button
                    variant="ghost"
                    size="icon"
                    onClick={onClose}
                    className="h-8 w-8"
                  >
                    <CloseIcon style={{ fontSize: 20 }} />
                  </Button>
                </div>
              </DrawerHeader>
              <form onSubmit={handleSubmit} className="p-4 space-y-4 overflow-y-auto flex-1">
                {renderFormContent()}
              </form>
            </DrawerContent>
          </Drawer>
        )}
      </>
    );
  }

  // Desktop: Use Modal
  return (
    <>
      {renderMapPicker()}
      <AnimatePresence>
        {!showMapPicker && (
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            onClick={onClose}
            className="fixed inset-0 bg-background/80 backdrop-blur-sm z-50 flex items-center justify-center p-4 overflo...
          >
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              exit={{ opacity: 0 }}
              transition={{ duration: 0.2 }}
              onClick={(e) => e.stopPropagation()}
              className="w-full max-w-2xl my-8"
            >
              <Card className="overflow-hidden shadow-elevation-4 border-2 border-border/50 bg-gradient-to-br from-backg...
                {/* Header with Enhanced Design */}
```

```
1345 |                 <div className="relative bg-gradient-to-br from-primary via-primary/90 to-
success/80 p-8 border-b border
1346 |                   {/* Decorative background elements */}
1347 |                   <div className="absolute inset-0 opacity-10">
1348 |                     <div className="absolute top-0 right-0 w-64 h-64 bg-white rounded-full
blur-3xl" />
1349 |                     <div className="absolute bottom-0 left-0 w-48 h-48 bg-success rounded-full
blur-2xl" />
1350 |                   </div>
1351 |
1352 |                   <button
1353 |                     onClick={onClose}
1354 |                     className="absolute top-4 right-4 p-2 bg-white/20 backdrop-blur-md hover:bg-
white/30 rounded-full tr...>.
1355 |
1356 |                     <CloseIcon className="text-white" fontSize="small" />
1357 |                   </button>
1358 |
1359 |                   <div className="relative flex items-center gap-4">
1360 |                     <div className="relative">
1361 |                       <div className="absolute inset-0 bg-white/20 rounded-2xl blur-xl" />
1362 |                       <div className="relative p-4 bg-white/10 backdrop-blur-md rounded-2xl
border border-white/20 shado...<EventIcon className="text-white" style={{ fontSize: 32 }} />
1363 |
1364 |                       </div>
1365 |                     </div>
1366 |                     <div className="flex-1">
1367 |                       <h2 className="text-3xl font-bold text-white drop-shadow-lg">{modalTitle}</
h2>
1368 |                       <p className="text-sm text-white/90 mt-1.5 drop-shadow-md">
1369 |                         {modalDescription}
1370 |                       </p>
1371 |                     </div>
1372 |                   </div>
1373 |                 </div>
1374 |
1375 |                 {/* Form with Enhanced Styling */}
1376 |                 <form onSubmit={handleSubmit} className="p-6 sm:p-8 space-y-6 bg-gradient-to-b
from-background to-backgr...{renderFormContent()}
1377 |
1378 |                 </form>
1379 |               </Card>
1380 |             </motion.div>
1381 |           </motion.div>
1382 |         )}
1383 |       </AnimatePresence>
1384 |     </>
1385 |   );
1386 | };
1387 |
```

File: src/components/EventDetailModal.tsx

Lines: 536

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import { Avatar } from "@mui/material";
import CloseIcon from "@mui/icons-material/Close";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import EventIcon from "@mui/icons-material/Event";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import AccessTimeIcon from "@mui/icons-material/AccessTime";
import StarIcon from "@mui/icons-material/Star";
import SendIcon from "@mui/icons-material/Send";
import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
import EditIcon from "@mui/icons-material/Edit";
import DeleteIcon from "@mui/icons-material/Delete";
import NavigationIcon from "@mui/icons-material/Navigation";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import PeopleIcon from "@mui/icons-material/People";
import { toast } from "sonner";
import { deleteEvent, addComment, listenToComments, type EventComment } from "@/services/
eventService";
import { useAuth } from "@/hooks/useAuth";
import { openGoogleMapsNavigation } from "@/utils/navigation";

type EventType = "running" | "cycling" | "walking" | "others";

interface Event {
  id: string | number;
  title: string;
  description: string;
  type: EventType;
  category: "user" | "sponsored";
  date: string;
  time: string;
  location: string;
  distance: string;
  distanceValue: number;
  participants: string[] | number;
  maxParticipants?: number;
  hostId?: string;
  hostName?: string;
  hostAvatar?: string;
  sponsorLogo?: string;
  lat: number;
  lng: number;
  isJoined?: boolean;
}


interface EventDetailModalProps {
  event: Event | null;
```

```
  67 |   onClose: () => void;
  68 |   onJoin: (eventId: string | number) => void;
  69 |   onEdit?: (eventId: string | number) => void;
  70 |   onDelete?: (eventId: string | number) => void;
  71 | }
  72 |
  73 | export const EventDetailModal = ({ event, onClose, onJoin, onEdit, onDelete }:
EventDetailModalProps) => {
  74 |   const { user } = useAuth();
  75 |   const [activeTab, setActiveTab] = useState("details");
  76 |   const [commentText, setCommentText] = useState("");
  77 |   const [comments, setComments] = useState<EventComment[]>([]);
  78 |   const [showDeleteDialog, setShowDeleteDialog] = useState(false);
  79 |   const [isAddingComment, setIsAddingComment] = useState(false);
  80 |
  81 |   const participantsCount = Array.isArray(event?.participants)
  82 |     ? event.participants.length
  83 |     : (event?.participants || 0);
  84 |
  85 |   // Check if current user has joined the event
  86 |   const isUserJoined = event && user?.uid && Array.isArray(event.participants) &&
  87 | event.participants.includes(user.uid);
  88 |   // Listen to comments from Firebase
  89 |   useEffect(() => {
  90 |     if (!event?.id) {
  91 |       setComments([]);
  92 |       return;
  93 |     }
  94 |
  95 |     const unsubscribe = listenToComments(event.id, (fetchedComments) => {
  96 |       setComments(fetchedComments);
  97 |     });
  98 |
  99 |     return () => {
 100 |       unsubscribe();
 101 |     };
 102 |   }, [event?.id]);
 103 |
 104 |   if (!event) return null;
 105 |
 106 |   const getActivityIcon = () => {
 107 |     switch (event.type) {
 108 |       case "running":
 109 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 24 }} />;
 110 |       case "cycling":
 111 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 24 }} />;
 112 |       case "walking":
 113 |         return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 24 }} />;
 114 |       case "others":
 115 |       default:
 116 |         return <FitnessCenterIcon className="text-secondary" style={{ fontSize: 24 }} />;
 117 |     }
 118 |   };
 119 |
 120 |   const handleAddComment = async () => {
 121 |     if (!commentText.trim() || !user?.uid || !event?.id) {
 122 |       return;
 123 |     }
 124 |
 125 |     // Check if user has joined the event
 126 |     if (!isUserJoined) {
 127 |       toast.error("You must join the event to comment");
 128 |       return;
 129 |     }
 130 |
 131 |     setIsAddingComment(true);
 132 |     try {
 133 |       await addComment(
 134 |         event.id,
 135 |         user.uid,
 136 |         user.displayName || user.email?.split('@')[0] || "User",
 137 |         user.photoURL || `https://ui-avatars.com/api/?name=${encodeURIComponent(user.displayName
|| user.email || 'User'...
```

```
138 |         commentText.trim()
139 |       );
140 |       setCommentText("");
141 |       toast.success("Comment added!");
142 |     } catch (error: any) {
143 |       console.error("Error adding comment:", error);
144 |       toast.error(error.message || "Failed to add comment");
145 |     } finally {
146 |       setIsAddingComment(false);
147 |     }
148 |   };
149 |
150 |   const handleDeleteClick = () => {
151 |     if (!event || !user?.uid) return;
152 |
153 |     if (event.hostId !== user.uid) {
154 |       toast.error("Only the event creator can delete this event");
155 |       return;
156 |     }
157 |
158 |     setShowDeleteDialog(true);
159 |   };
160 |
161 |   const handleDeleteConfirm = async () => {
162 |     if (!event || !user?.uid) return;
163 |
164 |     try {
165 |       await deleteEvent(String(event.id), user.uid);
166 |       toast.success("Event deleted successfully");
167 |       setShowDeleteDialog(false);
168 |       onClose();
169 |       if (onDelete) {
170 |         onDelete(event.id);
171 |       }
172 |     } catch (error: any) {
173 |       toast.error(error.message || "Failed to delete event");
174 |       setShowDeleteDialog(false);
175 |     }
176 |   };
177 |
178 |   const handleEdit = () => {
179 |     if (!event || !user?.uid) return;
180 |
181 |     if (event.hostId !== user.uid) {
182 |       toast.error("Only the event creator can edit this event");
183 |       return;
184 |     }
185 |
186 |     if (onEdit) {
187 |       onEdit(event.id);
188 |     }
189 |   };
190 |
191 |   // Check if current user is the event creator
192 |   const isEventCreator = event && user?.uid && event.hostId === user.uid;
193 |
194 |
195 |   return (
196 |     <AnimatePresence>
197 |       <motion.div
198 |         initial={{ opacity: 0 }}
199 |         animate={{ opacity: 1 }}
200 |         exit={{ opacity: 0 }}
201 |         onClick={onClose}
202 |         className="fixed inset-0 bg-background/80 backdrop-blur-sm z-[80] flex items-center
justify-center p-0 md:p-4"
203 |       >
204 |         <motion.div
205 |           initial={{ scale: 0.95, opacity: 0, y: 20 }}
206 |           animate={{ scale: 1, opacity: 1, y: 0 }}
207 |           exit={{ scale: 0.95, opacity: 0, y: 20 }}
208 |           transition={{ type: "spring", stiffness: 300, damping: 30 }}
```

```
          onClick={(e) => e.stopPropagation()}
          className="w-full h-full md:h-auto md:max-w-3xl my-0 md:my-8 flex"
        >
          <Card className="flex flex-col h-full md:h-auto overflow-hidden shadow-elevation-4
border-2 border-border/50 w...
            {/* Enhanced Header with Gradient Background */}
            <div className="relative bg-gradient-to-r from-primary via-primary to-success p-6
pb-6 border-b border-prime...
              <button
                onClick={onClose}
                aria-label="Close event details"
                className="absolute top-4 right-4 p-2 bg-white/20 backdrop-blur-sm hover:bg-
white/30 rounded-full transi...
              >
                <CloseIcon style={{ fontSize: 20 }} />
              </button>

              {/* Distance - Top Right Corner */}
              {event.distanceValue !== Infinity && (
                <div>
                  <p>
                    {event.distance} away
                  </p>
                </div>
              )}

              <div>
                {/* Title & Type */}
                <div>
                  <div>
                    {event.type === "running" && <DirectionsRunIcon />}
                    {event.type === "cycling" && <DirectionsBikeIcon />}
                    {event.type === "walking" && <DirectionsWalkIcon />}
                    {event.type === "others" && <FitnessCenterIcon />}
                  </div>
                  <div>
                    <div>
                      <h2>{event.title}</h2>
                      {event.category === "sponsored" && (
                        <Badge>
                          <StarIcon />
                          Sponsored
                        </Badge>
                      )}
                    </div>
                    <p>{event.description}</p>
                  </div>
                </div>

                {/* Date & Time Cards */}
                <div>
                  <div>
                    <EventIcon />
                    <div>
                      <p>Date</p>
                      <p>{event.date}</p>
                    </div>
                  </div>
                  <div>
                    <AccessTimeIcon />
                    <div>
                      <p>Time</p>
                      <p>{event.time}</p>
                    </div>
                  </div>
                </div>

                {/* Action Buttons */}
                <div>
                  {/* Creator Actions - Edit/Delete */}
                  {isEventCreator && (
                    <div>
                      <Button
                        onClick={handleEdit}
```

```
                        variant="outline"
                        className="h-12 font-semibold bg-white/10 border-white/30 text-white
hover:bg-white/20 backdrop-...>
                        <EditIcon className="mr-2" style={{ fontSize: 20 }} />
                        Edit Event
                      </Button>
                      <Button
                        onClick={handleDeleteClick}
                        variant="outline"
                        className="h-12 font-semibold bg-white/10 border-white/30 text-white
hover:bg-red-500/20 hover:b...>
                        <DeleteIcon className="mr-2" style={{ fontSize: 20 }} />
                        Delete Event
                      </Button>
                    </div>
                  )}

                  {/* Join/Leave Button - Only show if user is not the event owner */}
                  {!isEventCreator && (
                    <Button
                      onClick={() => onJoin(event.id)}
                      className={`w-full h-12 font-semibold transition-all ${
                        event.isJoined
                          ? "bg-destructive text-destructive-foreground hover:bg-destructive/90
border-2 border-destruct...
                          : "bg-white/20 backdrop-blur-md text-white border-2 border-white/30
hover:bg-white/30"
                      }`}
                    >
                      {event.isJoined ? (
                        <>
                          <DeleteIcon className="mr-2" style={{ fontSize: 20 }} />
                          Leave Event
                        </>
                      ) : (
                        <>
                          <PeopleIcon className="mr-2" style={{ fontSize: 20 }} />
                          Join Event
                        </>
                      )}
                    </Button>
                  )}
                </div>
              </div>
            </div>

            {/* Tabs Content */}
            <div className="flex-1 flex flex-col overflow-hidden">
              <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full flex-1 flex
flex-col overflow-hidde<TabsList className="w-full grid grid-cols-2 rounded-none border-b border-border
h-auto bg-muted/30">     <TabsTrigger
                    value="details"
                    className="py-3 data-[state=active]:border-b-2 data-[state=active]:border-
primary data-[state=active...
                    <EventIcon className="mr-2" style={{ fontSize: 18 }} />
                    Details
                  </TabsTrigger>
                  <TabsTrigger
                    value="comments"
                    className="py-3 data-[state=active]:border-b-2 data-[state=active]:border-
primary data-[state=active...
                    <ChatBubbleOutlineIcon className="mr-2" style={{ fontSize: 18 }} />
                    Comments ({comments.length})
                  </TabsTrigger>
                </TabsList>

                <div className="flex-1 overflow-y-auto">
                  {/* Details Tab */}
                  <TabsContent value="details" className="p-6 space-y-6">
                    {/* Location */}
                    <div className="space-y-3">
                      <h3 className="text-lg font-bold flex items-center gap-2">
                        <LocationOnIcon style={{ fontSize: 20 }} className="text-primary" />
```

```
                        Location
                      </h3>
                      <div className="bg-gradient-to-br from-muted/50 via-muted/30 to-muted/50
rounded-xl p-4 border-2 b...
                        <p className="font-semibold text-foreground mb-2">{event.location}</p>
                        <p className="text-sm text-muted-foreground mb-3">
                          {event.distance} from your location
                        </p>
                        <Button
                          variant="outline"
                          size="sm"
                          className="w-full sm:w-auto border-2 hover:bg-primary/10 hover:border-
primary/50"
                          onClick={() => {
                            if (event.lat && event.lng) {
                              openGoogleMapsNavigation(event.lat, event.lng);
                            } else {
                              toast.error("Location not available");
                            }
                          }}
                        >
                          <NavigationIcon className="mr-2" style={{ fontSize: 16 }} />
                          View on Map
                        </Button>
                      </div>
                    </div>

                    {/* Event Info */}
                    <div className="space-y-3">
                      <h3 className="text-lg font-bold flex items-center gap-2">
                        <EventIcon style={{ fontSize: 20 }} className="text-primary" />
                        Event Information
                      </h3>
                      <div className="grid grid-cols-2 gap-3">
                        <div className="bg-gradient-to-br from-muted/50 via-muted/30 to-muted/50
rounded-xl p-4 border-2...
                          <p className="text-xs text-muted-foreground mb-1">Activity Type</p>
                          <p className="font-semibold capitalize text-foreground">{event.type}</
p>
                        </div>
                        <div className="bg-gradient-to-br from-muted/50 via-muted/30 to-muted/50
rounded-xl p-4 border-2...
                          <p className="text-xs text-muted-foreground mb-1">Participants</p>
                          <p className="font-semibold text-foreground">
                            {participantsCount}
                            {event.maxParticipants ? ` / ${event.maxParticipants}` : ""}
                          </p>
                        </div>
                      </div>
                    </div>

                    {/* Host/Sponsor */}
                    {event.category === "user" && event.hostName ? (
                      <div className="space-y-3">
                        <h3 className="text-lg font-bold">Hosted By</h3>
                        <div className="flex items-center gap-3 bg-gradient-to-br from-muted/50
via-muted/30 to-muted/50...
                          <Avatar src={event.hostAvatar} alt={event.hostName} sx={{ width: 48,
height: 48 }} />
                          <div className="flex-1">
                            <p className="font-semibold text-foreground">{event.hostName}</p>
                            <p className="text-sm text-muted-foreground">Event Organizer</p>
                          </div>
                        </div>
                      </div>
                    ) : event.sponsorLogo ? (
                      <div className="space-y-3">
                        <h3 className="text-lg font-bold">Official Sponsor</h3>
                        <div className="flex items-center gap-3 bg-gradient-to-br from-muted/50
via-muted/30 to-muted/50...
                          <img src={event.sponsorLogo} alt="Sponsor" className="w-12 h-12
rounded" />
                          <div className="flex-1">
                            <p className="font-semibold text-foreground">Sponsored Event</p>
                            <p className="text-sm text-muted-foreground">Official Brand
Partnership</p>
                          </div>
                        </div>
                      </div>
                    ) : null}
                  </TabsContent>
```

```
                  {/* Comments Tab */}
                  <TabsContent value="comments" className="p-6 space-y-4">
                    {!isUserJoined ? (
                      <div className="text-center py-12 bg-muted/50 rounded-lg border border-
border">
                        <PeopleIcon style={{ fontSize: 48 }} className="text-muted-foreground mx-
auto mb-3" />
                        <p className="text-base font-semibold text-foreground">Join to see
comments</p>
                        <p className="text-sm text-muted-foreground mt-1">You must join this
event to view and add comme...</div>
                    ) : (
                      <>
                        {/* Comments List */}
                        <div className="space-y-4 max-h-[400px] overflow-y-auto pr-2">
                          {comments.length === 0 ? (
                            <div className="text-center py-8 bg-muted/50 rounded-lg border
border-border">
                              <ChatBubbleOutlineIcon style={{ fontSize: 48 }} className="text-
muted-foreground mx-auto m...
                              <p className="text-sm text-muted-foreground">No comments yet</p>
                              <p className="text-xs text-muted-foreground mt-1">Be the first to
comment!</p>
                            </div>
                          ) : (
                            comments.map((comment, index) => (
                              <motion.div
                                key={comment.id}
                                initial={{ opacity: 0, y: 10 }}
                                animate={{ opacity: 1, y: 0 }}
                                transition={{ delay: index * 0.05 }}
                                className="flex gap-3"
                              >
                                <Avatar
                                  src={comment.userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(comm...
                                  alt={comment.userName}
                                  sx={{ width: 40, height: 40 }}
                                />
                                <div className="flex-1 bg-gradient-to-br from-muted/50 via-
muted/30 to-muted/50 rounded-...
                                  <div className="flex items-center justify-between mb-1">
                                    <p className="font-semibold text-sm text-
foreground">{comment.userName}</p>
                                    <p className="text-xs text-muted-foreground">
                                      {comment.timestamp ? new
Date(comment.timestamp).toLocaleDateString() : "Just now"...
                                    </p>
                                  </div>
                                  <p className="text-sm text-foreground">{comment.text}</p>
                                </div>
                              </motion.div>
                            ))
                          )}
                        </div>

                        {/* Add Comment */}
                        <div className="flex gap-2 pt-4 border-t border-border">
                          <Avatar
                            src={user?.photoURL || `https://ui-avatars.com/api/?
name=${encodeURIComponent(user?.displayN...
                            alt={user?.displayName || "You"}
                            sx={{ width: 40, height: 40 }}
                          />
                          <div className="flex-1 flex gap-2">
                            <Input
                              value={commentText}
                              onChange={(e) => setCommentText(e.target.value)}
                              placeholder="Add a comment..."
                              className="flex-1 bg-muted/50 border-2"
                              disabled={isAddingComment}
                              onKeyPress={(e) => {
                                if (e.key === "Enter" && !e.shiftKey && !isAddingComment) {
                                  e.preventDefault();
                                  handleAddComment();
                                }
                              }}
                            />
                            <Button
                              onClick={handleAddComment}
                              disabled={!commentText.trim() || isAddingComment}
```

```
493 |                               className="bg-gradient-to-r from-primary via-primary to-success
hover:from-primary/90 hove...      >
494 |
495 |                               {isAddingComment ? (
496 |                                 <div className="w-5 h-5 border-2 border-white border-t-
transparent rounded-full animate-... ) : (
497 |
498 |                                 <SendIcon style={{ fontSize: 20 }} />
499 |                               )}
500 |                             </Button>
501 |                           </div>
502 |                         </div>
503 |                       </>
504 |                     )}
505 |                   </TabsContent>
506 |                 </div>
507 |               </Tabs>
508 |             </div>
509 |           </Card>
510 |         </motion.div>
511 |       </motion.div>
512 |
513 |       {/* Delete Confirmation Dialog */}
514 |       <AlertDialog open={showDeleteDialog} onOpenChange={setShowDeleteDialog}>
515 |         <AlertDialogContent>
516 |           <AlertDialogHeader>
517 |             <AlertDialogTitle>Delete Event</AlertDialogTitle>
518 |             <AlertDialogDescription>
519 |               Are you sure you want to delete "{event.title}"? This action cannot be undone and
will remove the event fo...
520 |             </AlertDialogDescription>
521 |           </AlertDialogHeader>
522 |           <AlertDialogFooter>
523 |             <AlertDialogCancel>Cancel</AlertDialogCancel>
524 |             <AlertDialogAction
525 |               onClick={handleDeleteConfirm}
526 |               className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
527 |             >
528 |               Delete Event
529 |             </AlertDialogAction>
530 |           </AlertDialogFooter>
531 |         </AlertDialogContent>
532 |       </AlertDialog>
533 |     </AnimatePresence>
534 |   );
535 | };
536 |
```

File: src/components/EventDetailsPanel.tsx

Lines: 608

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import { Avatar } from "@mui/material";
import { useAuth } from "@/hooks/useAuth";
import { getUserData } from "@/services/authService";
import { deleteEvent, addComment, listenToComments, deleteComment, type EventComment } from "@/services/eventService";
import { openGoogleMapsNavigation } from "@/utils/navigation";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import EventIcon from "@mui/icons-material/Event";
import PeopleIcon from "@mui/icons-material/People";
import AccessTimeIcon from "@mui/icons-material/AccessTime";
import SendIcon from "@mui/icons-material/Send";
import CloseIcon from "@mui/icons-material/Close";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import EditIcon from "@mui/icons-material/Edit";
import DeleteIcon from "@mui/icons-material/Delete";
import NavigationIcon from "@mui/icons-material/Navigation";
import { toast } from "sonner";

type EventType = "running" | "cycling" | "walking" | "others";

interface Event {
  id: string;
  title: string;
  description: string;
  type: EventType;
  category: "user" | "sponsored";
  date: string;
  time: string;
  location: string;
  distance: string;
  distanceValue: number;
  participants: string[];
  maxParticipants?: number;
  hostId?: string;
  hostName?: string;
  hostAvatar?: string;
  lat: number;
  lng: number;
  isJoined?: boolean;
}

interface ParticipantData {
  id: string;
  name: string;
  username?: string;
  avatar: string;
  activity?: string;
  activities?: string[];
}

// Using EventComment from eventService for type consistency

interface EventDetailsPanelProps {
  event: Event | null;
  onClose: () => void;
  onJoin: (eventId: string) => void;
  onLeave: (eventId: string) => void;
  onEdit?: (eventId: string) => void;
  onDelete?: (eventId: string) => void;
```

```
  67 |   onShowRoute?: (lat: number, lng: number) => void;
  68 | }
  69 |
  70 | export const EventDetailsPanel = ({ event, onClose, onJoin, onLeave, onEdit, onDelete,
onShowRoute }: EventDetailsPanelP...
  71 |   const { user } = useAuth();
  72 |   const [commentText, setCommentText] = useState("");
  73 |   const [comments, setComments] = useState<EventComment[]>([]);
  74 |   const [participantsData, setParticipantsData] = useState<ParticipantData[]>([]);
  75 |   const [loadingParticipants, setLoadingParticipants] = useState(false);
  76 |   const [isAddingComment, setIsAddingComment] = useState(false);
  77 |   const [deletingCommentId, setDeletingCommentId] = useState<string | null>(null);
  78 |
  79 |   // Check if current user has joined the event
  80 |   const isUserJoined = event && user?.uid && Array.isArray(event.participants) &&
event.participants.includes(user.uid);
  81 |
  82 |   // Listen to comments from Firebase
  83 |   useEffect(() => {
  84 |     if (!event?.id) {
  85 |       setComments([]);
  86 |       return;
  87 |     }
  88 |
  89 |     const unsubscribe = listenToComments(event.id, (fetchedComments) => {
  90 |       setComments(fetchedComments);
  91 |     });
  92 |
  93 |     return () => {
  94 |       unsubscribe();
  95 |     };
  96 |   }, [event?.id]);
  97 |
  98 |   // Fetch participant profiles when event changes
  99 |   useEffect(() => {
 100 |     if (!event?.participants || !Array.isArray(event.participants)) {
 101 |       setParticipantsData([]);
 102 |       return;
 103 |     }
 104 |
 105 |     const fetchParticipants = async () => {
 106 |       setLoadingParticipants(true);
 107 |       try {
 108 |         const participantPromises = event.participants.map(async (userId: string) => {
 109 |           try {
 110 |             const userData = await getUserData(userId);
 111 |             if (userData) {
 112 |               return {
 113 |                 id: userId,
 114 |                 name: userData.name || userData.username || "User",
 115 |                 username: userData.username,
 116 |                 avatar: userData.photoURL || `https://ui-avatars.com/api/?name=${userData.name
|| userData.username || 'U...
 117 |                 activity: userData.activity,
 118 |                 activities: userData.activities || (userData.activity ? [userData.activity] :
[])
 119 |               };
 120 |             }
 121 |             return null;
 122 |           } catch (error) {
 123 |             console.error(`Error fetching user ${userId}:`, error);
 124 |             return null;
 125 |           }
 126 |         });
 127 |
 128 |         const results = await Promise.all(participantPromises);
 129 |         setParticipantsData(results.filter((p): p is ParticipantData => p !== null));
 130 |       } catch (error) {
 131 |         console.error("Error fetching participants:", error);
 132 |       } finally {
 133 |         setLoadingParticipants(false);
 134 |       }
 135 |     };
 136 |
 137 |     fetchParticipants();
```

```
138 |   }, [event?.participants]);
139 |
140 |   const participantsCount = Array.isArray(event?.participants)
141 |     ? event.participants.length
142 |     : 0;
143 |
144 |   const getActivityIcon = (type: EventType) => {
145 |     switch (type) {
146 |       case "running":
147 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 24 }} />;
148 |       case "cycling":
149 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 24 }} />;
150 |       case "walking":
151 |         return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 24 }} />;
152 |       case "others":
153 |         return <FitnessCenterIcon className="text-secondary" style={{ fontSize: 24 }} />;
154 |     }
155 |   };
156 |
157 |   const getSmallActivityIcon = (activity: string) => {
158 |     switch (activity) {
159 |       case "running":
160 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 14 }} />;
161 |       case "cycling":
162 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 14 }} />;
163 |       case "walking":
164 |         return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 14 }} />;
165 |       default:
166 |         return null;
167 |     }
168 |   };
169 |
170 |   const handleAddComment = async () => {
171 |     if (!commentText.trim() || !event || !user?.uid) return;
172 |
173 |     // Check if user has joined the event
174 |     if (!isUserJoined && !event.isJoined) {
175 |       toast.error("Please join the event to add comments");
176 |       return;
177 |     }
178 |
179 |     setIsAddingComment(true);
180 |     try {
181 |       await addComment(
182 |         event.id,
183 |         user.uid,
184 |         user.displayName || "User",
185 |         user.photoURL || "",
186 |         commentText.trim()
187 |       );
188 |       setCommentText("");
189 |       toast.success("Comment added!");
190 |     } catch (error: any) {
191 |       toast.error(error.message || "Failed to add comment");
192 |     } finally {
193 |       setIsAddingComment(false);
194 |     }
195 |   };
196 |
197 |   const handleDeleteComment = async (commentId: string) => {
198 |     if (!event || !user?.uid) return;
199 |
200 |     setDeletingCommentId(commentId);
201 |     try {
202 |       await deleteComment(event.id, commentId, user.uid);
203 |       toast.success("Comment deleted");
204 |     } catch (error: any) {
205 |       toast.error(error.message || "Failed to delete comment");
206 |     } finally {
207 |       setDeletingCommentId(null);
208 |     }
```

```
  };

  // Check if user can delete a comment (author or event host)
  const canDeleteComment = (comment: EventComment) => {
    if (!user?.uid || !event) return false;
    return comment.userId === user.uid || event.hostId === user.uid;
  };

  const handleDelete = async () => {
    if (!event || !user?.uid) return;

    if (event.hostId !== user.uid) {
      toast.error("Only the event creator can delete this event");
      return;
    }

    if (!window.confirm("Are you sure you want to delete this event? This action cannot be undone.")) { return;
    }

    try {
      await deleteEvent(event.id, user.uid);
      toast.success("Event deleted successfully");
      onClose();
      if (onDelete) {
        onDelete(event.id);
      }
    } catch (error: any) {
      toast.error(error.message || "Failed to delete event");
    }
  };

  const handleEdit = () => {
    if (!event || !user?.uid) return;

    if (event.hostId !== user.uid) {
      toast.error("Only the event creator can edit this event");
      return;
    }

    if (onEdit) {
      onEdit(event.id);
    }
  };

  // Check if current user is the event creator
  const isEventCreator = event && user?.uid && event.hostId === user.uid;


  return (
    <AnimatePresence>
      {event ? (
        <>
          {/* Backdrop - Click outside to close */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            onClick={onClose}
            className="fixed inset-0 bg-background/60 backdrop-blur-sm z-30"
          />

          {/* Panel */}
          <motion.div
            initial={{ y: "100%" }}
            animate={{ y: 0 }}
            exit={{ y: "100%" }}
            transition={{ type: "spring", stiffness: 300, damping: 30 }}
            onClick={(e) => e.stopPropagation()}
            className="fixed left-0 right-0 z-40 bg-card border-t border-border rounded-t-3xl shadow-elevation-4 flex fl
            style={{..
```

```
              bottom: `calc(4.5rem + env(safe-area-inset-bottom, 0px))`,
              maxHeight: "calc(100vh - 4.5rem - env(safe-area-inset-bottom, 0px))",
              height: "calc(100vh - 4.5rem - env(safe-area-inset-bottom, 0px))"
            }}
          >
          {/* Enhanced Header with Gradient Background */}
          <div className="relative bg-gradient-to-r from-primary via-primary to-success p-4 pb-6
rounded-t-3xl flex-shri
            {/* Drag Handle */}
            <div className="flex justify-center mb-3">
              <div className="w-12 h-1.5 bg-white/30 rounded-full" />
            </div>

            {/* Close Button - Top Right */}
            <button
              onClick={onClose}
              className="absolute top-4 right-4 p-2 bg-white/20 backdrop-blur-sm hover:bg-
white/30 rounded-full transiti
              aria-label="Close event details"
            >
              <CloseIcon style={{ fontSize: 20 }} className="text-white" />
            </button>


            {/* Event Header */}
            <div className="flex items-start gap-3 pr-24">
              <div className="p-3 bg-white/10 backdrop-blur-md rounded-xl border border-
white/20">
                {getActivityIcon(event.type)}
              </div>
              <div className="flex-1 min-w-0">
                <h2 className="text-xl sm:text-2xl font-bold text-white truncate drop-shadow-
md">{event.title}</h2>
                <p className="text-sm text-white/90 mt-1 line-clamp-2 drop-shadow-sm">
                  {event.description}
                </p>
              </div>
            </div>
          </div>

          <div className="overflow-y-auto flex-1 min-h-0">
            <div className="p-4 pb-8 space-y-4">

              {/* Who's Joining Section - Prominent, Moved Up */}
              <div className="bg-gradient-to-br from-primary/10 via-success/5 to-primary/10
rounded-xl p-5 border-2 bord
                <h3 className="text-lg font-bold mb-4 flex items-center gap-2">
                  <PeopleIcon style={{ fontSize: 24 }} className="text-primary" />
                  Who's Joining ({participantsCount})
                </h3>

                {loadingParticipants ? (
                  <div className="text-center py-8">
                    <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-primary
mx-auto"></div>
                    <p className="text-sm text-muted-foreground mt-3">Loading participants...</p>
                  </div>
                ) : participantsData.length === 0 ? (
                  <div className="text-center py-8 bg-muted/50 rounded-lg border border-border">
                    <PeopleIcon style={{ fontSize: 48 }} className="text-muted-foreground mx-
auto mb-2" />
                    <p className="text-base font-semibold text-foreground">No participants yet</
p>
                    <p className="text-sm text-muted-foreground mt-1">Be the first to join!</p>
                  </div>
                ) : (
                  <div className="grid grid-cols-3 sm:grid-cols-4 gap-3">
                    {participantsData.map((participant, index) => {
                      const activities = participant.activities || [];
                      const isCurrentUser = participant.id === user?.uid;

                      return (
                        <motion.div
                          key={participant.id}
                          initial={{ opacity: 0, scale: 0.9 }}
                          animate={{ opacity: 1, scale: 1 }}
                          transition={{ delay: index * 0.05 }}
                          className={`flex flex-col items-center p-3 border-2 rounded-xl bg-card
hover:bg-secondary tran...
                            isCurrentUser ? "border-primary ring-2 ring-primary/40 bg-
primary/5" : "border-border"
```

```
                        }`}
                      >
                        <Avatar
                          src={participant.avatar}
                          alt={participant.name}
                          sx={{ width: 64, height: 64 }}
                          className={isCurrentUser ? "ring-2 ring-primary" : ""}
                        />
                        <p className="text-sm font-semibold mt-2 text-center truncate w-full">
                          {isCurrentUser ? "You" : participant.name}
                        </p>
                        {activities.length > 0 && (
                          <div className="flex gap-1 mt-1">
                            {activities.map((activity) => (
                              <span key={activity}>
                                {getSmallActivityIcon(activity)}
                              </span>
                            ))}
                          </div>
                        )}
                      </motion.div>
                    );
                  })}
                </div>
              )}
            </div>

            {/* Combined Date & Time Card */}
            <div className="bg-gradient-to-br from-background via-primary/5 via-success/5 to-
background rounded-xl p-4">
                <div className="flex items-center gap-4">
                  <div className="flex items-center gap-3 flex-1">
                    <div className="p-2.5 bg-primary/15 rounded-lg">
                      <EventIcon style={{ fontSize: 20 }} className="text-primary" />
                    </div>
                    <div className="flex-1 min-w-0">
                      <p className="text-xs text-muted-foreground font-medium mb-0.5">Date</p>
                      <p className="text-sm font-bold text-foreground truncate">{event.date}</p>
                    </div>
                  </div>
                  <div className="w-px h-12 bg-border"></div>
                  <div className="flex items-center gap-3 flex-1">
                    <div className="p-2.5 bg-success/15 rounded-lg">
                      <AccessTimeIcon style={{ fontSize: 20 }} className="text-success" />
                    </div>
                    <div className="flex-1 min-w-0">
                      <p className="text-xs text-muted-foreground font-medium mb-0.5">Time</p>
                      <p className="text-sm font-bold text-foreground truncate">{event.time}</p>
                    </div>
                  </div>
                </div>
                {/* Distance - Subtle text below */}
                {event.distanceValue !== Infinity && (
                  <div className="mt-3 pt-3 border-t border-border/50">
                    <p className="text-xs text-muted-foreground text-center">
                      Ø=ÜÍ {event.distance} away
                    </p>
                  </div>
                )}
            </div>

            {/* Action Buttons - Horizontal Layout */}
            <div className="flex gap-3">
              {/* View on Map Button - Enhanced */}
              <Button
                onClick={() => {
                  if (event.lat && event.lng) {
                    // If onShowRoute callback is provided, show route on map
                    // Otherwise, fallback to opening Google Maps externally
                    if (onShowRoute) {
                      onShowRoute(event.lat, event.lng);
                      // Close the panel after showing route
```

```
                        onClose();
                      } else {
                        openGoogleMapsNavigation(event.lat, event.lng);
                      }
                    } else {
                      toast.error("Location not available");
                    }
                  }}
                  className="flex-1 min-h-[52px] text-base font-semibold text-white bg-gradient-to-r from-primary via-p...
                  <NavigationIcon style={{ fontSize: 20 }} className="mr-2" />
                  <span>View on Map</span>
                </Button>

                {/* Join/Leave Button - Enhanced */}
                <Button
                  onClick={() => {
                    if (event.isJoined || isUserJoined) {
                      onLeave(event.id);
                    } else {
                      onJoin(event.id);
                    }
                  }}
                  className={`flex-1 min-h-[52px] text-base font-semibold transition-all ${
                    event.isJoined || isUserJoined
                      ? "bg-destructive text-destructive-foreground hover:bg-destructive/90 border-2 border-destructive ...
                      : "bg-gradient-to-r from-primary via-primary to-success hover:from-primary/90 hover:via-primary/90...
                  }`}
                  variant="default"
                >
                  {event.isJoined || isUserJoined ? (
                    <>
                      <CheckCircleIcon style={{ fontSize: 20 }} className="mr-2" />
                      <span className="hidden sm:inline">Joined - Leave</span>
                      <span className="sm:hidden">Leave</span>
                    </>
                  ) : (
                    <>
                      <PeopleIcon style={{ fontSize: 20 }} className="mr-2" />
                      <span>Join Event</span>
                    </>
                  )}
                </Button>
              </div>

              {/* Event Creator/Host Section - Enhanced */}
              {event.hostName && (
                <div className="bg-gradient-to-br from-muted/60 via-muted/40 to-muted/60 rounded-xl p-4 border border-bo...
                  <div className="flex items-center gap-3">
                    <div className="relative">
                      <Avatar
                        src={event.hostAvatar}
                        alt={event.hostName}
                        sx={{ width: 48, height: 48 }}
                        className="ring-2 ring-primary/20"
                      />
                      <div className="absolute -bottom-1 -right-1 w-4 h-4 bg-primary rounded-full border-2 border-backgr... <CheckCircleIcon style={{ fontSize: 12 }} className="text-white" />
                      </div>
                    </div>
                    <div className="flex-1 min-w-0">
                      <p className="text-sm font-bold text-foreground truncate">{event.hostName}</p>
                      <p className="text-xs text-muted-foreground font-medium">Event Organizer</p>
                    </div>
                  </div>
                </div>
              )}

              {/* Creator Actions - Edit/Delete - Enhanced */}
              {isEventCreator && (
                <div className="flex gap-3 pb-3 border-b border-border/50">
```

```
                  <Button
                    onClick={handleEdit}
                    variant="outline"
                    className="flex-1 min-h-[48px] border-2 hover:bg-primary/10 hover:border-
primary/50 font-semibold">
                    <EditIcon style={{ fontSize: 20 }} className="mr-1 sm:mr-2" />
                    <span className="hidden sm:inline">Edit Event</span>
                    <span className="sm:hidden">Edit</span>
                  </Button>
                  <Button
                    onClick={handleDelete}
                    variant="outline"
                    className="flex-1 min-h-[48px] border-2 border-destructive/50 text-
destructive hover:text-destructiv...
                    <DeleteIcon style={{ fontSize: 20 }} className="mr-1 sm:mr-2" />
                    <span className="hidden sm:inline">Delete Event</span>
                    <span className="sm:hidden">Delete</span>
                  </Button>
                </div>
              )}

              {/* Comments Section */}
              <div className="border-t border-border pt-4">
                <h3 className="text-sm font-semibold mb-3">Comments</h3>

                {(isUserJoined || event.isJoined) ? (
                  <>
                    {/* Comments List */}
                    <div className="space-y-3 mb-4 max-h-32 overflow-y-auto">
                      {comments.length === 0 ? (
                        <p className="text-sm text-muted-foreground">No comments yet</p>
                      ) : (
                        comments.map((comment) => (
                          <div key={comment.id} className="flex gap-2 group">
                            <Avatar
                              src={comment.userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(comment....alt={comment.userName}
                              sx={{ width: 32, height: 32 }}
                            />
                            <div className="flex-1">
                              <div className="flex items-center gap-2">
                                <span className="text-sm font-medium">{comment.userName}</span>
                                <span className="text-xs text-muted-foreground">
                                  {new Date(comment.timestamp).toLocaleDateString(undefined, {
                                    month: 'short',
                                    day: 'numeric',
                                    hour: '2-digit',
                                    minute: '2-digit'
                                  })}
                                </span>
                                {canDeleteComment(comment) && (
                                  <button
                                    onClick={() => handleDeleteComment(comment.id)}
                                    disabled={deletingCommentId === comment.id}
                                    className="ml-auto opacity-0 group-hover:opacity-100
transition-opacity text-muted-f...
                                    title="Delete comment"
                                  >
                                    <DeleteIcon style={{ fontSize: 16 }} />
                                  </button>
                                )}
                              </div>
                              <p className="text-sm text-foreground mt-1">{comment.text}</p>
                            </div>
                          </div>
                        ))
                      )}
                    </div>

                    {/* Comment Input */}
                    <div className="flex gap-2">
                      <Input
```

```
                        type="text"
                        placeholder="Add a comment..."
                        value={commentText}
                        onChange={(e) => setCommentText(e.target.value)}
                        onKeyPress={(e) => {
                          if (e.key === "Enter" && !e.shiftKey) {
                            e.preventDefault();
                            handleAddComment();
                          }
                        }}
                        className="flex-1"
                        disabled={isAddingComment}
                      />
                      <Button
                        onClick={handleAddComment}
                        disabled={!commentText.trim() || isAddingComment}
                        size="icon"
                      >
                        {isAddingComment ? (
                          <div className="w-4 h-4 border-2 border-white border-t-transparent
rounded-full animate-spin" ... ) : (
                          <SendIcon style={{ fontSize: 18 }} />
                        )}
                      </Button>
                    </div>
                  </>
                ) : (
                  <div className="text-center py-4 bg-muted/50 rounded-lg border border-border">
                    <PeopleIcon style={{ fontSize: 32 }} className="text-muted-foreground mx-
auto mb-2" />
                    <p className="text-sm font-semibold text-foreground">Join to see comments</p>
                    <p className="text-xs text-muted-foreground mt-1">
                      You must join this event to view and add comments
                    </p>
                  </div>
                )}
              </div>
            </div>
          </motion.div>
          </>
        ) : null}
      </AnimatePresence>
    );
};
```

File: src/components/EventsTopBar.tsx

Lines: 180

```tsx
import { useState, ReactNode } from "react";
import { motion } from "framer-motion";
import { Input } from "@/components/ui/input";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import SearchIcon from "@mui/icons-material/Search";
import ArrowDropDownIcon from "@mui/icons-material/ArrowDropDown";

type EventType = "running" | "cycling" | "walking" | "others";

interface EventsTopBarProps {
  activityFilter: EventType | "all";
  onActivityFilterChange: (filter: EventType | "all") => void;
  searchQuery: string;
  onSearchChange: (query: string) => void;
  /** Optional element to render on the right side (e.g., notification bell) */
  rightSlot?: ReactNode;
}

export const EventsTopBar = ({
  activityFilter,
  onActivityFilterChange,
  searchQuery,
  onSearchChange,
  rightSlot,
}: EventsTopBarProps) => {
  const getActivityIcon = (type: EventType | "all") => {
    switch (type) {
      case "running":
        return <DirectionsRunIcon style={{ fontSize: 20 }} className="text-success" />;
      case "cycling":
        return <DirectionsBikeIcon style={{ fontSize: 20 }} className="text-primary" />;
      case "walking":
        return <DirectionsWalkIcon style={{ fontSize: 20 }} className="text-warning" />;
      case "others":
        return <FitnessCenterIcon style={{ fontSize: 20 }} className="text-secondary" />;
      default:
        return <FitnessCenterIcon style={{ fontSize: 20 }} className="text-muted-foreground" />;
    }
  };

  const getActivityLabel = (type: EventType | "all") => {
    switch (type) {
      case "running":
        return "Running";
      case "cycling":
        return "Cycling";
      case "walking":
        return "Walking";
      case "others":
        return "Others";
      default:
        return "All Activities";
    }
  };

  return (
    <motion.div
```

```
  67 |       initial={{ opacity: 0, y: -20 }}
  68 |       animate={{ opacity: 1, y: 0 }}
  69 |       className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-20 border-b
border-border/50"
  70 |       style={{
  71 |         paddingTop: 'env(safe-area-inset-top)',
  72 |       }}
  73 |     >
  74 |       <div className="max-w-7xl mx-auto px-4 sm:px-6 py-3">
  75 |         <div className="flex items-center gap-2 sm:gap-3">
  76 |           {/* Workout Dropdown - First Icon */}
  77 |           <DropdownMenu>
  78 |             <DropdownMenuTrigger asChild>
  79 |               <motion.button
  80 |                 whileTap={{ scale: 0.95 }}
  81 |                 className="flex items-center gap-1 sm:gap-2 px-2 sm:px-3 py-2 rounded-lg bg-
muted hover:bg-accent transi...
  82 |                 {getActivityIcon(activityFilter)}
  83 |                 <span className="text-sm font-medium hidden sm:inline">
  84 |                   {getActivityLabel(activityFilter)}
  85 |                 </span>
  86 |                 <ArrowDropDownIcon style={{ fontSize: 18 }} className="text-muted-foreground
flex-shrink-0" />
  87 |               </motion.button>
  88 |             </DropdownMenuTrigger>
  89 |             <DropdownMenuContent align="start" className="w-48">
  90 |               <DropdownMenuItem
  91 |                 onClick={() => onActivityFilterChange("all")}
  92 |                 className="flex items-center gap-2"
  93 |               >
  94 |                 <FitnessCenterIcon style={{ fontSize: 18 }} className="text-muted-foreground" />
  95 |                 <span>All Activities</span>
  96 |                 {activityFilter === "all" && (
  97 |                   <span className="ml-auto text-primary">' </span>
  98 |                 )}
  99 |               </DropdownMenuItem>
 100 |               <DropdownMenuItem
 101 |                 onClick={() => onActivityFilterChange("running")}
 102 |                 className="flex items-center gap-2"
 103 |               >
 104 |                 <DirectionsRunIcon style={{ fontSize: 18 }} className="text-success" />
 105 |                 <span>Running</span>
 106 |                 {activityFilter === "running" && (
 107 |                   <span className="ml-auto text-primary">' </span>
 108 |                 )}
 109 |               </DropdownMenuItem>
 110 |               <DropdownMenuItem
 111 |                 onClick={() => onActivityFilterChange("cycling")}
 112 |                 className="flex items-center gap-2"
 113 |               >
 114 |                 <DirectionsBikeIcon style={{ fontSize: 18 }} className="text-primary" />
 115 |                 <span>Cycling</span>
 116 |                 {activityFilter === "cycling" && (
 117 |                   <span className="ml-auto text-primary">' </span>
 118 |                 )}
 119 |               </DropdownMenuItem>
 120 |               <DropdownMenuItem
 121 |                 onClick={() => onActivityFilterChange("walking")}
 122 |                 className="flex items-center gap-2"
 123 |               >
 124 |                 <DirectionsWalkIcon style={{ fontSize: 18 }} className="text-warning" />
 125 |                 <span>Walking</span>
 126 |                 {activityFilter === "walking" && (
 127 |                   <span className="ml-auto text-primary">' </span>
 128 |                 )}
 129 |               </DropdownMenuItem>
 130 |               <DropdownMenuItem
 131 |                 onClick={() => onActivityFilterChange("others")}
 132 |                 className="flex items-center gap-2"
 133 |               >
 134 |                 <FitnessCenterIcon style={{ fontSize: 18 }} className="text-secondary" />
 135 |                 <span>Others</span>
 136 |                 {activityFilter === "others" && (
 137 |
```

```
                    <span className="ml-auto text-primary">' </span>
                  )}
                </DropdownMenuItem>
              </DropdownMenuContent>
            </DropdownMenu>

            {/* Search Input */}
            <div className="flex-1 relative min-w-0">
              <SearchIcon
                className="absolute left-3 top-1/2 -translate-y-1/2 text-muted-foreground"
                style={{ fontSize: 20 }}
              />
              <Input
                type="text"
                placeholder="Search events..."
                value={searchQuery}
                onChange={(e) => onSearchChange(e.target.value)}
                className="pl-10 pr-8 sm:pr-10 h-10 sm:h-10 w-full bg-background min-h-[44px]"
              />
              {searchQuery && (
                <button
                  onClick={() => onSearchChange("")}
                  aria-label="Clear search"
hover:text-foreground touch-
                >
                  ×
                </button>
              )}
            </div>

            {/* Right Slot (e.g., Notification Bell) */}
            {rightSlot && (
              <div className="flex-shrink-0">
                {rightSlot}
              </div>
            )}
          </div>
        </div>
      </motion.div>
    );
};
```

File: src/components/FitnessLevelAvatar.tsx

Lines: 126

```
  1 | import * as React from "react";
  2 | import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
  3 | import { FitnessLevel } from "@/contexts/UserContext";
  4 | import { cn } from "@/lib/utils";
  5 | import { getProfilePictureUrl } from "@/utils/profilePicture";
  6 | import "@/styles/animations.css";
  7 |
  8 | interface FitnessLevelAvatarProps {
  9 |   photoURL?: string | null;
 10 |   name?: string;
 11 |   fitnessLevel?: FitnessLevel | string;
 12 |   size?: "sm" | "md" | "lg" | "xl";
 13 |   className?: string;
 14 |   showGlow?: boolean;
 15 | }
 16 |
 17 | /**
 18 |  * Avatar component with fitness level glow effect
 19 |  *
 20 |  * This component wraps the standard Avatar and adds a colored glow
 21 |  * around the avatar based on the user's fitness level:
 22 |  * - Beginner: Blue glow
 23 |  * - Intermediate: Green glow
 24 |  * - Pro: Purple glow
 25 |  *
 26 |  * The glow effect uses CSS box-shadow to create a soft, animated
 27 |  * border that pulses slightly to draw attention to the fitness level.
 28 |  */
 29 | export const FitnessLevelAvatar = React.forwardRef<
 30 |   HTMLDivElement,
 31 |   FitnessLevelAvatarProps
 32 | >(({ photoURL, name, fitnessLevel, size = "md", className, showGlow = true, ...props }, ref) => {
 33 |   // Get fitness level colors and styles
 34 |   const getFitnessLevelStyles = (level?: FitnessLevel | string) => {
 35 |     if (!level) return { glow: "", border: "", ring: "", pulse: "" };
 36 |
 37 |     const normalizedLevel = (level as string).toLowerCase();
 38 |
 39 |     switch (normalizedLevel) {
 40 |       case "beginner":
 41 |         return {
 42 |           glow: "shadow-[0_0_12px_rgba(59,130,246,0.6),0_0_24px_rgba(59,130,246,0.3)]",
 43 |           border: "border-blue-500",
 44 |           ring: "ring-2 ring-blue-400/50",
 45 |           pulse: "animate-glow-pulse-blue"
 46 |         };
 47 |       case "intermediate":
 48 |         return {
 49 |           glow: "shadow-[0_0_12px_rgba(34,197,94,0.6),0_0_24px_rgba(34,197,94,0.3)]",
 50 |           border: "border-green-500",
 51 |           ring: "ring-2 ring-green-400/50",
 52 |           pulse: "animate-glow-pulse-green"
 53 |         };
 54 |       case "pro":
 55 |         return {
 56 |           glow: "shadow-[0_0_12px_rgba(168,85,247,0.6),0_0_24px_rgba(168,85,247,0.3)]",
 57 |           border: "border-purple-500",
 58 |           ring: "ring-2 ring-purple-400/50",
 59 |           pulse: "animate-glow-pulse-purple"
 60 |         };
 61 |       default:
 62 |         return { glow: "", border: "border-border", ring: "", pulse: "" };
 63 |     }
 64 |   };
 65 |
 66 |   // Size classes
```

```
  const sizeClasses = {
    sm: "w-8 h-8",
    md: "w-12 h-12",
    lg: "w-16 h-16",
    xl: "w-20 h-20"
  };

  const styles = getFitnessLevelStyles(fitnessLevel);
  const sizeClass = sizeClasses[size];

  // Get profile picture using utility function for consistent fallback
  const profilePictureUrl = getProfilePictureUrl(photoURL, null, name);

  return (
    <div
      ref={ref}
      className={cn(
        "relative inline-block",
        showGlow && styles.glow,
        showGlow && styles.pulse,
        className
      )}
      {...props}
    >
      <Avatar
        className={cn(
          sizeClass,
          "border-2 transition-all duration-300 rounded-full overflow-hidden",
          showGlow ? styles.border : "border-border",
          showGlow && styles.ring
        )}
      >
        <AvatarImage
          src={profilePictureUrl}
          className="rounded-full object-cover"
        />
        <AvatarFallback className="rounded-full">
          {name?.charAt(0)?.toUpperCase() || "U"}
        </AvatarFallback>
      </Avatar>

      {/* Optional: Add a small badge indicator */}
      {fitnessLevel && showGlow && (
        <div
          className={cn(
            "absolute -bottom-0.5 -right-0.5 w-3 h-3 rounded-full border-2 border-background",
            fitnessLevel.toLowerCase() === "beginner" && "bg-blue-500",
            fitnessLevel.toLowerCase() === "intermediate" && "bg-green-500",
            fitnessLevel.toLowerCase() === "pro" && "bg-purple-500"
          )}
          title={fitnessLevel.charAt(0).toUpperCase() + fitnessLevel.slice(1)}
        />
      )}
    </div>
  );
});

FitnessLevelAvatar.displayName = "FitnessLevelAvatar";

```

File: src/components/FriendRequestModal.tsx

Lines: 78

```
import { X } from "lucide-react";
import { Button } from "@/components/ui/button";

interface FriendRequestModalProps {
  isOpen: boolean;
  onClose: () => void;
  userName: string;
  userAvatar?: string;
  onAccept: () => void;
  onDecline: () => void;
}

export const FriendRequestModal = ({
  isOpen,
  onClose,
  userName,
  userAvatar,
  onAccept,
  onDecline,
}: FriendRequestModalProps) => {
  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 z-50 flex items-end sm:items-center justify-center">
      {/* Backdrop */}
      <div
        className="absolute inset-0 bg-black/80"
        onClick={onClose}
      />

      {/* Modal */}
      <div className="relative w-full sm:max-w-md bg-background rounded-t-[20px] sm:rounded-
[20px] p-6 animate-slide-i...
        {/* Close button */}
        <button
          onClick={onClose}
          className="absolute right-4 top-4 p-2 rounded-full hover:bg-accent transition-colors"
          aria-label="Close"
        >
          <X className="h-5 w-5 text-muted-foreground" />
        </button>

        {/* Header */}
        <h2 className="text-xl font-bold text-foreground mb-6">
          Friend Request
        </h2>

        {/* User Info */}
        <div className="flex flex-col items-center text-center mb-8">
          <div className="w-[120px] h-[120px] rounded-full bg-gradient-to-br from-primary to-
primary/60 flex items-cente...
            {userAvatar || userName.charAt(0).toUpperCase()}
          </div>
          <h3 className="text-2xl font-bold text-foreground mb-2">{userName}</h3>
          <p className="text-muted-foreground">
            wants to add you as a friend
          </p>
        </div>

        {/* Action Buttons */}
        <div className="flex gap-3">
          <Button
            variant="outline"
            onClick={onDecline}
            className="flex-1 h-14 text-base border-2 border-destructive text-destructive
hover:bg-destructive hover:tex...
          >
            Decline
          </Button>
```

```
  67 |           <Button
  68 |             onClick={onAccept}
  69 |             className="flex-1 h-14 text-base bg-[hsl(142,76%,36%)] hover:bg-[hsl(142,76%,30%)]
text-white transition-col...
  70 |
  71 |             Accept
  72 |           </Button>
  73 |         </div>
  74 |       </div>
  75 |     </div>
  76 |   );
  77 | };
  78 |
```

File: src/components/InactivityWarningModal.tsx

Lines: 138

```
  1 | import { useEffect, useState } from "react";
  2 | import { Button } from "@/components/ui/button";
  3 | import {
  4 |   Dialog,
  5 |   DialogContent,
  6 |   DialogHeader,
  7 |   DialogTitle,
  8 |   DialogDescription,
  9 | } from "@/components/ui/dialog";
 10 | import { motion } from "framer-motion";
 11 | import WarningIcon from "@mui/icons-material/Warning";
 12 | import PauseIcon from "@mui/icons-material/Pause";
 13 |
 14 | interface InactivityWarningModalProps {
 15 |   open: boolean;
 16 |   onOpenChange: (open: boolean) => void;
 17 |   onDismiss: () => void;
 18 |   onPause: () => void;
 19 |   /** Time in seconds until auto-pause */
 20 |   autoPauseSeconds?: number;
 21 | }
 22 |
 23 | export const InactivityWarningModal = ({
 24 |   open,
 25 |   onOpenChange,
 26 |   onDismiss,
 27 |   onPause,
 28 |   autoPauseSeconds = 120, // 2 minutes default
 29 | }: InactivityWarningModalProps) => {
 30 |   const [countdown, setCountdown] = useState(autoPauseSeconds);
 31 |
 32 |   // Reset countdown when modal opens
 33 |   useEffect(() => {
 34 |     if (open) {
 35 |       setCountdown(autoPauseSeconds);
 36 |     }
 37 |   }, [open, autoPauseSeconds]);
 38 |
 39 |   // Countdown timer
 40 |   useEffect(() => {
 41 |     if (!open || countdown <= 0) {
 42 |       return;
 43 |     }
 44 |
 45 |     const interval = setInterval(() => {
 46 |       setCountdown((prev) => {
 47 |         if (prev <= 1) {
 48 |           // Auto-pause when countdown reaches 0
 49 |           clearInterval(interval);
 50 |           onPause();
 51 |           return 0;
 52 |         }
 53 |         return prev - 1;
 54 |       });
 55 |     }, 1000);
 56 |
 57 |     return () => clearInterval(interval);
 58 |   }, [open, countdown, onPause]);
 59 |
 60 |   const formatTime = (seconds: number): string => {
 61 |     const mins = Math.floor(seconds / 60);
 62 |     const secs = seconds % 60;
 63 |     return `${mins}:${secs.toString().padStart(2, "0")}`;
 64 |   };
 65 |
 66 |   const handleDismiss = () => {
```

```
    onDismiss();
    onOpenChange(false);
  };

  const handlePause = () => {
    onPause();
    onOpenChange(false);
  };

  return (
    <Dialog open={open} onOpenChange={onOpenChange}>
      <DialogContent className="sm:max-w-md" aria-describedby="inactivity-warning-description">
        <DialogHeader>
          <DialogTitle className="flex items-center gap-2">
            <WarningIcon className="text-warning" style={{ fontSize: 24 }} />
            No Movement Detected
          </DialogTitle>
        </DialogHeader>
        <div id="inactivity-warning-description" className="sr-only">
          No movement detected for 5 minutes. Your workout will be auto-paused in
{formatTime(countdown)} if you don't r...
        </div>

        <div className="space-y-4 py-4">
          <motion.div
            initial={{ scale: 0.9, opacity: 0 }}
            animate={{ scale: 1, opacity: 1 }}
            className="text-center space-y-2"
          >
            <p className="text-sm text-muted-foreground">
              We haven't detected any movement for 5 minutes. Are you still working out?
            </p>

            {countdown > 0 && (
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                className="flex items-center justify-center gap-2 mt-4"
              >
                <span className="text-xs text-muted-foreground">Auto-pause in:</span>
                <span className={`text-lg font-bold tabular-nums ${
                  countdown <= 30 ? "text-destructive" : "text-foreground"
                }`}>
                  {formatTime(countdown)}
                </span>
              </motion.div>
            )}
          </motion.div>

          <div className="flex gap-3 pt-2">
            <Button
              variant="outline"
              onClick={handleDismiss}
              className="flex-1"
            >
              I'm Still Here
            </Button>
            <Button
              onClick={handlePause}
              variant="secondary"
              className="flex-1"
            >
              <PauseIcon className="mr-2" style={{ fontSize: 18 }} />
              Pause Workout
            </Button>
          </div>
        </div>
      </DialogContent>
    </Dialog>
  );
};

```

File: src/components/LocationSharingModalSimple.tsx

Lines: 64

```
  1 | import { Button } from "@/components/ui/button";
  2 | import {
  3 |   Dialog,
  4 |   DialogContent,
  5 |   DialogHeader,
  6 |   DialogTitle,
  7 | } from "@/components/ui/dialog";
  8 | import LocationOnIcon from "@mui/icons-material/LocationOn";
  9 |
 10 | interface LocationSharingModalProps {
 11 |   open: boolean;
 12 |   onOpenChange: (open: boolean) => void;
 13 |   onShareLocation: () => void;
 14 |   currentLocation: { lat: number; lng: number } | null;
 15 |   isGettingLocation: boolean;
 16 | }
 17 |
 18 | const LocationSharingModal = ({
 19 |   open,
 20 |   onOpenChange,
 21 |   onShareLocation,
 22 |   currentLocation,
 23 |   isGettingLocation,
 24 | }: LocationSharingModalProps) => {
 25 |   return (
 26 |     <Dialog open={open} onOpenChange={onOpenChange}>
 27 |       <DialogContent className="sm:max-w-md">
 28 |         <DialogHeader>
 29 |           <DialogTitle>Share Your Location</DialogTitle>
 30 |         </DialogHeader>
 31 |
 32 |         <div className="space-y-6 py-4">
 33 |           {/* Location Status */}
 34 |           {isGettingLocation && (
 35 |             <div className="text-center text-sm text-muted-foreground animate-pulse">
 36 |               Ø=ÜÍ Getting your location...
 37 |             </div>
 38 |           )}
 39 |
 40 |           {!isGettingLocation && currentLocation && (
 41 |             <div className="text-center text-sm text-muted-foreground">
 42 |               Ø=ÜÍ Location ready: {currentLocation.lat.toFixed(4)}, {currentLocation.lng.toFixed(4)}
 43 |             </div>
 44 |           )}
 45 |
 46 |           {/* Share Button */}
 47 |           <Button
 48 |             onClick={onShareLocation}
 49 |             disabled={!currentLocation || isGettingLocation}
 50 |             className="w-full h-12 text-base"
 51 |             size="lg"
 52 |           >
 53 |             <LocationOnIcon className="mr-2" style={{ fontSize: 20 }} />
 54 |             {isGettingLocation ? "Getting location..." : "Share my location"}
 55 |           </Button>
 56 |         </div>
 57 |       </DialogContent>
 58 |     </Dialog>
 59 |   );
 60 | };
 61 |
 62 | export default LocationSharingModal;
 63 |
 64 |
```

File: src/components/MatchActionsModal.tsx

Lines: 284

```
import { useState } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
import { Badge } from "@/components/ui/badge";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import HourglassEmptyIcon from "@mui/icons-material/HourglassEmpty";
import CloseIcon from "@mui/icons-material/Close";
import ChatIcon from "@mui/icons-material/Chat";
import PersonIcon from "@mui/icons-material/Person";
import BlockIcon from "@mui/icons-material/Block";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import { toast } from "sonner";

type FriendStatus = "not_friends" | "request_pending" | "request_received" | "friends" | "denied";

interface MatchActionsModalProps {
  isOpen: boolean;
  onClose: () => void;
  user: {
    id: string;
    name: string;
    avatar: string;
    activity: string;
    distance: string;
    fitnessLevel?: string;
  };
  friendStatus?: FriendStatus;
  cooldownDays?: number;
  onAddFriend: () => void;
  onDecline: () => void;
  onChat: () => void;
  onViewProfile: () => void;
  onPoke?: () => void;
  hasPoked?: boolean;
  isWorkoutActive?: boolean;
}

export const MatchActionsModal = ({
  isOpen,
  onClose,
  user,
  friendStatus = "not_friends",
  cooldownDays = 0,
  onAddFriend,
  onDecline,
  onChat,
  onViewProfile,
  onPoke,
  hasPoked = false,
  isWorkoutActive = false
}: MatchActionsModalProps) => {
  const getActivityIcon = () => {
    switch (user.activity.toLowerCase()) {
      case "running":
        return <DirectionsRunIcon className="text-success" style={{ fontSize: 24 }} />;
      case "cycling":
        return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 24 }} />;
      case "walking":
        return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 24 }} />;
      default:
```

```
        return <DirectionsRunIcon style={{ fontSize: 24 }} />;
    }
  };

  const getFitnessLevelBadge = () => {
    if (!user.fitnessLevel) return null;
    const level = user.fitnessLevel.toLowerCase();
    const colors = {
      beginner: "bg-blue-100 text-blue-800",
      intermediate: "bg-green-100 text-green-800",
      pro: "bg-purple-100 text-purple-800"
    };
    return (
      <Badge className={colors[level as keyof typeof colors] || "bg-gray-100 text-gray-800"}>
        {user.fitnessLevel}
      </Badge>
    );
  };

  const handleDecline = () => {
    onDecline();
    toast.success(`${user.name} has been hidden. You can see them again in 3 days.`);
    onClose();
  };

  return (
    <AnimatePresence>
      {isOpen && (
        <>
          {/* Backdrop */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            onClick={onClose}
            className="fixed inset-0 bg-black/50 z-40"
          />

          {/* Modal */}
          <motion.div
            initial={{ y: "100%" }}
            animate={{ y: 0 }}
            exit={{ y: "100%" }}
            transition={{ type: "spring", damping: 25, stiffness: 300 }}
            className="fixed bottom-0 left-0 right-0 z-50 bg-background rounded-t-3xl shadow-
elevation-4 max-h-[90vh] ov...
            <div className="p-6 space-y-6">
              {/* Header */}
              <div className="flex items-center justify-between">
                <h2 className="text-2xl font-bold">Match Found</h2>
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={onClose}
                  className="h-10 w-10"
                >
                  <CloseIcon />
                </Button>
              </div>

              {/* User Info Card */}
              <Card className="p-6 space-y-4">
                <div className="flex items-center gap-4">
                  <Avatar className="h-20 w-20">
                    <AvatarImage src={user.avatar} alt={user.name} />
                    <AvatarFallback>{user.name.charAt(0)}</AvatarFallback>
                  </Avatar>
                  <div className="flex-1 space-y-2">
                    <div className="flex items-center gap-2">
                      <h3 className="text-xl font-bold">{user.name}</h3>
                      {getFitnessLevelBadge()}
```

```
                    </div>
                    <div className="flex items-center gap-2 text-muted-foreground">
                      {getActivityIcon()}
                      <span className="text-sm capitalize">{user.activity}</span>
                      <span className="text-sm">•</span>
                      <span className="text-sm">{user.distance} away</span>
                    </div>
                  </div>
                </div>
              </Card>

              {/* Action Buttons */}
              <div className="space-y-3">
                {/* Poke Button - Show before Add Friend (only when workout is active) */}
                {onPoke && friendStatus === "not_friends" && !hasPoked && isWorkoutActive && (
                  <Button
                    onClick={() => {
                      onPoke();
                      onClose();
                    }}
                    className="w-full h-14 text-base font-semibold bg-purple-500 hover:bg-
purple-600 text-white"
                  >
                    <TouchAppIcon className="mr-2" />
                    Poke {user.name}
                  </Button>
                )}

                {/* Show disabled poke button when workout is not active */}
                {onPoke && friendStatus === "not_friends" && !hasPoked && !isWorkoutActive && (
                  <Button
                    disabled
                    variant="outline"
                    className="w-full h-14 text-base font-semibold opacity-50"
                    title="You must have an active workout session to poke someone"
                  >
                    <TouchAppIcon className="mr-2" />
                    Poke {user.name} (Start workout first)
                  </Button>
                )}

                {hasPoked && (
                  <Button
                    disabled
                    variant="outline"
                    className="w-full h-14 text-base font-semibold opacity-50"
                  >
                    <TouchAppIcon className="mr-2" />
                    Poke Sent
                  </Button>
                )}

                {/* Add Friend Button */}
                {friendStatus === "not_friends" && cooldownDays === 0 && (
                  <Button
                    onClick={onAddFriend}
                    variant="outline"
                    className="w-full h-14 text-base font-semibold"
                  >
                    <PersonAddIcon className="mr-2" />
                    Add Friend
                  </Button>
                )}

                {friendStatus === "request_pending" && (
                  <Button
                    disabled
                    variant="outline"
                    className="w-full h-14 text-base font-semibold opacity-50"
                  >
                    <HourglassEmptyIcon className="mr-2" />
                    Request Pending
```

```
                  </Button>
                )}

                {friendStatus === "friends" && (
                  <Button
                    disabled
                    variant="outline"
                    className="w-full h-14 text-base font-semibold bg-success/10 border-success/30 text-success" >
                    <CheckCircleIcon className="mr-2" />
                    Friend
                  </Button>
                )}

                {cooldownDays > 0 && (
                  <div className="space-y-2">
                    <Button
                      disabled
                      variant="outline"
                      className="w-full h-14 text-base font-semibold opacity-50"
                    >
                      <PersonAddIcon className="mr-2" />
                      Add Friend
                    </Button>
                    <p className="text-xs text-center text-muted-foreground">
                      Try again in {cooldownDays} day{cooldownDays !== 1 ? "s" : ""}
                    </p>
                  </div>
                )}

                {/* Chat Button */}
                <Button
                  onClick={() => {
                    onChat();
                    onClose();
                  }}
                  className="w-full h-14 text-base font-semibold bg-primary text-primary-foreground"
                >
                  <ChatIcon className="mr-2" />
                  Chat
                </Button>

                {/* View Profile Button */}
                <Button
                  onClick={() => {
                    onViewProfile();
                    onClose();
                  }}
                  variant="outline"
                  className="w-full h-14 text-base font-semibold"
                >
                  <PersonIcon className="mr-2" />
                  View Profile
                </Button>

                {/* Decline Button */}
                {friendStatus !== "friends" && (
                  <Button
                    onClick={handleDecline}
                    variant="ghost"
                    className="w-full h-12 text-base text-destructive hover:text-destructive hover:bg-destructive/10" >
                    <BlockIcon className="mr-2" />
                    Decline (Hide for 3 days)
                  </Button>
                )}
              </div>
            </div>
          </motion.div>
        </>
      )}
```

```
280 |     </AnimatePresence>
281 |   );
282 | };
283 |
284 |
```

File: src/components/MessageModal.tsx

Lines: 228

```
import { useState } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Textarea } from "@/components/ui/textarea";
import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
import CloseIcon from "@mui/icons-material/Close";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";

interface MessageModalProps {
  show: boolean;
  userName: string;
  onClose: () => void;
  onSend: (message: string, isTemplate: boolean) => void;
}

const TEMPLATE_MESSAGES = [
  {
    id: 1,
    text: "Hi! Want to workout together?",
    emoji: "Ø=Ü¬",
  },
  {
    id: 2,
    text: "Great to see another runner nearby!",
    emoji: "Ø=Ü¬",
  },
  {
    id: 3,
    text: "Would you like to join me for a run?",
    emoji: "Ø=Ü¬",
  },
  {
    id: 4,
    text: "Nice to meet you! Let's connect.",
    emoji: "Ø=Ü¬",
  },
];

export const MessageModal = ({ show, userName, onClose, onSend }: MessageModalProps) => {
  const [selectedTemplate, setSelectedTemplate] = useState<number | null>(null);
  const [customMessage, setCustomMessage] = useState("");
  const [isSending, setIsSending] = useState(false);

  const maxLength = 500;
  const messageToSend = selectedTemplate
    ? TEMPLATE_MESSAGES.find(t => t.id === selectedTemplate)?.text || ""
    : customMessage;
  const canSend = messageToSend.trim().length > 0 && !isSending;

  const handleTemplateClick = (templateId: number) => {
    if (selectedTemplate === templateId) {
      setSelectedTemplate(null);
    } else {
      setSelectedTemplate(templateId);
      setCustomMessage(""); // Clear custom message when template selected
    }
  };

  const handleCustomMessageChange = (value: string) => {
    if (value.length <= maxLength) {
      setCustomMessage(value);
      setSelectedTemplate(null); // Clear template when typing
    }
  };

  const handleSend = async () => {
```

```
    if (!canSend) return;

    setIsSending(true);

    // Simulate sending delay
    await new Promise(resolve => setTimeout(resolve, 500));

    onSend(messageToSend, selectedTemplate !== null);

    // Reset state
    setSelectedTemplate(null);
    setCustomMessage("");
    setIsSending(false);
  };

  const handleClose = () => {
    setSelectedTemplate(null);
    setCustomMessage("");
    setIsSending(false);
    onClose();
  };

  return (
    <AnimatePresence>
      {show && (
        <>
          {/* Backdrop */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            onClick={handleClose}
            className="fixed inset-0 bg-black/50 backdrop-blur-sm z-40"
          />

          {/* Modal */}
          <motion.div
            initial={{ y: "100%", opacity: 0 }}
            animate={{ y: 0, opacity: 1 }}
            exit={{ y: "100%", opacity: 0 }}
            transition={{ type: "spring", stiffness: 300, damping: 30 }}
            className="fixed bottom-0 left-0 right-0 z-50 bg-card rounded-t-3xl shadow-
elevation-4 max-h-[85vh] overflow...
            {/* Header */}
            <div className="flex items-center justify-between p-4 border-b border-border sticky
top-0 bg-card z-10"> <div className="flex items-center gap-2">
                <ChatBubbleOutlineIcon className="text-primary" />
                <h2 className="text-lg font-bold">Send Message to {userName}</h2>
              </div>
              <Button
                variant="ghost"
                size="icon"
                onClick={handleClose}
                className="rounded-full"
              >
                <CloseIcon />
              </Button>
            </div>

            {/* Content */}
            <div className="p-4 overflow-y-auto max-h-[calc(85vh-140px)] space-y-4">
              {/* Quick Messages Section */}
              <div>
                <p className="text-sm font-semibold mb-3 text-muted-foreground">Quick Messages:</
p>
                <div className="space-y-2">
                  {TEMPLATE_MESSAGES.map((template) => (
                    <motion.button
                      key={template.id}
                      whileTap={{ scale: 0.98 }}
                      onClick={() => handleTemplateClick(template.id)}
                      className={`
```

```
                        w-full p-4 rounded-xl text-left transition-all duration-200
                        border-2 min-h-[56px] touch-target relative
                        ${selectedTemplate === template.id
                          ? "border-primary bg-primary/10 shadow-elevation-2"
                          : "border-border bg-card hover:bg-secondary"
                        }
                      `}
                    >
                      <div className="flex items-start gap-3">
                        <span className="text-xl flex-shrink-0">{template.emoji}</span>
                        <p className={`text-sm font-medium flex-1 ${selectedTemplate ===
template.id ? "text-foreground"..{template.text}
                        </p>
                        {selectedTemplate === template.id && (
                          <motion.div
                            initial={{ scale: 0 }}
                            animate={{ scale: 1 }}
                            transition={{ type: "spring", stiffness: 400, damping: 15 }}
                          >
                            <CheckCircleIcon className="text-primary" fontSize="small" />
                          </motion.div>
                        )}
                      </div>
                    </motion.button>
                  ))}
                </div>
              </div>

              {/* Custom Message Section */}
              <div>
                <p className="text-sm font-semibold mb-2 text-muted-foreground">Or write your
own</p>
                <div className="relative">
                  <Textarea
                    value={customMessage}
                    onChange={(e) => handleCustomMessageChange(e.target.value)}
                    placeholder="Type your message..."
                    className={`
                      min-h-[100px] resize-none text-sm
                      ${customMessage.length > 0 ? "border-primary border-2" : ""}
                    `}
                  />
                  <p className="absolute bottom-2 right-2 text-xs text-muted-foreground">
                    {customMessage.length}/{maxLength}
                  </p>
                </div>
              </div>
            </div>

            {/* Footer */}
            <div className="p-4 border-t border-border bg-card sticky bottom-0">
              <div className="flex gap-3">
                <Button
                  variant="outline"
                  onClick={handleClose}
                  className="flex-1 h-12 text-base font-semibold"
                  disabled={isSending}
                >
                  Cancel
                </Button>
                <Button
                  onClick={handleSend}
                  disabled={!canSend}
                  className={`
                    flex-1 h-12 text-base font-semibold
                    bg-primary text-primary-foreground hover:bg-primary/90
                    ${canSend && !isSending ? "animate-pulse-slow" : ""}
                  `}
                >
                  {isSending ? (
                    <motion.div
                      animate={{ rotate: 360 }}
```

```
                      transition={{ duration: 1, repeat: Infinity, ease: "linear" }}
                    >
                      <ChatBubbleOutlineIcon />
                    </motion.div>
                  ) : (
                    <>
                      <ChatBubbleOutlineIcon className="mr-2" fontSize="small" />
                      Send Message
                    </>
                  )}
                </Button>
              </div>
            </div>
          </motion.div>
        </>
      )}
    </AnimatePresence>
  );
};

```

File: src/components/NavLink.tsx

Lines: 29

```tsx
import { NavLink as RouterNavLink, NavLinkProps } from "react-router-dom";
import { forwardRef } from "react";
import { cn } from "@/lib/utils";

interface NavLinkCompatProps extends Omit<NavLinkProps, "className"> {
  className?: string;
  activeClassName?: string;
  pendingClassName?: string;
}

const NavLink = forwardRef<HTMLAnchorElement, NavLinkCompatProps>(
  ({ className, activeClassName, pendingClassName, to, ...props }, ref) => {
    return (
      <RouterNavLink
        ref={ref}
        to={to}
        className={({ isActive, isPending }) =>
          cn(className, isActive && activeClassName, isPending && pendingClassName)
        }
        {...props}
      />
    );
  },
);

NavLink.displayName = "NavLink";

export { NavLink };

```

File: src/components/NearbyUsersAccordion.tsx

Lines: 258

```tsx
// Accordion component for displaying nearby matched users
import { useState } from "react";
import { motion, AnimatePresence } from "framer-motion";
import {
  Accordion,
  AccordionContent,
  AccordionItem,
  AccordionTrigger,
} from "@/components/ui/accordion";
import { Card } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
import { FitnessLevelAvatar } from "@/components/FitnessLevelAvatar";
import { MatchResult } from "@/services/matchingService";
import { formatDistance } from "@/utils/distance";
import { getDisplayName } from "@/utils/anonymousName";
import { getProfilePictureUrl } from "@/utils/profilePicture";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import MessageIcon from "@mui/icons-material/Message";
import PersonIcon from "@mui/icons-material/Person";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import SpeedIcon from "@mui/icons-material/Speed";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import PeopleIcon from "@mui/icons-material/People";

interface NearbyUsersAccordionProps {
  matches: MatchResult[];
  onViewProfile: (userId: string) => void;
  onAddFriend: (userId: string) => void;
  onSendMessage: (userId: string) => void;
  onPoke?: (userId: string) => void;
  loading?: boolean;
  pokes?: string[]; // Array of user IDs who poked the current user
  isWorkoutActive?: boolean;
}

export const NearbyUsersAccordion = ({
  matches,
  onViewProfile,
  onAddFriend,
  onSendMessage,
  onPoke,
  loading = false,
  pokes = [],
  isWorkoutActive = false
}: NearbyUsersAccordionProps) => {
  const getActivityIcon = (activity: string) => {
    switch (activity.toLowerCase()) {
      case "running":
        return <DirectionsRunIcon className="text-success" style={{ fontSize: 20 }} />;
      case "cycling":
        return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 20 }} />;
      case "walking":
        return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 20 }} />;
      default:
        return null;
    }
  };

  const getFitnessLevelColor = (level: string) => {
    switch (level) {
      case "beginner":
```

```
  67 |         return "bg-blue-500";
  68 |       case "intermediate":
  69 |         return "bg-yellow-500";
  70 |       case "pro":
  71 |         return "bg-red-500";
  72 |       default:
  73 |         return "bg-gray-500";
  74 |     }
  75 |   };
  76 |
  77 |   const getScoreColor = (score: number) => {
  78 |     if (score >= 0.8) return "bg-success text-success-foreground";
  79 |     if (score >= 0.6) return "bg-primary text-primary-foreground";
  80 |     if (score >= 0.4) return "bg-warning text-warning-foreground";
  81 |     return "bg-muted text-muted-foreground";
  82 |   };
  83 |
  84 |   if (loading) {
  85 |     return (
  86 |       <Card className="p-4">
  87 |         <div className="text-center text-muted-foreground">Loading matches...</div>
  88 |       </Card>
  89 |     );
  90 |   }
  91 |
  92 |   if (matches.length === 0) {
  93 |     return (
  94 |       <Card className="p-6 text-center">
  95 |         <PeopleIcon style={{ fontSize: 48 }} className="text-muted-foreground/30 mx-auto mb-2" />
  96 |         <p className="text-sm text-muted-foreground">No matches found nearby</p>
  97 |         <p className="text-xs text-muted-foreground mt-1">Try adjusting your preferences in
settings</p> </Card>
  99 |     );
 100 |   }
 101 |
 102 |   // Sort matches: poked users first
 103 |   const sortedMatches = [...matches].sort((a, b) => {
 104 |     const aPoked = pokes.includes(a.user.uid);
 105 |     const bPoked = pokes.includes(b.user.uid);
 106 |     if (aPoked && !bPoked) return -1;
 107 |     if (!aPoked && bPoked) return 1;
 108 |     return 0;
 109 |   });
 110 |
 111 |   return (
 112 |     <Card className="overflow-hidden">
 113 |       <Accordion type="single" collapsible className="w-full">
 114 |         <AccordionItem value="matches" className="border-0">
 115 |           <AccordionTrigger className="px-4 py-3 hover:no-underline">
 116 |             <div className="flex items-center justify-between w-full pr-4">
 117 |               <div className="flex items-center gap-2">
 118 |                 <PeopleIcon style={{ fontSize: 20 }} />
 119 |                 <span className="font-semibold">Nearby Matches</span>
 120 |                 <Badge variant="secondary" className="ml-2">
 121 |                   {matches.length}
 122 |                 </Badge>
 123 |               </div>
 124 |             </div>
 125 |           </AccordionTrigger>
 126 |           <AccordionContent className="px-0 pb-0">
 127 |             <div className="space-y-2 pb-2">
 128 |               {sortedMatches.map((match, index) => {
 129 |                 const user = match.user;
 130 |                 const username = user.name || null;
 131 |                 const activity = user.activity || null;
 132 |                 const displayName = getDisplayName(username, user.uid, activity);
 133 |                 const distanceKm = match.distance / 1000;
 134 |                 const score = match.score;
 135 |                 const isPoked = pokes.includes(user.uid);
 136 |
 137 |                 // Get profile picture using utility function
```

```
                const profilePictureUrl = getProfilePictureUrl(
                  user.photoURL,
                  user.avatar,
                  displayName
                );

                return (
                  <motion.div
                    key={user.uid}
                    data-poked={isPoked}
                    initial={{ opacity: 0, y: 10 }}
                    animate={{ opacity: 1, y: 0 }}
                    transition={{ delay: index * 0.05 }}
                    className={`px-4 py-3 border-b border-border/50 last:border-0 hover:bg-
muted/50 transition-colors re isPoked ? 'bg-purple-50/50 dark:bg-purple-950/20 border-purple-300/30' : ''
                    }`}
                  >
                    <div className="flex items-center gap-3">
                      {/* Avatar - Clickable to view profile */}
                      <button
                        onClick={() => onViewProfile(user.uid)}
                        className="cursor-pointer hover:opacity-80 transition-opacity"
                      >
                        <FitnessLevelAvatar
                          photoURL={profilePictureUrl}
                          name={displayName}
                          fitnessLevel={user.fitnessLevel}
                          size="md"
                          showGlow={true}
                        />
                      </button>

                      {/* User Info */}
                      <div className="flex-1 min-w-0">
                        <div className="flex items-center gap-2 mb-1">
                          <h4 className="font-semibold text-sm truncate">
                            {displayName}
                          </h4>
                          {isPoked && (
                            <Badge className="bg-purple-500 text-white text-[10px] font-bold
px-1.5 py-0.5 rounded-full ...       POKED YOU!
                            </Badge>
                          )}
                          <Badge className="bg-primary text-primary-foreground text-xs px-1.5
py-0">                      {formatDistance(distanceKm)}
                          </Badge>
                        </div>

                        {/* Activity & Fitness Level */}
                        <div className="flex items-center gap-2 text-xs mb-2">
                          <div className="flex items-center gap-1">
                            {getActivityIcon(user.activity)}
                            <span className="capitalize">{user.activity}</span>
                          </div>
                          <span>•</span>
                          <Badge
                            variant="outline"
                            className={`${getFitnessLevelColor(user.fitnessLevel)} text-white
border-0 text-xs px-1.5 py...    >
                            {user.fitnessLevel}
                          </Badge>
                          {user.pace > 0 && (
                            <>
                              <span>•</span>
                              <div className="flex items-center gap-1">
                                <SpeedIcon style={{ fontSize: 14 }} />
                                <span>
                                  {user.activity === "cycling"
                                    ? `${user.pace.toFixed(1)} km/h`
                                    : `${user.pace.toFixed(1)} min/km`}
                                </span>
```

```
                              </div>
                            </>
                          )}
                        </div>

                        {/* Actions */}
                        <div className="flex items-center justify-center gap-2 mt-2">
                          {onPoke && isWorkoutActive && (
                            <Button
                              size="sm"
                              className="flex-1 h-8 text-xs justify-center bg-purple-500
hover:bg-purple-600 text-white"
                              onClick={() => onPoke(user.uid)}
                            >
                              <TouchAppIcon style={{ fontSize: 14 }} className="mr-1" />
                              Poke
                            </Button>
                          )}
                          <Button
                            size="sm"
                            variant="outline"
                            className="flex-1 h-8 text-xs justify-center"
                            onClick={() => onAddFriend(user.uid)}
                          >
                            <PersonAddIcon style={{ fontSize: 14 }} className="mr-1" />
                            Add
                          </Button>
                          <Button
                            size="sm"
                            variant="outline"
                            className="flex-1 h-8 text-xs justify-center"
                            onClick={() => onSendMessage(user.uid)}
                          >
                            <MessageIcon style={{ fontSize: 14 }} className="mr-1" />
                            Message
                          </Button>
                        </div>
                      </div>
                    </div>
                  </motion.div>
                );
              })}
            </div>
          </AccordionContent>
        </AccordionItem>
      </Accordion>
    </Card>
  );
};
```

File: src/components/NearestEventModal.tsx

Lines: 194

```tsx
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
import { Badge } from "@/components/ui/badge";
import CloseIcon from "@mui/icons-material/Close";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import EventIcon from "@mui/icons-material/Event";
import AccessTimeIcon from "@mui/icons-material/AccessTime";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import PeopleIcon from "@mui/icons-material/People";
import { formatDistance } from "@/utils/distance";

interface Event {
  id: string;
  title: string;
  description: string;
  type: "running" | "cycling" | "walking" | "others";
  date: string;
  time: string;
  location: string;
  lat: number;
  lng: number;
  hostName: string;
  hostAvatar?: string;
  participants?: string[];
  distance?: string;
  distanceValue?: number;
}

interface NearestEventModalProps {
  isOpen: boolean;
  event: Event | null;
  onClose: () => void;
  onViewDetails: () => void;
}

export const NearestEventModal = ({
  isOpen,
  event,
  onClose,
  onViewDetails
}: NearestEventModalProps) => {
  if (!event) return null;

  const getActivityIcon = () => {
    switch (event.type) {
      case "running":
        return <DirectionsRunIcon style={{ fontSize: 24 }} className="text-success" />;
      case "cycling":
        return <DirectionsBikeIcon style={{ fontSize: 24 }} className="text-primary" />;
      case "walking":
        return <DirectionsWalkIcon style={{ fontSize: 24 }} className="text-warning" />;
      default:
        return <FitnessCenterIcon style={{ fontSize: 24 }} className="text-secondary" />;
    }
  };

  const participantsCount = Array.isArray(event.participants) ? event.participants.length : 0;

  return (
    <AnimatePresence>
      {isOpen && (
```

```
  67 |         <>
  68 |           {/* Backdrop */}
  69 |           <motion.div
  70 |             initial={{ opacity: 0 }}
  71 |             animate={{ opacity: 1 }}
  72 |             exit={{ opacity: 0 }}
  73 |             className="fixed inset-0 bg-black/60 backdrop-blur-sm z-50"
  74 |             onClick={onClose}
  75 |           />
  76 |
  77 |           {/* Modal Card */}
  78 |           <motion.div
  79 |             initial={{ opacity: 0, scale: 0.8, y: 50 }}
  80 |             animate={{ opacity: 1, scale: 1, y: 0 }}
  81 |             exit={{ opacity: 0, scale: 0.8, y: 50 }}
  82 |             transition={{ type: "spring", stiffness: 300, damping: 25 }}
  83 |             className="fixed top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2 z-50 w-[90%] max-
w-md"
  84 |             onClick={(e) => e.stopPropagation()}
  85 |           >
  86 |             <Card className="bg-card rounded-3xl shadow-elevation-4 border-2 border-primary/50
overflow-hidden">
  87 |               {/* Header */}
  88 |               <div className="bg-gradient-to-r from-primary via-primary to-success p-4 pb-6
relative">
  89 |                 <button
  90 |                   onClick={onClose}
  91 |                   className="absolute top-3 right-3 p-1.5 rounded-full bg-background/20 hover:bg-
background/30 transitio>..
  92 |
  93 |                   <CloseIcon style={{ fontSize: 20 }} className="text-white" />
  94 |                 </button>
  95 |                 <div className="flex items-center gap-3 mt-2">
  96 |                   <div className="p-2 bg-background/20 rounded-full">
  97 |                     {getActivityIcon()}
  98 |                   </div>
  99 |                   <div className="flex-1">
 100 |                     <h3 className="text-lg font-bold text-white">Nearest Event Found!</h3>
 101 |                     <p className="text-xs text-white/80">We found an event near you</p>
 102 |                   </div>
 103 |                 </div>
 104 |               </div>
 105 |
 106 |               {/* Content */}
 107 |               <div className="p-5 space-y-4">
 108 |                 {/* Event Title */}
 109 |                 <div>
 110 |                   <h2 className="text-xl font-bold text-foreground mb-1">{event.title}</h2>
 111 |                   {event.description && (
 112 |                     <p className="text-sm text-muted-foreground line-clamp-2">{event.description}
</p>
 113 |                   )}
 114 |                 </div>
 115 |
 116 |                 {/* Event Info Grid */}
 117 |                 <div className="grid grid-cols-2 gap-3">
 118 |                   {/* Date & Time */}
 119 |                   <div className="flex items-center gap-2 p-3 bg-muted/50 rounded-xl">
 120 |                     <EventIcon style={{ fontSize: 18 }} className="text-primary" />
 121 |                     <div className="flex-1 min-w-0">
 122 |                       <p className="text-xs text-muted-foreground">Date</p>
 123 |                       <p className="text-sm font-semibold truncate">{event.date}</p>
 124 |                     </div>
 125 |                   </div>
 126 |                   <div className="flex items-center gap-2 p-3 bg-muted/50 rounded-xl">
 127 |                     <AccessTimeIcon style={{ fontSize: 18 }} className="text-primary" />
 128 |                     <div className="flex-1 min-w-0">
 129 |                       <p className="text-xs text-muted-foreground">Time</p>
 130 |                       <p className="text-sm font-semibold truncate">{event.time}</p>
 131 |                     </div>
 132 |                   </div>
 133 |                 </div>
 134 |
 135 |                 {/* Distance & Participants */}
 136 |                 <div className="flex items-center justify-between p-3 bg-primary/5 rounded-xl
border border-primary/20">
 137 |                   <div className="flex items-center gap-2">
```

```
                    <LocationOnIcon style={{ fontSize: 18 }} className="text-primary" />
                    <div>
                      <p className="text-xs text-muted-foreground">Distance</p>
                      <p className="text-sm font-bold text-primary">
                        {event.distance || "Unknown"}
                      </p>
                    </div>
                  </div>
                  {participantsCount > 0 && (
                    <div className="flex items-center gap-2">
                      <PeopleIcon style={{ fontSize: 18 }} className="text-muted-foreground" />
                      <div>
                        <p className="text-xs text-muted-foreground">Joining</p>
                        <p className="text-sm font-bold text-foreground">{participantsCount}</p>
                      </div>
                    </div>
                  )}
                </div>

                {/* Host Info */}
                <div className="flex items-center gap-3 p-3 bg-muted/30 rounded-xl">
                  <Avatar className="w-10 h-10">
                    <AvatarImage src={event.hostAvatar} />
                    <AvatarFallback>{event.hostName.charAt(0).toUpperCase()}</AvatarFallback>
                  </Avatar>
                  <div className="flex-1 min-w-0">
                    <p className="text-xs text-muted-foreground">Hosted by</p>
                    <p className="text-sm font-semibold truncate">{event.hostName}</p>
                  </div>
                </div>

                {/* Action Buttons */}
                <div className="flex gap-3 pt-2">
                  <Button
                    variant="outline"
                    onClick={onClose}
                    className="flex-1"
                  >
                    Close
                  </Button>
                  <Button
                    onClick={onViewDetails}
                    className="flex-1 bg-gradient-to-r from-primary via-primary to-success
hover:from-primary/90 hover:v...
                  >
                    View Details
                  </Button>
                </div>
              </div>
            </Card>
          </motion.div>
        </>
      )}
    </AnimatePresence>
  );
};
```

File: src/components/NotificationBanner.tsx

Lines: 52

```
  1 | import { motion, AnimatePresence } from "framer-motion";
  2 | import NotificationsActiveIcon from "@mui/icons-material/NotificationsActive";
  3 | import CloseIcon from "@mui/icons-material/Close";
  4 |
  5 | interface NotificationBannerProps {
  6 |   show: boolean;
  7 |   onDismiss: () => void;
  8 |   onTap: () => void;
  9 | }
 10 |
 11 | export const NotificationBanner = ({ show, onDismiss, onTap }: NotificationBannerProps) => {
 12 |   return (
 13 |     <AnimatePresence>
 14 |       {show && (
 15 |         <motion.div
 16 |           initial={{ opacity: 0, y: -100 }}
 17 |           animate={{ opacity: 1, y: 0 }}
 18 |           exit={{ opacity: 0, y: -100 }}
 19 |           transition={{ type: "spring", stiffness: 300, damping: 30 }}
 20 |           className="fixed top-4 left-4 right-4 z-[100]"
 21 |         >
 22 |           <motion.div
 23 |             whileTap={{ scale: 0.98 }}
 24 |             onClick={onTap}
 25 |             className="bg-warning text-warning-foreground px-5 py-4 rounded-2xl shadow-
elevation-4 flex items-center gap...
 26 |           >
 27 |             <motion.div
 28 |               animate={{ scale: [1, 1.2, 1] }}
 29 |               transition={{ duration: 1, repeat: Infinity }}
 30 |             >
 31 |               <NotificationsActiveIcon style={{ fontSize: 28 }} />
 32 |             </motion.div>
 33 |             <div className="flex-1">
 34 |               <p className="font-bold text-sm">Active user nearby!</p>
 35 |               <p className="text-xs opacity-90">Tap to start your workout</p>
 36 |             </div>
 37 |             <button
 38 |               onClick={(e) => {
 39 |                 e.stopPropagation();
 40 |                 onDismiss();
 41 |               }}
 42 |               className="touch-target p-1 hover:bg-warning-foreground/10 rounded-full transition-
colors"
 43 |             >
 44 |               <CloseIcon style={{ fontSize: 20 }} />
 45 |             </button>
 46 |           </motion.div>
 47 |         </motion.div>
 48 |       )}
 49 |     </AnimatePresence>
 50 |   );
 51 | };
 52 |
```

File: src/components/NotificationBell.tsx

Lines: 127

```
  1 | /**
  2 |  * NotificationBell Component
  3 |  *
  4 |  * A reusable notification bell icon that:
  5 |  * - Turns YELLOW when there are unread notifications
  6 |  * - Shows a count badge with the number of unread notifications
  7 |  * - Provides consistent styling across all screens
  8 |  *
  9 |  * Usage:
 10 |  * <NotificationBell
 11 |  *   unreadCount={unreadCount}
 12 |  *   onClick={() => setShowNotificationDrawer(true)}
 13 |  * />
 14 |  */
 15 |
 16 | import { motion } from "framer-motion";
 17 | import NotificationsIcon from "@mui/icons-material/Notifications";
 18 |
 19 | interface NotificationBellProps {
 20 |   /** Number of unread notifications */
 21 |   unreadCount: number;
 22 |   /** Click handler to open notification drawer */
 23 |   onClick: () => void;
 24 |   /** Optional: Background style variant for different screen themes */
 25 |   variant?: "light" | "dark";
 26 |   /** Optional: Size of the bell icon */
 27 |   size?: "sm" | "md" | "lg";
 28 | }
 29 |
 30 | export const NotificationBell = ({
 31 |   unreadCount,
 32 |   onClick,
 33 |   variant = "light",
 34 |   size = "md",
 35 | }: NotificationBellProps) => {
 36 |   const hasNotifications = unreadCount > 0;
 37 |
 38 |   // Size configurations
 39 |   const sizeConfig = {
 40 |     sm: { button: 32, icon: 20, badge: "min-w-[16px] h-4 text-[9px] -top-0.5 -right-0.5" },
 41 |     md: { button: 40, icon: 24, badge: "min-w-[20px] h-5 text-[11px] -top-0.5 -right-0.5" },
 42 |     lg: { button: 48, icon: 28, badge: "min-w-[24px] h-6 text-xs -top-1 -right-1" },
 43 |   };
 44 |
 45 |   const config = sizeConfig[size];
 46 |
 47 |   // Colors based on variant (for different background contexts)
 48 |   const bgColors = {
 49 |     light: {
 50 |       hover: "hover:bg-muted",
 51 |       ring: "ring-offset-card",
 52 |       badgeBorder: "border-card",
 53 |       iconDefault: "text-foreground",
 54 |     },
 55 |     dark: {
 56 |       hover: "hover:bg-gray-700",
 57 |       ring: "ring-offset-gray-800",
 58 |       badgeBorder: "border-gray-800",
 59 |       iconDefault: "text-white",
 60 |     },
 61 |   };
 62 |
 63 |   const colors = bgColors[variant];
 64 |
 65 |   return (
 66 |     <motion.button
```

```
  67 |       whileTap={{ scale: 0.95 }}
  68 |       whileHover={{ scale: 1.05 }}
  69 |       onClick={onClick}
  70 |       className={`relative touch-target bg-transparent rounded-full transition-all
${colors.hover} ${
  71 |         hasNotifications
  72 |           ? `ring-2 ring-yellow-400 ring-offset-2 ${colors.ring}`
  73 |           : ''
  74 |       }`}
  75 |       style={{ width: config.button, height: config.button }}
  76 |       title={hasNotifications
  77 |         ? `${unreadCount} unread notification${unreadCount > 1 ? 's' : ''}`
  78 |         : "Notifications"
  79 |       }
  80 |       aria-label={hasNotifications
  81 |         ? `${unreadCount} unread notifications`
  82 |         : "Notifications"
  83 |       }
  84 |     >
  85 |       {/* Bell Icon - Yellow when there are notifications */}
  86 |       <NotificationsIcon
  87 |         style={{ fontSize: config.icon }}
  88 |         className={hasNotifications
  89 |           ? 'text-yellow-400 drop-shadow-[0_0_8px_rgba(250,204,21,0.5)]'
  90 |           : colors.iconDefault
  91 |         }
  92 |       />
  93 |
  94 |       {/* Notification Count Badge */}
  95 |       {hasNotifications && (
  96 |         <motion.span
  97 |           initial={{ scale: 0 }}
  98 |           animate={{ scale: 1 }}
  99 |           transition={{ type: "spring", stiffness: 500, damping: 25 }}
 100 |           className={`absolute ${config.badge} bg-yellow-500 text-gray-900 font-bold rounded-
full px-1.5 flex items-cent...
 101 |         >
 102 |           {unreadCount > 99 ? '99+' : unreadCount}
 103 |         </motion.span>
 104 |       )}
 105 |
 106 |       {/* Pulse animation for attention when there are notifications */}
 107 |       {hasNotifications && (
 108 |         <motion.span
 109 |           animate={{
 110 |             scale: [1, 1.4, 1],
 111 |             opacity: [0.7, 0, 0.7],
 112 |           }}
 113 |           transition={{
 114 |             duration: 2,
 115 |             repeat: Infinity,
 116 |             ease: "easeInOut"
 117 |           }}
 118 |           className="absolute inset-0 rounded-full bg-yellow-400/30"
 119 |         />
 120 |       )}
 121 |     </motion.button>
 122 |   );
 123 | };
 124 |
 125 | export default NotificationBell;
 126 |
 127 |
```

File: src/components/NotificationSystem.tsx

Lines: 379

```
  1 | import { useState, useEffect } from "react";
  2 | import { motion, AnimatePresence } from "framer-motion";
  3 | import { useNavigate } from "react-router-dom";
  4 | import Avatar from "@mui/material/Avatar";
  5 | import MailIcon from "@mui/icons-material/Mail";
  6 | import PersonAddIcon from "@mui/icons-material/PersonAdd";
  7 | import CloseIcon from "@mui/icons-material/Close";
  8 | import TouchAppIcon from "@mui/icons-material/TouchApp";
  9 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
 10 | import EmojiEventsIcon from "@mui/icons-material/EmojiEvents";
 11 | import ChatBubbleIcon from "@mui/icons-material/ChatBubble";
 12 | import {
 13 |   Notification as FirebaseNotification,
 14 |   NotificationType,
 15 |   listenToNotifications,
 16 |   markNotificationAsRead,
 17 |   markAllNotificationsAsRead,
 18 |   createNotification
 19 | } from "@/services/notificationService";
 20 |
 21 | export type { NotificationType };
 22 |
 23 | // Extended notification interface with backward compatibility
 24 | export interface Notification extends FirebaseNotification {
 25 |   // Backward compatibility fields (computed from fromUserId, fromUserName, fromUserAvatar)
 26 |   userId?: number | string; // Computed from fromUserId
 27 |   userName?: string; // Alias for fromUserName
 28 |   userAvatar?: string; // Alias for fromUserAvatar
 29 | }
 30 |
 31 | interface NotificationSystemProps {
 32 |   notifications: Notification[];
 33 |   onDismiss: (id: string) => void;
 34 |   onTap: (notification: Notification) => void;
 35 | }
 36 |
 37 | export const NotificationSystem = ({
 38 |   notifications,
 39 |   onDismiss,
 40 |   onTap,
 41 | }: NotificationSystemProps) => {
 42 |   const [visibleNotifications, setVisibleNotifications] = useState<Notification[]>([]);
 43 |
 44 |   useEffect(() => {
 45 |     // Show only the most recent unread notification
 46 |     const unreadNotifications = notifications.filter(n => !n.read);
 47 |     if (unreadNotifications.length > 0) {
 48 |       setVisibleNotifications([unreadNotifications[0]]);
 49 |
 50 |       // Auto-dismiss after 5 seconds
 51 |       const timer = setTimeout(() => {
 52 |         onDismiss(unreadNotifications[0].id);
 53 |       }, 5000);
 54 |
 55 |       return () => clearTimeout(timer);
 56 |     } else {
 57 |       setVisibleNotifications([]);
 58 |     }
 59 |   }, [notifications, onDismiss]);
 60 |
 61 |   return (
 62 |     <AnimatePresence>
 63 |       {visibleNotifications.map((notification) => (
 64 |         <motion.div
 65 |           key={notification.id}
 66 |           initial={{ y: -100, opacity: 0 }}
```

```
  67 |           animate={{ y: 0, opacity: 1 }}
  68 |           exit={{ y: -100, opacity: 0 }}
  69 |           transition={{ type: "spring", stiffness: 300, damping: 25 }}
  70 |           className="fixed top-4 left-4 right-4 z-[9999] max-w-md mx-auto"
  71 |         >
  72 |           <div
  73 |             onClick={() => onTap(notification)}
  74 |             className="bg-card/95 backdrop-blur-xl rounded-2xl shadow-elevation-4 border-2
border-border/50 overflow-hid...
  75 |               <div className="p-4 flex items-center gap-3">
  76 |               {/* Icon/Avatar */}
  77 |               <div className="flex-shrink-0 relative">
  78 |                 <Avatar
  79 |                   src={notification.fromUserAvatar || notification.userAvatar}
  80 |                   alt={notification.fromUserName || notification.userName || "User"}
  81 |                   sx={{ width: 48, height: 48 }}
  82 |                 />
  83 |                 <div className={`absolute -bottom-1 -right-1 w-6 h-6 rounded-full flex items-
center justify-center ${ notification.type === "message"
  84 |                     ? "bg-primary"
  85 |                     : notification.type === "message_request"
  86 |                     ? "bg-blue-500"
  87 |                     : notification.type === "friend_request"
  88 |                     ? "bg-warning"
  89 |                     : notification.type === "poke"
  90 |                     ? "bg-purple-500"
  91 |                     : notification.type === "workout_complete"
  92 |                     ? "bg-success"
  93 |                     : notification.type === "achievement"
  94 |                     ? "bg-warning"
  95 |                     : notification.type === "friend_accepted"
  96 |                     ? "bg-success"
  97 |                     : "bg-gray-500"
  98 |                 }`}>
  99 |                   {notification.type === "message" && (
 100 |                     <MailIcon style={{ fontSize: 14 }} className="text-white" />
 101 |                   )}
 102 |                   {notification.type === "message_request" && (
 103 |                     <ChatBubbleIcon style={{ fontSize: 14 }} className="text-white" />
 104 |                   )}
 105 |                   {notification.type === "friend_request" && (
 106 |                     <PersonAddIcon style={{ fontSize: 14 }} className="text-white" />
 107 |                   )}
 108 |                   {notification.type === "poke" && (
 109 |                     <TouchAppIcon style={{ fontSize: 14 }} className="text-white" />
 110 |                   )}
 111 |                   {notification.type === "friend_accepted" && (
 112 |                     <CheckCircleIcon style={{ fontSize: 14 }} className="text-white" />
 113 |                   )}
 114 |                   {notification.type === "workout_complete" && (
 115 |                     <CheckCircleIcon style={{ fontSize: 14 }} className="text-white" />
 116 |                   )}
 117 |                   {notification.type === "achievement" && (
 118 |                     <EmojiEventsIcon style={{ fontSize: 14 }} className="text-white" />
 119 |                   )}
 120 |                 </div>
 121 |               </div>
 122 |
 123 |               {/* Content */}
 124 |               <div className="flex-1 min-w-0">
 125 |                 <p className="font-semibold text-sm text-foreground truncate">
 126 |                   {notification.fromUserName || notification.userName || "User"}
 127 |                 </p>
 128 |                 <p className="text-sm text-muted-foreground truncate">
 129 |                   {notification.type === "message" && (
 130 |                     notification.message || "Sent you a message"
 131 |                   )}
 132 |                   {notification.type === "message_request" && (
 133 |                     notification.message || "wants to start a conversation with you"
 134 |                   )}
 135 |                   {notification.type === "friend_request" && "wants to add you as a friend"}
```

```
                  {notification.type === "poke" && (notification.message || "poked you! They're interested in matching").}
                  {notification.type === "friend_accepted" && "accepted your friend request"}
                  {notification.type === "workout_complete" && (
                    notification.message || "Workout completed successfully!"
                  )}
                  {notification.type === "achievement" && (
                    notification.message || "Congrats for a new achievement!"
                  )}
                </p>
              </div>

              {/* Close button */}
              <button
                onClick={(e) => {
                  e.stopPropagation();
                  onDismiss(notification.id);
                }}
                className="flex-shrink-0 p-2 hover:bg-accent rounded-full transition-colors"
              >
                <CloseIcon style={{ fontSize: 18 }} className="text-muted-foreground" />
              </button>
            </div>

            {/* Progress bar */}
            <motion.div
              initial={{ width: "100%" }}
              animate={{ width: "0%" }}
              transition={{ duration: 5, ease: "linear" }}
              className={`h-1 ${
                notification.type === "message"
                  ? "bg-primary"
                  : notification.type === "message_request"
                  ? "bg-blue-500"
                  : notification.type === "friend_request"
                  ? "bg-warning"
                  : notification.type === "poke"
                  ? "bg-purple-500"
                  : notification.type === "workout_complete"
                  ? "bg-success"
                  : notification.type === "achievement"
                  ? "bg-warning"
                  : notification.type === "friend_accepted"
                  ? "bg-success"
                  : "bg-gray-500"
              }`}
            />
          </div>
        </motion.div>
      ))}
    </AnimatePresence>
  );
};

// Badge Counter Component
interface BadgeCounterProps {
  count: number;
  variant?: "default" | "primary" | "warning" | "success";
  size?: "sm" | "md" | "lg";
}

export const BadgeCounter = ({
  count,
  variant = "default",
  size = "md"
}: BadgeCounterProps) => {
  if (count <= 0) return null;

  const sizeClasses = {
    sm: "w-4 h-4 text-[10px]",
    md: "w-5 h-5 text-xs",
    lg: "w-6 h-6 text-sm",
```

```
  };

  const variantClasses = {
    default: "bg-destructive text-destructive-foreground",
    primary: "bg-primary text-primary-foreground",
    warning: "bg-warning text-warning-foreground",
    success: "bg-success text-success-foreground",
  };

  return (
    <motion.div
      initial={{ scale: 0 }}
      animate={{ scale: 1 }}
      exit={{ scale: 0 }}
      transition={{ type: "spring", stiffness: 500, damping: 25 }}
      className={`
        ${sizeClasses[size]}
        ${variantClasses[variant]}
        rounded-full flex items-center justify-center font-bold
        border-2 border-background
      `}
    >
      {count > 99 ? "99+" : count}
    </motion.div>
  );
};

// Hook to manage notifications
export const useNotifications = (currentUserId?: string | null) => {
  const [notifications, setNotifications] = useState<Notification[]>([]);
  const navigate = useNavigate();

  // Listen to Firebase notifications
  useEffect(() => {
    if (!currentUserId) {
      setNotifications([]);
      return;
    }

    const unsubscribe = listenToNotifications(currentUserId, (firebaseNotifications) => {
      // Convert Firebase notifications to extended format with backward compatibility
      const extendedNotifications: Notification[] = firebaseNotifications.map((notif) => ({
        ...notif,
        // Backward compatibility fields
        userId: notif.fromUserId,
        userName: notif.fromUserName,
        userAvatar: notif.fromUserAvatar,
      }));
      setNotifications(extendedNotifications);
    });

    return () => unsubscribe();
  }, [currentUserId]);

  const addNotification = async (notification: Omit<Notification, "id" | "timestamp" | "read">)
=> {
    if (!currentUserId) return;

    try {
      // Convert to Firebase format
      const firebaseNotification = {
        type: notification.type,
        fromUserId: notification.fromUserId || (notification.userId as string),
        fromUserName: notification.fromUserName || notification.userName || "User",
        fromUserAvatar: notification.fromUserAvatar || notification.userAvatar || "",
        message: notification.message,
        workoutId: notification.workoutId,
        linkType: notification.linkType,
      };

      await createNotification(currentUserId, firebaseNotification);
    } catch (error) {
```

```
280 |       console.error("Error adding notification:", error);
281 |     }
282 |   };
283 |
284 |   const dismissNotification = async (id: string) => {
285 |     if (!currentUserId) return;
286 |
287 |     try {
288 |       await markNotificationAsRead(currentUserId, id);
289 |     } catch (error) {
290 |       console.error("Error dismissing notification:", error);
291 |     }
292 |   };
293 |
294 |   const markAllAsRead = async () => {
295 |     if (!currentUserId) return;
296 |
297 |     try {
298 |       await markAllNotificationsAsRead(currentUserId);
299 |     } catch (error) {
300 |       console.error("Error marking all as read:", error);
301 |     }
302 |   };
303 |
304 |   const handleNotificationTap = (notification: Notification) => {
305 |     dismissNotification(notification.id);
306 |
307 |     const userId = notification.fromUserId || notification.userId;
308 |     const userName = notification.fromUserName || notification.userName;
309 |     const userAvatar = notification.fromUserAvatar || notification.userAvatar;
310 |
311 |     if (notification.type === "message" || notification.type === "message_request") {
312 |       navigate("/chat", {
313 |         state: {
314 |           user: {
315 |             id: userId,
316 |             name: userName,
317 |             avatar: userAvatar,
318 |           },
319 |         },
320 |       });
321 |     } else if (notification.type === "friend_request") {
322 |       // Navigate to friends page with requests tab
323 |       navigate("/friends", { state: { tab: "requests" } });
324 |     } else if (notification.type === "poke") {
325 |       // Navigate to map, optionally with workout context if workoutId exists
326 |       if (notification.workoutId) {
327 |         navigate("/map", {
328 |           state: {
329 |             workoutId: notification.workoutId,
330 |             focusUserId: userId,
331 |           },
332 |         });
333 |       } else {
334 |         navigate("/map", {
335 |           state: {
336 |             focusUserId: userId,
337 |           },
338 |         });
339 |       }
340 |     } else if (notification.type === "friend_accepted") {
341 |       navigate("/friends", { state: { tab: "friends" } });
342 |     } else if (notification.type === "workout_complete") {
343 |       // Navigate to workout history page with specific workout ID
344 |       navigate("/workout-history", {
345 |         state: { workoutId: notification.workoutId }
346 |       });
347 |     } else if (notification.type === "achievement") {
348 |       // Navigate to workout history or profile for achievements
349 |       navigate("/workout-history");
350 |     } else if (notification.type === "username_change_required") {
```

```
351 |       // Navigate to edit profile page so user can update their username
352 |       navigate("/edit-profile");
353 |     }
354 |   };
355 |
356 |   const unreadCount = notifications.filter(n => !n.read).length;
357 |   const unreadMessageCount = notifications.filter(
358 |     n => !n.read && (n.type === "message" || n.type === "message_request")
359 |   ).length;
360 |   const unreadFriendRequestCount = notifications.filter(
361 |     n => !n.read && n.type === "friend_request"
362 |   ).length;
363 |   const unreadPokeCount = notifications.filter(
364 |     n => !n.read && n.type === "poke"
365 |   ).length;
366 |
367 |   return {
368 |     notifications,
369 |     addNotification,
370 |     dismissNotification,
371 |     markAllAsRead,
372 |     handleNotificationTap,
373 |     unreadCount,
374 |     unreadMessageCount,
375 |     unreadFriendRequestCount,
376 |     unreadPokeCount,
377 |   };
378 | };
379 |
```

File: src/components/NotificationTestButton.tsx

Lines: 27

```
   1 | import { motion } from "framer-motion";
   2 | import { useNotificationContext } from "@/contexts/NotificationContext";
   3 | import NotificationsIcon from "@mui/icons-material/Notifications";
   4 | import { toast } from "sonner";
   5 |
   6 | export const NotificationTestButton = () => {
   7 |   const { addNotification } = useNotificationContext();
   8 |
   9 |   // REMOVED: Mock users - notifications will come from real Firebase events
  10 |   // This button can be removed in production or kept for development testing
  11 |   const sendTestNotification = () => {
  12 |     toast.info("Real notifications will appear automatically from Firebase events");
  13 |   };
  14 |
  15 |   return (
  16 |     <motion.button
  17 |       whileTap={{ scale: 0.95 }}
  18 |       onClick={sendTestNotification}
  19 |       className="fixed bottom-24 left-4 z-20 touch-target bg-warning text-warning-foreground
rounded-full shadow-elevati...
  20 |       style={{ width: 56, height: 56 }}
  21 |       title="Send test notification"
  22 |     >
  23 |       <NotificationsIcon style={{ fontSize: 28 }} />
  24 |     </motion.button>
  25 |   );
  26 | };
  27 |
```

File: src/components/PokeModal.tsx

Lines: 182

```tsx
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import ChatIcon from "@mui/icons-material/Chat";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import CloseIcon from "@mui/icons-material/Close";
import { useNavigate } from "react-router-dom";

interface PokeModalProps {
  isOpen: boolean;
  onClose: () => void;
  userName: string;
  userAvatar?: string;
  userId: string;
  onAccept: () => void;
  onDismiss: () => void;
  onChat?: () => void;
  onAddFriend?: () => void;
}

/**
 * PokeModal Component
 *
 * This modal appears when a user receives a poke from someone.
 * A poke is a lightweight way to show interest in matching with someone.
 *
 * The user can:
 * - Accept the poke (acknowledge it and potentially start chatting)
 * - Dismiss the poke (ignore it)
 * - Chat with the user directly
 * - Add them as a friend
 */
export const PokeModal = ({
  isOpen,
  onClose,
  userName,
  userAvatar,
  userId,
  onAccept,
  onDismiss,
  onChat,
  onAddFriend
}: PokeModalProps) => {
  const navigate = useNavigate();

  const handleChat = () => {
    if (onChat) {
      onChat();
    } else {
      navigate("/chat", {
        state: {
          user: {
            id: userId,
            name: userName,
            avatar: userAvatar || ""
          }
        }
      });
    }
    onClose();
  };

  return (
    <AnimatePresence>
      {isOpen && (
```

```
  67 |         <>
  68 |           {/* Backdrop */}
  69 |           <motion.div
  70 |             initial={{ opacity: 0 }}
  71 |             animate={{ opacity: 1 }}
  72 |             exit={{ opacity: 0 }}
  73 |             onClick={onClose}
  74 |             className="fixed inset-0 bg-black/50 z-50"
  75 |           />
  76 |
  77 |           {/* Modal */}
  78 |           <motion.div
  79 |             initial={{ y: "100%" }}
  80 |             animate={{ y: 0 }}
  81 |             exit={{ y: "100%" }}
  82 |             transition={{ type: "spring", damping: 25, stiffness: 300 }}
  83 |             className="fixed bottom-0 left-0 right-0 z-50 bg-background rounded-t-3xl shadow-
elevation-4 p-6 max-h-[90vh...
  84 |           >
  85 |             {/* Header */}
  86 |             <div className="flex items-center justify-between mb-6">
  87 |               <div className="flex items-center gap-3">
  88 |                 <div className="w-12 h-12 rounded-full bg-purple-500/20 flex items-center
justify-center">
  89 |                   <TouchAppIcon className="text-purple-500" style={{ fontSize: 28 }} />
  90 |                 </div>
  91 |                 <div>
  92 |                   <h2 className="text-2xl font-bold text-foreground">You Got Poked!</h2>
  93 |                   <p className="text-sm text-muted-foreground">Someone is interested in
matching</p>
  94 |                 </div>
  95 |               </div>
  96 |               <Button
  97 |                 variant="ghost"
  98 |                 size="icon"
  99 |                 onClick={onClose}
 100 |                 className="rounded-full"
 101 |               >
 102 |                 <CloseIcon />
 103 |               </Button>
 104 |             </div>
 105 |
 106 |             {/* User Info */}
 107 |             <div className="flex flex-col items-center gap-4 mb-6">
 108 |               <Avatar className="h-24 w-24 border-4 border-purple-500/30">
 109 |                 <AvatarImage src={userAvatar} alt={userName} />
 110 |                 <AvatarFallback className="text-2xl">
 111 |                   {userName.charAt(0).toUpperCase()}
 112 |                 </AvatarFallback>
 113 |               </Avatar>
 114 |               <div className="text-center">
 115 |                 <h3 className="text-xl font-bold text-foreground">{userName}</h3>
 116 |                 <p className="text-sm text-muted-foreground mt-1">
 117 |                   poked you! They're interested in matching
 118 |                 </p>
 119 |               </div>
 120 |             </div>
 121 |
 122 |             {/* Action Buttons */}
 123 |             <div className="space-y-3">
 124 |               {/* Accept Poke Button */}
 125 |               <Button
 126 |                 onClick={() => {
 127 |                   onAccept();
 128 |                   onClose();
 129 |                 }}
 130 |                 className="w-full h-14 text-base font-semibold bg-purple-500 hover:bg-purple-600
text-white"
 131 |               >
 132 |                 <TouchAppIcon className="mr-2" style={{ fontSize: 20 }} />
 133 |                 Accept Poke
 134 |               </Button>
 135 |
 136 |               {/* Chat Button */}
 137 |               {onChat && (
```

```
                <Button
                  onClick={handleChat}
                  variant="outline"
                  className="w-full h-14 text-base font-semibold"
                >
                  <ChatIcon className="mr-2" style={{ fontSize: 20 }} />
                  Start Chat
                </Button>
              )}

              {/* Add Friend Button */}
              {onAddFriend && (
                <Button
                  onClick={() => {
                    onAddFriend();
                    onClose();
                  }}
                  variant="outline"
                  className="w-full h-14 text-base font-semibold"
                >
                  <PersonAddIcon className="mr-2" style={{ fontSize: 20 }} />
                  Add Friend
                </Button>
              )}

              {/* Dismiss Button */}
              <Button
                onClick={() => {
                  onDismiss();
                  onClose();
                }}
                variant="ghost"
                className="w-full h-12 text-base text-muted-foreground hover:text-foreground"
              >
                Dismiss
              </Button>
            </div>
          </motion.div>
        </>
      )}
    </AnimatePresence>
  );
};

```

File: src/components/PokeNotificationModal.tsx

Lines: 153

```
import { motion, AnimatePresence } from "framer-motion";
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import CloseIcon from "@mui/icons-material/Close";
import { Button } from "@/components/ui/button";

interface PokeNotificationModalProps {
  isOpen: boolean;
  userName: string;
  userAvatar?: string;
  onClose: () => void;
  onViewProfile: () => void;
  onChat: () => void;
}

/**
 * PokeNotificationModal Component
 *
 * A prominent notification that appears in the center of the screen
 * when someone pokes you. Highlights the user who poked you.
 */
export const PokeNotificationModal = ({
  isOpen,
  userName,
  userAvatar,
  onClose,
  onViewProfile,
  onChat
}: PokeNotificationModalProps) => {
  return (
    <AnimatePresence>
      {isOpen && (
        <>
          {/* Backdrop */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            className="fixed inset-0 bg-black/60 backdrop-blur-sm z-50"
            onClick={onClose}
          />

          {/* Notification Card */}
          <motion.div
            initial={{ opacity: 0, scale: 0.8, y: 50 }}
            animate={{ opacity: 1, scale: 1, y: 0 }}
            exit={{ opacity: 0, scale: 0.8, y: 50 }}
            transition={{ type: "spring", stiffness: 300, damping: 25 }}
            className="fixed top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2 z-50 w-[90%] max-
w-md"
            onClick={(e) => e.stopPropagation()}
          >
            <div className="bg-card rounded-3xl shadow-elevation-4 border-2 border-purple-500/50
overflow-hidden">
              {/* Pulsing border effect */}
              <motion.div
                animate={{
                  boxShadow: [
                    "0 0 0 0 rgba(168, 85, 247, 0.7)",
                    "0 0 0 10px rgba(168, 85, 247, 0)",
                    "0 0 0 0 rgba(168, 85, 247, 0)"
                  ]
                }}
                transition={{
                  duration: 2,
                  repeat: Infinity,
                  ease: "easeInOut"
                }}
```

```
                  className="absolute inset-0 rounded-3xl pointer-events-none"
                />

                <div className="p-6 space-y-4 relative">
                  {/* Close button */}
                  <button
                    onClick={onClose}
                    className="absolute top-4 right-4 p-2 hover:bg-muted rounded-full transition-colors z-10"
                  >
                    <CloseIcon style={{ fontSize: 20 }} className="text-muted-foreground" />
                  </button>

                  {/* Icon and Title */}
                  <div className="flex flex-col items-center gap-3 pt-2">
                    <motion.div
                      animate={{
                        scale: [1, 1.1, 1],
                        rotate: [0, 5, -5, 0]
                      }}
                      transition={{
                        duration: 1.5,
                        repeat: Infinity,
                        ease: "easeInOut"
                      }}
                      className="w-16 h-16 rounded-full bg-purple-500/20 flex items-center justify-center"
                    >
                      <TouchAppIcon className="text-purple-500" style={{ fontSize: 40 }} />
                    </motion.div>

                    <div className="text-center">
                      <h2 className="text-2xl font-bold text-foreground mb-1">
                        You Got Poked!
                      </h2>
                      <p className="text-sm text-muted-foreground">
                        Someone is interested in matching
                      </p>
                    </div>
                  </div>

                  {/* User Info */}
                  <div className="flex flex-col items-center gap-3 py-2">
                    <Avatar className="h-20 w-20 border-4 border-purple-500/50 ring-4 ring-purple-500/20">
                      <AvatarImage src={userAvatar} alt={userName} />
                      <AvatarFallback className="text-2xl bg-purple-500/20 text-purple-500">
                        {userName.charAt(0).toUpperCase()}
                      </AvatarFallback>
                    </Avatar>
                    <div className="text-center">
                      <h3 className="text-xl font-bold text-foreground">{userName}</h3>
                      <p className="text-sm text-muted-foreground mt-1">
                        poked you! They're nearby
                      </p>
                    </div>
                  </div>

                  {/* Action Buttons */}
                  <div className="flex gap-3 pt-2">
                    <Button
                      onClick={() => {
                        onViewProfile();
                        onClose();
                      }}
                      variant="outline"
                      className="flex-1 h-12"
                    >
                      View Profile
                    </Button>
                    <Button
                      onClick={() => {
                        onChat();
                        onClose();
```

```
                    }}
                    className="flex-1 h-12 bg-purple-500 hover:bg-purple-600 text-white"
                  >
                    Chat
                  </Button>
                </div>
              </div>
            </div>
          </motion.div>
        </>
      )}
    </AnimatePresence>
  );
};

```

File: src/components/QuickCheckInModal.tsx

Lines: 339

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import CloseIcon from "@mui/icons-material/Close";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import SearchIcon from "@mui/icons-material/Search";
import CheckBoxIcon from "@mui/icons-material/CheckBox";
import CheckBoxOutlineBlankIcon from "@mui/icons-material/CheckBoxOutlineBlank";
import { toast } from "sonner";
import { getAllVenuesSync, searchVenues, Venue } from "@/services/venueService";
import { useAuth } from "@/hooks/useAuth";
import { saveUserVenuePreferences, getUserVenuePreferences, Activity } from "@/services/
venuePreferenceService";
import { VenueRequestModal } from "./VenueRequestModal";

interface QuickCheckInModalProps {
  onClose: () => void;
}

export const QuickCheckInModal = ({ onClose }: QuickCheckInModalProps) => {
  const { user } = useAuth();
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedActivities, setSelectedActivities] = useState<Activity[]>([]);
  const [selectedVenues, setSelectedVenues] = useState<string[]>([]);
  const [venues, setVenues] = useState<Venue[]>(getAllVenuesSync());
  const [loading, setLoading] = useState(false);
  const [showRequestModal, setShowRequestModal] = useState(false);

  // Load existing preferences on mount
  useEffect(() => {
    const loadPreferences = async () => {
      if (!user?.uid) return;

      try {
        const preferences = await getUserVenuePreferences(user.uid);
        if (preferences) {
          setSelectedActivities(preferences.activities || []);
          setSelectedVenues(preferences.venues || []);
        }
      } catch (error) {
        console.error("Error loading preferences:", error);
      }
    };

    loadPreferences();
  }, [user?.uid]);

  useEffect(() => {
    if (searchQuery.trim()) {
      setVenues(searchVenues(searchQuery));
    } else {
      setVenues(getAllVenuesSync());
    }
  }, [searchQuery]);

  const handleToggleActivity = (activity: Activity) => {
    setSelectedActivities(prev => {
      if (prev.includes(activity)) {
        return prev.filter(a => a !== activity);
      } else {
        return [...prev, activity];
      }
```

```
    });
  };

  const handleToggleVenue = (venueId: string) => {
    setSelectedVenues(prev => {
      if (prev.includes(venueId)) {
        return prev.filter(id => id !== venueId);
      } else {
        return [...prev, venueId];
      }
    });
  };

  const handleSetLocation = async () => {
    if (selectedVenues.length === 0) {
      toast.error("Please select at least one venue");
      return;
    }

    if (selectedActivities.length === 0) {
      toast.error("Please select at least one activity");
      return;
    }

    if (!user?.uid) {
      toast.error("Please log in to set location");
      return;
    }

    setLoading(true);
    try {
      await saveUserVenuePreferences(user.uid, selectedVenues, selectedActivities);
      toast.success("Location preferences saved!");
      onClose();
    } catch (error: any) {
      toast.error(error.message || "Failed to save preferences");
    } finally {
      setLoading(false);
    }
  };

  const getActivityIcon = (activity: Activity) => {
    switch (activity) {
      case "running":
        return <DirectionsRunIcon className="text-success" style={{ fontSize: 24 }} />;
      case "cycling":
        return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 24 }} />;
      case "walking":
        return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 24 }} />;
    }
  };

  const activities: { id: Activity; label: string; icon: typeof DirectionsRunIcon; color:
string }[] = [
    { id: "running", label: "Running", icon: DirectionsRunIcon, color: "success" },
    { id: "cycling", label: "Cycling", icon: DirectionsBikeIcon, color: "primary" },
    { id: "walking", label: "Walking", icon: DirectionsWalkIcon, color: "warning" },
  ];

  return (
    <AnimatePresence>
      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        exit={{ opacity: 0 }}
        onClick={onClose}
        className="fixed inset-0 bg-background/80 backdrop-blur-sm z-50 flex items-center
justify-center p-4"
      >
        <motion.div
          initial={{ scale: 0.95, opacity: 0, y: 20 }}
          animate={{ scale: 1, opacity: 1, y: 0 }}
          exit={{ scale: 0.95, opacity: 0, y: 20 }}
```

```
          transition={{ type: "spring", stiffness: 300, damping: 30 }}
          onClick={(e) => e.stopPropagation()}
          className="w-full max-w-2xl max-h-[90vh] flex flex-col"
        >
          <Card className="flex flex-col overflow-hidden shadow-elevation-4 border-2 border-border/50 h-full">
            {/* Header - Fixed */}
            <div className="relative bg-gradient-to-br from-primary/10 via-success/5 to-warning/10 p-6 border-b border-b...
              <button
                onClick={onClose}
                className="absolute top-4 right-4 p-2 bg-background/80 backdrop-blur-sm hover:bg-secondary rounded-full ...
              >
                <CloseIcon fontSize="small" />
              </button>

              <div className="flex items-center gap-3 pr-10">
                <div>
                  <h2 className="text-2xl font-bold">Add venues</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    Select venues and activities to display on other users. (Only your profile picture and user is visib...</p>
                </div>
              </div>
            </div>

            {/* Content - Scrollable */}
            <div className="flex-1 overflow-y-auto p-6 space-y-6">
              {/* Activity Selection */}
              <div className="space-y-3">
                <label className="text-base font-semibold">Activities</label>
                <div className="grid grid-cols-3 gap-3">
                  {activities.map((activity) => {
                    const Icon = activity.icon;
                    const isSelected = selectedActivities.includes(activity.id);
                    return (
                      <motion.button
                        key={activity.id}
                        whileTap={{ scale: 0.95 }}
                        onClick={() => handleToggleActivity(activity.id)}
                        className={`
                          flex flex-col items-center justify-center p-4 rounded-xl border-2 transition-all duration-300 ...
                          ${
                            isSelected
                              ? activity.color === "success"
                                ? "border-success bg-success/10 shadow-elevation-2"
                                : activity.color === "primary"
                                ? "border-primary bg-primary/10 shadow-elevation-2"
                                : "border-warning bg-warning/10 shadow-elevation-2"
                              : "border-border bg-card hover:bg-secondary"
                          }
                        `}
                      >
                        {isSelected && (
                          <div className="absolute top-2 right-2">
                            <CheckBoxIcon
                              className={
                                activity.color === "success"
                                  ? "text-success"
                                  : activity.color === "primary"
                                  ? "text-primary"
                                  : "text-warning"
                              }
                              style={{ fontSize: 20 }}
                            />
                          </div>
                        )}
                        <Icon
                          className={
                            isSelected
```

```
                              ? activity.color === "success"
                                ? "text-success"
                                : activity.color === "primary"
                                ? "text-primary"
                                : "text-warning"
                              : "text-muted-foreground"
                          }
                          style={{ fontSize: 32 }}
                        />
                        <span
                          className={`text-sm mt-2 font-semibold ${
                            isSelected ? "text-foreground" : "text-muted-foreground"
                          }`}
                        >
                          {activity.label}
                        </span>
                      </motion.button>
                    );
                  })}
                </div>
              </div>

              {/* Venue Search */}
              <div className="space-y-3">
                <div className="flex items-center justify-between">
                  <label className="text-base font-semibold">Select Venue</label>
                  <button
                    type="button"
                    onClick={() => setShowRequestModal(true)}
                    className="text-sm text-primary hover:text-primary/80 underline"
                  >
                    Can't find your venue?
                  </button>
                </div>
                <div className="relative">
                  <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style...<Input
                    placeholder="Search venues..."
                    value={searchQuery}
                    onChange={(e) => setSearchQuery(e.target.value)}
                    className="pl-10 h-12"
                  />
                </div>
              </div>

              {/* Venue List - No nested scrolling */}
              <div className="space-y-2">
                {venues.length === 0 ? (
                  <div className="text-center py-8 text-muted-foreground">
                    <p>No venues found</p>
                  </div>
                ) : (
                  venues.map((venue) => {
                    const isSelected = selectedVenues.includes(venue.id);

                    return (
                      <motion.button
                        key={venue.id}
                        whileTap={{ scale: 0.98 }}
                        onClick={() => handleToggleVenue(venue.id)}
                        className={`
                          w-full text-left p-4 rounded-lg border-2 transition-all duration-200
                          ${
                            isSelected
                              ? "border-primary bg-primary/10"
                              : "border-border bg-card hover:bg-secondary hover:border-
primary/50"
                          }
                        `}
                      >
                        <div className="flex items-start gap-3">
                          {isSelected ? (
```

```
                            <CheckBoxIcon className="text-primary flex-shrink-0 mt-0.5"
style={{ fontSize: 20 }} />
                          ) : (
                            <CheckBoxOutlineBlankIcon className="text-muted-foreground flex-
shrink-0 mt-0.5" style={{ fo...
                          )}
                          <div className="flex-1 min-w-0">
                            <div className="flex items-center gap-2 mb-1">
                              <LocationOnIcon
                                className={isSelected ? "text-primary" : "text-muted-foreground"}
                                style={{ fontSize: 18 }}
                              />
                              <h3 className="font-semibold text-base truncate">{venue.name}</h3>
                            </div>
                            <p className="text-sm text-muted-foreground line-
clamp-2">{venue.description}</p>
                            <p className="text-xs text-muted-foreground mt-1">{venue.city}</p>
                          </div>
                        </div>
                      </motion.button>
                    );
                  })
                )}
              </div>
            </div>

            {/* Action Buttons - Fixed */}
            <div className="flex gap-3 p-6 pt-4 border-t border-border flex-shrink-0">
              <Button
                type="button"
                variant="outline"
                onClick={onClose}
                className="flex-1 h-12"
              >
                Cancel
              </Button>
              <Button
                onClick={handleSetLocation}
                disabled={selectedVenues.length === 0 || selectedActivities.length === 0 ||
loading}
                className="flex-1 h-12 font-semibold"
              >
                {loading ? "Saving..." : "Set location"}
              </Button>
            </div>
          </Card>
        </motion.div>
      </motion.div>

      {/* Venue Request Modal */}
      {showRequestModal && (
        <VenueRequestModal
          onClose={() => setShowRequestModal(false)}
          onSuccess={() => {
            setShowRequestModal(false);
            toast.success("Request submitted successfully!");
          }}
        />
      )}
    </AnimatePresence>
  );
};
```

File: src/components/ReportUserModal.tsx

Lines: 121

```
  1 | import { useState } from "react";
  2 | import { Button } from "@/components/ui/button";
  3 | import {
  4 |   Dialog,
  5 |   DialogContent,
  6 |   DialogHeader,
  7 |   DialogTitle,
  8 |   DialogDescription,
  9 | } from "@/components/ui/dialog";
 10 | import { Textarea } from "@/components/ui/textarea";
 11 | import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group";
 12 | import { Label } from "@/components/ui/label";
 13 |
 14 | interface ReportUserModalProps {
 15 |   open: boolean;
 16 |   onOpenChange: (open: boolean) => void;
 17 |   userName: string;
 18 |   onReport: (reason: string, details?: string) => Promise<void>;
 19 | }
 20 |
 21 | const REPORT_REASONS = [
 22 |   { value: "harassment", label: "Harassment or bullying" },
 23 |   { value: "spam", label: "Spam or scams" },
 24 |   { value: "inappropriate", label: "Inappropriate content" },
 25 |   { value: "fake", label: "Fake profile" },
 26 |   { value: "other", label: "Other" },
 27 | ];
 28 |
 29 | export const ReportUserModal = ({
 30 |   open,
 31 |   onOpenChange,
 32 |   userName,
 33 |   onReport,
 34 | }: ReportUserModalProps) => {
 35 |   const [selectedReason, setSelectedReason] = useState<string>("");
 36 |   const [details, setDetails] = useState<string>("");
 37 |   const [isSubmitting, setIsSubmitting] = useState(false);
 38 |
 39 |   const handleSubmit = async () => {
 40 |     if (!selectedReason) return;
 41 |
 42 |     setIsSubmitting(true);
 43 |     try {
 44 |       await onReport(selectedReason, details.trim() || undefined);
 45 |       // Reset form
 46 |       setSelectedReason("");
 47 |       setDetails("");
 48 |       onOpenChange(false);
 49 |     } catch (error) {
 50 |       console.error("Error submitting report:", error);
 51 |     } finally {
 52 |       setIsSubmitting(false);
 53 |     }
 54 |   };
 55 |
 56 |   return (
 57 |     <Dialog open={open} onOpenChange={onOpenChange}>
 58 |       <DialogContent className="sm:max-w-md">
 59 |         <DialogHeader>
 60 |           <DialogTitle>Report {userName}</DialogTitle>
 61 |           <DialogDescription>
 62 |             Help us keep PaceMatch safe. Your report will be reviewed by our team.
 63 |           </DialogDescription>
 64 |         </DialogHeader>
 65 |
 66 |         <div className="space-y-6 py-4">
```

```
          <div className="space-y-3">
            <Label className="text-base font-medium">What's the reason for reporting?</Label>
            <RadioGroup value={selectedReason} onValueChange={setSelectedReason}>
              {REPORT_REASONS.map((reason) => (
                <div key={reason.value} className="flex items-center space-x-2">
                  <RadioGroupItem value={reason.value} id={reason.value} />
                  <Label htmlFor={reason.value} className="cursor-pointer font-normal">
                    {reason.label}
                  </Label>
                </div>
              ))}
            </RadioGroup>
          </div>

          {selectedReason && (
            <div className="space-y-2">
              <Label htmlFor="details">Additional details (optional)</Label>
              <Textarea
                id="details"
                placeholder="Please provide any additional information that might help us understand the issue..."
                value={details}
                onChange={(e) => setDetails(e.target.value)}
                rows={4}
                maxLength={500}
              />
              <p className="text-xs text-muted-foreground text-right">
                {details.length}/500
              </p>
            </div>
          )}

          <div className="flex gap-3">
            <Button
              variant="outline"
              className="flex-1"
              onClick={() => onOpenChange(false)}
              disabled={isSubmitting}
            >
              Cancel
            </Button>
            <Button
              className="flex-1"
              onClick={handleSubmit}
              disabled={!selectedReason || isSubmitting}
            >
              {isSubmitting ? "Submitting..." : "Submit Report"}
            </Button>
          </div>
        </div>
      </DialogContent>
    </Dialog>
  );
};

```

File: src/components/UserDetailDrawer.tsx

Lines: 489

```
// User Detail Drawer for Admin - Shows comprehensive user data
import { useState, useEffect } from "react";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
} from "@/components/ui/sheet";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { Card } from "@/components/ui/card";
import { getUserData } from "@/services/authService";
import { getUserWorkouts } from "@/services/workoutService";
import { getUserFriends } from "@/services/friendService";
import { getUserConversations } from "@/services/messageService";
import { removeUserPhoto, removeUserPhotoURL, logAdminAction } from "@/services/adminService";
import { useAuth } from "@/hooks/useAuth";
import { toast } from "sonner";
import PersonIcon from "@mui/icons-material/Person";
import EmailIcon from "@mui/icons-material/Email";
import CalendarTodayIcon from "@mui/icons-material/CalendarToday";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import PeopleIcon from "@mui/icons-material/People";
import ChatIcon from "@mui/icons-material/Chat";
import CodeIcon from "@mui/icons-material/Code";
import DownloadIcon from "@mui/icons-material/Download";
import PhotoIcon from "@mui/icons-material/Photo";
import DeleteIcon from "@mui/icons-material/Delete";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";

interface UserDetailDrawerProps {
  userId: string | null;
  open: boolean;
  onOpenChange: (open: boolean) => void;
}

const UserDetailDrawer = ({ userId, open, onOpenChange }: UserDetailDrawerProps) => {
  const { user: currentAdmin } = useAuth();
  const [userData, setUserData] = useState<any>(null);
  const [workouts, setWorkouts] = useState<any[]>([]);
  const [friends, setFriends] = useState<any[]>([]);
  const [conversations, setConversations] = useState<any[]>([]);
  const [loading, setLoading] = useState(false);
  const [activeTab, setActiveTab] = useState("profile");
  const [deletePhotoDialog, setDeletePhotoDialog] = useState<{
    open: boolean;
    type: "photoURL" | "photo";
    index?: number;
  }>({ open: false, type: "photo" });

  useEffect(() => {
    if (open && userId) {
      loadUserData();
    } else {
```

```
      // Reset when closed
      setUserData(null);
      setWorkouts([]);
      setFriends([]);
      setConversations([]);
      setActiveTab("profile");
    }
  }, [open, userId]);

  const loadUserData = async () => {
    if (!userId) return;

    try {
      setLoading(true);

      // Load user profile
      const profile = await getUserData(userId);
      setUserData(profile);

      // Load workouts
      try {
        const userWorkouts = await getUserWorkouts(userId);
        setWorkouts(userWorkouts || []);
      } catch (error) {
        console.error("Error loading workouts:", error);
      }

      // Load friends
      try {
        const userFriends = await getUserFriends(userId);
        setFriends(userFriends || []);
      } catch (error) {
        console.error("Error loading friends:", error);
      }

      // Load conversations
      try {
        const userConvs = await getUserConversations(userId);
        setConversations(userConvs || []);
      } catch (error) {
        console.error("Error loading conversations:", error);
      }
    } catch (error) {
      console.error("Error loading user data:", error);
      toast.error("Failed to load user data");
    } finally {
      setLoading(false);
    }
  };

  const handleExportJSON = () => {
    if (!userData) return;

    const exportData = {
      profile: userData,
      workouts,
      friends,
      conversations,
      exportedAt: new Date().toISOString()
    };

    const blob = new Blob([JSON.stringify(exportData, null, 2)], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `user-${userId}-${Date.now()}.json`;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);

```

```
138 |     toast.success("User data exported");
139 |   };
140 |
141 |   const handleRemovePhoto = async () => {
142 |     if (!userId) return;
143 |
144 |     try {
145 |       if (deletePhotoDialog.type === "photoURL") {
146 |         await removeUserPhotoURL(userId);
147 |         if (currentAdmin?.email) {
148 |           await logAdminAction(currentAdmin.email, "remove_user_photo", {
149 |             userId,
150 |             userName: userData?.name,
151 |             photoType: "photoURL"
152 |           });
153 |         }
154 |         toast.success("Profile photo removed");
155 |       } else if (deletePhotoDialog.type === "photo" && deletePhotoDialog.index !== undefined) {
156 |         await removeUserPhoto(userId, deletePhotoDialog.index);
157 |         if (currentAdmin?.email) {
158 |           await logAdminAction(currentAdmin.email, "remove_user_photo", {
159 |             userId,
160 |             userName: userData?.name,
161 |             photoType: "uploaded_photo",
162 |             photoIndex: deletePhotoDialog.index
163 |           });
164 |         }
165 |         toast.success("Photo removed");
166 |       }
167 |
168 |       // Reload user data to reflect changes
169 |       await loadUserData();
170 |       setDeletePhotoDialog({ open: false, type: "photo" });
171 |     } catch (error) {
172 |       console.error("Error removing photo:", error);
173 |       toast.error("Failed to remove photo");
174 |     }
175 |   };
176 |
177 |   const formatDate = (timestamp?: number) => {
178 |     if (!timestamp) return "N/A";
179 |     return new Date(timestamp).toLocaleString();
180 |   };
181 |
182 |   if (!userId) return null;
183 |
184 |   return (
185 |     <Sheet open={open} onOpenChange={onOpenChange}>
186 |       <SheetContent className="w-full sm:max-w-2xl overflow-y-auto">
187 |         {loading ? (
188 |           <div className="flex items-center justify-center h-64">
189 |             <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
190 |           </div>
191 |         ) : userData ? (
192 |           <>
193 |             <SheetHeader>
194 |               <div className="flex items-center justify-between">
195 |                 <div className="flex items-center gap-4">
196 |                   <Avatar className="h-16 w-16">
197 |                     <AvatarImage src={userData.photoURL} />
198 |                     <AvatarFallback>
199 |                       {userData.name?.[0]?.toUpperCase() || userData.email?.[0]?.toUpperCase()
|200 |                       ||"U"}
201 |                     </AvatarFallback>
202 |                   </Avatar>
203 |                   <div>
204 |                     <SheetTitle>{userData.name || "No name"}</SheetTitle>
205 |                     <SheetDescription>{userId}</SheetDescription>
206 |                   </div>
207 |                 </div>
208 |                 <Button variant="outline" size="sm" onClick={handleExportJSON}>
209 |                   <DownloadIcon className="mr-2" style={{ fontSize: 16 }} />
```

```
                    Export JSON
                  </Button>
                </div>
              </SheetHeader>

              <Tabs value={activeTab} onValueChange={setActiveTab} className="mt-6">
                <TabsList className="grid w-full grid-cols-5">
                  <TabsTrigger value="profile">Profile</TabsTrigger>
                  <TabsTrigger value="photos">Photos</TabsTrigger>
                  <TabsTrigger value="workouts">Workouts ({workouts.length})</TabsTrigger>
                  <TabsTrigger value="friends">Friends ({friends.length})</TabsTrigger>
                  <TabsTrigger value="messages">Messages ({conversations.length})</TabsTrigger>
                </TabsList>

                <TabsContent value="profile" className="space-y-4 mt-4">
                  <Card className="p-4">
                    <h3 className="font-semibold mb-4">Basic Information</h3>
                    <div className="space-y-3">
                      <div className="flex items-center gap-2">
                        <EmailIcon style={{ fontSize: 18 }} />
                        <span className="text-sm text-muted-foreground">Email:</span>
                        <span className="text-sm">{userData.email || "N/A"}</span>
                      </div>
                      <div className="flex items-center gap-2">
                        <PersonIcon style={{ fontSize: 18 }} />
                        <span className="text-sm text-muted-foreground">Activity:</span>
                        <Badge>{userData.activity || "N/A"}</Badge>
                      </div>
                      <div className="flex items-center gap-2">
                        <CalendarTodayIcon style={{ fontSize: 18 }} />
                        <span className="text-sm text-muted-foreground">Joined:</span>
                        <span className="text-sm">{formatDate(userData.createdAt)}</span>
                      </div>
                      <div className="flex items-center gap-2">
                        <CalendarTodayIcon style={{ fontSize: 18 }} />
                        <span className="text-sm text-muted-foreground">Last Activity:</span>
                        <span className="text-sm">{formatDate(userData.timestamp)}</span>
                      </div>
                      {userData.status && (
                        <div className="flex items-center gap-2">
                          <span className="text-sm text-muted-foreground">Status:</span>
                          <Badge variant={userData.status === "banned" ? "destructive" :
userData.status === "suspended" ? {userData.status}
                          </Badge>
                        </div>
                      )}
                    </div>
                  </Card>

                  <Card className="p-4">
                    <h3 className="font-semibold mb-4">Location Data</h3>
                    <div className="space-y-2 text-sm">
                      <div>
                        <span className="text-muted-foreground">Latitude:</span> {userData.lat ||
"N/A"}
                      </div>
                      <div>
                        <span className="text-muted-foreground">Longitude:</span> {userData.lng ||
"N/A"}
                      </div>
                      <div>
                        <span className="text-muted-foreground">Visible:</span>
{userData.visible ? "Yes" : "No"}
                      </div>
                      <div>
                        <span className="text-muted-foreground">General Location:</span>
{userData.generalLocation || "N/A...
                      </div>
                    </div>
                  </Card>

                  <Card className="p-4">
                    <h3 className="font-semibold mb-4">Preferences</h3>
                    <div className="space-y-2 text-sm">
                      <div>
```

```
                      <span className="text-muted-foreground">Fitness Level:</span>
{userData.fitnessLevel || "N/A"}
                    </div>
                    <div>
                      <span className="text-muted-foreground">Search Filter:</span>
{userData.searchFilter || "N/A"}
                    </div>
                    <div>
                      <span className="text-muted-foreground">Radius Preference:</span>
{userData.radiusPreference || "N...
                    </div>
                  </div>
                </Card>

                <Card className="p-4">
                  <h3 className="font-semibold mb-4">Raw JSON Data</h3>
                  <pre className="text-xs bg-muted p-3 rounded overflow-auto max-h-64">
                    {JSON.stringify(userData, null, 2)}
                  </pre>
                </Card>
              </TabsContent>

              <TabsContent value="photos" className="space-y-4 mt-4">
                {/* Profile Photo (photoURL) */}
                {userData.photoURL && (
                  <Card className="p-4">
                    <div className="flex items-center justify-between mb-3">
                      <h3 className="font-semibold">Profile Photo (Auth)</h3>
                      <Button
                        variant="destructive"
                        size="sm"
                        onClick={() => setDeletePhotoDialog({ open: true, type: "photoURL" })}
                      >
                        <DeleteIcon className="mr-1" style={{ fontSize: 16 }} />
                        Remove
                      </Button>
                    </div>
                    <div className="flex justify-center">
                      <img
                        src={userData.photoURL}
                        alt="Profile"
                        className="max-w-full h-auto max-h-64 rounded-lg border"
                        onError={(e) => {
                          (e.target as HTMLImageElement).style.display = 'none';
                        }}
                      />
                    </div>
                  </Card>
                )}

                {/* Uploaded Photos */}
                <Card className="p-4">
                  <div className="flex items-center justify-between mb-3">
                    <h3 className="font-semibold">
                      Uploaded Photos ({userData.photos?.length || 0})
                    </h3>
                  </div>
                  {!userData.photos || userData.photos.length === 0 ? (
                    <p className="text-muted-foreground text-center py-8">No uploaded photos</p>
                  ) : (
                    <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
                      {userData.photos.map((photo: string, index: number) => (
                        <div key={index} className="relative group">
                          <img
                            src={photo}
                            alt={`Photo ${index + 1}`}
                            className="w-full h-auto rounded-lg border"
                            onError={(e) => {
                              (e.target as HTMLImageElement).style.display = 'none';
                            }}
                          />
                          <Button
                            variant="destructive"
                            size="sm"
```

```
                            className="absolute top-2 right-2 opacity-0 group-hover:opacity-100 transition-opacity"
                            onClick={() => setDeletePhotoDialog({ open: true, type: "photo", index })}
                          >
                            <DeleteIcon style={{ fontSize: 16 }} />
                          </Button>
                          <div className="absolute bottom-2 left-2 bg-black/50 text-white text-xs px-2 py-1 rounded">
                            Photo {index + 1}
                          </div>
                        </div>
                      ))}
                    </div>
                  )}
                </Card>
              </TabsContent>

              <TabsContent value="workouts" className="space-y-2 mt-4">
                {workouts.length === 0 ? (
                  <p className="text-muted-foreground text-center py-8">No workouts found</p>
                ) : (
                  workouts.map((workout, index) => (
                    <Card key={index} className="p-4">
                      <div className="flex items-center justify-between">
                        <div>
                          <div className="flex items-center gap-2">
                            <DirectionsRunIcon style={{ fontSize: 18 }} />
                            <span className="font-medium">{workout.activityType || workout.activity}</span>
                          </div>
                          <p className="text-sm text-muted-foreground mt-1">
                            {formatDate(workout.date || workout.createdAt)}
                          </p>
                          {workout.distance && (
                            <p className="text-sm">Distance: {workout.distance} km</p>
                          )}
                          {workout.duration && (
                            <p className="text-sm">Duration: {Math.floor(workout.duration / 60)} min</p>
                          )}
                        </div>
                      </div>
                    </Card>
                  ))
                )}
              </TabsContent>

              <TabsContent value="friends" className="space-y-2 mt-4">
                {friends.length === 0 ? (
                  <p className="text-muted-foreground text-center py-8">No friends found</p>
                ) : (
                  friends.map((friendId, index) => (
                    <Card key={index} className="p-4">
                      <div className="flex items-center justify-between">
                        <div className="flex items-center gap-2">
                          <PeopleIcon style={{ fontSize: 18 }} />
                          <div>
                            <p className="font-medium">Friend ID: {friendId}</p>
                          </div>
                        </div>
                        <Button
                          variant="ghost"
                          size="sm"
                          onClick={() => {
                            onOpenChange(false);
                            window.location.href = `/admin/users?search=${friendId}`;
                          }}
                        >
                          View User
                        </Button>
                      </div>
                    </Card>
                  ))
                )}
              </TabsContent>
```

```
                <TabsContent value="messages" className="space-y-2 mt-4">
                  {conversations.length === 0 ? (
                    <p className="text-muted-foreground text-center py-8">No conversations found</p>
                  ) : (
                    conversations.map((conv, index) => (
                      <Card key={index} className="p-4">
                        <div className="flex items-center justify-between">
                          <div className="flex items-center gap-2">
                            <ChatIcon style={{ fontSize: 18 }} />
                            <div>
                              <p className="font-medium">Conversation: {conv.conversationId || conv.id}</p>
                              {conv.lastMessage && (
                                <p className="text-sm text-muted-foreground truncate max-w-md">
                                  {conv.lastMessage}
                                </p>
                              )}
                              {conv.lastMessageTime && (
                                <p className="text-xs text-muted-foreground">
                                  {formatDate(conv.lastMessageTime)}
                                </p>
                              )}
                            </div>
                          </div>
                        </div>
                      </Card>
                    ))
                  )}
                </TabsContent>
              </Tabs>

              {/* Delete Photo Dialog */}
              <AlertDialog open={deletePhotoDialog.open}>
                <AlertDialogContent>
                  <AlertDialogHeader>
                    <AlertDialogTitle>Remove Photo</AlertDialogTitle>
                    <AlertDialogDescription>
                      Are you sure you want to remove this photo from the user's profile?
                      This action cannot be undone.
                    </AlertDialogDescription>
                  </AlertDialogHeader>
                  <AlertDialogFooter>
                    <AlertDialogCancel onClick={() => setDeletePhotoDialog({ open: false, type: "photo" })}>
                      Cancel
                    </AlertDialogCancel>
                    <AlertDialogAction
                      onClick={handleRemovePhoto}
                      className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
                    >
                      Remove
                    </AlertDialogAction>
                  </AlertDialogFooter>
                </AlertDialogContent>
              </AlertDialog>
            </>
          ) : (
            <div className="text-center py-8 text-muted-foreground">
              User not found
            </div>
          )}
        </SheetContent>
      </Sheet>
    );
};

export default UserDetailDrawer;
```

File: src/components/VenueRequestModal.tsx

Lines: 195

```
  1 | import { useState } from "react";
  2 | import { motion, AnimatePresence } from "framer-motion";
  3 | import { Button } from "@/components/ui/button";
  4 | import { Input } from "@/components/ui/input";
  5 | import { Card } from "@/components/ui/card";
  6 | import CloseIcon from "@mui/icons-material/Close";
  7 | import LocationOnIcon from "@mui/icons-material/LocationOn";
  8 | import { toast } from "sonner";
  9 | import { useAuth } from "@/hooks/useAuth";
 10 | import { submitVenueRequest } from "@/services/venueRequestService";
 11 |
 12 | interface VenueRequestModalProps {
 13 |   onClose: () => void;
 14 |   onSuccess?: () => void;
 15 | }
 16 |
 17 | export const VenueRequestModal = ({ onClose, onSuccess }: VenueRequestModalProps) => {
 18 |   const { user } = useAuth();
 19 |   const [venueName, setVenueName] = useState("");
 20 |   const [location, setLocation] = useState("");
 21 |   const [loading, setLoading] = useState(false);
 22 |   const [submitted, setSubmitted] = useState(false);
 23 |
 24 |   const handleSubmit = async (e: React.FormEvent) => {
 25 |     e.preventDefault();
 26 |
 27 |     if (!venueName.trim()) {
 28 |       toast.error("Please enter a venue name");
 29 |       return;
 30 |     }
 31 |
 32 |     if (!location.trim()) {
 33 |       toast.error("Please enter a location");
 34 |       return;
 35 |     }
 36 |
 37 |     if (!user?.uid) {
 38 |       toast.error("Please log in to submit a request");
 39 |       return;
 40 |     }
 41 |
 42 |     setLoading(true);
 43 |     try {
 44 |       await submitVenueRequest(user.uid, venueName.trim(), location.trim());
 45 |       setSubmitted(true);
 46 |       if (onSuccess) {
 47 |         onSuccess();
 48 |       }
 49 |     } catch (error: any) {
 50 |       toast.error(error.message || "Failed to submit request. Please try again.");
 51 |     } finally {
 52 |       setLoading(false);
 53 |     }
 54 |   };
 55 |
 56 |   if (submitted) {
 57 |     return (
 58 |       <AnimatePresence>
 59 |         <motion.div
 60 |           initial={{ opacity: 0 }}
 61 |           animate={{ opacity: 1 }}
 62 |           exit={{ opacity: 0 }}
 63 |           onClick={onClose}
 64 |           className="fixed inset-0 bg-background/80 backdrop-blur-sm z-50 flex items-center
justify-center p-4"
 65 |         >
 66 |           <motion.div
```

```
            initial={{ scale: 0.95, opacity: 0, y: 20 }}
            animate={{ scale: 1, opacity: 1, y: 0 }}
            exit={{ scale: 0.95, opacity: 0, y: 20 }}
            transition={{ type: "spring", stiffness: 300, damping: 30 }}
            onClick={(e) => e.stopPropagation()}
            className="w-full max-w-md"
          >
            <Card className="overflow-hidden shadow-elevation-4 border-2 border-border/50">
              <div className="p-6 text-center space-y-4">
                <div className="mx-auto w-16 h-16 bg-success/10 rounded-full flex items-center
justify-center">
                  <LocationOnIcon className="text-success" style={{ fontSize: 32 }} />
                </div>
                <div>
                  <h2 className="text-2xl font-bold mb-2">Thank you!</h2>
                  <p className="text-muted-foreground">
                    Admins will review your request as soon as possible.
                  </p>
                </div>
                <Button onClick={onClose} className="w-full h-12">
                  Close
                </Button>
              </div>
            </Card>
          </motion.div>
        </motion.div>
      </AnimatePresence>
    );
  }

  return (
    <AnimatePresence>
      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        exit={{ opacity: 0 }}
        onClick={onClose}
        className="fixed inset-0 bg-background/80 backdrop-blur-sm z-50 flex items-center
justify-center p-4"
      >
        <motion.div
          initial={{ scale: 0.95, opacity: 0, y: 20 }}
          animate={{ scale: 1, opacity: 1, y: 0 }}
          exit={{ scale: 0.95, opacity: 0, y: 20 }}
          transition={{ type: "spring", stiffness: 300, damping: 30 }}
          onClick={(e) => e.stopPropagation()}
          className="w-full max-w-md"
        >
          <Card className="overflow-hidden shadow-elevation-4 border-2 border-border/50">
            {/* Header */}
            <div className="relative bg-gradient-to-br from-primary/10 via-success/5 to-
warning/10 p-6 border-b border-b...
              <button
                onClick={onClose}
                className="absolute top-4 right-4 p-2 bg-background/80 backdrop-blur-sm hover:bg-
secondary rounded-full ...
              >
                <CloseIcon fontSize="small" />
              </button>

              <div className="flex items-center gap-3 pr-10">
                <div className="p-3 bg-background/80 backdrop-blur-sm rounded-xl">
                  <LocationOnIcon className="text-primary" style={{ fontSize: 28 }} />
                </div>
                <div>
                  <h2 className="text-2xl font-bold">Request a Venue</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    Can't find your workout location? Let us know!
                  </p>
                </div>
              </div>
            </div>

            {/* Form */}
            <form onSubmit={handleSubmit} className="p-6 space-y-4">
```

```
              <div className="space-y-2">
                <label htmlFor="venueName" className="text-sm font-semibold">
                  Venue Name <span className="text-destructive">*</span>
                </label>
                <Input
                  id="venueName"
                  value={venueName}
                  onChange={(e) => setVenueName(e.target.value)}
                  placeholder="e.g., Rizal Park, UP Diliman"
                  className="h-12"
                  required
                />
              </div>

              <div className="space-y-2">
                <label htmlFor="location" className="text-sm font-semibold">
                  General Location <span className="text-destructive">*</span>
                </label>
                <Input
                  id="location"
                  value={location}
                  onChange={(e) => setLocation(e.target.value)}
                  placeholder="e.g., Quezon City, Makati, Pasig"
                  className="h-12"
                  required
                />
                <p className="text-xs text-muted-foreground">
                  Please provide the city or general area where this venue is located
                </p>
              </div>

              {/* Action Buttons */}
              <div className="flex gap-3 pt-4">
                <Button
                  type="button"
                  variant="outline"
                  onClick={onClose}
                  className="flex-1 h-12"
                >
                  Cancel
                </Button>
                <Button
                  type="submit"
                  disabled={loading || !venueName.trim() || !location.trim()}
                  className="flex-1 h-12 font-semibold"
                >
                  {loading ? "Submitting..." : "Submit Request"}
                </Button>
              </div>
            </form>
          </Card>
        </motion.div>
      </motion.div>
    </AnimatePresence>
  );
};

```

File: src/components/WeatherWidget.tsx

Lines: 225

```
  1 | import { useState, useEffect } from "react";
  2 | import { motion } from "framer-motion";
  3 | import { Card } from "@/components/ui/card";
  4 | import { CircularProgress } from "@mui/material";
  5 | import {
  6 |   getCurrentWeather,
  7 |   getTodayForecast,
  8 |   convertTemperature,
  9 |   getWeatherIcon,
 10 |   type WeatherData,
 11 |   type ForecastData
 12 | } from "@/services/weatherService";
 13 | import { toast } from "sonner";
 14 | import CloudIcon from "@mui/icons-material/Cloud";
 15 | import WaterDropIcon from "@mui/icons-material/WaterDrop";
 16 | import AirIcon from "@mui/icons-material/Air";
 17 |
 18 | interface WeatherWidgetProps {
 19 |   location: { lat: number; lng: number } | null;
 20 |   useMetric?: boolean;
 21 | }
 22 |
 23 | export const WeatherWidget = ({ location, useMetric = true }: WeatherWidgetProps) => {
 24 |   const [currentWeather, setCurrentWeather] = useState<WeatherData | null>(null);
 25 |   const [forecast, setForecast] = useState<ForecastData | null>(null);
 26 |   const [loading, setLoading] = useState(true);
 27 |   const [error, setError] = useState<string | null>(null);
 28 |
 29 |   useEffect(() => {
 30 |     if (!location) {
 31 |       setLoading(false);
 32 |       return;
 33 |     }
 34 |
 35 |     const fetchWeather = async () => {
 36 |       setLoading(true);
 37 |       setError(null);
 38 |
 39 |       try {
 40 |         const [current, todayForecast] = await Promise.all([
 41 |           getCurrentWeather(location.lat, location.lng),
 42 |           getTodayForecast(location.lat, location.lng),
 43 |         ]);
 44 |
 45 |         setCurrentWeather(current);
 46 |         setForecast(todayForecast);
 47 |       } catch (err: any) {
 48 |         console.error("Error fetching weather:", err);
 49 |         const errorMessage = err.message || "Unable to fetch weather data";
 50 |         setError(errorMessage);
 51 |
 52 |         // Show more specific error messages
 53 |         if (errorMessage.includes("API key")) {
 54 |           console.warn("& þ  OpenWeatherMap API key not configured. Add VITE_OPENWEATHER_API_KEY to your
 55 | env file.");// Don't show toast for missing API key
 56 |         } else if (errorMessage.includes("401") || errorMessage.includes("Invalid API key")) {
 57 |           toast.error("Invalid weather API key. Please check your configuration.");
 58 |         } else if (errorMessage.includes("429") || errorMessage.includes("rate limit")) {
 59 |           toast.error("Weather API rate limit exceeded. Please try again later.");
 60 |         } else {
 61 |           toast.error(`Failed to load weather: ${errorMessage}`);
 62 |         }
 63 |       } finally {
 64 |         setLoading(false);
 65 |       }
 66 |     };
```

```
67 |
68 |     fetchWeather();
69 |   }, [location?.lat, location?.lng]);
70 |
71 |   if (!location) {
72 |     return null;
73 |   }
74 |
75 |   if (loading) {
76 |     return (
77 |       <motion.div
78 |         initial={{ opacity: 0, y: -10 }}
79 |         animate={{ opacity: 1, y: 0 }}
80 |         className="mb-4"
81 |       >
82 |         <Card className="p-4 bg-card/50 backdrop-blur-sm border-border/50">
83 |           <div className="flex items-center justify-center gap-3">
84 |             <CircularProgress size={20} />
85 |             <span className="text-sm text-muted-foreground">Loading weather...</span>
86 |           </div>
87 |         </Card>
88 |       </motion.div>
89 |     );
90 |   }
91 |
92 |   if (error) {
93 |     // Show helpful error message for invalid API key
94 |     if (error.includes("Invalid API key") || error.includes("401")) {
95 |       return (
96 |         <motion.div
97 |           initial={{ opacity: 0, y: -10 }}
98 |           animate={{ opacity: 1, y: 0 }}
99 |           className="mb-4"
100 |         >
101 |           <Card className="p-4 bg-warning/10 backdrop-blur-sm border-warning/30">
102 |             <div className="flex items-center gap-2 text-sm">
103 |               <CloudIcon className="text-warning" style={{ fontSize: 16 }} />
104 |               <div>
105 |                 <span className="font-medium text-warning">Weather API key invalid</span>
106 |                 <p className="text-xs text-muted-foreground mt-1">
107 |                   Please check your OpenWeatherMap API key. It may need activation.
108 |                 </p>
109 |               </div>
110 |             </div>
111 |           </Card>
112 |         </motion.div>
113 |       );
114 |     }
115 |
116 |     // Don't show error state if API key is missing (silent fail)
117 |     if (error.includes("API key is not configured")) {
118 |       return null;
119 |     }
120 |
121 |     return (
122 |       <motion.div
123 |         initial={{ opacity: 0, y: -10 }}
124 |         animate={{ opacity: 1, y: 0 }}
125 |         className="mb-4"
126 |       >
127 |         <Card className="p-4 bg-card/50 backdrop-blur-sm border-border/50">
128 |           <div className="flex items-center gap-2 text-sm text-muted-foreground">
129 |             <CloudIcon style={{ fontSize: 16 }} />
130 |             <span>Weather unavailable: {error}</span>
131 |           </div>
132 |         </Card>
133 |       </motion.div>
134 |     );
135 |   }
136 |
137 |   if (!currentWeather || !forecast) {
```

```
138 |     return null;
139 |   }
140 |
141 |   const currentTemp = convertTemperature(currentWeather.temperature, useMetric);
142 |   const feelsLikeTemp = currentWeather.feelsLike
143 |     ? convertTemperature(currentWeather.feelsLike, useMetric)
144 |     : null;
145 |   const highTemp = convertTemperature(forecast.high, useMetric);
146 |   const lowTemp = convertTemperature(forecast.low, useMetric);
147 |   const weatherEmoji = getWeatherIcon(currentWeather.condition, currentWeather.icon);
148 |
149 |   return (
150 |     <motion.div
151 |       initial={{ opacity: 0, y: -10 }}
152 |       animate={{ opacity: 1, y: 0 }}
153 |       transition={{ delay: 0.1 }}
154 |       className="mb-4"
155 |     >
156 |       <Card className="p-4 bg-gradient-to-r from-primary/10 via-card/50 to-success/10 backdrop-
blur-sm border-border/50">
157 |         <div className="flex items-center justify-between gap-4">
158 |           {/* Current Weather */}
159 |           <div className="flex items-center gap-3 flex-1">
160 |             <div className="text-4xl">{weatherEmoji}</div>
161 |             <div className="flex-1">
162 |               <div className="flex items-baseline gap-2">
163 |                 <span className="text-3xl font-bold text-foreground">
164 |                   {currentTemp.value}
165 |                 </span>
166 |                 <span className="text-lg text-muted-foreground">
167 |                   {currentTemp.unit}
168 |                 </span>
169 |               </div>
170 |               <div className="flex items-center gap-2 mt-1">
171 |                 <span className="text-sm text-muted-foreground capitalize">
172 |                   {currentWeather.description}
173 |                 </span>
174 |                 {feelsLikeTemp && (
175 |                   <>
176 |                     <span className="text-xs text-muted-foreground">•</span>
177 |                     <span className="text-xs text-muted-foreground">
178 |                       Feels like {feelsLikeTemp.value}{feelsLikeTemp.unit}
179 |                     </span>
180 |                   </>
181 |                 )}
182 |               </div>
183 |             </div>
184 |           </div>
185 |
186 |           {/* Today's Forecast */}
187 |           <div className="flex items-center gap-4 border-l border-border/50 pl-4">
188 |             <div className="text-right">
189 |               <div className="flex items-center gap-2 text-sm">
190 |                 <span className="text-muted-foreground">High</span>
191 |                 <span className="font-semibold text-foreground">
192 |                   {highTemp.value}{highTemp.unit}
193 |                 </span>
194 |               </div>
195 |               <div className="flex items-center gap-2 text-sm mt-1">
196 |                 <span className="text-muted-foreground">Low</span>
197 |                 <span className="font-semibold text-foreground">
198 |                   {lowTemp.value}{lowTemp.unit}
199 |                 </span>
200 |               </div>
201 |             </div>
202 |           </div>
203 |
204 |           {/* Additional Info */}
205 |           <div className="flex items-center gap-3 text-xs text-muted-foreground border-l border-
border/50 pl-4">
206 |             {currentWeather.humidity !== undefined && (
207 |               <div className="flex items-center gap-1">
208 |                 <WaterDropIcon style={{ fontSize: 16 }} />
```

```
209 |                 <span>{currentWeather.humidity}%</span>
210 |               </div>
211 |             )}
212 |             {currentWeather.windSpeed !== undefined && (
213 |               <div className="flex items-center gap-1">
214 |                 <AirIcon style={{ fontSize: 16 }} />
215 |                 <span>{currentWeather.windSpeed} km/h</span>
216 |               </div>
217 |             )}
218 |           </div>
219 |         </div>
220 |       </Card>
221 |     </motion.div>
222 |   );
223 | };
224 |
225 |
```

File: src/components/WorkoutDetailModal.tsx

Lines: 505

```
import { motion, AnimatePresence } from "framer-motion";
import { useNavigate } from "react-router-dom";
import { Button } from "@/components/ui/button";
import { Card, CardContent } from "@/components/ui/card";
import { Dialog, DialogContent, DialogHeader, DialogTitle } from "@/components/ui/dialog";
import { ScrollArea } from "@/components/ui/scroll-area";
import { Avatar } from "@mui/material";
import React, { useState, useEffect } from "react";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import TimerIcon from "@mui/icons-material/Timer";
import SpeedIcon from "@mui/icons-material/Speed";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import PeopleIcon from "@mui/icons-material/People";
import CloseIcon from "@mui/icons-material/Close";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import SendIcon from "@mui/icons-material/Send";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import PersonIcon from "@mui/icons-material/Person";
import { format } from "date-fns";
import { toast } from "sonner";
import type { WorkoutHistory, NearbyUser } from "@/contexts/UserContext";
import { useAuth } from "@/hooks/useAuth";
import { useUser } from "@/contexts/UserContext";
import { sendFriendRequest } from "@/services/friendService";
import { listenToFriendRequests } from "@/services/friendService";
import { ProfileView } from "@/pages/ProfileView";

interface WorkoutDetailModalProps {
  isOpen: boolean;
  onClose: () => void;
  workout: WorkoutHistory;
  useMetric: boolean;
}

type FriendStatus = "not_friends" | "request_pending" | "request_received" | "friends" |
"denied";

export const WorkoutDetailModal = ({
  isOpen,
  onClose,
  workout,
  useMetric,
}: WorkoutDetailModalProps) => {
  const navigate = useNavigate();
  const { user } = useAuth();
  const { userProfile } = useUser();
  const [friendStatuses, setFriendStatuses] = useState<Record<string, { status: FriendStatus;
cooldownUntil?: number; }>>({});
  const [friendRequests, setFriendRequests] = useState<{ incoming: string[]; outgoing: string[] }
>({ incoming: [], outgo
  const [selectedUser, setSelectedUser] = useState<NearbyUser | null>(null);
  const [showProfileView, setShowProfileView] = useState(false);

  // Listen to friend requests
  useEffect(() => {
    if (!user?.uid || !isOpen) return;

    const unsubscribe = listenToFriendRequests(user.uid, (requests) => {
      setFriendRequests(requests);
    });

    return () => unsubscribe();
  }, [user?.uid, isOpen]);

  // Determine friend status for a user
  const getFriendStatus = (userId: string | number): FriendStatus => {
    const id = typeof userId === "number" ? userId.toString() : userId;
```

```
67 |
68 |     // Check if already friends
69 |     const friends = userProfile?.friends || [];
70 |     if (friends.some((f: any) => String(f) === id)) {
71 |       return "friends";
72 |     }
73 |
74 |     // Check if request pending (outgoing)
75 |     if (friendRequests.outgoing.includes(id)) {
76 |       return "request_pending";
77 |     }
78 |
79 |     // Check if request received (incoming)
80 |     if (friendRequests.incoming.includes(id)) {
81 |       return "request_received";
82 |     }
83 |
84 |     // Check local state
85 |     const localStatus = friendStatuses[id];
86 |     if (localStatus) {
87 |       return localStatus.status;
88 |     }
89 |
90 |     return "not_friends";
91 |   };
92 |
93 |   const handleAddFriend = async (userData: NearbyUser) => {
94 |     if (!user?.uid) {
95 |       toast.error("You must be logged in to add friends");
96 |       return;
97 |     }
98 |
99 |     const userId = typeof userData.id === "number" ? userData.id.toString() : userData.id;
100 |
101 |     try {
102 |       await sendFriendRequest(user.uid, userId);
103 |       setFriendStatuses(prev => ({
104 |         ...prev,
105 |         [userId]: { status: "request_pending" }
106 |       }));
107 |       toast.success(`Friend request sent to ${userData.name}`);
108 |     } catch (error) {
109 |       console.error("Error sending friend request:", error);
110 |       toast.error("Failed to send friend request. Please try again.");
111 |     }
112 |   };
113 |
114 |   const handleSendMessage = (userData: NearbyUser) => {
115 |     navigate("/chat", {
116 |       state: {
117 |         user: {
118 |           id: userData.id,
119 |           name: userData.name,
120 |           avatar: userData.avatar || ""
121 |         }
122 |       }
123 |     });
124 |     onClose(); // Close workout modal when navigating
125 |   };
126 |
127 |   const handleViewProfile = (userData: NearbyUser) => {
128 |     setSelectedUser(userData);
129 |     setShowProfileView(true);
130 |   };
131 |
132 |   const getCooldownDays = (userId: string | number): number => {
133 |     const id = typeof userId === "number" ? userId.toString() : userId;
134 |     const cooldownUntil = friendStatuses[id]?.cooldownUntil;
135 |     if (!cooldownUntil) return 0;
136 |     const now = Date.now();
137 |     const diff = cooldownUntil - now;
```

```
    if (diff <= 0) return 0;
    return Math.ceil(diff / (1000 * 60 * 60 * 24));
  };
  const formatTime = (seconds: number) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;
    if (hours > 0) {
      return `${hours}:${minutes.toString().padStart(2, "0")}:${secs.toString().padStart(2, "0")}
`;
    }
    return `${minutes}:${secs.toString().padStart(2, "0")}`;
  };

  const convertDistance = (km: number) => {
    if (useMetric) return { value: km.toFixed(2), unit: "km" };
    return { value: (km * 0.621371).toFixed(2), unit: "mi" };
  };

  const convertSpeed = (kmh: number) => {
    if (useMetric) return { value: kmh.toFixed(1), unit: "km/h" };
    return { value: (kmh * 0.621371).toFixed(1), unit: "mph" };
  };

  const getActivityConfig = (activity: "running" | "cycling" | "walking") => {
    const configs = {
      running: {
        icon: DirectionsRunIcon,
        color: "success",
        label: "Running",
        bgClass: "bg-success/10",
        textClass: "text-success",
        borderClass: "border-success/30"
      },
      cycling: {
        icon: DirectionsBikeIcon,
        color: "primary",
        label: "Cycling",
        bgClass: "bg-primary/10",
        textClass: "text-primary",
        borderClass: "border-primary/30"
      },
      walking: {
        icon: DirectionsWalkIcon,
        color: "warning",
        label: "Walking",
        bgClass: "bg-warning/10",
        textClass: "text-warning",
        borderClass: "border-warning/30"
      },
    };
    return configs[activity];
  };

  const config = getActivityConfig(workout.activity);
  const Icon = config.icon;
  const distanceData = convertDistance(workout.distance);
  const speedData = convertSpeed(workout.avgSpeed);

  return (
    <Dialog open={isOpen} onOpenChange={onClose}>
      <DialogContent className="max-w-2xl w-full mx-4 max-h-[95vh] sm:max-h-[90vh] p-0 overflow-
hidden">
        <ScrollArea className="max-h-[95vh] sm:max-h-[90vh]">
          <div className="p-4 sm:p-6 space-y-4 sm:space-y-6">
            {/* Header */}
            <DialogHeader>
              <div className="flex items-start gap-2 sm:gap-3">
                <div className={`p-2 sm:p-3 rounded-xl ${config.bgClass} flex-shrink-0`}>
                  <Icon className={config.textClass} style={{ fontSize: 24 }} sx={{ fontSize:
{ xs: 24, sm: 32 } }} /></div>
                <div className="flex-1 min-w-0">
                  <DialogTitle className="text-lg sm:text-2xl leading-tight">{config.label}
Session</DialogTitle>
```

```
                    <p className="text-xs sm:text-sm text-muted-foreground mt-1 break-words">
                      {format(new Date(workout.date), "EEEE, MMMM d, yyyy 'at' h:mm a")}
                    </p>
                  </div>
                </div>
              </DialogHeader>

              {/* Caption */}
              {(workout as any).caption && (
                <div className="p-3 sm:p-4 bg-muted/30 rounded-lg border border-border">
                  <p className="text-xs sm:text-sm leading-relaxed">{(workout as any).caption}</p>
                </div>
              )}

              {/* Location */}
              {workout.location && (
                <div className="flex items-center gap-2 text-muted-foreground">
                  <LocationOnIcon style={{ fontSize: 18 }} sx={{ fontSize: { xs: 18, sm: 20 } }}
className="flex-shrink-0" />
                  <span className="text-xs sm:text-sm truncate">{workout.location}</span>
                </div>
              )}

              {/* Stats Grid */}
              <div className="grid grid-cols-2 sm:grid-cols-4 gap-2 sm:gap-4">
                <Card className={`${config.bgClass} border-2 ${config.borderClass}`}>
                  <CardContent className="p-3 sm:p-4 text-center">
                    <TimerIcon className={config.textClass} sx={{ fontSize: { xs: 20, sm: 28 } }} /
>                    <div className="text-lg sm:text-2xl font-bold mt-1
sm:mt-2">{formatTime(workout.duration)}</div>
                    <div className="text-[10px] sm:text-xs text-muted-foreground mt-0.5
sm:mt-1">Duration</div></CardContent>
                </Card>

                <Card className={`${config.bgClass} border-2 ${config.borderClass}`}>
                  <CardContent className="p-3 sm:p-4 text-center">
                    <div className="text-2xl sm:text-3xl">Ø=ÜÍ</div>
                    <div className="text-lg sm:text-2xl font-bold mt-1 sm:mt-2">
                      {distanceData.value}
                    </div>
                    <div className="text-[10px] sm:text-xs text-muted-foreground mt-0.5
sm:mt-1">{distanceData.unit}</div>
                  </CardContent>
                </Card>

                <Card className={`${config.bgClass} border-2 ${config.borderClass}`}>
                  <CardContent className="p-3 sm:p-4 text-center">
                    <SpeedIcon className={config.textClass} sx={{ fontSize: { xs: 20, sm: 28 } }} /
>                    <div className="text-lg sm:text-2xl font-bold mt-1 sm:mt-2">
                      {speedData.value}
                    </div>
                    <div className="text-[10px] sm:text-xs text-muted-foreground mt-0.5
sm:mt-1">{speedData.unit}</div>
                  </CardContent>
                </Card>
              </div>

              {/* Photos */}
              {(workout as any).photos && (workout as any).photos.length > 0 && (
                <div className={`grid gap-3 ${(workout as any).photos.length === 1 ? 'grid-
cols-1' : (workout as any).photos.length === 3 && idx === 0 ?
'col-span-2' : ''}`}>
                  {(workout as any).photos.map((photo: string, idx: number) => (
                    <div
                      key={idx}
                      className={`relative rounded-xl overflow-hidden ${(workout as
any).photos.length === 1 ? 'max-h-[400px]' : ''}`}
                      style={{ paddingTop: (workout as any).photos.length === 1 ? '56.25%' :
'100%' }}
                    >
                      <img
                        src={photo}
                        alt={`Workout photo ${idx + 1}`}
                        className="absolute inset-0 w-full h-full object-cover"
                      />
                    </div>
                  ))}
                </div>
              )}
```

```
            {/* Route Map Placeholder */}
            <Card>
              <CardContent className="p-0">
                <div className="bg-gradient-to-br from-primary/10 via-success/5 to-warning/10
rounded-lg h-32 sm:h-48 fl...
                  <div className="text-center px-2">
                    <LocationOnIcon sx={{ fontSize: { xs: 32, sm:...
                    <p className="text-xs sm:text-sm text-muted-foreground">Route map preview</p>
                    <p className="text-[10px] sm:text-xs text-muted-foreground mt-0.5
sm:mt-1">GPS tracking feature comi...
                  </div>
                </CardContent>
              </Card>

              {/* Nearby Users During Workout */}
              {workout.nearbyUsers && workout.nearbyUsers.length > 0 && (
                <Card>
                  <CardContent className="p-4 sm:p-6 space-y-3 sm:space-y-4">
                    <div className="flex items-center gap-2">
                      <PeopleIcon sx={{ fontSize: { xs: 20,
sm: 24 } }} />
                      <h3 className="text-base sm:text-lg font-bold">
                        People Nearby During Workout
                      </h3>
                    </div>
                    <p className="text-xs sm:text-sm text-muted-foreground">
                      {workout.nearbyUsers.length} {workout.nearbyUsers.length === 1 ? "person
was" : "people were"} activ...
                    </p>

                    <div className="space-y-2 sm:space-y-3">
                      {workout.nearbyUsers.map((user, index) => {
                        const friendStatus = getFriendStatus(user.id);
                        const cooldownDays = getCooldownDays(user.id);

                        return (
                          <motion.div
                            key={user.id}
                            initial={{ opacity: 0, x: -10 }}
                            animate={{ opacity: 1, x: 0 }}
                            transition={{ delay: index * 0.05 }}
                            className="flex items-center gap-2 sm:gap-3 p-3 sm:p-4 rounded-xl bg-
muted/50 hover:bg-muted t...
                          >
                            <button
                              onClick={() => handleViewProfile(user)}
                              className="flex-shrink-0"
                            >
                              <Avatar
                                src={user.avatar}
                                alt={user.name}
                                sx={{ width: 48, height: 48, cursor: "pointer" }}
                                className="sm:w-[56px] sm:h-[56px]"
                              />
                            </button>
                            <div className="flex-1 min-w-0">
                              <div className="flex items-center gap-1.5 sm:gap-2 mb-0.5 sm:mb-1
flex-wrap">
                                <button
                                  onClick={() => handleViewProfile(user)}
                                  className="font-semibold hover:underline text-left text-sm
sm:text-base truncate max-w-[...
                                >
                                  {user.name}
                                </button>
                                {friendStatus === "friends" && (
                                  <div className="flex items-center gap-1 px-1.5 sm:px-2 py-0.5 bg-
success/10 border borde...
                                    <CheckCircleIcon className="text-success" sx={{ fontSize:
{ xs: 12, sm: 14 } }} />
                                    <span className="text-[10px] sm:text-xs text-success font-
medium">Friends</span>
                                  </div>
                                )}
                                {friendStatus === "request_pending" && (
                                  <div className="px-1.5 sm:px-2 py-0.5 bg-warning/10 border
border-warning rounded-full f...
                                    <span className="text-[10px] sm:text-xs text-warning font-
medium">Request Sent</span>
                                  </div>
                                )}
```

```
                            </div>
                            <div className="flex items-center gap-1.5 sm:gap-2 text-xs sm:text-
sm:text-muted-foreground ...
                              <span>
                                {user.activity === "Running" && "Ø<ßÃ"}
                                {user.activity === "Cycling" && "Ø=Þ´"}
                                {user.activity === "Walking" && "Ø=Þ¶"}
                              </span>
                              <span className="capitalize">{user.activity?.toLowerCase()}</span>
                              <span>•</span>
                              <span className="truncate">{user.distance} away</span>
                            </div>
                          </div>

                          {/* Action Buttons */}
                          <div className="flex gap-1 sm:gap-2 flex-shrink-0">
                            <Button
                              size="sm"
                              variant="outline"
                              onClick={(e) => {
                                e.stopPropagation();
                                handleViewProfile(user);
                              }}
                              className="h-8 sm:h-9 px-2 sm:px-3"
                              title="View Profile"
                            >
                              <PersonIcon sx={{ fontSize: { xs: 16, sm: 18 } }} />
                            </Button>
                            {friendStatus === "not_friends" && (
                              <Button
                                size="sm"
                                variant="outline"
                                onClick={(e) => {
                                  e.stopPropagation();
                                  handleAddFriend(user);
                                }}
                                disabled={cooldownDays > 0}
                                className="h-8 sm:h-9 px-2 sm:px-3 text-xs"
                                title={cooldownDays > 0 ? `Try again in ${cooldownDays}
day${cooldownDays !== 1 ? 's' : ...  >
                                <PersonAddIcon sx={{ fontSize: { xs: 16, sm: 18 } }}
className="mr-0.5 sm:mr-1" />
                                <span className="hidden sm:inline">Add</span>
                              </Button>
                            )}
                            {friendStatus === "request_pending" && (
                              <Button
                                size="sm"
                                variant="outline"
                                disabled
                                className="h-8 sm:h-9 px-2 sm:px-3 text-xs"
                              >
                                <span className="hidden sm:inline">Pending</span>
                                <span className="sm:hidden">...</span>
                              </Button>
                            )}
                            <Button
                              size="sm"
                              onClick={(e) => {
                                e.stopPropagation();
                                handleSendMessage(user);
                              }}
                              className="h-8 sm:h-9 px-2 sm:px-3"
                              title="Send Message"
                            >
                              <SendIcon sx={{ fontSize: { xs: 16, sm: 18 } }} />
                            </Button>
                          </div>
                        </motion.div>
                      );
                    })}
                  </div>

```

```
                  <div className="mt-3 sm:mt-4 p-3 sm:p-4 bg-primary/5 rounded-lg border border-primary/20">
                    <div className="space-y-1.5 sm:space-y-2">
                      <p className="text-xs sm:text-sm font-semibold text-foreground flex items-center gap-1.5 sm:gap-2"...
                        <PeopleIcon className="flex-shrink-0" sx={{ fontSize: { xs: 16, sm: 18 } }} />
                        Social Workout Feature
                      </p>
                      <p className="text-[10px] sm:text-xs text-muted-foreground leading-relaxed">
                        Connect with people who were active nearby during your workout. Send friend requests to stay mot...
                      </p>
                      <p className="text-[10px] sm:text-xs text-muted-foreground">
                        <span className="font-semibold text-foreground">Privacy:</span> These users could also see your ...</p>
                    </div>
                  </div>
                </CardContent>
              </Card>
            )}

            {/* No nearby users message */}
            {(!workout.nearbyUsers || workout.nearbyUsers.length === 0) && (
              <Card>
                <CardContent className="p-6 sm:p-8 text-center">
                  <PeopleIcon className="text-muted-foreground/30 mx-auto mb-2 sm:mb-3" sx={{ fontSize: { xs: 40, sm: 56 } }} />
                  <h3 className="font-semibold text-base sm:text-lg mb-1.5 sm:mb-2">No One Nearby</h3>
                  <p className="text-xs sm:text-sm text-muted-foreground mb-3 sm:mb-4">
                    No other users were active within 2km during this workout
                  </p>
                  <p className="text-[10px] sm:text-xs text-muted-foreground">
                    Try working out in popular areas or during peak times to connect with other fitness enthusiasts!
                  </p>
                </CardContent>
              </Card>
            )}

            {/* Close Button */}
            <Button onClick={onClose} className="w-full h-11 sm:h-12" size="lg">
              Close
            </Button>
          </div>
        </ScrollArea>
      </DialogContent>

      {/* Profile View Modal */}
      <AnimatePresence>
        {showProfileView && selectedUser && (
          <ProfileView
            user={{
              id: selectedUser.id,
              name: selectedUser.name,
              distance: selectedUser.distance || "Unknown",
              activity: selectedUser.activity || "Active",
              avatar: selectedUser.avatar || "",
              photos: [],
              bio: ""
            }}
            friendStatus={getFriendStatus(selectedUser.id)}
            cooldownDays={getCooldownDays(selectedUser.id)}
            onClose={() => {
              setShowProfileView(false);
              setSelectedUser(null);
            }}
            onSendMessage={() => {
              setShowProfileView(false);
              handleSendMessage(selectedUser);
            }}
            onAddFriend={() => {
              handleAddFriend(selectedUser);
              setShowProfileView(false);
            }}
            onAcceptFriend={() => {
              // This would be handled by the friend request modal
```

```
493 |               setShowProfileView(false);
494 |             }}
495 |             onDeclineFriend={() => {
496 |               // This would be handled by the friend request modal
497 |               setShowProfileView(false);
498 |             }}
499 |           />
500 |         )}
501 |       </AnimatePresence>
502 |     </Dialog>
503 |   );
504 | };
505 |
```

File: src/components/WorkoutHistoryFeed.tsx

Lines: 237

```
  1 | import { useState, useEffect, useMemo } from "react";
  2 | import { motion } from "framer-motion";
  3 | import { Card } from "@/components/ui/card";
  4 | import { Button } from "@/components/ui/button";
  5 | import { Avatar } from "@mui/material";
  6 | import { listenToAllUsersWorkouts, WorkoutWithUser } from "@/services/workoutService";
  7 | import { getDisplayName } from "@/utils/anonymousName";
  8 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
  9 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
 10 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
 11 | import LocationOnIcon from "@mui/icons-material/LocationOn";
 12 | import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
 13 | import { formatDistanceToNow } from "date-fns";
 14 | import { Activity } from "@/contexts/UserContext";
 15 |
 16 | interface WorkoutHistoryFeedProps {
 17 |   currentUserId: string;
 18 |   selectedActivity: string; // "all" | "running" | "cycling" | "walking"
 19 |   timeframe: "week" | "month" | "all";
 20 |   useMetric: boolean;
 21 |   onWorkoutClick: (workout: WorkoutWithUser) => void;
 22 | }
 23 |
 24 | export const WorkoutHistoryFeed = ({
 25 |   currentUserId,
 26 |   selectedActivity,
 27 |   timeframe,
 28 |   useMetric,
 29 |   onWorkoutClick,
 30 | }: WorkoutHistoryFeedProps) => {
 31 |   const [workouts, setWorkouts] = useState<WorkoutWithUser[]>([]);
 32 |   const [loading, setLoading] = useState(true);
 33 |
 34 |   // Listen to all users' workouts
 35 |   useEffect(() => {
 36 |     if (!currentUserId) {
 37 |       setLoading(false);
 38 |       return;
 39 |     }
 40 |
 41 |     const unsubscribe = listenToAllUsersWorkouts(currentUserId, (allWorkouts) => {
 42 |       setWorkouts(allWorkouts);
 43 |       setLoading(false);
 44 |     });
 45 |
 46 |     return () => unsubscribe();
 47 |   }, [currentUserId]);
 48 |
 49 |   // Filter and sort workouts
 50 |   const filteredWorkouts = useMemo(() => {
 51 |     let filtered = [...workouts];
 52 |
 53 |     // Filter by activity type
 54 |     if (selectedActivity !== "all") {
 55 |       filtered = filtered.filter(
 56 |         (item) => item.workout.activity === selectedActivity
 57 |       );
 58 |     }
 59 |
 60 |     // Filter by timeframe
 61 |     const now = new Date();
 62 |     if (timeframe === "week") {
 63 |       const weekAgo = new Date(now.getTime() - 7 * 24 * 60 * 60 * 1000);
 64 |       filtered = filtered.filter((item) => item.workout.date >= weekAgo);
 65 |     } else if (timeframe === "month") {
 66 |       const monthAgo = new Date(now.getTime() - 30 * 24 * 60 * 60 * 1000);
```

```
 67 |       filtered = filtered.filter((item) => item.workout.date >= monthAgo);
 68 |     }
 69 |     // "all" doesn't filter by time
 70 |
 71 |     // Sort by date (most recent first) - already sorted in service, but ensure it
 72 |     return filtered.sort(
 73 |       (a, b) => b.workout.date.getTime() - a.workout.date.getTime()
 74 |     );
 75 |   }, [workouts, selectedActivity, timeframe]);
 76 |
 77 |   const formatTime = (seconds: number) => {
 78 |     const hours = Math.floor(seconds / 3600);
 79 |     const minutes = Math.floor((seconds % 3600) / 60);
 80 |     if (hours > 0) {
 81 |       return `${hours}h ${minutes}m`;
 82 |     }
 83 |     return `${minutes}m`;
 84 |   };
 85 |
 86 |   const convertDistance = (km: number) => {
 87 |     if (useMetric) return `${km.toFixed(1)} km`;
 88 |     return `${(km * 0.621371).toFixed(1)} mi`;
 89 |   };
 90 |
 91 |   const convertSpeed = (kmh: number) => {
 92 |     if (useMetric) return `${kmh.toFixed(1)} km/h`;
 93 |     return `${(kmh * 0.621371).toFixed(1)} mph`;
 94 |   };
 95 |
 96 |   const getActivityIcon = (activity: Activity) => {
 97 |     switch (activity) {
 98 |       case "running":
 99 |         return DirectionsRunIcon;
100 |       case "cycling":
101 |         return DirectionsBikeIcon;
102 |       case "walking":
103 |         return DirectionsWalkIcon;
104 |       default:
105 |         return FitnessCenterIcon;
106 |     }
107 |   };
108 |
109 |   const getActivityColor = (activity: Activity) => {
110 |     switch (activity) {
111 |       case "running":
112 |         return "text-success";
113 |       case "cycling":
114 |         return "text-primary";
115 |       case "walking":
116 |         return "text-warning";
117 |       default:
118 |         return "text-muted-foreground";
119 |     }
120 |   };
121 |
122 |   if (loading) {
123 |     return (
124 |       <Card className="p-12 text-center">
125 |         <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary mx-auto"></
div>
126 |         <p className="text-muted-foreground mt-4">Loading workout history...</p>
127 |       </Card>
128 |     );
129 |   }
130 |
131 |   if (filteredWorkouts.length === 0) {
132 |     return (
133 |       <div className="text-center py-12">
134 |         <FitnessCenterIcon className="mx-auto text-muted-foreground" style={{ fontSize: 48 }} />
135 |         <p className="text-muted-foreground mt-4">No workouts found</p>
136 |         <p className="text-sm text-muted-foreground mt-1">
137 |           {selectedActivity !== "all"
```

```
            ? `No ${selectedActivity} workouts in the selected timeframe`
            : "No workouts match your filters"}
        </p>
      </div>
    );
  }

  return (
    <div className="space-y-3">
      {filteredWorkouts.map((item, index) => {
        const ActivityIcon = getActivityIcon(item.workout.activity);
        const activityColor = getActivityColor(item.workout.activity);
        const user = item.userData;
        const workout = item.workout;
        const username = user?.name || user?.username || null;
        const activity = user?.activity || workout.activity || null;
        const displayName = getDisplayName(username, item.userId, activity);

        return (
          <motion.div
            key={`${item.userId}-${workout.id}`}
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: index * 0.05 }}
            className="flex items-center gap-4 p-4 border border-border rounded-lg hover:shadow-
md transition-shadow cur
            onClick={() => onWorkoutClick(item)}
          >
            <div className="relative">
              <Avatar
                src={user?.photoURL || `https://ui-avatars.com/api/?
name=${encodeURIComponent(displayName)}`}
                alt={displayName}
                sx={{ width: 56, height: 56 }}
              />
              <div className={`absolute -bottom-1 -right-1 w-6 h-6 rounded-full border-2 border-
card flex items-center j
                <ActivityIcon className={activityColor} style={{ fontSize: 14 }} />
              </div>
            </div>

            <div className="flex-1 min-w-0">
              <div className="flex items-center gap-2">
                <h3 className="font-semibold text-sm truncate">
                  {displayName}
                </h3>
              </div>

              <div className="flex items-center gap-2 mt-1">
                <ActivityIcon className={activityColor} style={{ fontSize: 16 }} />
                <span className="text-xs text-muted-foreground capitalize">
                  {workout.activity}
                </span>
                <span className="text-xs text-muted-foreground">•</span>
                <span className="text-xs text-muted-foreground">
                  {formatDistanceToNow(workout.date, { addSuffix: true })}
                </span>
              </div>

              <div className="flex items-center gap-3 mt-2 text-xs">
                <span className="text-muted-foreground">
                  {formatTime(workout.duration)}
                </span>
                <span className="text-muted-foreground">•</span>
                <span className="text-muted-foreground">
                  {convertDistance(workout.distance)}
                </span>
                <span className="text-muted-foreground">•</span>
                <span className="text-muted-foreground">
                  {convertSpeed(workout.avgSpeed)}
                </span>
              </div>

              {workout.location && (
```

```
                <div className="flex items-center gap-1 mt-1">
                  <LocationOnIcon className="text-primary" style={{ fontSize: 14 }} />
                  <span className="text-xs text-muted-foreground truncate">
                    {workout.location}
                  </span>
                </div>
              )}
            </div>

            <div className="text-right">
              <Button
                variant="ghost"
                size="sm"
                onClick={(e) => {
                  e.stopPropagation();
                  onWorkoutClick(item);
                }}
              >
                View
              </Button>
            </div>
          </motion.div>
        );
      })}
    </div>
  );
};

```

File: src/components/WorkoutPost.tsx

Lines: 282

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Card } from "@/components/ui/card";
import { Avatar } from "@mui/material";
import { WorkoutPost as FirebaseWorkoutPost, toggleKudos } from "@/services/feedService";
import { getUserData } from "@/services/authService";
import { ProfileView } from "@/pages/ProfileView";
import { getDisplayName } from "@/utils/anonymousName";
import { getProfilePictureUrl } from "@/utils/profilePicture";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FavoriteBorderIcon from "@mui/icons-material/FavoriteBorder";
import FavoriteIcon from "@mui/icons-material/Favorite";
import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
import ShareIcon from "@mui/icons-material/Share";
import { formatDistanceToNow } from "date-fns";
import { toast } from "sonner";

interface WorkoutPostProps {
  post: FirebaseWorkoutPost;
  onCommentClick: (post: FirebaseWorkoutPost) => void;
  useMetric: boolean;
  currentUserId: string;
}

export const WorkoutPost = ({ post, onCommentClick, useMetric, currentUserId }:
WorkoutPostProps) => {
  const [user, setUser] = useState<any>(null);
  const [kudos, setKudos] = useState<string[]>(post.kudos || []);
  const [showProfile, setShowProfile] = useState(false);
  const [loading, setLoading] = useState(true);
  const hasKudos = kudos.includes(currentUserId);

  // Fetch user data from Firebase
  useEffect(() => {
    const fetchUser = async () => {
      try {
        const userData = await getUserData(post.userId);
        if (userData) {
          const username = userData.name || userData.username || null;
          const activity = userData.activity || null;
          const displayName = getDisplayName(username, post.userId, activity);
          setUser({
            id: post.userId,
            username: displayName,
            avatar: getProfilePictureUrl(userData.photoURL, userData.avatar, displayName),
            bio: userData.bio,
            photos: userData.photos || []
          });
        }
        setLoading(false);
      } catch (error) {
        console.error("Error fetching user data:", error);
        setLoading(false);
      }
    };

    fetchUser();
  }, [post.userId]);

  // Update kudos when post changes
  useEffect(() => {
    setKudos(post.kudos || []);
  }, [post.kudos]);

  const activityConfig = {
```

```
    running: { icon: DirectionsRunIcon, color: "success", label: "Running" },
    cycling: { icon: DirectionsBikeIcon, color: "primary", label: "Cycling" },
    walking: { icon: DirectionsWalkIcon, color: "warning", label: "Walking" },
  };

  const config = activityConfig[post.workout.activity];
  const Icon = config.icon;

  const formatTime = (seconds: number) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    if (hours > 0) return `${hours}h ${minutes}m`;
    return `${minutes}m`;
  };

  const convertDistance = (km: number) => {
    if (useMetric) return `${km.toFixed(1)} km`;
    return `${(km * 0.621371).toFixed(1)} mi`;
  };

  const convertSpeed = (kmh: number) => {
    if (useMetric) return `${kmh.toFixed(1)} km/h`;
    return `${(kmh * 0.621371).toFixed(1)} mph`;
  };

  const handleKudos = async () => {
    try {
      const nowHasKudos = await toggleKudos(post.id, currentUserId);
      // Update local state optimistically
      if (nowHasKudos) {
        setKudos([...kudos, currentUserId]);
        toast.success("Kudos given!");
      } else {
        setKudos(kudos.filter(id => id !== currentUserId));
        toast.success("Kudos removed");
      }
    } catch (error) {
      console.error("Error toggling kudos:", error);
      toast.error("Failed to update kudos");
    }
  };

  const handleShare = () => {
    toast.success("Share feature coming soon!");
  };

  if (loading) {
    return (
      <Card className="p-4">
        <div className="animate-pulse space-y-4">
          <div className="flex items-center gap-3">
            <div className="w-12 h-12 bg-muted rounded-full" />
            <div className="flex-1 space-y-2">
              <div className="h-4 bg-muted rounded w-1/4" />
              <div className="h-3 bg-muted rounded w-1/3" />
            </div>
          </div>
        </div>
      </Card>
    );
  }

  if (!user) return null;

  return (
    <>
      <Card className="p-4 space-y-4 hover:shadow-elevation-2 transition-shadow">
        {/* Header */}
        <div className="flex items-center gap-3">
          <Avatar
            src={user.avatar}
```

```
            alt={user.username}
            sx={{ width: 48, height: 48 }}
            className="cursor-pointer"
            onClick={() => setShowProfile(true)}
          />
          <div className="flex-1">
            <h3
              className="font-bold cursor-pointer hover:text-primary transition-colors"
              onClick={() => setShowProfile(true)}
            >
              {user.username}
            </h3>
          <p className="text-xs text-muted-foreground">
            {formatDistanceToNow(post.timestamp, { addSuffix: true })}
          </p>
        </div>
        <div
          className={`flex items-center gap-1 px-3 py-1.5 rounded-full ${
            config.color === "success"
              ? "bg-success/10 text-success"
              : config.color === "primary"
              ? "bg-primary/10 text-primary"
              : "bg-warning/10 text-warning"
          }`}
        >
          <Icon style={{ fontSize: 16 }} />
          <span className="text-xs font-medium">{config.label}</span>
        </div>
      </div>

      {/* Caption */}
      {post.caption && <p className="text-sm">{post.caption}</p>}

      {/* Workout Stats */}
      <div className="grid grid-cols-4 gap-2 bg-muted/50 rounded-lg p-3">
        <div className="text-center">
          <div className="text-lg font-bold">{formatTime(post.workout.duration)}</div>
          <div className="text-xs text-muted-foreground">Time</div>
        </div>
        <div className="text-center">
          <div className="text-lg font-bold">{convertDistance(post.workout.distance)}</div>
          <div className="text-xs text-muted-foreground">Distance</div>
        </div>
        <div className="text-center">
          <div className="text-lg font-bold">{convertSpeed(post.workout.avgSpeed)}</div>
          <div className="text-xs text-muted-foreground">Pace</div>
        </div>
      </div>

      {/* Photos */}
      {post.photos && post.photos.length > 0 && (
        <div className={`grid gap-2 ${post.photos.length === 1 ? 'grid-cols-1' :
post.photos.length === 2 ? 'grid-cols-2' : 'grid-cols-2'}`}>
          {post.photos.slice(0, 3).map((photo, idx) => (
            <div
              key={idx}
              className={`relative rounded-lg overflow-hidden ${post.photos!.length === 3 && idx ===0
? 'col-span-2' : ''}`}
              style={{ paddingTop: post.photos!.length === 1 ? '56.25%' : '100%' }}
            >
              <img
                src={photo}
                alt={`Workout photo ${idx + 1}`}
                className="absolute inset-0 w-full h-full object-cover"
              />
            </div>
          ))}
        </div>
      )}

      {/* Location */}
      {post.workout.location && (
        <p className="text-xs text-muted-foreground">Ø=ÜÍ {post.workout.location}</p>
```

```
      )}

      {/* Actions */}
      <div className="flex items-center gap-6 pt-2 border-t border-border">
        <motion.button
          whileTap={{ scale: 0.95 }}
          onClick={handleKudos}
          className="flex items-center gap-2 text-sm hover:text-primary transition-colors"
        >
          {hasKudos ? (
            <FavoriteIcon className="text-red-500" style={{ fontSize: 20 }} />
          ) : (
            <FavoriteBorderIcon style={{ fontSize: 20 }} />
          )}
          <span className="font-medium">{kudos.length}</span>
        </motion.button>

        <motion.button
          whileTap={{ scale: 0.95 }}
          onClick={() => onCommentClick(post)}
          className="flex items-center gap-2 text-sm hover:text-primary transition-colors"
        >
          <ChatBubbleOutlineIcon style={{ fontSize: 20 }} />
          <span className="font-medium">{post.comments.length}</span>
        </motion.button>

        <motion.button
          whileTap={{ scale: 0.95 }}
          onClick={handleShare}
          className="flex items-center gap-2 text-sm hover:text-primary transition-colors ml-auto"
        >
          <ShareIcon style={{ fontSize: 20 }} />
        </motion.button>
      </div>

        {/* Kudos summary */}
        {kudos.length > 0 && (
          <p className="text-xs text-muted-foreground">
            {kudos.length === 1 ? "1 kudos" : `${kudos.length} kudos`}
          </p>
        )}
      </Card>

      {/* Profile View Modal */}
      <AnimatePresence>
        {showProfile && user && (
          <ProfileView
            user={{
              id: user.id,
              name: user.username,
              distance: "2.5 km",
              activity: post.workout.activity.charAt(0).toUpperCase() +
post.workout.activity.slice(1),
              avatar: user.avatar,
              photos: user.photos || [],
              bio: user.bio,
            }}
            friendStatus="not_friends"
            onClose={() => setShowProfile(false)}
            onSendMessage={() => {
              setShowProfile(false);
              toast.success(`Opening chat with ${user.username}`);
            }}
            onAddFriend={() => {
              toast.success(`Friend request sent to ${user.username}`);
            }}
            onAcceptFriend={() => {}}
            onDeclineFriend={() => {}}
          />
        )}
      </AnimatePresence>
    </>
```

```
280 |   );
281 | };
282 |
```

File: src/components/WorkoutSummaryModal.tsx

Lines: 510

```
  1 | import { useState, useEffect } from "react";
  2 | import { motion } from "framer-motion";
  3 | import { useNavigate } from "react-router-dom";
  4 | import { Button } from "@/components/ui/button";
  5 | import { Card } from "@/components/ui/card";
  6 | import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
  7 | import {
  8 |   AlertDialog,
  9 |   AlertDialogAction,
 10 |   AlertDialogCancel,
 11 |   AlertDialogContent,
 12 |   AlertDialogDescription,
 13 |   AlertDialogFooter,
 14 |   AlertDialogHeader,
 15 |   AlertDialogTitle,
 16 | } from "@/components/ui/alert-dialog";
 17 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
 18 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
 19 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
 20 | import TimerIcon from "@mui/icons-material/Timer";
 21 | import SpeedIcon from "@mui/icons-material/Speed";
 22 | import CloseIcon from "@mui/icons-material/Close";
 23 | import PeopleIcon from "@mui/icons-material/People";
 24 | import TouchAppIcon from "@mui/icons-material/TouchApp";
 25 | import RouteIcon from "@mui/icons-material/Route";
 26 | import { toast } from "sonner";
 27 | import { getUserData } from "@/services/authService";
 28 | import { CircularProgress } from "@mui/material";
 29 |
 30 | interface NearbyUser {
 31 |   id: string;
 32 |   name: string;
 33 |   avatar?: string;
 34 |   activity?: string;
 35 |   distance?: string;
 36 |   distanceValue?: number;
 37 |   isSameActivity?: boolean; // Flag indicating if user's activity matches workout activity
 38 | }
 39 |
 40 | interface LocationHistoryPoint {
 41 |   lat: number;
 42 |   lng: number;
 43 | }
 44 |
 45 | interface WorkoutSummaryModalProps {
 46 |   isOpen: boolean;
 47 |   onClose: () => void;
 48 |   onSave: () => void;
 49 |   onDiscard: () => void;
 50 |   activity: "running" | "cycling" | "walking";
 51 |   duration: number; // in seconds
 52 |   distance: number; // in km
 53 |   avgSpeed: number; // in km/h
 54 |   useMetric: boolean;
 55 |   nearbyUsers?: NearbyUser[];
 56 |   pokes?: string[]; // Array of user IDs who poked during workout
 57 |   locationHistory?: LocationHistoryPoint[]; // Route trail points for map visualization
 58 | }
 59 |
 60 | export const WorkoutSummaryModal = ({
 61 |   isOpen,
 62 |   onClose,
 63 |   onSave,
 64 |   onDiscard,
 65 |   activity,
 66 |   duration,
```

```
 67 |   distance,
 68 |   avgSpeed,
 69 |   useMetric,
 70 |   nearbyUsers = [],
 71 |   pokes = [],
 72 |   locationHistory = [],
 73 | }: WorkoutSummaryModalProps) => {
 74 |   const navigate = useNavigate();
 75 |   const [pokeUsers, setPokeUsers] = useState<Record<string, any>>({});
 76 |   const [loadingPokes, setLoadingPokes] = useState(false);
 77 |   const [showDiscardConfirmation, setShowDiscardConfirmation] = useState(false);
 78 |
 79 |   // Fetch user data for pokes
 80 |   useEffect(() => {
 81 |     if (!isOpen || pokes.length === 0) {
 82 |       setPokeUsers({});
 83 |       return;
 84 |     }
 85 |
 86 |     const fetchPokeUsers = async () => {
 87 |       setLoadingPokes(true);
 88 |       try {
 89 |         const userDataPromises = pokes.map(async (userId) => {
 90 |           try {
 91 |             const userData = await getUserData(userId);
 92 |             return { userId, userData };
 93 |           } catch (error) {
 94 |             console.error(`Error fetching user data for ${userId}:`, error);
 95 |             return { userId, userData: null };
 96 |           }
 97 |         });
 98 |
 99 |         const results = await Promise.all(userDataPromises);
100 |         const usersMap: Record<string, any> = {};
101 |         results.forEach(({ userId, userData }) => {
102 |           if (userData) {
103 |             usersMap[userId] = userData;
104 |           }
105 |         });
106 |         setPokeUsers(usersMap);
107 |       } catch (error) {
108 |         console.error("Error fetching poke users:", error);
109 |       } finally {
110 |         setLoadingPokes(false);
111 |       }
112 |     };
113 |
114 |     fetchPokeUsers();
115 |   }, [isOpen, pokes]);
116 |
117 |   if (!isOpen) return null;
118 |
119 |   const handleSave = () => {
120 |     onSave();
121 |   };
122 |
123 |   const handleDiscardClick = () => {
124 |     setShowDiscardConfirmation(true);
125 |   };
126 |
127 |   const handleConfirmDiscard = () => {
128 |     setShowDiscardConfirmation(false);
129 |     onDiscard();
130 |     onClose();
131 |     // Navigate back to map view
132 |     navigate("/map");
133 |   };
134 |
135 |   const formatTime = (seconds: number) => {
136 |     const hours = Math.floor(seconds / 3600);
137 |     const minutes = Math.floor((seconds % 3600) / 60);
```

```
138 |     const secs = seconds % 60;
139 |     if (hours > 0) {
140 |       return `${hours}:${minutes.toString().padStart(2, "0")}:${secs.toString().padStart(2, "0")}
`;
141 |     }
142 |     return `${minutes}:${secs.toString().padStart(2, "0")}`;
143 |   };
144 |
145 |   const convertDistance = (km: number) => {
146 |     if (useMetric) return { value: km.toFixed(2), unit: "km" };
147 |     return { value: (km * 0.621371).toFixed(2), unit: "mi" };
148 |   };
149 |
150 |   const convertSpeed = (kmh: number) => {
151 |     if (useMetric) return { value: kmh.toFixed(1), unit: "km/h" };
152 |     return { value: (kmh * 0.621371).toFixed(1), unit: "mph" };
153 |   };
154 |
155 |   const activityConfig = {
156 |     running: { icon: DirectionsRunIcon, color: "success", label: "Running" },
157 |     cycling: { icon: DirectionsBikeIcon, color: "primary", label: "Cycling" },
158 |     walking: { icon: DirectionsWalkIcon, color: "warning", label: "Walking" },
159 |   };
160 |
161 |   const config = activityConfig[activity];
162 |   const Icon = config.icon;
163 |   const distanceData = convertDistance(distance);
164 |   const speedData = convertSpeed(avgSpeed);
165 |
166 |   return (
167 |     <div className="fixed inset-0 z-[9999] flex items-center justify-center p-3 bg-background/80
backdrop-blur-sm">
168 |       <motion.div
169 |         initial={{ opacity: 0, scale: 0.9, y: 20 }}
170 |         animate={{ opacity: 1, scale: 1, y: 0 }}
171 |         exit={{ opacity: 0, scale: 0.9, y: 20 }}
172 |         className="w-full max-w-md max-h-[90vh] overflow-hidden flex flex-col"
173 |       >
174 |         <Card className="p-4 space-y-4 shadow-elevation-4 overflow-y-auto max-h-[90vh]">
175 |           {/* Header */}
176 |           <div className="flex items-center justify-between">
177 |             <div className="flex items-center gap-2">
178 |               <div
179 |                 className={`p-2 rounded-lg ${
180 |                   config.color === "success"
181 |                     ? "bg-success/15"
182 |                     : config.color === "primary"
183 |                     ? "bg-primary/15"
184 |                     : "bg-warning/15"
185 |                 }`}
186 |               >
187 |                 <Icon
188 |                   className={
189 |                     config.color === "success"
190 |                       ? "text-success"
191 |                       : config.color === "primary"
192 |                       ? "text-primary"
193 |                       : "text-warning"
194 |                   }
195 |                   style={{ fontSize: 24 }}
196 |                 />
197 |               </div>
198 |               <div>
199 |                 <h2 className="text-xl font-bold">Workout Complete!</h2>
200 |                 <p className="text-xs text-muted-foreground">{config.label}</p>
201 |               </div>
202 |             </div>
203 |             <button
204 |               onClick={onClose}
205 |               className="p-1.5 hover:bg-accent rounded-full transition-colors"
206 |             >
207 |               <CloseIcon className="text-muted-foreground" style={{ fontSize: 20 }} />
208 |             </button>
```

```
          </div>

          {/* Nearby People Section - Most Prominent, At Top */}
          {nearbyUsers.length > 0 && (
            <motion.div
              initial={{ opacity: 0, y: 10 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ delay: 0.2 }}
              className="space-y-2 bg-gradient-to-br from-primary/15 via-success/10 to-primary/5 rounded-lg p-3 border b...
              >
              <div className="flex items-center gap-2">
                <div className="p-1.5 bg-primary/20 rounded-md">
                  <PeopleIcon className="text-primary" style={{ fontSize: 18 }} />
                </div>
                <div>
                  <h3 className="text-sm font-bold">Nearby People ({nearbyUsers.length})</h3>
                  <p className="text-xs text-muted-foreground">Found during workout</p>
                </div>
              </div>
              <div className="space-y-2 max-h-64 overflow-y-auto">
                {nearbyUsers.map((user, index) => {
                  const ActivityIcon =
                    user.activity === "running" ? DirectionsRunIcon :
                    user.activity === "cycling" ? DirectionsBikeIcon :
                    DirectionsWalkIcon;

                  // Check if user's activity matches workout activity
                  // Use isSameActivity flag if available, otherwise compare directly
                  const isSameActivity = user.isSameActivity !== undefined
                    ? user.isSameActivity
                    : user.activity && user.activity.toLowerCase() === activity.toLowerCase();
                  const shouldGreyOut = !isSameActivity;

                  return (
                    <motion.div
                      key={user.id}
                      initial={{ opacity: 0, x: -10 }}
                      animate={{ opacity: 1, x: 0 }}
                      transition={{ delay: 0.2 + (index * 0.03) }}
                      className={`flex items-center gap-2.5 p-2.5 bg-card/90 rounded-lg border transition-all ${
                        shouldGreyOut
                          ? "opacity-50 border-border/30 hover:border-border/50 hover:bg-card/70"
                          : "border-border/50 hover:border-primary/50 hover:bg-card"
                      }`}
                    >
                      <Avatar className={`w-12 h-12 border-2 flex-shrink-0 ${
                        shouldGreyOut ? "border-border/30" : "border-primary/50"
                      }`}>
                        <AvatarImage src={user.avatar || `https://ui-avatars.com/api/?name=${encodeURIComponent(user.nam...
                        <AvatarFallback className="text-sm font-bold">{user.name.charAt(0).toUpperCase()}</AvatarFallbac...
                      </Avatar>
                      <div className="flex-1 min-w-0">
                        <p className={`text-sm font-semibold truncate ${
                          shouldGreyOut ? "text-muted-foreground" : ""
                        }`}>{user.name}</p>
                        <div className="flex items-center gap-1.5 mt-0.5 flex-wrap">
                          {user.activity && (
                            <span className={`text-xs capitalize flex items-center gap-1 px-1.5 py-0.5 rounded ${
                              shouldGreyOut
                                ? "text-muted-foreground bg-muted/30"
                                : "text-foreground bg-muted/50"
                            }`}>
                              <ActivityIcon style={{ fontSize: 12 }} />
                              {user.activity}
                            </span>
                          )}
                          {user.distance && (
                            <span className="text-xs text-muted-foreground px-1.5 py-0.5 bg-muted/30 rounded">
                              {user.distance}
                            </span>
                          )}
```

```
                        </div>
                      </div>
                    </motion.div>
                  );
                })}
              </div>
            </motion.div>
          )}

          {/* Workout Stats - Compact Display */}
          <motion.div
            initial={{ opacity: 0, y: 10 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: 0.3 }}
            className="bg-muted/30 rounded-lg p-3 border border-border/50"
          >
            <h4 className="text-xs font-semibold text-muted-foreground mb-2 uppercase tracking-wide">Summary</h4> {/* Stats Grid */}
            <div className="grid grid-cols-3 gap-2">
              {/* Total Time */}
              <div className="text-center">
                <TimerIcon className="text-primary mx-auto mb-0.5" style={{ fontSize: 16 }} />
                <p className="text-base font-bold tabular-nums">
                  {formatTime(duration)}
                </p>
                <p className="text-[10px] text-muted-foreground">Time</p>
              </div>

              {/* Distance */}
              <div className="text-center">
                <Icon
                  className={
                    config.color === "success"
                      ? "text-success mx-auto mb-0.5"
                      : config.color === "primary"
                      ? "text-primary mx-auto mb-0.5"
                      : "text-warning mx-auto mb-0.5"
                  }
                  style={{ fontSize: 16 }}
                />
                <p className="text-base font-bold tabular-nums">
                  {distanceData.value}
                </p>
                <p className="text-[10px] text-muted-foreground">{distanceData.unit}</p>
              </div>

              {/* Average Speed */}
              <div className="text-center">
                <SpeedIcon className="text-success mx-auto mb-0.5" style={{ fontSize: 16 }} />
                <p className="text-base font-bold tabular-nums">
                  {speedData.value}
                </p>
                <p className="text-[10px] text-muted-foreground">{speedData.unit}</p>
              </div>
            </div>
          </motion.div>

          {/* Route Map Visualization */}
          {locationHistory.length > 1 && (() => {
            // Calculate route bounds
            const lats = locationHistory.map(p => p.lat);
            const lngs = locationHistory.map(p => p.lng);
            const minLat = Math.min(...lats);
            const maxLat = Math.max(...lats);
            const minLng = Math.min(...lngs);
            const maxLng = Math.max(...lngs);

            // Calculate center
            const centerLat = (minLat + maxLat) / 2;
            const centerLng = (minLng + maxLng) / 2;

```

```
            // Create encoded polyline path for Google Maps Static API
            // Simple encoding: convert lat/lng to string format
            const pathPoints = locationHistory.map(p => `${p.lat},${p.lng}`).join('|');

            // Get Google Maps API key from environment
            const apiKey = import.meta.env.VITE_GOOGLE_MAPS_API_KEY || '';

            // Build static map URL
            const staticMapUrl = apiKey
              ? `https://maps.googleapis.com/maps/api/staticmap?size=400x200&zoom=13&path=color:0x1976d2|weight:4|${path...
              : null;

            return (
              <motion.div
                initial={{ opacity: 0, y: 10 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ delay: 0.4 }}
                className="space-y-2"
              >
                <div className="flex items-center gap-2">
                  <RouteIcon className="text-primary" style={{ fontSize: 16 }} />
                  <h3 className="font-semibold text-xs">Route Map</h3>
                </div>
                <Card className="overflow-hidden border border-border/50">
                  {staticMapUrl ? (
                    <div className="relative w-full" style={{ paddingTop: '50%' }}>
                      <img
                        src={staticMapUrl}
                        alt="Workout route"
                        className="absolute inset-0 w-full h-full object-cover"
                        onError={(e) => {
                          // Fallback if image fails to load
                          e.currentTarget.style.display = 'none';
                          const fallback = e.currentTarget.nextElementSibling as HTMLElement;
                          if (fallback) fallback.style.display = 'flex';
                        }}
                      />
                      <div className="absolute inset-0 bg-gradient-to-br from-primary/10 via-success/5 to-warning/10 fle...
                        <div>
                          <RouteIcon className="text-muted-foreground mx-auto mb-1" style={{ fontSize: 32 }} />
                          <p className="text-xs text-muted-foreground">Route visualization</p>
                          <p className="text-[10px] text-muted-foreground mt-0.5">{locationHistory.length} points tracke...
                        </div>
                      </div>
                    </div>
                  ) : (
                    <div className="bg-gradient-to-br from-primary/10 via-success/5 to-warning/10 rounded-lg h-32 flex i...
                      <div>
                        <RouteIcon className="text-muted-foreground mx-auto mb-1" style={{ fontSize: 32 }} />
                        <p className="text-xs text-muted-foreground">Route visualization</p>
                        <p className="text-[10px] text-muted-foreground mt-0.5">{locationHistory.length} points tracked<...
                      </div>
                    </div>
                  )}
                </Card>
              </motion.div>
            );
          })()}

          {/* Pokes Received Section */}
          {pokes.length > 0 && (
            <motion.div
              initial={{ opacity: 0, y: 10 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ delay: 0.3 }}
              className="space-y-2"
            >
              <div className="flex items-center gap-2">
                <TouchAppIcon className="text-primary" style={{ fontSize: 16 }} />
                <h3 className="font-semibold text-xs">Pokes ({pokes.length})</h3>
              </div>
```

```
              {loadingPokes ? (
                <div className="flex items-center justify-center py-2">
                  <CircularProgress size={20 } />
                </div>
              ) : (
                <div className="space-y-1.5 max-h-32 overflow-y-auto">
                  {pokes.map((pokeUserId) => {
                    const userData = pokeUsers[pokeUserId];
                    const userName = userData?.name || "Unknown User";
                    const userAvatar = userData?.photoURL;

                    return (
                      <motion.div
                        key={pokeUserId}
                        initial={{ opacity: 0, x: -10 }}
                        animate={{ opacity: 1, x: 0 }}
                        className="flex items-center gap-2 p-2 bg-muted/50 rounded-lg border
border-border"
                      >
                        <Avatar className="w-8 h-8">
                          <AvatarImage src={userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(userNam<AvatarFallback className="text-xs">{userName.charAt(0).toUpperCase()}
</AvatarFallback>
                        </Avatar>
                        <div className="flex-1 min-w-0">
                          <p className="text-xs font-medium truncate">{userName}</p>
                          <p className="text-[10px] text-muted-foreground">Poked you</p>
                        </div>
                      </motion.div>
                    );
                  })}
                </div>
              )}
            </motion.div>
          )}

          {/* Empty State */}
          {pokes.length === 0 && nearbyUsers.length === 0 && (
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              transition={{ delay: 0.2 }}
              className="text-center py-4"
            >
              <PeopleIcon className="text-muted-foreground mx-auto mb-1" style={{ fontSize:
32 }} />
              <p className="text-xs text-muted-foreground">No nearby people or pokes during this
workout</p>
            </motion.div>
          )}

          {/* Action Buttons */}
          <div className="flex flex-col gap-2 pt-2">
            <Button
              onClick={handleSave}
              className="w-full h-11 text-sm font-bold bg-blue-600 hover:bg-blue-700 text-white"
            >
              Save Workout
            </Button>
            <Button
              onClick={handleDiscardClick}
              className="w-full h-11 text-sm font-bold bg-red-600 hover:bg-red-700 text-white"
            >
              Discard
            </Button>
          </div>

          {/* Discard Confirmation Dialog */}
          <AlertDialog open={showDiscardConfirmation} onOpenChange={setShowDiscardConfirmation}>
            <AlertDialogContent>
              <AlertDialogHeader>
                <AlertDialogTitle>Discard Workout?</AlertDialogTitle>
                <AlertDialogDescription>
                  Are you sure you want to discard this workout? All data will be lost and
cannot be recovered.
                </AlertDialogDescription>
```

```
              </AlertDialogHeader>
              <AlertDialogFooter>
                <AlertDialogCancel>Cancel</AlertDialogCancel>
                <AlertDialogAction
                  onClick={handleConfirmDiscard}
                  className="bg-red-600 hover:bg-red-700 text-white"
                >
                  Discard
                </AlertDialogAction>
              </AlertDialogFooter>
            </AlertDialogContent>
          </AlertDialog>
        </Card>
      </motion.div>
    </div>
  );
};

```

File: src/components/ui/accordion.tsx

Lines: 53

```
import * as React from "react";
import * as AccordionPrimitive from "@radix-ui/react-accordion";
import { ChevronDown } from "lucide-react";

import { cn } from "@/lib/utils";

const Accordion = AccordionPrimitive.Root;

const AccordionItem = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Item>
>(({ className, ...props }, ref) => (
  <AccordionPrimitive.Item ref={ref} className={cn("border-b", className)} {...props} />
));
AccordionItem.displayName = "AccordionItem";

const AccordionTrigger = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Trigger>
>(({ className, children, ...props }, ref) => (
  <AccordionPrimitive.Header className="flex">
    <AccordionPrimitive.Trigger
      ref={ref}
      className={cn(
        "flex flex-1 items-center justify-between py-4 font-medium transition-all
hover:underline [&[data-state=open]>sv...
        className,
      )}
      {...props}
    >
      {children}
      <ChevronDown className="h-4 w-4 shrink-0 transition-transform duration-200" />
    </AccordionPrimitive.Trigger>
  </AccordionPrimitive.Header>
));
AccordionTrigger.displayName = AccordionPrimitive.Trigger.displayName;

const AccordionContent = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Content>
>(({ className, children, ...props }, ref) => (
  <AccordionPrimitive.Content
    ref={ref}
    className="overflow-hidden text-sm transition-all data-[state=closed]:animate-accordion-up
data-[state=open]:animate...
    {...props}
  >
    <div className={cn("pb-4 pt-0", className)}>{children}</div>
  </AccordionPrimitive.Content>
));

AccordionContent.displayName = AccordionPrimitive.Content.displayName;

export { Accordion, AccordionItem, AccordionTrigger, AccordionContent };

```

File: src/components/ui/alert-dialog.tsx

Lines: 105

```
import * as React from "react";
import * as AlertDialogPrimitive from "@radix-ui/react-alert-dialog";

import { cn } from "@/lib/utils";
import { buttonVariants } from "@/components/ui/button";

const AlertDialog = AlertDialogPrimitive.Root;

const AlertDialogTrigger = AlertDialogPrimitive.Trigger;

const AlertDialogPortal = AlertDialogPrimitive.Portal;

const AlertDialogOverlay = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Overlay>
>(({ className, ...props }, ref) => (
  <AlertDialogPrimitive.Overlay
    className={cn(
      "fixed inset-0 z-[9998] bg-black/80 data-[state=open]:animate-in data-
[state=closed]:animate-out data-[state=close...
      className,
    )}
    {...props}
    ref={ref}
  />
));
AlertDialogOverlay.displayName = AlertDialogPrimitive.Overlay.displayName;

const AlertDialogContent = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Content>
>(({ className, ...props }, ref) => (
  <AlertDialogPortal>
    <AlertDialogOverlay />
    <AlertDialogPrimitive.Content
      ref={ref}
      className={cn(
        "fixed left-[50%] top-[50%] z-[9999] grid w-full max-w-lg translate-x-[-50%] translate-y-
[-50%] gap-4 border bg-
      className,.
      )}
      {...props}
    />
  </AlertDialogPortal>
));
AlertDialogContent.displayName = AlertDialogPrimitive.Content.displayName;

const AlertDialogHeader = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("flex flex-col space-y-2 text-center sm:text-left", className)} {...props} />
);
AlertDialogHeader.displayName = "AlertDialogHeader";

const AlertDialogFooter = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
className)} {...props} />
);
AlertDialogFooter.displayName = "AlertDialogFooter";

const AlertDialogTitle = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Title>,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Title>
>(({ className, ...props }, ref) => (
  <AlertDialogPrimitive.Title ref={ref} className={cn("text-lg font-semibold", className)}
{...props} />
));
AlertDialogTitle.displayName = AlertDialogPrimitive.Title.displayName;

const AlertDialogDescription = React.forwardRef<
  React.ElementRef<typeof AlertDialogPrimitive.Description>,
  React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Description>
```

```
  67 | >(({ className, ...props }, ref) => (
  68 |   <AlertDialogPrimitive.Description ref={ref} className={cn("text-sm text-muted-foreground",
className)} {...props} />
  69 | ));
  70 | AlertDialogDescription.displayName = AlertDialogPrimitive.Description.displayName;
  71 |
  72 | const AlertDialogAction = React.forwardRef<
  73 |   React.ElementRef<typeof AlertDialogPrimitive.Action>,
  74 |   React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Action>
  75 | >(({ className, ...props }, ref) => (
  76 |   <AlertDialogPrimitive.Action ref={ref} className={cn(buttonVariants(), className)} {...props} /
> 77 | ));
  78 | AlertDialogAction.displayName = AlertDialogPrimitive.Action.displayName;
  79 |
  80 | const AlertDialogCancel = React.forwardRef<
  81 |   React.ElementRef<typeof AlertDialogPrimitive.Cancel>,
  82 |   React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Cancel>
  83 | >(({ className, ...props }, ref) => (
  84 |   <AlertDialogPrimitive.Cancel
  85 |     ref={ref}
  86 |     className={cn(buttonVariants({ variant: "outline" }), "mt-2 sm:mt-0", className)}
  87 |     {...props}
  88 |   />
  89 | ));
  90 | AlertDialogCancel.displayName = AlertDialogPrimitive.Cancel.displayName;
  91 |
  92 | export {
  93 |   AlertDialog,
  94 |   AlertDialogPortal,
  95 |   AlertDialogOverlay,
  96 |   AlertDialogTrigger,
  97 |   AlertDialogContent,
  98 |   AlertDialogHeader,
  99 |   AlertDialogFooter,
 100 |   AlertDialogTitle,
 101 |   AlertDialogDescription,
 102 |   AlertDialogAction,
 103 |   AlertDialogCancel,
 104 | };
 105 |
```

File: src/components/ui/alert.tsx

Lines: 44

```
   1 | import * as React from "react";
   2 | import { cva, type VariantProps } from "class-variance-authority";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const alertVariants = cva(
   7 |   "relative w-full rounded-lg border p-4 [&>svg~*]:pl-7 [&>svg+div]:translate-y-[-3px]
[&>svg]:absolute [&>svg]:left-4 [...
   8 |   {
   9 |     variants: {
  10 |       variant: {
  11 |         default: "bg-background text-foreground",
  12 |         destructive: "border-destructive/50 text-destructive dark:border-destructive
[&>svg]:text-destructive",
  13 |       },
  14 |     },
  15 |     defaultVariants: {
  16 |       variant: "default",
  17 |     },
  18 |   },
  19 | );
  20 |
  21 | const Alert = React.forwardRef<
  22 |   HTMLDivElement,
  23 |   React.HTMLAttributes<HTMLDivElement> & VariantProps<typeof alertVariants>
  24 | >(({ className, variant, ...props }, ref) => (
  25 |   <div ref={ref} role="alert" className={cn(alertVariants({ variant }), className)} {...props} />
  26 | ));
  27 | Alert.displayName = "Alert";
  28 |
  29 | const AlertTitle = React.forwardRef<HTMLParagraphElement,
  30 |   React.HTMLAttributes<HTMLHeadingElement>>(({ className, ...props }, ref) => (
  31 |     <h5 ref={ref} className={cn("mb-1 font-medium leading-none tracking-tight", className)}
{...props} />
  32 |   ),
  33 | );
  34 | AlertTitle.displayName = "AlertTitle";
  35 |
  36 | const AlertDescription = React.forwardRef<HTMLParagraphElement,
  37 |   React.HTMLAttributes<HTMLParagraphElement>>(({ className, ...props }, ref) => (
  38 |     <div ref={ref} className={cn("text-sm [&_p]:leading-relaxed", className)} {...props} />
  39 |   ),
  40 | );
  41 | AlertDescription.displayName = "AlertDescription";
  42 |
  43 | export { Alert, AlertTitle, AlertDescription };
  44 |
```

File: src/components/ui/aspect-ratio.tsx

Lines: 6

```
1 | import * as AspectRatioPrimitive from "@radix-ui/react-aspect-ratio";
2 |
3 | const AspectRatio = AspectRatioPrimitive.Root;
4 |
5 | export { AspectRatio };
6 |
```

File: src/components/ui/avatar.tsx

Lines: 43

```
  1 | import * as React from "react";
  2 | import * as AvatarPrimitive from "@radix-ui/react-avatar";
  3 |
  4 | import { cn } from "@/lib/utils";
  5 |
  6 | const Avatar = React.forwardRef<
  7 |   React.ElementRef<typeof AvatarPrimitive.Root>,
  8 |   React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>
  9 | >(({ className, ...props }, ref) => (
 10 |   <AvatarPrimitive.Root
 11 |     ref={ref}
 12 |     className={cn("relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full", className)}
 13 |     {...props}
 14 |   />
 15 | ));
 16 | Avatar.displayName = AvatarPrimitive.Root.displayName;
 17 |
 18 | const AvatarImage = React.forwardRef<
 19 |   React.ElementRef<typeof AvatarPrimitive.Image>,
 20 |   React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>
 21 | >(({ className, ...props }, ref) => (
 22 |   <AvatarPrimitive.Image
 23 |     ref={ref}
 24 |     className={cn("aspect-square h-full w-full object-cover", className)}
 25 |     {...props}
 26 |   />
 27 | ));
 28 | AvatarImage.displayName = AvatarPrimitive.Image.displayName;
 29 |
 30 | const AvatarFallback = React.forwardRef<
 31 |   React.ElementRef<typeof AvatarPrimitive.Fallback>,
 32 |   React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>
 33 | >(({ className, ...props }, ref) => (
 34 |   <AvatarPrimitive.Fallback
 35 |     ref={ref}
 36 |     className={cn("flex h-full w-full items-center justify-center rounded-full bg-muted",
className)}{...props}
 38 |   />
 39 | ));
 40 | AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName;
 41 |
 42 | export { Avatar, AvatarImage, AvatarFallback };
 43 |
```

File: src/components/ui/badge.tsx

Lines: 30

```
   1 | import * as React from "react";
   2 | import { cva, type VariantProps } from "class-variance-authority";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const badgeVariants = cva(
   7 |   "inline-flex items-center rounded-full border px-2.5 py-0.5 text-xs font-semibold transition-colors focus:outline-none...
   8 |   {
   9 |     variants: {
  10 |       variant: {
  11 |         default: "border-transparent bg-primary text-primary-foreground hover:bg-primary/80",
  12 |         secondary: "border-transparent bg-secondary text-secondary-foreground hover:bg-secondary/80",
  13 |         destructive: "border-transparent bg-destructive text-destructive-foreground hover:bg-destructive/80",
  14 |         outline: "text-foreground",
  15 |       },
  16 |     },
  17 |     defaultVariants: {
  18 |       variant: "default",
  19 |     },
  20 |   },
  21 | );
  22 |
  23 | export interface BadgeProps extends React.HTMLAttributes<HTMLDivElement>, VariantProps<typeof badgeVariants> {}
  24 |
  25 | function Badge({ className, variant, ...props }: BadgeProps) {
  26 |   return <div className={cn(badgeVariants({ variant }), className)} {...props} />;
  27 | }
  28 |
  29 | export { Badge, badgeVariants };
  30 |
```

File: src/components/ui/breadcrumb.tsx

Lines: 91

```
import * as React from "react";
import { Slot } from "@radix-ui/react-slot";
import { ChevronRight, MoreHorizontal } from "lucide-react";

import { cn } from "@/lib/utils";

const Breadcrumb = React.forwardRef<
  HTMLElement,
  React.ComponentPropsWithoutRef<"nav"> & {
    separator?: React.ReactNode;
  }
>(({ ...props }, ref) => <nav ref={ref} aria-label="breadcrumb" {...props} />);
Breadcrumb.displayName = "Breadcrumb";

const BreadcrumbList = React.forwardRef<HTMLOListElement, React.ComponentPropsWithoutRef<"ol">>(
  ({ className, ...props }, ref) => (
    <ol
      ref={ref}
      className={cn(
        "flex flex-wrap items-center gap-1.5 break-words text-sm text-muted-foreground
sm:gap-2.5",
        className,
      )}
      {...props}
    />
  ),
);
BreadcrumbList.displayName = "BreadcrumbList";

const BreadcrumbItem = React.forwardRef<HTMLLIElement, React.ComponentPropsWithoutRef<"li">>(
  ({ className, ...props }, ref) => (
    <li ref={ref} className={cn("inline-flex items-center gap-1.5", className)} {...props} />
  ),
);
BreadcrumbItem.displayName = "BreadcrumbItem";

const BreadcrumbLink = React.forwardRef<
  HTMLAnchorElement,
  React.ComponentPropsWithoutRef<"a"> & {
    asChild?: boolean;
  }
>({ asChild, className, ...props }, ref) => {
  const Comp = asChild ? Slot : "a";

  return <Comp ref={ref} className={cn("transition-colors hover:text-foreground", className)}
{...props} />;
});
BreadcrumbLink.displayName = "BreadcrumbLink";

const BreadcrumbPage = React.forwardRef<HTMLSpanElement, React.ComponentPropsWithoutRef<"span">>(
  ({ className, ...props }, ref) => (
    <span
      ref={ref}
      role="link"
      aria-disabled="true"
      aria-current="page"
      className={cn("font-normal text-foreground", className)}
      {...props}
    />
  ),
);
BreadcrumbPage.displayName = "BreadcrumbPage";

const BreadcrumbSeparator = ({ children, className, ...props }: React.ComponentProps<"li">) => (
  <li role="presentation" aria-hidden="true" className={cn("[&>svg]:size-3.5", className)}
{...props}>{children ?? <ChevronRight />}
  </li>
);
```

```
67 | BreadcrumbSeparator.displayName = "BreadcrumbSeparator";
68 |
69 | const BreadcrumbEllipsis = ({ className, ...props }: React.ComponentProps<"span">) => (
70 |   <span
71 |     role="presentation"
72 |     aria-hidden="true"
73 |     className={cn("flex h-9 w-9 items-center justify-center", className)}
74 |     {...props}
75 |   >
76 |     <MoreHorizontal className="h-4 w-4" />
77 |     <span className="sr-only">More</span>
78 |   </span>
79 | );
80 | BreadcrumbEllipsis.displayName = "BreadcrumbElipssis";
81 |
82 | export {
83 |   Breadcrumb,
84 |   BreadcrumbList,
85 |   BreadcrumbItem,
86 |   BreadcrumbLink,
87 |   BreadcrumbPage,
88 |   BreadcrumbSeparator,
89 |   BreadcrumbEllipsis,
90 | };
91 |
```

File: src/components/ui/button.tsx

Lines: 48

```
   1 | import * as React from "react";
   2 | import { Slot } from "@radix-ui/react-slot";
   3 | import { cva, type VariantProps } from "class-variance-authority";
   4 |
   5 | import { cn } from "@/lib/utils";
   6 |
   7 | const buttonVariants = cva(
   8 |   "inline-flex items-center justify-center gap-2 whitespace-nowrap rounded-md text-sm font-
medium ring-offset-background...
   9 |     {
  10 |       variants: {
  11 |         variant: {
  12 |           default: "bg-primary text-primary-foreground hover:bg-primary/90",
  13 |           destructive: "bg-destructive text-destructive-foreground hover:bg-destructive/90",
  14 |           outline: "border border-input bg-background hover:bg-accent hover:text-accent-
foreground",
  15 |           secondary: "bg-secondary text-secondary-foreground hover:bg-secondary/80",
  16 |           ghost: "hover:bg-accent hover:text-accent-foreground",
  17 |           link: "text-primary underline-offset-4 hover:underline",
  18 |         },
  19 |         size: {
  20 |           default: "h-10 px-4 py-2",
  21 |           sm: "h-9 rounded-md px-3",
  22 |           lg: "h-11 rounded-md px-8",
  23 |           icon: "h-10 w-10",
  24 |         },
  25 |       },
  26 |       defaultVariants: {
  27 |         variant: "default",
  28 |         size: "default",
  29 |       },
  30 |     },
  31 | );
  32 |
  33 | export interface ButtonProps
  34 |     extends React.ButtonHTMLAttributes<HTMLButtonElement>,
  35 |       VariantProps<typeof buttonVariants> {
  36 |     asChild?: boolean;
  37 | }
  38 |
  39 | const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  40 |     ({ className, variant, size, asChild = false, ...props }, ref) => {
  41 |       const Comp = asChild ? Slot : "button";
  42 |       return <Comp className={cn(buttonVariants({ variant, size, className }))} ref={ref}
{...props},/>;
  43 |     },
  44 | );
  45 | Button.displayName = "Button";
  46 |
  47 | export { Button, buttonVariants };
  48 |
```

File: src/components/ui/calendar.tsx

Lines: 55

```
import * as React from "react";
import { ChevronLeft, ChevronRight } from "lucide-react";
import { DayPicker } from "react-day-picker";

import { cn } from "@/lib/utils";
import { buttonVariants } from "@/components/ui/button";

export type CalendarProps = React.ComponentProps<typeof DayPicker>;

function Calendar({ className, classNames, showOutsideDays = true, ...props }: CalendarProps) {
  return (
    <DayPicker
      showOutsideDays={showOutsideDays}
      className={cn("p-3", className)}
      classNames={{
        months: "flex flex-col sm:flex-row space-y-4 sm:space-x-4 sm:space-y-0",
        month: "space-y-4",
        caption: "flex justify-center pt-1 relative items-center",
        caption_label: "text-sm font-medium",
        nav: "space-x-1 flex items-center",
        nav_button: cn(
          buttonVariants({ variant: "outline" }),
          "h-7 w-7 bg-transparent p-0 opacity-50 hover:opacity-100",
        ),
        nav_button_previous: "absolute left-1",
        nav_button_next: "absolute right-1",
        table: "w-full border-collapse space-y-1",
        head_row: "flex",
        head_cell: "text-muted-foreground rounded-md w-9 font-normal text-[0.8rem]",
        row: "flex w-full mt-2",
        cell: "h-9 w-9 text-center text-sm p-0 relative [&:has([aria-selected].day-range-end)]:rounded-r-md [&:has([aria-selected])]:bg-accent first:[&:has([aria-selected])]:rounded-l-md last:[&:has([aria-selected])]:rounded-r-md focus-within:relative focus-within:z-20",
        day: cn(buttonVariants({ variant: "ghost" }), "h-9 w-9 p-0 font-normal aria-selected:opacity-100"),
        day_range_end: "day-range-end",
        day_selected:
          "bg-primary text-primary-foreground hover:bg-primary hover:text-primary-foreground focus:bg-primary focus:text-primary-foreground",
        day_today: "bg-accent text-accent-foreground",
        day_outside:
          "day-outside text-muted-foreground opacity-50 aria-selected:bg-accent/50 aria-selected:text-muted-foreground aria-selected:opacity-30",
        day_disabled: "text-muted-foreground opacity-50",
        day_range_middle: "aria-selected:bg-accent aria-selected:text-accent-foreground",
        day_hidden: "invisible",
        ...classNames,
      }}
      components={{
        IconLeft: ({ ..._props }) => <ChevronLeft className="h-4 w-4" />,
        IconRight: ({ ..._props }) => <ChevronRight className="h-4 w-4" />,
      }}
      {...props}
    />
  );
}
Calendar.displayName = "Calendar";

export { Calendar };
```

File: src/components/ui/card.tsx

Lines: 44

```
import * as React from "react";

import { cn } from "@/lib/utils";

const Card = React.forwardRef<HTMLDivElement,
React.HTMLAttributes<HTMLDivElement>>(({ className, ...props }, ref) => (
  <div ref={ref} className={cn("rounded-lg border bg-card text-card-foreground shadow-sm",
className)} {...props} />
));
Card.displayName = "Card";

const CardHeader = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => (
    <div ref={ref} className={cn("flex flex-col space-y-1.5 p-6", className)} {...props} />
  ),
);
CardHeader.displayName = "CardHeader";

const CardTitle = React.forwardRef<HTMLParagraphElement,
React.HTMLAttributes<HTMLHeadingElement>>(
  ({ className, ...props }, ref) => (
    <h3 ref={ref} className={cn("text-2xl font-semibold leading-none tracking-tight",
className)} {...props} />
  ),
);
CardTitle.displayName = "CardTitle";

const CardDescription = React.forwardRef<HTMLParagraphElement,
React.HTMLAttributes<HTMLParagraphElement>>(
  ({ className, ...props }, ref) => (
    <p ref={ref} className={cn("text-sm text-muted-foreground", className)} {...props} />
  ),
);
CardDescription.displayName = "CardDescription";

const CardContent = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => <div ref={ref} className={cn("p-6 pt-0", className)}
{...props} />,
);
CardContent.displayName = "CardContent";

const CardFooter = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => (
    <div ref={ref} className={cn("flex items-center p-6 pt-0", className)} {...props} />
  ),
);
CardFooter.displayName = "CardFooter";

export { Card, CardHeader, CardFooter, CardTitle, CardDescription, CardContent };

```

File: src/components/ui/carousel.tsx

Lines: 225

```tsx
import * as React from "react";
import useEmblaCarousel, { type UseEmblaCarouselType } from "embla-carousel-react";
import { ArrowLeft, ArrowRight } from "lucide-react";

import { cn } from "@/lib/utils";
import { Button } from "@/components/ui/button";

type CarouselApi = UseEmblaCarouselType[1];
type UseCarouselParameters = Parameters<typeof useEmblaCarousel>;
type CarouselOptions = UseCarouselParameters[0];
type CarouselPlugin = UseCarouselParameters[1];

type CarouselProps = {
  opts?: CarouselOptions;
  plugins?: CarouselPlugin;
  orientation?: "horizontal" | "vertical";
  setApi?: (api: CarouselApi) => void;
};

type CarouselContextProps = {
  carouselRef: ReturnType<typeof useEmblaCarousel>[0];
  api: ReturnType<typeof useEmblaCarousel>[1];
  scrollPrev: () => void;
  scrollNext: () => void;
  canScrollPrev: boolean;
  canScrollNext: boolean;
} & CarouselProps;

const CarouselContext = React.createContext<CarouselContextProps | null>(null);

function useCarousel() {
  const context = React.useContext(CarouselContext);

  if (!context) {
    throw new Error("useCarousel must be used within a <Carousel />");
  }

  return context;
}

const Carousel = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement> &
CarouselProps>(
{ orientation = "horizontal", opts, setApi, plugins, className, children, ...props }, ref) =>
    const [carouselRef, api] = useEmblaCarousel(
      {
        ...opts,
        axis: orientation === "horizontal" ? "x" : "y",
      },
      plugins,
    );
    const [canScrollPrev, setCanScrollPrev] = React.useState(false);
    const [canScrollNext, setCanScrollNext] = React.useState(false);

    const onSelect = React.useCallback((api: CarouselApi) => {
      if (!api) {
        return;
      }

      setCanScrollPrev(api.canScrollPrev());
      setCanScrollNext(api.canScrollNext());
    }, []);

    const scrollPrev = React.useCallback(() => {
      api?.scrollPrev();
    }, [api]);

    const scrollNext = React.useCallback(() => {
```

```
      api?.scrollNext();
    }, [api]);

    const handleKeyDown = React.useCallback(
      (event: React.KeyboardEvent<HTMLDivElement>) => {
        if (event.key === "ArrowLeft") {
          event.preventDefault();
          scrollPrev();
        } else if (event.key === "ArrowRight") {
          event.preventDefault();
          scrollNext();
        }
      },
      [scrollPrev, scrollNext],
    );

    React.useEffect(() => {
      if (!api || !setApi) {
        return;
      }

      setApi(api);
    }, [api, setApi]);

    React.useEffect(() => {
      if (!api) {
        return;
      }

      onSelect(api);
      api.on("reInit", onSelect);
      api.on("select", onSelect);

      return () => {
        api?.off("select", onSelect);
      };
    }, [api, onSelect]);

    return (
      <CarouselContext.Provider
        value={{
          carouselRef,
          api: api,
          opts,
          orientation: orientation || (opts?.axis === "y" ? "vertical" : "horizontal"),
          scrollPrev,
          scrollNext,
          canScrollPrev,
          canScrollNext,
        }}
      >
        <div
          ref={ref}
          onKeyDownCapture={handleKeyDown}
          className={cn("relative", className)}
          role="region"
          aria-roledescription="carousel"
          {...props}
        >
          {children}
        </div>
      </CarouselContext.Provider>
    );
  },
);
Carousel.displayName = "Carousel";

const CarouselContent = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => {
    const { carouselRef, orientation } = useCarousel();

```

```
    return (
      <div ref={carouselRef} className="overflow-hidden">
        <div
          ref={ref}
          className={cn("flex", orientation === "horizontal" ? "-ml-4" : "-mt-4 flex-col",
className)}
          {...props}
        />
      </div>
    );
  },
);
CarouselContent.displayName = "CarouselContent";

const CarouselItem = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => {
    const { orientation } = useCarousel();

    return (
      <div
        ref={ref}
        role="group"
        aria-roledescription="slide"
        className={cn("min-w-0 shrink-0 grow-0 basis-full", orientation === "horizontal" ?
"pl-4" : "pt-4", className)}
        {...props}
      />
    );
  },
);
CarouselItem.displayName = "CarouselItem";

const CarouselPrevious = React.forwardRef<HTMLButtonElement, React.ComponentProps<typeof
Button>>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
    const { orientation, scrollPrev, canScrollPrev } = useCarousel();

    return (
      <Button
        ref={ref}
        variant={variant}
        size={size}
        className={cn(
          "absolute h-8 w-8 rounded-full",
          orientation === "horizontal"
            ? "-left-12 top-1/2 -translate-y-1/2"
            : "-top-12 left-1/2 -translate-x-1/2 rotate-90",
          className,
        )}
        disabled={!canScrollPrev}
        onClick={scrollPrev}
        {...props}
      >
        <ArrowLeft className="h-4 w-4" />
        <span className="sr-only">Previous slide</span>
      </Button>
    );
  },
);
CarouselPrevious.displayName = "CarouselPrevious";

const CarouselNext = React.forwardRef<HTMLButtonElement, React.ComponentProps<typeof Button>>(
  ({ className, variant = "outline", size = "icon", ...props }, ref) => {
    const { orientation, scrollNext, canScrollNext } = useCarousel();

    return (
      <Button
        ref={ref}
        variant={variant}
        size={size}
        className={cn(
          "absolute h-8 w-8 rounded-full",
          orientation === "horizontal"
            ? "-right-12 top-1/2 -translate-y-1/2"
```

```
209 |             : "-bottom-12 left-1/2 -translate-x-1/2 rotate-90",
210 |           className,
211 |         )}
212 |         disabled={!canScrollNext}
213 |         onClick={scrollNext}
214 |         {...props}
215 |       >
216 |         <ArrowRight className="h-4 w-4" />
217 |         <span className="sr-only">Next slide</span>
218 |       </Button>
219 |     );
220 |   },
221 | );
222 | CarouselNext.displayName = "CarouselNext";
223 |
224 | export { type CarouselApi, Carousel, CarouselContent, CarouselItem, CarouselPrevious,
CarouselNext };
225 |
```

File: src/components/ui/chart.tsx

Lines: 304

```
   1 | import * as React from "react";
   2 | import * as RechartsPrimitive from "recharts";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | // Format: { THEME_NAME: CSS_SELECTOR }
   7 | const THEMES = { light: "", dark: ".dark" } as const;
   8 |
   9 | export type ChartConfig = {
  10 |   [k in string]: {
  11 |     label?: React.ReactNode;
  12 |     icon?: React.ComponentType;
  13 |   } & ({ color?: string; theme?: never } | { color?: never; theme: Record<keyof typeof THEMES,
string> });
  14 | };
  15 |
  16 | type ChartContextProps = {
  17 |   config: ChartConfig;
  18 | };
  19 |
  20 | const ChartContext = React.createContext<ChartContextProps | null>(null);
  21 |
  22 | function useChart() {
  23 |   const context = React.useContext(ChartContext);
  24 |
  25 |   if (!context) {
  26 |     throw new Error("useChart must be used within a <ChartContainer />");
  27 |   }
  28 |
  29 |   return context;
  30 | }
  31 |
  32 | const ChartContainer = React.forwardRef<
  33 |   HTMLDivElement,
  34 |   React.ComponentProps<"div"> & {
  35 |     config: ChartConfig;
  36 |     children: React.ComponentProps<typeof RechartsPrimitive.ResponsiveContainer>["children"];
  37 |   }
  38 | >(({ id, className, children, config, ...props }, ref) => {
  39 |   const uniqueId = React.useId();
  40 |   const chartId = `chart-${id || uniqueId.replace(/:/g, "")}`;
  41 |
  42 |   return (
  43 |     <ChartContext.Provider value={{ config }}>
  44 |       <div
  45 |         data-chart={chartId}
  46 |         ref={ref}
  47 |         className={cn(
  48 |           "flex aspect-video justify-center text-xs [&_.recharts-cartesian-axis-tick_text]:fill-
muted-foreground [&_.rec
  49 |           className,
  50 |         )}
  51 |         {...props}
  52 |       >
  53 |         <ChartStyle id={chartId} config={config} />
  54 |         <RechartsPrimitive.ResponsiveContainer>{children}</RechartsPrimitive.ResponsiveContainer>
  55 |       </div>
  56 |     </ChartContext.Provider>
  57 |   );
  58 | });
  59 | ChartContainer.displayName = "Chart";
  60 |
  61 | const ChartStyle = ({ id, config }: { id: string; config: ChartConfig }) => {
  62 |   const colorConfig = Object.entries(config).filter(([_, config]) => config.theme ||
config.color);
  63 |
  64 |   if (!colorConfig.length) {
  65 |     return null;
  66 |   }
```

```
 67 |
 68 |   return (
 69 |     <style
 70 |       dangerouslySetInnerHTML={{
 71 |         __html: Object.entries(THEMES)
 72 |           .map(
 73 |             ([theme, prefix]) => `
 74 | ${prefix} [data-chart=${id}] {
 75 | ${colorConfig
 76 |   .map(([key, itemConfig]) => {
 77 |     const color = itemConfig.theme?.[theme as keyof typeof itemConfig.theme] || itemConfig.color;
 78 |     return color ? `  --color-${key}: ${color};` : null;
 79 |   })
 80 |   .join("\n")}
 81 | }
 82 | `,
 83 |           )
 84 |           .join("\n"),
 85 |       }}
 86 |     />
 87 |   );
 88 | };
 89 |
 90 | const ChartTooltip = RechartsPrimitive.Tooltip;
 91 |
 92 | const ChartTooltipContent = React.forwardRef<
 93 |   HTMLDivElement,
 94 |   React.ComponentProps<typeof RechartsPrimitive.Tooltip> &
 95 |     React.ComponentProps<"div"> & {
 96 |       hideLabel?: boolean;
 97 |       hideIndicator?: boolean;
 98 |       indicator?: "line" | "dot" | "dashed";
 99 |       nameKey?: string;
100 |       labelKey?: string;
101 |     }
102 | >(
103 |   (
104 |     {
105 |       active,
106 |       payload,
107 |       className,
108 |       indicator = "dot",
109 |       hideLabel = false,
110 |       hideIndicator = false,
111 |       label,
112 |       labelFormatter,
113 |       labelClassName,
114 |       formatter,
115 |       color,
116 |       nameKey,
117 |       labelKey,
118 |     },
119 |     ref,
120 |   ) => {
121 |     const { config } = useChart();
122 |
123 |     const tooltipLabel = React.useMemo(() => {
124 |       if (hideLabel || !payload?.length) {
125 |         return null;
126 |       }
127 |
128 |       const [item] = payload;
129 |       const key = `${labelKey || item.dataKey || item.name || "value"}`;
130 |       const itemConfig = getPayloadConfigFromPayload(config, item, key);
131 |       const value =
132 |         !labelKey && typeof label === "string"
133 |           ? config[label as keyof typeof config]?.label || label
134 |           : itemConfig?.label;
135 |
136 |       if (labelFormatter) {
137 |         return <div className={cn("font-medium", labelClassName)}>{labelFormatter(value,
payload)}</div>;
```

```
      }

      if (!value) {
        return null;
      }

      return <div className={cn("font-medium", labelClassName)}>{value}</div>;
    }, [label, labelFormatter, payload, hideLabel, labelClassName, config, labelKey]);

    if (!active || !payload?.length) {
      return null;
    }

    const nestLabel = payload.length === 1 && indicator !== "dot";

    return (
      <div
        ref={ref}
        className={cn(
          "grid min-w-[8rem] items-start gap-1.5 rounded-lg border border-border/50 bg-
background px-2.5 py-1.5 text-xs ...
          className,
        )}
      >
        {!nestLabel ? tooltipLabel : null}
        <div className="grid gap-1.5">
          {payload.map((item, index) => {
            const key = `${nameKey || item.name || item.dataKey || "value"}`;
            const itemConfig = getPayloadConfigFromPayload(config, item, key);
            const indicatorColor = color || item.payload.fill || item.color;

            return (
              <div
                key={item.dataKey}
                className={cn(
                  "flex w-full flex-wrap items-stretch gap-2 [&>svg]:h-2.5 [&>svg]:w-2.5
[&>svg]:text-muted-foreground",
                  indicator === "dot" && "items-center",
                )}
              >
                {formatter && item?.value !== undefined && item.name ? (
                  formatter(item.value, item.name, item, index, item.payload)
                ) : (
                  <>
                    {itemConfig?.icon ? (
                      <itemConfig.icon />
                    ) : (
                      !hideIndicator && (
                        <div
                          className={cn("shrink-0 rounded-[2px] border-[--color-border] bg-[--
color-bg]", {
                            "h-2.5 w-2.5": indicator === "dot",
                            "w-1": indicator === "line",
                            "w-0 border-[1.5px] border-dashed bg-transparent": indicator ===
"dashed",
                            "my-0.5": nestLabel && indicator === "dashed",
                          })}
                          style={
                            {
                              "--color-bg": indicatorColor,
                              "--color-border": indicatorColor,
                            } as React.CSSProperties
                          }
                        />
                      )
                    )}
                    <div
                      className={cn(
                        "flex flex-1 justify-between leading-none",
                        nestLabel ? "items-end" : "items-center",
                      )}
                    >
                      <div className="grid gap-1.5">
                        {nestLabel ? tooltipLabel : null}
                        <span className="text-muted-foreground">{itemConfig?.label || item.name}
</span>
```

```
                      </div>
                      {item.value && (
                        <span className="font-mono font-medium tabular-nums text-foreground">
                          {item.value.toLocaleString()}
                        </span>
                      )}
                    </div>
                  </>
                )}
              </div>
            );
          })}
        </div>
      </div>
    );
  },
);
ChartTooltipContent.displayName = "ChartTooltip";

const ChartLegend = RechartsPrimitive.Legend;

const ChartLegendContent = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> &
    Pick<RechartsPrimitive.LegendProps, "payload" | "verticalAlign"> & {
      hideIcon?: boolean;
      nameKey?: string;
    }
>(({ className, hideIcon = false, payload, verticalAlign = "bottom", nameKey }, ref) => {
  const { config } = useChart();

  if (!payload?.length) {
    return null;
  }

  return (
    <div
      ref={ref}
      className={cn("flex items-center justify-center gap-4", verticalAlign === "top" ? "pb-3" : "pt-3", className)}
    >
      {payload.map((item) => {
        const key = `${nameKey || item.dataKey || "value"}`;
        const itemConfig = getPayloadConfigFromPayload(config, item, key);

        return (
          <div
            key={item.value}
            className={cn("flex items-center gap-1.5 [&>svg]:h-3 [&>svg]:w-3 [&>svg]:text-muted-foreground")}
          >
            {itemConfig?.icon && !hideIcon ? (
              <itemConfig.icon />
            ) : (
              <div
                className="h-2 w-2 shrink-0 rounded-[2px]"
                style={{
                  backgroundColor: item.color,
                }}
              />
            )}
            {itemConfig?.label}
          </div>
        );
      })}
    </div>
  );
});
ChartLegendContent.displayName = "ChartLegend";

// Helper to extract item config from a payload.
function getPayloadConfigFromPayload(config: ChartConfig, payload: unknown, key: string) {
  if (typeof payload !== "object" || payload === null) {
```

```
280 |     return undefined;
281 |   }
282 |
283 |   const payloadPayload =
284 |     "payload" in payload && typeof payload.payload === "object" && payload.payload !== null
285 |       ? payload.payload
286 |       : undefined;
287 |
288 |   let configLabelKey: string = key;
289 |
290 |   if (key in payload && typeof payload[key as keyof typeof payload] === "string") {
291 |     configLabelKey = payload[key as keyof typeof payload] as string;
292 |   } else if (
293 |     payloadPayload &&
294 |     key in payloadPayload &&
295 |     typeof payloadPayload[key as keyof typeof payloadPayload] === "string"
296 |   ) {
297 |     configLabelKey = payloadPayload[key as keyof typeof payloadPayload] as string;
298 |   }
299 |
300 |   return configLabelKey in config ? config[configLabelKey] : config[key as keyof typeof config];
301 | }
302 |
303 | export { ChartContainer, ChartTooltip, ChartTooltipContent, ChartLegend, ChartLegendContent,
ChartStyle };
304 |
```

File: src/components/ui/checkbox.tsx

Lines: 27

```
  1 | import * as React from "react";
  2 | import * as CheckboxPrimitive from "@radix-ui/react-checkbox";
  3 | import { Check } from "lucide-react";
  4 |
  5 | import { cn } from "@/lib/utils";
  6 |
  7 | const Checkbox = React.forwardRef<
  8 |   React.ElementRef<typeof CheckboxPrimitive.Root>,
  9 |   React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>
 10 | >(({ className, ...props }, ref) => (
 11 |   <CheckboxPrimitive.Root
 12 |     ref={ref}
 13 |     className={cn(
 14 |       "peer h-4 w-4 shrink-0 rounded-sm border border-primary ring-offset-background data-
[state=checked]:bg-primary dat...
 15 |       className,
 16 |     )}
 17 |     {...props}
 18 |   >
 19 |     <CheckboxPrimitive.Indicator className={cn("flex items-center justify-center text-current")}>
 20 |       <Check className="h-4 w-4" />
 21 |     </CheckboxPrimitive.Indicator>
 22 |   </CheckboxPrimitive.Root>
 23 | ));
 24 | Checkbox.displayName = CheckboxPrimitive.Root.displayName;
 25 |
 26 | export { Checkbox };
 27 |
```

File: src/components/ui/collapsible.tsx

Lines: 10

```
 1 | import * as CollapsiblePrimitive from "@radix-ui/react-collapsible";
 2 |
 3 | const Collapsible = CollapsiblePrimitive.Root;
 4 |
 5 | const CollapsibleTrigger = CollapsiblePrimitive.CollapsibleTrigger;
 6 |
 7 | const CollapsibleContent = CollapsiblePrimitive.CollapsibleContent;
 8 |
 9 | export { Collapsible, CollapsibleTrigger, CollapsibleContent };
10 |
```

File: src/components/ui/command.tsx

Lines: 133

```
import * as React from "react";
import { type DialogProps } from "@radix-ui/react-dialog";
import { Command as CommandPrimitive } from "cmdk";
import { Search } from "lucide-react";

import { cn } from "@/lib/utils";
import { Dialog, DialogContent } from "@/components/ui/dialog";

const Command = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive>,
  React.ComponentPropsWithoutRef<typeof CommandPrimitive>
>(({ className, ...props }, ref) => (
  <CommandPrimitive
    ref={ref}
    className={cn(
      "flex h-full w-full flex-col overflow-hidden rounded-md bg-popover text-popover-
foreground", className,
    )}
    {...props}
  />
));
Command.displayName = CommandPrimitive.displayName;

interface CommandDialogProps extends DialogProps {}

const CommandDialog = ({ children, ...props }: CommandDialogProps) => {
  return (
    <Dialog {...props}>
      <DialogContent className="overflow-hidden p-0 shadow-lg">
        <Command className="[&_[cmdk-group-heading]]:px-2 [&_[cmdk-group-heading]]:font-medium
[&_[cmdk-group-heading]]:...
          {children}
        </Command>
      </DialogContent>
    </Dialog>
  );
};

const CommandInput = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.Input>,
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.Input>
>(({ className, ...props }, ref) => (
  <div className="flex items-center border-b px-3" cmdk-input-wrapper="">
    <Search className="mr-2 h-4 w-4 shrink-0 opacity-50" />
    <CommandPrimitive.Input
      ref={ref}
      className={cn(
        "flex h-11 w-full rounded-md bg-transparent py-3 text-sm outline-none placeholder:text-
muted-foreground disabled...
        className,
      )}
      {...props}
    />
  </div>
));

CommandInput.displayName = CommandPrimitive.Input.displayName;

const CommandList = React.forwardRef<
  React.ElementRef<typeof CommandPrimitive.List>,
  React.ComponentPropsWithoutRef<typeof CommandPrimitive.List>
>(({ className, ...props }, ref) => (
  <CommandPrimitive.List
    ref={ref}
    className={cn("max-h-[300px] overflow-y-auto overflow-x-hidden", className)}
    {...props}
  />
));
```

```
  67 |
  68 | CommandList.displayName = CommandPrimitive.List.displayName;
  69 |
  70 | const CommandEmpty = React.forwardRef<
  71 |   React.ElementRef<typeof CommandPrimitive.Empty>,
  72 |   React.ComponentPropsWithoutRef<typeof CommandPrimitive.Empty>
  73 | >((props, ref) => <CommandPrimitive.Empty ref={ref} className="py-6 text-center text-
sm" {...props} />);
  74 |
  75 | CommandEmpty.displayName = CommandPrimitive.Empty.displayName;
  76 |
  77 | const CommandGroup = React.forwardRef<
  78 |   React.ElementRef<typeof CommandPrimitive.Group>,
  79 |   React.ComponentPropsWithoutRef<typeof CommandPrimitive.Group>
  80 | >(({ className, ...props }, ref) => (
  81 |   <CommandPrimitive.Group
  82 |     ref={ref}
  83 |     className={cn(
  84 |       "overflow-hidden p-1 text-foreground [&_[cmdk-group-heading]]:px-2 [&_[cmdk-group-
heading]]:py-1.5 [&_[cmdk-group-...
  85 |       className,
  86 |     )}
  87 |     {...props}
  88 |   />
  89 | ));
  90 |
  91 | CommandGroup.displayName = CommandPrimitive.Group.displayName;
  92 |
  93 | const CommandSeparator = React.forwardRef<
  94 |   React.ElementRef<typeof CommandPrimitive.Separator>,
  95 |   React.ComponentPropsWithoutRef<typeof CommandPrimitive.Separator>
  96 | >(({ className, ...props }, ref) => (
  97 |   <CommandPrimitive.Separator ref={ref} className={cn("-mx-1 h-px bg-border", className)}
{...props} />
  98 | ));
  99 | CommandSeparator.displayName = CommandPrimitive.Separator.displayName;
 100 |
 101 | const CommandItem = React.forwardRef<
 102 |   React.ElementRef<typeof CommandPrimitive.Item>,
 103 |   React.ComponentPropsWithoutRef<typeof CommandPrimitive.Item>
 104 | >(({ className, ...props }, ref) => (
 105 |   <CommandPrimitive.Item
 106 |     ref={ref}
 107 |     className={cn(
 108 |       "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm
outline-none data-[disabled=...
 109 |       className,
 110 |     )}
 111 |     {...props}
 112 |   />
 113 | ));
 114 |
 115 | CommandItem.displayName = CommandPrimitive.Item.displayName;
 116 |
 117 | const CommandShortcut = ({ className, ...props }: React.HTMLAttributes<HTMLSpanElement>) => {
 118 |   return <span className={cn("ml-auto text-xs tracking-widest text-muted-foreground",
className)} {...props} />;
 119 | };
 120 | CommandShortcut.displayName = "CommandShortcut";
 121 |
 122 | export {
 123 |   Command,
 124 |   CommandDialog,
 125 |   CommandInput,
 126 |   CommandList,
 127 |   CommandEmpty,
 128 |   CommandGroup,
 129 |   CommandItem,
 130 |   CommandShortcut,
 131 |   CommandSeparator,
 132 | };
 133 |
```

File: src/components/ui/context-menu.tsx

Lines: 179

```
import * as React from "react";
import * as ContextMenuPrimitive from "@radix-ui/react-context-menu";
import { Check, ChevronRight, Circle } from "lucide-react";

import { cn } from "@/lib/utils";

const ContextMenu = ContextMenuPrimitive.Root;

const ContextMenuTrigger = ContextMenuPrimitive.Trigger;

const ContextMenuGroup = ContextMenuPrimitive.Group;

const ContextMenuPortal = ContextMenuPrimitive.Portal;

const ContextMenuSub = ContextMenuPrimitive.Sub;

const ContextMenuRadioGroup = ContextMenuPrimitive.RadioGroup;

const ContextMenuSubTrigger = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.SubTrigger>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubTrigger> & {
    inset?: boolean;
  }
>(({ className, inset, children, ...props }, ref) => (
  <ContextMenuPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none
data-[state=open]:bg-accent
      inset && "pl-8",
      className,
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto h-4 w-4" />
  </ContextMenuPrimitive.SubTrigger>
));
ContextMenuSubTrigger.displayName = ContextMenuPrimitive.SubTrigger.displayName;

const ContextMenuSubContent = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.SubContent>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.SubContent>
>(({ className, ...props }, ref) => (
  <ContextMenuPrimitive.SubContent
    ref={ref}
    className={cn(
      "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground shadow-md data-[state=...
      className,
    )}
    {...props}
  />
));
ContextMenuSubContent.displayName = ContextMenuPrimitive.SubContent.displayName;

const ContextMenuContent = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Content>
>(({ className, ...props }, ref) => (
  <ContextMenuPrimitive.Portal>
    <ContextMenuPrimitive.Content
      ref={ref}
      className={cn(
        "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground shadow-md animate-in...
      )}
      {...props}
```

```
  67 |     />
  68 |   </ContextMenuPrimitive.Portal>
  69 | ));
  70 | ContextMenuContent.displayName = ContextMenuPrimitive.Content.displayName;
  71 |
  72 | const ContextMenuItem = React.forwardRef<
  73 |   React.ElementRef<typeof ContextMenuPrimitive.Item>,
  74 |   React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Item> & {
  75 |     inset?: boolean;
  76 |   }
  77 | >(({ className, inset, ...props }, ref) => (
  78 |   <ContextMenuPrimitive.Item
  79 |     ref={ref}
  80 |     className={cn(
  81 |       "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm
outline-none data-[disabled]...
  82 |       inset && "pl-8",...
  83 |       className,
  84 |     )}
  85 |     {...props}
  86 |   />
  87 | ));
  88 | ContextMenuItem.displayName = ContextMenuPrimitive.Item.displayName;
  89 |
  90 | const ContextMenuCheckboxItem = React.forwardRef<
  91 |   React.ElementRef<typeof ContextMenuPrimitive.CheckboxItem>,
  92 |   React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.CheckboxItem>
  93 | >(({ className, children, checked, ...props }, ref) => (
  94 |   <ContextMenuPrimitive.CheckboxItem
  95 |     ref={ref}
  96 |     className={cn(
  97 |       "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm
outline-none data-[disa...
  98 |       className,
  99 |     )}
 100 |     checked={checked}
 101 |     {...props}
 102 |   >
 103 |     <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
 104 |       <ContextMenuPrimitive.ItemIndicator>
 105 |         <Check className="h-4 w-4" />
 106 |       </ContextMenuPrimitive.ItemIndicator>
 107 |     </span>
 108 |     {children}
 109 |   </ContextMenuPrimitive.CheckboxItem>
 110 | ));
 111 | ContextMenuCheckboxItem.displayName = ContextMenuPrimitive.CheckboxItem.displayName;
 112 |
 113 | const ContextMenuRadioItem = React.forwardRef<
 114 |   React.ElementRef<typeof ContextMenuPrimitive.RadioItem>,
 115 |   React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.RadioItem>
 116 | >(({ className, children, ...props }, ref) => (
 117 |   <ContextMenuPrimitive.RadioItem
 118 |     ref={ref}
 119 |     className={cn(
 120 |       "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm
outline-none data-[disa...
 121 |       className,
 122 |     )}
 123 |     {...props}
 124 |   >
 125 |     <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
 126 |       <ContextMenuPrimitive.ItemIndicator>
 127 |         <Circle className="h-2 w-2 fill-current" />
 128 |       </ContextMenuPrimitive.ItemIndicator>
 129 |     </span>
 130 |     {children}
 131 |   </ContextMenuPrimitive.RadioItem>
 132 | ));
 133 | ContextMenuRadioItem.displayName = ContextMenuPrimitive.RadioItem.displayName;
 134 |
 135 | const ContextMenuLabel = React.forwardRef<
 136 |   React.ElementRef<typeof ContextMenuPrimitive.Label>,
 137 |   React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Label> & {
```

```
    inset?: boolean;
  }
>(({ className, inset, ...props }, ref) => (
  <ContextMenuPrimitive.Label
    ref={ref}
    className={cn("px-2 py-1.5 text-sm font-semibold text-foreground", inset && "pl-8", className)}{...props}
  />
));
ContextMenuLabel.displayName = ContextMenuPrimitive.Label.displayName;

const ContextMenuSeparator = React.forwardRef<
  React.ElementRef<typeof ContextMenuPrimitive.Separator>,
  React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Separator>
>(({ className, ...props }, ref) => (
  <ContextMenuPrimitive.Separator ref={ref} className={cn("-mx-1 my-1 h-px bg-border", className)} {...props} />
));
ContextMenuSeparator.displayName = ContextMenuPrimitive.Separator.displayName;

const ContextMenuShortcut = ({ className, ...props }: React.HTMLAttributes<HTMLSpanElement>) => {
  return <span className={cn("ml-auto text-xs tracking-widest text-muted-foreground", className)} {...props} />;
};
ContextMenuShortcut.displayName = "ContextMenuShortcut";

export {
  ContextMenu,
  ContextMenuTrigger,
  ContextMenuContent,
  ContextMenuItem,
  ContextMenuCheckboxItem,
  ContextMenuRadioItem,
  ContextMenuLabel,
  ContextMenuSeparator,
  ContextMenuShortcut,
  ContextMenuGroup,
  ContextMenuPortal,
  ContextMenuSub,
  ContextMenuSubContent,
  ContextMenuSubTrigger,
  ContextMenuRadioGroup,
};

```

File: src/components/ui/dialog.tsx

Lines: 96

```
import * as React from "react";
import * as DialogPrimitive from "@radix-ui/react-dialog";
import { X } from "lucide-react";

import { cn } from "@/lib/utils";

const Dialog = DialogPrimitive.Root;

const DialogTrigger = DialogPrimitive.Trigger;

const DialogPortal = DialogPrimitive.Portal;

const DialogClose = DialogPrimitive.Close;

const DialogOverlay = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Overlay>
>(({ className, ...props }, ref) => (
  <DialogPrimitive.Overlay
    ref={ref}
    className={cn(
      "fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-
out data-[state=closed]:f...
    )}
    {...props}
  />
));
DialogOverlay.displayName = DialogPrimitive.Overlay.displayName;

const DialogContent = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Content>
>(({ className, children, ...props }, ref) => (
  <DialogPortal>
    <DialogOverlay />
    <DialogPrimitive.Content
      ref={ref}
      className={cn(
        "fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-
[-50%] gap-4 border bg-back...
      )}
      {...props}
    >
      {children}
      <DialogPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-
background transition-o...
        <X className="h-4 w-4" />
        <span className="sr-only">Close</span>
      </DialogPrimitive.Close>
    </DialogPrimitive.Content>
  </DialogPortal>
));
DialogContent.displayName = DialogPrimitive.Content.displayName;

const DialogHeader = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("flex flex-col space-y-1.5 text-center sm:text-left", className)}
{...props} />
);
DialogHeader.displayName = "DialogHeader";

const DialogFooter = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",
className)} {...props} />
);
DialogFooter.displayName = "DialogFooter";

const DialogTitle = React.forwardRef<
  React.ElementRef<typeof DialogPrimitive.Title>,
  React.ComponentPropsWithoutRef<typeof DialogPrimitive.Title>
```

```
  67 | >(({ className, ...props }, ref) => (
  68 |   <DialogPrimitive.Title
  69 |     ref={ref}
  70 |     className={cn("text-lg font-semibold leading-none tracking-tight", className)}
  71 |     {...props}
  72 |   />
  73 | ));
  74 | DialogTitle.displayName = DialogPrimitive.Title.displayName;
  75 |
  76 | const DialogDescription = React.forwardRef<
  77 |   React.ElementRef<typeof DialogPrimitive.Description>,
  78 |   React.ComponentPropsWithoutRef<typeof DialogPrimitive.Description>
  79 | >(({ className, ...props }, ref) => (
  80 |   <DialogPrimitive.Description ref={ref} className={cn("text-sm text-muted-foreground",
className)} {...props} />
  81 | ));
  82 | DialogDescription.displayName = DialogPrimitive.Description.displayName;
  83 |
  84 | export {
  85 |   Dialog,
  86 |   DialogPortal,
  87 |   DialogOverlay,
  88 |   DialogClose,
  89 |   DialogTrigger,
  90 |   DialogContent,
  91 |   DialogHeader,
  92 |   DialogFooter,
  93 |   DialogTitle,
  94 |   DialogDescription,
  95 | };
  96 |
```

File: src/components/ui/drawer.tsx

Lines: 88

```
import * as React from "react";
import { Drawer as DrawerPrimitive } from "vaul";

import { cn } from "@/lib/utils";

const Drawer = ({ shouldScaleBackground = true, ...props }: React.ComponentProps<typeof DrawerPrimitive.Root>) => (
  <DrawerPrimitive.Root shouldScaleBackground={shouldScaleBackground} {...props} />
);
Drawer.displayName = "Drawer";

const DrawerTrigger = DrawerPrimitive.Trigger;

const DrawerPortal = DrawerPrimitive.Portal;

const DrawerClose = DrawerPrimitive.Close;

const DrawerOverlay = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Overlay>
>(({ className, ...props }, ref) => (
  <DrawerPrimitive.Overlay ref={ref} className={cn("fixed inset-0 z-50 bg-black/80", className)} {...props} />
));
DrawerOverlay.displayName = DrawerPrimitive.Overlay.displayName;

const DrawerContent = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Content>
>(({ className, children, ...props }, ref) => (
  <DrawerPortal>
    <DrawerOverlay />
    <DrawerPrimitive.Content
      ref={ref}
      className={cn(
        "fixed inset-x-0 bottom-0 z-50 mt-24 flex h-auto flex-col rounded-t-[10px] border bg-background",
        className,
      )}
      {...props}
    >
      <div className="mx-auto mt-4 h-2 w-[100px] rounded-full bg-muted" />
      {children}
    </DrawerPrimitive.Content>
  </DrawerPortal>
));
DrawerContent.displayName = "DrawerContent";

const DrawerHeader = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("grid gap-1.5 p-4 text-center sm:text-left", className)} {...props} />
);
DrawerHeader.displayName = "DrawerHeader";

const DrawerFooter = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  <div className={cn("mt-auto flex flex-col gap-2 p-4", className)} {...props} />
);
DrawerFooter.displayName = "DrawerFooter";

const DrawerTitle = React.forwardRef<
  React.ElementRef<typeof DrawerPrimitive.Title>,
  React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Title>
>(({ className, ...props }, ref) => (
  <DrawerPrimitive.Title
    ref={ref}
    className={cn("text-lg font-semibold leading-none tracking-tight", className)}
    {...props}
  />
));
DrawerTitle.displayName = DrawerPrimitive.Title.displayName;
```

```
  67 |
  68 | const DrawerDescription = React.forwardRef<
  69 |   React.ElementRef<typeof DrawerPrimitive.Description>,
  70 |   React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Description>
  71 | >(({ className, ...props }, ref) => (
  72 |   <DrawerPrimitive.Description ref={ref} className={cn("text-sm text-muted-foreground",
className)} {...props} />
  73 | ));
  74 | DrawerDescription.displayName = DrawerPrimitive.Description.displayName;
  75 |
  76 | export {
  77 |   Drawer,
  78 |   DrawerPortal,
  79 |   DrawerOverlay,
  80 |   DrawerTrigger,
  81 |   DrawerClose,
  82 |   DrawerContent,
  83 |   DrawerHeader,
  84 |   DrawerFooter,
  85 |   DrawerTitle,
  86 |   DrawerDescription,
  87 | };
  88 |
```

File: src/components/ui/dropdown-menu.tsx

Lines: 180

```
import * as React from "react";
import * as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu";
import { Check, ChevronRight, Circle } from "lucide-react";

import { cn } from "@/lib/utils";

const DropdownMenu = DropdownMenuPrimitive.Root;

const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger;

const DropdownMenuGroup = DropdownMenuPrimitive.Group;

const DropdownMenuPortal = DropdownMenuPrimitive.Portal;

const DropdownMenuSub = DropdownMenuPrimitive.Sub;

const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup;

const DropdownMenuSubTrigger = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {
    inset?: boolean;
  }
>(({ className, inset, children, ...props }, ref) => (
  <DropdownMenuPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none
data-[state=open]:bg-accent
      inset && "pl-8",
      className,
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto h-4 w-4" />
  </DropdownMenuPrimitive.SubTrigger>
));
DropdownMenuSubTrigger.displayName = DropdownMenuPrimitive.SubTrigger.displayName;

const DropdownMenuSubContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>
>(({ className, ...props }, ref) => (
  <DropdownMenuPrimitive.SubContent
    ref={ref}
    className={cn(
      "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground shadow-lg data-[state=...
      className,
    )}
    {...props}
  />
));
DropdownMenuSubContent.displayName = DropdownMenuPrimitive.SubContent.displayName;

const DropdownMenuContent = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>
>(({ className, sideOffset = 4, ...props }, ref) => (
  <DropdownMenuPrimitive.Portal>
    <DropdownMenuPrimitive.Content
      ref={ref}
      sideOffset={sideOffset}
      className={cn(
        "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground shadow-md data-[stat...
      className,
      )}
```

```
      {...props}
    />
  </DropdownMenuPrimitive.Portal>
));
DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName;

const DropdownMenuItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {
    inset?: boolean;
  }
>(({ className, inset, ...props }, ref) => (
  <DropdownMenuPrimitive.Item
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none transition-colo...
      inset && "pl-8",
      className,
    )}
    {...props}
  />
));
DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName;

const DropdownMenuCheckboxItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>
>(({ className, children, checked, ...props }, ref) => (
  <DropdownMenuPrimitive.CheckboxItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition...
      className,
    )}
    checked={checked}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <DropdownMenuPrimitive.ItemIndicator>
        <Check className="h-4 w-4" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.CheckboxItem>
));
DropdownMenuCheckboxItem.displayName = DropdownMenuPrimitive.CheckboxItem.displayName;

const DropdownMenuRadioItem = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,
  React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>
>(({ className, children, ...props }, ref) => (
  <DropdownMenuPrimitive.RadioItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition...
      className,
    )}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <DropdownMenuPrimitive.ItemIndicator>
        <Circle className="h-2 w-2 fill-current" />
      </DropdownMenuPrimitive.ItemIndicator>
    </span>
    {children}
  </DropdownMenuPrimitive.RadioItem>
));
DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName;

const DropdownMenuLabel = React.forwardRef<
  React.ElementRef<typeof DropdownMenuPrimitive.Label>,
```

```
138 |   React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {
139 |     inset?: boolean;
140 |   }
141 | >(({ className, inset, ...props }, ref) => (
142 |   <DropdownMenuPrimitive.Label
143 |     ref={ref}
144 |     className={cn("px-2 py-1.5 text-sm font-semibold", inset && "pl-8", className)}
145 |     {...props}
146 |   />
147 | ));
148 | DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName;
149 |
150 | const DropdownMenuSeparator = React.forwardRef<
151 |   React.ElementRef<typeof DropdownMenuPrimitive.Separator>,
152 |   React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>
153 | >(({ className, ...props }, ref) => (
154 |   <DropdownMenuPrimitive.Separator ref={ref} className={cn("-mx-1 my-1 h-px bg-muted",
className)} {...props} />
155 | ));
156 | DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName;
157 |
158 | const DropdownMenuShortcut = ({ className, ...props }: React.HTMLAttributes<HTMLSpanElement>) =>
{
159 |   return <span className={cn("ml-auto text-xs tracking-widest opacity-60", className)}
{...props} />;
160 | };
161 | DropdownMenuShortcut.displayName = "DropdownMenuShortcut";
162 |
163 | export {
164 |   DropdownMenu,
165 |   DropdownMenuTrigger,
166 |   DropdownMenuContent,
167 |   DropdownMenuItem,
168 |   DropdownMenuCheckboxItem,
169 |   DropdownMenuRadioItem,
170 |   DropdownMenuLabel,
171 |   DropdownMenuSeparator,
172 |   DropdownMenuShortcut,
173 |   DropdownMenuGroup,
174 |   DropdownMenuPortal,
175 |   DropdownMenuSub,
176 |   DropdownMenuSubContent,
177 |   DropdownMenuSubTrigger,
178 |   DropdownMenuRadioGroup,
179 | };
180 |
```

File: src/components/ui/form.tsx

Lines: 130

```
import * as React from "react";
import * as LabelPrimitive from "@radix-ui/react-label";
import { Slot } from "@radix-ui/react-slot";
import { Controller, ControllerProps, FieldPath, FieldValues, FormProvider, useFormContext } from "react-hook-form";

import { cn } from "@/lib/utils";
import { Label } from "@/components/ui/label";

const Form = FormProvider;

type FormFieldContextValue<
  TFieldValues extends FieldValues = FieldValues,
  TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>,
> = {
  name: TName;
};

const FormFieldContext = React.createContext<FormFieldContextValue>({} as FormFieldContextValue);

const FormField = <
  TFieldValues extends FieldValues = FieldValues,
  TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>,
>({
  ...props
}: ControllerProps<TFieldValues, TName>) => {
  return (
    <FormFieldContext.Provider value={{ name: props.name }}>
      <Controller {...props} />
    </FormFieldContext.Provider>
  );
};

const useFormField = () => {
  const fieldContext = React.useContext(FormFieldContext);
  const itemContext = React.useContext(FormItemContext);
  const { getFieldState, formState } = useFormContext();

  const fieldState = getFieldState(fieldContext.name, formState);

  if (!fieldContext) {
    throw new Error("useFormField should be used within <FormField>");
  }

  const { id } = itemContext;

  return {
    id,
    name: fieldContext.name,
    formItemId: `${id}-form-item`,
    formDescriptionId: `${id}-form-item-description`,
    formMessageId: `${id}-form-item-message`,
    ...fieldState,
  };
};

type FormItemContextValue = {
  id: string;
};

const FormItemContext = React.createContext<FormItemContextValue>({} as FormItemContextValue);

const FormItem = React.forwardRef<HTMLDivElement, React.HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => {
    const id = React.useId();

    return (
```

```
      <FormItemContext.Provider value={{ id }}>
        <div ref={ref} className={cn("space-y-2", className)} {...props} />
      </FormItemContext.Provider>
    );
  },
);
FormItem.displayName = "FormItem";

const FormLabel = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root>
>(({ className, ...props }, ref) => {
  const { error, formItemId } = useFormField();

  return <Label ref={ref} className={cn(error && "text-destructive", className)}
htmlFor={formItemId} {...props} />;
});
FormLabel.displayName = "FormLabel";

const FormControl = React.forwardRef<React.ElementRef<typeof Slot>,
React.ComponentPropsWithoutRef<typeof Slot>>(
  ({ ...props }, ref) => {
    const { error, formItemId, formDescriptionId, formMessageId } = useFormField();

    return (
      <Slot
        ref={ref}
        id={formItemId}
        aria-describedby={!error ? `${formDescriptionId}` : `${formDescriptionId}
${formMessageId}`}
        aria-invalid={!!error}
        {...props}
      />
    );
  },
);
FormControl.displayName = "FormControl";

const FormDescription = React.forwardRef<HTMLParagraphElement,
React.HTMLAttributes<HTMLParagraphElement>>(
  ({ className, ...props }, ref) => {
    const { formDescriptionId } = useFormField();

    return <p ref={ref} id={formDescriptionId} className={cn("text-sm text-muted-foreground",
className)} {...props} />;
  },
);
FormDescription.displayName = "FormDescription";

const FormMessage = React.forwardRef<HTMLParagraphElement,
React.HTMLAttributes<HTMLParagraphElement>>(({ className, children, ...props }, ref) => {
  const { error, formMessageId } = useFormField();
  const body = error ? String(error?.message) : children;

  if (!body) {
    return null;
  }

  return (
    <p ref={ref} id={formMessageId} className={cn("text-sm font-medium text-destructive",
className)} {...props}>
      {body}
    </p>
  );
},
);
FormMessage.displayName = "FormMessage";

export { useFormField, Form, FormItem, FormLabel, FormControl, FormDescription, FormMessage,
FormField };
```

File: src/components/ui/hover-card.tsx

Lines: 28

```
  1 | import * as React from "react";
  2 | import * as HoverCardPrimitive from "@radix-ui/react-hover-card";
  3 |
  4 | import { cn } from "@/lib/utils";
  5 |
  6 | const HoverCard = HoverCardPrimitive.Root;
  7 |
  8 | const HoverCardTrigger = HoverCardPrimitive.Trigger;
  9 |
 10 | const HoverCardContent = React.forwardRef<
 11 |   React.ElementRef<typeof HoverCardPrimitive.Content>,
 12 |   React.ComponentPropsWithoutRef<typeof HoverCardPrimitive.Content>
 13 | >(({ className, align = "center", sideOffset = 4, ...props }, ref) => (
 14 |   <HoverCardPrimitive.Content
 15 |     ref={ref}
 16 |     align={align}
 17 |     sideOffset={sideOffset}
 18 |     className={cn(
 19 |       "z-50 w-64 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none
data-[state=open]:anima...
 20 |       className,
 21 |     )}
 22 |     {...props}
 23 |   />
 24 | ));
 25 | HoverCardContent.displayName = HoverCardPrimitive.Content.displayName;
 26 |
 27 | export { HoverCard, HoverCardTrigger, HoverCardContent };
 28 |
```

File: src/components/ui/input-otp.tsx

Lines: 62

```
import * as React from "react";
import { OTPInput, OTPInputContext } from "input-otp";
import { Dot } from "lucide-react";

import { cn } from "@/lib/utils";

const InputOTP = React.forwardRef<React.ElementRef<typeof OTPInput>,
React.ComponentPropsWithoutRef<typeof OTPInput>>(
  ({ className, containerClassName, ...props }, ref) => (
    <OTPInput
      ref={ref}
      containerClassName={cn("flex items-center gap-2 has-[:disabled]:opacity-50",
containerClassName)}
      className={cn("disabled:cursor-not-allowed", className)}
      {...props}
    />
  ),
);
InputOTP.displayName = "InputOTP";

const InputOTPGroup = React.forwardRef<React.ElementRef<"div">,
React.ComponentPropsWithoutRef<"div">>(({ className, ...props }, ref) => <div ref={ref} className={cn("flex items-center",
className)} {...props} />,
);
InputOTPGroup.displayName = "InputOTPGroup";

const InputOTPSlot = React.forwardRef<
  React.ElementRef<"div">,
  React.ComponentPropsWithoutRef<"div"> & { index: number }
>(({ index, className, ...props }, ref) => {
  const inputOTPContext = React.useContext(OTPInputContext);
  const { char, hasFakeCaret, isActive } = inputOTPContext.slots[index];

  return (
    <div
      ref={ref}
      className={cn(
        "relative flex h-10 w-10 items-center justify-center border-y border-r border-input text-
smtransition-all first:...
        isActive && "z-10 ring-2 ring-ring ring-offset-background",
        className,
      )}
      {...props}
    >
      {char}
      {hasFakeCaret && (
        <div className="pointer-events-none absolute inset-0 flex items-center justify-center">
          <div className="animate-caret-blink h-4 w-px bg-foreground duration-1000" />
        </div>
      )}
    </div>
  );
});
InputOTPSlot.displayName = "InputOTPSlot";

const InputOTPSeparator = React.forwardRef<React.ElementRef<"div">,
React.ComponentPropsWithoutRef<"div">>(({ ...props }, ref) => (
  <div ref={ref} role="separator" {...props}>
    <Dot />
  </div>
  ),
);
InputOTPSeparator.displayName = "InputOTPSeparator";

export { InputOTP, InputOTPGroup, InputOTPSlot, InputOTPSeparator };

```

File: src/components/ui/input.tsx

Lines: 23

```
   1 | import * as React from "react";
   2 |
   3 | import { cn } from "@/lib/utils";
   4 |
   5 | const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>(
   6 |   ({ className, type, ...props }, ref) => {
   7 |     return (
   8 |       <input
   9 |         type={type}
  10 |         className={cn(
  11 |           "flex h-10 w-full rounded-md border border-input bg-background px-3 py-2 text-base
ring-offset-background file...
  12 |           className,
  13 |         )}
  14 |         ref={ref}
  15 |         {...props}
  16 |       />
  17 |     );
  18 |   },
  19 | );
  20 | Input.displayName = "Input";
  21 |
  22 | export { Input };
  23 |
```

File: src/components/ui/label.tsx

Lines: 18

```
   1 | import * as React from "react";
   2 | import * as LabelPrimitive from "@radix-ui/react-label";
   3 | import { cva, type VariantProps } from "class-variance-authority";
   4 |
   5 | import { cn } from "@/lib/utils";
   6 |
   7 | const labelVariants = cva("text-sm font-medium leading-none peer-disabled:cursor-not-allowed
peer-disabled:opacity-70");
   8 |
   9 | const Label = React.forwardRef<
  10 |   React.ElementRef<typeof LabelPrimitive.Root>,
  11 |   React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> & VariantProps<typeof labelVariants>
  12 | >(({ className, ...props }, ref) => (
  13 |   <LabelPrimitive.Root ref={ref} className={cn(labelVariants(), className)} {...props} />
  14 | ));
  15 | Label.displayName = LabelPrimitive.Root.displayName;
  16 |
  17 | export { Label };
  18 |
```

File: src/components/ui/menubar.tsx

Lines: 208

```
import * as React from "react";
import * as MenubarPrimitive from "@radix-ui/react-menubar";
import { Check, ChevronRight, Circle } from "lucide-react";

import { cn } from "@/lib/utils";

const MenubarMenu = MenubarPrimitive.Menu;

const MenubarGroup = MenubarPrimitive.Group;

const MenubarPortal = MenubarPrimitive.Portal;

const MenubarSub = MenubarPrimitive.Sub;

const MenubarRadioGroup = MenubarPrimitive.RadioGroup;

const Menubar = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Root>
>(({ className, ...props }, ref) => (
  <MenubarPrimitive.Root
    ref={ref}
    className={cn("flex h-10 items-center space-x-1 rounded-md border bg-background p-1",
className)}{...props}
  />
));
Menubar.displayName = MenubarPrimitive.Root.displayName;

const MenubarTrigger = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Trigger>
>(({ className, ...props }, ref) => (
  <MenubarPrimitive.Trigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-3 py-1.5 text-sm font-medium
outline-none data-[state=...
      className,
    )}
    {...props}
  />
));
MenubarTrigger.displayName = MenubarPrimitive.Trigger.displayName;

const MenubarSubTrigger = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.SubTrigger>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubTrigger> & {
    inset?: boolean;
  }
>(({ className, inset, children, ...props }, ref) => (
  <MenubarPrimitive.SubTrigger
    ref={ref}
    className={cn(
      "flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none
data-[state=open]:bg-acc...
      inset && "pl-8",
      className,
    )}
    {...props}
  >
    {children}
    <ChevronRight className="ml-auto h-4 w-4" />
  </MenubarPrimitive.SubTrigger>
));
MenubarSubTrigger.displayName = MenubarPrimitive.SubTrigger.displayName;

const MenubarSubContent = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.SubContent>,
```

```
  67 |   React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubContent>
  68 | >(({ className, ...props }, ref) => (
  69 |   <MenubarPrimitive.SubContent
  70 |     ref={ref}
  71 |     className={cn(
  72 |       "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground data-[state=open]:anim...
  73 |       className,
  74 |     )}
  75 |     {...props}
  76 |   />
  77 | ));
  78 | MenubarSubContent.displayName = MenubarPrimitive.SubContent.displayName;
  79 |
  80 | const MenubarContent = React.forwardRef<
  81 |   React.ElementRef<typeof MenubarPrimitive.Content>,
  82 |   React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Content>
  83 | >(({ className, align = "start", alignOffset = -4, sideOffset = 8, ...props }, ref) => (
  84 |   <MenubarPrimitive.Portal>
  85 |     <MenubarPrimitive.Content
  86 |       ref={ref}
  87 |       align={align}
  88 |       alignOffset={alignOffset}
  89 |       sideOffset={sideOffset}
  90 |       className={cn(
  91 |         "z-50 min-w-[12rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-
foreground shadow-md data-[sta...
  92 |         className,
  93 |       )}
  94 |       {...props}
  95 |     />
  96 |   </MenubarPrimitive.Portal>
  97 | ));
  98 | MenubarContent.displayName = MenubarPrimitive.Content.displayName;
  99 |
 100 | const MenubarItem = React.forwardRef<
 101 |   React.ElementRef<typeof MenubarPrimitive.Item>,
 102 |   React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Item> & {
 103 |     inset?: boolean;
 104 |   }
 105 | >(({ className, inset, ...props }, ref) => (
 106 |   <MenubarPrimitive.Item
 107 |     ref={ref}
 108 |     className={cn(
 109 |       "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm
outline-none data-[disabled]:...
 110 |       inset && "pl-8",
 111 |       className,
 112 |     )}
 113 |     {...props}
 114 |   />
 115 | ));
 116 | MenubarItem.displayName = MenubarPrimitive.Item.displayName;
 117 |
 118 | const MenubarCheckboxItem = React.forwardRef<
 119 |   React.ElementRef<typeof MenubarPrimitive.CheckboxItem>,
 120 |   React.ComponentPropsWithoutRef<typeof MenubarPrimitive.CheckboxItem>
 121 | >(({ className, children, checked, ...props }, ref) => (
 122 |   <MenubarPrimitive.CheckboxItem
 123 |     ref={ref}
 124 |     className={cn(
 125 |       "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm
outline-none data-[disa...
 126 |       className,
 127 |     )}
 128 |     checked={checked}
 129 |     {...props}
 130 |   >
 131 |     <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
 132 |       <MenubarPrimitive.ItemIndicator>
 133 |         <Check className="h-4 w-4" />
 134 |       </MenubarPrimitive.ItemIndicator>
 135 |     </span>
 136 |     {children}
 137 |   </MenubarPrimitive.CheckboxItem>
```

```
));
MenubarCheckboxItem.displayName = MenubarPrimitive.CheckboxItem.displayName;

const MenubarRadioItem = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.RadioItem>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.RadioItem>
>(({ className, children, ...props }, ref) => (
  <MenubarPrimitive.RadioItem
    ref={ref}
    className={cn(
      "relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none data-[disa...
    )}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <MenubarPrimitive.ItemIndicator>
        <Circle className="h-2 w-2 fill-current" />
      </MenubarPrimitive.ItemIndicator>
    </span>
    {children}
  </MenubarPrimitive.RadioItem>
));
MenubarRadioItem.displayName = MenubarPrimitive.RadioItem.displayName;

const MenubarLabel = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.Label>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Label> & {
    inset?: boolean;
  }
>(({ className, inset, ...props }, ref) => (
  <MenubarPrimitive.Label
    ref={ref}
    className={cn("px-2 py-1.5 text-sm font-semibold", inset && "pl-8", className)}
    {...props}
  />
));
MenubarLabel.displayName = MenubarPrimitive.Label.displayName;

const MenubarSeparator = React.forwardRef<
  React.ElementRef<typeof MenubarPrimitive.Separator>,
  React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Separator>
>(({ className, ...props }, ref) => (
  <MenubarPrimitive.Separator ref={ref} className={cn("-mx-1 my-1 h-px bg-muted", className)} {...props} />
));
MenubarSeparator.displayName = MenubarPrimitive.Separator.displayName;

const MenubarShortcut = ({ className, ...props }: React.HTMLAttributes<HTMLSpanElement>) => {
  return <span className={cn("ml-auto text-xs tracking-widest text-muted-foreground", className)} {...props} />;
};
MenubarShortcut.displayname = "MenubarShortcut";

export {
  Menubar,
  MenubarMenu,
  MenubarTrigger,
  MenubarContent,
  MenubarItem,
  MenubarSeparator,
  MenubarLabel,
  MenubarCheckboxItem,
  MenubarRadioGroup,
  MenubarRadioItem,
  MenubarPortal,
  MenubarSubContent,
  MenubarSubTrigger,
  MenubarGroup,
  MenubarSub,
  MenubarShortcut,
};
```

File: src/components/ui/navigation-menu.tsx

Lines: 121

```
import * as React from "react";
import * as NavigationMenuPrimitive from "@radix-ui/react-navigation-menu";
import { cva } from "class-variance-authority";
import { ChevronDown } from "lucide-react";

import { cn } from "@/lib/utils";

const NavigationMenu = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Root>
>(({ className, children, ...props }, ref) => (
  <NavigationMenuPrimitive.Root
    ref={ref}
    className={cn("relative z-10 flex max-w-max flex-1 items-center justify-center", className)}
    {...props}
  >
    {children}
    <NavigationMenuViewport />
  </NavigationMenuPrimitive.Root>
));
NavigationMenu.displayName = NavigationMenuPrimitive.Root.displayName;

const NavigationMenuList = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.List>,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.List>
>(({ className, ...props }, ref) => (
  <NavigationMenuPrimitive.List
    ref={ref}
    className={cn("group flex flex-1 list-none items-center justify-center space-x-1",
className)}{...props}
  />
));
NavigationMenuList.displayName = NavigationMenuPrimitive.List.displayName;

const NavigationMenuItem = NavigationMenuPrimitive.Item;

const navigationMenuTriggerStyle = cva(
  "group inline-flex h-10 w-max items-center justify-center rounded-md bg-background px-4 py-2
text-sm font-medium trans...
);

const NavigationMenuTrigger = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Trigger>
>(({ className, children, ...props }, ref) => (
  <NavigationMenuPrimitive.Trigger
    ref={ref}
    className={cn(navigationMenuTriggerStyle(), "group", className)}
    {...props}
  >
    {children}{" "}
    <ChevronDown
      className="relative top-[1px] ml-1 h-3 w-3 transition duration-200 group-data-
[state=open]:rotate-180"
      aria-hidden="true"
    />
  </NavigationMenuPrimitive.Trigger>
));
NavigationMenuTrigger.displayName = NavigationMenuPrimitive.Trigger.displayName;

const NavigationMenuContent = React.forwardRef<
  React.ElementRef<typeof NavigationMenuPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Content>
>(({ className, ...props }, ref) => (
  <NavigationMenuPrimitive.Content
    ref={ref}
    className={cn(
      "left-0 top-0 w-full data-[motion^=from-]:animate-in data-[motion^=to-]:animate-out data-
[motion^=from-]:fade-in d...
```

```
  67 |       className,
  68 |     )}
  69 |     {...props}
  70 |   />
  71 | ));
  72 | NavigationMenuContent.displayName = NavigationMenuPrimitive.Content.displayName;
  73 |
  74 | const NavigationMenuLink = NavigationMenuPrimitive.Link;
  75 |
  76 | const NavigationMenuViewport = React.forwardRef<
  77 |   React.ElementRef<typeof NavigationMenuPrimitive.Viewport>,
  78 |   React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Viewport>
  79 | >(({ className, ...props }, ref) => (
  80 |   <div className={cn("absolute left-0 top-full flex justify-center")}>
  81 |     <NavigationMenuPrimitive.Viewport
  82 |       className={cn(
  83 |         "origin-top-center relative mt-1.5 h-[var(--radix-navigation-menu-viewport-height)] w-
full overflow-hidden round...
  84 |         className,
  85 |       )}
  86 |       ref={ref}
  87 |       {...props}
  88 |     />
  89 |   </div>
  90 | ));
  91 | NavigationMenuViewport.displayName = NavigationMenuPrimitive.Viewport.displayName;
  92 |
  93 | const NavigationMenuIndicator = React.forwardRef<
  94 |   React.ElementRef<typeof NavigationMenuPrimitive.Indicator>,
  95 |   React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Indicator>
  96 | >(({ className, ...props }, ref) => (
  97 |   <NavigationMenuPrimitive.Indicator
  98 |     ref={ref}
  99 |     className={cn(
 100 |       "top-full z-[1] flex h-1.5 items-end justify-center overflow-hidden data-
[state=visible]:animate-in data-[state=hi...
 101 |       className,
 102 |     )}
 103 |     {...props}
 104 |   >
 105 |     <div className="relative top-[60%] h-2 w-2 rotate-45 rounded-tl-sm bg-border shadow-md" />
 106 |   </NavigationMenuPrimitive.Indicator>
 107 | ));
 108 | NavigationMenuIndicator.displayName = NavigationMenuPrimitive.Indicator.displayName;
 109 |
 110 | export {
 111 |   navigationMenuTriggerStyle,
 112 |   NavigationMenu,
 113 |   NavigationMenuList,
 114 |   NavigationMenuItem,
 115 |   NavigationMenuContent,
 116 |   NavigationMenuTrigger,
 117 |   NavigationMenuLink,
 118 |   NavigationMenuIndicator,
 119 |   NavigationMenuViewport,
 120 | };
 121 |
```

File: src/components/ui/pagination.tsx

Lines: 82

```
import * as React from "react";
import { ChevronLeft, ChevronRight, MoreHorizontal } from "lucide-react";

import { cn } from "@/lib/utils";
import { ButtonProps, buttonVariants } from "@/components/ui/button";

const Pagination = ({ className, ...props }: React.ComponentProps<"nav">) => (
  <nav
    role="navigation"
    aria-label="pagination"
    className={cn("mx-auto flex w-full justify-center", className)}
    {...props}
  />
);
Pagination.displayName = "Pagination";

const PaginationContent = React.forwardRef<HTMLUListElement, React.ComponentProps<"ul">>(
  ({ className, ...props }, ref) => (
    <ul ref={ref} className={cn("flex flex-row items-center gap-1", className)} {...props} />
  ),
);
PaginationContent.displayName = "PaginationContent";

const PaginationItem = React.forwardRef<HTMLLIElement,
React.ComponentProps<"li">>(({ className, ...props }, ref) => (
  <li ref={ref} className={cn("", className)} {...props} />
));
PaginationItem.displayName = "PaginationItem";

type PaginationLinkProps = {
  isActive?: boolean;
} & Pick<ButtonProps, "size"> &
  React.ComponentProps<"a">;

const PaginationLink = ({ className, isActive, size = "icon", ...props }: PaginationLinkProps)
=> (
  <a
    aria-current={isActive ? "page" : undefined}
    className={cn(
      buttonVariants({
        variant: isActive ? "outline" : "ghost",
        size,
      }),
      className,
    )}
    {...props}
  />
);
PaginationLink.displayName = "PaginationLink";

const PaginationPrevious = ({ className, ...props }: React.ComponentProps<typeof
PaginationLink>) => (
  <PaginationLink aria-label="Go to previous page" size="default" className={cn("gap-1 pl-2.5",
className)} {...props}>
    <ChevronLeft className="h-4 w-4" />
    <span>Previous</span>
  </PaginationLink>
);
PaginationPrevious.displayName = "PaginationPrevious";

const PaginationNext = ({ className, ...props }: React.ComponentProps<typeof PaginationLink>) =>
( <PaginationLink aria-label="Go to next page" size="default" className={cn("gap-1 pr-2.5",
className)} {...props}>
    <span>Next</span>
    <ChevronRight className="h-4 w-4" />
  </PaginationLink>
);
PaginationNext.displayName = "PaginationNext";

const PaginationEllipsis = ({ className, ...props }: React.ComponentProps<"span">) => (
  <span aria-hidden className={cn("flex h-9 w-9 items-center justify-center", className)}
{...props}>
```

```
    <MoreHorizontal className="h-4 w-4" />
    <span className="sr-only">More pages</span>
  </span>
);
PaginationEllipsis.displayName = "PaginationEllipsis";

export {
  Pagination,
  PaginationContent,
  PaginationEllipsis,
  PaginationItem,
  PaginationLink,
  PaginationNext,
  PaginationPrevious,
};

```

File: src/components/ui/popover.tsx

Lines: 30

```
   1 | import * as React from "react";
   2 | import * as PopoverPrimitive from "@radix-ui/react-popover";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const Popover = PopoverPrimitive.Root;
   7 |
   8 | const PopoverTrigger = PopoverPrimitive.Trigger;
   9 |
  10 | const PopoverContent = React.forwardRef<
  11 |   React.ElementRef<typeof PopoverPrimitive.Content>,
  12 |   React.ComponentPropsWithoutRef<typeof PopoverPrimitive.Content>
  13 | >(({ className, align = "center", sideOffset = 4, ...props }, ref) => (
  14 |   <PopoverPrimitive.Portal>
  15 |     <PopoverPrimitive.Content
  16 |       ref={ref}
  17 |       align={align}
  18 |       sideOffset={sideOffset}
  19 |       className={cn(
  20 |         "z-50 w-72 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-
none data-[state=open]:ani...
  21 |         className,
  22 |       )}
  23 |       {...props}
  24 |     />
  25 |   </PopoverPrimitive.Portal>
  26 | ));
  27 | PopoverContent.displayName = PopoverPrimitive.Content.displayName;
  28 |
  29 | export { Popover, PopoverTrigger, PopoverContent };
  30 |
```

File: src/components/ui/progress.tsx

Lines: 24

```
 1 | import * as React from "react";
 2 | import * as ProgressPrimitive from "@radix-ui/react-progress";
 3 |
 4 | import { cn } from "@/lib/utils";
 5 |
 6 | const Progress = React.forwardRef<
 7 |   React.ElementRef<typeof ProgressPrimitive.Root>,
 8 |   React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>
 9 | >(({ className, value, ...props }, ref) => (
10 |   <ProgressPrimitive.Root
11 |     ref={ref}
12 |     className={cn("relative h-4 w-full overflow-hidden rounded-full bg-secondary", className)}
13 |     {...props}
14 |   >
15 |     <ProgressPrimitive.Indicator
16 |       className="h-full w-full flex-1 bg-primary transition-all"
17 |       style={{ transform: `translateX(-${100 - (value || 0)}%)` }}
18 |     />
19 |   </ProgressPrimitive.Root>
20 | ));
21 | Progress.displayName = ProgressPrimitive.Root.displayName;
22 |
23 | export { Progress };
24 |
```

File: src/components/ui/radio-group.tsx

Lines: 37

```
   1 | import * as React from "react";
   2 | import * as RadioGroupPrimitive from "@radix-ui/react-radio-group";
   3 | import { Circle } from "lucide-react";
   4 |
   5 | import { cn } from "@/lib/utils";
   6 |
   7 | const RadioGroup = React.forwardRef<
   8 |   React.ElementRef<typeof RadioGroupPrimitive.Root>,
   9 |   React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Root>
  10 | >(({ className, ...props }, ref) => {
  11 |   return <RadioGroupPrimitive.Root className={cn("grid gap-2", className)} {...props} ref={ref} /
>;12 | });
  13 | RadioGroup.displayName = RadioGroupPrimitive.Root.displayName;
  14 |
  15 | const RadioGroupItem = React.forwardRef<
  16 |   React.ElementRef<typeof RadioGroupPrimitive.Item>,
  17 |   React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Item>
  18 | >(({ className, ...props }, ref) => {
  19 |   return (
  20 |     <RadioGroupPrimitive.Item
  21 |       ref={ref}
  22 |       className={cn(
  23 |         "aspect-square h-4 w-4 rounded-full border border-primary text-primary ring-offset-
background focus:outline-none...
  24 |         className,
  25 |       )}
  26 |       {...props}
  27 |     >
  28 |       <RadioGroupPrimitive.Indicator className="flex items-center justify-center">
  29 |         <Circle className="h-2.5 w-2.5 fill-current text-current" />
  30 |       </RadioGroupPrimitive.Indicator>
  31 |     </RadioGroupPrimitive.Item>
  32 |   );
  33 | });
  34 | RadioGroupItem.displayName = RadioGroupPrimitive.Item.displayName;
  35 |
  36 | export { RadioGroup, RadioGroupItem };
  37 |
```

File: src/components/ui/resizable.tsx

Lines: 38

```
import { GripVertical } from "lucide-react";
import * as ResizablePrimitive from "react-resizable-panels";

import { cn } from "@/lib/utils";

const ResizablePanelGroup = ({ className, ...props }: React.ComponentProps<typeof ResizablePrimitive.PanelGroup>) => (
  <ResizablePrimitive.PanelGroup
    className={cn("flex h-full w-full data-[panel-group-direction=vertical]:flex-col", className)}
    {...props}
  />
);

const ResizablePanel = ResizablePrimitive.Panel;

const ResizableHandle = ({
  withHandle,
  className,
  ...props
}: React.ComponentProps<typeof ResizablePrimitive.PanelResizeHandle> & {
  withHandle?: boolean;
}) => (
  <ResizablePrimitive.PanelResizeHandle
    className={cn(
      "relative flex w-px items-center justify-center bg-border after:absolute after:inset-y-0 after:left-1/2 after:w-1 ...
      className,
    )}
    {...props}
  >
    {withHandle && (
      <div className="z-10 flex h-4 w-3 items-center justify-center rounded-sm border bg-border">
        <GripVertical className="h-2.5 w-2.5" />
      </div>
    )}
  </ResizablePrimitive.PanelResizeHandle>
);

export { ResizablePanelGroup, ResizablePanel, ResizableHandle };

```

File: src/components/ui/scroll-area.tsx

Lines: 39

```
   1 | import * as React from "react";
   2 | import * as ScrollAreaPrimitive from "@radix-ui/react-scroll-area";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const ScrollArea = React.forwardRef<
   7 |   React.ElementRef<typeof ScrollAreaPrimitive.Root>,
   8 |   React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root>
   9 | >(({ className, children, ...props }, ref) => (
  10 |   <ScrollAreaPrimitive.Root ref={ref} className={cn("relative overflow-hidden", className)}
{...props}><ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">{children}</
ScrollAreaPrimitive.Viewport>
  12 |     <ScrollBar />
  13 |     <ScrollAreaPrimitive.Corner />
  14 |   </ScrollAreaPrimitive.Root>
  15 | ));
  16 | ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName;
  17 |
  18 | const ScrollBar = React.forwardRef<
  19 |   React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,
  20 |   React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>
  21 | >(({ className, orientation = "vertical", ...props }, ref) => (
  22 |   <ScrollAreaPrimitive.ScrollAreaScrollbar
  23 |     ref={ref}
  24 |     orientation={orientation}
  25 |     className={cn(
  26 |       "flex touch-none select-none transition-colors",
  27 |       orientation === "vertical" && "h-full w-2.5 border-l border-l-transparent p-[1px]",
  28 |       orientation === "horizontal" && "h-2.5 flex-col border-t border-t-transparent p-[1px]",
  29 |       className,
  30 |     )}
  31 |     {...props}
  32 |   >
  33 |     <ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />
  34 |   </ScrollAreaPrimitive.ScrollAreaScrollbar>
  35 | ));
  36 | ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName;
  37 |
  38 | export { ScrollArea, ScrollBar };
  39 |
```

File: src/components/ui/select.tsx

Lines: 144

```
import * as React from "react";
import * as SelectPrimitive from "@radix-ui/react-select";
import { Check, ChevronDown, ChevronUp } from "lucide-react";

import { cn } from "@/lib/utils";

const Select = SelectPrimitive.Root;

const SelectGroup = SelectPrimitive.Group;

const SelectValue = SelectPrimitive.Value;

const SelectTrigger = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Trigger>
>(({ className, children, ...props }, ref) => (
  <SelectPrimitive.Trigger
    ref={ref}
    className={cn(
      "flex h-10 w-full items-center justify-between rounded-md border border-input bg-
background px-3 py-2 text-sm ring...
    )}
    {...props}
  >
    {children}
    <SelectPrimitive.Icon asChild>
      <ChevronDown className="h-4 w-4 opacity-50" />
    </SelectPrimitive.Icon>
  </SelectPrimitive.Trigger>
));
SelectTrigger.displayName = SelectPrimitive.Trigger.displayName;

const SelectScrollUpButton = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.ScrollUpButton>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollUpButton>
>(({ className, ...props }, ref) => (
  <SelectPrimitive.ScrollUpButton
    ref={ref}
    className={cn("flex cursor-default items-center justify-center py-1", className)}
    {...props}
  >
    <ChevronUp className="h-4 w-4" />
  </SelectPrimitive.ScrollUpButton>
));
SelectScrollUpButton.displayName = SelectPrimitive.ScrollUpButton.displayName;

const SelectScrollDownButton = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.ScrollDownButton>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollDownButton>
>(({ className, ...props }, ref) => (
  <SelectPrimitive.ScrollDownButton
    ref={ref}
    className={cn("flex cursor-default items-center justify-center py-1", className)}
    {...props}
  >
    <ChevronDown className="h-4 w-4" />
  </SelectPrimitive.ScrollDownButton>
));
SelectScrollDownButton.displayName = SelectPrimitive.ScrollDownButton.displayName;

const SelectContent = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Content>
>(({ className, children, position = "popper", ...props }, ref) => (
  <SelectPrimitive.Portal>
    <SelectPrimitive.Content
```

```
      ref={ref}
      className={cn(
        "relative z-50 max-h-96 min-w-[8rem] overflow-hidden rounded-md border bg-popover text-popover-foreground shadow-...
        position === "popper" &&
          "data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=t...
        className,
      )}
      position={position}
      {...props}
    >
      <SelectScrollUpButton />
      <SelectPrimitive.Viewport
        className={cn(
          "p-1",
          position === "popper" &&
            "h-[var(--radix-select-trigger-height)] w-full min-w-[var(--radix-select-trigger-width)]",
        )}
      >
        {children}
      </SelectPrimitive.Viewport>
      <SelectScrollDownButton />
    </SelectPrimitive.Content>
  </SelectPrimitive.Portal>
));
SelectContent.displayName = SelectPrimitive.Content.displayName;

const SelectLabel = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Label>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Label>
>(({ className, ...props }, ref) => (
  <SelectPrimitive.Label ref={ref} className={cn("py-1.5 pl-8 pr-2 text-sm font-semibold", className)} {...props} />
));
SelectLabel.displayName = SelectPrimitive.Label.displayName;

const SelectItem = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Item>
>(({ className, children, ...props }, ref) => (
  <SelectPrimitive.Item
    ref={ref}
    className={cn(
      "relative flex w-full cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none dat...
      className,
    )}
    {...props}
  >
    <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">
      <SelectPrimitive.ItemIndicator>
        <Check className="h-4 w-4" />
      </SelectPrimitive.ItemIndicator>
    </span>

    <SelectPrimitive.ItemText>{children}</SelectPrimitive.ItemText>
  </SelectPrimitive.Item>
));
SelectItem.displayName = SelectPrimitive.Item.displayName;

const SelectSeparator = React.forwardRef<
  React.ElementRef<typeof SelectPrimitive.Separator>,
  React.ComponentPropsWithoutRef<typeof SelectPrimitive.Separator>
>(({ className, ...props }, ref) => (
  <SelectPrimitive.Separator ref={ref} className={cn("-mx-1 my-1 h-px bg-muted", className)} {...props} />
));
SelectSeparator.displayName = SelectPrimitive.Separator.displayName;

export {
  Select,
  SelectGroup,
  SelectValue,
  SelectTrigger,
  SelectContent,
```

```
138 |   SelectLabel,
139 |   SelectItem,
140 |   SelectSeparator,
141 |   SelectScrollUpButton,
142 |   SelectScrollDownButton,
143 | };
144 |
```

File: src/components/ui/separator.tsx

Lines: 21

```
   1 | import * as React from "react";
   2 | import * as SeparatorPrimitive from "@radix-ui/react-separator";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const Separator = React.forwardRef<
   7 |   React.ElementRef<typeof SeparatorPrimitive.Root>,
   8 |   React.ComponentPropsWithoutRef<typeof SeparatorPrimitive.Root>
   9 | >(({ className, orientation = "horizontal", decorative = true, ...props }, ref) => (
  10 |   <SeparatorPrimitive.Root
  11 |     ref={ref}
  12 |     decorative={decorative}
  13 |     orientation={orientation}
  14 |     className={cn("shrink-0 bg-border", orientation === "horizontal" ? "h-[1px] w-full" : "h-
full w-[1px]", className)}
  15 |     {...props}
  16 |   />
  17 | ));
  18 | Separator.displayName = SeparatorPrimitive.Root.displayName;
  19 |
  20 | export { Separator };
  21 |
```

File: src/components/ui/sheet.tsx

Lines: 108

```
import * as SheetPrimitive from "@radix-ui/react-dialog";
import { cva, type VariantProps } from "class-variance-authority";
import { X } from "lucide-react";
import * as React from "react";

import { cn } from "@/lib/utils";

const Sheet = SheetPrimitive.Root;

const SheetTrigger = SheetPrimitive.Trigger;

const SheetClose = SheetPrimitive.Close;

const SheetPortal = SheetPrimitive.Portal;

const SheetOverlay = React.forwardRef<
  React.ElementRef<typeof SheetPrimitive.Overlay>,
  React.ComponentPropsWithoutRef<typeof SheetPrimitive.Overlay>
>(({ className, ...props }, ref) => (
  <SheetPrimitive.Overlay
    className={cn(
      "fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-
out data-[state=closed]:f...
    )}
    {...props}
    ref={ref}
  />
));
SheetOverlay.displayName = SheetPrimitive.Overlay.displayName;

const sheetVariants = cva(
  "fixed z-50 gap-4 bg-background p-6 shadow-lg transition ease-in-out data-[state=open]:animate-
in data-[state=closed]:...
  {
    variants: {
      side: {
        top: "inset-x-0 top-0 border-b data-[state=closed]:slide-out-to-top data-
[state=open]:slide-in-from-top",
        bottom:
          "inset-x-0 bottom-0 border-t data-[state=closed]:slide-out-to-bottom data-
[state=open]:slide-in-from-bottom",
        left: "inset-y-0 left-0 h-full w-3/4 border-r data-[state=closed]:slide-out-to-left data-
[state=open]:slide-in-f...
        right:
          "inset-y-0 right-0 h-full w-3/4  border-l data-[state=closed]:slide-out-to-right data-
[state=open]:slide-in-fr...
      },
    },
    defaultVariants: {
      side: "right",
    },
  },
);

interface SheetContentProps
  extends React.ComponentPropsWithoutRef<typeof SheetPrimitive.Content>,
    VariantProps<typeof sheetVariants> {}

const SheetContent = React.forwardRef<React.ElementRef<typeof SheetPrimitive.Content>,
SheetContentProps>(({ side
    <SheetPortal>
      <SheetOverlay />
      <SheetPrimitive.Content ref={ref} className={cn(sheetVariants({ side }), className)}
{...props}>
        {children}
        <SheetPrimitive.Close
offset-background transition...
          <X className="h-4 w-4" />
          <span className="sr-only">Close</span>
        </SheetPrimitive.Close>
      </SheetPrimitive.Content>
    </SheetPortal>
  ),
```

```
  67 | );
  68 | SheetContent.displayName = SheetPrimitive.Content.displayName;
  69 |
  70 | const SheetHeader = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  71 |   <div className={cn("flex flex-col space-y-2 text-center sm:text-left", className)} {...props} />
> 72 | );
  73 | SheetHeader.displayName = "SheetHeader";
  74 |
  75 | const SheetFooter = ({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) => (
  76 |   <div className={cn("flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2", className)} {...props} />
  77 | );
  78 | SheetFooter.displayName = "SheetFooter";
  79 |
  80 | const SheetTitle = React.forwardRef<
  81 |   React.ElementRef<typeof SheetPrimitive.Title>,
  82 |   React.ComponentPropsWithoutRef<typeof SheetPrimitive.Title>
  83 | >(({ className, ...props }, ref) => (
  84 |   <SheetPrimitive.Title ref={ref} className={cn("text-lg font-semibold text-foreground", className)} {...props} />
  85 | ));
  86 | SheetTitle.displayName = SheetPrimitive.Title.displayName;
  87 |
  88 | const SheetDescription = React.forwardRef<
  89 |   React.ElementRef<typeof SheetPrimitive.Description>,
  90 |   React.ComponentPropsWithoutRef<typeof SheetPrimitive.Description>
  91 | >(({ className, ...props }, ref) => (
  92 |   <SheetPrimitive.Description ref={ref} className={cn("text-sm text-muted-foreground", className)} {...props} />
  93 | ));
  94 | SheetDescription.displayName = SheetPrimitive.Description.displayName;
  95 |
  96 | export {
  97 |   Sheet,
  98 |   SheetClose,
  99 |   SheetContent,
 100 |   SheetDescription,
 101 |   SheetFooter,
 102 |   SheetHeader,
 103 |   SheetOverlay,
 104 |   SheetPortal,
 105 |   SheetTitle,
 106 |   SheetTrigger,
 107 | };
 108 |
```

File: src/components/ui/sidebar.tsx

Lines: 638

```
   1 | import * as React from "react";
   2 | import { Slot } from "@radix-ui/react-slot";
   3 | import { VariantProps, cva } from "class-variance-authority";
   4 | import { PanelLeft } from "lucide-react";
   5 |
   6 | import { useIsMobile } from "@/hooks/use-mobile";
   7 | import { cn } from "@/lib/utils";
   8 | import { Button } from "@/components/ui/button";
   9 | import { Input } from "@/components/ui/input";
  10 | import { Separator } from "@/components/ui/separator";
  11 | import { Sheet, SheetContent } from "@/components/ui/sheet";
  12 | import { Skeleton } from "@/components/ui/skeleton";
  13 | import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/
tooltip";
  14 |
  15 | const SIDEBAR_COOKIE_NAME = "sidebar:state";
  16 | const SIDEBAR_COOKIE_MAX_AGE = 60 * 60 * 24 * 7;
  17 | const SIDEBAR_WIDTH = "16rem";
  18 | const SIDEBAR_WIDTH_MOBILE = "18rem";
  19 | const SIDEBAR_WIDTH_ICON = "3rem";
  20 | const SIDEBAR_KEYBOARD_SHORTCUT = "b";
  21 |
  22 | type SidebarContext = {
  23 |   state: "expanded" | "collapsed";
  24 |   open: boolean;
  25 |   setOpen: (open: boolean) => void;
  26 |   openMobile: boolean;
  27 |   setOpenMobile: (open: boolean) => void;
  28 |   isMobile: boolean;
  29 |   toggleSidebar: () => void;
  30 | };
  31 |
  32 | const SidebarContext = React.createContext<SidebarContext | null>(null);
  33 |
  34 | function useSidebar() {
  35 |   const context = React.useContext(SidebarContext);
  36 |   if (!context) {
  37 |     throw new Error("useSidebar must be used within a SidebarProvider.");
  38 |   }
  39 |
  40 |   return context;
  41 | }
  42 |
  43 | const SidebarProvider = React.forwardRef<
  44 |   HTMLDivElement,
  45 |   React.ComponentProps<"div"> & {
  46 |     defaultOpen?: boolean;
  47 |     open?: boolean;
  48 |     onOpenChange?: (open: boolean) => void;
  49 |   }
  50 | >(({ defaultOpen = true, open: openProp, onOpenChange: setOpenProp, className, style,
children, ...props }, ref) => {
  51 |   const isMobile = useIsMobile();
  52 |   const [openMobile, setOpenMobile] = React.useState(false);
  53 |
  54 |   // This is the internal state of the sidebar.
  55 |   // We use openProp and setOpenProp for control from outside the component.
  56 |   const [_open, _setOpen] = React.useState(defaultOpen);
  57 |   const open = openProp ?? _open;
  58 |   const setOpen = React.useCallback(
  59 |     (value: boolean | ((value: boolean) => boolean)) => {
  60 |       const openState = typeof value === "function" ? value(open) : value;
  61 |       if (setOpenProp) {
  62 |         setOpenProp(openState);
  63 |       } else {
  64 |         _setOpen(openState);
  65 |       }
  66 |
```

```
      // This sets the cookie to keep the sidebar state.
      document.cookie = `${SIDEBAR_COOKIE_NAME}=${openState}; path=/; max-age=${SIDEBAR_COOKIE_MAX_AGE}`;
    },
    [setOpenProp, open],
  );

  // Helper to toggle the sidebar.
  const toggleSidebar = React.useCallback(() => {
    return isMobile ? setOpenMobile((open) => !open) : setOpen((open) => !open);
  }, [isMobile, setOpen, setOpenMobile]);

  // Adds a keyboard shortcut to toggle the sidebar.
  React.useEffect(() => {
    const handleKeyDown = (event: KeyboardEvent) => {
      if (event.key === SIDEBAR_KEYBOARD_SHORTCUT && (event.metaKey || event.ctrlKey)) {
        event.preventDefault();
        toggleSidebar();
      }
    };

    window.addEventListener("keydown", handleKeyDown);
    return () => window.removeEventListener("keydown", handleKeyDown);
  }, [toggleSidebar]);

  // We add a state so that we can do data-state="expanded" or "collapsed".
  // This makes it easier to style the sidebar with Tailwind classes.
  const state = open ? "expanded" : "collapsed";

  const contextValue = React.useMemo<SidebarContext>(
    () => ({
      state,
      open,
      setOpen,
      isMobile,
      openMobile,
      setOpenMobile,
      toggleSidebar,
    }),
    [state, open, setOpen, isMobile, openMobile, setOpenMobile, toggleSidebar],
  );

  return (
    <SidebarContext.Provider value={contextValue}>
      <TooltipProvider delayDuration={0}>
        <div
          style={
            {
              "--sidebar-width": SIDEBAR_WIDTH,
              "--sidebar-width-icon": SIDEBAR_WIDTH_ICON,
              ...style,
            } as React.CSSProperties
          }
          ref={ref}
          {...props}
        >
          {children}
        </div>
      </TooltipProvider>
    </SidebarContext.Provider>
  );
});
SidebarProvider.displayName = "SidebarProvider";

const Sidebar = React.forwardRef<
  HTMLDivElement,
  React.ComponentProps<"div"> & {
    side?: "left" | "right";
    variant?: "sidebar" | "floating" | "inset";
    collapsible?: "offcanvas" | "icon" | "none";
  }
```

```
138 | >(({ side = "left", variant = "sidebar", collapsible = "offcanvas", className,
children, ...props }, ref) => {
139 |     const { isMobile, state, openMobile, setOpenMobile } = useSidebar();
140 |
141 |     if (collapsible === "none") {
142 |       return (
143 |         <div
144 |           className={cn("flex h-full w-[--sidebar-width] flex-col bg-sidebar text-sidebar-
foreground", className)}
145 |           ref={ref}
146 |           {...props}
147 |         >
148 |           {children}
149 |         </div>
150 |       );
151 |     }
152 |
153 |     if (isMobile) {
154 |       return (
155 |         <Sheet open={openMobile} onOpenChange={setOpenMobile} {...props}>
156 |           <SheetContent
157 |             data-sidebar="sidebar"
158 |             data-mobile="true"
159 |             className="w-[--sidebar-width] bg-sidebar p-0 text-sidebar-foreground
[&>button]:hidden"
160 |             style={
161 |               {
162 |                 "--sidebar-width": SIDEBAR_WIDTH_MOBILE,
163 |               } as React.CSSProperties
164 |             }
165 |             side={side}
166 |           >
167 |             <div className="flex h-full w-full flex-col">{children}</div>
168 |           </SheetContent>
169 |         </Sheet>
170 |       );
171 |     }
172 |
173 |     return (
174 |       <div
175 |         ref={ref}
176 |         className="group peer hidden text-sidebar-foreground md:block"
177 |         data-state={state}
178 |         data-collapsible={state === "collapsed" ? collapsible : ""}
179 |         data-variant={variant}
180 |         data-side={side}
181 |       >
182 |         {/* This is what handles the sidebar gap on desktop */}
183 |         <div
184 |           className={cn(
185 |             "relative h-svh w-[--sidebar-width] bg-transparent transition-[width] duration-200
ease-linear",
186 |             "group-data-[collapsible=offcanvas]:w-0",
187 |             "group-data-[side=right]:rotate-180",
188 |             variant === "floating" || variant === "inset"
189 |               ? "group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-
icon)+_theme(spacing.4))]"
190 |               : "group-data-[collapsible=icon]:w-[--sidebar-width-icon]",
191 |           )}
192 |         />
193 |         <div
194 |           className={cn(
195 |             "fixed inset-y-0 z-10 hidden h-svh w-[--sidebar-width] transition-[left,right,width]
duration-200 ease-linear",
196 |             side === "left"
197 |               ? "left-0 group-data-[collapsible=offcanvas]:left-[calc(var(--sidebar-width)*-1)]"
198 |               : "right-0 group-data-[collapsible=offcanvas]:right-[calc(var(--sidebar-width)*-1)]",
199 |             // Adjust the padding for floating and inset variants.
200 |             variant === "floating" || variant === "inset"
201 |               ? "p-2 group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-
icon)+_theme(spacing.4)+2px)]"
202 |               : "group-data-[collapsible=icon]:w-[--sidebar-width-icon] group-data-
[side=left]:border-r group-data-[side=r...
203 |             className,
204 |           )}
205 |           {...props}
206 |         >
207 |           <div
208 |             data-sidebar="sidebar"
```

```
          className="flex h-full w-full flex-col bg-sidebar group-data-
[variant=floating]:rounded-lg group-data-[variant...
          {children}
        </div>
      </div>
    </div>
  );
});
Sidebar.displayName = "Sidebar";

const SidebarTrigger = React.forwardRef<React.ElementRef<typeof Button>,
React.ComponentProps<typeof Button>>(
  ({ className, onClick, ...props }, ref) => {
    const { toggleSidebar } = useSidebar();

    return (
      <Button
        ref={ref}
        data-sidebar="trigger"
        variant="ghost"
        size="icon"
        className={cn("h-7 w-7", className)}
        onClick={(event) => {
          onClick?.(event);
          toggleSidebar();
        }}
        {...props}
      >
        <PanelLeft />
        <span className="sr-only">Toggle Sidebar</span>
      </Button>
    );
  },
);
SidebarTrigger.displayName = "SidebarTrigger";

const SidebarRail = React.forwardRef<HTMLButtonElement, React.ComponentProps<"button">>(
  ({ className, ...props }, ref) => {
    const { toggleSidebar } = useSidebar();

    return (
      <button
        ref={ref}
        data-sidebar="rail"
        aria-label="Toggle Sidebar"
        tabIndex={-1}
        onClick={toggleSidebar}
        title="Toggle Sidebar"
        className={cn(
          "absolute inset-y-0 z-20 hidden w-4 -translate-x-1/2 transition-all ease-linear
after:absolute after:inset-y-0 [[data-side=left]_&]:cursor-w-resize [[data-side=right]_&]:cursor-e-resize",
          "[[data-side=left][data-state=collapsed]_&]:cursor-e-resize [[data-side=right][data-
state=collapsed]_&]:cursor-w-resize",
          "group-data-[collapsible=offcanvas]:translate-x-0 group-data-
[collapsible=offcanvas]:after:left-full group-dat...
          "[[data-side=left][data-collapsible=offcanvas]_&]:-right-2",
          "[[data-side=right][data-collapsible=offcanvas]_&]:-left-2",
          className,
        )}
        {...props}
      />
    );
  },
);
SidebarRail.displayName = "SidebarRail";

const SidebarInset = React.forwardRef<HTMLDivElement,
React.ComponentProps<"main">>(({ className, ...props }, ref) => {
  return (
    <main
      ref={ref}
      className={cn(
        "relative flex min-h-svh flex-1 flex-col bg-background",
        "peer-data-[variant=inset]:min-h-[calc(100svh-theme(spacing.4))] md:peer-data-
[variant=inset]:m-2 md:peer-data-[...
        className,
```

```
      )}
      {...props}
    />
  );
});
SidebarInset.displayName = "SidebarInset";

const SidebarInput = React.forwardRef<React.ElementRef<typeof Input>,
React.ComponentProps<typeof Input>>(({ className, ...props }, ref) => {
    return (
      <Input
        ref={ref}
        data-sidebar="input"
        className={cn(
          "h-8 w-full bg-background shadow-none focus-visible:ring-2 focus-visible:ring-sidebar-ring",
          className,
        )}
        {...props}
      />
    );
  },
);
SidebarInput.displayName = "SidebarInput";

const SidebarHeader = React.forwardRef<HTMLDivElement,
React.ComponentProps<"div">>(({ className, ...props }, ref) => {
  return <div ref={ref} data-sidebar="header" className={cn("flex flex-col gap-2 p-2", className)} {...props} />;
});
SidebarHeader.displayName = "SidebarHeader";

const SidebarFooter = React.forwardRef<HTMLDivElement,
React.ComponentProps<"div">>(({ className, ...props }, ref) => {
  return <div ref={ref} data-sidebar="footer" className={cn("flex flex-col gap-2 p-2", className)} {...props} />;
});
SidebarFooter.displayName = "SidebarFooter";

const SidebarSeparator = React.forwardRef<React.ElementRef<typeof Separator>,
React.ComponentProps<typeof Separator>>(({ className, ...props }, ref) => {
    return (
      <Separator
        ref={ref}
        data-sidebar="separator"
        className={cn("mx-2 w-auto bg-sidebar-border", className)}
        {...props}
      />
    );
  },
);
SidebarSeparator.displayName = "SidebarSeparator";

const SidebarContent = React.forwardRef<HTMLDivElement,
React.ComponentProps<"div">>(({ className, ...props }, ref) => {
    return (
      <div
        ref={ref}
        data-sidebar="content"
        className={cn(
          "flex min-h-0 flex-1 flex-col gap-2 overflow-auto group-data-[collapsible=icon]:overflow-hidden",
          className,
        )}
        {...props}
      />
    );
});
SidebarContent.displayName = "SidebarContent";

const SidebarGroup = React.forwardRef<HTMLDivElement,
React.ComponentProps<"div">>(({ className, ...props }, ref) => {
    return (
      <div
        ref={ref}
        data-sidebar="group"
        className={cn("relative flex w-full min-w-0 flex-col p-2", className)}
        {...props}
      />
    );
```

```
 351 |   );
 352 | });
 353 | SidebarGroup.displayName = "SidebarGroup";
 354 |
 355 | const SidebarGroupLabel = React.forwardRef<HTMLDivElement, React.ComponentProps<"div"> &
{asChild?: boolean}>
 356 |   ({ className, asChild = false, ...props }, ref) => {
 357 |     const Comp = asChild ? Slot : "div";
 358 |
 359 |     return (
 360 |       <Comp
 361 |         ref={ref}
 362 |         data-sidebar="group-label"
 363 |         className={cn(
 364 |           "flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium text-sidebar-
foreground/70 outline-none ri...
 365 |           "group-data-[collapsible=icon]:-mt-8 group-data-[collapsible=icon]:opacity-0",
 366 |           className,
 367 |         )}
 368 |         {...props}
 369 |       />
 370 |     );
 371 |   },
 372 | );
 373 | SidebarGroupLabel.displayName = "SidebarGroupLabel";
 374 |
 375 | const SidebarGroupAction = React.forwardRef<HTMLButtonElement, React.ComponentProps<"button"> &
{asChild?: boolean}>
 376 |   ({ className, asChild = false, ...props }, ref) => {
 377 |     const Comp = asChild ? Slot : "button";
 378 |
 379 |     return (
 380 |       <Comp
 381 |         ref={ref}
 382 |         data-sidebar="group-action"
 383 |         className={cn(
 384 |           "absolute right-3 top-3.5 flex aspect-square w-5 items-center justify-center rounded-
md p-0 text-sidebar-foreg...
 385 |           // Increases the hit area of the button on mobile.
 386 |           "after:absolute after:-inset-2 after:md:hidden",
 387 |           "group-data-[collapsible=icon]:hidden",
 388 |           className,
 389 |         )}
 390 |         {...props}
 391 |       />
 392 |     );
 393 |   },
 394 | );
 395 | SidebarGroupAction.displayName = "SidebarGroupAction";
 396 |
 397 | const SidebarGroupContent = React.forwardRef<HTMLDivElement, React.ComponentProps<"div">>(
 398 |   ({ className, ...props }, ref) => (
 399 |     <div ref={ref} data-sidebar="group-content" className={cn("w-full text-sm", className)}
{...props} />
 400 |   ),
 401 | );
 402 | SidebarGroupContent.displayName = "SidebarGroupContent";
 403 |
 404 | const SidebarMenu = React.forwardRef<HTMLUListElement,
React.ComponentProps<"ul">>(({ className, ...props }, ref) => (
 405 |   <ul ref={ref} data-sidebar="menu" className={cn("flex w-full min-w-0 flex-col gap-1",
className)} {...props} />
 406 | ));
 407 | SidebarMenu.displayName = "SidebarMenu";
 408 |
 409 | const SidebarMenuItem = React.forwardRef<HTMLLIElement,
React.ComponentProps<"li">>(({ className, ...props }, ref) => (
 410 |   <li ref={ref} data-sidebar="menu-item" className={cn("group/menu-item relative", className)}
{...props} />
 411 | ));
 412 | SidebarMenuItem.displayName = "SidebarMenuItem";
 413 |
 414 | const sidebarMenuButtonVariants = cva(
 415 |   "peer/menu-button flex w-full items-center gap-2 overflow-hidden rounded-md p-2 text-left text-
sm outline-none ring-si...
 416 |   {
 417 |     variants: {
 418 |       variant: {
 419 |         default: "hover:bg-sidebar-accent hover:text-sidebar-accent-foreground",
 420 |         outline:
 421 |           "bg-background shadow-[0_0_0_1px_hsl(var(--sidebar-border))] hover:bg-sidebar-accent
hover:text-sidebar-accent...
```

```
      },
      size: {
        default: "h-8 text-sm",
        sm: "h-7 text-xs",
        lg: "h-12 text-sm group-data-[collapsible=icon]:!p-0",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
);

const SidebarMenuButton = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button"> & {
    asChild?: boolean;
    isActive?: boolean;
    tooltip?: string | React.ComponentProps<typeof TooltipContent>;
  } & VariantProps<typeof sidebarMenuButtonVariants>
>(({ asChild = false, isActive = false, variant = "default", size = "default", tooltip, className, ...props }, ref) => {
  const Comp = asChild ? Slot : "button";
  const { isMobile, state } = useSidebar();

  const button = (
    <Comp
      ref={ref}
      data-sidebar="menu-button"
      data-size={size}
      data-active={isActive}
      className={cn(sidebarMenuButtonVariants({ variant, size }), className)}
      {...props}
    />
  );

  if (!tooltip) {
    return button;
  }

  if (typeof tooltip === "string") {
    tooltip = {
      children: tooltip,
    };
  }

  return (
    <Tooltip>
      <TooltipTrigger asChild>{button}</TooltipTrigger>
      <TooltipContent side="right" align="center" hidden={state !== "collapsed" || isMobile}
{...tooltip} />
    </Tooltip>
  );
});
SidebarMenuButton.displayName = "SidebarMenuButton";

const SidebarMenuAction = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<"button"> & {
    asChild?: boolean;
    showOnHover?: boolean;
  }
>(({ className, asChild = false, showOnHover = false, ...props }, ref) => {
  const Comp = asChild ? Slot : "button";

  return (
    <Comp
      ref={ref}
      data-sidebar="menu-action"
      className={cn(
        "absolute right-1 top-1.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sidebar-foreground
        // Increases the hit area of the button on mobile.
```

```
493 |         "after:absolute after:-inset-2 after:md:hidden",
494 |         "peer-data-[size=sm]/menu-button:top-1",
495 |         "peer-data-[size=default]/menu-button:top-1.5",
496 |         "peer-data-[size=lg]/menu-button:top-2.5",
497 |         "group-data-[collapsible=icon]:hidden",
498 |         showOnHover &&
499 |           "group-focus-within/menu-item:opacity-100 group-hover/menu-item:opacity-100 data-
[state=open]:opacity-100 peer...
500 |         className,
501 |       )}
502 |       {...props}
503 |     />
504 |   );
505 | });
506 | SidebarMenuAction.displayName = "SidebarMenuAction";
507 |
508 | const SidebarMenuBadge = React.forwardRef<HTMLDivElement, React.ComponentProps<"div">>(
509 |   ({ className, ...props }, ref) => (
510 |     <div
511 |       ref={ref}
512 |       data-sidebar="menu-badge"
513 |       className={cn(
514 |         "pointer-events-none absolute right-1 flex h-5 min-w-5 select-none items-center justify-
center rounded-md px-1 t...
515 |         "peer-hover/menu-button:text-sidebar-accent-foreground peer-data-[active=true]/menu-
button:text-sidebar-accent-f...
516 |         "peer-data-[size=sm]/menu-button:top-1",
517 |         "peer-data-[size=default]/menu-button:top-1.5",
518 |         "peer-data-[size=lg]/menu-button:top-2.5",
519 |         "group-data-[collapsible=icon]:hidden",
520 |         className,
521 |       )}
522 |       {...props}
523 |     />
524 |   ),
525 | );
526 | SidebarMenuBadge.displayName = "SidebarMenuBadge";
527 |
528 | const SidebarMenuSkeleton = React.forwardRef<
529 |   HTMLDivElement,
530 |   React.ComponentProps<"div"> & {
531 |     showIcon?: boolean;
532 |   }
533 | >(({ className, showIcon = false, ...props }, ref) => {
534 |   // Random width between 50 to 90%.
535 |   const width = React.useMemo(() => {
536 |     return `${Math.floor(Math.random() * 40) + 50}%`;
537 |   }, []);
538 |
539 |   return (
540 |     <div
541 |       ref={ref}
542 |       data-sidebar="menu-skeleton"
543 |       className={cn("flex h-8 items-center gap-2 rounded-md px-2", className)}
544 |       {...props}
545 |     >
546 |       {showIcon && <Skeleton className="size-4 rounded-md" data-sidebar="menu-skeleton-icon" />}
547 |       <Skeleton
548 |         className="h-4 max-w-[--skeleton-width] flex-1"
549 |         data-sidebar="menu-skeleton-text"
550 |         style={
551 |           {
552 |             "--skeleton-width": width,
553 |           } as React.CSSProperties
554 |         }
555 |       />
556 |     </div>
557 |   );
558 | });
559 | SidebarMenuSkeleton.displayName = "SidebarMenuSkeleton";
560 |
561 | const SidebarMenuSub = React.forwardRef<HTMLUListElement, React.ComponentProps<"ul">>(
562 |   ({ className, ...props }, ref) => (
563 |     <ul
```

```
      ref={ref}
      data-sidebar="menu-sub"
      className={cn(
        "mx-3.5 flex min-w-0 translate-x-px flex-col gap-1 border-l border-sidebar-border px-2.5 py-0.5",
        "group-data-[collapsible=icon]:hidden",
        className,
      )}
      {...props}
    />
  ),
);
SidebarMenuSub.displayName = "SidebarMenuSub";

const SidebarMenuSubItem = React.forwardRef<HTMLLIElement, React.ComponentProps<"li">>(({ ...props }, ref) => (
  <li ref={ref} {...props} />
));
SidebarMenuSubItem.displayName = "SidebarMenuSubItem";

const SidebarMenuSubButton = React.forwardRef<
  HTMLAnchorElement,
  React.ComponentProps<"a"> & {
    asChild?: boolean;
    size?: "sm" | "md";
    isActive?: boolean;
  }
>(({ asChild = false, size = "md", isActive, className, ...props }, ref) => {
  const Comp = asChild ? Slot : "a";

  return (
    <Comp
      ref={ref}
      data-sidebar="menu-sub-button"
      data-size={size}
      data-active={isActive}
      className={cn(
        "flex h-7 min-w-0 -translate-x-px items-center gap-2 overflow-hidden rounded-md px-2 text-sidebar-foreground outl
        "data-[active=true]:bg-sidebar-accent data-[active=true]:text-sidebar-accent-foreground",
        size === "sm" && "text-xs",
        size === "md" && "text-sm",
        "group-data-[collapsible=icon]:hidden",
        className,
      )}
      {...props}
    />
  );
});
SidebarMenuSubButton.displayName = "SidebarMenuSubButton";

export {
  Sidebar,
  SidebarContent,
  SidebarFooter,
  SidebarGroup,
  SidebarGroupAction,
  SidebarGroupContent,
  SidebarGroupLabel,
  SidebarHeader,
  SidebarInput,
  SidebarInset,
  SidebarMenu,
  SidebarMenuAction,
  SidebarMenuBadge,
  SidebarMenuButton,
  SidebarMenuItem,
  SidebarMenuSkeleton,
  SidebarMenuSub,
  SidebarMenuSubButton,
  SidebarMenuSubItem,
  SidebarProvider,
  SidebarRail,
  SidebarSeparator,
```

```
635 |   SidebarTrigger,
636 |   useSidebar,
637 | };
638 |
```

File: src/components/ui/skeleton.tsx

Lines: 8

```
1 | import { cn } from "@/lib/utils";
2 |
3 | function Skeleton({ className, ...props }: React.HTMLAttributes<HTMLDivElement>) {
4 |   return <div className={cn("animate-pulse rounded-md bg-muted", className)} {...props} />;
5 | }
6 |
7 | export { Skeleton };
8 |
```

File: src/components/ui/slider.tsx

Lines: 24

```
   1 | import * as React from "react";
   2 | import * as SliderPrimitive from "@radix-ui/react-slider";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const Slider = React.forwardRef<
   7 |   React.ElementRef<typeof SliderPrimitive.Root>,
   8 |   React.ComponentPropsWithoutRef<typeof SliderPrimitive.Root>
   9 | >(({ className, ...props }, ref) => (
  10 |   <SliderPrimitive.Root
  11 |     ref={ref}
  12 |     className={cn("relative flex w-full touch-none select-none items-center", className)}
  13 |     {...props}
  14 |   >
  15 |     <SliderPrimitive.Track className="relative h-2 w-full grow overflow-hidden rounded-full bg-
secondary">  <SliderPrimitive.Range className="absolute h-full bg-primary" />
  17 |     </SliderPrimitive.Track>
  18 |     <SliderPrimitive.Thumb className="block h-5 w-5 rounded-full border-2 border-primary bg-
background ring-offset-backg
  19 |   </SliderPrimitive.Root>
  20 | ));
  21 | Slider.displayName = SliderPrimitive.Root.displayName;
  22 |
  23 | export { Slider };
  24 |
```

File: src/components/ui/sonner.tsx

Lines: 30

```
  1 | import { useTheme } from "next-themes";
  2 | import { Toaster as Sonner, toast } from "sonner";
  3 |
  4 | type ToasterProps = React.ComponentProps<typeof Sonner>;
  5 |
  6 | const Toaster = ({ ...props }: ToasterProps) => {
  7 |   const { theme = "system" } = useTheme();
  8 |
  9 |   return (
 10 |     <Sonner
 11 |       theme={theme as ToasterProps["theme"]}
 12 |       position="top-center"
 13 |       className="toaster group"
 14 |       style={{ zIndex: 100 }}
 15 |       toastOptions={{
 16 |         classNames: {
 17 |           toast:
 18 |             "group toast group-[.toaster]:bg-background group-[.toaster]:text-foreground group-
[.toaster]:border-border
 19 |           description: "group-[.toast]:text-muted-foreground",
 20 |           actionButton: "group-[.toast]:bg-primary group-[.toast]:text-primary-foreground",
 21 |           cancelButton: "group-[.toast]:bg-muted group-[.toast]:text-muted-foreground",
 22 |         },
 23 |       }}
 24 |       {...props}
 25 |     />
 26 |   );
 27 | };
 28 |
 29 | export { Toaster, toast };
 30 |
```

File: src/components/ui/switch.tsx

Lines: 28

```
   1 | import * as React from "react";
   2 | import * as SwitchPrimitives from "@radix-ui/react-switch";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const Switch = React.forwardRef<
   7 |   React.ElementRef<typeof SwitchPrimitives.Root>,
   8 |   React.ComponentPropsWithoutRef<typeof SwitchPrimitives.Root>
   9 | >(({ className, ...props }, ref) => (
  10 |   <SwitchPrimitives.Root
  11 |     className={cn(
  12 |       "peer inline-flex h-6 w-11 shrink-0 cursor-pointer items-center rounded-full border-2
border-transparent transitio...
  13 |       className,
  14 |     )}
  15 |     {...props}
  16 |     ref={ref}
  17 |   >
  18 |     <SwitchPrimitives.Thumb
  19 |       className={cn(
  20 |         "pointer-events-none block h-5 w-5 rounded-full bg-background shadow-lg ring-0
transition-transform data-[state=...
  21 |       )}
  22 |     />
  23 |   </SwitchPrimitives.Root>
  24 | ));
  25 | Switch.displayName = SwitchPrimitives.Root.displayName;
  26 |
  27 | export { Switch };
  28 |
```

File: src/components/ui/table.tsx

Lines: 73

```
import * as React from "react";

import { cn } from "@/lib/utils";

const Table = React.forwardRef<HTMLTableElement, React.HTMLAttributes<HTMLTableElement>>(
  ({ className, ...props }, ref) => (
    <div className="relative w-full overflow-auto">
      <table ref={ref} className={cn("w-full caption-bottom text-sm", className)} {...props} />
    </div>
  ),
);
Table.displayName = "Table";

const TableHeader = React.forwardRef<HTMLTableSectionElement,
React.HTMLAttributes<HTMLTableSectionElement>>(({ className, ...props }, ref) => <thead ref={ref} className={cn("[&_tr]:border-b",
className)} {...props} />,
);
TableHeader.displayName = "TableHeader";

const TableBody = React.forwardRef<HTMLTableSectionElement,
React.HTMLAttributes<HTMLTableSectionElement>>(({ className, ...props }, ref) => (
  <tbody ref={ref} className={cn("[&_tr:last-child]:border-0", className)} {...props} />
),
);
TableBody.displayName = "TableBody";

const TableFooter = React.forwardRef<HTMLTableSectionElement,
React.HTMLAttributes<HTMLTableSectionElement>>(({ className, ...props }, ref) => (
  <tfoot ref={ref} className={cn("border-t bg-muted/50 font-medium [&>tr]:last:border-b-0",
className)} {...props} />
));
TableFooter.displayName = "TableFooter";

const TableRow = React.forwardRef<HTMLTableRowElement,
React.HTMLAttributes<HTMLTableRowElement>>(({ className, ...props }, ref) => (
  <tr
    ref={ref}
    className={cn("border-b transition-colors data-[state=selected]:bg-muted hover:bg-
muted/50", className)}
    {...props}
  />
),
);
TableRow.displayName = "TableRow";

const TableHead = React.forwardRef<HTMLTableCellElement,
React.ThHTMLAttributes<HTMLTableCellElement>>(({ className, ...props }, ref) => (
  <th
    ref={ref}
    className={cn(
      "h-12 px-4 text-left align-middle font-medium text-muted-foreground
[&:has([role=checkbox])]:pr-0",
      className,
    )}
    {...props}
  />
),
);
TableHead.displayName = "TableHead";

const TableCell = React.forwardRef<HTMLTableCellElement,
React.TdHTMLAttributes<HTMLTableCellElement>>(({ className, ...props }, ref) => (
  <td ref={ref} className={cn("p-4 align-middle [&:has([role=checkbox])]:pr-0", className)}
{...props} />
),
);
TableCell.displayName = "TableCell";

const TableCaption = React.forwardRef<HTMLTableCaptionElement,
React.HTMLAttributes<HTMLTableCaptionElement>>(({ className, ...props }, ref) => (
```

```
  67 |     <caption ref={ref} className={cn("mt-4 text-sm text-muted-foreground", className)}
{...props} />
  68 | ,
  69 | );
  70 | TableCaption.displayName = "TableCaption";
  71 |
  72 | export { Table, TableHeader, TableBody, TableFooter, TableHead, TableRow, TableCell,
TableCaption };
  73 |
```

File: src/components/ui/tabs.tsx

Lines: 54

```
   1 | import * as React from "react";
   2 | import * as TabsPrimitive from "@radix-ui/react-tabs";
   3 |
   4 | import { cn } from "@/lib/utils";
   5 |
   6 | const Tabs = TabsPrimitive.Root;
   7 |
   8 | const TabsList = React.forwardRef<
   9 |   React.ElementRef<typeof TabsPrimitive.List>,
  10 |   React.ComponentPropsWithoutRef<typeof TabsPrimitive.List>
  11 | >(({ className, ...props }, ref) => (
  12 |   <TabsPrimitive.List
  13 |     ref={ref}
  14 |     className={cn(
  15 |       "inline-flex h-10 items-center justify-center rounded-md bg-muted p-1 text-muted-
foreground", className,
  16 |     )}
  17 |     {...props}
  18 |   />
  19 | ));
  20 | TabsList.displayName = TabsPrimitive.List.displayName;
  21 |
  22 | const TabsTrigger = React.forwardRef<
  23 |   React.ElementRef<typeof TabsPrimitive.Trigger>,
  24 |   React.ComponentPropsWithoutRef<typeof TabsPrimitive.Trigger>
  25 | >(({ className, ...props }, ref) => (
  26 |   <TabsPrimitive.Trigger
  27 |     ref={ref}
  28 |     className={cn(
  29 |       "inline-flex items-center justify-center whitespace-nowrap rounded-sm px-3 py-1.5 text-sm
font-medium ring-offset-...
  30 |       className,
  31 |     )}
  32 |     {...props}
  33 |   />
  34 | ));
  35 | TabsTrigger.displayName = TabsPrimitive.Trigger.displayName;
  36 |
  37 | const TabsContent = React.forwardRef<
  38 |   React.ElementRef<typeof TabsPrimitive.Content>,
  39 |   React.ComponentPropsWithoutRef<typeof TabsPrimitive.Content>
  40 | >(({ className, ...props }, ref) => (
  41 |   <TabsPrimitive.Content
  42 |     ref={ref}
  43 |     className={cn(
  44 |       "mt-2 ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-
visible:ring-ring focus-visible...
  45 |       className,
  46 |     )}
  47 |     {...props}
  48 |   />
  49 | ));
  50 | TabsContent.displayName = TabsPrimitive.Content.displayName;
  51 |
  52 | export { Tabs, TabsList, TabsTrigger, TabsContent };
  53 |
  54 |
```

File: src/components/ui/textarea.tsx

Lines: 22

```
   1 | import * as React from "react";
   2 |
   3 | import { cn } from "@/lib/utils";
   4 |
   5 | export interface TextareaProps extends React.TextareaHTMLAttributes<HTMLTextAreaElement> {}
   6 |
   7 | const Textarea = React.forwardRef<HTMLTextAreaElement, TextareaProps>(({ className, ...props },
ref) => {return (
   9 |     <textarea
  10 |       className={cn(
  11 |         "flex min-h-[80px] w-full rounded-md border border-input bg-background px-3 py-2 text-sm
ring-offset-background",
  12 |         className,
  13 |       )}
  14 |       ref={ref}
  15 |       {...props}
  16 |     />
  17 |   );
  18 | });
  19 | Textarea.displayName = "Textarea";
  20 |
  21 | export { Textarea };
  22 |
```

File: src/components/ui/toast.tsx

Lines: 112

```tsx
import * as React from "react";
import * as ToastPrimitives from "@radix-ui/react-toast";
import { cva, type VariantProps } from "class-variance-authority";
import { X } from "lucide-react";

import { cn } from "@/lib/utils";

const ToastProvider = ToastPrimitives.Provider;

const ToastViewport = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Viewport>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Viewport>
>(({ className, ...props }, ref) => (
  <ToastPrimitives.Viewport
    ref={ref}
    className={cn(
      "fixed top-0 z-[100] flex max-h-screen w-full flex-col-reverse p-4 sm:bottom-0 sm:right-0
sm:top-auto sm:flex-col ...
      className,
    )}
    {...props}
  />
));
ToastViewport.displayName = ToastPrimitives.Viewport.displayName;

const toastVariants = cva(
  "group pointer-events-auto relative flex w-full items-center justify-between space-x-4
overflow-hidden rounded-md bord...
  {
    variants: {
      variant: {
        default: "border bg-background text-foreground",
        destructive: "destructive group border-destructive bg-destructive text-destructive-
foreground", },
    },
    defaultVariants: {
      variant: "default",
    },
  },
);

const Toast = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Root>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Root> & VariantProps<typeof
toastVariants>
>(({ className, variant, ...props }, ref) => {
  return <ToastPrimitives.Root ref={ref} className={cn(toastVariants({ variant }), className)}
{...props} />;
});
Toast.displayName = ToastPrimitives.Root.displayName;

const ToastAction = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Action>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Action>
>(({ className, ...props }, ref) => (
  <ToastPrimitives.Action
    ref={ref}
    className={cn(
      "inline-flex h-8 shrink-0 items-center justify-center rounded-md border bg-transparent
px-3 text-sm font-medium ri...
      className,
    )}
    {...props}
  />
));
ToastAction.displayName = ToastPrimitives.Action.displayName;

const ToastClose = React.forwardRef<
  React.ElementRef<typeof ToastPrimitives.Close>,
  React.ComponentPropsWithoutRef<typeof ToastPrimitives.Close>
>(({ className, ...props }, ref) => (
```

```
  67 |   <ToastPrimitives.Close
  68 |     ref={ref}
  69 |     className={cn(
  70 |       "absolute right-2 top-2 rounded-md p-1 text-foreground/50 opacity-0 transition-opacity
group-hover:opacity-100 gro...
  71 |       className,
  72 |     )}
  73 |     toast-close=""
  74 |     {...props}
  75 |   >
  76 |     <X className="h-4 w-4" />
  77 |   </ToastPrimitives.Close>
  78 | ));
  79 | ToastClose.displayName = ToastPrimitives.Close.displayName;
  80 |
  81 | const ToastTitle = React.forwardRef<
  82 |   React.ElementRef<typeof ToastPrimitives.Title>,
  83 |   React.ComponentPropsWithoutRef<typeof ToastPrimitives.Title>
  84 | >(({ className, ...props }, ref) => (
  85 |   <ToastPrimitives.Title ref={ref} className={cn("text-sm font-semibold", className)}
{...props} />
  86 | ));
  87 | ToastTitle.displayName = ToastPrimitives.Title.displayName;
  88 |
  89 | const ToastDescription = React.forwardRef<
  90 |   React.ElementRef<typeof ToastPrimitives.Description>,
  91 |   React.ComponentPropsWithoutRef<typeof ToastPrimitives.Description>
  92 | >(({ className, ...props }, ref) => (
  93 |   <ToastPrimitives.Description ref={ref} className={cn("text-sm opacity-90", className)}
{...props} />
  94 | ));
  95 | ToastDescription.displayName = ToastPrimitives.Description.displayName;
  96 |
  97 | type ToastProps = React.ComponentPropsWithoutRef<typeof Toast>;
  98 |
  99 | type ToastActionElement = React.ReactElement<typeof ToastAction>;
 100 |
 101 | export {
 102 |   type ToastProps,
 103 |   type ToastActionElement,
 104 |   ToastProvider,
 105 |   ToastViewport,
 106 |   Toast,
 107 |   ToastTitle,
 108 |   ToastDescription,
 109 |   ToastClose,
 110 |   ToastAction,
 111 | };
 112 |
```

File: src/components/ui/toaster.tsx

Lines: 25

```
  1 | import { useToast } from "@/hooks/use-toast";
  2 | import { Toast, ToastClose, ToastDescription, ToastProvider, ToastTitle, ToastViewport } from "@/
components/ui/toast";
  3 |
  4 | export function Toaster() {
  5 |   const { toasts } = useToast();
  6 |
  7 |   return (
  8 |     <ToastProvider>
  9 |       {toasts.map(function ({ id, title, description, action, ...props }) {
 10 |         return (
 11 |           <Toast key={id} {...props}>
 12 |             <div className="grid gap-1">
 13 |               {title && <ToastTitle>{title}</ToastTitle>}
 14 |               {description && <ToastDescription>{description}</ToastDescription>}
 15 |             </div>
 16 |             {action}
 17 |             <ToastClose />
 18 |           </Toast>
 19 |         );
 20 |       })}
 21 |       <ToastViewport />
 22 |     </ToastProvider>
 23 |   );
 24 | }
 25 |
```

File: src/components/ui/toggle-group.tsx

Lines: 50

```
  1 | import * as React from "react";
  2 | import * as ToggleGroupPrimitive from "@radix-ui/react-toggle-group";
  3 | import { type VariantProps } from "class-variance-authority";
  4 |
  5 | import { cn } from "@/lib/utils";
  6 | import { toggleVariants } from "@/components/ui/toggle";
  7 |
  8 | const ToggleGroupContext = React.createContext<VariantProps<typeof toggleVariants>>({
  9 |   size: "default",
 10 |   variant: "default",
 11 | });
 12 |
 13 | const ToggleGroup = React.forwardRef<
 14 |   React.ElementRef<typeof ToggleGroupPrimitive.Root>,
 15 |   React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Root> & VariantProps<typeof
toggleVariants>
 16 | >(({ className, variant, size, children, ...props }, ref) => (
 17 |   <ToggleGroupPrimitive.Root ref={ref} className={cn("flex items-center justify-center gap-1",
className)} {...props}>
 18 |     <ToggleGroupContext.Provider value={{ variant, size }}>{children}</
ToggleGroupContext.Provider>
 19 |   </ToggleGroupPrimitive.Root>
 20 | ));
 21 |
 22 | ToggleGroup.displayName = ToggleGroupPrimitive.Root.displayName;
 23 |
 24 | const ToggleGroupItem = React.forwardRef<
 25 |   React.ElementRef<typeof ToggleGroupPrimitive.Item>,
 26 |   React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Item> & VariantProps<typeof
toggleVariants>
 27 | >(({ className, children, variant, size, ...props }, ref) => {
 28 |   const context = React.useContext(ToggleGroupContext);
 29 |
 30 |   return (
 31 |     <ToggleGroupPrimitive.Item
 32 |       ref={ref}
 33 |       className={cn(
 34 |         toggleVariants({
 35 |           variant: context.variant || variant,
 36 |           size: context.size || size,
 37 |         }),
 38 |         className,
 39 |       )}
 40 |       {...props}
 41 |     >
 42 |       {children}
 43 |     </ToggleGroupPrimitive.Item>
 44 |   );
 45 | });
 46 |
 47 | ToggleGroupItem.displayName = ToggleGroupPrimitive.Item.displayName;
 48 |
 49 | export { ToggleGroup, ToggleGroupItem };
 50 |
```

File: src/components/ui/toggle.tsx

Lines: 38

```
   1 | import * as React from "react";
   2 | import * as TogglePrimitive from "@radix-ui/react-toggle";
   3 | import { cva, type VariantProps } from "class-variance-authority";
   4 |
   5 | import { cn } from "@/lib/utils";
   6 |
   7 | const toggleVariants = cva(
   8 |   "inline-flex items-center justify-center rounded-md text-sm font-medium ring-offset-background
transition-colors hover...
   9 |     {
  10 |       variants: {
  11 |         variant: {
  12 |           default: "bg-transparent",
  13 |           outline: "border border-input bg-transparent hover:bg-accent hover:text-accent-
foreground", },
  14 |         size: {
  15 |           default: "h-10 px-3",
  16 |           sm: "h-9 px-2.5",
  17 |           lg: "h-11 px-5",
  18 |         },
  19 |       },
  20 |       defaultVariants: {
  21 |         variant: "default",
  22 |         size: "default",
  23 |       },
  24 |     },
  25 |   },
  26 | );
  27 |
  28 | const Toggle = React.forwardRef<
  29 |   React.ElementRef<typeof TogglePrimitive.Root>,
  30 |   React.ComponentPropsWithoutRef<typeof TogglePrimitive.Root> & VariantProps<typeof
toggleVariants>
  31 | >(({ className, variant, size, ...props }, ref) => (
  32 |   <TogglePrimitive.Root ref={ref} className={cn(toggleVariants({ variant, size, className }))}
{...props} />
  33 | ));
  34 |
  35 | Toggle.displayName = TogglePrimitive.Root.displayName;
  36 |
  37 | export { Toggle, toggleVariants };
  38 |
```

File: src/components/ui/tooltip.tsx

Lines: 29

```
  1 | import * as React from "react";
  2 | import * as TooltipPrimitive from "@radix-ui/react-tooltip";
  3 |
  4 | import { cn } from "@/lib/utils";
  5 |
  6 | const TooltipProvider = TooltipPrimitive.Provider;
  7 |
  8 | const Tooltip = TooltipPrimitive.Root;
  9 |
 10 | const TooltipTrigger = TooltipPrimitive.Trigger;
 11 |
 12 | const TooltipContent = React.forwardRef<
 13 |   React.ElementRef<typeof TooltipPrimitive.Content>,
 14 |   React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content>
 15 | >(({ className, sideOffset = 4, ...props }, ref) => (
 16 |   <TooltipPrimitive.Content
 17 |     ref={ref}
 18 |     sideOffset={sideOffset}
 19 |     className={cn(
 20 |       "z-50 overflow-hidden rounded-md border bg-popover px-3 py-1.5 text-sm text-popover-
foreground shadow-md animate-i...
 21 |       className,
 22 |     )}
 23 |     {...props}
 24 |   />
 25 | ));
 26 | TooltipContent.displayName = TooltipPrimitive.Content.displayName;
 27 |
 28 | export { Tooltip, TooltipTrigger, TooltipContent, TooltipProvider };
 29 |
```

File: src/components/ui/use-toast.ts

Lines: 4

```
1 | import { useToast, toast } from "@/hooks/use-toast";
2 |
3 | export { useToast, toast };
4 |
```

File: src/contexts/NotificationContext.tsx

Lines: 37

```
  1 | import { createContext, useContext, ReactNode } from "react";
  2 | import { useNotifications, Notification } from "@/components/NotificationSystem";
  3 | import { useAuth } from "@/hooks/useAuth";
  4 |
  5 | interface NotificationContextType {
  6 |   notifications: Notification[];
  7 |   addNotification: (notification: Omit<Notification, "id" | "timestamp" | "read">) => void;
  8 |   dismissNotification: (id: string) => void;
  9 |   markAllAsRead: () => void;
 10 |   handleNotificationTap: (notification: Notification) => void;
 11 |   unreadCount: number;
 12 |   unreadMessageCount: number;
 13 |   unreadFriendRequestCount: number;
 14 |   unreadPokeCount: number;
 15 | }
 16 |
 17 | const NotificationContext = createContext<NotificationContextType | undefined>(undefined);
 18 |
 19 | export const NotificationProvider = ({ children }: { children: ReactNode }) => {
 20 |   const { user } = useAuth();
 21 |   const notificationState = useNotifications(user?.uid || null);
 22 |
 23 |   return (
 24 |     <NotificationContext.Provider value={notificationState}>
 25 |       {children}
 26 |     </NotificationContext.Provider>
 27 |   );
 28 | };
 29 |
 30 | export const useNotificationContext = () => {
 31 |   const context = useContext(NotificationContext);
 32 |   if (context === undefined) {
 33 |     throw new Error("useNotificationContext must be used within a NotificationProvider");
 34 |   }
 35 |   return context;
 36 | };
 37 |
```

File: src/contexts/UserContext.tsx

Lines: 222

```
  1 | import { createContext, useContext, ReactNode, useState, useEffect } from "react";
  2 | import { useAuth } from "@/hooks/useAuth";
  3 | import { listenToUserWorkouts } from "@/services/workoutService";
  4 |
  5 | export type Activity = "running" | "cycling" | "walking";
  6 |
  7 | export interface NearbyUser {
  8 |   id: number;
  9 |   name: string;
 10 |   avatar: string;
 11 |   activity: string;
 12 |   distance: string;
 13 | }
 14 |
 15 | export interface WorkoutHistory {
 16 |   id: string;
 17 |   activity: Activity;
 18 |   date: Date;
 19 |   duration: number; // seconds
 20 |   distance: number; // km
 21 |   avgSpeed: number; // km/h
 22 |   nearbyUsers?: NearbyUser[]; // Users who were nearby during workout
 23 |   location?: string; // Location where workout happened
 24 | }
 25 |
 26 | export type FitnessLevel = "beginner" | "intermediate" | "pro";
 27 | export type RadiusPreference = "nearby" | "normal" | "wide";
 28 |
 29 | export interface VisibilitySettings {
 30 |   visibleToAllLevels: boolean;
 31 |   allowedLevels: FitnessLevel[];
 32 | }
 33 |
 34 | interface UserProfile {
 35 |   username: string;
 36 |   activities: Activity[];
 37 |   gender?: string;
 38 |   photos?: string[]; // URLs or base64 encoded images
 39 |   useMetric?: boolean; // true for km/km/h, false for mi/mph
 40 |   workoutHistory?: WorkoutHistory[];
 41 |   bio?: string; // User biography
 42 |   friends?: number[]; // Array of friend user IDs
 43 |   friendRequests?: { from: number; date: Date }[]; // Pending incoming requests
 44 |   sentRequests?: number[]; // Pending outgoing requests
 45 |   // Matching preferences
 46 |   fitnessLevel?: FitnessLevel;
 47 |   pace?: number; // min/km for running/walking, km/h for cycling
 48 |   visibility?: VisibilitySettings;
 49 |   radiusPreference?: RadiusPreference;
 50 | }
 51 |
 52 | interface UserContextType {
 53 |   userProfile: UserProfile | null;
 54 |   setUserProfile: (profile: UserProfile) => void;
 55 |   hasActivity: (activity: Activity) => boolean;
 56 |   useMetric: boolean;
 57 |   setUseMetric: (useMetric: boolean) => void;
 58 |   addWorkout: (workout: Omit<WorkoutHistory, "id">) => void;
 59 |   workoutHistory: WorkoutHistory[];
 60 | }
 61 |
 62 | const UserContext = createContext<UserContextType | undefined>(undefined);
 63 |
 64 | export const UserProvider = ({ children }: { children: ReactNode }) => {
 65 |   const [userProfile, setUserProfileState] = useState<UserProfile | null>(null);
 66 |   const { user } = useAuth();
```

```
 67 |
 68 |   // Clear profile when user logs out or changes
 69 |   useEffect(() => {
 70 |     if (!user?.uid) {
 71 |       // User logged out - clear profile and localStorage
 72 |       setUserProfileState(null);
 73 |       localStorage.removeItem("userProfile");
 74 |       localStorage.removeItem("userProfileUserId");
 75 |       return;
 76 |     }
 77 |   }, [user?.uid]);
 78 |
 79 |   // Load from localStorage on mount - only if it belongs to current user
 80 |   useEffect(() => {
 81 |     if (!user?.uid) return; // Don't load if no user is logged in
 82 |
 83 |     const storedUserId = localStorage.getItem("userProfileUserId");
 84 |     const stored = localStorage.getItem("userProfile");
 85 |
 86 |     // Only load if localStorage belongs to current user
 87 |     if (stored && storedUserId === user.uid) {
 88 |       try {
 89 |         const parsed = JSON.parse(stored);
 90 |         // Convert date strings back to Date objects
 91 |         if (parsed.workoutHistory) {
 92 |           parsed.workoutHistory = parsed.workoutHistory.map((w: any) => ({
 93 |             ...w,
 94 |             date: new Date(w.date),
 95 |           }));
 96 |         }
 97 |         setUserProfileState(parsed);
 98 |       } catch (e) {
 99 |         console.error("Failed to parse user profile:", e);
100 |         // Clear corrupted data
101 |         localStorage.removeItem("userProfile");
102 |         localStorage.removeItem("userProfileUserId");
103 |       }
104 |     } else if (stored && storedUserId !== user.uid) {
105 |       // Different user's data - clear it
106 |       console.log("& þ  Clearing localStorage from different user");
107 |       localStorage.removeItem("userProfile");
108 |       localStorage.removeItem("userProfileUserId");
109 |     }
110 |   }, [user?.uid]);
111 |
112 |   // Load workouts from Firebase when user is authenticated
113 |   useEffect(() => {
114 |     if (!user?.uid) return;
115 |
116 |     console.log("Ø=Üå Loading workouts from Firebase for user:", user.uid);
117 |
118 |     const unsubscribe = listenToUserWorkouts(user.uid, (workouts) => {
119 |       console.log("'  Loaded workouts from Firebase:", workouts.length);
120 |
121 |       // Update user profile with workouts from Firebase
122 |       setUserProfileState((prev) => {
123 |         // If profile doesn't exist, create a minimal one with workouts
124 |         if (!prev) {
125 |           const minimalProfile: UserProfile = {
126 |             username: "",
127 |             activities: [],
128 |             workoutHistory: workouts,
129 |             useMetric: true,
130 |           };
131 |           // Save to localStorage with user ID
132 |           localStorage.setItem("userProfile", JSON.stringify(minimalProfile));
133 |           localStorage.setItem("userProfileUserId", user.uid);
134 |           return minimalProfile;
135 |         }
136 |
137 |         // Merge Firebase workouts with existing profile
```

```
138 |         // Firebase workouts take precedence
139 |         const updated = {
140 |           ...prev,
141 |           workoutHistory: workouts,
142 |         };
143 |         // Save to localStorage with user ID
144 |         localStorage.setItem("userProfile", JSON.stringify(updated));
145 |         localStorage.setItem("userProfileUserId", user.uid);
146 |         return updated;
147 |       });
148 |     });
149 |
150 |     return () => {
151 |       unsubscribe();
152 |     };
153 |   }, [user?.uid]);
154 |
155 |   const setUserProfile = (profile: UserProfile) => {
156 |     setUserProfileState(profile);
157 |     if (user?.uid) {
158 |       localStorage.setItem("userProfile", JSON.stringify(profile));
159 |       localStorage.setItem("userProfileUserId", user.uid);
160 |     }
161 |   };
162 |
163 |   const hasActivity = (activity: Activity): boolean => {
164 |     return userProfile?.activities.includes(activity) ?? false;
165 |   };
166 |
167 |   const useMetric = userProfile?.useMetric ?? true;
168 |
169 |   const setUseMetric = (metric: boolean) => {
170 |     if (userProfile) {
171 |       const updated = { ...userProfile, useMetric: metric };
172 |       setUserProfile(updated);
173 |     }
174 |   };
175 |
176 |   const addWorkout = (workout: Omit<WorkoutHistory, "id">) => {
177 |     // Create a minimal profile if it doesn't exist
178 |     const currentProfile = userProfile || {
179 |       username: "",
180 |       activities: [],
181 |       workoutHistory: [],
182 |       useMetric: true,
183 |     };
184 |
185 |     const newWorkout: WorkoutHistory = {
186 |       ...workout,
187 |       id: Date.now().toString(),
188 |     };
189 |     const updated = {
190 |       ...currentProfile,
191 |       workoutHistory: [...(currentProfile.workoutHistory || []), newWorkout],
192 |     };
193 |     setUserProfile(updated);
194 |   };
195 |
196 |   const workoutHistory = userProfile?.workoutHistory || [];
197 |
198 |   return (
199 |     <UserContext.Provider
200 |       value={{
201 |         userProfile,
202 |         setUserProfile,
203 |         hasActivity,
204 |         useMetric,
205 |         setUseMetric,
206 |         addWorkout,
207 |         workoutHistory,
208 |       }}
```

```
209 |     >
210 |       {children}
211 |     </UserContext.Provider>
212 |   );
213 | };
214 |
215 | export const useUser = () => {
216 |   const context = useContext(UserContext);
217 |   if (context === undefined) {
218 |     throw new Error("useUser must be used within a UserProvider");
219 |   }
220 |   return context;
221 | };
222 |
```

File: src/hooks/use-mobile.tsx

Lines: 20

```
  1 | import * as React from "react";
  2 |
  3 | const MOBILE_BREAKPOINT = 768;
  4 |
  5 | export function useIsMobile() {
  6 |   const [isMobile, setIsMobile] = React.useState<boolean | undefined>(undefined);
  7 |
  8 |   React.useEffect(() => {
  9 |     const mql = window.matchMedia(`(max-width: ${MOBILE_BREAKPOINT - 1}px)`);
 10 |     const onChange = () => {
 11 |       setIsMobile(window.innerWidth < MOBILE_BREAKPOINT);
 12 |     };
 13 |     mql.addEventListener("change", onChange);
 14 |     setIsMobile(window.innerWidth < MOBILE_BREAKPOINT);
 15 |     return () => mql.removeEventListener("change", onChange);
 16 |   }, []);
 17 |
 18 |   return !!isMobile;
 19 | }
 20 |
```

File: src/hooks/use-toast.ts

Lines: 187

```
  1 | import * as React from "react";
  2 |
  3 | import type { ToastActionElement, ToastProps } from "@/components/ui/toast";
  4 |
  5 | const TOAST_LIMIT = 1;
  6 | const TOAST_REMOVE_DELAY = 1000000;
  7 |
  8 | type ToasterToast = ToastProps & {
  9 |   id: string;
 10 |   title?: React.ReactNode;
 11 |   description?: React.ReactNode;
 12 |   action?: ToastActionElement;
 13 | };
 14 |
 15 | const actionTypes = {
 16 |   ADD_TOAST: "ADD_TOAST",
 17 |   UPDATE_TOAST: "UPDATE_TOAST",
 18 |   DISMISS_TOAST: "DISMISS_TOAST",
 19 |   REMOVE_TOAST: "REMOVE_TOAST",
 20 | } as const;
 21 |
 22 | let count = 0;
 23 |
 24 | function genId() {
 25 |   count = (count + 1) % Number.MAX_SAFE_INTEGER;
 26 |   return count.toString();
 27 | }
 28 |
 29 | type ActionType = typeof actionTypes;
 30 |
 31 | type Action =
 32 |   | {
 33 |       type: ActionType["ADD_TOAST"];
 34 |       toast: ToasterToast;
 35 |     }
 36 |   | {
 37 |       type: ActionType["UPDATE_TOAST"];
 38 |       toast: Partial<ToasterToast>;
 39 |     }
 40 |   | {
 41 |       type: ActionType["DISMISS_TOAST"];
 42 |       toastId?: ToasterToast["id"];
 43 |     }
 44 |   | {
 45 |       type: ActionType["REMOVE_TOAST"];
 46 |       toastId?: ToasterToast["id"];
 47 |     };
 48 |
 49 | interface State {
 50 |   toasts: ToasterToast[];
 51 | }
 52 |
 53 | const toastTimeouts = new Map<string, ReturnType<typeof setTimeout>>();
 54 |
 55 | const addToRemoveQueue = (toastId: string) => {
 56 |   if (toastTimeouts.has(toastId)) {
 57 |     return;
 58 |   }
 59 |
 60 |   const timeout = setTimeout(() => {
 61 |     toastTimeouts.delete(toastId);
 62 |     dispatch({
 63 |       type: "REMOVE_TOAST",
 64 |       toastId: toastId,
 65 |     });
 66 |   }, TOAST_REMOVE_DELAY);
```

```
  67 |
  68 |   toastTimeouts.set(toastId, timeout);
  69 | };
  70 |
  71 | export const reducer = (state: State, action: Action): State => {
  72 |   switch (action.type) {
  73 |     case "ADD_TOAST":
  74 |       return {
  75 |         ...state,
  76 |         toasts: [action.toast, ...state.toasts].slice(0, TOAST_LIMIT),
  77 |       };
  78 |
  79 |     case "UPDATE_TOAST":
  80 |       return {
  81 |         ...state,
  82 |         toasts: state.toasts.map((t) => (t.id === action.toast.id ? { ...t, ...action.toast } :
t) ;
  83 |       };
  84 |
  85 |     case "DISMISS_TOAST": {
  86 |       const { toastId } = action;
  87 |
  88 |       // ! Side effects ! - This could be extracted into a dismissToast() action,
  89 |       // but I'll keep it here for simplicity
  90 |       if (toastId) {
  91 |         addToRemoveQueue(toastId);
  92 |       } else {
  93 |         state.toasts.forEach((toast) => {
  94 |           addToRemoveQueue(toast.id);
  95 |         });
  96 |       }
  97 |
  98 |       return {
  99 |         ...state,
 100 |         toasts: state.toasts.map((t) =>
 101 |           t.id === toastId || toastId === undefined
 102 |             ? {
 103 |                 ...t,
 104 |                 open: false,
 105 |               }
 106 |             : t,
 107 |         ),
 108 |       };
 109 |     }
 110 |     case "REMOVE_TOAST":
 111 |       if (action.toastId === undefined) {
 112 |         return {
 113 |           ...state,
 114 |           toasts: [],
 115 |         };
 116 |       }
 117 |       return {
 118 |         ...state,
 119 |         toasts: state.toasts.filter((t) => t.id !== action.toastId),
 120 |       };
 121 |   }
 122 | };
 123 |
 124 | const listeners: Array<(state: State) => void> = [];
 125 |
 126 | let memoryState: State = { toasts: [] };
 127 |
 128 | function dispatch(action: Action) {
 129 |   memoryState = reducer(memoryState, action);
 130 |   listeners.forEach((listener) => {
 131 |     listener(memoryState);
 132 |   });
 133 | }
 134 |
 135 | type Toast = Omit<ToasterToast, "id">;
 136 |
 137 | function toast({ ...props }: Toast) {
```

```
138 |   const id = genId();
139 |
140 |   const update = (props: ToasterToast) =>
141 |     dispatch({
142 |       type: "UPDATE_TOAST",
143 |       toast: { ...props, id },
144 |     });
145 |   const dismiss = () => dispatch({ type: "DISMISS_TOAST", toastId: id });
146 |
147 |   dispatch({
148 |     type: "ADD_TOAST",
149 |     toast: {
150 |       ...props,
151 |       id,
152 |       open: true,
153 |       onOpenChange: (open) => {
154 |         if (!open) dismiss();
155 |       },
156 |     },
157 |   });
158 |
159 |   return {
160 |     id: id,
161 |     dismiss,
162 |     update,
163 |   };
164 | }
165 |
166 | function useToast() {
167 |   const [state, setState] = React.useState<State>(memoryState);
168 |
169 |   React.useEffect(() => {
170 |     listeners.push(setState);
171 |     return () => {
172 |       const index = listeners.indexOf(setState);
173 |       if (index > -1) {
174 |         listeners.splice(index, 1);
175 |       }
176 |     };
177 |   }, [state]);
178 |
179 |   return {
180 |     ...state,
181 |     toast,
182 |     dismiss: (toastId?: string) => dispatch({ type: "DISMISS_TOAST", toastId }),
183 |   };
184 | }
185 |
186 | export { useToast, toast };
187 |
```

File: src/hooks/useAdmin.ts

Lines: 50

```
// Custom hook for admin status
import { useState, useEffect } from "react";
import { useAuth } from "./useAuth";
import { checkAdminStatus } from "@/services/adminService";

/**
 * Custom hook to check if current user is admin
 * @returns {Object} { isAdmin, loading, error }
 */
export const useAdmin = () => {
  const { user, loading: authLoading } = useAuth();
  const [isAdmin, setIsAdmin] = useState(false);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const checkAdmin = async () => {
      if (authLoading) {
        setLoading(true);
        return;
      }

      if (!user || !user.email) {
        setIsAdmin(false);
        setLoading(false);
        setError(null);
        return;
      }

      try {
        setLoading(true);
        const adminStatus = await checkAdminStatus(user.email);
        setIsAdmin(adminStatus);
        setError(null);
      } catch (err: any) {
        console.error("Error checking admin status:", err);
        setIsAdmin(false);
        setError(err.message || "Failed to check admin status");
      } finally {
        setLoading(false);
      }
    };

    checkAdmin();
  }, [user, authLoading]);

  return { isAdmin, loading: loading || authLoading, error };
};

```

File: src/hooks/useAuth.ts

Lines: 44

```
// Custom hook for authentication state
import { useState, useEffect } from "react";
import { onAuthStateChange } from "../services/authService";
import { User } from "firebase/auth";

export interface AuthUser {
  uid: string;
  displayName: string | null;
  email: string | null;
  photoURL: string | null;
}

/**
 * Custom hook to manage authentication state
 * @returns {Object} { user, loading, error }
 */
export const useAuth = () => {
  const [user, setUser] = useState<AuthUser | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const unsubscribe = onAuthStateChange((firebaseUser: User | null) => {
      if (firebaseUser) {
        setUser({
          uid: firebaseUser.uid,
          displayName: firebaseUser.displayName,
          email: firebaseUser.email,
          photoURL: firebaseUser.photoURL
        });
      } else {
        setUser(null);
      }
      setLoading(false);
      setError(null);
    });

    return () => unsubscribe();
  }, []);

  return { user, loading, error };
};

```

File: src/hooks/useLocation.ts

Lines: 387

```
// Custom hook for GPS location tracking
// Works in web browsers and native mobile apps (via Capacitor)
import { useState, useEffect, useRef, useCallback } from "react";
import { Geolocation } from "@capacitor/geolocation";
import { updateUserLocation } from "../services/locationService";
import { isNativePlatform } from "../utils/platform";

export interface Location {
  lat: number;
  lng: number;
}

/**
 * Custom hook to track user's GPS location
 * @param {string | null} userId - Current user ID
 * @param {boolean} isTracking - Whether tracking is active
 * @param {boolean} visible - Whether user is visible on map
 * @returns {Object} { location, error, isGettingLocation }
 */
export const useLocation = (
  userId: string | null,
  isTracking: boolean,
  visible: boolean
) => {
  const [location, setLocation] = useState<Location | null>(null);
  const [error, setError] = useState<string | null>(null);
  const [isGettingLocation, setIsGettingLocation] = useState(false);
  const browserWatchIdRef = useRef<number | null>(null);
  const nativeWatchIdRef = useRef<string | null>(null);
  const isNative = isNativePlatform();
  // Throttle Firebase updates to every 5-10 seconds (use 7.5 seconds as middle ground)
  const lastFirebaseUpdateRef = useRef<number>(0);
  const FIREBASE_UPDATE_INTERVAL = 7500; // 7.5 seconds (middle of 5-10 second range)

  const stopTracking = useCallback(async () => {
    // Stop browser geolocation
    if (browserWatchIdRef.current !== null && navigator.geolocation) {
      navigator.geolocation.clearWatch(browserWatchIdRef.current);
      browserWatchIdRef.current = null;
    }

    // Stop Capacitor geolocation
    if (nativeWatchIdRef.current !== null) {
      try {
        await Geolocation.clearWatch({ id: nativeWatchIdRef.current });
        nativeWatchIdRef.current = null;
      } catch (err) {
        console.error("Error clearing native watch:", err);
      }
    }
  }, []);

  useEffect(() => {
    // Always track if userId exists and isTracking is true
    if (!userId || !isTracking) {
      // Stop tracking if no user or tracking disabled
      stopTracking();
      setIsGettingLocation(false);
      return;
    }

    /**
     * Start native mobile tracking using Capacitor Geolocation API
     * This provides better accuracy and battery efficiency on mobile devices
     */
    const startNativeTracking = async () => {
```

```
  67 |       setIsGettingLocation(true);
  68 |
  69 |       try {
  70 |         // Request permissions first
  71 |         const permissionStatus = await Geolocation.requestPermissions();
  72 |
  73 |         if (permissionStatus.location !== 'granted') {
  74 |           setError("Location permission denied. Please allow location access in your device
settings.");
  75 |           setIsGettingLocation(false);
  76 |           return;
  77 |         }
  78 |
  79 |         // Get current position first
  80 |         const currentPosition = await Geolocation.getCurrentPosition({
  81 |           enableHighAccuracy: true,
  82 |           timeout: 10000
  83 |         });
  84 |
  85 |         const { latitude, longitude } = currentPosition.coords;
  86 |         const newLocation = { lat: latitude, lng: longitude };
  87 |
  88 |         setLocation(newLocation);
  89 |         setError(null);
  90 |         setIsGettingLocation(false);
  91 |
  92 |         // Always update Firebase on initial position (no throttling for first update)
  93 |         lastFirebaseUpdateRef.current = Date.now();
  94 |         await updateUserLocation(userId, latitude, longitude, visible);
  95 |
  96 |         // Watch position for continuous updates
  97 |         const watchId = await Geolocation.watchPosition(
  98 |           {
  99 |             enableHighAccuracy: true,
 100 |             timeout: 10000
 101 |           },
 102 |           async (position, err) => {
 103 |             if (err) {
 104 |               let errorMessage = "Location error";
 105 |               if (err.message) {
 106 |                 errorMessage = err.message;
 107 |               } else if (err.code === 'PERMISSION_DENIED') {
 108 |                 errorMessage = "Location permission denied. Please allow location access in your
device settings.";
 109 |               } else if (err.code === 'POSITION_UNAVAILABLE') {
 110 |                 errorMessage = "Location information unavailable.";
 111 |               } else if (err.code === 'TIMEOUT') {
 112 |                 errorMessage = "Location request timed out.";
 113 |               }
 114 |               setError(errorMessage);
 115 |               setIsGettingLocation(false);
 116 |               return;
 117 |             }
 118 |
 119 |             if (position) {
 120 |               const { latitude, longitude } = position.coords;
 121 |               const newLocation = { lat: latitude, lng: longitude };
 122 |
 123 |               setLocation(newLocation);
 124 |               setError(null);
 125 |
 126 |               // Throttle Firebase updates to every 5-10 seconds
 127 |               const now = Date.now();
 128 |               const timeSinceLastUpdate = now - lastFirebaseUpdateRef.current;
 129 |
 130 |               if (timeSinceLastUpdate >= FIREBASE_UPDATE_INTERVAL) {
 131 |                 lastFirebaseUpdateRef.current = now;
 132 |                 await updateUserLocation(userId, latitude, longitude, visible);
 133 |               }
 134 |             }
 135 |           }
 136 |         );
 137 |
```

```
        nativeWatchIdRef.current = watchId;
      } catch (err: any) {
        let errorMessage = "Location error";
        if (err?.message) {
          errorMessage = err.message;
        } else if (err?.code === 'PERMISSION_DENIED') {
          errorMessage = "Location permission denied. Please allow location access in your device settings.";
        }
        setError(errorMessage);
        setIsGettingLocation(false);
      }
    };

    /**
     * Start browser tracking using Web Geolocation API
     * Used when running in web browsers
     */
    const startBrowserTracking = () => {
      console.log("Ø<ß  Starting browser GPS tracking...");
      setIsGettingLocation(true);

      // Check if we're in a secure context (HTTPS or localhost)
      // Geolocation API requires secure context in modern browsers
      if (typeof window !== 'undefined' && !window.isSecureContext) {
        console.warn("& þ  Not in a secure context (HTTPS). Geolocation may not work.");
        console.warn("& þ  Current protocol:", window.location.protocol);
        console.warn("& þ  To fix: Use HTTPS or localhost for development.");
      }

      // Check if geolocation is supported
      if (!navigator.geolocation) {
        const errorMsg = "Geolocation is not supported by your browser";
        console.error("'L", errorMsg);
        setError(errorMsg);
        setIsGettingLocation(false);
        return;
      }

      // Check permission state if Permissions API is available (Chrome, Edge, etc.)
      // This is done asynchronously but doesn't block the geolocation request
      if ('permissions' in navigator) {
        navigator.permissions.query({ name: 'geolocation' as PermissionName })
          .then((permissionStatus) => {
            console.log("Ø=ÜÍ Permission state:", permissionStatus.state);

            if (permissionStatus.state === 'denied') {
              const errorMsg = "Location permission denied. Please allow location access in your browser settings.";
              console.error("'L", errorMsg);
              setError(errorMsg);
              setIsGettingLocation(false);
              return;
            }

            // Listen for permission changes
            permissionStatus.onchange = () => {
              console.log("Ø=ÜÍ Permission state changed to:", permissionStatus.state);
              if (permissionStatus.state === 'granted') {
                // Permission was granted, retry getting position if we don't have one yet
                console.log("Ø=ÜÍ Permission granted, location should start working now...");
                // The watchPosition will automatically start working once permission is granted
              } else if (permissionStatus.state === 'denied') {
                setError("Location permission denied. Please allow location access in your browser settings.");
                setIsGettingLocation(false);
              }
            };
          })
          .catch((err) => {
            // Permissions API might not be fully supported, continue anyway
            console.log("& þ  Permissions API not fully supported, continuing with geolocation request...", err);
          });
      }
```

```
      console.log("Ø=ÜÍ Requesting current position (will prompt for permission if needed)...");

      // Track which accuracy setting worked so we can use it for watchPosition
      let workingAccuracy = true; // Start with high accuracy

      // Try to get position with retry logic
      const tryGetPosition = (attempt: number = 1, useHighAccuracy: boolean = true) => {
        const maxAttempts = 2;
        const currentTimeout = useHighAccuracy ? 15000 : 30000; // 30s for low accuracy (network geolocation can be slow)
        const currentMaxAge = useHighAccuracy ? 0 : 120000; // Accept cached location up to 2 minutes for low accuracy

        console.log(`Ø=ÜÍ Attempting to get position (attempt ${attempt}/${maxAttempts}, highAccuracy: ${useHighAccuracy},...

        navigator.geolocation.getCurrentPosition(
        async (position) => {
          const { latitude, longitude, accuracy } = position.coords;
          const newLocation = { lat: latitude, lng: longitude };

          console.log("'  GPS position obtained:", newLocation);
          console.log(`Ø=ÜÍ Position accuracy: ${accuracy?.toFixed(1) || 'unknown'}m, highAccuracy setting: ${useHighAccur...

          // Remember which accuracy setting worked
          workingAccuracy = useHighAccuracy;

          setLocation(newLocation);
          setError(null);
          setIsGettingLocation(false);

          // Always update Firebase on initial position (no throttling for first update)
          lastFirebaseUpdateRef.current = Date.now();
          try {
            await updateUserLocation(userId, latitude, longitude, visible);
            console.log("'  Location updated to Firebase");
          } catch (error) {
            console.error("'L Error updating location to Firebase:", error);
          }

          // Watch position for updates - use the same accuracy setting that worked
          console.log(`Ø=ÜÍ Starting continuous GPS tracking (highAccuracy: ${workingAccuracy})...`);
          browserWatchIdRef.current = navigator.geolocation.watchPosition(
            async (position) => {
              const { latitude, longitude } = position.coords;
              const newLocation = { lat: latitude, lng: longitude };

              setLocation(newLocation);
              setError(null);

              // Throttle Firebase updates to every 5-10 seconds
              const now = Date.now();
              const timeSinceLastUpdate = now - lastFirebaseUpdateRef.current;

              if (timeSinceLastUpdate >= FIREBASE_UPDATE_INTERVAL) {
                lastFirebaseUpdateRef.current = now;
                try {
                  await updateUserLocation(userId, latitude, longitude, visible);
                } catch (error) {
                  console.error("'L Error updating location to Firebase:", error);
                }
              }
            },
            (err) => {
              let errorMessage = "Location error";
              console.error(`'L watchPosition error - code: ${err.code}, message: "${err.message}"`);
              switch (err.code) {
                case err.PERMISSION_DENIED:
                  errorMessage = "Location permission denied. Please allow location access in your browser settings.";
                  console.error("'L GPS Permission denied:", err);
                  break;
                case err.POSITION_UNAVAILABLE:
                  errorMessage = "Location information unavailable. Your location is still being tracked when available....
```

```
                  console.error("'L GPS Position unavailable:", err);
                  // Don't stop tracking - watchPosition will retry automatically
                  return;
                case err.TIMEOUT:
                  errorMessage = "Location request timed out. Retrying...";
                  console.error("'L GPS Request timeout:", err);
                  // Don't stop tracking - watchPosition will retry automatically
                  return;
                default:
                  errorMessage = err.message || "Unknown location error";
                  console.error("'L GPS Error:", err);
              }
              setError(errorMessage);
              setIsGettingLocation(false);
            },
            {
              // Use the same accuracy setting that worked for getCurrentPosition
              enableHighAccuracy: workingAccuracy,
              timeout: workingAccuracy ? 15000 : 30000, // Match the timeout used for
getCurrentPosition
              maximumAge: 10000 // Accept positions up to 10 seconds old
            }
          );
        },
        (err) => {
          let errorMessage = "Location error";
          let shouldRetry = false;

          // Log detailed error information for debugging
          console.error(`'L getCurrentPosition error - code: ${err.code}, message:
"${err.message}"`);
          console.error(`'L Error details - attempt: ${attempt}/${maxAttempts}, highAccuracy:
${useHighAccuracy}`);

          switch (err.code) {
            case err.PERMISSION_DENIED:
              errorMessage = "Location permission denied. Please allow location access in your
browser settings.";
              console.error("'L GPS Permission denied on initial request");
              console.error("Ø=Ü¡ Tip: Check browser URL bar for location icon, or go to browser
Settings > Privacy > Locat...");
              break;
            case err.POSITION_UNAVAILABLE:
              // If high accuracy failed and we haven't retried with low accuracy, try again
              if (useHighAccuracy && attempt < maxAttempts) {
                console.warn(`& þ  GPS Position unavailable with high accuracy (attempt ${attempt}).
Retrying with lower a...");
                console.warn("Ø=Ü¡ This is normal on desktop browsers - they use network-based
location instead of GPS.");
                shouldRetry = true;
                setTimeout(() => tryGetPosition(attempt + 1, false), 1000);
                return; // Don't set error yet, wait for retry
              }
              errorMessage = "GPS signal unavailable. Please check:\n• You're in an area with
GPS coverage\n• Location s...";
              console.error("'L GPS Position unavailable after all attempts");
              console.error("Ø=Ü¡ Possible causes: Browser blocked, no network location available,
or location services di...");
              break;
            case err.TIMEOUT:
              // If timeout and we haven't retried, try with lower accuracy
              if (useHighAccuracy && attempt < maxAttempts) {
                console.warn(`& þ  GPS Request timeout with high accuracy (attempt ${attempt}).
Retrying with lower accuracy...`);
                shouldRetry = true;
                setTimeout(() => tryGetPosition(attempt + 1, false), 1000);
                return; // Don't set error yet, wait for retry
              }
              errorMessage = "Location request timed out. Please try again or check your GPS
signal.";
              console.error("'L GPS Request timeout after all attempts");
              console.error("Ø=Ü¡ Network geolocation can be slow. Try refreshing the page.");
              break;
            default:
              errorMessage = err.message || "Unknown location error";
              console.error("'L GPS Error on initial request:", err);
          }

          if (!shouldRetry) {
            setError(errorMessage);
            setIsGettingLocation(false);
          }
```

```
351 |         },
352 |         {
353 |           enableHighAccuracy: useHighAccuracy,
354 |           timeout: currentTimeout,
355 |           maximumAge: currentMaxAge
356 |         }
357 |       );
358 |       };
359 |
360 |       // Start first attempt
361 |       tryGetPosition(1, true);
362 |     };
363 |
364 |     // Start tracking based on platform
365 |     if (isNative) {
366 |       startNativeTracking();
367 |     } else {
368 |       startBrowserTracking();
369 |     }
370 |
371 |     // Cleanup
372 |     return () => {
373 |       stopTracking();
374 |     };
375 |   }, [userId, isTracking, visible, stopTracking, isNative]);
376 |
377 |   // Ensure tracking stops on unmount
378 |   useEffect(() => {
379 |     return () => {
380 |       stopTracking();
381 |     };
382 |   }, [stopTracking]);
383 |
384 |   return { location, error, isGettingLocation, stopTracking };
385 | };
386 |
387 |
```

File: src/hooks/useMatching.ts

Lines: 156

```
   1 | // Custom hook for matching users using PACE-MATCH algorithm
   2 | import { useState, useEffect, useRef, useMemo } from "react";
   3 | import { listenToAllUsers } from "@/services/locationService";
   4 | import { matchUsers, MatchingUser, MatchResult, Activity, FitnessLevel, RadiusPreference,
VisibilitySettings, SearchFilter
   5 | import { listenToUserProfile } from "@/services/authService";
   6 |
   7 | export interface UseMatchingOptions {
   8 |   currentUserId: string;
   9 |   currentLocation: { lat: number; lng: number } | null;
  10 |   activity: Activity;
  11 |   fitnessLevel?: FitnessLevel;
  12 |   pace?: number;
  13 |   visibility?: VisibilitySettings;
  14 |   searchFilter?: SearchFilter; // Who do I want to find?
  15 |   radiusPreference?: RadiusPreference;
  16 | }
  17 |
  18 | export interface UseMatchingResult {
  19 |   matches: MatchResult[];
  20 |   loading: boolean;
  21 |   error: string | null;
  22 | }
  23 |
  24 | /**
  25 |  * Custom hook to get matched users in real-time
  26 |  */
  27 | export const useMatching = (options: UseMatchingOptions): UseMatchingResult => {
  28 |   const {
  29 |     currentUserId,
  30 |     currentLocation,
  31 |     activity,
  32 |     fitnessLevel = "intermediate",
  33 |     pace,
  34 |     visibility = { visibleToAllLevels: true, allowedLevels: ["beginner", "intermediate",
"pro"] },  searchFilter = "all",
  36 |     radiusPreference = "normal"
  37 |   } = options;
  38 |
  39 |   const [matches, setMatches] = useState<MatchResult[]>([]);
  40 |   const [loading, setLoading] = useState(true);
  41 |   const [error, setError] = useState<string | null>(null);
  42 |   const [userData, setUserData] = useState<any>(null);
  43 |
  44 |   // Extract location coordinates for stable dependency comparison
  45 |   const locationLat = currentLocation?.lat;
  46 |   const locationLng = currentLocation?.lng;
  47 |   const locationKey = useMemo(() =>
  48 |     locationLat && locationLng ? `${locationLat.toFixed(6)},${locationLng.toFixed(6)}` : null,
  49 |     [locationLat, locationLng]
  50 |   );
  51 |
  52 |   // Memoize visibility to prevent infinite loops
  53 |   const stableVisibility = useMemo(() => visibility, [
  54 |     visibility?.visibleToAllLevels,
  55 |     JSON.stringify(visibility?.allowedLevels)
  56 |   ]);
  57 |
  58 |   // Load current user's matching preferences
  59 |   useEffect(() => {
  60 |     if (!currentUserId) {
  61 |       setUserData(null);
  62 |       return;
  63 |     }
  64 |
  65 |     setLoading(true);
  66 |     const unsubscribe = listenToUserProfile(currentUserId, (data) => {
```

```
 67 |       setUserData(data);
 68 |     });
 69 |
 70 |     return () => unsubscribe();
 71 |   }, [currentUserId]);
 72 |
 73 |   // Perform matching when location or user data changes
 74 |   useEffect(() => {
 75 |     if (!currentLocation || !locationLat || !locationLng) {
 76 |       setMatches([]);
 77 |       setLoading(false);
 78 |       return;
 79 |     }
 80 |
 81 |     if (!currentUserId) {
 82 |       setMatches([]);
 83 |       setLoading(false);
 84 |       return;
 85 |     }
 86 |
 87 |     if (!userData) {
 88 |       setLoading(true);
 89 |       return;
 90 |     }
 91 |
 92 |     if (userData?.profileVisible === false) {
 93 |       setMatches([]);
 94 |       setError("Profile discovery is disabled");
 95 |       setLoading(false);
 96 |       return;
 97 |     }
 98 |
 99 |     setError(null);
100 |
101 |     // Use user data from Firebase or fallback to provided options
102 |     const effectiveFitnessLevel = userData?.fitnessLevel || fitnessLevel;
103 |     const effectivePace = userData?.pace || pace || null;
104 |     const effectiveVisibility = userData?.visibility || stableVisibility;
105 |     const effectiveSearchFilter = userData?.searchFilter || searchFilter;
106 |     const effectiveRadiusPreference = userData?.radiusPreference || radiusPreference;
107 |     const effectiveActivity = userData?.activity || activity;
108 |
109 |     // Create matching user object
110 |     const matchingUser: MatchingUser = {
111 |       uid: currentUserId,
112 |       location: currentLocation,
113 |       activity: effectiveActivity,
114 |       fitnessLevel: effectiveFitnessLevel,
115 |       pace: effectivePace || 0,
116 |       visibility: effectiveVisibility,
117 |       searchFilter: effectiveSearchFilter,
118 |       radiusPreference: effectiveRadiusPreference,
119 |       profileVisible: userData?.profileVisible !== false
120 |     };
121 |
122 |     const unsubscribe = listenToAllUsers((allUsers) => {
123 |       try {
124 |         // Count nearby users for density calculation
125 |         const nearbyCount = Object.keys(allUsers).length;
126 |
127 |         // Perform matching
128 |         const results = matchUsers(matchingUser, allUsers, nearbyCount);
129 |         setMatches(results);
130 |         setError(null);
131 |       } catch (err: any) {
132 |         console.error("Error matching users:", err);
133 |         setError(err.message || "Failed to match users");
134 |         setMatches([]);
135 |       } finally {
136 |         setLoading(false);
137 |       }
```

```
138 |     });
139 |
140 |     return () => unsubscribe();
141 |   }, [
142 |     locationKey,
143 |     currentUserId,
144 |     activity,
145 |     fitnessLevel,
146 |     pace,
147 |     stableVisibility,
148 |     searchFilter,
149 |     radiusPreference,
150 |     userData
151 |   ]);
152 |
153 |   return { matches, loading, error };
154 | };
155 |
156 |
```

File: src/hooks/useMovementDetection.ts

Lines: 218

```
  1 | // Movement detection hook for workout inactivity detection
  2 | // Uses GPS position history to detect if user is stationary
  3 | import { useEffect, useRef, useCallback } from "react";
  4 |
  5 | export interface Location {
  6 |   lat: number;
  7 |   lng: number;
  8 | }
  9 |
 10 | interface MovementDetectionOptions {
 11 |   /** Location updates to monitor */
 12 |   location: Location | null;
 13 |   /** Whether workout is currently active */
 14 |   isActive: boolean;
 15 |   /** Whether workout is paused */
 16 |   isPaused: boolean;
 17 |   /** Callback when stationary state detected */
 18 |   onStationary: () => void;
 19 |   /** Callback when movement detected after being stationary */
 20 |   onMovementDetected?: () => void;
 21 |   /** Distance threshold in meters (default: 10m) */
 22 |   distanceThreshold?: number;
 23 |   /** Detection window in minutes (default: 5 minutes) */
 24 |   detectionWindowMinutes?: number;
 25 | }
 26 |
 27 | interface LocationWithTimestamp extends Location {
 28 |   timestamp: number;
 29 | }
 30 |
 31 | /**
 32 |  * Calculate distance between two coordinates using Haversine formula
 33 |  * @returns distance in meters
 34 |  */
 35 | const calculateDistance = (loc1: Location, loc2: Location): number => {
 36 |   const R = 6371e3; // Earth's radius in meters
 37 |   const <c  Ò †Æö3 æÆ B ¢ Ö F‚å ’’ ò  ƒ °
 38 |   const <c" Ò †Æö3"æÆ B ¢ Ö F‚å ’’ ò  ƒ °
 39 |   const 9CÆ = ((loc2.lat - loc1.lat) * Math.PI) / 180;
 40 |   const 9C» = ((loc2.lng - loc1.lng) * Math.PI) / 180;
 41 |
 42 |   const a =
 43 |     Math.sin(9CÆ / 2) * Math.sin(9CÆ / 2) +
 44 |     Math.cos(<c ’ ¢ Ö F‚æ6÷2ƒÆ2) * Math.sin(9C» / 2) * Math.sin(9C» / 2);
 45 |   const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
 46 |
 47 |   return R * c; // Distance in meters
 48 | };
 49 |
 50 | /**
 51 |  * Hook to detect if user is stationary during workout
 52 |  * Monitors GPS positions over a time window and detects if movement is below threshold
 53 |  */
 54 | export const useMovementDetection = ({
 55 |   location,
 56 |   isActive,
 57 |   isPaused,
 58 |   onStationary,
 59 |   onMovementDetected,
 60 |   distanceThreshold = 10, // 10 meters
 61 |   detectionWindowMinutes = 5, // 5 minutes
 62 | }: MovementDetectionOptions) => {
 63 |   const locationHistoryRef = useRef<LocationWithTimestamp[]>([]);
 64 |   const isStationaryRef = useRef(false);
 65 |   const lastStationaryCallRef = useRef<number>(0);
 66 |   const checkIntervalRef = useRef<NodeJS.Timeout | null>(null);
```

```
  67 |   const workoutStartTimeRef = useRef<number | null>(null);
  68 |
  69 |   // Cleanup function
  70 |   const cleanup = useCallback(() => {
  71 |     if (checkIntervalRef.current) {
  72 |       clearInterval(checkIntervalRef.current);
  73 |       checkIntervalRef.current = null;
  74 |     }
  75 |   }, []);
  76 |
  77 |   // Check if user has moved significantly in the detection window
  78 |   const checkMovement = useCallback((currentPaused: boolean) => {
  79 |     const now = Date.now();
  80 |     const windowMs = detectionWindowMinutes * 60 * 1000;
  81 |     const cutoffTime = now - windowMs;
  82 |
  83 |     // Don't trigger stationary warning until full detection window has elapsed since workout
started
  84 |     if (workoutStartTimeRef.current) {
  85 |       const timeSinceStart = now - workoutStartTimeRef.current;
  86 |       if (timeSinceStart < windowMs) {
  87 |         // Not enough time has passed since workout started, wait longer
  88 |         return;
  89 |       }
  90 |     }
  91 |
  92 |     // Filter locations within the detection window
  93 |     const recentLocations = locationHistoryRef.current.filter(
  94 |       (loc) => loc.timestamp > cutoffTime
  95 |     );
  96 |
  97 |     // Need at least 2 locations to calculate movement
  98 |     if (recentLocations.length < 2) {
  99 |       // Not enough data yet, wait for more locations
 100 |       return;
 101 |     }
 102 |
 103 |     // Calculate total distance traveled in the window
 104 |     let totalDistance = 0;
 105 |     for (let i = 1; i < recentLocations.length; i++) {
 106 |       totalDistance += calculateDistance(
 107 |         recentLocations[i - 1],
 108 |         recentLocations[i]
 109 |       );
 110 |     }
 111 |
 112 |     // Also calculate straight-line distance from first to last point
 113 |     const straightLineDistance = calculateDistance(
 114 |       recentLocations[0],
 115 |       recentLocations[recentLocations.length - 1]
 116 |     );
 117 |
 118 |     // User is stationary if:
 119 |     // 1. Total distance traveled < threshold (they stayed in a small area)
 120 |     // 2. Straight-line distance < threshold (they didn't move far from starting point)
 121 |     const isCurrentlyStationary =
 122 |       totalDistance < distanceThreshold && straightLineDistance < distanceThreshold;
 123 |
 124 |     // Only trigger callbacks on state changes to avoid spam
 125 |     if (isCurrentlyStationary && !isStationaryRef.current) {
 126 |       // Just became stationary - only trigger if not paused
 127 |       isStationaryRef.current = true;
 128 |       // Only call onStationary when not paused (don't warn if already paused)
 129 |       if (!currentPaused) {
 130 |         // Throttle stationary callback to once per detection window
 131 |         const timeSinceLastCall = now - lastStationaryCallRef.current;
 132 |         if (timeSinceLastCall > windowMs) {
 133 |           onStationary();
 134 |           lastStationaryCallRef.current = now;
 135 |         }
 136 |       }
 137 |     } else if (!isCurrentlyStationary && isStationaryRef.current) {
```

```
138 |       // Movement detected after being stationary
139 |       isStationaryRef.current = false;
140 |       // Always call onMovementDetected when movement detected (for resume functionality)
141 |       if (onMovementDetected) {
142 |         onMovementDetected();
143 |       }
144 |     }
145 |   }, [distanceThreshold, detectionWindowMinutes, onStationary, onMovementDetected]);
146 |
147 |   // Add location to history when it updates
148 |   // Continue tracking even when paused to detect movement for resume
149 |   useEffect(() => {
150 |     if (!location || !isActive) {
151 |       return;
152 |     }
153 |
154 |     // Add new location with timestamp
155 |     locationHistoryRef.current.push({
156 |       ...location,
157 |       timestamp: Date.now(),
158 |     });
159 |
160 |     // Keep only locations from the last (detection window + 1 minute buffer)
161 |     const maxAge = (detectionWindowMinutes + 1) * 60 * 1000;
162 |     const cutoffTime = Date.now() - maxAge;
163 |     locationHistoryRef.current = locationHistoryRef.current.filter(
164 |       (loc) => loc.timestamp > cutoffTime
165 |     );
166 |   }, [location, isActive, detectionWindowMinutes]);
167 |
168 |   // Track workout start time and reset location history when workout starts
169 |   const prevIsActiveRef = useRef<boolean>(false);
170 |   useEffect(() => {
171 |     // Detect transition from inactive to active
172 |     if (isActive && !prevIsActiveRef.current) {
173 |       // Workout just started - reset tracking
174 |       workoutStartTimeRef.current = Date.now();
175 |       locationHistoryRef.current = []; // Clear location history for fresh start
176 |       isStationaryRef.current = false; // Reset stationary state
177 |       lastStationaryCallRef.current = 0; // Reset throttle timer
178 |     } else if (!isActive) {
179 |       // Workout stopped - clear start time
180 |       workoutStartTimeRef.current = null;
181 |     }
182 |
183 |     prevIsActiveRef.current = isActive;
184 |   }, [isActive]);
185 |
186 |   // Set up periodic checking
187 |   useEffect(() => {
188 |     if (!isActive) {
189 |       cleanup();
190 |       // Reset stationary state when workout stops
191 |       isStationaryRef.current = false;
192 |       return;
193 |     }
194 |
195 |     // Check every 30 seconds for movement
196 |     // Continue checking even when paused to detect movement for resume
197 |     checkIntervalRef.current = setInterval(() => {
198 |       checkMovement(isPaused);
199 |     }, 30000);
200 |
201 |     // Also check immediately after locations update
202 |     checkMovement(isPaused);
203 |
204 |     return cleanup;
205 |   }, [isActive, isPaused, checkMovement, cleanup]);
206 |
207 |   // Cleanup on unmount
208 |   useEffect(() => {
```

```
209 |     return cleanup;
210 |   }, [cleanup]);
211 |
212 |   return {
213 |     isStationary: isStationaryRef.current,
214 |     locationCount: locationHistoryRef.current.length,
215 |   };
216 | };
217 |
218 |
```

File: src/hooks/useNearbyUsers.ts

Lines: 224

```
// Custom hook for finding nearby users
import { useState, useEffect, useRef } from "react";
import { listenToAllUsers } from "../services/locationService";
import { filterUsersByDistance } from "../utils/distance";
import { addEncounteredUser } from "../services/encounteredUsersService";

export interface NearbyUser {
  id: string;
  name?: string;
  email?: string;
  photoURL?: string;
  activity?: string | null;
  gender?: string | null;
  lat: number;
  lng: number;
  visible?: boolean;
  distance: number;
  [key: string]: any;
}

export interface Location {
  lat: number;
  lng: number;
}

/**
 * Custom hook to find and filter nearby users
 * @param {Location | null} currentLocation - { lat, lng } of current user
 * @param {number} maxDistanceKm - Maximum distance in kilometers
 * @param {string} activityFilter - Filter by activity: "running", "cycling", "walking", or "all"
 * @param {string} genderFilter - Filter by gender: "male", "female", or "all"
 * @param {string | null} currentUserId - Current user's ID to exclude from results
 * @param {boolean} isWorkoutActive - Whether to show nearby users (always true now - shows users when location is avail
 * @returns {Object} { nearbyUsers, loading }
 */
export const useNearbyUsers = (
  currentLocation: Location | null,
  maxDistanceKm: number = 5,
  activityFilter: string = "all",
  genderFilter: string = "all",
  currentUserId: string | null = null,
  isWorkoutActive: boolean = true // Default to true - show users when location is available
) => {
  const [nearbyUsers, setNearbyUsers] = useState<NearbyUser[]>([]);
  const [loading, setLoading] = useState(true);
  // Track which users we've already recorded encounters for in this session
  // to avoid duplicate tracking on every location update
  const trackedEncountersRef = useRef<Set<string>>(new Set());
  const lastEncounterCheckRef = useRef<number>(0);

  useEffect(() => {
    // Show nearby users when location is available (not just during active workout)
    // This allows users to see who's nearby even when not actively working out
    if (!currentLocation || !currentLocation.lat || !currentLocation.lng) {
      setNearbyUsers([]);
      setLoading(false);
      return;
    }

    // Store the latest users data for periodic re-evaluation
    let latestUsers: Record<string, any> = {};

    // Function to filter and update nearby users
    const filterAndUpdateUsers = (users: Record<string, any>) => {
      latestUsers = users;
      // Use currentLocation from closure (will be latest value when effect re-runs)
```

```
67 |       console.log("=== NEARBY USERS DEBUG ===");
68 |       console.log("All users from Firebase:", users);
69 |       console.log("Total users in database:", Object.keys(users || {}).length);
70 |       console.log("Current user ID:", currentUserId);
71 |       console.log("Current location:", currentLocation);
72 |       console.log("Max distance:", maxDistanceKm);
73 |
74 |       if (!users || Object.keys(users).length === 0) {
75 |         console.log("& þ  No users in Firebase database!");
76 |         setNearbyUsers([]);
77 |         setLoading(false);
78 |         return;
79 |       }
80 |
81 |       // Filter out current user
82 |       const otherUsers = { ...users };
83 |       if (currentUserId && otherUsers[currentUserId]) {
84 |         console.log("Removing current user from list");
85 |         delete otherUsers[currentUserId];
86 |       }
87 |
88 |       console.log("Other users (after filtering self):", otherUsers);
89 |       console.log("Other users count:", Object.keys(otherUsers).length);
90 |
91 |       if (Object.keys(otherUsers).length === 0) {
92 |         console.log("& þ  No other users found (only yourself in database)");
93 |         setNearbyUsers([]);
94 |         setLoading(false);
95 |         return;
96 |       }
97 |
98 |       // Filter by distance (with additional safety check to exclude current user)
99 |       let filtered = filterUsersByDistance(
100 |         otherUsers,
101 |         currentLocation.lat,
102 |         currentLocation.lng,
103 |         maxDistanceKm,
104 |         currentUserId // Extra safety: exclude current user in distance filter too
105 |       );
106 |
107 |       console.log("Users after distance filter:", filtered);
108 |       console.log("Users after distance filter count:", filtered.length);
109 |
110 |       // Track encounters for users within discovery radius
111 |       // Throttle to avoid excessive database writes (check every 10 seconds)
112 |       const now = Date.now();
113 |       if (currentUserId && now - lastEncounterCheckRef.current > 10000) {
114 |         lastEncounterCheckRef.current = now;
115 |
116 |         filtered.forEach((user) => {
117 |           // Only track if we haven't tracked this user in this session
118 |           // or if it's been more than 5 minutes since last tracking
119 |           const encounterKey = `${user.id}`;
120 |
121 |           if (!trackedEncountersRef.current.has(encounterKey)) {
122 |             // Track the encounter
123 |             addEncounteredUser(
124 |               currentUserId,
125 |               user.id,
126 |               user.distance,
127 |               { lat: user.lat, lng: user.lng }
128 |             ).catch((error) => {
129 |               // Silently fail - encounter tracking is non-critical
130 |               if (process.env.NODE_ENV === 'development') {
131 |                 console.error("Error tracking encounter:", error);
132 |               }
133 |             });
134 |
135 |             trackedEncountersRef.current.add(encounterKey);
136 |
137 |             // Clean up old tracked encounters after 5 minutes to allow re-tracking
```

```
138 |             setTimeout(() => {
139 |               trackedEncountersRef.current.delete(encounterKey);
140 |             }, 5 * 60 * 1000);
141 |           }
142 |         });
143 |       }
144 |
145 |       // Filter by activity
146 |       if (activityFilter !== "all") {
147 |         filtered = filtered.filter(
148 |           (user) => user.activity === activityFilter
149 |         );
150 |       }
151 |
152 |       // Filter by gender
153 |       if (genderFilter !== "all") {
154 |         filtered = filtered.filter((user) => user.gender === genderFilter);
155 |       }
156 |
157 |       // Only hide users if visible is explicitly false (default to visible if not set)
158 |       const beforeVisibility = filtered.length;
159 |       filtered = filtered.filter((user) => {
160 |         const isVisible = user.visible !== false;
161 |         if (!isVisible) {
162 |           console.log(`User ${user.id} filtered out - visible:`, user.visible);
163 |         }
164 |         return isVisible;
165 |       });
166 |
167 |       console.log(`Visibility filter: ${beforeVisibility} !' ${filtered.length}`);
168 |
169 |       // Filter by active workout status - only show users with recent location updates (within
10 minutes)
170 |       // This ensures users are only visible when they have an active workout session
171 |       // Using 10 minutes to match "active friends" threshold and allow for brief pauses
172 |       const beforeActiveFilter = filtered.length;
173 |       // Reuse 'now' from line 102 - no need to redeclare
174 |       const activeThreshold = 10 * 60 * 1000; // 10 minutes in milliseconds
175 |
176 |       filtered = filtered.filter((user) => {
177 |         // User must have a timestamp indicating recent location update
178 |         if (!user.timestamp) {
179 |           console.log(`User ${user.id} filtered out - no timestamp (not actively tracking)`);
180 |           return false;
181 |         }
182 |
183 |         // Check if timestamp is recent (within 3 minutes)
184 |         const timeDiff = now - user.timestamp;
185 |         const isActive = timeDiff <= activeThreshold;
186 |
187 |         if (!isActive) {
188 |           console.log(`User ${user.id} filtered out - timestamp too old (${Math.round(timeDiff /
1000)}s ago, threshold:...
189 |         }
190 |
191 |         return isActive;
192 |       });
193 |
194 |       console.log(`Active workout filter: ${beforeActiveFilter} !' ${filtered.length}`);
195 |
196 |       console.log("Final nearby users:", filtered);
197 |       console.log("==========================");
198 |
199 |       setNearbyUsers(filtered);
200 |       setLoading(false);
201 |     };
202 |
203 |     // Set up Firebase listener
204 |     const unsubscribe = listenToAllUsers(filterAndUpdateUsers);
205 |
206 |     // Set up periodic re-evaluation to filter out inactive users
207 |     // This ensures users who stopped working out are removed even if Firebase doesn't update
208 |     const intervalId = setInterval(() => {
```

```
209 |       if (Object.keys(latestUsers).length > 0) {
210 |         // Re-run filtering with latest data to remove inactive users
211 |         filterAndUpdateUsers(latestUsers);
212 |       }
213 |     }, 30000); // Check every 30 seconds
214 |
215 |     return () => {
216 |       unsubscribe();
217 |       clearInterval(intervalId);
218 |     };
219 |   }, [currentLocation, maxDistanceKm, activityFilter, genderFilter, currentUserId,
isWorkoutActive]);
220 |
221 |   return { nearbyUsers, loading };
222 | };
223 |
224 |
```

File: src/hooks/useVenueCheckIns.ts

Lines: 150

```
// Hook for managing venue check-ins
import { useState, useEffect, useCallback } from "react";
import {
  checkInToVenue,
  checkOutFromVenue,
  listenToVenueCheckIns,
  listenToUserCheckIn,
  getUserCheckIn,
  CheckIn,
  UserCheckInData,
  VenueData
} from "@/services/checkInService";
import { useAuth } from "./useAuth";
import { getUserData } from "@/services/authService";

export interface UseVenueCheckInsOptions {
  venueId?: string;
  autoLoad?: boolean;
}

export interface UseVenueCheckInsResult {
  checkIns: CheckIn[];
  userCheckIn: CheckIn | null;
  loading: boolean;
  error: string | null;
  checkIn: (venueId: string, venueData: VenueData, activity: "running" | "cycling" | "walking")
=> Promise<void>;
  checkOut: () => Promise<void>;
  isCheckedIn: boolean;
  refreshCheckIns: () => Promise<void>;
}

/**
 * Custom hook to manage venue check-ins
 */
export const useVenueCheckIns = (options: UseVenueCheckInsOptions = {}): UseVenueCheckInsResult
=> {
  const { venueId, autoLoad = true } = options;
  const { user } = useAuth();
  const [checkIns, setCheckIns] = useState<CheckIn[]>([]);
  const [userCheckIn, setUserCheckIn] = useState<CheckIn | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  // Load user's current check-in
  useEffect(() => {
    if (!user?.uid || !autoLoad) {
      setLoading(false);
      return;
    }

    const unsubscribe = listenToUserCheckIn(user.uid, (checkIn) => {
      setUserCheckIn(checkIn);
      setLoading(false);
    });

    return () => unsubscribe();
  }, [user?.uid, autoLoad]);

  // Load check-ins for a specific venue
  useEffect(() => {
    if (!venueId || !autoLoad) {
      setCheckIns([]);
      setLoading(false);
      return;
    }

    setLoading(true);
```

```
    const unsubscribe = listenToVenueCheckIns(venueId, (venueCheckIns) => {
      setCheckIns(venueCheckIns);
      setLoading(false);
      setError(null);
    });

    return () => unsubscribe();
  }, [venueId, autoLoad]);

  const checkIn = useCallback(async (
    venueId: string,
    venueData: VenueData,
    activity: "running" | "cycling" | "walking"
  ) => {
    if (!user?.uid) {
      throw new Error("User must be logged in to check in");
    }

    try {
      setError(null);

      // Get user data
      const userData = await getUserData(user.uid);
      if (!userData) {
        throw new Error("User data not found");
      }

      const userCheckInData: UserCheckInData = {
        userId: user.uid,
        userName: userData.name || "Unknown User",
        userAvatar: userData.photoURL || "",
        activity
      };

      await checkInToVenue(user.uid, venueId, venueData, userCheckInData);
    } catch (err: any) {
      const errorMessage = err.message || "Failed to check in";
      setError(errorMessage);
      throw err;
    }
  }, [user?.uid]);

  const checkOut = useCallback(async () => {
    if (!user?.uid || !userCheckIn) {
      throw new Error("No active check-in to check out from");
    }

    try {
      setError(null);
      await checkOutFromVenue(user.uid, userCheckIn.venueId);
    } catch (err: any) {
      const errorMessage = err.message || "Failed to check out";
      setError(errorMessage);
      throw err;
    }
  }, [user?.uid, userCheckIn]);

  const refreshCheckIns = useCallback(async () => {
    if (!venueId) return;

    try {
      setLoading(true);
      setError(null);
      // The real-time listener will update automatically
      setLoading(false);
    } catch (err: any) {
      setError(err.message || "Failed to refresh check-ins");
      setLoading(false);
    }
  }, [venueId]);

```

```
138 |   return {
139 |     checkIns,
140 |     userCheckIn,
141 |     loading,
142 |     error,
143 |     checkIn,
144 |     checkOut,
145 |     isCheckedIn: userCheckIn !== null,
146 |     refreshCheckIns
147 |   };
148 | };
149 |
150 |
```

File: src/index.css

Lines: 281

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Safe Area Insets - Set as CSS custom properties for JavaScript access */
:root {
  --safe-area-inset-top: env(safe-area-inset-top, 0px);
  --safe-area-inset-bottom: env(safe-area-inset-bottom, 0px);
  --safe-area-inset-left: env(safe-area-inset-left, 0px);
  --safe-area-inset-right: env(safe-area-inset-right, 0px);
}

/* Definition of the design system. All colors, gradients, fonts, etc should be defined here.
All colors MUST be HSL.
*/

@layer base {
  :root {
    /* Base colors */
    --background: 0 0% 100%;
    --foreground: 0 0% 13%;

    /* Card colors */
    --card: 0 0% 100%;
    --card-foreground: 0 0% 13%;

    /* Popover colors */
    --popover: 0 0% 100%;
    --popover-foreground: 0 0% 13%;

    /* Primary Blue (#1976d2) */
    --primary: 207 79% 47%;
    --primary-foreground: 0 0% 100%;

    /* Success Green (#2e7d32) */
    --success: 123 50% 34%;
    --success-foreground: 0 0% 100%;

    /* Error Red (#d32f2f) */
    --destructive: 4 67% 51%;
    --destructive-foreground: 0 0% 100%;

    /* Warning Orange (#ff9800) */
    --warning: 36 100% 50%;
    --warning-foreground: 0 0% 13%;

    /* Purple (#9c27b0) */
    --purple: 291 64% 42%;
    --purple-foreground: 0 0% 100%;

    /* Secondary colors */
    --secondary: 210 40% 96%;
    --secondary-foreground: 0 0% 13%;

    /* Muted colors */
    --muted: 0 0% 96%;
    --muted-foreground: 0 0% 46%;

    /* Accent colors */
    --accent: 207 79% 47%;
    --accent-foreground: 0 0% 100%;

    /* Border colors */
    --border: 0 0% 88%;
    --input: 0 0% 88%;
    --ring: 207 79% 47%;
```

```
 67 |
 68 |     /* Radius */
 69 |     --radius: 0.5rem;
 70 |
 71 |     /* Activity colors */
 72 |     --activity-running: 122 39% 49%;
 73 |     --activity-cycling: 207 90% 54%;
 74 |     --activity-walking: 45 100% 51%;
 75 |
 76 |     /* Sidebar colors */
 77 |     --sidebar-background: 0 0% 98%;
 78 |     --sidebar-foreground: 240 5% 26%;
 79 |     --sidebar-primary: 240 6% 10%;
 80 |     --sidebar-primary-foreground: 0 0% 98%;
 81 |     --sidebar-accent: 240 5% 96%;
 82 |     --sidebar-accent-foreground: 240 6% 10%;
 83 |     --sidebar-border: 220 13% 91%;
 84 |     --sidebar-ring: 217 91% 60%;
 85 |
 86 |     /* Shadows */
 87 |     --shadow-sm: 0 1px 2px 0 rgb(0 0 0 / 0.05);
 88 |     --shadow-md: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1);
 89 |     --shadow-lg: 0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 / 0.1);
 90 |     --shadow-xl: 0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 / 0.1);
 91 |
 92 |     /* Transitions */
 93 |     --transition-fast: 150ms cubic-bezier(0.4, 0, 0.2, 1);
 94 |     --transition-base: 300ms cubic-bezier(0.4, 0, 0.2, 1);
 95 |     --transition-slow: 500ms cubic-bezier(0.4, 0, 0.2, 1);
 96 |   }
 97 |
 98 |   .dark {
 99 |     /* Base colors */
100 |     --background: 0 0% 13%;
101 |     --foreground: 0 0% 98%;
102 |
103 |     /* Card colors */
104 |     --card: 0 0% 13%;
105 |     --card-foreground: 0 0% 98%;
106 |
107 |     /* Popover colors */
108 |     --popover: 0 0% 13%;
109 |     --popover-foreground: 0 0% 98%;
110 |
111 |     /* Primary Blue */
112 |     --primary: 207 79% 47%;
113 |     --primary-foreground: 0 0% 100%;
114 |
115 |     /* Success Green */
116 |     --success: 123 50% 34%;
117 |     --success-foreground: 0 0% 100%;
118 |
119 |     /* Error Red */
120 |     --destructive: 4 67% 51%;
121 |     --destructive-foreground: 0 0% 100%;
122 |
123 |     /* Warning Orange */
124 |     --warning: 36 100% 50%;
125 |     --warning-foreground: 0 0% 13%;
126 |
127 |     /* Purple */
128 |     --purple: 291 64% 42%;
129 |     --purple-foreground: 0 0% 100%;
130 |
131 |     /* Secondary colors */
132 |     --secondary: 217 33% 18%;
133 |     --secondary-foreground: 0 0% 98%;
134 |
135 |     /* Muted colors */
136 |     --muted: 217 33% 18%;
137 |     --muted-foreground: 215 20% 65%;
```

```
138 |
139 |     /* Accent colors */
140 |     --accent: 207 79% 47%;
141 |     --accent-foreground: 0 0% 100%;
142 |
143 |     /* Border colors */
144 |     --border: 217 33% 18%;
145 |     --input: 217 33% 18%;
146 |     --ring: 213 27% 84%;
147 |
148 |     /* Activity colors */
149 |     --activity-running: 122 39% 49%;
150 |     --activity-cycling: 207 90% 54%;
151 |     --activity-walking: 45 100% 51%;
152 |
153 |     /* Sidebar colors */
154 |     --sidebar-background: 240 6% 10%;
155 |     --sidebar-foreground: 240 5% 96%;
156 |     --sidebar-primary: 224 76% 48%;
157 |     --sidebar-primary-foreground: 0 0% 100%;
158 |     --sidebar-accent: 240 4% 16%;
159 |     --sidebar-accent-foreground: 240 5% 96%;
160 |     --sidebar-border: 240 4% 16%;
161 |     --sidebar-ring: 217 91% 60%;
162 |   }
163 | }
164 |
165 | @layer base {
166 |   * {
167 |     @apply border-border;
168 |   }
169 |
170 |   body {
171 |     @apply bg-background text-foreground;
172 |     font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
173 |       'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
174 |       sans-serif;
175 |     -webkit-font-smoothing: antialiased;
176 |     -moz-osx-font-smoothing: grayscale;
177 |   }
178 |
179 |   /* Mobile-first optimizations */
180 |   @media (max-width: 768px) {
181 |     html {
182 |       /* Prevent text size adjustment on iOS */
183 |       -webkit-text-size-adjust: 100%;
184 |     }
185 |
186 |     body {
187 |       /* Prevent pull-to-refresh on mobile */
188 |       overscroll-behavior-y: contain;
189 |     }
190 |   }
191 | }
192 |
193 | @layer utilities {
194 |   /* Touch-friendly utilities */
195 |   .touch-target {
196 |     min-width: 48px;
197 |     min-height: 48px;
198 |   }
199 |
200 |   /* Smooth animations */
201 |   .animate-smooth {
202 |     transition: all var(--transition-base);
203 |   }
204 |
205 |   .animate-fast {
206 |     transition: all var(--transition-fast);
207 |   }
208 |
```

```
  /* Shadow utilities */
  .shadow-elevation-1 {
    box-shadow: var(--shadow-sm);
  }

  .shadow-elevation-2 {
    box-shadow: var(--shadow-md);
  }

  .shadow-elevation-3 {
    box-shadow: var(--shadow-lg);
  }

  .shadow-elevation-4 {
    box-shadow: var(--shadow-xl);
  }

  /* Activity color utilities */
  .text-activity-running {
    color: hsl(var(--activity-running));
  }

  .text-activity-cycling {
    color: hsl(var(--activity-cycling));
  }

  .text-activity-walking {
    color: hsl(var(--activity-walking));
  }

  .bg-activity-running {
    background-color: hsl(var(--activity-running));
  }

  .bg-activity-cycling {
    background-color: hsl(var(--activity-cycling));
  }

  .bg-activity-walking {
    background-color: hsl(var(--activity-walking));
  }
}

@keyframes slide-in-bottom {
  from {
    transform: translateY(100%);
    opacity: 0;
  }
  to {
    transform: translateY(0);
    opacity: 1;
  }
}

.animate-slide-in-bottom {
  animation: slide-in-bottom 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}

/* Ensure Sonner toast notifications appear above all other elements */
[data-sonner-toaster] {
  z-index: 100 !important;
}

/* Hide scrollbar utility */
.scrollbar-hide {
  -ms-overflow-style: none;  /* IE and Edge */
  scrollbar-width: none;  /* Firefox */
}

.scrollbar-hide::-webkit-scrollbar {
  display: none;  /* Chrome, Safari and Opera */
```

```
280 | }
281 |
```

File: src/lib/avatars.ts

Lines: 113

```
/**
 * Default Avatar Options for PaceMatch
 * Uses DiceBear API - free avatar generation service
 * https://www.dicebear.com/
 */

// Avatar styles available from DiceBear
export type AvatarStyle =
  | "adventurer"      // Cartoon-style people
  | "adventurer-neutral" // Gender-neutral cartoon
  | "avataaars"       // Bitmoji-like avatars
  | "big-ears"        // Cute big-eared characters
  | "big-ears-neutral" // Gender-neutral big ears
  | "bottts"          // Robot avatars
  | "fun-emoji"       // Fun emoji faces
  | "lorelei"         // Minimalist line art
  | "micah"           // Modern illustrated people
  | "notionists"      // Notion-style avatars
  | "open-peeps"      // Hand-drawn people
  | "personas"        // Simple avatar personas
  | "pixel-art"       // Retro pixel art
  | "thumbs"          // Thumbs up characters;

// Pre-defined default avatars for users to choose from
// These are seeded avatars so they're always consistent
export interface DefaultAvatar {
  id: string;
  name: string;
  url: string;
  style: AvatarStyle;
}

/**
 * Generate a DiceBear avatar URL
 * @param style - The avatar style
 * @param seed - A seed string (like username) for consistent generation
 * @param size - Image size in pixels (default 128)
 */
export const generateAvatarUrl = (
  style: AvatarStyle,
  seed: string,
  size: number = 128
): string => {
  return `https://api.dicebear.com/7.x/${style}/svg?seed=${encodeURIComponent(seed)}&size=${size}
`;};

/**
 * Pre-defined default avatars for profile selection
 * Each has a unique seed for consistent appearance
 */
export const DEFAULT_AVATARS: DefaultAvatar[] = [
  // Adventurer style - Cartoon people
  { id: "adv-1", name: "Runner", url: generateAvatarUrl("adventurer", "runner-pace-1"), style:
"adventurer" },{ id: "adv-2", name: "Cyclist", url: generateAvatarUrl("adventurer", "cyclist-pace-2"), style:
"adventurer" },{ id: "adv-3", name: "Walker", url: generateAvatarUrl("adventurer", "walker-pace-3"), style:
"adventurer" },{ id: "adv-4", name: "Athlete", url: generateAvatarUrl("adventurer", "athlete-pace-4"), style:
"adventurer" },
  // Avataaars - Bitmoji-like
  { id: "ava-1", name: "Sporty", url: generateAvatarUrl("avataaars", "sporty-match-1"), style:
"avataaars" },{ id: "ava-2", name: "Active", url: generateAvatarUrl("avataaars", "active-match-2"), style:
"avataaars" },{ id: "ava-3", name: "Energetic", url: generateAvatarUrl("avataaars", "energetic-match-3"),
style: "avataaars" },{ id: "ava-4", name: "Fit", url: generateAvatarUrl("avataaars", "fit-match-4"), style:
"avataaars" },
  // Big Ears - Cute characters
  { id: "big-1", name: "Buddy", url: generateAvatarUrl("big-ears", "buddy-pace-1"), style: "big-
ears" }, { id: "big-2", name: "Friend", url: generateAvatarUrl("big-ears", "friend-pace-2"), style:
"big-ears" },
```

```
  67 |
  68 |   // Micah - Modern illustrated
  69 |   { id: "mic-1", name: "Modern", url: generateAvatarUrl("micah", "modern-pace-1"), style:
"micah" },
  70 |   { id: "mic-2", name: "Fresh", url: generateAvatarUrl("micah", "fresh-pace-2"), style:
"micah" },
  71 |
  72 |   // Fun Emoji
  73 |   { id: "fun-1", name: "Happy", url: generateAvatarUrl("fun-emoji", "happy-match-1"), style:
"fun-emoji" },
  74 |   { id: "fun-2", name: "Cool", url: generateAvatarUrl("fun-emoji", "cool-match-2"), style: "fun-
emoji" },
  75 |
  76 |   // Pixel Art - Retro style
  77 |   { id: "pix-1", name: "Retro", url: generateAvatarUrl("pixel-art", "retro-pace-1"), style:
"pixel-art" },
  78 |   { id: "pix-2", name: "Classic", url: generateAvatarUrl("pixel-art", "classic-pace-2"), style:
"pixel-art" },
  79 | ];
  80 |
  81 | /**
  82 |  * Generate a random avatar based on username
  83 |  * Creates a unique avatar for each user
  84 |  */
  85 | export const generateUserAvatar = (username: string): string => {
  86 |   return generateAvatarUrl("adventurer", username, 150);
  87 | };
  88 |
  89 | /**
  90 |  * Fitness-themed color backgrounds for avatars
  91 |  */
  92 | export const AVATAR_BACKGROUNDS = [
  93 |   "b6e3f4", // Light blue
  94 |   "c0aede", // Light purple
  95 |   "d1d4f9", // Lavender
  96 |   "ffd5dc", // Light pink
  97 |   "ffdfbf", // Light orange
  98 |   "d4f4dd", // Light green
  99 | ];
 100 |
 101 | /**
 102 |  * Generate avatar with custom background
 103 |  */
 104 | export const generateAvatarWithBackground = (
 105 |   style: AvatarStyle,
 106 |   seed: string,
 107 |   backgroundColor?: string
 108 | ): string => {
 109 |   const bg = backgroundColor || AVATAR_BACKGROUNDS[Math.floor(Math.random() *
AVATAR_BACKGROUNDS.length)];
 110 |   return `https://api.dicebear.com/7.x/${style}/svg?seed=${encodeURIComponent(seed)}
&backgroundColor=${bg}`;
 111 | };
 112 |
 113 |
```

File: src/lib/dummyData.ts

Lines: 602

```
  1 | // Centralized dummy data generators for demo/testing purposes
  2 | // All dummy data uses "dummy-" prefix in IDs to distinguish from real Firebase data
  3 |
  4 | import { WorkoutPost, Comment } from "@/services/feedService";
  5 | import { Event } from "@/services/eventService";
  6 | import { Message } from "@/services/messageService";
  7 | import { WorkoutHistory } from "@/contexts/UserContext";
  8 |
  9 | // Dummy user profiles for consistency across the app
 10 | export const dummyUsers = [
 11 |   {
 12 |     id: "dummy-user-1",
 13 |     name: "Sarah Johnson",
 14 |     username: "sarah_runs",
 15 |     photoURL: "https://i.pravatar.cc/150?img=47",
 16 |     activities: ["running", "walking"] as const,
 17 |     fitnessLevel: "intermediate" as const,
 18 |     pace: 5.2,
 19 |   },
 20 |   {
 21 |     id: "dummy-user-2",
 22 |     name: "Mike Chen",
 23 |     username: "mike_cycles",
 24 |     photoURL: "https://i.pravatar.cc/150?img=33",
 25 |     activities: ["cycling", "running"] as const,
 26 |     fitnessLevel: "pro" as const,
 27 |     pace: 25.5,
 28 |   },
 29 |   {
 30 |     id: "dummy-user-3",
 31 |     name: "Emma Wilson",
 32 |     username: "emma_walks",
 33 |     photoURL: "https://i.pravatar.cc/150?img=20",
 34 |     activities: ["walking", "running"] as const,
 35 |     fitnessLevel: "beginner" as const,
 36 |     pace: 7.8,
 37 |   },
 38 |   {
 39 |     id: "dummy-user-4",
 40 |     name: "James Wilson",
 41 |     username: "james_runs",
 42 |     photoURL: "https://i.pravatar.cc/150?img=12",
 43 |     activities: ["running"] as const,
 44 |     fitnessLevel: "intermediate" as const,
 45 |     pace: 4.8,
 46 |   },
 47 |   {
 48 |     id: "dummy-user-5",
 49 |     name: "Lisa Anderson",
 50 |     username: "lisa_cycles",
 51 |     photoURL: "https://i.pravatar.cc/150?img=5",
 52 |     activities: ["cycling"] as const,
 53 |     fitnessLevel: "pro" as const,
 54 |     pace: 28.0,
 55 |   },
 56 |   {
 57 |     id: "dummy-user-6",
 58 |     name: "Alex Runner",
 59 |     username: "alex_runner",
 60 |     photoURL: "https://ui-avatars.com/api/?
name=Alex+Runner&size=128&background=4CAF50&color=fff",
 61 |     activities: ["running", "walking"] as const,
 62 |     fitnessLevel: "intermediate" as const,
 63 |     pace: 5.5,
 64 |   },
 65 |   {
 66 |     id: "dummy-user-7",
```

```
  67 |     name: "David Active",
  68 |     username: "david_active",
  69 |     photoURL: "https://ui-avatars.com/api/?
name=David+Active&size=128&background=E01F68&color=fff",
  70 |     activities: ["running", "walking"] as const,
  71 |     fitnessLevel: "beginner" as const,
  72 |     pace: 6.5,
  73 |   },
  74 |   {
  75 |     id: "dummy-user-8",
  76 |     name: "Sophie Martinez",
  77 |     username: "sophie_fit",
  78 |     photoURL: "https://i.pravatar.cc/150?img=68",
  79 |     activities: ["cycling", "running"] as const,
  80 |     fitnessLevel: "intermediate" as const,
  81 |     pace: 6.0,
  82 |   },
  83 | ];
  84 |
  85 | /**
  86 |  * Generate dummy workout posts
  87 |  */
  88 | export const generateDummyWorkoutPosts = (): WorkoutPost[] => {
  89 |   const now = Date.now();
  90 |   const posts: WorkoutPost[] = [];
  91 |
  92 |   // Post 1: Recent running workout
  93 |   posts.push({
  94 |     id: "dummy-post-1",
  95 |     userId: dummyUsers[0].id,
  96 |     workout: {
  97 |       id: "dummy-workout-1",
  98 |       activity: "running",
  99 |       date: new Date(now - 2 * 60 * 60 * 1000), // 2 hours ago
 100 |       duration: 3600, // 1 hour
 101 |       distance: 10.5,
 102 |       avgSpeed: 10.5,
 103 |       location: "Central Park, New York",
 104 |       nearbyUsers: [
 105 |         {
 106 |           id: 1,
 107 |           name: dummyUsers[1].name,
 108 |           avatar: dummyUsers[1].photoURL,
 109 |           activity: "cycling",
 110 |           distance: "0.5 km",
 111 |         },
 112 |       ],
 113 |     },
 114 |     caption: "Great morning run! The weather was perfect today Ø<ß ",
 115 |     photos: ["https://images.unsplash.com/photo-1571008887538-b36bb32f4571?w=400"],
 116 |     kudos: [dummyUsers[1].id, dummyUsers[2].id],
 117 |     comments: [
 118 |       {
 119 |         id: "dummy-comment-1",
 120 |         userId: dummyUsers[1].id,
 121 |         username: dummyUsers[1].username,
 122 |         avatar: dummyUsers[1].photoURL,
 123 |         text: "Amazing pace! Keep it up! Ø=Ùª",
 124 |         timestamp: now - 1.5 * 60 * 60 * 1000,
 125 |       },
 126 |     ],
 127 |     timestamp: now - 2 * 60 * 60 * 1000,
 128 |   });
 129 |
 130 |   // Post 2: Cycling workout
 131 |   posts.push({
 132 |     id: "dummy-post-2",
 133 |     userId: dummyUsers[1].id,
 134 |     workout: {
 135 |       id: "dummy-workout-2",
 136 |       activity: "cycling",
 137 |       date: new Date(now - 5 * 60 * 60 * 1000), // 5 hours ago
```

```
      duration: 5400, // 1.5 hours
      distance: 35.0,
      avgSpeed: 23.3,
      location: "Hudson River Path",
    },
    caption: "Long ride along the river. Feeling strong! Ø=Þ´",
    kudos: [dummyUsers[0].id, dummyUsers[3].id, dummyUsers[4].id],
    comments: [
      {
        id: "dummy-comment-2",
        userId: dummyUsers[0].id,
        username: dummyUsers[0].username,
        avatar: dummyUsers[0].photoURL,
        text: "That's an impressive distance!",
        timestamp: now - 4.5 * 60 * 60 * 1000,
      },
      {
        id: "dummy-comment-3",
        userId: dummyUsers[4].id,
        username: dummyUsers[4].username,
        avatar: dummyUsers[4].photoURL,
        text: "Great route! I'll have to try it sometime.",
        timestamp: now - 4 * 60 * 60 * 1000,
      },
    ],
    timestamp: now - 5 * 60 * 60 * 1000,
  });

  // Post 3: Walking workout
  posts.push({
    id: "dummy-post-3",
    userId: dummyUsers[2].id,
    workout: {
      id: "dummy-workout-3",
      activity: "walking",
      date: new Date(now - 24 * 60 * 60 * 1000), // 1 day ago
      duration: 2700, // 45 minutes
      distance: 3.5,
      avgSpeed: 4.7,
      location: "Riverside Park",
    },
    caption: "Peaceful evening walk. Perfect way to unwind Ø=Þ¶",
    photos: ["https://images.unsplash.com/photo-1544966503-7cc5ac882d5f?w=400"],
    kudos: [dummyUsers[0].id],
    comments: [],
    timestamp: now - 24 * 60 * 60 * 1000,
  });

  // Post 4: Running workout (yesterday)
  posts.push({
    id: "dummy-post-4",
    userId: dummyUsers[3].id,
    workout: {
      id: "dummy-workout-4",
      activity: "running",
      date: new Date(now - 30 * 60 * 60 * 1000), // 30 hours ago
      duration: 2400, // 40 minutes
      distance: 8.0,
      avgSpeed: 12.0,
      location: "Prospect Park",
    },
    caption: "Morning tempo run. Pushed myself today!",
    kudos: [dummyUsers[0].id, dummyUsers[1].id, dummyUsers[2].id],
    comments: [
      {
        id: "dummy-comment-4",
        userId: dummyUsers[0].id,
        username: dummyUsers[0].username,
        avatar: dummyUsers[0].photoURL,
        text: "Nice work! Your pace is improving!",
        timestamp: now - 29 * 60 * 60 * 1000,
```

```
      },
    ],
    timestamp: now - 30 * 60 * 60 * 1000,
  });

  // Post 5: Cycling workout (2 days ago)
  posts.push({
    id: "dummy-post-5",
    userId: dummyUsers[4].id,
    workout: {
      id: "dummy-workout-5",
      activity: "cycling",
      date: new Date(now - 48 * 60 * 60 * 1000), // 2 days ago
      duration: 7200, // 2 hours
      distance: 50.0,
      avgSpeed: 25.0,
      location: "Brooklyn Bridge Loop",
    },
    caption: "Epic ride today! 50km in the books Ø<ß‰",
    photos: ["https://images.unsplash.com/photo-1558618666-fcd25c85cd64?w=400"],
    kudos: [dummyUsers[1].id, dummyUsers[3].id],
    comments: [],
    timestamp: now - 48 * 60 * 60 * 1000,
  });

  return posts;
};

/**
 * Generate dummy events
 */
export const generateDummyEvents = (userLocation?: { lat: number; lng: number }): Event[] => {
  const now = Date.now();
  const baseLat = userLocation?.lat || 14.5995;
  const baseLng = userLocation?.lng || 120.9842;

  const events: Event[] = [];

  // Event 1: Upcoming running event (tomorrow)
  const tomorrow = new Date();
  tomorrow.setDate(tomorrow.getDate() + 1);
  events.push({
    id: "dummy-event-1",
    title: "Morning Run in Central Park",
    description: "Join us for an early morning run around Central Park. All fitness levels welcome! We'll meet at the ma...
    type: "running",
    category: "user",
    date: tomorrow.toISOString().split('T')[0],
    time: "07:00",
    location: "Central Park, New York",
    distance: "1.2 km",
    distanceValue: 1.2,
    lat: baseLat + 0.01,
    lng: baseLng + 0.01,
    hostId: dummyUsers[0].id,
    hostName: dummyUsers[0].name,
    hostAvatar: dummyUsers[0].photoURL,
    participants: [dummyUsers[0].id, dummyUsers[3].id],
    maxParticipants: 20,
    createdAt: now - 2 * 24 * 60 * 60 * 1000,
  });

  // Event 2: Upcoming cycling event (3 days from now)
  const threeDays = new Date();
  threeDays.setDate(threeDays.getDate() + 3);
  events.push({
    id: "dummy-event-2",
    title: "Weekend Cycling Group",
    description: "Long distance cycling group ride. We'll cover 40km at a moderate pace. Bring water and snacks!"
    type: "cycling",
    category: "user",
```

```
    date: threeDays.toISOString().split('T')[0],
    time: "08:00",
    location: "Hudson River Path",
    distance: "2.5 km",
    distanceValue: 2.5,
    lat: baseLat - 0.02,
    lng: baseLng + 0.015,
    hostId: dummyUsers[1].id,
    hostName: dummyUsers[1].name,
    hostAvatar: dummyUsers[1].photoURL,
    participants: [dummyUsers[1].id, dummyUsers[4].id, dummyUsers[7].id],
    maxParticipants: 15,
    createdAt: now - 5 * 24 * 60 * 60 * 1000,
  });

  // Event 3: Upcoming walking event (this weekend)
  const weekend = new Date();
  weekend.setDate(weekend.getDate() + (6 - weekend.getDay())); // Next Saturday
  events.push({
    id: "dummy-event-3",
    title: "Nature Walk & Social",
    description: "Relaxing nature walk followed by coffee. Perfect for beginners and those who want to enjoy the outdoor...",
    type: "walking",
    category: "user",
    date: weekend.toISOString().split('T')[0],
    time: "10:00",
    location: "Riverside Park",
    distance: "0.8 km",
    distanceValue: 0.8,
    lat: baseLat + 0.008,
    lng: baseLng - 0.012,
    hostId: dummyUsers[2].id,
    hostName: dummyUsers[2].name,
    hostAvatar: dummyUsers[2].photoURL,
    participants: [dummyUsers[2].id, dummyUsers[0].id],
    createdAt: now - 3 * 24 * 60 * 60 * 1000,
  });

  // Event 4: Sponsored event (next week)
  const nextWeek = new Date();
  nextWeek.setDate(nextWeek.getDate() + 7);
  events.push({
    id: "dummy-event-4",
    title: "City Marathon Training Run",
    description: "Official training run for the upcoming city marathon. Sponsored by local running store. Free water and...",
    type: "running",
    category: "sponsored",
    date: nextWeek.toISOString().split('T')[0],
    time: "06:30",
    location: "Prospect Park",
    distance: "3.0 km",
    distanceValue: 3.0,
    lat: baseLat - 0.025,
    lng: baseLng - 0.02,
    sponsorLogo: "https://via.placeholder.com/100x100?text=Sponsor",
    participants: [dummyUsers[3].id, dummyUsers[0].id, dummyUsers[6].id],
    maxParticipants: 50,
    createdAt: now - 7 * 24 * 60 * 60 * 1000,
  });

  // Event 5: Past event (yesterday)
  const yesterday = new Date();
  yesterday.setDate(yesterday.getDate() - 1);
  events.push({
    id: "dummy-event-5",
    title: "Evening Cycling Tour",
    description: "Evening cycling tour through the city. We'll stop at various landmarks and finish with dinner.",
    type: "cycling",
    category: "user",
    date: yesterday.toISOString().split('T')[0],
    time: "18:00",
```

```
    location: "Brooklyn Bridge",
    distance: "1.5 km",
    distanceValue: 1.5,
    lat: baseLat + 0.015,
    lng: baseLng - 0.01,
    hostId: dummyUsers[4].id,
    hostName: dummyUsers[4].name,
    hostAvatar: dummyUsers[4].photoURL,
    participants: [dummyUsers[4].id, dummyUsers[1].id],
    createdAt: now - 10 * 24 * 60 * 60 * 1000,
  });

  return events;
};

/**
 * Generate dummy conversations
 */
export const generateDummyConversations = (currentUserId: string) => {
  const now = Date.now();

  return [
    {
      conversationId: `${currentUserId}_${dummyUsers[0].id}`.split('_').sort().join('_'),
      otherUserId: dummyUsers[0].id,
      lastMessage: "Thanks for the great run today! Let's do it again soon.",
      lastMessageTime: now - 30 * 60 * 1000, // 30 minutes ago
      unreadCount: 0,
      userName: dummyUsers[0].name,
      avatar: dummyUsers[0].photoURL,
    },
    {
      conversationId: `${currentUserId}_${dummyUsers[1].id}`.split('_').sort().join('_'),
      otherUserId: dummyUsers[1].id,
      lastMessage: "Are you joining the cycling event this weekend?",
      lastMessageTime: now - 2 * 60 * 60 * 1000, // 2 hours ago
      unreadCount: 1,
      userName: dummyUsers[1].name,
      avatar: dummyUsers[1].photoURL,
    },
    {
      conversationId: `${currentUserId}_${dummyUsers[2].id}`.split('_').sort().join('_'),
      otherUserId: dummyUsers[2].id,
      lastMessage: "The nature walk was amazing! Thanks for organizing.",
      lastMessageTime: now - 5 * 60 * 60 * 1000, // 5 hours ago
      unreadCount: 0,
      userName: dummyUsers[2].name,
      avatar: dummyUsers[2].photoURL,
    },
    {
      conversationId: `${currentUserId}_${dummyUsers[3].id}`.split('_').sort().join('_'),
      otherUserId: dummyUsers[3].id,
      lastMessage: "Hey! Want to go for a run together tomorrow?",
      lastMessageTime: now - 24 * 60 * 60 * 1000, // 1 day ago
      unreadCount: 2,
      userName: dummyUsers[3].name,
      avatar: dummyUsers[3].photoURL,
    },
    {
      conversationId: `${currentUserId}_${dummyUsers[4].id}`.split('_').sort().join('_'),
      otherUserId: dummyUsers[4].id,
      lastMessage: "Great ride today! Your pace was impressive.",
      lastMessageTime: now - 2 * 24 * 60 * 60 * 1000, // 2 days ago
      unreadCount: 0,
      userName: dummyUsers[4].name,
      avatar: dummyUsers[4].photoURL,
    },
  ];
};

/**
```

```
 422 |  * Generate dummy chat messages
 423 |  */
 424 | export const generateDummyChatMessages = (currentUserId: string, otherUserId: string): Message[]
=425 |   const now = Date.now();
 426 |   const messages: Message[] = [];
 427 |
 428 |   // Determine which user is which
 429 |   const otherUser = dummyUsers.find(u => u.id === otherUserId);
 430 |   if (!otherUser) return [];
 431 |
 432 |   // Add conversation history
 433 |   messages.push({
 434 |     id: "dummy-msg-1",
 435 |     senderId: otherUser.id,
 436 |     receiverId: currentUserId,
 437 |     content: `Hey! I saw you're into ${otherUser.activities[0]}. Want to meet up for a workout?`,
 438 |     timestamp: now - 3 * 24 * 60 * 60 * 1000, // 3 days ago
 439 |     isRead: true,
 440 |   });
 441 |
 442 |   messages.push({
 443 |     id: "dummy-msg-2",
 444 |     senderId: currentUserId,
 445 |     receiverId: otherUser.id,
 446 |     content: "That sounds great! When are you free?",
 447 |     timestamp: now - 2 * 24 * 60 * 60 * 1000, // 2 days ago
 448 |     isRead: true,
 449 |   });
 450 |
 451 |   messages.push({
 452 |     id: "dummy-msg-3",
 453 |     senderId: otherUser.id,
 454 |     receiverId: currentUserId,
 455 |     content: "How about tomorrow morning? Around 7am?",
 456 |     timestamp: now - 2 * 24 * 60 * 60 * 1000 + 30 * 60 * 1000,
 457 |     isRead: true,
 458 |   });
 459 |
 460 |   messages.push({
 461 |     id: "dummy-msg-4",
 462 |     senderId: currentUserId,
 463 |     receiverId: otherUser.id,
 464 |     content: "Perfect! Let's meet at Central Park entrance.",
 465 |     timestamp: now - 2 * 24 * 60 * 60 * 1000 + 60 * 60 * 1000,
 466 |     isRead: true,
 467 |   });
 468 |
 469 |   messages.push({
 470 |     id: "dummy-msg-5",
 471 |     senderId: otherUser.id,
 472 |     receiverId: currentUserId,
 473 |     content: "Sounds good! See you there! Ø<ßÃ",
 474 |     timestamp: now - 1 * 24 * 60 * 60 * 1000, // 1 day ago
 475 |     isRead: true,
 476 |   });
 477 |
 478 |   messages.push({
 479 |     id: "dummy-msg-6",
 480 |     senderId: currentUserId,
 481 |     receiverId: otherUser.id,
 482 |     content: "Thanks for the great workout today!",
 483 |     timestamp: now - 12 * 60 * 60 * 1000, // 12 hours ago
 484 |     isRead: true,
 485 |   });
 486 |
 487 |   messages.push({
 488 |     id: "dummy-msg-7",
 489 |     senderId: otherUser.id,
 490 |     receiverId: currentUserId,
 491 |     content: "You too! Let's do it again soon!",
 492 |     timestamp: now - 30 * 60 * 1000, // 30 minutes ago
```

```
493 |     isRead: false,
494 |   });
495 |
496 |   return messages.sort((a, b) => a.timestamp - b.timestamp);
497 | };
498 |
499 | /**
500 |  * Generate dummy workout history
501 |  */
502 | export const generateDummyWorkoutHistory = (): WorkoutHistory[] => {
503 |   const now = Date.now();
504 |   const workouts: WorkoutHistory[] = [];
505 |
506 |   // Workout 1: Today
507 |   workouts.push({
508 |     id: "dummy-history-1",
509 |     activity: "running",
510 |     date: new Date(now - 2 * 60 * 60 * 1000),
511 |     duration: 3600, // 1 hour
512 |     distance: 10.5,
513 |     avgSpeed: 10.5,
514 |     location: "Central Park, New York",
515 |     nearbyUsers: [
516 |       {
517 |         id: 1,
518 |         name: dummyUsers[1].name,
519 |         avatar: dummyUsers[1].photoURL,
520 |         activity: "cycling",
521 |         distance: "0.5 km",
522 |       },
523 |     ],
524 |   });
525 |
526 |   // Workout 2: Yesterday
527 |   workouts.push({
528 |     id: "dummy-history-2",
529 |     activity: "cycling",
530 |     date: new Date(now - 26 * 60 * 60 * 1000),
531 |     duration: 5400, // 1.5 hours
532 |     distance: 35.0,
533 |     avgSpeed: 23.3,
534 |     location: "Hudson River Path",
535 |   });
536 |
537 |   // Workout 3: 2 days ago
538 |   workouts.push({
539 |     id: "dummy-history-3",
540 |     activity: "walking",
541 |     date: new Date(now - 50 * 60 * 60 * 1000),
542 |     duration: 2700, // 45 minutes
543 |     distance: 3.5,
544 |     avgSpeed: 4.7,
545 |     location: "Riverside Park",
546 |   });
547 |
548 |   // Workout 4: 3 days ago
549 |   workouts.push({
550 |     id: "dummy-history-4",
551 |     activity: "running",
552 |     date: new Date(now - 74 * 60 * 60 * 1000),
553 |     duration: 2400, // 40 minutes
554 |     distance: 8.0,
555 |     avgSpeed: 12.0,
556 |     location: "Prospect Park",
557 |   });
558 |
559 |   // Workout 5: 5 days ago
560 |   workouts.push({
561 |     id: "dummy-history-5",
562 |     activity: "cycling",
563 |     date: new Date(now - 122 * 60 * 60 * 1000),
```

```
564 |     duration: 7200, // 2 hours
565 |     distance: 50.0,
566 |     avgSpeed: 25.0,
567 |     location: "Brooklyn Bridge Loop",
568 |   });
569 |
570 |   // Workout 6: 1 week ago
571 |   workouts.push({
572 |     id: "dummy-history-6",
573 |     activity: "running",
574 |     date: new Date(now - 170 * 60 * 60 * 1000),
575 |     duration: 3000, // 50 minutes
576 |     distance: 9.5,
577 |     avgSpeed: 11.4,
578 |     location: "Central Park",
579 |     nearbyUsers: [
580 |       {
581 |         id: 1,
582 |         name: dummyUsers[3].name,
583 |         avatar: dummyUsers[3].photoURL,
584 |         activity: "running",
585 |         distance: "0.3 km",
586 |       },
587 |     ],
588 |   });
589 |
590 |   return workouts;
591 | };
592 |
593 | /**
594 |  * Global flag to enable/disable dummy data
595 |  * Set via environment variable VITE_ENABLE_DUMMY_DATA
596 |  * Defaults to false in production, true in development
597 |  */
598 | export const ENABLE_DUMMY_DATA =
599 |   import.meta.env.VITE_ENABLE_DUMMY_DATA === 'true' ||
600 |   (import.meta.env.VITE_ENV !== 'production' && import.meta.env.VITE_ENABLE_DUMMY_DATA !==
'false');
601 |
602 |
```

File: src/lib/messageStorage.ts

Lines: 94

```
export type MessageRequestStatus = 'pending' | 'accepted' | 'declined';

export interface ConversationMetadata {
  userId: number;
  isMuted: boolean;
  isArchived: boolean;
  lastMessageTime: number;
}

// Message request management
export const getMessageRequestStatus = (userId: number): MessageRequestStatus | null => {
  const stored = localStorage.getItem('messageRequests');
  if (!stored) return null;

  const requests = JSON.parse(stored);
  return requests[userId] || null;
};

export const acceptMessageRequest = (userId: number): void => {
  const stored = localStorage.getItem('messageRequests') || '{}';
  const requests = JSON.parse(stored);
  requests[userId] = 'accepted';
  localStorage.setItem('messageRequests', JSON.stringify(requests));
};

export const declineMessageRequest = (userId: number): void => {
  const stored = localStorage.getItem('messageRequests') || '{}';
  const requests = JSON.parse(stored);
  requests[userId] = 'declined';
  localStorage.setItem('messageRequests', JSON.stringify(requests));
};

export const deleteMessageRequest = (userId: number): void => {
  const stored = localStorage.getItem('messageRequests') || '{}';
  const requests = JSON.parse(stored);
  delete requests[userId];
  localStorage.setItem('messageRequests', JSON.stringify(requests));
};

// Blocked users management
export const getBlockedUsers = (): number[] => {
  const stored = localStorage.getItem('blockedUsers');
  return stored ? JSON.parse(stored) : [];
};

export const blockUser = (userId: number): void => {
  const blocked = getBlockedUsers();
  if (!blocked.includes(userId)) {
    blocked.push(userId);
    localStorage.setItem('blockedUsers', JSON.stringify(blocked));
  }
};

export const unblockUser = (userId: number): void => {
  const blocked = getBlockedUsers();
  const updated = blocked.filter(id => id !== userId);
  localStorage.setItem('blockedUsers', JSON.stringify(updated));
};

export const isUserBlocked = (userId: number): boolean => {
  return getBlockedUsers().includes(userId);
};

// Conversation metadata management
export const getConversationMetadata = (userId: number): ConversationMetadata => {
  const stored = localStorage.getItem(`conversation-${userId}`);
```

```
  if (stored) {
    return JSON.parse(stored);
  }
  return {
    userId,
    isMuted: false,
    isArchived: false,
    lastMessageTime: Date.now(),
  };
};

export const muteConversation = (userId: number, mute: boolean): void => {
  const metadata = getConversationMetadata(userId);
  metadata.isMuted = mute;
  localStorage.setItem(`conversation-${userId}`, JSON.stringify(metadata));
};

export const archiveConversation = (userId: number, archive: boolean): void => {
  const metadata = getConversationMetadata(userId);
  metadata.isArchived = archive;
  localStorage.setItem(`conversation-${userId}`, JSON.stringify(metadata));
};

export const deleteConversation = (userId: number): void => {
  localStorage.removeItem(`conversation-${userId}`);
  deleteMessageRequest(userId);
};

```

File: src/lib/mockData.ts

Lines: 61

```
  1 | import { Activity, WorkoutHistory } from "@/contexts/UserContext";
  2 |
  3 | export interface MockUser {
  4 |   id: number;
  5 |   username: string;
  6 |   avatar: string;
  7 |   activities: Activity[];
  8 |   bio?: string;
  9 |   photos?: string[];
 10 | }
 11 |
 12 | export interface Comment {
 13 |   id: string;
 14 |   userId: number;
 15 |   username: string;
 16 |   avatar: string;
 17 |   text: string;
 18 |   timestamp: Date;
 19 | }
 20 |
 21 | export interface WorkoutPost {
 22 |   id: string;
 23 |   userId: number;
 24 |   workout: WorkoutHistory;
 25 |   photos?: string[];
 26 |   caption?: string;
 27 |   kudos: number[];
 28 |   comments: Comment[];
 29 |   timestamp: Date;
 30 | }
 31 |
 32 | // Mock users database - REMOVED: Now using Firebase users
 33 | // Users are fetched from Firebase Realtime Database via locationService
 34 | export const mockUsers: MockUser[] = [];
 35 |
 36 | // REMOVED: Workout posts are now fetched from Firebase via feedService
 37 | // Use feedService.listenToWorkoutPosts() or feedService.getWorkoutPosts() instead
 38 | export const mockWorkoutPosts: WorkoutPost[] = [];
 39 |
 40 | // Helper to get user by ID - REMOVED: Now fetch users from Firebase
 41 | // Use locationService or getUserData from authService instead
 42 | export const getMockUserById = (id: number): MockUser | undefined => {
 43 |   return undefined; // No longer using mock users
 44 | };
 45 |
 46 | // Mock conversation data with message request status
 47 | export interface MockConversation {
 48 |   id: number;
 49 |   userId: number;
 50 |   userName: string;
 51 |   avatar: string;
 52 |   lastMessage: string;
 53 |   timestamp: number;
 54 |   unreadCount: number;
 55 |   isRequest?: boolean; // true if this is a message request
 56 | }
 57 |
 58 | // REMOVED: Conversations are now fetched from Firebase via messageService
 59 | // Use messageService.getUserConversations() instead
 60 | export const mockConversations: MockConversation[] = [];
 61 |
```

File: src/lib/socialStorage.ts

Lines: 127

```
  1 | import { Comment } from "./mockData";
  2 |
  3 | // Kudos management
  4 | export const getKudosForPost = (postId: string): number[] => {
  5 |   const stored = localStorage.getItem(`kudos-${postId}`);
  6 |   return stored ? JSON.parse(stored) : [];
  7 | };
  8 |
  9 | export const toggleKudos = (postId: string, userId: number): boolean => {
 10 |   const kudos = getKudosForPost(postId);
 11 |   const hasKudos = kudos.includes(userId);
 12 |
 13 |   if (hasKudos) {
 14 |     const updated = kudos.filter(id => id !== userId);
 15 |     localStorage.setItem(`kudos-${postId}`, JSON.stringify(updated));
 16 |     return false;
 17 |   } else {
 18 |     const updated = [...kudos, userId];
 19 |     localStorage.setItem(`kudos-${postId}`, JSON.stringify(updated));
 20 |     return true;
 21 |   }
 22 | };
 23 |
 24 | // Comments management
 25 | export const getCommentsForPost = (postId: string): Comment[] => {
 26 |   const stored = localStorage.getItem(`comments-${postId}`);
 27 |   if (!stored) return [];
 28 |
 29 |   const parsed = JSON.parse(stored);
 30 |   return parsed.map((c: any) => ({
 31 |     ...c,
 32 |     timestamp: new Date(c.timestamp),
 33 |   }));
 34 | };
 35 |
 36 | export const addComment = (postId: string, comment: Comment): void => {
 37 |   const comments = getCommentsForPost(postId);
 38 |   comments.push(comment);
 39 |   localStorage.setItem(`comments-${postId}`, JSON.stringify(comments));
 40 | };
 41 |
 42 | export const deleteComment = (postId: string, commentId: string): void => {
 43 |   const comments = getCommentsForPost(postId);
 44 |   const updated = comments.filter(c => c.id !== commentId);
 45 |   localStorage.setItem(`comments-${postId}`, JSON.stringify(updated));
 46 | };
 47 |
 48 | // Friend requests
 49 | export const getPendingRequests = (): { incoming: number[]; outgoing: number[] } => {
 50 |   const stored = localStorage.getItem("friendRequests");
 51 |   return stored ? JSON.parse(stored) : { incoming: [2, 7], outgoing: [9] };
 52 | };
 53 |
 54 | export const sendFriendRequest = (toUserId: number): void => {
 55 |   const requests = getPendingRequests();
 56 |   if (!requests.outgoing.includes(toUserId)) {
 57 |     requests.outgoing.push(toUserId);
 58 |     localStorage.setItem("friendRequests", JSON.stringify(requests));
 59 |   }
 60 | };
 61 |
 62 | export const acceptFriendRequest = (fromUserId: number): void => {
 63 |   const requests = getPendingRequests();
 64 |   requests.incoming = requests.incoming.filter(id => id !== fromUserId);
 65 |   localStorage.setItem("friendRequests", JSON.stringify(requests));
 66 |
```

```
  // Add to friends list in UserContext
  const userProfile = localStorage.getItem("userProfile");
  if (userProfile) {
    const profile = JSON.parse(userProfile);
    profile.friends = profile.friends || [];
    if (!profile.friends.includes(fromUserId)) {
      profile.friends.push(fromUserId);
      localStorage.setItem("userProfile", JSON.stringify(profile));
    }
  }
};

export const declineFriendRequest = (fromUserId: number): void => {
  const requests = getPendingRequests();
  requests.incoming = requests.incoming.filter(id => id !== fromUserId);
  localStorage.setItem("friendRequests", JSON.stringify(requests));
};

export const cancelFriendRequest = (toUserId: number): void => {
  const requests = getPendingRequests();
  requests.outgoing = requests.outgoing.filter(id => id !== toUserId);
  localStorage.setItem("friendRequests", JSON.stringify(requests));
};

// Unfriend a user
export const unfriend = (userId: number): void => {
  const userProfile = localStorage.getItem("userProfile");
  if (userProfile) {
    const profile = JSON.parse(userProfile);
    profile.friends = profile.friends || [];
    profile.friends = profile.friends.filter((id: number) => id !== userId);
    localStorage.setItem("userProfile", JSON.stringify(profile));
  }

  // Also remove from location sharing settings
  const locationSharing = getLocationSharingSettings();
  delete locationSharing[userId];
  localStorage.setItem("locationSharing", JSON.stringify(locationSharing));
};

// Location sharing settings (which friends can see your location during workouts)
export const getLocationSharingSettings = (): Record<number, boolean> => {
  const stored = localStorage.getItem("locationSharing");
  return stored ? JSON.parse(stored) : {};
};

export const setLocationSharing = (friendId: number, enabled: boolean): void => {
  const settings = getLocationSharingSettings();
  if (enabled) {
    settings[friendId] = true;
  } else {
    delete settings[friendId];
  }
  localStorage.setItem("locationSharing", JSON.stringify(settings));
};

export const canFriendSeeLocation = (friendId: number): boolean => {
  const settings = getLocationSharingSettings();
  return settings[friendId] === true;
};

```

File: src/lib/utils.ts

Lines: 7

```
1 | import { clsx, type ClassValue } from "clsx";
2 | import { twMerge } from "tailwind-merge";
3 |
4 | export function cn(...inputs: ClassValue[]) {
5 |   return twMerge(clsx(inputs));
6 | }
7 |
```

File: src/main.tsx

Lines: 14

```
 1 | import { createRoot } from "react-dom/client";
 2 | import App from "./App.tsx";
 3 | import "./index.css";
 4 | import { initializeStatusBar } from "./utils/safeArea";
 5 | import { initEmailJS } from "./services/emailServiceSimple";
 6 |
 7 | // Initialize StatusBar for edge-to-edge display on mobile
 8 | initializeStatusBar().catch(console.error);
 9 |
10 | // Initialize EmailJS for email OTP (if configured)
11 | initEmailJS();
12 |
13 | createRoot(document.getElementById("root")!).render(<App />);
14 |
```

File: src/pages/AdminAnalytics.tsx

Lines: 280

```
  1 | // Admin Analytics Dashboard
  2 | import { useState, useEffect } from "react";
  3 | import { Card } from "@/components/ui/card";
  4 | import { Button } from "@/components/ui/button";
  5 | import { getSystemStats, getAllUsers } from "@/services/adminService";
  6 | import { toast } from "sonner";
  7 | import BarChartIcon from "@mui/icons-material/BarChart";
  8 | import PeopleIcon from "@mui/icons-material/People";
  9 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
 10 | import EventIcon from "@mui/icons-material/Event";
 11 | import ReportProblemIcon from "@mui/icons-material/ReportProblem";
 12 | import DownloadIcon from "@mui/icons-material/Download";
 13 |
 14 | const AdminAnalytics = () => {
 15 |   const [stats, setStats] = useState<any>(null);
 16 |   const [loading, setLoading] = useState(true);
 17 |   const [userGrowth, setUserGrowth] = useState<any[]>([]);
 18 |   const [activityDistribution, setActivityDistribution] = useState<any>({});
 19 |
 20 |   useEffect(() => {
 21 |     loadAnalytics();
 22 |   }, []);
 23 |
 24 |   const loadAnalytics = async () => {
 25 |     try {
 26 |       setLoading(true);
 27 |       const statsData = await getSystemStats();
 28 |       setStats(statsData);
 29 |
 30 |       // Calculate user growth (simplified - by creation date)
 31 |       const users = await getAllUsers();
 32 |       const growthData: any = {};
 33 |
 34 |       users.forEach((user: any) => {
 35 |         if (user.createdAt) {
 36 |           const date = new Date(user.createdAt).toLocaleDateString('en-US', { month: 'short',
year: 'numeric' });
 37 |           growthData[date] = (growthData[date] || 0) + 1;
 38 |         }
 39 |       });
 40 |
 41 |       const growthArray = Object.entries(growthData)
 42 |         .map(([date, count]) => ({ date, count }))
 43 |         .sort((a, b) => new Date(a.date).getTime() - new Date(b.date).getTime());
 44 |
 45 |       setUserGrowth(growthArray);
 46 |
 47 |       // Calculate activity distribution
 48 |       const activities: any = {};
 49 |       users.forEach((user: any) => {
 50 |         const activity = user.activity || "unknown";
 51 |         activities[activity] = (activities[activity] || 0) + 1;
 52 |       });
 53 |       setActivityDistribution(activities);
 54 |     } catch (error) {
 55 |       console.error("Error loading analytics:", error);
 56 |       toast.error("Failed to load analytics");
 57 |     } finally {
 58 |       setLoading(false);
 59 |     }
 60 |   };
 61 |
 62 |   const exportData = () => {
 63 |     const data = {
 64 |       stats,
 65 |       userGrowth,
 66 |       activityDistribution,
```

```
      exportedAt: new Date().toISOString()
    };

    const blob = new Blob([JSON.stringify(data, null, 2)], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `pacematch-analytics-${Date.now()}.json`;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);

    toast.success("Analytics data exported");
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-64">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
        <div>
          <h1 className="text-3xl font-bold">Analytics Dashboard</h1>
          <p className="text-muted-foreground mt-1">
            View app statistics and user insights
          </p>
        </div>
        <Button onClick={exportData} variant="outline">
          <DownloadIcon className="mr-2" style={{ fontSize: 18 }} />
          Export Data
        </Button>
      </div>

      {/* Key Metrics */}
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-5 gap-4">
        <Card className="p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm font-medium text-muted-foreground">Total Users</p>
              <p className="text-3xl font-bold mt-2">{stats?.totalUsers || 0}</p>
            </div>
            <div className="p-3 bg-primary/10 rounded-lg">
              <PeopleIcon className="text-primary" style={{ fontSize: 28 }} />
            </div>
          </div>
        </Card>

        <Card className="p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm font-medium text-muted-foreground">Active (30d)</p>
              <p className="text-3xl font-bold mt-2">{stats?.activeUsers || 0}</p>
            </div>
            <div className="p-3 bg-success/10 rounded-lg">
              <PeopleIcon className="text-success" style={{ fontSize: 28 }} />
            </div>
          </div>
        </Card>

        <Card className="p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm font-medium text-muted-foreground">Total Workouts</p>
              <p className="text-3xl font-bold mt-2">{stats?.totalWorkouts || 0}</p>
```

```
            </div>
            <div className="p-3 bg-warning/10 rounded-lg">
              <DirectionsRunIcon className="text-warning" style={{ fontSize: 28 }} />
            </div>
          </div>
        </Card>

        <Card className="p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm font-medium text-muted-foreground">Total Events</p>
              <p className="text-3xl font-bold mt-2">{stats?.totalEvents || 0}</p>
            </div>
            <div className="p-3 bg-info/10 rounded-lg">
              <EventIcon className="text-info" style={{ fontSize: 28 }} />
            </div>
          </div>
        </Card>

        <Card className="p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-sm font-medium text-muted-foreground">Pending Reports</p>
              <p className="text-3xl font-bold mt-2">{stats?.pendingReports || 0}</p>
            </div>
            <div className="p-3 bg-destructive/10 rounded-lg">
              <ReportProblemIcon className="text-destructive" style={{ fontSize: 28 }} />
            </div>
          </div>
        </Card>
      </div>

      {/* User Growth Chart */}
      <Card className="p-6">
        <h3 className="text-lg font-semibold mb-4">User Growth</h3>
        {userGrowth.length === 0 ? (
          <p className="text-sm text-muted-foreground">No growth data available</p>
        ) : (
          <div className="space-y-2">
            {userGrowth.map((item, index) => (
              <div key={index} className="flex items-center gap-4">
                <div className="w-24 text-sm text-muted-foreground">{item.date}</div>
                <div className="flex-1 bg-muted rounded-full h-6 relative overflow-hidden">
                  <div
                    className="bg-primary h-full rounded-full flex items-center justify-end pr-2"
                    style={{ width: `${(item.count / Math.max(...userGrowth.map(i => i.count)))
* 100}%` }}
                  >
                    <span className="text-xs font-medium text-primary-foreground">{item.count}</
span>
                  </div>
                </div>
              </div>
            ))}
          </div>
        )}
      </Card>

      {/* Activity Distribution */}
      <Card className="p-6">
        <h3 className="text-lg font-semibold mb-4">Activity Distribution</h3>
        {Object.keys(activityDistribution).length === 0 ? (
          <p className="text-sm text-muted-foreground">No activity data available</p>
        ) : (
          <div className="space-y-3">
            {Object.entries(activityDistribution).map(([activity, count]: [string, any]) => {
              const total = Object.values(activityDistribution).reduce((sum: number, val: any)
=> sum + val, 0);
              const percentage = ((count / total) * 100).toFixed(1);

              return (
                <div key={activity} className="space-y-2">
                  <div className="flex items-center justify-between">
                    <span className="text-sm font-medium capitalize">{activity}</span>
```

```
                    <span className="text-sm text-muted-foreground">{count} users ({percentage}%)</span>
                  </div>
                  <div className="w-full bg-muted rounded-full h-2">
                    <div
                      className="bg-primary h-2 rounded-full"
                      style={{ width: `${percentage}%` }}
                    />
                  </div>
                </div>
              );
            })}
          </div>
        )}
      </Card>

      {/* Additional Stats */}
      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
        <Card className="p-6">
          <h3 className="text-lg font-semibold mb-4">Engagement Metrics</h3>
          <div className="space-y-3">
            <div className="flex justify-between">
              <span className="text-sm text-muted-foreground">Active Rate (30d)</span>
              <span className="text-sm font-medium">
                {stats?.totalUsers > 0
                  ? ((stats.activeUsers / stats.totalUsers) * 100).toFixed(1)
                  : 0}%
              </span>
            </div>
            <div className="flex justify-between">
              <span className="text-sm text-muted-foreground">Avg Workouts per User</span>
              <span className="text-sm font-medium">
                {stats?.totalUsers > 0
                  ? (stats.totalWorkouts / stats.totalUsers).toFixed(1)
                  : 0}
              </span>
            </div>
            <div className="flex justify-between">
              <span className="text-sm text-muted-foreground">Avg Events per User</span>
              <span className="text-sm font-medium">
                {stats?.totalUsers > 0
                  ? (stats.totalEvents / stats.totalUsers).toFixed(1)
                  : 0}
              </span>
            </div>
          </div>
        </Card>

        <Card className="p-6">
          <h3 className="text-lg font-semibold mb-4">System Health</h3>
          <div className="space-y-3">
            <div className="flex justify-between">
              <span className="text-sm text-muted-foreground">Report Rate</span>
              <span className="text-sm font-medium">
                {stats?.totalUsers > 0
                  ? ((stats.pendingReports / stats.totalUsers) * 100).toFixed(2)
                  : 0}%
              </span>
            </div>
            <div className="flex justify-between">
              <span className="text-sm text-muted-foreground">Data Freshness</span>
              <span className="text-sm font-medium text-success">Up to date</span>
            </div>
          </div>
        </Card>
      </div>
    </div>
  );
};

export default AdminAnalytics;
```

File: src/pages/AdminComments.tsx

Lines: 636

```
   1 | // Admin Comments Moderation Page
   2 | import { useState, useEffect } from "react";
   3 | import { Card } from "@/components/ui/card";
   4 | import { Button } from "@/components/ui/button";
   5 | import { Input } from "@/components/ui/input";
   6 | import { Badge } from "@/components/ui/badge";
   7 | import { Textarea } from "@/components/ui/textarea";
   8 | import {
   9 |   Table,
  10 |   TableBody,
  11 |   TableCell,
  12 |   TableHead,
  13 |   TableHeader,
  14 |   TableRow,
  15 | } from "@/components/ui/table";
  16 | import {
  17 |   AlertDialog,
  18 |   AlertDialogAction,
  19 |   AlertDialogCancel,
  20 |   AlertDialogContent,
  21 |   AlertDialogDescription,
  22 |   AlertDialogFooter,
  23 |   AlertDialogHeader,
  24 |   AlertDialogTitle,
  25 | } from "@/components/ui/alert-dialog";
  26 | import {
  27 |   Dialog,
  28 |   DialogContent,
  29 |   DialogDescription,
  30 |   DialogHeader,
  31 |   DialogTitle,
  32 |   DialogFooter,
  33 | } from "@/components/ui/dialog";
  34 | import { ref, get } from "firebase/database";
  35 | import { database } from "@/services/firebase";
  36 | import { Event, EventComment, deleteComment, getAllEventComments } from "@/services/
eventService";
  37 | import { getUserData } from "@/services/authService";
  38 | import { logAdminAction, sendAdminNotification, warnUser } from "@/services/adminService";
  39 | import { suspendCommenting, restoreCommenting, isCommentingSuspended, CommentingSuspension }
from "@/services/userServic
  40 | import { useAuth } from "@/hooks/useAuth";
  41 | import { toast } from "sonner";
  42 | import { useNavigate } from "react-router-dom";
  43 | import Avatar from "@mui/material/Avatar";
  44 | import SearchIcon from "@mui/icons-material/Search";
  45 | import DeleteIcon from "@mui/icons-material/Delete";
  46 | import WarningIcon from "@mui/icons-material/Warning";
  47 | import BlockIcon from "@mui/icons-material/Block";
  48 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
  49 | import EventIcon from "@mui/icons-material/Event";
  50 | import PersonIcon from "@mui/icons-material/Person";
  51 | import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
  52 | import RefreshIcon from "@mui/icons-material/Refresh";
  53 |
  54 | interface CommentWithContext extends EventComment {
  55 |   eventId: string;
  56 |   eventTitle: string;
  57 | }
  58 |
  59 | interface UserSuspensionStatus {
  60 |   [userId: string]: CommentingSuspension | null;
  61 | }
  62 |
  63 | const AdminComments = () => {
  64 |   const navigate = useNavigate();
  65 |   const { user: currentAdmin } = useAuth();
  66 |   const [comments, setComments] = useState<CommentWithContext[]>([]);
```

```
  67 |   const [filteredComments, setFilteredComments] = useState<CommentWithContext[]>([]);
  68 |   const [loading, setLoading] = useState(true);
  69 |   const [searchQuery, setSearchQuery] = useState("");
  70 |   const [selectedComment, setSelectedComment] = useState<CommentWithContext | null>(null);
  71 |   const [deleteDialog, setDeleteDialog] = useState(false);
  72 |   const [warnDialog, setWarnDialog] = useState(false);
  73 |   const [suspendDialog, setSuspendDialog] = useState(false);
  74 |   const [deleteReason, setDeleteReason] = useState("");
  75 |   const [warnReason, setWarnReason] = useState("");
  76 |   const [suspendReason, setSuspendReason] = useState("");
  77 |   const [suspendDuration, setSuspendDuration] = useState<number>(7); // days
  78 |   const [userData, setUserData] = useState<Record<string, any>>({});
  79 |   const [userSuspensions, setUserSuspensions] = useState<UserSuspensionStatus>({});
  80 |
  81 |   useEffect(() => {
  82 |     loadComments();
  83 |   }, []);
  84 |
  85 |   useEffect(() => {
  86 |     filterComments();
  87 |   }, [comments, searchQuery]);
  88 |
  89 |   const loadComments = async () => {
  90 |     try {
  91 |       setLoading(true);
  92 |
  93 |       // First get all events
  94 |       const eventsRef = ref(database, "events");
  95 |       const eventsSnapshot = await get(eventsRef);
  96 |
  97 |       if (!eventsSnapshot.exists()) {
  98 |         setComments([]);
  99 |         return;
 100 |       }
 101 |
 102 |       const allComments: CommentWithContext[] = [];
 103 |       const userIds = new Set<string>();
 104 |
 105 |       // Get comments from each event
 106 |       eventsSnapshot.forEach((eventChild) => {
 107 |         const eventData = eventChild.val() as Event;
 108 |         const eventId = eventChild.key!;
 109 |         const eventTitle = eventData.title || "Untitled Event";
 110 |
 111 |         if (eventData.comments) {
 112 |           Object.entries(eventData.comments).forEach(([commentId, commentData]: [string, any])
=> {
 113 |             allComments.push({
 114 |               ...commentData,
 115 |               id: commentId,
 116 |               eventId,
 117 |               eventTitle
 118 |             });
 119 |             userIds.add(commentData.userId);
 120 |           });
 121 |         }
 122 |         return false;
 123 |       });
 124 |
 125 |       // Sort by timestamp (newest first)
 126 |       allComments.sort((a, b) => (b.timestamp || 0) - (a.timestamp || 0));
 127 |       setComments(allComments);
 128 |
 129 |       // Load user data and suspension status
 130 |       const userDataMap: Record<string, any> = {};
 131 |       const suspensionMap: UserSuspensionStatus = {};
 132 |
 133 |       for (const userId of userIds) {
 134 |         try {
 135 |           const data = await getUserData(userId);
 136 |           if (data) {
 137 |             userDataMap[userId] = data;
```

```
          }
          const suspension = await isCommentingSuspended(userId);
          suspensionMap[userId] = suspension;
        } catch (error) {
          console.error(`Error loading user ${userId}:`, error);
        }
      }

      setUserData(userDataMap);
      setUserSuspensions(suspensionMap);
    } catch (error) {
      console.error("Error loading comments:", error);
      toast.error("Failed to load comments");
    } finally {
      setLoading(false);
    }
  };

  const filterComments = () => {
    let filtered = [...comments];

    if (searchQuery.trim()) {
      const query = searchQuery.toLowerCase();
      filtered = filtered.filter((comment) =>
        comment.text.toLowerCase().includes(query) ||
        comment.userName.toLowerCase().includes(query) ||
        comment.eventTitle.toLowerCase().includes(query)
      );
    }

    setFilteredComments(filtered);
  };

  const handleDeleteComment = async () => {
    if (!selectedComment || !currentAdmin?.uid) return;

    try {
      await deleteComment(selectedComment.eventId, selectedComment.id, currentAdmin.uid, true);

      // Log admin action
      await logAdminAction(currentAdmin.email || "admin", "delete_comment", {
        commentId: selectedComment.id,
        eventId: selectedComment.eventId,
        userId: selectedComment.userId,
        reason: deleteReason
      });

      // Send notification to user
      await sendAdminNotification({
        userId: selectedComment.userId,
        adminId: currentAdmin.uid,
        adminName: currentAdmin.displayName || "Admin",
        actionType: "comment_deleted",
        reason: deleteReason || "Violation of community guidelines",
        eventId: selectedComment.eventId,
        eventTitle: selectedComment.eventTitle
      });

      toast.success("Comment deleted successfully");
      setDeleteDialog(false);
      setSelectedComment(null);
      setDeleteReason("");
      loadComments();
    } catch (error: any) {
      console.error("Error deleting comment:", error);
      toast.error(error.message || "Failed to delete comment");
    }
  };

  const handleWarnUser = async () => {
    if (!selectedComment || !currentAdmin?.uid) return;
```

```
209 |
210 |     try {
211 |       await warnUser(
212 |         selectedComment.userId,
213 |         currentAdmin.uid,
214 |         currentAdmin.displayName || "Admin",
215 |         warnReason || "Inappropriate comment"
216 |       );
217 |
218 |       toast.success("Warning sent to user");
219 |       setWarnDialog(false);
220 |       setSelectedComment(null);
221 |       setWarnReason("");
222 |     } catch (error: any) {
223 |       console.error("Error warning user:", error);
224 |       toast.error(error.message || "Failed to send warning");
225 |     }
226 |   };
227 |
228 |   const handleSuspendCommenting = async () => {
229 |     if (!selectedComment || !currentAdmin?.uid) return;
230 |
231 |     try {
232 |       const suspendedUntil = Date.now() + (suspendDuration * 24 * 60 * 60 * 1000);
233 |
234 |       await suspendCommenting(
235 |         selectedComment.userId,
236 |         currentAdmin.uid,
237 |         suspendReason || "Repeated violations",
238 |         suspendedUntil
239 |       );
240 |
241 |       // Log admin action
242 |       await logAdminAction(currentAdmin.email || "admin", "suspend_commenting", {
243 |         userId: selectedComment.userId,
244 |         reason: suspendReason,
245 |         duration: suspendDuration
246 |       });
247 |
248 |       // Send notification to user
249 |       await sendAdminNotification({
250 |         userId: selectedComment.userId,
251 |         adminId: currentAdmin.uid,
252 |         adminName: currentAdmin.displayName || "Admin",
253 |         actionType: "comment_suspended",
254 |         reason: suspendReason || "Repeated violations"
255 |       });
256 |
257 |       toast.success(`User commenting suspended for ${suspendDuration} days`);
258 |       setSuspendDialog(false);
259 |       setSelectedComment(null);
260 |       setSuspendReason("");
261 |       setSuspendDuration(7);
262 |
263 |       // Update suspension status
264 |       setUserSuspensions((prev) => ({
265 |         ...prev,
266 |         [selectedComment.userId]: {
267 |           suspended: true,
268 |           suspendedAt: Date.now(),
269 |           suspendedUntil,
270 |           reason: suspendReason,
271 |           adminId: currentAdmin.uid
272 |         }
273 |       }));
274 |     } catch (error: any) {
275 |       console.error("Error suspending commenting:", error);
276 |       toast.error(error.message || "Failed to suspend commenting");
277 |     }
278 |   };
279 |
```

```
  const handleRestoreCommenting = async (userId: string) => {
    if (!currentAdmin?.uid) return;

    try {
      await restoreCommenting(userId);

      // Log admin action
      await logAdminAction(currentAdmin.email || "admin", "restore_commenting", {
        userId
      });

      // Send notification to user
      await sendAdminNotification({
        userId,
        adminId: currentAdmin.uid,
        adminName: currentAdmin.displayName || "Admin",
        actionType: "comment_restored",
        reason: "Privileges restored"
      });

      toast.success("User commenting privileges restored");

      // Update suspension status
      setUserSuspensions((prev) => ({
        ...prev,
        [userId]: null
      }));
    } catch (error: any) {
      console.error("Error restoring commenting:", error);
      toast.error(error.message || "Failed to restore commenting");
    }
  };

  const formatDate = (timestamp: number) => {
    return new Date(timestamp).toLocaleDateString(undefined, {
      year: "numeric",
      month: "short",
      day: "numeric",
      hour: "2-digit",
      minute: "2-digit"
    });
  };

  return (
    <div className="p-6 space-y-6">
      {/* Header */}
      <div className="flex flex-col md:flex-row md:items-center md:justify-between gap-4">
        <div>
          <h1 className="text-2xl font-bold">Comment Moderation</h1>
          <p className="text-muted-foreground">
            Review and moderate user comments across all events
          </p>
        </div>
        <div className="flex gap-2">
          <Button variant="outline" onClick={loadComments} disabled={loading}>
            <RefreshIcon className="mr-2" style={{ fontSize: 18 }} />
            Refresh
          </Button>
        </div>
      </div>

      {/* Stats Cards */}
      <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-primary/10 rounded-lg">
              <ChatBubbleOutlineIcon className="text-primary" />
            </div>
            <div>
              <p className="text-2xl font-bold">{comments.length}</p>
              <p className="text-sm text-muted-foreground">Total Comments</p>
```

```
            </div>
          </div>
        </Card>
        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-orange-500/10 rounded-lg">
              <BlockIcon className="text-orange-500" />
            </div>
            <div>
              <p className="text-2xl font-bold">
                {Object.values(userSuspensions).filter(s => s !== null).length}
              </p>
              <p className="text-sm text-muted-foreground">Suspended Users</p>
            </div>
          </div>
        </Card>
        <Card className="p-4">
          <div className="flex items-center gap-3">
            <div className="p-2 bg-green-500/10 rounded-lg">
              <EventIcon className="text-green-500" />
            </div>
            <div>
              <p className="text-2xl font-bold">
                {new Set(comments.map(c => c.eventId)).size}
              </p>
              <p className="text-sm text-muted-foreground">Events with Comments</p>
            </div>
          </div>
        </Card>
      </div>

      {/* Search */}
      <div className="flex gap-4">
        <div className="relative flex-1">
          <SearchIcon className="absolute left-3 top-1/2 -translate-y-1/2 text-muted-foreground"
style={{ fontSize: 20 }}...
          <Input
            placeholder="Search by comment text, user, or event..."
            value={searchQuery}
            onChange={(e) => setSearchQuery(e.target.value)}
            className="pl-10"
          />
        </div>
      </div>

      {/* Comments Table */}
      <Card>
        <Table>
          <TableHeader>
            <TableRow>
              <TableHead>User</TableHead>
              <TableHead>Comment</TableHead>
              <TableHead>Event</TableHead>
              <TableHead>Date</TableHead>
              <TableHead>Status</TableHead>
              <TableHead className="text-right">Actions</TableHead>
            </TableRow>
          </TableHeader>
          <TableBody>
            {loading ? (
              <TableRow>
                <TableCell colSpan={6} className="text-center py-8">
                  Loading comments...
                </TableCell>
              </TableRow>
            ) : filteredComments.length === 0 ? (
              <TableRow>
                <TableCell colSpan={6} className="text-center py-8 text-muted-foreground">
                  No comments found
                </TableCell>
              </TableRow>
            ) : (
```

```
422 |                filteredComments.map((comment) => (
423 |                  <TableRow key={`${comment.eventId}-${comment.id}`}>
424 |                    <TableCell>
425 |                      <div className="flex items-center gap-2">
426 |                        <Avatar
427 |                          src={comment.userAvatar}
428 |                          alt={comment.userName}
429 |                          sx={{ width: 32, height: 32 }}
430 |                        />
431 |                        <div>
432 |                          <p className="font-medium">{comment.userName}</p>
433 |                          <p className="text-xs text-muted-foreground">
434 |                            {userData[comment.userId]?.email || comment.userId.slice(0, 8)}
435 |                          </p>
436 |                        </div>
437 |                      </div>
438 |                    </TableCell>
439 |                    <TableCell>
440 |                      <p className="max-w-xs truncate">{comment.text}</p>
441 |                    </TableCell>
442 |                    <TableCell>
443 |                      <Badge variant="outline">{comment.eventTitle}</Badge>
444 |                    </TableCell>
445 |                    <TableCell>
446 |                      <p className="text-sm text-muted-foreground">
447 |                        {formatDate(comment.timestamp)}
448 |                      </p>
449 |                    </TableCell>
450 |                    <TableCell>
451 |                      {userSuspensions[comment.userId] ? (
452 |                        <Badge variant="destructive">Suspended</Badge>
453 |                      ) : (
454 |                        <Badge variant="secondary">Active</Badge>
455 |                      )}
456 |                    </TableCell>
457 |                    <TableCell className="text-right">
458 |                      <div className="flex justify-end gap-1">
459 |                        <Button
460 |                          variant="ghost"
461 |                          size="icon"
462 |                          onClick={() => {
463 |                            setSelectedComment(comment);
464 |                            setWarnDialog(true);
465 |                          }}
466 |                          title="Warn User"
467 |                        >
468 |                          <WarningIcon style={{ fontSize: 18 }} className="text-yellow-500" />
469 |                        </Button>
470 |                        <Button
471 |                          variant="ghost"
472 |                          size="icon"
473 |                          onClick={() => {
474 |                            setSelectedComment(comment);
475 |                            setDeleteDialog(true);
476 |                          }}
477 |                          title="Delete Comment"
478 |                        >
479 |                          <DeleteIcon style={{ fontSize: 18 }} className="text-destructive" />
480 |                        </Button>
481 |                        {userSuspensions[comment.userId] ? (
482 |                          <Button
483 |                            variant="ghost"
484 |                            size="icon"
485 |                            onClick={() => handleRestoreCommenting(comment.userId)}
486 |                            title="Restore Commenting"
487 |                          >
488 |                            <CheckCircleIcon style={{ fontSize: 18 }} className="text-green-500" />
489 |                          </Button>
490 |                        ) : (
491 |                          <Button
492 |                            variant="ghost"
```

```
                          size="icon"
                          onClick={() => {
                            setSelectedComment(comment);
                            setSuspendDialog(true);
                          }}
                          title="Suspend Commenting"
                        >
                          <BlockIcon style={{ fontSize: 18 }} className="text-orange-500" />
                        </Button>
                      )}
                    </div>
                  </TableCell>
                </TableRow>
              ))
            )}
          </TableBody>
        </Table>
      </Card>

      {/* Delete Comment Dialog */}
      <AlertDialog open={deleteDialog} onOpenChange={setDeleteDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Delete Comment</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to delete this comment? The user will be notified.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <div className="py-4">
            <p className="text-sm font-medium mb-2">Comment:</p>
            <p className="text-sm text-muted-foreground bg-muted p-3 rounded-lg">
              "{selectedComment?.text}"
            </p>
            <div className="mt-4">
              <label className="text-sm font-medium">Reason for deletion:</label>
              <Textarea
                placeholder="Enter reason for deletion..."
                value={deleteReason}
                onChange={(e) => setDeleteReason(e.target.value)}
                className="mt-2"
              />
            </div>
          </div>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setDeleteReason("")}>Cancel</AlertDialogCancel>
            <AlertDialogAction onClick={handleDeleteComment} className="bg-destructive text-
destructive-foreground">
              Delete Comment
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Warn User Dialog */}
      <Dialog open={warnDialog} onOpenChange={setWarnDialog}>
        <DialogContent>
          <DialogHeader>
            <DialogTitle>Warn User</DialogTitle>
            <DialogDescription>
              Send a warning to {selectedComment?.userName} about their comment.
            </DialogDescription>
          </DialogHeader>
          <div className="py-4">
            <p className="text-sm font-medium mb-2">Comment:</p>
            <p className="text-sm text-muted-foreground bg-muted p-3 rounded-lg">
              "{selectedComment?.text}"
            </p>
            <div className="mt-4">
              <label className="text-sm font-medium">Warning message:</label>
              <Textarea
                placeholder="Enter warning message..."
                value={warnReason}
```

```
                onChange={(e) => setWarnReason(e.target.value)}
                className="mt-2"
              />
            </div>
          </div>
          <DialogFooter>
            <Button variant="outline" onClick={() => {
              setWarnDialog(false);
              setWarnReason("");
            }}>
              Cancel
            </Button>
            <Button onClick={handleWarnUser} className="bg-yellow-500 hover:bg-yellow-600">
              <WarningIcon className="mr-2" style={{ fontSize: 18 }} />
              Send Warning
            </Button>
          </DialogFooter>
        </DialogContent>
      </Dialog>

      {/* Suspend Commenting Dialog */}
      <Dialog open={suspendDialog} onOpenChange={setSuspendDialog}>
        <DialogContent>
          <DialogHeader>
            <DialogTitle>Suspend Commenting Privileges</DialogTitle>
            <DialogDescription>
              Suspend {selectedComment?.userName}'s ability to comment on events.
            </DialogDescription>
          </DialogHeader>
          <div className="py-4 space-y-4">
            <div>
              <label className="text-sm font-medium">Suspension duration (days):</label>
              <Input
                type="number"
                min={1}
                max={365}
                value={suspendDuration}
                onChange={(e) => setSuspendDuration(Number(e.target.value))}
                className="mt-2"
              />
            </div>
            <div>
              <label className="text-sm font-medium">Reason for suspension:</label>
              <Textarea
                placeholder="Enter reason for suspension..."
                value={suspendReason}
                onChange={(e) => setSuspendReason(e.target.value)}
                className="mt-2"
              />
            </div>
          </div>
          <DialogFooter>
            <Button variant="outline" onClick={() => {
              setSuspendDialog(false);
              setSuspendReason("");
              setSuspendDuration(7);
            }}>
              Cancel
            </Button>
            <Button onClick={handleSuspendCommenting} className="bg-orange-500 hover:bg-
orange-600">
              <BlockIcon className="mr-2" style={{ fontSize: 18 }} />
              Suspend Commenting
            </Button>
          </DialogFooter>
        </DialogContent>
      </Dialog>
    </div>
  );
};

export default AdminComments;
```

635 |
636 |

File: src/pages/AdminDashboard.tsx

Lines: 326

```tsx
// Admin Dashboard - Main admin interface
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
import { useAuth } from "@/hooks/useAuth";
import { useAdmin } from "@/hooks/useAdmin";
import { signOut } from "@/services/authService";
import { getSystemStats } from "@/services/adminService";
import DashboardIcon from "@mui/icons-material/Dashboard";
import PeopleIcon from "@mui/icons-material/People";
import BarChartIcon from "@mui/icons-material/BarChart";
import ReportProblemIcon from "@mui/icons-material/ReportProblem";
import EventIcon from "@mui/icons-material/Event";
import SettingsIcon from "@mui/icons-material/Settings";
import LogoutIcon from "@mui/icons-material/Logout";
import AdminPanelSettingsIcon from "@mui/icons-material/AdminPanelSettings";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
import { toast } from "sonner";

const AdminDashboard = () => {
  const navigate = useNavigate();
  const { user } = useAuth();
  const { isAdmin, loading } = useAdmin();
  const [stats, setStats] = useState<any>(null);
  const [loadingStats, setLoadingStats] = useState(true);
  const [sidebarOpen, setSidebarOpen] = useState(true);

  useEffect(() => {
    if (!loading && !isAdmin) {
      navigate("/admin/login", { replace: true });
    }
  }, [isAdmin, loading, navigate]);

  useEffect(() => {
    const loadStats = async () => {
      try {
        const { getSystemStats } = await import("@/services/adminService");
        const statsData = await getSystemStats();
        setStats(statsData);
      } catch (error) {
        console.error("Error loading stats:", error);
        toast.error("Failed to load statistics");
      } finally {
        setLoadingStats(false);
      }
    };

    if (isAdmin) {
      loadStats();
    }
  }, [isAdmin]);

  const handleLogout = async () => {
    try {
      await signOut();
      navigate("/admin/login", { replace: true });
      toast.success("Logged out successfully");
    } catch (error) {
      console.error("Error signing out:", error);
      toast.error("Failed to sign out");
    }
```

```
  67 |   };
  68 |
  69 |   const menuItems = [
  70 |     { id: "dashboard", label: "Overview", icon: DashboardIcon, path: "/admin/dashboard" },
  71 |     { id: "users", label: "User Management", icon: PeopleIcon, path: "/admin/users" },
  72 |     { id: "analytics", label: "Analytics", icon: BarChartIcon, path: "/admin/analytics" },
  73 |     { id: "moderation", label: "Content Moderation", icon: ReportProblemIcon, path: "/admin/
moderation"},
  74 |     { id: "comments", label: "Comment Moderation", icon: ChatBubbleOutlineIcon, path: "/admin/
comments" },
  75 |     { id: "events", label: "Events", icon: EventIcon, path: "/admin/events" },
  76 |     { id: "venues", label: "Venues", icon: LocationOnIcon, path: "/admin/venues" },
  77 |     { id: "settings", label: "System Settings", icon: SettingsIcon, path: "/admin/settings" },
  78 |   ];
  79 |
  80 |   if (loading) {
  81 |     return (
  82 |       <div className="min-h-screen flex items-center justify-center">
  83 |         <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
  84 |       </div>
  85 |     );
  86 |   }
  87 |
  88 |   if (!isAdmin) {
  89 |     return null;
  90 |   }
  91 |
  92 |   const currentPath = window.location.pathname;
  93 |
  94 |   return (
  95 |     <div className="min-h-screen bg-background flex">
  96 |       {/* Sidebar */}
  97 |       <motion.div
  98 |         initial={{ x: -300 }}
  99 |         animate={{ x: sidebarOpen ? 0 : -300 }}
 100 |         transition={{ duration: 0.3 }}
 101 |         className={`fixed lg:static inset-y-0 left-0 z-50 w-64 bg-card border-r border-border
flex flex-col ${
 102 |           sidebarOpen ? "translate-x-0" : "-translate-x-full"
 103 |         } lg:translate-x-0`}
 104 |       >
 105 |         {/* Sidebar Header */}
 106 |         <div className="p-6 border-b border-border">
 107 |           <div className="flex items-center gap-3">
 108 |             <div className="p-2 bg-destructive/10 rounded-lg">
 109 |               <AdminPanelSettingsIcon className="text-destructive" style={{ fontSize: 24 }} />
 110 |             </div>
 111 |             <div>
 112 |               <h1 className="text-lg font-bold">Admin Portal</h1>
 113 |               <p className="text-xs text-muted-foreground">PaceMatch</p>
 114 |             </div>
 115 |           </div>
 116 |         </div>
 117 |
 118 |         {/* Navigation */}
 119 |         <nav className="flex-1 p-4 space-y-2 overflow-y-auto">
 120 |           {menuItems.map((item) => {
 121 |             const Icon = item.icon;
 122 |             const isActive = currentPath === item.path || (item.path !== "/admin/dashboard" &&
currentPath.startsWith(it...
 123 |
 124 |             return (
 125 |               <Button
 126 |                 key={item.id}
 127 |                 variant={isActive ? "default" : "ghost"}
 128 |                 className="w-full justify-start"
 129 |                 onClick={() => navigate(item.path)}
 130 |               >
 131 |                 <Icon className="mr-3" style={{ fontSize: 20 }} />
 132 |                 {item.label}
 133 |               </Button>
 134 |             );
 135 |           })}
 136 |         </nav>
 137 |
```

```
        {/* User Info & Logout */}
        <div className="p-4 border-t border-border space-y-4">
          <div className="flex items-center gap-3 p-3 bg-muted/50 rounded-lg">
            <Avatar className="h-10 w-10">
              <AvatarImage src={user?.photoURL || undefined} />
              <AvatarFallback>
                {user?.displayName?.[0]?.toUpperCase() || user?.email?.[0]?.toUpperCase() || "A"}
              </AvatarFallback>
            </Avatar>
            <div className="flex-1 min-w-0">
              <p className="text-sm font-medium truncate">{user?.displayName || "Admin"}</p>
              <p className="text-xs text-muted-foreground truncate">{user?.email}</p>
            </div>
          </div>
          <Button
            variant="outline"
            className="w-full"
            onClick={handleLogout}
          >
            <LogoutIcon className="mr-2" style={{ fontSize: 18 }} />
            Logout
          </Button>
        </div>
      </motion.div>

      {/* Main Content */}
      <div className="flex-1 flex flex-col lg:ml-0">
        {/* Top Bar */}
        <div classNam="sticky top-0 z-40 bg-background/95 backdrop-blur supports-[backdrop-filter]:bg-background/60 bor
          <div className="flex items-center justify-between p-4">
            <div className="flex items-center gap-4">
              <Button
                variant="ghost"
                size="icon"
                className="lg:hidden"
                onClick={() => setSidebarOpen(!sidebarOpen)}
              >
                <svg
                  xmlns="http://www.w3.org/2000/svg"
                  width="24"
                  height="24"
                  viewBox="0 0 24 24"
                  fill="none"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                >
                  <line x1="3" y1="6" x2="21" y2="6"></line>
                  <line x1="3" y1="12" x2="21" y2="12"></line>
                  <line x1="3" y1="18" x2="21" y2="18"></line>
                </svg>
              </Button>
              <h2 className="text-2xl font-bold">
                {menuItems.find(item => currentPath === item.path || (item.path !== "/admin/dashboard" && currentPath.st...
              </h2>
            </div>
          </div>
        </div>

        {/* Dashboard Content */}
        <div className="flex-1 p-6 overflow-y-auto">
          {currentPath === "/admin/dashboard" && (
            <div className="space-y-6">
              {/* Stats Cards */}
              <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
                <Card className="p-6">
                  <div className="flex items-center justify-between">
                    <div>
                      <p className="text-sm font-medium text-muted-foreground">Total Users</p>
                      <p className="text-3xl font-bold mt-2">
```

```
                        {loadingStats ? (
                          <span className="text-muted-foreground">...</span>
                        ) : (
                          stats?.totalUsers || 0
                        )}
                      </p>
                    </div>
                    <div className="p-3 bg-primary/10 rounded-lg">
                      <PeopleIcon className="text-primary" style={{ fontSize: 28 }} />
                    </div>
                  </div>
                </Card>

                <Card className="p-6">
                  <div className="flex items-center justify-between">
                    <div>
                      <p className="text-sm font-medium text-muted-foreground">Active Users
(30d)</p>
                      <p className="text-3xl font-bold mt-2">
                        {loadingStats ? (
                          <span className="text-muted-foreground">...</span>
                        ) : (
                          stats?.activeUsers || 0
                        )}
                      </p>
                    </div>
                    <div className="p-3 bg-success/10 rounded-lg">
                      <PeopleIcon className="text-success" style={{ fontSize: 28 }} />
                    </div>
                  </div>
                </Card>

                <Card className="p-6">
                  <div className="flex items-center justify-between">
                    <div>
                      <p className="text-sm font-medium text-muted-foreground">Total Workouts</p>
                      <p className="text-3xl font-bold mt-2">
                        {loadingStats ? (
                          <span className="text-muted-foreground">...</span>
                        ) : (
                          stats?.totalWorkouts || 0
                        )}
                      </p>
                    </div>
                    <div className="p-3 bg-warning/10 rounded-lg">
                      <BarChartIcon className="text-warning" style={{ fontSize: 28 }} />
                    </div>
                  </div>
                </Card>

                <Card className="p-6">
                  <div className="flex items-center justify-between">
                    <div>
                      <p className="text-sm font-medium text-muted-foreground">Pending Reports</
p>
                      <p className="text-3xl font-bold mt-2">
                        {loadingStats ? (
                          <span className="text-muted-foreground">...</span>
                        ) : (
                          stats?.pendingReports || 0
                        )}
                      </p>
                    </div>
                    <div className="p-3 bg-destructive/10 rounded-lg">
                      <ReportProblemIcon className="text-destructive" style={{ fontSize: 28 }} />
                    </div>
                  </div>
                </Card>
              </div>

              {/* Quick Actions */}
              <Card className="p-6">
                <h3 className="text-lg font-semibold mb-4">Quick Actions</h3>
```

```
                <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                  <Button
                    variant="outline"
                    className="h-auto p-4 flex flex-col items-start"
                    onClick={() => navigate("/admin/users")}
                  >
                    <PeopleIcon className="mb-2" style={{ fontSize: 24 }} />
                    <span className="font-medium">Manage Users</span>
                    <span className="text-xs text-muted-foreground mt-1">View and manage all
users</span>
                  </Button>
                  <Button
                    variant="outline"
                    className="h-auto p-4 flex flex-col items-start"
                    onClick={() => navigate("/admin/moderation")}
                  >
                    <ReportProblemIcon className="mb-2" style={{ fontSize: 24 }} />
                    <span className="font-medium">Review Reports</span>
                    <span className="text-xs text-muted-foreground mt-1">Moderate content and
reports</span>
                  </Button>
                  <Button
                    variant="outline"
                    className="h-auto p-4 flex flex-col items-start"
                    onClick={() => navigate("/admin/settings")}
                  >
                    <SettingsIcon className="mb-2" style={{ fontSize: 24 }} />
                    <span className="font-medium">System Settings</span>
                    <span className="text-xs text-muted-foreground mt-1">Configure app settings</
span>
                  </Button>
                </div>
              </Card>

              {/* Recent Activity Placeholder */}
              <Card className="p-6">
                <h3 className="text-lg font-semibold mb-4">Recent Activity</h3>
                <p className="text-sm text-muted-foreground">Activity feed coming soon...</p>
              </Card>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

export default AdminDashboard;

```

File: src/pages/AdminEvents.tsx

Lines: 785

```
// Admin Events Management Page
import { useState, useEffect } from "react";
import { Card } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@/components/ui/table";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import { ref, get, remove, set } from "firebase/database";
import { database } from "@/services/firebase";
import { Event } from "@/services/eventService";
import { getUserData } from "@/services/authService";
import { logAdminAction } from "@/services/adminService";
import { useAuth } from "@/hooks/useAuth";
import { toast } from "sonner";
import { useNavigate } from "react-router-dom";
import SearchIcon from "@mui/icons-material/Search";
import DeleteIcon from "@mui/icons-material/Delete";
import EventIcon from "@mui/icons-material/Event";
import PeopleIcon from "@mui/icons-material/People";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import CancelIcon from "@mui/icons-material/Cancel";
import VisibilityIcon from "@mui/icons-material/Visibility";
import ChatBubbleOutlineIcon from "@mui/icons-material/ChatBubbleOutline";
import DeleteSweepIcon from "@mui/icons-material/DeleteSweep";
import WarningIcon from "@mui/icons-material/Warning";
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";

const AdminEvents = () => {
  const navigate = useNavigate();
  const { user: currentAdmin } = useAuth();
  const [events, setEvents] = useState<Event[]>([]);
  const [filteredEvents, setFilteredEvents] = useState<Event[]>([]);
  const [loading, setLoading] = useState(true);
  const [searchQuery, setSearchQuery] = useState("");
  const [typeFilter, setTypeFilter] = useState<"all" | "running" | "cycling" | "walking" |
"others">("all");
  const [categoryFilter, setCategoryFilter] = useState<"all" | "user" | "sponsored">("all");
  const [statusFilter, setStatusFilter] = useState<"all" | "active" | "expired">("all");
  const [selectedEvent, setSelectedEvent] = useState<Event | null>(null);
  const [deleteDialog, setDeleteDialog] = useState(false);
  const [cancelDialog, setCancelDialog] = useState(false);
  const [participantsDialog, setParticipantsDialog] = useState(false);
  const [deleteExpiredDialog, setDeleteExpiredDialog] = useState(false);
  const [deletingExpired, setDeletingExpired] = useState(false);
```

```
 67 |   const [userData, setUserData] = useState<Record<string, any>>({});
 68 |   const [participantsData, setParticipantsData] = useState<Record<string, any>>({});
 69 |
 70 |   useEffect(() => {
 71 |     loadEvents();
 72 |   }, []);
 73 |
 74 |   useEffect(() => {
 75 |     filterEvents();
 76 |   }, [events, searchQuery, typeFilter, categoryFilter, statusFilter]);
 77 |
 78 |   const loadEvents = async () => {
 79 |     try {
 80 |       setLoading(true);
 81 |       const eventsRef = ref(database, "events");
 82 |       const snapshot = await get(eventsRef);
 83 |
 84 |       if (!snapshot.exists()) {
 85 |         setEvents([]);
 86 |         return;
 87 |       }
 88 |
 89 |       const eventsList: Event[] = [];
 90 |       snapshot.forEach((child) => {
 91 |         eventsList.push({
 92 |           id: child.key!,
 93 |           ...child.val()
 94 |         });
 95 |         return false;
 96 |       });
 97 |
 98 |       // Sort by creation date (newest first)
 99 |       eventsList.sort((a, b) => (b.createdAt || 0) - (a.createdAt || 0));
100 |       setEvents(eventsList);
101 |
102 |       // Load user data for hosts
103 |       const userIds = new Set<string>();
104 |       eventsList.forEach((event) => {
105 |         if (event.hostId) {
106 |           userIds.add(event.hostId);
107 |         }
108 |       });
109 |
110 |       const userDataMap: Record<string, any> = {};
111 |       for (const userId of userIds) {
112 |         try {
113 |           const data = await getUserData(userId);
114 |           if (data) {
115 |             userDataMap[userId] = data;
116 |           }
117 |         } catch (error) {
118 |           console.error(`Error loading user ${userId}:`, error);
119 |         }
120 |       }
121 |       setUserData(userDataMap);
122 |     } catch (error) {
123 |       console.error("Error loading events:", error);
124 |       toast.error("Failed to load events");
125 |     } finally {
126 |       setLoading(false);
127 |     }
128 |   };
129 |
130 |   /**
131 |    * Check if an event is expired (event date/time has passed)
132 |    * An event is considered expired if the current time is past the event's date and time
133 |    */
134 |   const isEventExpired = (event: Event): boolean => {
135 |     if (!event.date || !event.time) {
136 |       return false; // Can't determine if expired without date/time
137 |     }
```

```
138 |
139 |     try {
140 |       // Parse the event date and time
141 |       let eventDateTime: Date;
142 |
143 |       // Handle different date formats
144 |       if (event.date.includes('T')) {
145 |         // Already includes time
146 |         eventDateTime = new Date(event.date);
147 |       } else if (event.date.match(/^\d{4}-\d{2}-\d{2}$/)) {
148 |         // ISO date format (YYYY-MM-DD) - combine with time
149 |         eventDateTime = new Date(`${event.date}T${event.time}`);
150 |       } else if (event.date.includes('/')) {
151 |         // Format like "12/25/2024" or "MM/DD/YYYY"
152 |         const parts = event.date.split('/');
153 |         if (parts.length === 3) {
154 |           const firstPart = parseInt(parts[0]);
155 |           if (firstPart > 12) {
156 |             // DD/MM/YYYY format
157 |             eventDateTime = new Date(`${parts[2]}-${parts[1]}-${parts[0]}T${event.time}`);
158 |           } else {
159 |             // MM/DD/YYYY format
160 |             eventDateTime = new Date(`${parts[2]}-${parts[0]}-${parts[1]}T${event.time}`);
161 |           }
162 |         } else {
163 |           eventDateTime = new Date(`${event.date}T${event.time}`);
164 |         }
165 |       } else {
166 |         // Try parsing as-is or with time appended
167 |         eventDateTime = new Date(`${event.date}T${event.time}`);
168 |       }
169 |
170 |       // Validate the date
171 |       if (isNaN(eventDateTime.getTime())) {
172 |         return false; // Invalid date, don't consider expired
173 |       }
174 |
175 |       // Check if current time is past the event time
176 |       const now = new Date();
177 |       return now > eventDateTime;
178 |     } catch (error) {
179 |       console.error("Error checking if event is expired:", error);
180 |       return false;
181 |     }
182 |   };
183 |
184 |   /**
185 |    * Get all expired events
186 |    */
187 |   const getExpiredEvents = (): Event[] => {
188 |     return events.filter(event => isEventExpired(event));
189 |   };
190 |
191 |   const filterEvents = () => {
192 |     let filtered = [...events];
193 |
194 |     // Filter by type
195 |     if (typeFilter !== "all") {
196 |       filtered = filtered.filter(event => event.type === typeFilter);
197 |     }
198 |
199 |     // Filter by category
200 |     if (categoryFilter !== "all") {
201 |       filtered = filtered.filter(event => event.category === categoryFilter);
202 |     }
203 |
204 |     // Filter by status (active/expired)
205 |     if (statusFilter === "active") {
206 |       filtered = filtered.filter(event => !isEventExpired(event));
207 |     } else if (statusFilter === "expired") {
208 |       filtered = filtered.filter(event => isEventExpired(event));
```

```
    }

    // Filter by search query
    if (searchQuery) {
      const query = searchQuery.toLowerCase();
      filtered = filtered.filter(event =>
        event.title?.toLowerCase().includes(query) ||
        event.description?.toLowerCase().includes(query) ||
        event.location?.toLowerCase().includes(query) ||
        event.hostName?.toLowerCase().includes(query)
      );
    }

    setFilteredEvents(filtered);
  };

  const handleDelete = async (event: Event) => {
    try {
      const eventRef = ref(database, `events/${event.id}`);
      await remove(eventRef);
      if (currentAdmin?.email) {
        await logAdminAction(currentAdmin.email, "delete_event", {
          eventId: event.id,
          eventTitle: event.title,
          hostId: event.hostId,
          participantsCount: event.participants?.length || 0
        });
      }
      toast.success(`Event "${event.title}" has been deleted`);
      await loadEvents();
      setDeleteDialog(false);
      setSelectedEvent(null);
    } catch (error) {
      console.error("Error deleting event:", error);
      toast.error("Failed to delete event");
    }
  };

  const handleCancel = async (event: Event) => {
    try {
      const eventRef = ref(database, `events/${event.id}`);
      const eventData = await get(eventRef);
      if (eventData.exists()) {
        await set(eventRef, {
          ...eventData.val(),
          cancelled: true,
          cancelledAt: Date.now()
        });
        if (currentAdmin?.email) {
          await logAdminAction(currentAdmin.email, "cancel_event", {
            eventId: event.id,
            eventTitle: event.title,
            hostId: event.hostId,
            participantsCount: event.participants?.length || 0
          });
        }
        toast.success(`Event "${event.title}" has been cancelled`);
        await loadEvents();
        setCancelDialog(false);
        setSelectedEvent(null);
      }
    } catch (error) {
      console.error("Error cancelling event:", error);
      toast.error("Failed to cancel event");
    }
  };

  /**
   * Delete all expired events
   * This function finds all expired events and deletes them from the database
   */
```

```
  const handleDeleteExpiredEvents = async () => {
    try {
      setDeletingExpired(true);
      const expiredEvents = getExpiredEvents();

      if (expiredEvents.length === 0) {
        toast.info("No expired events to delete");
        setDeleteExpiredDialog(false);
        return;
      }

      // Delete each expired event
      const deletePromises = expiredEvents.map(async (event) => {
        const eventRef = ref(database, `events/${event.id}`);
        await remove(eventRef);

        // Log admin action for each deletion
        if (currentAdmin?.email) {
          await logAdminAction(currentAdmin.email, "delete_expired_event", {
            eventId: event.id,
            eventTitle: event.title,
            hostId: event.hostId,
            participantsCount: event.participants?.length || 0,
            eventDate: event.date,
            eventTime: event.time
          });
        }
      });

      await Promise.all(deletePromises);

      toast.success(`Successfully deleted ${expiredEvents.length} expired event(s)`);
      await loadEvents();
      setDeleteExpiredDialog(false);
    } catch (error) {
      console.error("Error deleting expired events:", error);
      toast.error("Failed to delete expired events");
    } finally {
      setDeletingExpired(false);
    }
  };

  const handleViewParticipants = async (event: Event) => {
    setSelectedEvent(event);
    try {
      // Load participant data
      const participantIds = event.participants || [];
      const participantDataMap: Record<string, any> = {};

      for (const userId of participantIds) {
        try {
          const data = await getUserData(userId);
          if (data) {
            participantDataMap[userId] = data;
          }
        } catch (error) {
          console.error(`Error loading participant ${userId}:`, error);
        }
      }

      setParticipantsData(participantDataMap);
      setParticipantsDialog(true);
    } catch (error) {
      console.error("Error loading participants:", error);
      toast.error("Failed to load participants");
    }
  };

  const formatDate = (dateString: string) => {
    try {
      return new Date(dateString).toLocaleDateString();
```

```
351 |     } catch {
352 |       return dateString;
353 |     }
354 |   };
355 |
356 |   const formatDateTime = (dateString: string, timeString: string) => {
357 |     try {
358 |       const date = new Date(`${dateString}T${timeString}`);
359 |       return date.toLocaleString();
360 |     } catch {
361 |       return `${dateString} ${timeString}`;
362 |     }
363 |   };
364 |
365 |   const getTypeBadge = (type: string) => {
366 |     const colors: Record<string, any> = {
367 |       running: "success",
368 |       cycling: "info",
369 |       walking: "warning",
370 |       others: "default"
371 |     };
372 |     return <Badge variant={colors[type] || "default"}>{type}</Badge>;
373 |   };
374 |
375 |   if (loading) {
376 |     return (
377 |       <div className="flex items-center justify-center h-64">
378 |         <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
379 |       </div>
380 |     );
381 |   }
382 |
383 |   return (
384 |     <div className="space-y-6">
385 |       {/* Header */}
386 |       <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
387 |         <div>
388 |           <div className="flex items-center gap-2 mb-2">
389 |             <Button variant="ghost" size="sm" onClick={() => navigate("/admin/dashboard")}>
390 |               !• Back to Dashboard
391 |             </Button>
392 |           </div>
393 |           <h1 className="text-3xl font-bold">Events Management</h1>
394 |           <p className="text-muted-foreground mt-1">
395 |             View and manage all events
396 |           </p>
397 |         </div>
398 |         <div className="flex items-center gap-4">
399 |           <div className="text-sm text-muted-foreground">
400 |             {filteredEvents.length} events
401 |             {statusFilter === "expired" && ` (${getExpiredEvents().length} expired)`}
402 |           </div>
403 |           {getExpiredEvents().length > 0 && (
404 |             <Button
405 |               variant="destructive"
406 |               size="sm"
407 |               onClick={() => setDeleteExpiredDialog(true)}
408 |               className="gap-2"
409 |             >
410 |               <DeleteSweepIcon style={{ fontSize: 18 }} />
411 |               Delete All Expired ({getExpiredEvents().length})
412 |             </Button>
413 |           )}
414 |         </div>
415 |       </div>
416 |
417 |       {/* Filters */}
418 |       <Card className="p-4">
419 |         <div className="flex flex-col gap-4">
420 |           <div className="relative">
421 |             <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-
foreground" style={{ fo...
```

```
              <Input
                placeholder="Search events..."
                value={searchQuery}
                onChange={(e) => setSearchQuery(e.target.value)}
                className="pl-10"
              />
            </div>
            <div className="flex flex-wrap gap-2">
              <div className="flex gap-2">
                <span className="text-sm text-muted-foreground self-center mr-2">Type:</span>
                <Button
                  variant={typeFilter === "all" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setTypeFilter("all")}
                >
                  All
                </Button>
                <Button
                  variant={typeFilter === "running" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setTypeFilter("running")}
                >
                  Running
                </Button>
                <Button
                  variant={typeFilter === "cycling" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setTypeFilter("cycling")}
                >
                  Cycling
                </Button>
                <Button
                  variant={typeFilter === "walking" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setTypeFilter("walking")}
                >
                  Walking
                </Button>
                <Button
                  variant={typeFilter === "others" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setTypeFilter("others")}
                >
                  Others
                </Button>
              </div>
              <div className="flex gap-2">
                <span className="text-sm text-muted-foreground self-center mr-2">Category:</span>
                <Button
                  variant={categoryFilter === "all" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setCategoryFilter("all")}
                >
                  All
                </Button>
                <Button
                  variant={categoryFilter === "user" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setCategoryFilter("user")}
                >
                  User Created
                </Button>
                <Button
                  variant={categoryFilter === "sponsored" ? "default" : "outline"}
                  size="sm"
                  onClick={() => setCategoryFilter("sponsored")}
                >
                  Sponsored
                </Button>
              </div>
              <div className="flex gap-2">
```

```
 493 |               <span className="text-sm text-muted-foreground self-center mr-2">Status:</span>
 494 |               <Button
 495 |                 variant={statusFilter === "all" ? "default" : "outline"}
 496 |                 size="sm"
 497 |                 onClick={() => setStatusFilter("all")}
 498 |               >
 499 |                 All
 500 |               </Button>
 501 |               <Button
 502 |                 variant={statusFilter === "active" ? "default" : "outline"}
 503 |                 size="sm"
 504 |                 onClick={() => setStatusFilter("active")}
 505 |               >
 506 |                 Active
 507 |               </Button>
 508 |               <Button
 509 |                 variant={statusFilter === "expired" ? "default" : "outline"}
 510 |                 size="sm"
 511 |                 onClick={() => setStatusFilter("expired")}
 512 |                 className="gap-1"
 513 |               >
 514 |                 <WarningIcon style={{ fontSize: 16 }} />
 515 |                 Expired ({getExpiredEvents().length})
 516 |               </Button>
 517 |             </div>
 518 |           </div>
 519 |         </div>
 520 |       </Card>
 521 |
 522 |       {/* Events Table */}
 523 |       <Card>
 524 |         <div className="overflow-x-auto">
 525 |           <Table>
 526 |             <TableHeader>
 527 |               <TableRow>
 528 |                 <TableHead>Event</TableHead>
 529 |                 <TableHead>Type</TableHead>
 530 |                 <TableHead>Host</TableHead>
 531 |                 <TableHead>Date & Time</TableHead>
 532 |                 <TableHead>Location</TableHead>
 533 |                 <TableHead>Participants</TableHead>
 534 |                 <TableHead className="text-right">Actions</TableHead>
 535 |               </TableRow>
 536 |             </TableHeader>
 537 |             <TableBody>
 538 |               {filteredEvents.length === 0 ? (
 539 |                 <TableRow>
 540 |                   <TableCell colSpan={7} className="text-center py-8 text-muted-foreground">
 541 |                     No events found
 542 |                   </TableCell>
 543 |                 </TableRow>
 544 |               ) : (
 545 |                 filteredEvents.map((event) => {
 546 |                   const expired = isEventExpired(event);
 547 |                   return (
 548 |                   <TableRow key={event.id} className={expired ? "opacity-60 bg-muted/30" : ""}>
 549 |                     <TableCell>
 550 |                       <div>
 551 |                         <div className="flex items-center gap-2">
 552 |                           <p className="font-medium">{event.title}</p>
 553 |                           {expired && (
 554 |                             <Badge variant="destructive" className="text-xs">
 555 |                               <WarningIcon style={{ fontSize: 12 }} className="mr-1" />
 556 |                               Expired
 557 |                             </Badge>
 558 |                           )}
 559 |                         </div>
 560 |                         <p className="text-xs text-muted-foreground line-clamp-2">
 561 |                           {event.description}
 562 |                         </p>
 563 |                         <Badge variant={event.category === "sponsored" ? "default" :
"secondary"} className="mt-1">
```

```
                        {event.category}
                      </Badge>
                    </div>
                  </TableCell>
                  <TableCell>{getTypeBadge(event.type)}</TableCell>
                  <TableCell>
                    {event.hostName || (event.hostId && userData[event.hostId]?.name) || "N/A"}
                  </TableCell>
                  <TableCell>
                    <div className="text-sm">
                      <div>{formatDate(event.date)}</div>
                      <div className="text-xs text-muted-foreground">{event.time}</div>
                    </div>
                  </TableCell>
                  <TableCell>
                    <div className="flex items-center gap-1 text-sm">
                      <LocationOnIcon style={{ fontSize: 16 }} />
                      <span className="max-w-xs truncate">{event.location}</span>
                    </div>
                  </TableCell>
                  <TableCell>
                    <div className="flex items-center gap-1">
                      <PeopleIcon style={{ fontSize: 16 }} />
                      <span>
                        {event.participants?.length || 0}
                        {event.maxParticipants && ` / ${event.maxParticipants}`}
                      </span>
                    </div>
                  </TableCell>
                  <TableCell className="text-right">
                    <div className="flex justify-end gap-2">
                      <Button
                        variant="outline"
                        size="sm"
                        onClick={() => handleViewParticipants(event)}
                      >
                        <VisibilityIcon className="mr-1" style={{ fontSize: 16 }} />
                        Participants
                      </Button>
                      <Button
                        variant="outline"
                        size="sm"
                        onClick={() => navigate(`/admin/comments?eventId=${event.id}`)}
                      >
                        <ChatBubbleOutlineIcon className="mr-1" style={{ fontSize: 16 }} />
                        Comments
                        {event.comments && Object.keys(event.comments).length > 0 && (
                          <Badge variant="secondary" className="ml-1">
                            {Object.keys(event.comments).length}
                          </Badge>
                        )}
                      </Button>
                      {!event.cancelled && (
                        <Button
                          variant="outline"
                          size="sm"
                          onClick={() => {
                            setSelectedEvent(event);
                            setCancelDialog(true);
                          }}
                        >
                          <CancelIcon className="mr-1" style={{ fontSize: 16 }} />
                          Cancel
                        </Button>
                      )}
                      <Button
                        variant="destructive"
                        size="sm"
                        onClick={() => {
                          setSelectedEvent(event);
                          setDeleteDialog(true);
```

```
                          }}
                        >
                          <DeleteIcon className="mr-1" style={{ fontSize: 16 }} />
                          Delete
                        </Button>
                      </div>
                    </TableCell>
                  </TableRow>
                  );
                })
              )}
            </TableBody>
          </Table>
        </div>
      </Card>

      {/* Delete Dialog */}
      <AlertDialog open={deleteDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Delete Event</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to delete "{selectedEvent?.title}"?
              This action cannot be undone and will remove the event for all participants.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => {
              setDeleteDialog(false);
              setSelectedEvent(null);
            }}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction
              onClick={() => selectedEvent && handleDelete(selectedEvent)}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Delete
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Cancel Dialog */}
      <AlertDialog open={cancelDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Cancel Event</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to cancel "{selectedEvent?.title}"?
              The event will be marked as cancelled but will remain in the database.
Participants will be notified.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => {
              setCancelDialog(false);
              setSelectedEvent(null);
            }}>
              No, Keep Active
            </AlertDialogCancel>
            <AlertDialogAction
              onClick={() => selectedEvent && handleCancel(selectedEvent)}
            >
              Cancel Event
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Participants Dialog */}
      <Dialog open={participantsDialog} onOpenChange={setParticipantsDialog}>
```

```
        <DialogContent className="max-w-2xl max-h-[80vh] overflow-y-auto">
          <DialogHeader>
            <DialogTitle>
              Participants - {selectedEvent?.title}
            </DialogTitle>
            <DialogDescription>
              {selectedEvent?.participants?.length || 0} participant(s)
            </DialogDescription>
          </DialogHeader>
          <div className="space-y-2 mt-4">
            {selectedEvent?.participants && selectedEvent.participants.length > 0 ? (
              selectedEvent.participants.map((userId) => {
                const participant = participantsData[userId];
                return (
                  <div key={userId} className="flex items-center justify-between p-3 border
rounded-lg">
                    <div className="flex items-center gap-3">
                      <div
                        <div className="w-10 h-10 rounded-full bg-primary/10 flex items-center
justify-center">
                          {participant?.name?.[0]?.toUpperCase() || userId[0]?.toUpperCase() ||
"?"}
                        </div>
                        <div>
                          <p className="font-medium">{participant?.name || "Unknown User"}</p>
                          <p className="text-xs text-muted-foreground">{userId}</p>
                          {participant?.email && (
                            <p className="text-xs text-muted-foreground">{participant.email}</p>
                          )}
                        </div>
                      </div>
                      <Button
                        variant="ghost"
                        size="sm"
                        onClick={() => navigate(`/admin/users?search=${userId}`)}
                      >
                        View User
                      </Button>
                    </div>
                  );
                })
              ) : (
                <p className="text-muted-foreground text-center py-4">No participants</p>
              )}
            </div>
          </DialogContent>
        </Dialog>

        {/* Delete Expired Events Dialog */}
        <AlertDialog open={deleteExpiredDialog}>
          <AlertDialogContent>
            <AlertDialogHeader>
              <AlertDialogTitle>Delete All Expired Events</AlertDialogTitle>
              <AlertDialogDescription>
                Are you sure you want to delete all {getExpiredEvents().length} expired event(s)?
                This action cannot be undone and will permanently remove these events from the
database.
                <br /><br />
                <strong>This will affect all participants who joined these events.</strong>
              </AlertDialogDescription>
            </AlertDialogHeader>
            <AlertDialogFooter>
              <AlertDialogCancel
                onClick={() => setDeleteExpiredDialog(false)}
                disabled={deletingExpired}
              >
                Cancel
              </AlertDialogCancel>
              <AlertDialogAction
                onClick={handleDeleteExpiredEvents}
                className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
                disabled={deletingExpired}
              >
                {deletingExpired ? "Deleting..." : `Delete ${getExpiredEvents().length} Event(s)`}
              </AlertDialogAction>
            </AlertDialogFooter>
```

```
777 |         </AlertDialogContent>
778 |       </AlertDialog>
779 |     </div>
780 |   );
781 | };
782 |
783 | export default AdminEvents;
784 |
785 |
```

File: src/pages/AdminLogin.tsx

Lines: 500

```
// Admin login page
import { useState, useEffect, useRef } from "react";
import { motion } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { useNavigate, useLocation } from "react-router-dom";
import EmailIcon from "@mui/icons-material/Email";
import LockIcon from "@mui/icons-material/Lock";
import PersonIcon from "@mui/icons-material/Person";
import AdminPanelSettingsIcon from "@mui/icons-material/AdminPanelSettings";
import { signInWithEmail, signInWithGoogle, signUpWithEmail, checkEmailExists } from "@/services/authService";
import { checkAdminStatus } from "@/services/adminService";
import { toast } from "sonner";
import { useAuth } from "@/hooks/useAuth";
import { auth } from "@/services/firebase";

const AdminLogin = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { user, loading: authLoading } = useAuth();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [displayName, setDisplayName] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [isSignUp, setIsSignUp] = useState(false);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const hasRedirected = useRef(false);

  // Redirect if user is already authenticated and is admin
  useEffect(() => {
    if (location.pathname !== "/admin/login") {
      return;
    }

    if (hasRedirected.current || authLoading || !user) {
      return;
    }

    const checkAdminAndRedirect = async () => {
      try {
        const isAdmin = await checkAdminStatus(user.email);
        if (isAdmin) {
          hasRedirected.current = true;
          navigate("/admin/dashboard", { replace: true });
        } else {
          // User is logged in but not admin
          setError("Access denied. Admin privileges required.");
          // Sign them out so they can try with admin account
          const { signOut } = await import("@/services/authService");
          await signOut();
        }
      } catch (err) {
        console.error("Error checking admin status:", err);
      }
    };

    checkAdminAndRedirect();
  }, [user, authLoading, navigate, location.pathname]);

  const handleEmailSignIn = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);
    setError(null);

    try {
```

```
 67 |       const userData = await signInWithEmail(email, password);
 68 |
 69 |       await new Promise(resolve => setTimeout(resolve, 500));
 70 |
 71 |       if (!auth.currentUser) {
 72 |         setLoading(false);
 73 |         return;
 74 |       }
 75 |
 76 |       const isAdmin = await checkAdminStatus(userData.email);
 77 |
 78 |       if (!isAdmin) {
 79 |         const { signOut } = await import("@/services/authService");
 80 |         await signOut();
 81 |         setError("Access denied. This account does not have admin privileges.");
 82 |         setLoading(false);
 83 |         return;
 84 |       }
 85 |
 86 |       hasRedirected.current = true;
 87 |       navigate("/admin/dashboard", { replace: true });
 88 |       setLoading(false);
 89 |     } catch (err: any) {
 90 |       console.error("Error signing in:", err);
 91 |       setError(err.message || "Failed to sign in. Please try again.");
 92 |       setLoading(false);
 93 |     }
 94 |   };
 95 |
 96 |   const handleEmailSignUp = async (e: React.FormEvent) => {
 97 |     e.preventDefault();
 98 |     if (password !== confirmPassword) {
 99 |       setError("Passwords do not match.");
100 |       return;
101 |     }
102 |
103 |     if (!displayName.trim()) {
104 |       setError("Please enter your name.");
105 |       return;
106 |     }
107 |
108 |     setLoading(true);
109 |     setError(null);
110 |
111 |     try {
112 |       // Check if email is already registered before attempting to create account
113 |       console.log("Ø=Ý  Checking if email is already registered:", email);
114 |       const emailExists = await checkEmailExists(email);
115 |
116 |       if (emailExists) {
117 |         setError("This email is already registered. Please sign in instead.");
118 |         setLoading(false);
119 |         return;
120 |       }
121 |
122 |       const newUser = await signUpWithEmail(email, password, displayName.trim());
123 |
124 |       await new Promise(resolve => setTimeout(resolve, 500));
125 |
126 |       if (!auth.currentUser) {
127 |         setLoading(false);
128 |         return;
129 |       }
130 |
131 |       const isAdmin = await checkAdminStatus(newUser.email);
132 |
133 |       if (!isAdmin) {
134 |         toast.info("Account created! Ask an existing admin to grant access to the admin
dashboard.");  navigate("/login", { replace: true });
136 |         return;
137 |       }
```

```
138 |
139 |       hasRedirected.current = true;
140 |       navigate("/admin/dashboard", { replace: true });
141 |     } catch (err: any) {
142 |       console.error("Error creating account:", err);
143 |       setError(err.message || "Failed to create account. Please try again.");
144 |     } finally {
145 |       setLoading(false);
146 |     }
147 |   };
148 |
149 |   const handleGoogleSignIn = async () => {
150 |     setLoading(true);
151 |     setError(null);
152 |
153 |     try {
154 |       const userData = await signInWithGoogle();
155 |
156 |       // If redirect was used, handleRedirectResult will be called in App.tsx
157 |       if (!userData) {
158 |         // Redirect was initiated, wait for it to complete
159 |         return;
160 |       }
161 |
162 |       // Wait for auth state to update
163 |       await new Promise(resolve => setTimeout(resolve, 500));
164 |
165 |       if (!auth.currentUser) {
166 |         setLoading(false);
167 |         return;
168 |       }
169 |
170 |       // Check if user is admin
171 |       const isAdmin = await checkAdminStatus(userData.email);
172 |
173 |       if (!isAdmin) {
174 |         const { signOut } = await import("@/services/authService");
175 |         await signOut();
176 |         setError("Access denied. This account does not have admin privileges.");
177 |         setLoading(false);
178 |         return;
179 |       }
180 |
181 |       // Admin login successful
182 |       hasRedirected.current = true;
183 |       navigate("/admin/dashboard", { replace: true });
184 |       setLoading(false);
185 |     } catch (err: any) {
186 |       console.error("Error signing in with Google:", err);
187 |       setError(err.message || "Failed to sign in with Google. Please try again.");
188 |       setLoading(false);
189 |     }
190 |   };
191 |
192 |   // Handle redirect result from Google sign-in
193 |   useEffect(() => {
194 |     const handleRedirect = async () => {
195 |       if (auth.currentUser && user) {
196 |         try {
197 |           const isAdmin = await checkAdminStatus(user.email);
198 |           if (isAdmin) {
199 |             navigate("/admin/dashboard", { replace: true });
200 |           } else {
201 |             const { signOut } = await import("@/services/authService");
202 |             await signOut();
203 |             setError("Access denied. This account does not have admin privileges.");
204 |           }
205 |         } catch (err) {
206 |           console.error("Error checking admin after redirect:", err);
207 |         }
208 |       }
```

```
    };

    if (user && location.pathname === "/admin/login") {
      handleRedirect();
    }
  }, [user, navigate, location.pathname]);

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/10 via-background to-destructive/10 flex items-center justify-center p-4">
      {/* Animated background blobs */}
      <div className="absolute inset-0 overflow-hidden pointer-events-none">
        <motion.div
          animate={{
            x: [0, 100, 0],
            y: [0, -100, 0],
            scale: [1, 1.2, 1],
          }}
          transition={{ duration: 20, repeat: Infinity, ease: "linear" }}
          className="absolute top-0 left-0 w-72 h-72 bg-primary/5 rounded-full blur-3xl"
        />
        <motion.div
          animate={{
            x: [0, -100, 0],
            y: [0, 100, 0],
            scale: [1, 1.3, 1],
          }}
          transition={{ duration: 25, repeat: Infinity, ease: "linear" }}
          className="absolute bottom-0 right-0 w-96 h-96 bg-destructive/5 rounded-full blur-3xl"
        />
      </div>

      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.6, ease: "easeOut" }}
        className="w-full max-w-md space-y-8 relative z-10"
      >
        {/* Logo and branding */}
        <div className="text-center space-y-6">
          <motion.div
            initial={{ scale: 0.5, opacity: 0 }}
            animate={{ scale: 1, opacity: 1 }}
            transition={{
              duration: 0.6,
              delay: 0.2,
              type: "spring",
              stiffness: 200
            }}
            className="flex justify-center"
          >
            <div className="relative">
              <motion.div
                animate={{
                  boxShadow: [
                    "0 0 20px rgba(220, 38, 38, 0.3)",
                    "0 0 40px rgba(220, 38, 38, 0.5)",
                    "0 0 20px rgba(220, 38, 38, 0.3)",
                  ]
                }}
                transition={{ duration: 2, repeat: Infinity }}
                className="bg-gradient-to-br from-destructive to-destructive/80 rounded-full p-8 shadow-elevation-4"
              >
                <AdminPanelSettingsIcon
                  sx={{ fontSize: 56 }}
                  style={{ fontSize: 56 }}
                  />
              </motion.div>
            </div>
          </motion.div>

          <motion.div
            initial={{ opacity: 0, y: 10 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ duration: 0.5, delay: 0.4 }}
```

```
          >
            <h1 className="text-4xl font-extrabold bg-gradient-to-r from-destructive via-primary to-destructive bg-clip-...">Admin Portal</h1>
            <p className="text-lg text-muted-foreground mt-3 font-medium">
              PaceMatch Administration
            </p>
          </motion.div>
        </div>

        {/* Error Alert */}
        {error && (
          <motion.div
            initial={{ opacity: 0, y: -10 }}
            animate={{ opacity: 1, y: 0 }}
            className="w-full"
          >
            <div className="bg-destructive/10 border border-destructive/20 text-destructive px-4 py-3 rounded-lg text-sm">{error}</div>
          </motion.div>
        )}

        {/* Login/Sign-up toggle */}
        <div className="flex gap-2 p-1 bg-muted rounded-lg">
          <Button
            variant={!isSignUp ? "default" : "ghost"}
            className="flex-1"
            onClick={() => {
              setIsSignUp(false);
              setError(null);
            }}
          >
            I already have an account
          </Button>
          <Button
            variant={isSignUp ? "default" : "ghost"}
            className="flex-1"
            onClick={() => {
              setIsSignUp(true);
              setError(null);
            }}
          >
            Create new admin account
          </Button>
        </div>

        {/* Login / Signup Form */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5, delay: 0.6 }}
          className="space-y-4"
        >
          <form onSubmit={isSignUp ? handleEmailSignUp : handleEmailSignIn} className="space-y-4">
            {isSignUp && (
              <div className="space-y-2">
                <label htmlFor="admin-name" className="text-sm font-medium text-muted-foreground">
                  Full Name
                </label>
                <div className="relative">
                  <PersonIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-foreground" style...
                  <Input
                    id="admin-name"
                    type="text"
                    placeholder="Jane Admin"
                    value={displayName}
                    onChange={(e) => setDisplayName(e.target.value)}
                    required
                    disabled={loading}
                    className="pl-10"
```

```
                  />
                </div>
              </div>
            )}

            {/* Email Input */}
            <div className="space-y-2">
              <label htmlFor="admin-email" className="text-sm font-medium text-muted-foreground">
                Email Address
              </label>
              <div className="relative">
                <EmailIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{
                <Input
                  id="admin-email"
                  type="email"
                  placeholder="admin@example.com"
                  value={email}
                  onChange={(e) => setEmail(e.target.value)}
                  required
                  disabled={loading}
                  className="pl-10"
                />
              </div>
            </div>

            {/* Password Input */}
            <div className="space-y-2">
              <label htmlFor="admin-password" className="text-sm font-medium text-muted-
foreground">
                Password
              </label>
              <div className="relative">
                <LockIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{
                <Input
                  id="admin-password"
                  type="password"
                  placeholder="Enter your password"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                  required
                  disabled={loading}
                  className="pl-10"
                />
              </div>
            </div>

            {isSignUp && (
              <div className="space-y-2">
                <label htmlFor="admin-password-confirm" className="text-sm font-medium text-
muted-foreground">
                  Confirm Password
                </label>
                <div className="relative">
                  <LockIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{
                  <Input
                    id="admin-password-confirm"
                    type="password"
                    placeholder="Confirm your password"
                    value={confirmPassword}
                    onChange={(e) => setConfirmPassword(e.target.value)}
                    required
                    disabled={loading}
                    className="pl-10"
                  />
                </div>
              </div>
            )}

            {/* Sign In Button */}
            <Button
              type="submit"
              className="w-full"
              disabled={loading}
```

```
              size="lg"
            >
              {loading ? (
                <div className="flex items-center gap-2">
                  <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-white"></div>
                  {isSignUp ? "Creating account..." : "Signing in..."}
                </div>
              ) : (
                isSignUp ? "Create Account" : "Sign In"
              )}
            </Button>
          </form>

          {/* Divider */}
          <div className="relative">
            <div className="absolute inset-0 flex items-center">
              <span className="w-full border-t" />
            </div>
            <div className="relative flex justify-center text-xs uppercase">
              <span className="bg-background px-2 text-muted-foreground">
                Or continue with
              </span>
            </div>
          </div>

          {/* Google Sign In Button */}
          <Button
            type="button"
            variant="outline"
            className="w-full"
            onClick={handleGoogleSignIn}
            disabled={loading}
            size="lg"
          >
            <svg className="mr-2 h-4 w-4" viewBox="0 0 24 24">
              <path
                d="M22.56 12.25c0-.78-.07-1.53-.2-2.25H12v4.26h5.92c-.26 1.37-1.04 2.53-2.21 3.31v2.77h3.57c2.08-1.92 3.28-4.74 3.28-8.09z"
                fill="#4285F4"
              />
              <path
                d="M12 23c2.97 0 5.46-.98 7.28-2.66l-3.57-2.77c-.98.66-2.23 1.06-3.71 1.06-2.86 0-5.29-1.93-6.16-4.53H2.18v2.84C3.99 20.53 7.7 23 12 23z"
                fill="#34A853"
              />
              <path
                d="M5.84 14.09c-.22-.66-.35-1.36-.35-2.09s.13-1.43.35-2.09V7.07H2.18C1.43 8.55 1 10.22 1 12s.43 3.45 1.18 4.93l2.85-2.22.81-.62z"
                fill="#FBBC05"
              />
              <path
                d="M12 5.38c1.62 0 3.06.56 4.21 1.64l3.15-3.15C17.45 2.09 14.97 1 12 1 7.7 1 3.99 3.47 2.18 7.07l3.66 2.84c.87-2.6 3.3-4.53 6.16-4.53z"
                fill="#EA4335"
              />
            </svg>
            Sign in with Google
          </Button>
        </motion.div>

        {/* Back to app link */}
        <motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ duration: 0.5, delay: 0.8 }}
          className="text-center"
        >
          <Button
            variant="ghost"
            onClick={() => navigate("/login")}
            className="text-sm text-muted-foreground"
          >
            Back to regular login
          </Button>
        </motion.div>
```

```
493 |       </motion.div>
494 |     </div>
495 |   );
496 | };
497 |
498 | export default AdminLogin;
499 |
500 |
```

File: src/pages/AdminModeration.tsx

Lines: 616

```tsx
// Admin Content Moderation Page
import { useState, useEffect } from "react";
import { useNavigate, useSearchParams } from "react-router-dom";
import { Card } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@/components/ui/table";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
  DialogDescription,
} from "@/components/ui/dialog";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import Avatar from "@mui/material/Avatar";
import { getAllReports, resolveReport, logAdminAction } from "@/services/adminService";
import { getUserData } from "@/services/authService";
import { useAuth } from "@/hooks/useAuth";
import { toast } from "sonner";
import SearchIcon from "@mui/icons-material/Search";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import CancelIcon from "@mui/icons-material/Cancel";
import PersonIcon from "@mui/icons-material/Person";
import ReportProblemIcon from "@mui/icons-material/ReportProblem";

interface Report {
  id: string;
  reporterId: string;
  reportedUserId: string;
  reason: string;
  description?: string;
  reportedAt: number;
  status?: "pending" | "resolved";
  resolvedAt?: number;
  resolvedAction?: string;
}

const AdminModeration = () => {
  const navigate = useNavigate();
  const { user: currentAdmin } = useAuth();
  const [searchParams] = useSearchParams();
  const [reports, setReports] = useState<Report[]>([]);
  const [filteredReports, setFilteredReports] = useState<Report[]>([]);
  const [loading, setLoading] = useState(true);
  const [searchQuery, setSearchQuery] = useState("");
  const [statusFilter, setStatusFilter] = useState<"all" | "pending" | "resolved">("all");
  const [selectedReport, setSelectedReport] = useState<Report | null>(null);
```

```
  const [actionDialog, setActionDialog] = useState<{
    open: boolean;
    action: "resolve" | "dismiss" | null;
  }>({ open: false, action: null });
  const [userData, setUserData] = useState<Record<string, any>>({});
  const [showProfileView, setShowProfileView] = useState(false);
  const [profileViewType, setProfileViewType] = useState<"reporter" | "reported">("reported");

  const userIdFilter = searchParams.get("userId");

  useEffect(() => {
    loadReports();
  }, []);

  useEffect(() => {
    filterReports();
  }, [reports, searchQuery, statusFilter, userIdFilter]);

  const loadReports = async () => {
    try {
      setLoading(true);
      const allReports = await getAllReports();
      setReports(allReports);

      // Load user data for all unique user IDs
      const userIds = new Set<string>();
      allReports.forEach((report: Report) => {
        userIds.add(report.reporterId);
        userIds.add(report.reportedUserId);
      });

      const userDataMap: Record<string, any> = {};
      for (const userId of userIds) {
        try {
          const data = await getUserData(userId);
          if (data) {
            userDataMap[userId] = data;
          }
        } catch (error) {
          console.error(`Error loading user ${userId}:`, error);
        }
      }
      setUserData(userDataMap);
    } catch (error) {
      console.error("Error loading reports:", error);
      toast.error("Failed to load reports");
    } finally {
      setLoading(false);
    }
  };

  const filterReports = () => {
    let filtered = [...reports];

    // Filter by user ID if specified
    if (userIdFilter) {
      filtered = filtered.filter(
        report => report.reportedUserId === userIdFilter
      );
    }

    // Filter by status
    if (statusFilter !== "all") {
      filtered = filtered.filter(report => {
        const status = report.status || "pending";
        return status === statusFilter;
      });
    }

    // Filter by search query
    if (searchQuery) {
```

```
      const query = searchQuery.toLowerCase();
      filtered = filtered.filter(report => {
        const reporterName = userData[report.reporterId]?.name || "";
        const reportedName = userData[report.reportedUserId]?.name || "";
        return (
          report.reason?.toLowerCase().includes(query) ||
          report.description?.toLowerCase().includes(query) ||
          reporterName.toLowerCase().includes(query) ||
          reportedName.toLowerCase().includes(query)
        );
      });
    }

    setFilteredReports(filtered);
  };

  const handleResolve = async (report: Report, action: string = "resolved") => {
    try {
      await resolveReport(report.id, action);
      if (currentAdmin?.email) {
        await logAdminAction(currentAdmin.email, "resolve_report", {
          reportId: report.id,
          reportedUserId: report.reportedUserId,
          reporterId: report.reporterId,
          reason: report.reason,
          action
        });
      }
      toast.success("Report resolved");
      await loadReports();
      setActionDialog({ open: false, action: null });
    } catch (error) {
      console.error("Error resolving report:", error);
      toast.error("Failed to resolve report");
    }
  };

  const formatDate = (timestamp: number) => {
    return new Date(timestamp).toLocaleString();
  };

  const getReasonBadge = (reason: string) => {
    const colors: Record<string, string> = {
      harassment: "destructive",
      spam: "warning",
      inappropriate: "destructive",
      fake: "secondary",
      other: "default"
    };
    return <Badge variant={colors[reason.toLowerCase()] as any || "default"}>{reason}</Badge>;
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-64">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
        <div>
          <div className="flex items-center gap-2 mb-2">
            <Button variant="ghost" size="sm" onClick={() => navigate("/admin/dashboard")}>
              !• Back to Dashboard
            </Button>
          </div>
          <h1 className="text-3xl font-bold">Content Moderation</h1>
```

```
209 |           <p className="text-muted-foreground mt-1">
210 |             Review and manage user reports
211 |           </p>
212 |         </div>
213 |         <div className="text-sm text-muted-foreground">
214 |           {filteredReports.length} reports
215 |           {userIdFilter && ` (filtered by user)`}
216 |         </div>
217 |       </div>
218 |
219 |       {/* Filters */}
220 |       <Card className="p-4">
221 |         <div className="flex flex-col sm:flex-row gap-4">
222 |           <div className="flex-1 relative">
223 |             <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-
foreground" style={{ fo...
224 |             <Input
225 |               placeholder="Search reports..."
226 |               value={searchQuery}
227 |               onChange={(e) => setSearchQuery(e.target.value)}
228 |               className="pl-10"
229 |             />
230 |           </div>
231 |           <div className="flex gap-2">
232 |             <Button
233 |               variant={statusFilter === "all" ? "default" : "outline"}
234 |               onClick={() => setStatusFilter("all")}
235 |             >
236 |               All
237 |             </Button>
238 |             <Button
239 |               variant={statusFilter === "pending" ? "default" : "outline"}
240 |               onClick={() => setStatusFilter("pending")}
241 |             >
242 |               Pending
243 |             </Button>
244 |             <Button
245 |               variant={statusFilter === "resolved" ? "default" : "outline"}
246 |               onClick={() => setStatusFilter("resolved")}
247 |             >
248 |               Resolved
249 |             </Button>
250 |           </div>
251 |           {userIdFilter && (
252 |             <Button
253 |               variant="outline"
254 |               onClick={() => navigate("/admin/moderation")}
255 |             >
256 |               Clear Filter
257 |             </Button>
258 |           )}
259 |         </div>
260 |       </Card>
261 |
262 |       {/* Reports Table */}
263 |       <Card>
264 |         <div className="overflow-x-auto">
265 |           <Table>
266 |             <TableHeader>
267 |               <TableRow>
268 |                 <TableHead>Reported User</TableHead>
269 |                 <TableHead>Reporter</TableHead>
270 |                 <TableHead>Reason</TableHead>
271 |                 <TableHead>Description</TableHead>
272 |                 <TableHead>Date</TableHead>
273 |                 <TableHead>Status</TableHead>
274 |                 <TableHead className="text-right">Actions</TableHead>
275 |               </TableRow>
276 |             </TableHeader>
277 |             <TableBody>
278 |               {filteredReports.length === 0 ? (
279 |                 <TableRow>
```

```
                  <TableCell colSpan={7} className="text-center py-8 text-muted-foreground">
                    No reports found
                  </TableCell>
                </TableRow>
              ) : (
                filteredReports.map((report) => {
                  const reportedUser = userData[report.reportedUserId];
                  const reporter = userData[report.reporterId];
                  const status = report.status || "pending";

                  return (
                    <TableRow key={report.id}>
                      <TableCell>
                        <div className="flex items-center gap-2">
                          <PersonIcon style={{ fontSize: 18 }} />
                          <div>
                            <p className="font-medium">
                              {reportedUser?.name || "Unknown User"}
                            </p>
                            <p className="text-xs text-muted-foreground">
                              {report.reportedUserId}
                            </p>
                          </div>
                        </div>
                      </TableCell>
                      <TableCell>
                        <div className="flex items-center gap-2">
                          <PersonIcon style={{ fontSize: 18 }} />
                          <div>
                            <p className="text-sm font-medium">
                              {reporter?.name || "Unknown User"}
                            </p>
                            <p className="text-xs text-muted-foreground">
                              {report.reporterId}
                            </p>
                          </div>
                        </div>
                      </TableCell>
                      <TableCell>{getReasonBadge(report.reason)}</TableCell>
                      <TableCell>
                        <div className="max-w-xs truncate">
                          {report.description || "No description"}
                        </div>
                      </TableCell>
                      <TableCell className="text-sm">
                        {formatDate(report.reportedAt)}
                      </TableCell>
                      <TableCell>
                        {status === "resolved" ? (
                          <Badge variant="success">Resolved</Badge>
                        ) : (
                          <Badge variant="warning">Pending</Badge>
                        )}
                      </TableCell>
                      <TableCell className="text-right">
                        <div className="flex justify-end gap-2 flex-wrap">
                          {status === "pending" && (
                            <>
                              <Button
                                variant="outline"
                                size="sm"
                                onClick={() => {
                                  setSelectedReport(report);
                                  setActionDialog({ open: true, action: "resolve" });
                                }}
                              >
                                <CheckCircleIcon className="mr-1" style={{ fontSize: 16 }} />
                                Resolve
                              </Button>
                              <Button
                                variant="outline"
```

```
                                size="sm"
                                onClick={() => {
                                  setSelectedReport(report);
                                  setActionDialog({ open: true, action: "dismiss" });
                                }}
                              >
                                <CancelIcon className="mr-1" style={{ fontSize: 16 }} />
                                Dismiss
                              </Button>
                            </>
                          )}
                          <Button
                            variant="outline"
                            size="sm"
                            onClick={() => {
                              setSelectedReport(report);
                              setProfileViewType("reported");
                              setShowProfileView(true);
                            }}
                            title="View reported user profile"
                          >
                            <PersonIcon className="mr-1" style={{ fontSize: 16 }} />
                            View Reported
                          </Button>
                          <Button
                            variant="outline"
                            size="sm"
                            onClick={() => {
                              setSelectedReport(report);
                              setProfileViewType("reporter");
                              setShowProfileView(true);
                            }}
                            title="View reporter profile"
                          >
                            <PersonIcon className="mr-1" style={{ fontSize: 16 }} />
                            View Reporter
                          </Button>
                        </div>
                      </TableCell>
                    </TableRow>
                  );
                })
              )}
            </TableBody>
          </Table>
        </div>
      </Card>

      {/* Action Dialogs */}
      <AlertDialog open={actionDialog.open && actionDialog.action === "resolve"}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Resolve Report</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to mark this report as resolved?
              You can take action on the reported user from the User Management page.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, action: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedReport && handleResolve(selectedReport,
"resolved")}>
              Resolve
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      <AlertDialog open={actionDialog.open && actionDialog.action === "dismiss"}>
        <AlertDialogContent>
```

```
          <AlertDialogHeader>
            <AlertDialogTitle>Dismiss Report</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to dismiss this report? This will mark it as resolved without taking action.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, action: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedReport && handleResolve(selectedReport, "dismissed")}>
              Dismiss
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Detailed Profile View Dialog */}
      <Dialog open={showProfileView} onOpenChange={setShowProfileView}>
        <DialogContent className="max-w-4xl max-h-[90vh] overflow-y-auto">
          <DialogHeader>
            <DialogTitle>Report Details</DialogTitle>
            <DialogDescription>
              View profiles of both users involved in this report
            </DialogDescription>
          </DialogHeader>

          {selectedReport && (
            <div className="space-y-6 mt-4">
              {/* Report Information */}
              <Card className="p-4">
                <div className="space-y-2">
                  <div className="flex items-center justify-between">
                    <span className="text-sm font-medium">Reason:</span>
                    {getReasonBadge(selectedReport.reason)}
                  </div>
                  {selectedReport.description && (
                    <div>
                      <span className="text-sm font-medium">Description:</span>
                      <p className="text-sm text-muted-foreground mt-1">{selectedReport.description}</p>
                    </div>
                  )}
                  <div className="flex items-center justify-between">
                    <span className="text-sm font-medium">Reported At:</span>
                    <span className="text-sm text-muted-foreground">{formatDate(selectedReport.reportedAt)}</span>
                  </div>
                  <div className="flex items-center justify-between">
                    <span className="text-sm font-medium">Status:</span>
                    {selectedReport.status === "resolved" ? (
                      <Badge variant="success">Resolved</Badge>
                    ) : (
                      <Badge variant="warning">Pending</Badge>
                    )}
                  </div>
                </div>
              </Card>

              {/* User Profiles */}
              <Tabs defaultValue={profileViewType} className="w-full">
                <TabsList className="grid w-full grid-cols-2">
                  <TabsTrigger value="reported">Reported User</TabsTrigger>
                  <TabsTrigger value="reporter">Reporter</TabsTrigger>
                </TabsList>

                <TabsContent value="reported" className="space-y-4 mt-4">
                  {userData[selectedReport.reportedUserId] ? (
                    <Card className="p-6">
                      <div className="flex items-start gap-4">
                        <Avatar
                          src={userData[selectedReport.reportedUserId]?.photoURL || `https://ui-avatars.com/api/?name=${...
                          alt={userData[selectedReport.reportedUserId]?.name || "User"}
```

```
493 |                         sx={{ width: 80, height: 80 }}
494 |                       />
495 |                       <div className="flex-1 space-y-2">
496 |                         <div>
497 |                           <h3 className="text-xl font-bold">
498 |                             {userData[selectedReport.reportedUserId]?.name || "Unknown User"}
499 |                           </h3>
500 |                           <p className="text-sm text-muted-foreground">
501 |                             {userData[selectedReport.reportedUserId]?.username || ""}
502 |                           </p>
503 |                           <p className="text-sm text-muted-foreground">
504 |                             {userData[selectedReport.reportedUserId]?.email || ""}
505 |                           </p>
506 |                           <p className="text-xs text-muted-foreground mt-1">
507 |                             User ID: {selectedReport.reportedUserId}
508 |                           </p>
509 |                         </div>
510 |                         {userData[selectedReport.reportedUserId]?.bio && (
511 |                           <div className="pt-2 border-t border-border">
512 |                             <p className="text-sm text-muted-foreground">
513 |                               {userData[selectedReport.reportedUserId].bio}
514 |                             </p>
515 |                           </div>
516 |                         )}
517 |                         <div className="flex gap-2 pt-2">
518 |                           <Button
519 |                             variant="outline"
520 |                             size="sm"
521 |                             onClick={() => navigate(`/admin/users?
search=${selectedReport.reportedUserId}`)}
522 |                           >
523 |                             View Full Profile
524 |                           </Button>
525 |                           <Button
526 |                             variant="outline"
527 |                             size="sm"
528 |                             onClick={() => {
529 |                               setShowProfileView(false);
530 |                               navigate(`/admin/users?search=${selectedReport.reportedUserId}`);
531 |                             }}
532 |                           >
533 |                             Manage User
534 |                           </Button>
535 |                         </div>
536 |                       </div>
537 |                     </div>
538 |                   </Card>
539 |                 ) : (
540 |                   <Card className="p-6">
541 |                     <p className="text-muted-foreground">Loading user data...</p>
542 |                   </Card>
543 |                 )}
544 |               </TabsContent>
545 |
546 |               <TabsContent value="reporter" className="space-y-4 mt-4">
547 |                 {userData[selectedReport.reporterId] ? (
548 |                   <Card className="p-6">
549 |                     <div className="flex items-start gap-4">
550 |                       <Avatar
551 |                         src={userData[selectedReport.reporterId]?.photoURL || `https://ui-
avatars.com/api/?name=${user...
552 |                         alt={userData[selectedReport.reporterId]?.name || "User"}
553 |                         sx={{ width: 80, height: 80 }}
554 |                       />
555 |                       <div className="flex-1 space-y-2">
556 |                         <div>
557 |                           <h3 className="text-xl font-bold">
558 |                             {userData[selectedReport.reporterId]?.name || "Unknown User"}
559 |                           </h3>
560 |                           <p className="text-sm text-muted-foreground">
561 |                             {userData[selectedReport.reporterId]?.username || ""}
562 |                           </p>
563 |                           <p className="text-sm text-muted-foreground">
```

```
                              {userData[selectedReport.reporterId]?.email || ""}
                            </p>
                            <p className="text-xs text-muted-foreground mt-1">
                              User ID: {selectedReport.reporterId}
                            </p>
                          </div>
                          {userData[selectedReport.reporterId]?.bio && (
                            <div className="pt-2 border-t border-border">
                              <p className="text-sm text-muted-foreground">
                                {userData[selectedReport.reporterId].bio}
                              </p>
                            </div>
                          )}
                          <div className="flex gap-2 pt-2">
                            <Button
                              variant="outline"
                              size="sm"
                              onClick={() => navigate(`/admin/users?
search=${selectedReport.reporterId}`)}
                            >
                              View Full Profile
                            </Button>
                            <Button
                              variant="outline"
                              size="sm"
                              onClick={() => {
                                setShowProfileView(false);
                                navigate(`/admin/users?search=${selectedReport.reporterId}`);
                              }}
                            >
                              Manage User
                            </Button>
                          </div>
                        </div>
                      </div>
                    </Card>
                  ) : (
                    <Card className="p-6">
                      <p className="text-muted-foreground">Loading user data...</p>
                    </Card>
                  )}
                </TabsContent>
              </Tabs>
            </div>
          )}
        </DialogContent>
      </Dialog>
    </div>
  );
};

export default AdminModeration;
```

File: src/pages/AdminSettings.tsx

Lines: 526

```
// Admin System Settings Page
import { useState, useEffect } from "react";
import { Card } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import { Switch } from "@/components/ui/switch";
import { Label } from "@/components/ui/label";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@/components/ui/table";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import { ref, get, set, remove } from "firebase/database";
import { database } from "@/services/firebase";
import { getAdminEmails, addAdminEmail, removeAdminEmail, getAdminLogs, logAdminAction } from "@/
services/adminService";
import { useAuth } from "@/hooks/useAuth";
import { toast } from "sonner";
import SettingsIcon from "@mui/icons-material/Settings";
import AddIcon from "@mui/icons-material/Add";
import DeleteIcon from "@mui/icons-material/Delete";
import EmailIcon from "@mui/icons-material/Email";
import HistoryIcon from "@mui/icons-material/History";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { useNavigate } from "react-router-dom";

const AdminSettings = () => {
  const navigate = useNavigate();
  const { user: currentAdmin } = useAuth();
  const [adminEmails, setAdminEmails] = useState<string[]>([]);
  const [newAdminEmail, setNewAdminEmail] = useState("");
  const [loading, setLoading] = useState(true);
  const [addingEmail, setAddingEmail] = useState(false);
  const [adminLogs, setAdminLogs] = useState<any[]>([]);
  const [loadingLogs, setLoadingLogs] = useState(false);
  const [deleteDialog, setDeleteDialog] = useState<{ open: boolean; email: string | null }>({
    open: false,
    email: null
  });

  // Feature flags
  const [featureFlags, setFeatureFlags] = useState({
    newRegistrations: true,
    workoutTracking: true,
    nearbyUsers: true,
    notifications: true,
    maintenanceMode: false
  });

  useEffect(() => {
    loadSettings();
    loadAdminLogs();
  }, []);
```

```
 67 |
 68 |   const loadSettings = async () => {
 69 |     try {
 70 |       setLoading(true);
 71 |
 72 |       // Load admin emails
 73 |       const emails = await getAdminEmails();
 74 |       setAdminEmails(emails);
 75 |
 76 |       // Load feature flags
 77 |       const flagsRef = ref(database, "systemSettings/featureFlags");
 78 |       const flagsSnapshot = await get(flagsRef);
 79 |       if (flagsSnapshot.exists()) {
 80 |         setFeatureFlags({
 81 |           ...featureFlags,
 82 |           ...flagsSnapshot.val()
 83 |         });
 84 |       }
 85 |     } catch (error) {
 86 |       console.error("Error loading settings:", error);
 87 |       toast.error("Failed to load settings");
 88 |     } finally {
 89 |       setLoading(false);
 90 |     }
 91 |   };
 92 |
 93 |   const loadAdminLogs = async () => {
 94 |     try {
 95 |       setLoadingLogs(true);
 96 |       const logs = await getAdminLogs(50);
 97 |       setAdminLogs(logs);
 98 |     } catch (error) {
 99 |       console.error("Error loading admin logs:", error);
100 |     } finally {
101 |       setLoadingLogs(false);
102 |     }
103 |   };
104 |
105 |   const handleAddAdminEmail = async () => {
106 |     if (!newAdminEmail || !newAdminEmail.includes("@")) {
107 |       toast.error("Please enter a valid email address");
108 |       return;
109 |     }
110 |
111 |     if (adminEmails.includes(newAdminEmail)) {
112 |       toast.error("This email is already an admin");
113 |       return;
114 |     }
115 |
116 |     try {
117 |       setAddingEmail(true);
118 |       await addAdminEmail(newAdminEmail);
119 |       if (currentAdmin?.email) {
120 |         await logAdminAction(currentAdmin.email, "add_admin_email", {
121 |           addedEmail: newAdminEmail
122 |         });
123 |       }
124 |       toast.success(`Admin email ${newAdminEmail} added successfully`);
125 |       setNewAdminEmail("");
126 |       await loadSettings();
127 |       await loadAdminLogs();
128 |     } catch (error) {
129 |       console.error("Error adding admin email:", error);
130 |       toast.error("Failed to add admin email");
131 |     } finally {
132 |       setAddingEmail(false);
133 |     }
134 |   };
135 |
136 |   const handleRemoveAdminEmail = async (email: string) => {
137 |     try {
```

```
138 |       await removeAdminEmail(email);
139 |       if (currentAdmin?.email) {
140 |         await logAdminAction(currentAdmin.email, "remove_admin_email", {
141 |           removedEmail: email
142 |         });
143 |       }
144 |       toast.success(`Admin email ${email} removed successfully`);
145 |       await loadSettings();
146 |       await loadAdminLogs();
147 |       setDeleteDialog({ open: false, email: null });
148 |     } catch (error) {
149 |       console.error("Error removing admin email:", error);
150 |       toast.error("Failed to remove admin email");
151 |     }
152 |   };
153 |
154 |   const handleToggleFeatureFlag = async (flag: keyof typeof featureFlags) => {
155 |     try {
156 |       const newValue = !featureFlags[flag];
157 |       const flagsRef = ref(database, `systemSettings/featureFlags/${flag}`);
158 |       await set(flagsRef, newValue);
159 |
160 |       setFeatureFlags({
161 |         ...featureFlags,
162 |         [flag]: newValue
163 |       });
164 |
165 |       toast.success(`${flag} ${newValue ? "enabled" : "disabled"}`);
166 |     } catch (error) {
167 |       console.error("Error updating feature flag:", error);
168 |       toast.error("Failed to update setting");
169 |     }
170 |   };
171 |
172 |   if (loading) {
173 |     return (
174 |       <div className="flex items-center justify-center h-64">
175 |         <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
176 |       </div>
177 |     );
178 |   }
179 |
180 |   const formatLogDate = (timestamp: number) => {
181 |     return new Date(timestamp).toLocaleString();
182 |   };
183 |
184 |   const getActionLabel = (action: string) => {
185 |     const labels: Record<string, string> = {
186 |       promote_admin: "Promoted User to Admin",
187 |       demote_admin: "Demoted Admin",
188 |       suspend_user: "Suspended User",
189 |       ban_user: "Banned User",
190 |       unban_user: "Unbanned User",
191 |       delete_user: "Deleted User",
192 |       resolve_report: "Resolved Report",
193 |       delete_event: "Deleted Event",
194 |       cancel_event: "Cancelled Event",
195 |       add_admin_email: "Added Admin Email",
196 |       remove_admin_email: "Removed Admin Email"
197 |     };
198 |     return labels[action] || action;
199 |   };
200 |
201 |   return (
202 |     <div className="space-y-6">
203 |       {/* Header */}
204 |       <div>
205 |         <div className="flex items-center gap-2 mb-2">
206 |           <Button variant="ghost" size="sm" onClick={() => navigate("/admin/dashboard")}>
207 |             !• Back to Dashboard
208 |           </Button>
```

```
209 |         </div>
210 |         <h1 className="text-3xl font-bold">System Settings</h1>
211 |         <p className="text-muted-foreground mt-1">
212 |           Configure app settings and manage admin access
213 |         </p>
214 |       </div>
215 |
216 |       <Tabs defaultValue="settings" className="space-y-6">
217 |         <TabsList>
218 |           <TabsTrigger value="settings">Settings</TabsTrigger>
219 |           <TabsTrigger value="admins">Admin List</TabsTrigger>
220 |           <TabsTrigger value="logs">Activity Logs</TabsTrigger>
221 |         </TabsList>
222 |
223 |         <TabsContent value="settings" className="space-y-6">
224 |
225 |       {/* Feature Flags */}
226 |       <Card className="p-6">
227 |         <h3 className="text-lg font-semibold mb-4">Feature Flags</h3>
228 |         <div className="space-y-4">
229 |           <div className="flex items-center justify-between">
230 |             <div className="space-y-0.5">
231 |               <Label htmlFor="new-registrations">New Registrations</Label>
232 |               <p className="text-sm text-muted-foreground">
233 |                 Allow new users to sign up
234 |               </p>
235 |             </div>
236 |             <Switch
237 |               id="new-registrations"
238 |               checked={featureFlags.newRegistrations}
239 |               onCheckedChange={() => handleToggleFeatureFlag("newRegistrations")}
240 |             />
241 |           </div>
242 |
243 |           <div className="flex items-center justify-between">
244 |             <div className="space-y-0.5">
245 |               <Label htmlFor="workout-tracking">Workout Tracking</Label>
246 |               <p className="text-sm text-muted-foreground">
247 |                 Enable workout tracking features
248 |               </p>
249 |             </div>
250 |             <Switch
251 |               id="workout-tracking"
252 |               checked={featureFlags.workoutTracking}
253 |               onCheckedChange={() => handleToggleFeatureFlag("workoutTracking")}
254 |             />
255 |           </div>
256 |
257 |           <div className="flex items-center justify-between">
258 |             <div className="space-y-0.5">
259 |               <Label htmlFor="nearby-users">Nearby Users</Label>
260 |               <p className="text-sm text-muted-foreground">
261 |                 Show nearby users on map
262 |               </p>
263 |             </div>
264 |             <Switch
265 |               id="nearby-users"
266 |               checked={featureFlags.nearbyUsers}
267 |               onCheckedChange={() => handleToggleFeatureFlag("nearbyUsers")}
268 |             />
269 |           </div>
270 |
271 |           <div className="flex items-center justify-between">
272 |             <div className="space-y-0.5">
273 |               <Label htmlFor="notifications">Notifications</Label>
274 |               <p className="text-sm text-muted-foreground">
275 |                 Enable push notifications
276 |               </p>
277 |             </div>
278 |             <Switch
279 |               id="notifications"
```

```
              checked={featureFlags.notifications}
              onCheckedChange={() => handleToggleFeatureFlag("notifications")}
            />
          </div>

          <div className="flex items-center justify-between">
            <div className="space-y-0.5">
              <Label htmlFor="maintenance-mode">Maintenance Mode</Label>
              <p className="text-sm text-muted-foreground">
                Disable app access for maintenance
              </p>
            </div>
            <Switch
              id="maintenance-mode"
              checked={featureFlags.maintenanceMode}
              onCheckedChange={() => handleToggleFeatureFlag("maintenanceMode")}
            />
          </div>
        </div>
      </Card>

        {/* Admin Email Management */}
        <Card className="p-6">
        <h3 className="text-lg font-semibold mb-4">Admin Email Management</h3>

        {/* Add Admin Email */}
        <div className="mb-6">
          <Label htmlFor="new-admin-email" className="mb-2 block">
            Add New Admin Email
          </Label>
          <div className="flex gap-2">
            <div className="flex-1 relative">
              <EmailIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{ f...
              <Input
                id="new-admin-email"
                type="email"
                placeholder="admin@example.com"
                value={newAdminEmail}
                onChange={(e) => setNewAdminEmail(e.target.value)}
                className="pl-10"
                onKeyDown={(e) => {
                  if (e.key === "Enter") {
                    handleAddAdminEmail();
                  }
                }}
              />
            </div>
            <Button
              onClick={handleAddAdminEmail}
              disabled={addingEmail || !newAdminEmail}
            >
              <AddIcon className="mr-2" style={{ fontSize: 18 }} />
              Add
            </Button>
          </div>
        </div>

        {/* Admin Emails List */}
        <div>
          <Label className="mb-2 block">Current Admin Emails</Label>
          {adminEmails.length === 0 ? (
            <p className="text-sm text-muted-foreground py-4">
              No admin emails configured
            </p>
          ) : (
            <Table>
              <TableHeader>
                <TableRow>
                  <TableHead>Email</TableHead>
                  <TableHead className="text-right">Actions</TableHead>
                </TableRow>
```

```
              </TableHeader>
              <TableBody>
                {adminEmails.map((email) => (
                  <TableRow key={email}>
                    <TableCell>
                      <div className="flex items-center gap-2">
                        <EmailIcon style={{ fontSize: 18 }} />
                        <span>{email}</span>
                      </div>
                    </TableCell>
                    <TableCell className="text-right">
                      <Button
                        variant="destructive"
                        size="sm"
                        onClick={() => setDeleteDialog({ open: true, email })}
                      >
                        <DeleteIcon className="mr-1" style={{ fontSize: 16 }} />
                        Remove
                      </Button>
                    </TableCell>
                  </TableRow>
                ))}
              </TableBody>
            </Table>
          )}
        </div>
      </Card>

        {/* System Information */}
        <Card className="p-6">
        <h3 className="text-lg font-semibold mb-4">System Information</h3>
        <div className="space-y-3">
          <div className="flex justify-between">
            <span className="text-sm text-muted-foreground">App Version</span>
            <span className="text-sm font-medium">1.0.0</span>
          </div>
          <div className="flex justify-between">
            <span className="text-sm text-muted-foreground">Database</span>
            <span className="text-sm font-medium">Firebase Realtime Database</span>
          </div>
          <div className="flex justify-between">
            <span className="text-sm text-muted-foreground">Last Updated</span>
            <span className="text-sm font-medium">
              {new Date().toLocaleString()}
            </span>
          </div>
        </div>
      </Card>

        {/* Database Management */}
        <Card className="p-6">
        <h3 className="text-lg font-semibold mb-4">Database Management</h3>
        <div className="space-y-3">
          <p className="text-sm text-muted-foreground">
            Database management operations should be performed through the Firebase Console.
          </p>
          <Button
            variant="outline"
            onClick={() => window.open("https://console.firebase.google.com", "_blank")}
          >
            Open Firebase Console
          </Button>
        </div>
      </Card>

        </TabsContent>

        <TabsContent value="admins" className="space-y-6">
          <Card className="p-6">
            <h3 className="text-lg font-semibold mb-4">Current Admin Emails</h3>
            {adminEmails.length === 0 ? (
```

```
              <p className="text-sm text-muted-foreground py-4">
                No admin emails configured
              </p>
            ) : (
              <div className="space-y-2">
                {adminEmails.map((email) => (
                  <div key={email} className="flex items-center justify-between p-3 border
rounded-lg">
                    <div className="flex items-center gap-2">
                      <EmailIcon style={{ fontSize: 18 }} />
                      <span>{email}</span>
                    </div>
                    <Button
                      variant="destructive"
                      size="sm"
                      onClick={() => setDeleteDialog({ open: true, email })}
                    >
                      <DeleteIcon className="mr-1" style={{ fontSize: 16 }} />
                      Remove
                    </Button>
                  </div>
                ))}
              </div>
            )}
          </Card>
        </TabsContent>

        <TabsContent value="logs" className="space-y-6">
          <Card className="p-6">
            <div className="flex items-center justify-between mb-4">
              <h3 className="text-lg font-semibold">Admin Activity Logs</h3>
              <Button variant="outline" size="sm" onClick={loadAdminLogs} disabled={loadingLogs}>
                <HistoryIcon className="mr-2" style={{ fontSize: 16 }} />
                Refresh
              </Button>
            </div>
            {loadingLogs ? (
              <div className="flex items-center justify-center py-8">
                <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-primary"></
div>
              </div>
            ) : adminLogs.length === 0 ? (
              <p className="text-sm text-muted-foreground py-8 text-center">
                No admin logs found
              </p>
            ) : (
              <div className="space-y-2 max-h-[600px] overflow-y-auto">
                {adminLogs.map((log) => (
                  <div key={log.id} className="p-3 border rounded-lg">
                    <div className="flex items-start justify-between">
                      <div className="flex-1">
                        <div className="flex items-center gap-2 mb-1">
                          <Badge variant="outline">{getActionLabel(log.action)}</Badge>
                          <span className="text-sm text-muted-foreground">
                            by {log.adminEmail}
                          </span>
                        </div>
                        {log.details && Object.keys(log.details).length > 0 && (
                          <div className="mt-2 text-xs text-muted-foreground">
                            <pre className="bg-muted p-2 rounded overflow-auto">
                              {JSON.stringify(log.details, null, 2)}
                            </pre>
                          </div>
                        )}
                        <p className="text-xs text-muted-foreground mt-2">
                          {formatLogDate(log.timestamp)}
                        </p>
                      </div>
                    </div>
                  </div>
                ))}
              </div>
            )}
```

```
          </Card>
        </TabsContent>
      </Tabs>

      {/* Delete Admin Email Dialog */}
      <AlertDialog open={deleteDialog.open}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Remove Admin Email</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to remove {deleteDialog.email} from admin access?
              This user will no longer be able to access the admin panel.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setDeleteDialog({ open: false, email: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction
              onClick={() => deleteDialog.email && handleRemoveAdminEmail(deleteDialog.email)}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Remove
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>
    </div>
  );
};

export default AdminSettings;

```

File: src/pages/AdminSetup.tsx

Lines: 158

```
  1 | // One-time admin setup page - Add first admin email
  2 | import { useState } from "react";
  3 | import { Button } from "@/components/ui/button";
  4 | import { Input } from "@/components/ui/input";
  5 | import { Card } from "@/components/ui/card";
  6 | import { ref, set, get } from "firebase/database";
  7 | import { database } from "@/services/firebase";
  8 | import { toast } from "sonner";
  9 | import EmailIcon from "@mui/icons-material/Email";
 10 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
 11 |
 12 | const AdminSetup = () => {
 13 |   const [email, setEmail] = useState("mattycycling@gmail.com");
 14 |   const [loading, setLoading] = useState(false);
 15 |   const [success, setSuccess] = useState(false);
 16 |
 17 |   const handleAddAdmin = async () => {
 18 |     if (!email || !email.includes("@")) {
 19 |       toast.error("Please enter a valid email address");
 20 |       return;
 21 |     }
 22 |
 23 |     try {
 24 |       setLoading(true);
 25 |
 26 |       // Check if adminEmails node exists, if not create it
 27 |       const adminEmailsRef = ref(database, "adminEmails");
 28 |       const snapshot = await get(adminEmailsRef);
 29 |
 30 |       // Add the email
 31 |       const emailRef = ref(database, `adminEmails/${email}`);
 32 |       await set(emailRef, true);
 33 |
 34 |       toast.success(`Admin email ${email} added successfully!`);
 35 |       setSuccess(true);
 36 |     } catch (error: any) {
 37 |       console.error("Error adding admin email:", error);
 38 |       toast.error(`Failed to add admin email: ${error.message}`);
 39 |     } finally {
 40 |       setLoading(false);
 41 |     }
 42 |   };
 43 |
 44 |   const handleVerify = async () => {
 45 |     try {
 46 |       const emailRef = ref(database, `adminEmails/${email}`);
 47 |       const snapshot = await get(emailRef);
 48 |
 49 |       if (snapshot.exists() && snapshot.val() === true) {
 50 |         toast.success("Admin email is already set up!");
 51 |         setSuccess(true);
 52 |       } else {
 53 |         toast.info("Admin email not found. Please add it.");
 54 |         setSuccess(false);
 55 |       }
 56 |     } catch (error: any) {
 57 |       console.error("Error verifying:", error);
 58 |       toast.error("Failed to verify admin email");
 59 |     }
 60 |   };
 61 |
 62 |   if (success) {
 63 |     return (
 64 |       <div className="min-h-screen flex items-center justify-center p-6 bg-gradient-to-br from-
primary/10 via-background
 65 |         <Card className="p-8 max-w-md w-full text-center space-y-4">
 66 |           <CheckCircleIcon className="mx-auto text-success" style={{ fontSize: 64 }} />
```

```
          <h1 className="text-2xl font-bold">Admin Email Added!</h1>
          <p className="text-muted-foreground">
            The email <strong>{email}</strong> has been successfully added as an admin.
          </p>
          <div className="space-y-2 pt-4">
            <p className="text-sm text-muted-foreground">Next steps:</p>
            <ol className="text-sm text-left space-y-1 list-decimal list-inside">
              <li>Go to <code className="bg-muted px-1 rounded">/admin/login</code></li>
              <li>Sign in with <strong>{email}</strong></li>
              <li>You'll be redirected to the admin dashboard</li>
            </ol>
          </div>
          <Button onClick={() => window.location.href = "/admin/login"} className="w-full mt-4">
            Go to Admin Login
          </Button>
          <Button
            variant="outline"
            onClick={() => {
              setSuccess(false);
              setEmail("");
            }}
            className="w-full"
          >
            Add Another Email
          </Button>
        </Card>
      </div>
    );
  }

  return (
    <div className="min-h-screen flex items-center justify-center p-6 bg-gradient-to-br from-
primary/10 via-background t...
        <Card className="p-8 max-w-md w-full space-y-6">
          <div className="text-center space-y-2">
            <h1 className="text-3xl font-bold">Admin Setup</h1>
            <p className="text-muted-foreground">
              Add your first admin email to Firebase Database
            </p>
          </div>

          <div className="space-y-4">
            <div className="space-y-2">
              <label htmlFor="admin-email" className="text-sm font-medium">
                Admin Email Address
              </label>
              <div className="relative">
                <EmailIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{ f...
                <input
                  id="admin-email"
                  type="email"
                  placeholder="admin@example.com"
                  value={email}
                  onChange={(e) => setEmail(e.target.value)}
                  className="pl-10"
                />
              </div>
            </div>

            <div className="flex gap-2">
              <Button
                onClick={handleAddAdmin}
                disabled={loading || !email}
                className="flex-1"
              >
                {loading ? "Adding..." : "Add Admin Email"}
              </Button>
              <Button
                variant="outline"
                onClick={handleVerify}
                disabled={loading}
              >
```

```
138 |               Verify
139 |             </Button>
140 |           </div>
141 |         </div>
142 |
143 |         <div className="pt-4 border-t space-y-2">
144 |           <p className="text-xs text-muted-foreground">
145 |             <strong>Note:</strong> This page can be deleted after setup. It's a one-time setup
tool.
146 |           </p>
147 |           <p className="text-xs text-muted-foreground">
148 |             Make sure you're authenticated with Firebase (logged in to the app) for this to work.
149 |           </p>
150 |         </div>
151 |       </Card>
152 |     </div>
153 |   );
154 | };
155 |
156 | export default AdminSetup;
157 |
158 |
```

File: src/pages/AdminUsers.tsx

Lines: 741

```
// Admin Users Management Page
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@/components/ui/table";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
import UserDetailDrawer from "@/components/UserDetailDrawer";
import { getAllUsers, suspendUser, banUser, unbanUser, unsuspendUser, deleteUser,
getUserReports, promoteUserToAdmin, de
import { useAuth } from "@/hooks/useAuth";
import { toast } from "sonner";
import SearchIcon from "@mui/icons-material/Search";
import MoreVertIcon from "@mui/icons-material/MoreVert";
import PersonIcon from "@mui/icons-material/Person";
import BlockIcon from "@mui/icons-material/Block";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import DeleteIcon from "@mui/icons-material/Delete";
import ReportProblemIcon from "@mui/icons-material/ReportProblem";
import AdminPanelSettingsIcon from "@mui/icons-material/AdminPanelSettings";
import PersonRemoveIcon from "@mui/icons-material/PersonRemove";
import EditIcon from "@mui/icons-material/Edit";

interface User {
  uid: string;
  name: string;
  email: string;
  photoURL?: string;
  status?: "active" | "suspended" | "banned";
  suspendedUntil?: number;
  suspendedReason?: string;
  bannedAt?: number;
  bannedReason?: string;
  createdAt?: number;
  timestamp?: number;
  isAdmin?: boolean;
}

const AdminUsers = () => {
  const navigate = useNavigate();
  const { user: currentAdmin } = useAuth();
  const [users, setUsers] = useState<User[]>([]);
```

```
  67 |   const [filteredUsers, setFilteredUsers] = useState<User[]>([]);
  68 |   const [loading, setLoading] = useState(true);
  69 |   const [searchQuery, setSearchQuery] = useState("");
  70 |   const [statusFilter, setStatusFilter] = useState<"all" | "active" | "suspended" |
"banned">("all");
  71 |   const [selectedUser, setSelectedUser] = useState<User | null>(null);
  72 |   const [actionDialog, setActionDialog] = useState<{
  73 |     open: boolean;
  74 |     type: "suspend" | "ban" | "unban" | "unsuspend" | "delete" | "promote" | "demote" |
"changeUsername" | null;
  75 |   }>({ open: false, type: null });
  76 |   const [adminStatuses, setAdminStatuses] = useState<Record<string, boolean>>({});
  77 |   const [selectedUserId, setSelectedUserId] = useState<string | null>(null);
  78 |   const [userDetailOpen, setUserDetailOpen] = useState(false);
  79 |
  80 |   useEffect(() => {
  81 |     loadUsers();
  82 |   }, []);
  83 |
  84 |   useEffect(() => {
  85 |     filterUsers();
  86 |   }, [users, searchQuery, statusFilter]);
  87 |
  88 |   const loadUsers = async () => {
  89 |     try {
  90 |       setLoading(true);
  91 |       const allUsers = await getAllUsers();
  92 |       setUsers(allUsers);
  93 |
  94 |       // Check admin status for each user
  95 |       const adminMap: Record<string, boolean> = {};
  96 |       for (const user of allUsers) {
  97 |         if (user.email) {
  98 |           adminMap[user.uid] = await isUserAdmin(user.email);
  99 |         }
 100 |       }
 101 |       setAdminStatuses(adminMap);
 102 |     } catch (error) {
 103 |       console.error("Error loading users:", error);
 104 |       toast.error("Failed to load users");
 105 |     } finally {
 106 |       setLoading(false);
 107 |     }
 108 |   };
 109 |
 110 |   const filterUsers = () => {
 111 |     let filtered = [...users];
 112 |
 113 |     // Filter by status
 114 |     if (statusFilter !== "all") {
 115 |       filtered = filtered.filter(user => {
 116 |         const status = user.status || "active";
 117 |         return status === statusFilter;
 118 |       });
 119 |     }
 120 |
 121 |     // Filter by search query
 122 |     if (searchQuery) {
 123 |       const query = searchQuery.toLowerCase();
 124 |       filtered = filtered.filter(user =>
 125 |         user.name?.toLowerCase().includes(query) ||
 126 |         user.email?.toLowerCase().includes(query) ||
 127 |         user.uid.toLowerCase().includes(query)
 128 |       );
 129 |     }
 130 |
 131 |     setFilteredUsers(filtered);
 132 |   };
 133 |
 134 |   const handleSuspend = async (user: User) => {
 135 |     try {
 136 |       await suspendUser(user.uid, 7, "Suspended by admin");
 137 |       if (currentAdmin?.email) {
```

```
138 |         await logAdminAction(currentAdmin.email, "suspend_user", {
139 |           userId: user.uid,
140 |           userName: user.name,
141 |           userEmail: user.email
142 |         });
143 |       }
144 |       toast.success(`User ${user.name || user.email} has been suspended`);
145 |       await loadUsers();
146 |       setActionDialog({ open: false, type: null });
147 |     } catch (error) {
148 |       console.error("Error suspending user:", error);
149 |       toast.error("Failed to suspend user");
150 |     }
151 |   };
152 |
153 |   const handleBan = async (user: User) => {
154 |     try {
155 |       await banUser(user.uid, "Banned by admin");
156 |       if (currentAdmin?.email) {
157 |         await logAdminAction(currentAdmin.email, "ban_user", {
158 |           userId: user.uid,
159 |           userName: user.name,
160 |           userEmail: user.email
161 |         });
162 |       }
163 |       toast.success(`User ${user.name || user.email} has been banned`);
164 |       await loadUsers();
165 |       setActionDialog({ open: false, type: null });
166 |     } catch (error) {
167 |       console.error("Error banning user:", error);
168 |       toast.error("Failed to ban user");
169 |     }
170 |   };
171 |
172 |   const handleUnban = async (user: User) => {
173 |     try {
174 |       await unbanUser(user.uid);
175 |       toast.success(`User ${user.name || user.email} has been unbanned`);
176 |       await loadUsers();
177 |       setActionDialog({ open: false, type: null });
178 |     } catch (error) {
179 |       console.error("Error unbanning user:", error);
180 |       toast.error("Failed to unban user");
181 |     }
182 |   };
183 |
184 |   const handleUnsuspend = async (user: User) => {
185 |     try {
186 |       await unsuspendUser(user.uid);
187 |       toast.success(`User ${user.name || user.email} has been unsuspended`);
188 |       await loadUsers();
189 |       setActionDialog({ open: false, type: null });
190 |     } catch (error) {
191 |       console.error("Error unsuspending user:", error);
192 |       toast.error("Failed to unsuspend user");
193 |     }
194 |   };
195 |
196 |   const handleDelete = async (user: User) => {
197 |     try {
198 |       await deleteUser(user.uid);
199 |       if (currentAdmin?.email) {
200 |         await logAdminAction(currentAdmin.email, "delete_user", {
201 |           userId: user.uid,
202 |           userName: user.name,
203 |           userEmail: user.email
204 |         });
205 |       }
206 |       toast.success(`User ${user.name || user.email} has been deleted`);
207 |       await loadUsers();
208 |       setActionDialog({ open: false, type: null });
```

```
209 |     } catch (error) {
210 |       console.error("Error deleting user:", error);
211 |       toast.error("Failed to delete user");
212 |     }
213 |   };
214 |
215 |   const handleViewReports = async (user: User) => {
216 |     try {
217 |       const reports = await getUserReports(user.uid);
218 |       if (reports.length === 0) {
219 |         toast.info("No reports found for this user");
220 |       } else {
221 |         navigate(`/admin/moderation?userId=${user.uid}`);
222 |       }
223 |     } catch (error) {
224 |       console.error("Error loading reports:", error);
225 |       toast.error("Failed to load reports");
226 |     }
227 |   };
228 |
229 |   const handlePromoteToAdmin = async (user: User) => {
230 |     if (!user.email) {
231 |       toast.error("User email is required to promote to admin");
232 |       return;
233 |     }
234 |     try {
235 |       await promoteUserToAdmin(user.email);
236 |       if (currentAdmin?.email) {
237 |         await logAdminAction(currentAdmin.email, "promote_admin", {
238 |           promotedEmail: user.email,
239 |           promotedUserId: user.uid,
240 |           promotedUserName: user.name
241 |         });
242 |       }
243 |       toast.success(`${user.name || user.email} has been promoted to admin`);
244 |       await loadUsers();
245 |       setActionDialog({ open: false, type: null });
246 |     } catch (error) {
247 |       console.error("Error promoting user:", error);
248 |       toast.error("Failed to promote user to admin");
249 |     }
250 |   };
251 |
252 |   const handleDemoteAdmin = async (user: User) => {
253 |     if (!user.email) {
254 |       toast.error("User email is required to demote admin");
255 |       return;
256 |     }
257 |     try {
258 |       await demoteAdmin(user.email);
259 |       if (currentAdmin?.email) {
260 |         await logAdminAction(currentAdmin.email, "demote_admin", {
261 |           demotedEmail: user.email,
262 |           demotedUserId: user.uid,
263 |           demotedUserName: user.name
264 |         });
265 |       }
266 |       toast.success(`${user.name || user.email} has been demoted from admin`);
267 |       await loadUsers();
268 |       setActionDialog({ open: false, type: null });
269 |     } catch (error) {
270 |       console.error("Error demoting admin:", error);
271 |       toast.error("Failed to demote admin");
272 |     }
273 |   };
274 |
275 |   const handleChangeUsername = async (user: User) => {
276 |     if (!currentAdmin?.uid || !currentAdmin?.displayName) {
277 |       toast.error("Admin information is required");
278 |       return;
279 |     }
```

```
    try {
      await requestUsernameChange(
        user.uid,
        "Inappropriate username - please update to an appropriate username",
        currentAdmin.uid,
        currentAdmin.displayName
      );
      if (currentAdmin.email) {
        await logAdminAction(currentAdmin.email, "change_username", {
          userId: user.uid,
          userName: user.name,
          userEmail: user.email,
          oldUsername: user.username || user.name
        });
      }
      toast.success(`Username changed for ${user.name || user.email}. User has been notified.`);
      await loadUsers();
      setActionDialog({ open: false, type: null });
    } catch (error) {
      console.error("Error changing username:", error);
      toast.error("Failed to change username");
    }
  };

  const getStatusBadge = (user: User) => {
    const status = user.status || "active";

    if (status === "banned") {
      return <Badge variant="destructive">Banned</Badge>;
    }
    if (status === "suspended") {
      const isExpired = user.suspendedUntil && user.suspendedUntil < Date.now();
      return (
        <Badge variant={isExpired ? "secondary" : "warning"}>
          {isExpired ? "Suspension Expired" : "Suspended"}
        </Badge>
      );
    }
    return <Badge variant="success">Active</Badge>;
  };

  const formatDate = (timestamp?: number) => {
    if (!timestamp) return "N/A";
    return new Date(timestamp).toLocaleDateString();
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-64">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
        <div>
          <div>
            <Button variant="ghost" size="sm" onClick={() => navigate("/admin/dashboard")}>
              !• Back to Dashboard
            </Button>
          </div>
          <h1 className="text-3xl font-bold">User Management</h1>
          <p className="text-muted-foreground mt-1">
            Manage all users, view profiles, and moderate accounts
          </p>
        </div>
        <div>
          Total: {filteredUsers.length} users
```

```
        </div>
      </div>

      {/* Filters */}
      <Card className="p-4">
        <div className="flex flex-col sm:flex-row gap-4">
          <div className="flex-1 relative">
            <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-
foreground" style={{ fo
              <Input
                placeholder="Search by name, email, or ID..."
                value={searchQuery}
                onChange={(e) => setSearchQuery(e.target.value)}
                className="pl-10"
              />
            </div>
            <div className="flex gap-2">
              <Button
                variant={statusFilter === "all" ? "default" : "outline"}
                onClick={() => setStatusFilter("all")}
              >
                All
              </Button>
              <Button
                variant={statusFilter === "active" ? "default" : "outline"}
                onClick={() => setStatusFilter("active")}
              >
                Active
              </Button>
              <Button
                variant={statusFilter === "suspended" ? "default" : "outline"}
                onClick={() => setStatusFilter("suspended")}
              >
                Suspended
              </Button>
              <Button
                variant={statusFilter === "banned" ? "default" : "outline"}
                onClick={() => setStatusFilter("banned")}
              >
                Banned
              </Button>
            </div>
          </div>
        </Card>

        {/* Users Table */}
        <Card>
          <div className="overflow-x-auto">
            <Table>
              <TableHeader>
                <TableRow>
                  <TableHead>User</TableHead>
                  <TableHead>Email</TableHead>
                  <TableHead>Status</TableHead>
                  <TableHead>Joined</TableHead>
                  <TableHead>Last Activity</TableHead>
                  <TableHead className="text-right">Actions</TableHead>
                </TableRow>
              </TableHeader>
              <TableBody>
                {filteredUsers.length === 0 ? (
                  <TableRow>
                    <TableCell colSpan={6} className="text-center py-8 text-muted-foreground">
                      No users found
                    </TableCell>
                  </TableRow>
                ) : (
                  filteredUsers.map((user) => (
                    <TableRow key={user.uid}>
                      <TableCell>
                        <div className="flex items-center gap-3">
                          <Avatar className="h-10 w-10">
```

```
422 |                         <AvatarImage src={user.photoURL} />
423 |                         <AvatarFallback>
424 |                           {user.name?.[0]?.toUpperCase() || user.email?.[0]?.toUpperCase() ||
"425 |                         </AvatarFallback>
426 |                       </Avatar>
427 |                       <div>
428 |                         <button
429 |                           onClick={() => {
430 |                             setSelectedUserId(user.uid);
431 |                             setUserDetailOpen(true);
432 |                           }}
433 |                           className="font-medium hover:underline text-left"
434 |                         >
435 |                           {user.name || "No name"}
436 |                         </button>
437 |                         <p className="text-xs text-muted-foreground">{user.uid}</p>
438 |                       </div>
439 |                     </div>
440 |                   </TableCell>
441 |                   <TableCell>
442 |                     <div className="flex items-center gap-2">
443 |                       {user.email || "N/A"}
444 |                       {adminStatuses[user.uid] && (
445 |                         <Badge variant="default" className="ml-2">
446 |                           <AdminPanelSettingsIcon className="mr-1" style={{ fontSize: 12 }} />
447 |                           Admin
448 |                         </Badge>
449 |                       )}
450 |                     </div>
451 |                   </TableCell>
452 |                   <TableCell>{getStatusBadge(user)}</TableCell>
453 |                   <TableCell>{formatDate(user.createdAt)}</TableCell>
454 |                   <TableCell>{formatDate(user.timestamp)}</TableCell>
455 |                   <TableCell className="text-right">
456 |                     <DropdownMenu>
457 |                       <DropdownMenuTrigger asChild>
458 |                         <Button variant="ghost" size="icon">
459 |                           <MoreVertIcon style={{ fontSize: 20 }} />
460 |                         </Button>
461 |                       </DropdownMenuTrigger>
462 |                       <DropdownMenuContent align="end">
463 |                         <DropdownMenuItem onClick={() => handleViewReports(user)}>
464 |                           <ReportProblemIcon className="mr-2" style={{ fontSize: 18 }} />
465 |                           View Reports
466 |                         </DropdownMenuItem>
467 |                         {!adminStatuses[user.uid] && user.email && (
468 |                           <DropdownMenuItem
469 |                             onClick={() => {
470 |                               setSelectedUser(user);
471 |                               setActionDialog({ open: true, type: "promote" });
472 |                             }}
473 |                             className="text-primary"
474 |                           >
475 |                             <AdminPanelSettingsIcon className="mr-2" style={{ fontSize: 18 }} /
>476 |                             Promote to Admin
477 |                           </DropdownMenuItem>
478 |                         )}
479 |                         {adminStatuses[user.uid] && user.email && (
480 |                           <DropdownMenuItem
481 |                             onClick={() => {
482 |                               setSelectedUser(user);
483 |                               setActionDialog({ open: true, type: "demote" });
484 |                             }}
485 |                             className="text-warning"
486 |                           >
487 |                             <PersonRemoveIcon className="mr-2" style={{ fontSize: 18 }} />
488 |                             Demote from Admin
489 |                           </DropdownMenuItem>
490 |                         )}
491 |                         {user.status === "active" && (
492 |                           <>
```

```
                            <DropdownMenuItem
                              onClick={() => {
                                setSelectedUser(user);
                                setActionDialog({ open: true, type: "suspend" });
                              }}
                            >
                              <BlockIcon className="mr-2" style={{ fontSize: 18 }} />
                              Suspend
                            </DropdownMenuItem>
                            <DropdownMenuItem
                              onClick={() => {
                                setSelectedUser(user);
                                setActionDialog({ open: true, type: "ban" });
                              }}
                              className="text-destructive"
                            >
                              <BlockIcon className="mr-2" style={{ fontSize: 18 }} />
                              Ban
                            </DropdownMenuItem>
                          </>
                        )}
                        {user.status === "suspended" && (
                          <DropdownMenuItem
                            onClick={() => {
                              setSelectedUser(user);
                              setActionDialog({ open: true, type: "unsuspend" });
                            }}
                          >
                            <CheckCircleIcon className="mr-2" style={{ fontSize: 18 }} />
                            Unsuspend
                          </DropdownMenuItem>
                        )}
                        {user.status === "banned" && (
                          <DropdownMenuItem
                            onClick={() => {
                              setSelectedUser(user);
                              setActionDialog({ open: true, type: "unban" });
                            }}
                          >
                            <CheckCircleIcon className="mr-2" style={{ fontSize: 18 }} />
                            Unban
                          </DropdownMenuItem>
                        )}
                        <DropdownMenuItem
                          onClick={() => {
                            setSelectedUser(user);
                            setActionDialog({ open: true, type: "changeUsername" });
                          }}
                          className="text-warning"
                        >
                          <EditIcon className="mr-2" style={{ fontSize: 18 }} />
                          Change Username
                        </DropdownMenuItem>
                        <DropdownMenuItem
                          onClick={() => {
                            setSelectedUser(user);
                            setActionDialog({ open: true, type: "delete" });
                          }}
                          className="text-destructive"
                        >
                          <DeleteIcon className="mr-2" style={{ fontSize: 18 }} />
                          Delete
                        </DropdownMenuItem>
                      </DropdownMenuContent>
                    </DropdownMenu>
                  </TableCell>
                </TableRow>
              ))
            )}
          </TableBody>
        </Table>
```

```
564 |         </div>
565 |       </Card>
566 |
567 |       {/* Action Dialogs */}
568 |       <AlertDialog open={actionDialog.open && actionDialog.type === "suspend"}>
569 |         <AlertDialogContent>
570 |           <AlertDialogHeader>
571 |             <AlertDialogTitle>Suspend User</AlertDialogTitle>
572 |             <AlertDialogDescription>
573 |               Are you sure you want to suspend {selectedUser?.name || selectedUser?.email}?
574 |               The user will be unable to access the app for 7 days.
575 |             </AlertDialogDescription>
576 |           </AlertDialogHeader>
577 |           <AlertDialogFooter>
578 |             <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
579 |               Cancel
580 |             </AlertDialogCancel>
581 |             <AlertDialogAction onClick={() => selectedUser && handleSuspend(selectedUser)}>
582 |               Suspend
583 |             </AlertDialogAction>
584 |           </AlertDialogFooter>
585 |         </AlertDialogContent>
586 |       </AlertDialog>
587 |
588 |       <AlertDialog open={actionDialog.open && actionDialog.type === "ban"}>
589 |         <AlertDialogContent>
590 |           <AlertDialogHeader>
591 |             <AlertDialogTitle>Ban User</AlertDialogTitle>
592 |             <AlertDialogDescription>
593 |               Are you sure you want to permanently ban {selectedUser?.name ||
selectedUser?.email}? This action cannot be undone easily.
594 |
595 |             </AlertDialogDescription>
596 |           </AlertDialogHeader>
597 |           <AlertDialogFooter>
598 |             <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
599 |               Cancel
600 |             </AlertDialogCancel>
601 |             <AlertDialogAction onClick={() => selectedUser && handleBan(selectedUser)}>
602 |               Ban
603 |             </AlertDialogAction>
604 |           </AlertDialogFooter>
605 |         </AlertDialogContent>
606 |       </AlertDialog>
607 |
608 |       <AlertDialog open={actionDialog.open && actionDialog.type === "unban"}>
609 |         <AlertDialogContent>
610 |           <AlertDialogHeader>
611 |             <AlertDialogTitle>Unban User</AlertDialogTitle>
612 |             <AlertDialogDescription>
613 |               Are you sure you want to unban {selectedUser?.name || selectedUser?.email}?
614 |             </AlertDialogDescription>
615 |           </AlertDialogHeader>
616 |           <AlertDialogFooter>
617 |             <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
618 |               Cancel
619 |             </AlertDialogCancel>
620 |             <AlertDialogAction onClick={() => selectedUser && handleUnban(selectedUser)}>
621 |               Unban
622 |             </AlertDialogAction>
623 |           </AlertDialogFooter>
624 |         </AlertDialogContent>
625 |       </AlertDialog>
626 |
627 |       <AlertDialog open={actionDialog.open && actionDialog.type === "unsuspend"}>
628 |         <AlertDialogContent>
629 |           <AlertDialogHeader>
630 |             <AlertDialogTitle>Unsuspend User</AlertDialogTitle>
631 |             <AlertDialogDescription>
632 |               Are you sure you want to unsuspend {selectedUser?.name || selectedUser?.email}?
633 |             </AlertDialogDescription>
634 |           </AlertDialogHeader>
```

```
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedUser && handleUnsuspend(selectedUser)}>
              Unsuspend
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      <AlertDialog open={actionDialog.open && actionDialog.type === "delete"}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Delete User</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to permanently delete {selectedUser?.name ||
selectedUser?.email}? This will remove all user data including workouts, friends, and messages. This
action cannot be undone.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction
              onClick={() => selectedUser && handleDelete(selectedUser)}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Delete
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      <AlertDialog open={actionDialog.open && actionDialog.type === "promote"}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Promote to Admin</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to promote {selectedUser?.name || selectedUser?.email} to
admin? They will have full access to the admin dashboard and all administrative functions.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedUser && handlePromoteToAdmin(selectedUser)}
>
              Promote
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      <AlertDialog open={actionDialog.open && actionDialog.type === "demote"}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Demote from Admin</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to remove admin privileges from {selectedUser?.name ||
selectedUser?.email}? They will no longer have access to the admin dashboard.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedUser && handleDemoteAdmin(selectedUser)}>
              Demote
            </AlertDialogAction>
          </AlertDialogFooter>
```

```
        </AlertDialogContent>
      </AlertDialog>

      <AlertDialog open={actionDialog.open && actionDialog.type === "changeUsername"}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Change Username</AlertDialogTitle>
            <AlertDialogDescription>
              This will change {selectedUser?.name || selectedUser?.email}'s username to a
default format (user {selected... and send them a notification to update it. The user will be notified that their
username was changed due t...
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel onClick={() => setActionDialog({ open: false, type: null })}>
              Cancel
            </AlertDialogCancel>
            <AlertDialogAction onClick={() => selectedUser && handleChangeUsername(selectedUser)}
>             Change Username
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* User Detail Drawer */}
      <UserDetailDrawer
        userId={selectedUserId}
        open={userDetailOpen}
        onOpenChange={setUserDetailOpen}
      />
    </div>
  );
};

export default AdminUsers;

```

File: src/pages/AdminVenues.tsx

Lines: 682

```
  1 | // Admin Venues Management Page
  2 | import { useState, useEffect } from "react";
  3 | import { Card } from "@/components/ui/card";
  4 | import { Button } from "@/components/ui/button";
  5 | import { Input } from "@/components/ui/input";
  6 | import { Badge } from "@/components/ui/badge";
  7 | import { Label } from "@/components/ui/label";
  8 | import {
  9 |   Table,
 10 |   TableBody,
 11 |   TableCell,
 12 |   TableHead,
 13 |   TableHeader,
 14 |   TableRow,
 15 | } from "@/components/ui/table";
 16 | import {
 17 |   AlertDialog,
 18 |   AlertDialogAction,
 19 |   AlertDialogCancel,
 20 |   AlertDialogContent,
 21 |   AlertDialogDescription,
 22 |   AlertDialogFooter,
 23 |   AlertDialogHeader,
 24 |   AlertDialogTitle,
 25 | } from "@/components/ui/alert-dialog";
 26 | import {
 27 |   Dialog,
 28 |   DialogContent,
 29 |   DialogDescription,
 30 |   DialogHeader,
 31 |   DialogTitle,
 32 | } from "@/components/ui/dialog";
 33 | import {
 34 |   Select,
 35 |   SelectContent,
 36 |   SelectItem,
 37 |   SelectTrigger,
 38 |   SelectValue,
 39 | } from "@/components/ui/select";
 40 | import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
 41 | import { useAuth } from "@/hooks/useAuth";
 42 | import { useNavigate } from "react-router-dom";
 43 | import { toast } from "sonner";
 44 | import LocationOnIcon from "@mui/icons-material/LocationOn";
 45 | import AddIcon from "@mui/icons-material/Add";
 46 | import DeleteIcon from "@mui/icons-material/Delete";
 47 | import SearchIcon from "@mui/icons-material/Search";
 48 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
 49 | import CancelIcon from "@mui/icons-material/Cancel";
 50 | import {
 51 |   getAllVenues,
 52 |   addVenue,
 53 |   updateVenue,
 54 |   deleteVenue,
 55 |   Venue,
 56 |   VenueCategory,
 57 | } from "@/services/venueService";
 58 | import {
 59 |   listenToVenueRequests,
 60 |   updateVenueRequestStatus,
 61 |   VenueRequest,
 62 | } from "@/services/venueRequestService";
 63 | import { getUserData } from "@/services/authService";
 64 |
 65 | const AdminVenues = () => {
 66 |   const navigate = useNavigate();
```

```
 67 |   const { user: currentAdmin } = useAuth();
 68 |   const [venues, setVenues] = useState<Venue[]>([]);
 69 |   const [venueRequests, setVenueRequests] = useState<VenueRequest[]>([]);
 70 |   const [loading, setLoading] = useState(true);
 71 |   const [searchQuery, setSearchQuery] = useState("");
 72 |   const [filteredVenues, setFilteredVenues] = useState<Venue[]>([]);
 73 |   const [selectedVenue, setSelectedVenue] = useState<Venue | null>(null);
 74 |   const [deleteDialog, setDeleteDialog] = useState(false);
 75 |   const [addDialog, setAddDialog] = useState(false);
 76 |   const [editDialog, setEditDialog] = useState(false);
 77 |   const [userData, setUserData] = useState<Record<string, any>>({});
 78 |
 79 |   // Form state
 80 |   const [formData, setFormData] = useState({
 81 |     name: "",
 82 |     description: "",
 83 |     lat: "",
 84 |     lng: "",
 85 |     radius: "",
 86 |     category: "park" as VenueCategory,
 87 |     city: "",
 88 |   });
 89 |
 90 |   useEffect(() => {
 91 |     loadVenues();
 92 |     loadVenueRequests();
 93 |   }, []);
 94 |
 95 |   useEffect(() => {
 96 |     filterVenues();
 97 |   }, [venues, searchQuery]);
 98 |
 99 |   const loadVenues = async () => {
100 |     try {
101 |       setLoading(true);
102 |       const venuesList = await getAllVenues();
103 |       setVenues(venuesList);
104 |       setFilteredVenues(venuesList);
105 |     } catch (error) {
106 |       console.error("Error loading venues:", error);
107 |       toast.error("Failed to load venues");
108 |     } finally {
109 |       setLoading(false);
110 |     }
111 |   };
112 |
113 |   const loadVenueRequests = () => {
114 |     const unsubscribe = listenToVenueRequests((requests) => {
115 |       setVenueRequests(requests);
116 |
117 |       // Load user data for requests
118 |       const userIds = new Set<string>();
119 |       requests.forEach((request) => {
120 |         userIds.add(request.userId);
121 |       });
122 |
123 |       Promise.all(
124 |         Array.from(userIds).map(async (userId) => {
125 |           try {
126 |             const data = await getUserData(userId);
127 |             if (data) {
128 |               setUserData((prev) => ({ ...prev, [userId]: data }));
129 |             }
130 |           } catch (error) {
131 |             console.error(`Error loading user ${userId}:`, error);
132 |           }
133 |         })
134 |       );
135 |     });
136 |
137 |     return unsubscribe;
```

```
  };

  const filterVenues = () => {
    if (!searchQuery.trim()) {
      setFilteredVenues(venues);
      return;
    }

    const lowerQuery = searchQuery.toLowerCase();
    const filtered = venues.filter(
      (venue) =>
        venue.name.toLowerCase().includes(lowerQuery) ||
        venue.description.toLowerCase().includes(lowerQuery) ||
        venue.city.toLowerCase().includes(lowerQuery)
    );
    setFilteredVenues(filtered);
  };

  const handleAddVenue = async () => {
    if (!formData.name || !formData.city || !formData.lat || !formData.lng) {
      toast.error("Please fill in all required fields");
      return;
    }

    try {
      const lat = parseFloat(formData.lat);
      const lng = parseFloat(formData.lng);
      const radius = parseFloat(formData.radius) || 1000;

      if (isNaN(lat) || isNaN(lng) || isNaN(radius)) {
        toast.error("Please enter valid numbers for coordinates and radius");
        return;
      }

      await addVenue(
        {
          name: formData.name,
          description: formData.description,
          lat,
          lng,
          radius,
          category: formData.category,
          city: formData.city,
        },
        currentAdmin?.uid || "admin"
      );

      toast.success("Venue added successfully");
      setAddDialog(false);
      resetForm();
      loadVenues();
    } catch (error: any) {
      toast.error(error.message || "Failed to add venue");
    }
  };

  const handleEditVenue = async () => {
    if (!selectedVenue || !formData.name || !formData.city) {
      return;
    }

    try {
      const lat = parseFloat(formData.lat);
      const lng = parseFloat(formData.lng);
      const radius = parseFloat(formData.radius) || 1000;

      await updateVenue(selectedVenue.id, {
        name: formData.name,
        description: formData.description,
        lat,
        lng,
```

```
209 |         radius,
210 |         category: formData.category,
211 |         city: formData.city,
212 |       });
213 |
214 |       toast.success("Venue updated successfully");
215 |       setEditDialog(false);
216 |       resetForm();
217 |       setSelectedVenue(null);
218 |       loadVenues();
219 |     } catch (error: any) {
220 |       toast.error(error.message || "Failed to update venue");
221 |     }
222 |   };
223 |
224 |   const handleDeleteVenue = async () => {
225 |     if (!selectedVenue) return;
226 |
227 |     try {
228 |       await deleteVenue(selectedVenue.id);
229 |       toast.success("Venue deleted successfully");
230 |       setDeleteDialog(false);
231 |       setSelectedVenue(null);
232 |       loadVenues();
233 |     } catch (error: any) {
234 |       toast.error(error.message || "Failed to delete venue");
235 |     }
236 |   };
237 |
238 |   const handleApproveRequest = async (request: VenueRequest) => {
239 |     try {
240 |       await updateVenueRequestStatus(request.id, "approved", currentAdmin?.uid || "admin");
241 |       toast.success("Request approved");
242 |     } catch (error: any) {
243 |       toast.error(error.message || "Failed to approve request");
244 |     }
245 |   };
246 |
247 |   const handleRejectRequest = async (request: VenueRequest) => {
248 |     try {
249 |       await updateVenueRequestStatus(request.id, "rejected", currentAdmin?.uid || "admin");
250 |       toast.success("Request rejected");
251 |     } catch (error: any) {
252 |       toast.error(error.message || "Failed to reject request");
253 |     }
254 |   };
255 |
256 |   const resetForm = () => {
257 |     setFormData({
258 |       name: "",
259 |       description: "",
260 |       lat: "",
261 |       lng: "",
262 |       radius: "",
263 |       category: "park",
264 |       city: "",
265 |     });
266 |   };
267 |
268 |   const openEditDialog = (venue: Venue) => {
269 |     setSelectedVenue(venue);
270 |     setFormData({
271 |       name: venue.name,
272 |       description: venue.description,
273 |       lat: venue.lat.toString(),
274 |       lng: venue.lng.toString(),
275 |       radius: venue.radius.toString(),
276 |       category: venue.category,
277 |       city: venue.city,
278 |     });
279 |     setEditDialog(true);
```

```
  };

  const getCategoryBadge = (category: VenueCategory) => {
    const colors: Record<string, any> = {
      park: "success",
      university: "info",
      commercial: "default",
      sports: "warning",
      other: "secondary",
    };
    return <Badge variant={colors[category] || "default"}>{category}</Badge>;
  };

  const getStatusBadge = (status: string) => {
    const colors: Record<string, any> = {
      pending: "warning",
      approved: "success",
      rejected: "destructive",
    };
    return <Badge variant={colors[status] || "default"}>{status}</Badge>;
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center h-64">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      {/* Header */}
      <div>
        <div className="flex items-center gap-2 mb-2">
          <Button variant="ghost" size="sm" onClick={() => navigate("/admin/dashboard")}>
            !• Back to Dashboard
          </Button>
        </div>
        <h1 className="text-3xl font-bold">Venues Management</h1>
        <p className="text-muted-foreground mt-1">
          Manage venues and review user requests
        </p>
      </div>

      <Tabs defaultValue="venues" className="space-y-6">
        <TabsList>
          <TabsTrigger value="venues">Venues ({venues.length})</TabsTrigger>
          <TabsTrigger value="requests">
            Requests ({venueRequests.filter((r) => r.status === "pending").length})
          </TabsTrigger>
        </TabsList>

        <TabsContent value="venues" className="space-y-6">
          {/* Search and Add */}
          <Card className="p-4">
            <div className="flex flex-col sm:flex-row gap-4">
              <div className="relative flex-1">
                <SearchIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={{
                  <Input
                    placeholder="Search venues..."
                    value={searchQuery}
                    onChange={(e) => setSearchQuery(e.target.value)}
                    className="pl-10"
                  />
                </div>
                <Button onClick={() => setAddDialog(true)}>
                  <AddIcon className="mr-2" style={{ fontSize: 20 }} />
                  Add Venue
                </Button>
              </div>
```

```
          </Card>

          {/* Venues Table */}
          <Card>
            <Table>
              <TableHeader>
                <TableRow>
                  <TableHead>Name</TableHead>
                  <TableHead>City</TableHead>
                  <TableHead>Category</TableHead>
                  <TableHead>Coordinates</TableHead>
                  <TableHead>Radius</TableHead>
                  <TableHead>Actions</TableHead>
                </TableRow>
              </TableHeader>
              <TableBody>
                {filteredVenues.length === 0 ? (
                  <TableRow>
                    <TableCell colSpan={6} className="text-center py-8 text-muted-foreground">
                      No venues found
                    </TableCell>
                  </TableRow>
                ) : (
                  filteredVenues.map((venue) => (
                    <TableRow key={venue.id}>
                      <TableCell className="font-medium">{venue.name}</TableCell>
                      <TableCell>{venue.city}</TableCell>
                      <TableCell>{getCategoryBadge(venue.category)}</TableCell>
                      <TableCell className="text-sm">
                        {venue.lat.toFixed(4)}, {venue.lng.toFixed(4)}
                      </TableCell>
                      <TableCell>{venue.radius}m</TableCell>
                      <TableCell>
                        <div className="flex gap-2">
                          <Button
                            variant="outline"
                            size="sm"
                            onClick={() => openEditDialog(venue)}
                          >
                            Edit
                          </Button>
                          <Button
                            variant="destructive"
                            size="sm"
                            onClick={() => {
                              setSelectedVenue(venue);
                              setDeleteDialog(true);
                            }}
                          >
                            <DeleteIcon style={{ fontSize: 16 }} />
                          </Button>
                        </div>
                      </TableCell>
                    </TableRow>
                  ))
                )}
              </TableBody>
            </Table>
          </Card>
        </TabsContent>

        <TabsContent value="requests" className="space-y-6">
          <Card>
            <Table>
              <TableHeader>
                <TableRow>
                  <TableHead>Venue Name</TableHead>
                  <TableHead>Location</TableHead>
                  <TableHead>Requested By</TableHead>
                  <TableHead>Date</TableHead>
                  <TableHead>Status</TableHead>
```

```
                  <TableHead>Actions</TableHead>
                </TableRow>
              </TableHeader>
              <TableBody>
                {venueRequests.length === 0 ? (
                  <TableRow>
                    <TableCell colSpan={6} className="text-center py-8 text-muted-foreground">
                      No venue requests
                    </TableCell>
                  </TableRow>
                ) : (
                  venueRequests.map((request) => (
                    <TableRow key={request.id}>
                      <TableCell className="font-medium">{request.venueName}</TableCell>
                      <TableCell>{request.location}</TableCell>
                      <TableCell>
                        {userData[request.userId]?.username || userData[request.userId]?.name ||
"Unknown"}
                      </TableCell>
                      <TableCell>
                        {new Date(request.createdAt).toLocaleDateString()}
                      </TableCell>
                      <TableCell>{getStatusBadge(request.status)}</TableCell>
                      <TableCell>
                        {request.status === "pending" && (
                          <div className="flex gap-2">
                            <Button
                              variant="outline"
                              size="sm"
                              onClick={() => handleApproveRequest(request)}
                            >
                              <CheckCircleIcon className="mr-1" style={{ fontSize: 16 }} />
                              Approve
                            </Button>
                            <Button
                              variant="destructive"
                              size="sm"
                              onClick={() => handleRejectRequest(request)}
                            >
                              <CancelIcon className="mr-1" style={{ fontSize: 16 }} />
                              Reject
                            </Button>
                          </div>
                        )}
                      </TableCell>
                    </TableRow>
                  ))
                )}
              </TableBody>
            </Table>
          </Card>
        </TabsContent>
      </Tabs>

      {/* Add Venue Dialog */}
      <Dialog open={addDialog} onOpenChange={setAddDialog}>
        <DialogContent className="max-w-2xl max-h-[90vh] overflow-y-auto">
          <DialogHeader>
            <DialogTitle>Add New Venue</DialogTitle>
            <DialogDescription>
              Add a new venue to the list
            </DialogDescription>
          </DialogHeader>
          <div className="space-y-4">
            <div>
              <Label>Name *</Label>
              <Input
                value={formData.name}
                onChange={(e) => setFormData({ ...formData, name: e.target.value })}
                placeholder="e.g., Rizal Park"
              />
            </div>
```

```
493 |             <div>
494 |               <Label>Description</Label>
495 |               <Input
496 |                 value={formData.description}
497 |                 onChange={(e) => setFormData({ ...formData, description: e.target.value })}
498 |                 placeholder="Description of the venue"
499 |               />
500 |             </div>
501 |             <div className="grid grid-cols-2 gap-4">
502 |               <div>
503 |                 <Label>Latitude *</Label>
504 |                 <Input
505 |                   type="number"
506 |                   step="any"
507 |                   value={formData.lat}
508 |                   onChange={(e) => setFormData({ ...formData, lat: e.target.value })}
509 |                   placeholder="14.5832"
510 |                 />
511 |               </div>
512 |               <div>
513 |                 <Label>Longitude *</Label>
514 |                 <Input
515 |                   type="number"
516 |                   step="any"
517 |                   value={formData.lng}
518 |                   onChange={(e) => setFormData({ ...formData, lng: e.target.value })}
519 |                   placeholder="120.9794"
520 |                 />
521 |               </div>
522 |             </div>
523 |             <div className="grid grid-cols-2 gap-4">
524 |               <div>
525 |                 <Label>Radius (meters)</Label>
526 |                 <Input
527 |                   type="number"
528 |                   value={formData.radius}
529 |                   onChange={(e) => setFormData({ ...formData, radius: e.target.value })}
530 |                   placeholder="1000"
531 |                 />
532 |               </div>
533 |               <div>
534 |                 <Label>Category</Label>
535 |                 <Select
536 |                   value={formData.category}
537 |                   onValueChange={(value) => setFormData({ ...formData, category: value as
VenueCategory })}
538 |                 >
539 |                   <SelectTrigger>
540 |                     <SelectValue />
541 |                   </SelectTrigger>
542 |                   <SelectContent>
543 |                     <SelectItem value="park">Park</SelectItem>
544 |                     <SelectItem value="university">University</SelectItem>
545 |                     <SelectItem value="commercial">Commercial</SelectItem>
546 |                     <SelectItem value="sports">Sports</SelectItem>
547 |                     <SelectItem value="other">Other</SelectItem>
548 |                   </SelectContent>
549 |                 </Select>
550 |               </div>
551 |             </div>
552 |             <div>
553 |               <Label>City *</Label>
554 |               <Input
555 |                 value={formData.city}
556 |                 onChange={(e) => setFormData({ ...formData, city: e.target.value })}
557 |                 placeholder="e.g., Manila"
558 |               />
559 |             </div>
560 |             <div className="flex gap-2 justify-end">
561 |               <Button variant="outline" onClick={() => setAddDialog(false)}>
562 |                 Cancel
563 |               </Button>
```

```
564 |               <Button onClick={handleAddVenue}>Add Venue</Button>
565 |             </div>
566 |           </div>
567 |         </DialogContent>
568 |       </Dialog>
569 |
570 |       {/* Edit Venue Dialog */}
571 |       <Dialog open={editDialog} onOpenChange={setEditDialog}>
572 |         <DialogContent className="max-w-2xl max-h-[90vh] overflow-y-auto">
573 |           <DialogHeader>
574 |             <DialogTitle>Edit Venue</DialogTitle>
575 |             <DialogDescription>
576 |               Update venue information
577 |             </DialogDescription>
578 |           </DialogHeader>
579 |           <div className="space-y-4">
580 |             <div>
581 |               <Label>Name *</Label>
582 |               <Input
583 |                 value={formData.name}
584 |                 onChange={(e) => setFormData({ ...formData, name: e.target.value })}
585 |               />
586 |             </div>
587 |             <div>
588 |               <Label>Description</Label>
589 |               <Input
590 |                 value={formData.description}
591 |                 onChange={(e) => setFormData({ ...formData, description: e.target.value })}
592 |               />
593 |             </div>
594 |             <div className="grid grid-cols-2 gap-4">
595 |               <div>
596 |                 <Label>Latitude *</Label>
597 |                 <Input
598 |                   type="number"
599 |                   step="any"
600 |                   value={formData.lat}
601 |                   onChange={(e) => setFormData({ ...formData, lat: e.target.value })}
602 |                 />
603 |               </div>
604 |               <div>
605 |                 <Label>Longitude *</Label>
606 |                 <Input
607 |                   type="number"
608 |                   step="any"
609 |                   value={formData.lng}
610 |                   onChange={(e) => setFormData({ ...formData, lng: e.target.value })}
611 |                 />
612 |               </div>
613 |             </div>
614 |             <div className="grid grid-cols-2 gap-4">
615 |               <div>
616 |                 <Label>Radius (meters)</Label>
617 |                 <Input
618 |                   type="number"
619 |                   value={formData.radius}
620 |                   onChange={(e) => setFormData({ ...formData, radius: e.target.value })}
621 |                 />
622 |               </div>
623 |               <div>
624 |                 <Label>Category</Label>
625 |                 <Select
626 |                   value={formData.category}
627 |                   onValueChange={(value) => setFormData({ ...formData, category: value as
VenueCategory })}
628 |                 >
629 |                   <SelectTrigger>
630 |                     <SelectValue />
631 |                   </SelectTrigger>
632 |                   <SelectContent>
633 |                     <SelectItem value="park">Park</SelectItem>
634 |                     <SelectItem value="university">University</SelectItem>
```

```
                    <SelectItem value="commercial">Commercial</SelectItem>
                    <SelectItem value="sports">Sports</SelectItem>
                    <SelectItem value="other">Other</SelectItem>
                  </SelectContent>
                </Select>
              </div>
            </div>
            <div>
              <Label>City *</Label>
              <Input
                value={formData.city}
                onChange={(e) => setFormData({ ...formData, city: e.target.value })}
              />
            </div>
            <div className="flex gap-2 justify-end">
              <Button variant="outline" onClick={() => setEditDialog(false)}>
                Cancel
              </Button>
              <Button onClick={handleEditVenue}>Update Venue</Button>
            </div>
          </div>
        </DialogContent>
      </Dialog>

      {/* Delete Confirmation */}
      <AlertDialog open={deleteDialog} onOpenChange={setDeleteDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Delete Venue</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to delete "{selectedVenue?.name}"? This action cannot be
undone.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel>Cancel</AlertDialogCancel>
            <AlertDialogAction onClick={handleDeleteVenue} className="bg-destructive text-
destructive-foreground">
              Delete
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>
    </div>
  );
};

export default AdminVenues;
```

File: src/pages/Chat.tsx

Lines: 934

```
import { useState, useRef, useEffect } from "react";
import { motion } from "framer-motion";
import { useNavigate, useLocation } from "react-router-dom";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import SendIcon from "@mui/icons-material/Send";
import MoreVertIcon from "@mui/icons-material/MoreVert";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import NavigationIcon from "@mui/icons-material/Navigation";
import StopIcon from "@mui/icons-material/Stop";
import PersonIcon from "@mui/icons-material/Person";
import NotificationsOffIcon from "@mui/icons-material/NotificationsOff";
import NotificationsIcon from "@mui/icons-material/Notifications";
import BlockIcon from "@mui/icons-material/Block";
import ReportIcon from "@mui/icons-material/Report";
import DeleteIcon from "@mui/icons-material/Delete";
import CloseIcon from "@mui/icons-material/Close";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import Avatar from "@mui/material/Avatar";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
  DialogDescription,
} from "@/components/ui/dialog";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuSeparator,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import {
  AlertDialog,
  AlertDialogAction,
  AlertDialogCancel,
  AlertDialogContent,
  AlertDialogDescription,
  AlertDialogFooter,
  AlertDialogHeader,
  AlertDialogTitle,
} from "@/components/ui/alert-dialog";
import { toast } from "sonner";
import { ProfileView } from "./ProfileView";
import { useUser } from "@/contexts/UserContext";
import { useAuth } from "@/hooks/useAuth";
import { useLocation as useLocationHook } from "@/hooks/useLocation";
import {
  listenToUserFriends,
  listenToFriendRequests,
  sendFriendRequest,
  acceptFriendRequest,
  declineFriendRequest,
} from "@/services/friendService";
import { listenToMessages, sendMessage, sendLocationMessage, markMessagesAsRead,
deleteConversation, Message as FirebaseMessage } from "@/services/chatService";
import { blockUser, reportUser, isUserBlocked } from "@/services/userService";
import { ReportUserModal } from "@/components/ReportUserModal";
import { generateDummyChatMessages, ENABLE_DUMMY_DATA } from "@/lib/dummyData";
import LocationSharingModal from "@/components/LocationSharingModalSimple";
import { openGoogleMapsNavigation } from "@/utils/navigation";
import { getUserData } from "@/services/authService";
import { getDisplayName } from "@/utils/anonymousName";

interface ChatUser {
```

```
  67 |   id: string; // Firebase UID
  68 |   name: string;
  69 |   avatar: string;
  70 | }
  71 |
  72 | const Chat = () => {
  73 |   const navigate = useNavigate();
  74 |   const location = useLocation();
  75 |   const messagesEndRef = useRef<HTMLDivElement>(null);
  76 |   const inputRef = useRef<HTMLTextAreaElement>(null);
  77 |   const { userProfile, setUserProfile } = useUser();
  78 |   const { user: currentUser } = useAuth();
  79 |
  80 |   // Get user from navigation state - required, no default
  81 |   const chatUser: ChatUser | null = location.state?.user || null;
  82 |
  83 |   // Redirect if no user provided
  84 |   useEffect(() => {
  85 |     if (!chatUser) {
  86 |       navigate("/messages");
  87 |     }
  88 |   }, [chatUser, navigate]);
  89 |
  90 |   const [messageInput, setMessageInput] = useState("");
  91 |   const [isTyping, setIsTyping] = useState(false);
  92 |   const [showProfile, setShowProfile] = useState(false);
  93 |   const [messages, setMessages] = useState<FirebaseMessage[]>([]);
  94 |   const [friends, setFriends] = useState<string[]>([]);
  95 |   const [friendRequests, setFriendRequests] = useState<{ incoming: string[]; outgoing: string[] }
>({ incoming: [], outgo...
  96 |
  97 |   // Location sharing modal state
  98 |   const [showLocationModal, setShowLocationModal] = useState(false);
  99 |   const [showConfirmModal, setShowConfirmModal] = useState(false);
 100 |   const [pendingLocation, setPendingLocation] = useState<{ lat: number; lng: number } |
null>(null);
 101 |
 102 |   // Menu state
 103 |   const [showReportModal, setShowReportModal] = useState(false);
 104 |   const [showBlockDialog, setShowBlockDialog] = useState(false);
 105 |   const [showDeleteDialog, setShowDeleteDialog] = useState(false);
 106 |   const [isMuted, setIsMuted] = useState(false);
 107 |   const [isBlocked, setIsBlocked] = useState(false);
 108 |   const [isMessageRequest, setIsMessageRequest] = useState(false);
 109 |   const [chatUserData, setChatUserData] = useState<{ username: string | null; activity: string |
null } | null>(null);
 110 |   const [displayName, setDisplayName] = useState<string>("");
 111 |
 112 |   // Fetch chat user data to get username and activity for display name
 113 |   useEffect(() => {
 114 |     if (!chatUser?.id) {
 115 |       setChatUserData(null);
 116 |       setDisplayName("");
 117 |       return;
 118 |     }
 119 |
 120 |     const fetchChatUserData = async () => {
 121 |       try {
 122 |         const userData = await getUserData(chatUser.id);
 123 |         const username = userData?.name || null; // name field contains username after profile
setup
 124 |         const activity = userData?.activity || null;
 125 |         setChatUserData({ username, activity });
 126 |         setDisplayName(getDisplayName(username, chatUser.id, activity));
 127 |       } catch (error) {
 128 |         console.error("Error fetching chat user data:", error);
 129 |         // Fallback to anonymous name if fetch fails
 130 |         setDisplayName(getDisplayName(null, chatUser.id, null));
 131 |       }
 132 |     };
 133 |
 134 |     fetchChatUserData();
 135 |   }, [chatUser?.id]);
 136 |
 137 |   // Get current user location when modal is open
```

```
138 |   const { location: currentLocation, isGettingLocation } = useLocationHook(
139 |     currentUser?.uid || null,
140 |     showLocationModal, // Only track when modal is open
141 |     true
142 |   );
143 |
144 |   // Listen to real-time messages from Firebase
145 |   useEffect(() => {
146 |     if (!currentUser?.uid || !chatUser?.id) return;
147 |
148 |     const unsubscribe = listenToMessages(
149 |       currentUser.uid,
150 |       chatUser.id,
151 |       (firebaseMessages) => {
152 |         // Add dummy messages if enabled and no real messages exist
153 |         if (ENABLE_DUMMY_DATA && firebaseMessages.length === 0) {
154 |           const dummyMessages = generateDummyChatMessages(currentUser.uid, chatUser.id);
155 |           setMessages(dummyMessages);
156 |         } else {
157 |           setMessages(firebaseMessages);
158 |         }
159 |         // Mark messages as read when viewing chat
160 |         markMessagesAsRead(currentUser.uid, chatUser.id).catch(console.error);
161 |       }
162 |     );
163 |
164 |     return () => unsubscribe();
165 |   }, [currentUser?.uid, chatUser?.id]);
166 |
167 |   // Listen to friends list from Firebase
168 |   useEffect(() => {
169 |     if (!currentUser?.uid) {
170 |       setFriends([]);
171 |       return;
172 |     }
173 |
174 |     const unsubscribe = listenToUserFriends(currentUser.uid, (friendIds) => {
175 |       setFriends(friendIds);
176 |     });
177 |
178 |     return () => unsubscribe();
179 |   }, [currentUser?.uid]);
180 |
181 |   // Listen to friend requests from Firebase
182 |   useEffect(() => {
183 |     if (!currentUser?.uid) {
184 |       setFriendRequests({ incoming: [], outgoing: [] });
185 |       return;
186 |     }
187 |
188 |     const unsubscribe = listenToFriendRequests(currentUser.uid, (requests) => {
189 |       setFriendRequests(requests);
190 |     });
191 |
192 |     return () => unsubscribe();
193 |   }, [currentUser?.uid]);
194 |
195 |   // Check if chat is muted (localStorage)
196 |   useEffect(() => {
197 |     if (!currentUser?.uid || !chatUser?.id) return;
198 |
199 |     const mutedChats = JSON.parse(localStorage.getItem("mutedChats") || "[]");
200 |     const conversationId = [currentUser.uid, chatUser.id].sort().join("_");
201 |     setIsMuted(mutedChats.includes(conversationId));
202 |   }, [currentUser?.uid, chatUser?.id]);
203 |
204 |   // Check if user is blocked
205 |   useEffect(() => {
206 |     const checkBlocked = async () => {
207 |       if (!currentUser?.uid || !chatUser?.id) return;
208 |       const blocked = await isUserBlocked(currentUser.uid, chatUser.id);
```

```
209 |       setIsBlocked(blocked);
210 |     };
211 |     checkBlocked();
212 |   }, [currentUser?.uid, chatUser?.id]);
213 |
214 |   // Determine friend status and check if this is a message request
215 |   useEffect(() => {
216 |     if (!chatUser || !currentUser?.uid) {
217 |       setIsMessageRequest(false);
218 |       return;
219 |     }
220 |
221 |     // It's a message request if:
222 |     // 1. Not friends
223 |     // 2. Has messages (conversation exists)
224 |     // When they accept, messages are marked as read, but it's still a request until they become
friends
225 |     const isNotFriend = !friends.includes(chatUser.id);
226 |     const hasMessages = messages.length > 0;
227 |     setIsMessageRequest(isNotFriend && hasMessages);
228 |   }, [chatUser, friends, messages.length, currentUser?.uid]);
229 |
230 |   const getFriendStatus = () => {
231 |     if (!chatUser) return "not_friends";
232 |
233 |     if (friends.includes(chatUser.id)) return "friends";
234 |     if (friendRequests.incoming.includes(chatUser.id)) return "request_received";
235 |     if (friendRequests.outgoing.includes(chatUser.id)) return "request_pending";
236 |     return "not_friends";
237 |   };
238 |
239 |   // Handle accepting message request
240 |   const handleAcceptMessageRequest = async () => {
241 |     if (!currentUser?.uid || !chatUser?.id) return;
242 |
243 |     // Mark messages as read to "accept" the request
244 |     try {
245 |       await markMessagesAsRead(currentUser.uid, chatUser.id);
246 |       setIsMessageRequest(false);
247 |       toast.success(`Accepted message from ${displayName || chatUser.name}`);
248 |     } catch (error) {
249 |       console.error("Error accepting message request:", error);
250 |       toast.error("Failed to accept message request");
251 |     }
252 |   };
253 |
254 |   // Handle declining message request (delete conversation)
255 |   const handleDeclineMessageRequest = async () => {
256 |     if (!currentUser?.uid || !chatUser?.id) return;
257 |
258 |     try {
259 |       await deleteConversation(currentUser.uid, chatUser.id);
260 |       setIsMessageRequest(false);
261 |       toast.success(`Declined message from ${displayName || chatUser.name}`);
262 |       navigate("/messages");
263 |     } catch (error) {
264 |       console.error("Error declining message request:", error);
265 |       toast.error("Failed to decline message request");
266 |     }
267 |   };
268 |
269 |   const maxCharacters = 500;
270 |
271 |   useEffect(() => {
272 |     scrollToBottom();
273 |   }, [messages]);
274 |
275 |   const scrollToBottom = () => {
276 |     messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
277 |   };
278 |
279 |   const getRelativeTime = (timestamp: number): string => {
```

```
    const now = Date.now();
    const diff = now - timestamp;
    const minutes = Math.floor(diff / (1000 * 60));
    const hours = Math.floor(diff / (1000 * 60 * 60));

    if (minutes < 1) return "Just now";
    if (minutes < 60) return `${minutes}m ago`;
    if (hours < 24) return `${hours}h ago`;

    const date = new Date(timestamp);
    return date.toLocaleTimeString('en-US', { hour: 'numeric', minute: '2-digit' });
  };

  const handleSendMessage = async () => {
    if (!messageInput.trim() || !currentUser?.uid || !chatUser?.id) return;

    // Check if blocked before attempting to send
    if (isBlocked) {
      toast.error("You cannot send messages. You have been blocked.");
      return;
    }

    try {
      await sendMessage(currentUser.uid, chatUser.id, messageInput.trim());
      setMessageInput("");
      toast.success("Message sent!");
    } catch (error) {
      console.error("Error sending message:", error);
      const errorMessage = error instanceof Error ? error.message : "Failed to send message. Please try again.";
      toast.error(errorMessage);
    }
  };

  const handleKeyPress = (e: React.KeyboardEvent) => {
    if (e.key === "Enter" && !e.shiftKey) {
      e.preventDefault();
      handleSendMessage();
    }
  };

  // Profile view handlers
  const handleAddFriend = async () => {
    if (!chatUser || !currentUser?.uid) return;

    try {
      await sendFriendRequest(currentUser.uid, chatUser.id);
      toast.success(`Friend request sent to ${displayName || chatUser.name}`);
      setShowProfile(false);
    } catch (error) {
      console.error("Error sending friend request:", error);
      toast.error("Failed to send friend request");
    }
  };

  const handleAcceptFriend = async () => {
    if (!chatUser || !currentUser?.uid) return;

    try {
      await acceptFriendRequest(currentUser.uid, chatUser.id);
      toast.success(`You are now friends with ${displayName || chatUser.name}`);
      setShowProfile(false);
    } catch (error) {
      console.error("Error accepting friend request:", error);
      toast.error("Failed to accept friend request");
    }
  };

  const handleDeclineFriend = async () => {
    if (!chatUser || !currentUser?.uid) return;

    try {
```

```
351 |       await declineFriendRequest(currentUser.uid, chatUser.id);
352 |       toast.success("Friend request declined");
353 |       setShowProfile(false);
354 |     } catch (error) {
355 |       console.error("Error declining friend request:", error);
356 |       toast.error("Failed to decline friend request");
357 |     }
358 |   };
359 |
360 |   const handleSendMessageFromProfile = () => {
361 |     setShowProfile(false);
362 |     // Already in chat, just close the profile
363 |   };
364 |
365 |   // Location sharing handlers
366 |   const handleShareLocation = () => {
367 |     if (!currentUser?.uid) {
368 |       toast.error("You must be logged in to share location");
369 |       return;
370 |     }
371 |
372 |     if (!chatUser?.id) {
373 |       toast.error("No chat user found");
374 |       return;
375 |     }
376 |
377 |     // Check if blocked before attempting to share location
378 |     if (isBlocked) {
379 |       toast.error("You cannot share location. You have been blocked.");
380 |       return;
381 |     }
382 |
383 |     if (!currentLocation) {
384 |       toast.error("Unable to get your location. Please check your location permissions.");
385 |       return;
386 |     }
387 |
388 |     // Show confirmation modal
389 |     setPendingLocation(currentLocation);
390 |     setShowLocationModal(false);
391 |     setShowConfirmModal(true);
392 |   };
393 |
394 |   const handleConfirmShare = async () => {
395 |     if (!currentUser?.uid || !chatUser?.id || !pendingLocation) return;
396 |
397 |     try {
398 |       // Send location message
399 |       await sendLocationMessage(
400 |         currentUser.uid,
401 |         chatUser.id,
402 |         pendingLocation
403 |       );
404 |
405 |       // Send text message with coordinates
406 |       await sendMessage(
407 |         currentUser.uid,
408 |         chatUser.id,
409 |         `Ø=ÜÍ My current location:\n${pendingLocation.lat.toFixed(6)},
${pendingLocation.lng.toFixed(6)}`
410 |       );
411 |
412 |       toast.success(`Location sent to ${displayName || chatUser.name}`);
413 |       setShowConfirmModal(false);
414 |       setPendingLocation(null);
415 |     } catch (error) {
416 |       console.error("'L Error sending location:", error);
417 |       toast.error("Failed to send location");
418 |     }
419 |   };
420 |
421 |   // Menu handlers
```

```
422 |   const handleToggleMute = () => {
423 |     if (!currentUser?.uid || !chatUser?.id) return;
424 |
425 |     const conversationId = [currentUser.uid, chatUser.id].sort().join("_");
426 |     const mutedChats = JSON.parse(localStorage.getItem("mutedChats") || "[]");
427 |
428 |     if (isMuted) {
429 |       const updated = mutedChats.filter((id: string) => id !== conversationId);
430 |       localStorage.setItem("mutedChats", JSON.stringify(updated));
431 |       setIsMuted(false);
432 |       toast.success(`Notifications enabled for ${displayName || chatUser.name}`);
433 |     } else {
434 |       mutedChats.push(conversationId);
435 |       localStorage.setItem("mutedChats", JSON.stringify(mutedChats));
436 |       setIsMuted(true);
437 |       toast.success(`Notifications muted for ${displayName || chatUser.name}`);
438 |     }
439 |   };
440 |
441 |   const handleBlockUser = async () => {
442 |     if (!currentUser?.uid || !chatUser?.id) return;
443 |
444 |     try {
445 |       await blockUser(currentUser.uid, chatUser.id);
446 |       setIsBlocked(true);
447 |       toast.success(`${displayName || chatUser.name} has been blocked`);
448 |       setShowBlockDialog(false);
449 |       // Navigate back to messages after blocking
450 |       setTimeout(() => navigate("/messages"), 1000);
451 |     } catch (error) {
452 |       console.error("'L Error blocking user:", error);
453 |       toast.error("Failed to block user");
454 |     }
455 |   };
456 |
457 |   const handleReportUser = async (reason: string, details?: string) => {
458 |     if (!currentUser?.uid || !chatUser?.id) return;
459 |
460 |     try {
461 |       await reportUser(currentUser.uid, chatUser.id, reason, details);
462 |       toast.success("Thank you for your report. We'll review it soon.");
463 |       setShowReportModal(false);
464 |     } catch (error) {
465 |       console.error("'L Error reporting user:", error);
466 |       toast.error("Failed to submit report");
467 |       throw error;
468 |     }
469 |   };
470 |
471 |   const handleDeleteConversation = async () => {
472 |     if (!currentUser?.uid || !chatUser?.id) return;
473 |
474 |     try {
475 |       await deleteConversation(currentUser.uid, chatUser.id);
476 |       toast.success("Conversation deleted");
477 |       setShowDeleteDialog(false);
478 |       // Navigate back to messages after deletion
479 |       navigate("/messages");
480 |     } catch (error) {
481 |       console.error("'L Error deleting conversation:", error);
482 |       const errorMessage = error instanceof Error ? error.message : "Failed to delete
conversation";
483 |       toast.error(errorMessage);
484 |     }
485 |   };
486 |
487 |
488 |   return (
489 |     <div className="min-h-screen bg-background flex flex-col">
490 |       {/* Header */}
491 |       <div
492 |         className="sticky top-0 z-10 bg-background border-b border-border"
```

```
        style={{
          paddingTop: 'env(safe-area-inset-top)',
        }}
      >
        <div className="flex items-center justify-between p-4">
          <div className="flex items-center gap-3 flex-1 min-w-0">
            <motion.button
              whileTap={{ scale: 0.95 }}
              onClick={() => navigate("/messages")}
              className="touch-target p-2 -ml-2 rounded-full hover:bg-accent transition-colors flex-shrink-0"
            >
              <ArrowBackIcon style={{ fontSize: 24 }} />
            </motion.button>
            {chatUser && (
              <motion.button
                whileTap={{ scale: 0.98 }}
                onClick={() => setShowProfile(true)}
                className="flex items-center gap-3 flex-1 min-w-0 rounded-lg hover:bg-accent/50 transition-colors p-2 -m...
                <Avatar
                  src={chatUser.avatar}
                  alt={displayName || chatUser.name}
                  sx={{ width: 40, height: 40 }}
                  className="flex-shrink-0"
                />
                <div className="flex-1 min-w-0 text-left">
                  <h1 className="text-lg font-bold text-foreground truncate">
                    {displayName || chatUser.name}
                  </h1>
                </div>
              </motion.button>
            )}
          </div>
          <DropdownMenu>
            <DropdownMenuTrigger asChild>
              <motion.button
                whileTap={{ scale: 0.95 }}
                className="touch-target p-2 rounded-full hover:bg-accent transition-colors flex-shrink-0"
              >
                <MoreVertIcon style={{ fontSize: 24 }} className="text-muted-foreground" />
              </motion.button>
            </DropdownMenuTrigger>
            <DropdownMenuContent align="end" className="w-56">
              <DropdownMenuItem onClick={() => setShowProfile(true)}>
                <PersonIcon style={{ fontSize: 18 }} className="mr-2" />
                View Profile
              </DropdownMenuItem>
              <DropdownMenuItem onClick={handleToggleMute}>
                {isMuted ? (
                  <>
                    <NotificationsIcon style={{ fontSize: 18 }} className="mr-2" />
                    Unmute Notifications
                  </>
                ) : (
                  <>
                    <NotificationsOffIcon style={{ fontSize: 18 }} className="mr-2" />
                    Mute Notifications
                  </>
                )}
              </DropdownMenuItem>
              <DropdownMenuSeparator />
              <DropdownMenuItem
                onClick={() => setShowReportModal(true)}
                className="text-orange-600 focus:text-orange-600"
              >
                <ReportIcon style={{ fontSize: 18 }} className="mr-2" />
                Report User
              </DropdownMenuItem>
              <DropdownMenuItem
                onClick={() => setShowBlockDialog(true)}
                className="text-red-600 focus:text-red-600"
```

```
564 |               >
565 |                 <BlockIcon style={{ fontSize: 18 }} className="mr-2" />
566 |                 Block User
567 |               </DropdownMenuItem>
568 |               <DropdownMenuSeparator />
569 |               <DropdownMenuItem
570 |                 onClick={() => setShowDeleteDialog(true)}
571 |                 className="text-red-600 focus:text-red-600"
572 |               >
573 |                 <DeleteIcon style={{ fontSize: 18 }} className="mr-2" />
574 |                 Delete Conversation
575 |               </DropdownMenuItem>
576 |             </DropdownMenuContent>
577 |           </DropdownMenu>
578 |         </div>
579 |       </div>
580 |
581 |       {/* Messages Area */}
582 |       <div className="flex-1 overflow-y-auto p-4 space-y-4">
583 |         {!chatUser ? (
584 |           <div className="flex items-center justify-center h-full">
585 |             <p className="text-muted-foreground">No user selected</p>
586 |           </div>
587 |         ) : isBlocked ? (
588 |           <div className="flex items-center justify-center h-full">
589 |             <div className="text-center space-y-2">
590 |               <BlockIcon className="mx-auto text-red-500" style={{ fontSize: 48 }} />
591 |               <p className="text-lg font-semibold">You have been blocked</p>
592 |               <p className="text-sm text-muted-foreground">You cannot send messages to this
user.</p>
593 |             </div>
594 |           </div>
595 |         ) : messages.length === 0 ? (
596 |           <div className="flex items-center justify-center h-full">
597 |             <p className="text-muted-foreground">No messages yet. Start the conversation!</p>
598 |           </div>
599 |         ) : (
600 |           messages.map((message, index) => {
601 |             const isMe = message.senderId === currentUser?.uid;
602 |             const showAvatar = !isMe && (
603 |               index === 0 || messages[index - 1].senderId === currentUser?.uid || messages[index
-1].senderId !== messa...
604 |             );
605 |
606 |             return (
607 |               <motion.div
608 |                 key={message.id}
609 |                 initial={{ opacity: 0, y: 10 }}
610 |                 animate={{ opacity: 1, y: 0 }}
611 |                 transition={{ delay: index * 0.05 }}
612 |                 className={`flex gap-2 ${isMe ? "justify-end" : "justify-start"}`}
613 |               >
614 |                 {/* Avatar for received messages */}
615 |                 {!isMe && (
616 |                   <div className="flex-shrink-0 w-8">
617 |                     {showAvatar && (
618 |                       <Avatar
619 |                         src={chatUser.avatar}
620 |                         alt={displayName || chatUser.name}
621 |                         sx={{ width: 32, height: 32 }}
622 |                       />
623 |                     )}
624 |                   </div>
625 |                 )}
626 |
627 |                 {/* Message Bubble */}
628 |                 <div
629 |                   className={`max-w-[75%] ${isMe ? "order-last" : ""}`}
630 |                 >
631 |                   {message.type === 'location' && message.location ? (
632 |                     // Location message
633 |                     <div
634 |                       className={`rounded-2xl px-4 py-3 ${
```

```
                        isMe
                          ? "bg-primary text-primary-foreground rounded-br-md"
                          : "bg-muted text-foreground rounded-bl-md"
                      }`}
                    >
                      <div className="flex items-center gap-2 mb-2">
                        <LocationOnIcon style={{ fontSize: 20 }} />
                        <p className="text-sm font-medium">Shared location</p>
                      </div>
                      <p className="text-xs opacity-90 mb-3">
                        {message.location.lat.toFixed(6)}, {message.location.lng.toFixed(6)}
                      </p>
                      {message.expiresAt && Date.now() < message.expiresAt && (
                        <p className="text-xs opacity-75 mb-3">
                          Expires in {formatTime(Math.floor((message.expiresAt - Date.now()) /
1000))}
                        </p>
                      )}
                      <Button
                        size="sm"
                        variant={isMe ? "secondary" : "default"}
                        onClick={() => openGoogleMapsNavigation(message.location!.lat,
message.location!.lng)}
                        className="w-full"
                      >
                        <NavigationIcon style={{ fontSize: 16 }} className="mr-2" />
                        Start navigation using Google Maps
                      </Button>
                    </div>
                  ) : (
                    // Text message
                    <div
                      className={`rounded-2xl px-4 py-2.5 ${
                        isMe
                          ? "bg-primary text-primary-foreground rounded-br-md"
                          : "bg-muted text-foreground rounded-bl-md"
                      }`}
                    >
                      <p className="text-sm leading-relaxed whitespace-pre-wrap break-words">
                        {message.content}
                      </p>
                    </div>
                  )}
                  <p className={`text-xs text-muted-foreground mt-1 px-1 ${
                    isMe ? "text-right" : "text-left"
                  }`}>
                    {getRelativeTime(message.timestamp)}
                  </p>
                </div>
              </motion.div>
            );
          })
        )}

        {/* Typing Indicator */}
        {isTyping && chatUser && (
          <motion.div
            initial={{ opacity: 0, y: 10 }}
            animate={{ opacity: 1, y: 0 }}
            className="flex gap-2 justify-start"
          >
            <Avatar
              src={chatUser.avatar}
              alt={displayName || chatUser.name}
              sx={{ width: 32, height: 32 }}
            />
            <div className="bg-muted rounded-2xl rounded-bl-md px-4 py-3">
              <div className="flex gap-1">
                <motion.div
                  animate={{ opacity: [0.4, 1, 0.4] }}
                  transition={{ duration: 1, repeat: Infinity, delay: 0 }}
                  className="w-2 h-2 bg-muted-foreground rounded-full"
                />
```

```
                <motion.div
                  animate={{ opacity: [0.4, 1, 0.4] }}
                  transition={{ duration: 1, repeat: Infinity, delay: 0.2 }}
                  className="w-2 h-2 bg-muted-foreground rounded-full"
                />
                <motion.div
                  animate={{ opacity: [0.4, 1, 0.4] }}
                  transition={{ duration: 1, repeat: Infinity, delay: 0.4 }}
                  className="w-2 h-2 bg-muted-foreground rounded-full"
                />
              </div>
            </div>
          </motion.div>
        )}

        <div ref={messagesEndRef} />
      </div>

      {/* Message Request Banner */}
      {isMessageRequest && chatUser && (
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="border-t border-border p-4 bg-muted/50"
        >
          <div className="max-w-2xl mx-auto space-y-3">
            <div className="text-center space-y-1">
              <p className="text-sm font-semibold">
                {displayName || chatUser.name} wants to message you
              </p>
              <p className="text-xs text-muted-foreground">
                Accept to continue the conversation
              </p>
            </div>
            <div className="flex gap-2">
              <Button
                onClick={handleDeclineMessageRequest}
                variant="outline"
                className="flex-1"
              >
                <CloseIcon style={{ fontSize: 16 }} className="mr-2" />
                Decline
              </Button>
              <Button
                onClick={handleAcceptMessageRequest}
                className="flex-1"
              >
                <CheckCircleIcon style={{ fontSize: 16 }} className="mr-2" />
                Accept
              </Button>
            </div>
          </div>
        </motion.div>
      )}

      {/* Input Area */}
      {chatUser && (
        <div className="border-t border-border p-4 bg-background">
        <div className="flex gap-2 items-end">
          <div className="flex-1 relative">
            <textarea
              ref={inputRef}
              value={messageInput}
              onChange={(e) => {
                if (e.target.value.length <= maxCharacters) {
                  setMessageInput(e.target.value);
                }
              }}
              onKeyPress={handleKeyPress}
              placeholder={isBlocked ? "You have been blocked" : isMessageRequest ? "Accept request to send messages" ...
              rows={1}
```

```
              disabled={isBlocked || isMessageRequest}
              className="w-full resize-none rounded-2xl border border-input bg-background px-4 py-3 text-sm ring-offset-
              style={{
                minHeight: "44px",
                maxHeight: "128px",
              }}
            />
            <div className="absolute bottom-2 right-3 text-xs text-muted-foreground">
              {messageInput.length}/{maxCharacters}
            </div>
          </div>
          <Button
            onClick={() => setShowLocationModal(true)}
            size="icon"
            variant="outline"
            disabled={isBlocked || isMessageRequest}
            className="rounded-full h-11 w-11 flex-shrink-0"
          >
            <LocationOnIcon style={{ fontSize: 20 }} />
          </Button>
          <Button
            onClick={handleSendMessage}
            disabled={!messageInput.trim() || isBlocked || isMessageRequest}
            size="icon"
            className="rounded-full h-11 w-11 flex-shrink-0 bg-primary hover:bg-primary/90"
          >
            <SendIcon style={{ fontSize: 20 }} />
          </Button>
        </div>
      </div>
      )}

      {/* Location Sharing Modal */}
      <LocationSharingModal
        open={showLocationModal}
        onOpenChange={setShowLocationModal}
        onShareLocation={handleShareLocation}
        currentLocation={currentLocation}
        isGettingLocation={isGettingLocation}
      />

      {/* Confirmation Modal */}
      {showConfirmModal && pendingLocation && chatUser && (
        <Dialog open={showConfirmModal} onOpenChange={setShowConfirmModal}>
          <DialogContent className="sm:max-w-md">
            <DialogHeader>
              <DialogTitle>Share Your Location?</DialogTitle>
            </DialogHeader>
            <div className="space-y-4 py-4">
              <p className="text-sm text-muted-foreground">
                You're about to share your current location with <span className="font-semibold">{displayName || chatUse...
              </p>
              <div className="bg-muted p-3 rounded-lg">
                <p className="text-xs font-mono">
                  Ø=ÜÍ {pendingLocation.lat.toFixed(6)}, {pendingLocation.lng.toFixed(6)}
                </p>
              </div>
              <div className="flex gap-3">
                <Button
                  variant="outline"
                  className="flex-1"
                  onClick={() => {
                    setShowConfirmModal(false);
                    setPendingLocation(null);
                  }}
                >
                  Cancel
                </Button>
                <Button
                  className="flex-1"
                  onClick={handleConfirmShare}
```

```
                >
                  <LocationOnIcon className="mr-2" style={{ fontSize: 18 }} />
                  Send Location
                </Button>
              </div>
            </div>
          </DialogContent>
        </Dialog>
      )}

      {/* Profile View Modal */}
      {showProfile && chatUser && (
        <ProfileView
          user={{
            id: chatUser.id,
            name: displayName || chatUser.name,
            avatar: chatUser.avatar,
            distance: "In your chat",
            activity: "Active",
            photos: [chatUser.avatar],
          }}
          friendStatus={getFriendStatus()}
          onClose={() => setShowProfile(false)}
          onSendMessage={handleSendMessageFromProfile}
          onAddFriend={handleAddFriend}
          onAcceptFriend={handleAcceptFriend}
          onDeclineFriend={handleDeclineFriend}
        />
      )}

      {/* Report User Modal */}
      {chatUser && (
        <ReportUserModal
          open={showReportModal}
          onOpenChange={setShowReportModal}
          userName={displayName || chatUser.name}
          onReport={handleReportUser}
        />
      )}

      {/* Block User Confirmation Dialog */}
      <AlertDialog open={showBlockDialog} onOpenChange={setShowBlockDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Block {chatUser?.name}?</AlertDialogTitle>
            <AlertDialogDescription>
              This will prevent {chatUser?.name} from sending you messages and you won't see
them on the map. You can unb
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel>Cancel</AlertDialogCancel>
            <AlertDialogAction
              onClick={handleBlockUser}
              className="bg-red-600 hover:bg-red-700"
            >
              Block
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Delete Conversation Confirmation Dialog */}
      <AlertDialog open={showDeleteDialog} onOpenChange={setShowDeleteDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Delete conversation?</AlertDialogTitle>
            <AlertDialogDescription>
              This will permanently delete all messages with {chatUser?.name}. This action
cannot be undone.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
```

```
            <AlertDialogCancel>Cancel</AlertDialogCancel>
            <AlertDialogAction
              onClick={handleDeleteConversation}
              className="bg-red-600 hover:bg-red-700"
            >
              Delete
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>
    </div>
  );
};

export default Chat;

```

File: src/pages/EditProfile.tsx

Lines: 579

```
   1 | import { useState, useEffect, useRef } from "react";
   2 | import { motion, AnimatePresence } from "framer-motion";
   3 | import { useNavigate } from "react-router-dom";
   4 | import { useUser } from "@/contexts/UserContext";
   5 | import { Button } from "@/components/ui/button";
   6 | import { Input } from "@/components/ui/input";
   7 | import { Label } from "@/components/ui/label";
   8 | import { Textarea } from "@/components/ui/textarea";
   9 | import { Card } from "@/components/ui/card";
  10 | import { Checkbox } from "@/components/ui/checkbox";
  11 | import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/
select";
  12 | import { Avatar } from "@mui/material";
  13 | import ArrowBackIcon from "@mui/icons-material/ArrowBack";
  14 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
  15 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
  16 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
  17 | import AddPhotoAlternateIcon from "@mui/icons-material/AddPhotoAlternate";
  18 | import DeleteIcon from "@mui/icons-material/Delete";
  19 | import CameraAltIcon from "@mui/icons-material/CameraAlt";
  20 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
  21 | import { toast } from "sonner";
  22 | import { useAuth } from "@/hooks/useAuth";
  23 | import { getUserData, updateUserProfile } from "@/services/authService";
  24 | import { DEFAULT_AVATARS, generateUserAvatar } from "@/lib/avatars";
  25 | import { storage } from "@/services/firebase";
  26 | import { ref, uploadBytes, getDownloadURL } from "firebase/storage";
  27 |
  28 | const EditProfile = () => {
  29 |   const navigate = useNavigate();
  30 |   const { user } = useAuth();
  31 |   const { userProfile, setUserProfile } = useUser();
  32 |
  33 |   const [username, setUsername] = useState("");
  34 |   const [bio, setBio] = useState("");
  35 |   const [photos, setPhotos] = useState<string[]>([]);
  36 |   const [selectedActivities, setSelectedActivities] = useState<("running" | "cycling" |
"walking")[]>([]);
  37 |   const [selectedPhotoUrl, setSelectedPhotoUrl] = useState<string | null>(null);
  38 |   const [showAvatarPicker, setShowAvatarPicker] = useState(false);
  39 |   const [uploadingPhoto, setUploadingPhoto] = useState(false);
  40 |   const fileInputRef = useRef<HTMLInputElement>(null);
  41 |   const [loading, setLoading] = useState(true);
  42 |   const [saving, setSaving] = useState(false);
  43 |
  44 |   // Load user data from Firebase on mount to get onboarding selections
  45 |   useEffect(() => {
  46 |     const loadUserData = async () => {
  47 |       if (user) {
  48 |         try {
  49 |           const userData = await getUserData(user.uid);
  50 |           if (userData) {
  51 |             // Load username from Firebase (stored as 'name')
  52 |             if (userData.name) {
  53 |               setUsername(userData.name);
  54 |             }
  55 |
  56 |             // Load bio from Firebase
  57 |             if (userData.bio) {
  58 |               setBio(userData.bio);
  59 |             }
  60 |
  61 |             // Load photos from Firebase
  62 |             if (userData.photos && Array.isArray(userData.photos)) {
  63 |               setPhotos(userData.photos);
  64 |             }
  65 |
  66 |             // Load photoURL from Firebase (profile photo/avatar)
```

```
            if (userData.photoURL) {
              setSelectedPhotoUrl(userData.photoURL);
            }

            // Load activities from Firebase - prioritize activities array over single activity
            if (userData.activities && Array.isArray(userData.activities) &&
userData.activities.length > 0) {
              setSelectedActivities(userData.activities);
            } else if (userData.activity) {
              // Fallback to single activity if array doesn't exist
              const activity = userData.activity as "running" | "cycling" | "walking";
              setSelectedActivities([activity]);
            } else {
              // Default to running if nothing is set
              setSelectedActivities(["running"]);
            }
          }
        } catch (error) {
          console.error("Error loading user data:", error);
        } finally {
          setLoading(false);
        }
      } else {
        setLoading(false);
      }
    };

    loadUserData();
  }, [user]);

  const activities = [
    { id: "running", label: "Running", icon: DirectionsRunIcon, color: "success" },
    { id: "cycling", label: "Cycling", icon: DirectionsBikeIcon, color: "primary" },
    { id: "walking", label: "Walking", icon: DirectionsWalkIcon, color: "warning" },
  ] as const;

  const handleActivityToggle = (activityId: "running" | "cycling" | "walking") => {
    setSelectedActivities(prev => {
      if (prev.includes(activityId)) {
        if (prev.length === 1) {
          toast.error("You must have at least one activity selected");
          return prev;
        }
        return prev.filter(a => a !== activityId);
      } else {
        return [...prev, activityId];
      }
    });
  };

  // Handle profile photo/avatar upload
  const handleProfilePhotoUpload = async (event: React.ChangeEvent<HTMLInputElement>) => {
    const file = event.target.files?.[0];
    if (!file || !user) return;

    // Validate file type
    if (!file.type.startsWith('image/')) {
      toast.error("Please select an image file");
      return;
    }

    // Validate file size (max 5MB)
    if (file.size > 5 * 1024 * 1024) {
      toast.error("Image must be less than 5MB");
      return;
    }

    setUploadingPhoto(true);
    try {
      // Create a reference to the file in Firebase Storage
      const storageRef = ref(storage, `profile-photos/${user.uid}/${Date.now()}_${file.name}`);

```

```
138 |       // Upload the file
139 |       const snapshot = await uploadBytes(storageRef, file);
140 |
141 |       // Get the download URL
142 |       const downloadUrl = await getDownloadURL(snapshot.ref);
143 |
144 |       setSelectedPhotoUrl(downloadUrl);
145 |       setShowAvatarPicker(false);
146 |       toast.success("Profile photo updated successfully!");
147 |     } catch (error: any) {
148 |       console.error("Error uploading photo:", error);
149 |       toast.error("Failed to upload photo. Please try again.");
150 |     } finally {
151 |       setUploadingPhoto(false);
152 |     }
153 |   };
154 |
155 |   // Handle default avatar selection
156 |   const handleAvatarSelect = (avatarUrl: string) => {
157 |     setSelectedPhotoUrl(avatarUrl);
158 |     setShowAvatarPicker(false);
159 |     toast.success("Avatar selected!");
160 |   };
161 |
162 |   // Get current display photo URL
163 |   const getDisplayPhotoUrl = (): string => {
164 |     if (selectedPhotoUrl) return selectedPhotoUrl;
165 |     if (user?.photoURL) return user.photoURL;
166 |     return generateUserAvatar(username || user?.displayName || user?.email || 'User');
167 |   };
168 |
169 |   // Handle additional photos upload (up to 3)
170 |   const handlePhotosUpload = (e: React.ChangeEvent<HTMLInputElement>) => {
171 |     const files = e.target.files;
172 |     if (!files) return;
173 |
174 |     if (photos.length >= 3) {
175 |       toast.error("Maximum 3 photos allowed");
176 |       return;
177 |     }
178 |
179 |     Array.from(files).forEach(file => {
180 |       if (photos.length >= 3) return;
181 |
182 |       const reader = new FileReader();
183 |       reader.onloadend = () => {
184 |         setPhotos(prev => [...prev, reader.result as string]);
185 |       };
186 |       reader.readAsDataURL(file);
187 |     });
188 |   };
189 |
190 |   const handleRemovePhoto = (index: number) => {
191 |     setPhotos(prev => prev.filter((_, i) => i !== index));
192 |   };
193 |
194 |   const handleSave = async () => {
195 |     if (!user) {
196 |       toast.error("You must be logged in to save changes");
197 |       return;
198 |     }
199 |
200 |     if (!username.trim()) {
201 |       toast.error("Username cannot be empty");
202 |       return;
203 |     }
204 |
205 |     if (selectedActivities.length === 0) {
206 |       toast.error("Please select at least one activity");
207 |       return;
208 |     }
```

```
209 |
210 |     setSaving(true);
211 |     try {
212 |       // Prepare data to save to Firebase
213 |       const profileData: any = {
214 |         name: username.trim(), // Save username as 'name' in Firebase
215 |         bio: bio.trim() || null,
216 |         photos: photos.length > 0 ? photos : null,
217 |         photoURL: selectedPhotoUrl || null, // Save profile photo/avatar URL
218 |         // Save activities - use first activity as primary activity for compatibility
219 |         activity: selectedActivities[0], // Primary activity (for backward compatibility)
220 |         activities: selectedActivities, // Array of all selected activities
221 |       };
222 |
223 |       // Save to Firebase
224 |       await updateUserProfile(user.uid, profileData);
225 |
226 |       // Also update local context
227 |       setUserProfile({
228 |         ...userProfile!,
229 |         username: username.trim(),
230 |         bio: bio.trim(),
231 |         photos,
232 |         activities: selectedActivities,
233 |       });
234 |
235 |       toast.success("Profile updated successfully!");
236 |       navigate("/settings");
237 |     } catch (error: any) {
238 |       console.error("Error saving profile:", error);
239 |       toast.error("Failed to save profile. Please try again.");
240 |     } finally {
241 |       setSaving(false);
242 |     }
243 |   };
244 |
245 |   return (
246 |     <div className="min-h-screen bg-gradient-to-br from-primary/5 via-background to-success/5">
247 |       {/* Header */}
248 |       <motion.div
249 |         initial={{ opacity: 0, y: -20 }}
250 |         animate={{ opacity: 1, y: 0 }}
251 |         className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b
border-border/50"
252 |       >
253 |         <div className="max-w-2xl mx-auto px-6 py-5 flex items-center gap-4">
254 |           <motion.button
255 |             whileTap={{ scale: 0.95 }}
256 |             onClick={() => navigate("/settings")}
257 |             className="touch-target p-2 hover:bg-secondary rounded-xl transition-all
duration-200"
258 |           >
259 |             <ArrowBackIcon style={{ fontSize: 28 }} />
260 |           </motion.button>
261 |           <h1 className="text-3xl font-bold">Edit Profile</h1>
262 |         </div>
263 |       </motion.div>
264 |
265 |       <div className="max-w-2xl mx-auto px-6 py-6 space-y-6 pb-10">
266 |         {/* Profile Picture - Clickable with Avatar Picker */}
267 |         <motion.div
268 |           initial={{ opacity: 0, y: 20 }}
269 |           animate={{ opacity: 1, y: 0 }}
270 |           transition={{ duration: 0.4 }}
271 |           className="flex flex-col items-center gap-3"
272 |         >
273 |           <div className="relative group">
274 |             <button
275 |               type="button"
276 |               onClick={() => setShowAvatarPicker(true)}
277 |               className="relative rounded-full focus:outline-none focus:ring-4 focus:ring-
primary/30 transition-all dura...
278 |             >
279 |               <Avatar
```

```
                sx={{ width: 120, height: 120 }}
                alt="Profile"
                src={getDisplayPhotoUrl()}
                className="border-4 border-primary/20 shadow-elevation-3"
              />
              {/* Camera overlay on hover */}
              <div className="absolute inset-0 bg-black/40 rounded-full flex items-center justify-center opacity-0 group...
                <CameraAltIcon className="text-white" style={{ fontSize: 40 }} />
              </div>
              {/* Selected indicator */}
              {selectedPhotoUrl && (
                <div className="absolute -bottom-1 -right-1 bg-green-500 rounded-full p-1.5">
                  <CheckCircleIcon className="text-white" style={{ fontSize: 20 }} />
                </div>
              )}
            </button>
          </div>
          <button
            type="button"
            onClick={() => setShowAvatarPicker(true)}
            className="text-sm text-primary hover:text-primary/80 font-medium transition-colors"
          >
            {selectedPhotoUrl ? "Change Photo" : "Add Photo"}
          </button>
        </motion.div>

        {/* Avatar Picker Modal */}
        <AnimatePresence>
          {showAvatarPicker && (
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              exit={{ opacity: 0 }}
              className="fixed inset-0 bg-black/60 z-50 flex items-center justify-center p-4"
              onClick={() => setShowAvatarPicker(false)}
            >
              <motion.div
                initial={{ scale: 0.9, opacity: 0 }}
                animate={{ scale: 1, opacity: 1 }}
                exit={{ scale: 0.9, opacity: 0 }}
                className="bg-card rounded-2xl p-6 max-w-md w-full max-h-[80vh] overflow-y-auto shadow-xl"
                onClick={(e) => e.stopPropagation()}
              >
                <div className="flex justify-between items-center mb-4">
                  <h3 className="text-xl font-bold text-foreground">Choose Your Photo</h3>
                  <button
                    type="button"
                    onClick={() => setShowAvatarPicker(false)}
                    className="text-muted-foreground hover:text-foreground transition-colors text-2xl leading-none"
                  >
                    ×
                  </button>
                </div>

                {/* Upload Photo Option */}
                <div className="mb-6">
                  <input
                    ref={fileInputRef}
                    type="file"
                    accept="image/*"
                    onChange={handleProfilePhotoUpload}
                    className="hidden"
                  />
                  <button
                    type="button"
                    onClick={() => fileInputRef.current?.click()}
                    disabled={uploadingPhoto}
                    className="w-full flex items-center justify-center gap-3 p-4 rounded-xl border-2 border-dashed borde...
                    {uploadingPhoto ? (
                      <>
```

```
                        <div className="w-6 h-6 border-2 border-primary border-t-transparent
rounded-full animate-spin" ... <span className="font-medium text-primary">Uploading...</span>
                      </>
                    ) : (
                      <>
                        <AddPhotoAlternateIcon className="text-primary" style={{ fontSize:
28 }} />
                        <span className="font-medium text-primary">Upload Your Photo</span>
                      </>
                    )}
                  </button>
                  <p className="text-xs text-muted-foreground text-center mt-2">
                    Max 5MB • JPG, PNG, GIF
                  </p>
                </div>

                {/* Divider */}
                <div className="flex items-center gap-3 mb-6">
                  <div className="flex-1 h-px bg-border" />
                  <span className="text-sm text-muted-foreground">or choose an avatar</span>
                  <div className="flex-1 h-px bg-border" />
                </div>

                {/* Default Avatars Grid */}
                <div className="grid grid-cols-4 gap-3">
                  {DEFAULT_AVATARS.map((avatar) => (
                    <button
                      key={avatar.id}
                      type="button"
                      onClick={() => handleAvatarSelect(avatar.url)}
                      className={`relative rounded-xl p-2 transition-all duration-200
hover:scale-105 ${
                        selectedPhotoUrl === avatar.url
                          ? "bg-primary/20 ring-2 ring-primary"
                          : "bg-secondary hover:bg-secondary/80"
                      }`}
                    >
                      <img
                        src={avatar.url}
                        alt={avatar.name}
                        className="w-full aspect-square rounded-lg"
                      />
                      {selectedPhotoUrl === avatar.url && (
                        <div className="absolute -top-1 -right-1 bg-primary rounded-full p-0.5">
                          <CheckCircleIcon className="text-white" style={{ fontSize: 14 }} />
                        </div>
                      )}
                    </button>
                  ))}
                </div>

                {/* Close button */}
                <Button
                  type="button"
                  onClick={() => setShowAvatarPicker(false)}
                  className="w-full mt-6"
                  variant="outline"
                >
                  Done
                </Button>
              </motion.div>
            </motion.div>
          )}
        </AnimatePresence>

        {/* Photos Section */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1 }}
        >
          <Card className="p-6 space-y-4">
            <div>
```

```
              <Label className="text-lg font-bold">Photos</Label>
              <p className="text-sm text-muted-foreground">Add up to 3 photos</p>
            </div>
            <div className="grid grid-cols-3 gap-4">
              {photos.map((photo, index) => (
                <div key={index} className="relative aspect-square rounded-lg overflow-hidden
border-2 border-border"> <img src={photo} alt={`Profile ${index + 1}`} className="w-full h-full object-
cover" />
                  <button
                    onClick={() => handleRemovePhoto(index)}
                    className="absolute top-2 right-2 p-1 bg-destructive text-destructive-
foreground rounded-full hover:...
                    <DeleteIcon style={{ fontSize: 16 }} />
                  </button>
                </div>
              ))}
              {photos.length < 3 && (
                <label className="aspect-square border-2 border-dashed border-border rounded-lg
flex flex-col items-cent...<input
                    type="file"
                    accept="image/*"
                    multiple
                    className="hidden"
                    onChange={handlePhotosUpload}
                  />
                  <AddPhotoAlternateIcon className="text-muted-foreground" style={{ fontSize:
32 }}/>
                  <span className="text-xs text-muted-foreground mt-2">Add Photo</span>
                </label>
              )}
            </div>
          </Card>
        </motion.div>

        {/* Basic Info */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.2 }}
        >
          <Card className="p-6 space-y-4">
            <Label className="text-lg font-bold">Basic Info</Label>

            <div className="space-y-2">
              <Label htmlFor="username">Username</Label>
              <Input
                id="username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
                placeholder="Your username"
                className="h-12"
              />
              <p className="text-xs text-muted-foreground">
                This name will be displayed to other users
              </p>
            </div>

            <div className="space-y-2">
              <Label htmlFor="bio">Bio</Label>
              <Textarea
                id="bio"
                value={bio}
                onChange={(e) => setBio(e.target.value)}
                placeholder="Tell us about yourself..."
                maxLength={500}
                rows={4}
              />
              <p className="text-xs text-muted-foreground text-right">{bio.length}/500</p>
            </div>
          </Card>
        </motion.div>

        {/* Activities */}
```

```
493 |         <motion.div
494 |           initial={{ opacity: 0, y: 20 }}
495 |           animate={{ opacity: 1, y: 0 }}
496 |           transition={{ delay: 0.3 }}
497 |         >
498 |           <Card className="p-6 space-y-4">
499 |             <div>
500 |               <Label className="text-lg font-bold">Activities</Label>
501 |               <p className="text-sm text-muted-foreground">Select at least one</p>
502 |             </div>
503 |             <div className="space-y-2">
504 |               {activities.map((activity) => {
505 |                 const Icon = activity.icon;
506 |                 const isSelected = selectedActivities.includes(activity.id as "running" |
"cycling" | "walking");
507 |
508 |                 return (
509 |                   <motion.div
510 |                     key={activity.id}
511 |                     whileTap={{ scale: 0.98 }}
512 |                     className={`
513 |                       flex items-center justify-between p-3 rounded-xl border-2 transition-all
duration-300 cursor-point... ${isSelected
514 |
515 |                         ? activity.color === 'success'
516 |                           ? 'border-success/30 bg-success/10'
517 |                           : activity.color === 'primary'
518 |                           ? 'border-primary/30 bg-primary/10'
519 |                           : 'border-warning/30 bg-warning/10'
520 |                         : 'border-border bg-muted/30'
521 |                       }
522 |                     `}
523 |                     onClick={() => handleActivityToggle(activity.id as "running" | "cycling" |
"walking")}
524 |                   >
525 |                     <div className="flex items-center gap-3 flex-1">
526 |                       <Icon
527 |                         className={
528 |                           isSelected
529 |                             ? activity.color === 'success'
530 |                               ? 'text-success'
531 |                               : activity.color === 'primary'
532 |                               ? 'text-primary'
533 |                               : 'text-warning'
534 |                             : 'text-muted-foreground'
535 |                         }
536 |                         style={{ fontSize: 24 }}
537 |                       />
538 |                       <span className="font-medium">{activity.label}</span>
539 |                     </div>
540 |                     <div
541 |                       className="flex-shrink-0"
542 |                       onClick={(e) => e.stopPropagation()}
543 |                     >
544 |                       <Checkbox
545 |                         checked={isSelected}
546 |                         onCheckedChange={() => {
547 |                           handleActivityToggle(activity.id as "running" | "cycling" | "walking");
548 |                         }}
549 |                         onClick={(e) => e.stopPropagation()}
550 |                       />
551 |                     </div>
552 |                   </motion.div>
553 |                 );
554 |               })}
555 |             </div>
556 |           </Card>
557 |         </motion.div>
558 |
559 |         {/* Save Button */}
560 |         <motion.div
561 |           initial={{ opacity: 0, y: 20 }}
562 |           animate={{ opacity: 1, y: 0 }}
563 |           transition={{ delay: 0.4 }}
```

```
564 |         >
565 |           <Button
566 |             onClick={handleSave}
567 |             disabled={saving || loading}
568 |             className="w-full h-12 text-base font-bold"
569 |           >
570 |             {saving ? "Saving..." : "Save Changes"}
571 |           </Button>
572 |         </motion.div>
573 |       </div>
574 |     </div>
575 |   );
576 | };
577 |
578 | export default EditProfile;
579 |
```

File: src/pages/Events.tsx

Lines: 2435

```
import { useState, useEffect, useMemo, useRef, useCallback } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Textarea } from "@/components/ui/textarea";
import { useNavigate } from "react-router-dom";
import { useAuth } from "@/hooks/useAuth";
import { listenToEvents, Event as FirebaseEvent, joinEvent, leaveEvent, listenToEventCheckIns, createEvent, deleteEvent } from "@/services/eventService";
import { getUserData } from "@/services/authService";
import { calculateDistance, formatDistance } from "@/utils/distance";
import { openGoogleMapsNavigation } from "@/utils/navigation";
import { GoogleMap, Marker, InfoWindow, OverlayView, useJsApiLoader, Autocomplete, Polyline } from "@react-google-maps/api";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import ExploreIcon from "@mui/icons-material/Explore";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import EventIcon from "@mui/icons-material/Event";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import PeopleIcon from "@mui/icons-material/People";
import AccessTimeIcon from "@mui/icons-material/AccessTime";
import NavigationIcon from "@mui/icons-material/Navigation";
import StarIcon from "@mui/icons-material/Star";
import AddIcon from "@mui/icons-material/Add";
import EditIcon from "@mui/icons-material/Edit";
import ViewListIcon from "@mui/icons-material/ViewList";
import MapIcon from "@mui/icons-material/Map";
import FilterListIcon from "@mui/icons-material/FilterList";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import CloseIcon from "@mui/icons-material/Close";
import ClearIcon from "@mui/icons-material/Clear";
import LaunchIcon from "@mui/icons-material/Launch";
import NotificationsIcon from "@mui/icons-material/Notifications";
import MailIcon from "@mui/icons-material/Mail";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import EmojiEventsIcon from "@mui/icons-material/EmojiEvents";
import ChatBubbleIcon from "@mui/icons-material/ChatBubble";
import MyLocationIcon from "@mui/icons-material/MyLocation";
import FitScreenIcon from "@mui/icons-material/FitScreen";
import Avatar from "@mui/material/Avatar";
import { NotificationBell } from "@/components/NotificationBell";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { toast } from "sonner";
import { useNotificationContext } from "@/contexts/NotificationContext";
import { EventDetailModal } from "@/components/EventDetailModal";
import { CreateEventModal, type CreateEventFormData } from "@/components/CreateEventModal";
import { EventsTopBar } from "@/components/EventsTopBar";
import { EventDetailsPanel } from "@/components/EventDetailsPanel";
import BottomNavigation from "@/components/BottomNavigation";
import { generateDummyEvents, ENABLE_DUMMY_DATA } from "@/lib/dummyData";
import {
  Drawer,
  DrawerContent,
  DrawerHeader,
  DrawerTitle,
  DrawerDescription,
} from "@/components/ui/drawer";

const libraries: ("places")[] = ["places"];

type EventType = "running" | "cycling" | "walking" | "others";
type EventCategory = "all" | "user" | "sponsored";
```

```
 67 |
 68 | interface Event {
 69 |   id: string;
 70 |   title: string;
 71 |   description: string;
 72 |   type: EventType;
 73 |   category: "user" | "sponsored";
 74 |   date: string;
 75 |   time: string;
 76 |   location: string;
 77 |   distance: string;
 78 |   distanceValue: number;
 79 |   participants: string[]; // Array of user IDs
 80 |   maxParticipants?: number;
 81 |   hostId?: string;
 82 |   hostName?: string;
 83 |   hostAvatar?: string;
 84 |   sponsorLogo?: string;
 85 |   lat: number;
 86 |   lng: number;
 87 |   isJoined?: boolean;
 88 |   isPast?: boolean;
 89 |   isGreyedOut?: boolean;
 90 |   createdAt?: number; // Timestamp from Firebase
 91 | }
 92 |
 93 | // Custom Event Marker Component
 94 | interface EventMarkerProps {
 95 |   event: Event;
 96 |   checkInCount: number;
 97 |   countdown: string;
 98 |   onClick: () => void;
 99 | }
100 |
101 | const EventMarker = ({ event, checkInCount, countdown, onClick }: EventMarkerProps) => {
102 |   const getActivityColor = (type: EventType): string => {
103 |     switch (type) {
104 |       case "running":
105 |         return "#22c55e"; // success/green
106 |       case "cycling":
107 |         return "#3b82f6"; // primary/blue
108 |       case "walking":
109 |         return "#eab308"; // warning/yellow
110 |       case "others":
111 |         return "#a855f7"; // secondary/purple
112 |       default:
113 |         return "#6b7280"; // gray
114 |     }
115 |   };
116 |
117 |   const getActivityIcon = (type: EventType) => {
118 |     switch (type) {
119 |       case "running":
120 |         return <DirectionsRunIcon style={{ fontSize: 16 }} className="text-success" />;
121 |       case "cycling":
122 |         return <DirectionsBikeIcon style={{ fontSize: 16 }} className="text-primary" />;
123 |       case "walking":
124 |         return <DirectionsWalkIcon style={{ fontSize: 16 }} className="text-warning" />;
125 |       case "others":
126 |         return <FitnessCenterIcon style={{ fontSize: 16 }} className="text-secondary" />;
127 |       default:
128 |         return <FitnessCenterIcon style={{ fontSize: 16 }} className="text-muted-foreground" />;
129 |     }
130 |   };
131 |
132 |   const borderColor = getActivityColor(event.type);
133 |   const participantsCount = Array.isArray(event.participants) ? event.participants.length : 0;
134 |
135 |   return (
136 |     <div
137 |       onClick={onClick}
```

```
      className="relative cursor-pointer transform transition-transform hover:scale-110"
      style={{ transform: "translate(-50%, -50%)" }}
    >
      {/* Profile Photo Marker */}
      <div
        className="relative rounded-full border-4 shadow-lg overflow-hidden"
        style={{
          width: "56px",
          height: "56px",
          borderColor: borderColor,
        }}
      >
        <img
          src={event.hostAvatar || `https://ui-avatars.com/api/?name=${event.hostName || 'User'}
`}
          alt={event.hostName || "Host"}
          className="w-full h-full object-cover"
          onError={(e) => {
            e.currentTarget.src = `https://ui-avatars.com/api/?name=${event.hostName || 'User'}`;
          }}
        />
        {/* Check-in Count Badge - Only show if no participants */}
        {checkInCount > 0 && participantsCount === 0 && (
          <div className="absolute -bottom-1 -right-1 bg-primary text-primary-foreground text-xs
font-bold rounded-full
            {checkInCount}
          </div>
        )}
      </div>
      {/* Activity Type Icon Badge with Countdown and Participant Count - Outside Top Right */}
      <div className="absolute -top-2 -right-2 flex flex-col items-center gap-1 z-10">
        {/* Countdown Timer - Above Activity Icon */}
        <div className="bg-background/95 backdrop-blur-sm text-xs font-semibold px-2 py-1
rounded-full border border-bor...
          {countdown}
        </div>
        <div className="bg-background rounded-full p-1.5 border-2 shadow-lg"
style={{ borderColor: borderColor }}>
          {getActivityIcon(event.type)}
        </div>
        {/* Participant Count - Below Activity Icon */}
        {participantsCount > 0 && (
          <div className="bg-success text-white text-xs font-bold rounded-full min-w-[20px] h-5
flex items-center justif...
            {participantsCount}
          </div>
        )}
      </div>
    </div>
  );
};

const Events = () => {
  const navigate = useNavigate();
  const { user: currentUser, loading: authLoading } = useAuth();
  const { unreadCount, notifications, dismissNotification, markAllAsRead,
handleNotificationTap } = useNotificationCente...
  const [viewMode, setViewMode] = useState<"map" | "list">("map");
  const [activityFilter, setActivityFilter] = useState<EventType | "all">("all");
  const [categoryFilter, setCategoryFilter] = useState<EventCategory>("all");
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedEvent, setSelectedEvent] = useState<Event | null>(null);
  const [showEventDetail, setShowEventDetail] = useState(false);
  const [showCreateEvent, setShowCreateEvent] = useState(false);
  const [showEditEvent, setShowEditEvent] = useState(false);
  const [eventBeingEdited, setEventBeingEdited] = useState<Event | null>(null);
  const [showNotificationDrawer, setShowNotificationDrawer] = useState(false);
  const [events, setEvents] = useState<Event[]>([]);
  const [loading, setLoading] = useState(true);
  const [drawerOpen, setDrawerOpen] = useState(false); // Mobile drawer state
  const [checkInCounts, setCheckInCounts] = useState<Record<string, number>>({});
  const [mapCenter, setMapCenter] = useState<{ lat: number; lng: number }>({ lat: 14.5995, lng:
120.9842 });
  const [mapZoom, setMapZoom] = useState(13);
  const [mapRef, setMapRef] = useState<any>(null);
  const [currentTime, setCurrentTime] = useState(new Date());

  // Pin mode state for inline event creation
```

```
  const [isPinMode, setIsPinMode] = useState(false);
  const [selectedEventLocation, setSelectedEventLocation] = useState<{ lat: number; lng: number;
address?: string } | null>(null);
  const [showEventForm, setShowEventForm] = useState(false);
  const [pinModeAddress, setPinModeAddress] = useState<string>("");

  // Inline event form state
  const [inlineEventForm, setInlineEventForm] = useState({
    title: "",
    description: "",
    activityType: "running" as "running" | "cycling" | "walking" | "others",
    date: "",
    time: "",
    maxParticipants: undefined as number | undefined,
  });
  const [isSubmittingEvent, setIsSubmittingEvent] = useState(false);

  // Autocomplete for location search in pin mode
  const [autocomplete, setAutocomplete] = useState<google.maps.places.Autocomplete | null>(null);
  const [searchValue, setSearchValue] = useState("");

  // Pending location for center-pin mode (tracks map center as user drags)
  const [pendingLocation, setPendingLocation] = useState<{ lat: number; lng: number } |
null>(null);
  const [isReverseGeocoding, setIsReverseGeocoding] = useState(false);
  // Flag to prevent handleMapIdle from overwriting search-selected location (using ref for sync
update)
  const isSearchLocationSetRef = useRef(false);

  // Route state for showing directions
  const [routePath, setRoutePath] = useState<google.maps.LatLng[]>([]);
  const [showRoute, setShowRoute] = useState(false);
  const [routeDestination, setRouteDestination] = useState<{ lat: number; lng: number } |
null>(null);

  // Google Maps API loader
  const googleMapsApiKey = import.meta.env.VITE_GOOGLE_MAPS_API_KEY;
  const hasApiKey = googleMapsApiKey && googleMapsApiKey.trim().length > 0;

  const { isLoaded, loadError } = useJsApiLoader({
    id: 'google-map-script',
    googleMapsApiKey: hasApiKey ? googleMapsApiKey : '',
    libraries: libraries,
    version: 'weekly', // Use weekly version for latest features and fixes
  });

  // Track if we've already logged an error to prevent spam
  const errorLoggedRef = useRef(false);

  // Log error details for debugging (with debouncing to prevent spam)
  useEffect(() => {
    if (loadError && !errorLoggedRef.current) {
      // Only log once to prevent console spam from rapid retries
      errorLoggedRef.current = true;

      console.error('Ø=Ýúþ  Google Maps Load Error:', loadError);
      if (!hasApiKey) {
        console.error(''L VITE_GOOGLE_MAPS_API_KEY is missing or empty in .env file');
        console.error('Ø=Ü¡ Solution: Create a .env file in the project root with:');
        console.error('   VITE_GOOGLE_MAPS_API_KEY=your_api_key_here');
        console.error('');
        console.error('Ø=ÜÖ To get an API key:');
        console.error('   1. Go to https://console.cloud.google.com/');
        console.error('   2. Enable "Maps JavaScript API"');
        console.error('   3. Create an API key in "Credentials"');
        console.error('   4. Add it to your .env file');
      } else {
        console.error(''L API Key exists but Google Maps failed to load');
        console.error('Ø=Ü¡ Troubleshooting steps:');
        console.error('   1. Verify API key is valid in Google Cloud Console');
        console.error('   2. Ensure "Maps JavaScript API" is enabled');
        console.error('   3. Check API key restrictions (HTTP referrers, IP addresses)');
        console.error('   4. Verify your domain is allowed in API key restrictions');
        console.error('   5. Check browser console network tab for detailed error');
        console.error('   6. Restart your dev server after adding/changing .env file');
```

```
      }
    }

    // Reset error flag if load succeeds
    if (isLoaded) {
      errorLoggedRef.current = false;
    }
  }, [loadError, hasApiKey, isLoaded]);

  // User location state - loaded lazily when needed
  const [userLocation, setUserLocation] = useState<{ lat: number; lng: number } | null>(null);
  const [isGettingLocation, setIsGettingLocation] = useState(false);

  // Lazy load user location with better error handling
  const requestUserLocation = useCallback((): Promise<{ lat: number; lng: number } | null> => {
    return new Promise((resolve) => {
      // If location already exists, return it immediately
      if (userLocation) {
        resolve(userLocation);
        return;
      }

      // If already requesting, wait for it
      if (isGettingLocation) {
        // Wait for location to be set
        const checkInterval = setInterval(() => {
          if (userLocation) {
            clearInterval(checkInterval);
            resolve(userLocation);
          }
        }, 100);
        setTimeout(() => clearInterval(checkInterval), 5000); // Timeout after 5 seconds
        return;
      }

      if (!navigator.geolocation) {
        toast.error("Geolocation is not supported by your browser");
        resolve(null);
        return;
      }

      setIsGettingLocation(true);

      navigator.geolocation.getCurrentPosition(
        (position) => {
          const location = {
            lat: position.coords.latitude,
            lng: position.coords.longitude,
          };
          setUserLocation(location);
          setIsGettingLocation(false);
          resolve(location);
          // Note: Nearest event navigation is handled by useEffect that watches userLocation
        },
        (error) => {
          setIsGettingLocation(false);
          let errorMessage = "Could not get your location";

          switch (error.code) {
            case error.PERMISSION_DENIED:
              errorMessage = "Location access denied. Please enable location permissions in your
browser settings to use
              toast.error(errorMessage, { duration: 5000 });
              break;
            case error.POSITION_UNAVAILABLE:
              errorMessage = "Location information is unavailable. Please check your GPS
connection.";
              toast.error(errorMessage, { duration: 4000 });
              break;
            case error.TIMEOUT:
              errorMessage = "Location request timed out. Please try again.";
              toast.error(errorMessage, { duration: 3000 });
              break;
```

```
351 |             default:
352 |               errorMessage = "An unknown error occurred while getting your location.";
353 |               toast.error(errorMessage, { duration: 3000 });
354 |               break;
355 |           }
356 |
357 |           console.warn("Could not get user location:", error);
358 |           resolve(null);
359 |         },
360 |         {
361 |           enableHighAccuracy: true,
362 |           timeout: 10000, // 10 second timeout
363 |           maximumAge: 60000 // Accept cached location up to 1 minute old
364 |         }
365 |       );
366 |     });
367 |   }, [userLocation, isGettingLocation]);
368 |
369 |   // Calculate estimated travel time based on distance
370 |   const calculateTravelTime = useCallback((distanceKm: number, mode: 'walking' | 'driving' |
'cycling' = 'driving'): string => {
371 |     if (!distanceKm || distanceKm === Infinity) return "Unknown";
372 |
373 |     let speedKmh: number;
374 |     switch (mode) {
375 |       case 'walking':
376 |         speedKmh = 5; // Average walking speed
377 |         break;
378 |       case 'cycling':
379 |         speedKmh = 15; // Average cycling speed
380 |         break;
381 |       case 'driving':
382 |       default:
383 |         speedKmh = 50; // Average city driving speed
384 |         break;
385 |     }
386 |
387 |     const hours = distanceKm / speedKmh;
388 |     const minutes = Math.round(hours * 60);
389 |
390 |     if (minutes < 1) return "< 1 min";
391 |     if (minutes < 60) return `${minutes} min`;
392 |
393 |     const hrs = Math.floor(minutes / 60);
394 |     const mins = minutes % 60;
395 |     return mins > 0 ? `${hrs}h ${mins}m` : `${hrs}h`;
396 |   }, []);
397 |
398 |   // Countdown timer utility function
399 |   const calculateCountdown = useCallback((date: string, time: string): string => {
400 |     try {
401 |       // Parse date (could be in various formats)
402 |       let eventDate: Date;
403 |
404 |       // If date is already a full ISO string with time
405 |       if (date.includes('T')) {
406 |         eventDate = new Date(date);
407 |       } else if (date.match(/^\d{4}-\d{2}-\d{2}$/)) {
408 |         // ISO date format (YYYY-MM-DD) - combine with time
409 |         eventDate = new Date(`${date}T${time}`);
410 |       } else if (date.includes('/')) {
411 |         // Assume format like "12/25/2024" or "MM/DD/YYYY"
412 |         const parts = date.split('/');
413 |         if (parts.length === 3) {
414 |           // Check if it's MM/DD/YYYY or DD/MM/YYYY by checking first part
415 |           const firstPart = parseInt(parts[0]);
416 |           if (firstPart > 12) {
417 |             // DD/MM/YYYY format
418 |             eventDate = new Date(`${parts[2]}-${parts[1]}-${parts[0]}T${time}`);
419 |           } else {
420 |             // MM/DD/YYYY format
421 |             eventDate = new Date(`${parts[2]}-${parts[0]}-${parts[1]}T${time}`);
```

```
422 |           }
423 |         } else {
424 |           eventDate = new Date(`${date}T${time}`);
425 |         }
426 |       } else {
427 |         // Try parsing as-is or with time appended
428 |         eventDate = new Date(`${date}T${time}`);
429 |       }
430 |
431 |       // Validate the date
432 |       if (isNaN(eventDate.getTime())) {
433 |         return "Invalid date";
434 |       }
435 |
436 |       const now = currentTime;
437 |       const diff = eventDate.getTime() - now.getTime();
438 |
439 |       if (diff < 0) {
440 |         const pastDiff = Math.abs(diff);
441 |         const pastHours = Math.floor(pastDiff / (1000 * 60 * 60));
442 |         if (pastHours < 24) {
443 |           return "Started";
444 |         }
445 |         return "Expired";
446 |       }
447 |
448 |       const hours = Math.floor(diff / (1000 * 60 * 60));
449 |       const minutes = Math.floor((diff % (1000 * 60 * 60)) / (1000 * 60));
450 |
451 |       if (hours > 24) {
452 |         const days = Math.floor(hours / 24);
453 |         return `${days}d ${hours % 24}h`;
454 |       } else if (hours > 0) {
455 |         return `${hours}h ${minutes}m`;
456 |       } else if (minutes > 0) {
457 |         return `${minutes}m`;
458 |       } else {
459 |         return "Starting soon";
460 |       }
461 |     } catch (error) {
462 |       return "Invalid date";
463 |     }
464 |   }, [currentTime]);
465 |
466 |   // Function to get directions from user location to event location
467 |   const showDirectionsToEvent = useCallback((destinationLat: number, destinationLng: number) => {
468 |     if (!userLocation || !isLoaded || !window.google) {
469 |       toast.error("Location services are not available");
470 |       return;
471 |     }
472 |
473 |     if (!window.google.maps.DirectionsService || !window.google.maps.DirectionsRenderer) {
474 |       toast.error("Directions service is not available");
475 |       return;
476 |     }
477 |
478 |     const directionsService = new window.google.maps.DirectionsService();
479 |     const origin = new window.google.maps.LatLng(userLocation.lat, userLocation.lng);
480 |     const destination = new window.google.maps.LatLng(destinationLat, destinationLng);
481 |
482 |     directionsService.route(
483 |       {
484 |         origin: origin,
485 |         destination: destination,
486 |         travelMode: window.google.maps.TravelMode.DRIVING,
487 |       },
488 |       (result, status) => {
489 |         if (status === window.google.maps.DirectionsStatus.OK && result) {
490 |           // Extract the route path from the result
491 |           const path: google.maps.LatLng[] = [];
492 |           result.routes[0].legs[0].steps.forEach((step) => {
```

```
493 |             step.path.forEach((point) => {
494 |               path.push(point);
495 |             });
496 |           });
497 |
498 |           setRoutePath(path);
499 |           setRouteDestination({ lat: destinationLat, lng: destinationLng });
500 |           setShowRoute(true);
501 |
502 |           // Adjust map view to fit both origin and destination
503 |           if (mapRef) {
504 |             const bounds = new window.google.maps.LatLngBounds();
505 |             bounds.extend(origin);
506 |             bounds.extend(destination);
507 |             mapRef.fitBounds(bounds);
508 |
509 |             // Add some padding
510 |             mapRef.setOptions({
511 |               padding: { top: 100, right: 100, bottom: 100, left: 100 }
512 |             });
513 |           }
514 |
515 |           toast.success("Route displayed on map");
516 |         } else {
517 |           toast.error("Could not calculate route. Please try again.");
518 |         }
519 |       }
520 |     );
521 |   }, [userLocation, isLoaded, mapRef]);
522 |
523 |   // Function to clear the route
524 |   const clearRoute = useCallback(() => {
525 |     setShowRoute(false);
526 |     setRoutePath([]);
527 |     setRouteDestination(null);
528 |   }, []);
529 |
530 |   // Function to open route in Google Maps for navigation
531 |   const openRouteInGoogleMaps = useCallback(() => {
532 |     if (routeDestination) {
533 |       openGoogleMapsNavigation(routeDestination.lat, routeDestination.lng);
534 |     }
535 |   }, [routeDestination]);
536 |
537 |   // Update current time every minute for countdown
538 |   useEffect(() => {
539 |     const interval = setInterval(() => {
540 |       setCurrentTime(new Date());
541 |     }, 60000); // Update every minute
542 |
543 |     return () => clearInterval(interval);
544 |   }, []);
545 |
546 |   // Mark all notifications as read when notification drawer opens
547 |   useEffect(() => {
548 |     if (showNotificationDrawer && unreadCount > 0) {
549 |       markAllAsRead();
550 |     }
551 |   }, [showNotificationDrawer, unreadCount, markAllAsRead]);
552 |
553 |   // Listen to events from Firebase (only when authenticated)
554 |   useEffect(() => {
555 |     // Don't set up listener if auth is still loading or user is not authenticated
556 |     if (authLoading || !currentUser?.uid) {
557 |       setLoading(false);
558 |       return;
559 |     }
560 |
561 |     const unsubscribe = listenToEvents((firebaseEvents: FirebaseEvent[]) => {
562 |       // Transform Firebase events to match local Event interface
563 |       let transformedEvents: Event[] = firebaseEvents.map((event) => {
```

```
564 |         // Calculate distance if user location is available
565 |         let distance = "Unknown";
566 |         let distanceValue = Infinity;
567 |
568 |         if (userLocation && event.lat && event.lng) {
569 |           const dist = calculateDistance(
570 |             userLocation.lat,
571 |             userLocation.lng,
572 |             event.lat,
573 |             event.lng
574 |           );
575 |           if (dist !== null) {
576 |             distanceValue = dist;
577 |             distance = formatDistance(dist);
578 |           }
579 |         }
580 |
581 |         // Calculate if event is past or greyed out (24 hours after event)
582 |         let isPast = false;
583 |         let isGreyedOut = false;
584 |
585 |         try {
586 |           const eventDateTime = new Date(`${event.date}T${event.time || '00:00'}`);
587 |           if (!isNaN(eventDateTime.getTime())) {
588 |             const now = new Date();
589 |             const eventEndTime = new Date(eventDateTime);
590 |             eventEndTime.setHours(eventEndTime.getHours() + 24);
591 |             isPast = now > eventEndTime;
592 |             isGreyedOut = now > eventEndTime;
593 |           }
594 |         } catch (error) {
595 |           console.error("Error calculating event time:", error);
596 |         }
597 |
598 |         return {
599 |           ...event,
600 |           isJoined: currentUser?.uid && event.participants && Array.isArray(event.participants)
601 |             ? event.participants.includes(currentUser.uid)
602 |             : false,
603 |           distance,
604 |           distanceValue,
605 |           isPast,
606 |           isGreyedOut,
607 |         };
608 |       });
609 |
610 |       // Add dummy events if enabled and no real events exist
611 |       if (ENABLE_DUMMY_DATA && transformedEvents.length === 0 && userLocation) {
612 |         const dummyEvents = generateDummyEvents(userLocation);
613 |         transformedEvents = dummyEvents.map((event) => ({
614 |           ...event,
615 |           isJoined: currentUser?.uid ? event.participants.includes(currentUser.uid) : false,
616 |         }));
617 |       }
618 |
619 |       setEvents(transformedEvents);
620 |       setLoading(false);
621 |     });
622 |
623 |     return () => unsubscribe();
624 |   }, [currentUser?.uid, userLocation, authLoading]);
625 |
626 |   // Listen to check-ins for each event
627 |   useEffect(() => {
628 |     const unsubscribes: (() => void)[] = [];
629 |
630 |     events.forEach((event) => {
631 |       const unsubscribe = listenToEventCheckIns(event.id, (checkIns) => {
632 |         setCheckInCounts((prev) => ({
633 |           ...prev,
634 |           [event.id]: checkIns.length,
```

```
635 |         }));
636 |       });
637 |       unsubscribes.push(unsubscribe);
638 |     });
639 |
640 |     return () => {
641 |       unsubscribes.forEach((unsub) => unsub());
642 |     };
643 |   }, [events]);
644 |
645 |   // Filter events
646 |   const filteredEvents = events.filter((event) => {
647 |     const matchesActivity = activityFilter === "all" || event.type === activityFilter;
648 |     const matchesCategory = categoryFilter === "all" || event.category === categoryFilter;
649 |
650 |     // Search filtering - case-insensitive search in title, description, and location
651 |     const matchesSearch = searchQuery.trim() === "" ||
652 |       event.title.toLowerCase().includes(searchQuery.toLowerCase()) ||
653 |       event.description.toLowerCase().includes(searchQuery.toLowerCase()) ||
654 |       event.location.toLowerCase().includes(searchQuery.toLowerCase());
655 |
656 |     return matchesActivity && matchesCategory && matchesSearch;
657 |   });
658 |
659 |   // Sort by distance
660 |   const sortedEvents = [...filteredEvents].sort((a, b) => a.distanceValue - b.distanceValue);
661 |
662 |   // State for search results dropdown
663 |   const [showSearchResults, setShowSearchResults] = useState(false);
664 |   const searchInputRef = useRef<HTMLInputElement>(null);
665 |
666 |   // Handle selecting a search result
667 |   const handleSearchResultClick = (event: Event) => {
668 |     // Pan map to event location
669 |     if (mapRef) {
670 |       mapRef.panTo({ lat: event.lat, lng: event.lng });
671 |       mapRef.setZoom(15);
672 |     }
673 |     setMapCenter({ lat: event.lat, lng: event.lng });
674 |
675 |     // Show event details
676 |     setSelectedEvent(event);
677 |     setShowEventDetail(true);
678 |
679 |     // Close search dropdown and clear search
680 |     setShowSearchResults(false);
681 |     setSearchQuery("");
682 |
683 |     // Switch to map view if in list view
684 |     if (viewMode !== "map") {
685 |       setViewMode("map");
686 |     }
687 |   };
688 |
689 |   const handleJoinEvent = async (eventId: string) => {
690 |     if (!currentUser?.uid) {
691 |       toast.error("Please log in to join events");
692 |       return;
693 |     }
694 |
695 |     try {
696 |       await joinEvent(eventId, currentUser.uid);
697 |       toast.success("You've joined the event!");
698 |
699 |       // Update the selectedEvent to reflect the join immediately
700 |       setSelectedEvent((prev) => {
701 |         if (!prev || prev.id !== eventId) return prev;
702 |         const currentParticipants = Array.isArray(prev.participants) ? prev.participants : [];
703 |         if (!currentParticipants.includes(currentUser.uid)) {
704 |           return {
705 |             ...prev,
```

```
706 |             participants: [...currentParticipants, currentUser.uid],
707 |             isJoined: true
708 |           };
709 |         }
710 |         return prev;
711 |       });
712 |     } catch (error: any) {
713 |       console.error("Error joining event:", error);
714 |       toast.error(error.message || "Failed to join event");
715 |     }
716 |   };
717 |
718 |   const handleLeaveEvent = async (eventId: string) => {
719 |     if (!currentUser?.uid) {
720 |       toast.error("Please log in to leave events");
721 |       return;
722 |     }
723 |
724 |     try {
725 |       await leaveEvent(eventId, currentUser.uid);
726 |       toast.success("You've left the event");
727 |
728 |       // Update the selectedEvent to reflect the leave immediately
729 |       setSelectedEvent((prev) => {
730 |         if (!prev || prev.id !== eventId) return prev;
731 |         const currentParticipants = Array.isArray(prev.participants) ? prev.participants : [];
732 |         return {
733 |           ...prev,
734 |           participants: currentParticipants.filter(id => id !== currentUser.uid),
735 |           isJoined: false
736 |         };
737 |       });
738 |     } catch (error: any) {
739 |       console.error("Error leaving event:", error);
740 |       toast.error(error.message || "Failed to leave event");
741 |     }
742 |   };
743 |
744 |   const handleEventClick = (event: Event) => {
745 |     // Clear any existing route when selecting a new event
746 |     setShowRoute(false);
747 |     setRoutePath([]);
748 |     setRouteDestination(null);
749 |
750 |     // Zoom to event location if map is available and in map view
751 |     if (mapRef && viewMode === "map") {
752 |       mapRef.panTo({ lat: event.lat, lng: event.lng });
753 |       mapRef.setZoom(15);
754 |     }
755 |
756 |     setSelectedEvent(event);
757 |     setShowEventDetail(true);
758 |   };
759 |
760 |   const handleEditEvent = (eventId: string) => {
761 |     if (!currentUser?.uid) {
762 |       toast.error("Please log in to edit events");
763 |       return;
764 |     }
765 |
766 |     const targetEvent = events.find((event) => event.id === eventId);
767 |     if (!targetEvent) {
768 |       toast.error("Event not found");
769 |       return;
770 |     }
771 |
772 |     if (targetEvent.hostId !== currentUser.uid) {
773 |       toast.error("Only the event creator can edit this event");
774 |       return;
775 |     }
776 |
```

```
777 |     setEventBeingEdited(targetEvent);
778 |     setShowEditEvent(true);
779 |     setShowEventDetail(false);
780 |   };
781 |
782 |   const handleCloseEditModal = () => {
783 |     setShowEditEvent(false);
784 |     setEventBeingEdited(null);
785 |   };
786 |
787 |   const handleUpdateEvent = async (eventId: string, updatedData: CreateEventFormData) => {
788 |     if (!currentUser?.uid) {
789 |       toast.error("Please log in to edit events");
790 |       return;
791 |     }
792 |
793 |     const existingEvent = events.find((event) => event.id === eventId);
794 |
795 |     try {
796 |       await updateEvent(eventId, currentUser.uid, {
797 |         title: updatedData.title,
798 |         description: updatedData.description,
799 |         type: updatedData.activityType,
800 |         date: updatedData.date,
801 |         time: updatedData.time,
802 |         location: updatedData.location,
803 |         lat: updatedData.lat ?? existingEvent?.lat,
804 |         lng: updatedData.lng ?? existingEvent?.lng,
805 |         maxParticipants: updatedData.maxParticipants,
806 |       });
807 |
808 |       toast.success("Event updated successfully");
809 |
810 |       setSelectedEvent((prev) =>
811 |         prev && prev.id === eventId
812 |           ? {
813 |               ...prev,
814 |               title: updatedData.title,
815 |               description: updatedData.description,
816 |               type: updatedData.activityType as EventType,
817 |               date: updatedData.date,
818 |               time: updatedData.time,
819 |               location: updatedData.location,
820 |               lat: updatedData.lat ?? prev.lat,
821 |               lng: updatedData.lng ?? prev.lng,
822 |               maxParticipants: updatedData.maxParticipants,
823 |             }
824 |           : prev
825 |       );
826 |
827 |       handleCloseEditModal();
828 |     } catch (error: any) {
829 |       console.error("Error updating event:", error);
830 |       toast.error(error.message || "Failed to update event");
831 |     }
832 |   };
833 |
834 |   const handleDeleteEvent = async (eventId: string) => {
835 |     if (!currentUser?.uid) {
836 |       toast.error("Please log in to delete events");
837 |       return;
838 |     }
839 |
840 |     try {
841 |       await deleteEvent(eventId, currentUser.uid);
842 |       toast.success("Event deleted successfully");
843 |       setShowEventDetail(false);
844 |       setSelectedEvent(null);
845 |       // The real-time listener will update the events automatically
846 |     } catch (error: any) {
847 |       console.error("Error deleting event:", error);
```

```
848 |       toast.error(error.message || "Failed to delete event");
849 |     }
850 |   };
851 |
852 |   const handleCreateEvent = async (eventData: any) => {
853 |     if (!currentUser?.uid) {
854 |       toast.error("Please log in to create events");
855 |       return;
856 |     }
857 |
858 |     try {
859 |       // Get user data for host info
860 |       const userData = await getUserData(currentUser.uid);
861 |
862 |       // Get location - use selected location from form, then userLocation, then user's saved
location or default
863 |       let eventLat = eventData.lat || userLocation?.lat || userData?.lat || 14.5995;
864 |       let eventLng = eventData.lng || userLocation?.lng || userData?.lng || 120.9842;
865 |
866 |       // Format date properly (ISO string)
867 |       const eventDate = new Date(eventData.date);
868 |       const formattedDate = eventDate.toISOString().split('T')[0];
869 |
870 |       // Prepare event data, only include maxParticipants if it's defined
871 |       const eventPayload: any = {
872 |         title: eventData.title,
873 |         description: eventData.description,
874 |         type: eventData.activityType,
875 |         category: "user",
876 |         date: formattedDate,
877 |         time: eventData.time,
878 |         location: eventData.location,
879 |         distance: "0.0 km", // Will be calculated by listeners
880 |         distanceValue: 0,
881 |         lat: eventLat,
882 |         lng: eventLng,
883 |         hostName: userData?.name || currentUser.displayName || "User",
884 |         hostAvatar: userData?.photoURL || currentUser.photoURL || "",
885 |       };
886 |
887 |       // Only add maxParticipants if it's defined and greater than 0
888 |       if (eventData.maxParticipants && eventData.maxParticipants > 0) {
889 |         eventPayload.maxParticipants = eventData.maxParticipants;
890 |       }
891 |
892 |       // Create event in Firebase
893 |       const eventId = await createEvent(currentUser.uid, eventPayload);
894 |
895 |       toast.success("Event created successfully!");
896 |       setShowCreateEvent(false);
897 |       // The real-time listener will update the events automatically
898 |     } catch (error: any) {
899 |       console.error("Error creating event:", error);
900 |       toast.error(error.message || "Failed to create event");
901 |     }
902 |   };
903 |
904 |   // Handle inline event form submission
905 |   const handleInlineEventSubmit = async () => {
906 |     if (!currentUser?.uid) {
907 |       toast.error("Please log in to create events");
908 |       return;
909 |     }
910 |
911 |     if (!selectedEventLocation) {
912 |       toast.error("Please select a location on the map");
913 |       return;
914 |     }
915 |
916 |     // Validate form
917 |     if (!inlineEventForm.title.trim()) {
918 |       toast.error("Please enter an event title");
```

```
      return;
    }
    if (!inlineEventForm.description.trim()) {
      toast.error("Please enter an event description");
      return;
    }
    if (!inlineEventForm.date) {
      toast.error("Please select a date");
      return;
    }
    if (!inlineEventForm.time) {
      toast.error("Please select a time");
      return;
    }

    setIsSubmittingEvent(true);

    try {
      // Get user data for host info
      const userData = await getUserData(currentUser.uid);

      // Format date properly (ISO string)
      const eventDate = new Date(inlineEventForm.date);
      const formattedDate = eventDate.toISOString().split('T')[0];

      // Prepare event data
      const eventPayload: any = {
        title: inlineEventForm.title,
        description: inlineEventForm.description,
        type: inlineEventForm.activityType,
        category: "user",
        date: formattedDate,
        time: inlineEventForm.time,
        location: pinModeAddress || `${selectedEventLocation.lat.toFixed(6)},
${selectedEventLocation.lng.toFixed(6)}`,
        distance: "0.0 km",
        distanceValue: 0,
        lat: selectedEventLocation.lat,
        lng: selectedEventLocation.lng,
        hostName: userData?.name || currentUser.displayName || "User",
        hostAvatar: userData?.photoURL || currentUser.photoURL || "",
      };

      if (inlineEventForm.maxParticipants && inlineEventForm.maxParticipants > 0) {
        eventPayload.maxParticipants = inlineEventForm.maxParticipants;
      }

      // Create event in Firebase
      await createEvent(currentUser.uid, eventPayload);

      toast.success("Event created successfully!");

      // Reset form and close
      setShowEventForm(false);
      setSelectedEventLocation(null);
      setPinModeAddress("");
      setInlineEventForm({
        title: "",
        description: "",
        activityType: "running",
        date: "",
        time: "",
        maxParticipants: undefined,
      });
    } catch (error: any) {
      console.error("Error creating event:", error);
      toast.error(error.message || "Failed to create event");
    } finally {
      setIsSubmittingEvent(false);
    }
  };

```

```
  // Cancel inline event creation
  const handleCancelInlineEvent = () => {
    setShowEventForm(false);
    setSelectedEventLocation(null);
    setPinModeAddress("");
    setSearchValue("");
    setInlineEventForm({
      title: "",
      description: "",
      activityType: "running",
      date: "",
      time: "",
      maxParticipants: undefined,
    });
  };

  // Handle place selection from autocomplete - only pan map, don't confirm location yet
  const handlePlaceSelect = () => {
    if (autocomplete) {
      const place = autocomplete.getPlace();
      if (place.geometry && place.geometry.location) {
        const lat = place.geometry.location.lat();
        const lng = place.geometry.location.lng();

        // Set flag to prevent handleMapIdle from overwriting this location (sync via ref)
        isSearchLocationSetRef.current = true;

        // Update map center state to prevent snapping back
        setMapCenter({ lat, lng });
        setMapZoom(16);

        // Pan map to the selected location (pin stays in center)
        if (mapRef) {
          mapRef.panTo({ lat, lng });
          mapRef.setZoom(16);
        }

        // Update the pending location and address - this is the exact search result location
        setPendingLocation({ lat, lng });
        setPinModeAddress(place.formatted_address || place.name || "");
        setSearchValue(place.formatted_address || place.name || "");
      }
    }
  };

  // Reset search location flag when user manually drags the map
  const handleMapDragStart = useCallback(() => {
    isSearchLocationSetRef.current = false;
  }, []);

  // Reverse geocode map center when user stops dragging
  const handleMapIdle = useCallback(() => {
    if (!isPinMode || !mapRef) return;

    // Skip if location was just set from search result (prevent overwriting)
    if (isSearchLocationSetRef.current) {
      return;
    }

    const center = mapRef.getCenter();
    if (!center) return;

    const lat = center.lat();
    const lng = center.lng();

    setPendingLocation({ lat, lng });
    setIsReverseGeocoding(true);

    // Reverse geocode to get address
    const geocoder = new window.google.maps.Geocoder();
    geocoder.geocode({ location: { lat, lng } }, (results, status) => {
```

```
1061 |       setIsReverseGeocoding(false);
1062 |       if (status === "OK" && results && results[0]) {
1063 |         setPinModeAddress(results[0].formatted_address);
1064 |       } else {
1065 |         setPinModeAddress(`${lat.toFixed(6)}, ${lng.toFixed(6)}`);
1066 |       }
1067 |     });
1068 |   }, [isPinMode, mapRef]);
1069 |
1070 |   // Confirm the current map center as the event location
1071 |   const handleConfirmLocation = () => {
1072 |     if (pendingLocation) {
1073 |       setSelectedEventLocation(pendingLocation);
1074 |       setShowEventForm(true);
1075 |       setIsPinMode(false);
1076 |     }
1077 |   };
1078 |
1079 |   // Center map on user's GPS location
1080 |   const handleCenterOnMe = async () => {
1081 |     // Request location if not available
1082 |     const location = await requestUserLocation();
1083 |
1084 |     if (location && mapRef) {
1085 |       setMapCenter({ lat: location.lat, lng: location.lng });
1086 |       mapRef.panTo({ lat: location.lat, lng: location.lng });
1087 |       mapRef.setZoom(15);
1088 |     }
1089 |   };
1090 |
1091 |   // Fit all events on the map
1092 |   const handleFitAllEvents = () => {
1093 |     if (!mapRef || !isLoaded || sortedEvents.length === 0) {
1094 |       toast.info("No events to display");
1095 |       return;
1096 |     }
1097 |
1098 |     try {
1099 |       const bounds = new window.google.maps.LatLngBounds();
1100 |
1101 |       // Add all event locations to bounds
1102 |       sortedEvents.forEach((event) => {
1103 |         if (event.lat && event.lng) {
1104 |           bounds.extend(new window.google.maps.LatLng(event.lat, event.lng));
1105 |         }
1106 |       });
1107 |
1108 |       // If user location is available, include it in bounds
1109 |       if (userLocation) {
1110 |         bounds.extend(new window.google.maps.LatLng(userLocation.lat, userLocation.lng));
1111 |       }
1112 |
1113 |       // Fit bounds with padding
1114 |       mapRef.fitBounds(bounds);
1115 |       mapRef.setOptions({
1116 |         padding: { top: 100, right: 100, bottom: 200, left: 100 } // Extra bottom padding for
action bar
1117 |       });
1118 |
1119 |       toast.success(`Showing ${sortedEvents.length} event${sortedEvents.length !== 1 ? 's' : ''}
in view`);
1120 |     } catch (error) {
1121 |       console.error("Error fitting events to bounds:", error);
1122 |       toast.error("Could not fit all events on map");
1123 |     }
1124 |   };
1125 |
1126 |   const getActivityIcon = (type: EventType) => {
1127 |     switch (type) {
1128 |       case "running":
1129 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 20 }} />;
1130 |       case "cycling":
1131 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 20 }} />;
```

```
      case "walking":
        return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 20 }} />;
      case "others":
        return <FitnessCenterIcon className="text-secondary" style={{ fontSize: 20 }} />;
    }
  };

  const getActivityColor = (type: EventType) => {
    switch (type) {
      case "running":
        return "success";
      case "cycling":
        return "primary";
      case "walking":
        return "warning";
      case "others":
        return "secondary";
    }
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/10 via-background to-success/10 pb-20">
      {/* Header */}
      <motion.div
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b border-border/50"
        style={{
          paddingTop: 'env(safe-area-inset-top)',
        }}
      >
        <div className="max-w-2xl mx-auto px-6 py-5">
          <div className="flex items-center justify-between">
            <h1 className="text-3xl font-bold">Events</h1>
            {/* Notification Bell - Always Visible */}
            <NotificationBell
              unreadCount={unreadCount}
              onClick={() => setShowNotificationDrawer(true)}
              variant="light"
            />
          </div>
        </div>
      </motion.div>

      {/* Content */}
      <div className="max-w-7xl mx-auto px-0 sm:px-4 sm:px-6 pb-20 sm:pb-24">
        <AnimatePresence mode="wait">
          {viewMode === "map" ? (
            <motion.div
              key="map"
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              exit={{ opacity: 0, y: -20 }}
              transition={{ duration: 0.3 }}
              className="relative w-full"
            >
              {/* Map Container */}
              <div className="relative w-full h-[calc(100vh-200px)] sm:h-[600px] rounded-2xl overflow-hidden shadow-elevation-3">
                {/* Pin Mode Banner with Search */}
                {isPinMode && (
                  <motion.div
                    initial={{ opacity: 0, y: -20 }}
                    animate={{ opacity: 1, y: 0 }}
                    exit={{ opacity: 0, y: -20 }}
                    className="absolute top-4 left-4 right-4 z-50"
                  >
                    <div className="bg-card rounded-xl shadow-lg border border-border overflow-hidden">
                      {/* Search Input */}
                      <div className="p-3 border-b border-border">
                        <Autocomplete
                          onLoad={(auto) => setAutocomplete(auto)}
```

```
                          onPlaceChanged={handlePlaceSelect}
                        >
                          <div className="relative">
                            <input
                              type="text"
                              placeholder="Search for a location..."
                              value={searchValue}
                              onChange={(e) => setSearchValue(e.target.value)}
                              className="w-full pl-10 pr-4 py-2.5 bg-muted/50 border border-
border rounded-lg text-foreg...    />
                            <LocationOnIcon
                              style={{ fontSize: 20 }}
                              className="absolute left-3 top-1/2 -translate-y-1/2 text-muted-
foreground"                    />
                          </div>
                        </Autocomplete>
                      </div>

                      {/* Current Address Display */}
                      <div className="px-3 py-2.5 bg-muted/30 border-b border-border">
                        <div className="flex items-center gap-2">
                          <div className="flex-1 min-w-0">
                            {isReverseGeocoding ? (
                              <div className="flex items-center gap-2 text-muted-foreground">
                                <div className="w-4 h-4 border-2 border-primary border-t-
transparent rounded-full animat...    <span className="text-sm">Getting address...</span>
                              </div>
                            ) : pinModeAddress ? (
                              <p className="text-sm text-foreground truncate">{pinModeAddress}</
p>                          ) : (
                              <p className="text-sm text-muted-foreground">Drag map to select
location</p>                  )}
                          </div>
                        </div>
                      </div>

                      {/* Instructions */}
                      <div className="px-3 py-2 bg-primary/5 flex items-center justify-between
gap-2">                 <div className="flex items-center gap-2 text-muted-foreground">
                          <MapIcon style={{ fontSize: 16 }} />
                          <span className="text-xs">Drag the map to move the pin</span>
                        </div>
                        <Button
                          variant="ghost"
                          size="sm"
                          onClick={() => {
                            setIsPinMode(false);
                            setSelectedEventLocation(null);
                            setPendingLocation(null);
                            setPinModeAddress("");
                            setSearchValue("");
                          }}
                          className="text-destructive hover:bg-destructive/10 h-7 px-2 text-xs"
                        >
                          Cancel
                        </Button>
                      </div>
                    </div>
                  </motion.div>
                )}

                {/* Center-fixed pin overlay when in pin mode (Uber-style) */}
                {isPinMode && (
                  <div className="pointer-events-none absolute inset-0 flex items-center justify-
center z-40">             <div className="relative flex flex-col items-center">
                      {/* Pin icon - positioned so tip is at center */}
                      <div className="transform -translate-y-1/2">
                        <LocationOnIcon
                          style={{ fontSize: 56, color: '#ef4444' }}
                          className="drop-shadow-lg"
                        />
```

```
                      </div>
                      {/* Shadow dot at the exact center */}
                      <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2 w-2 h-2 bg-black/30 ro...
                    </div>
                  </div>
                )}

                {/* Center on Me Button - Floating button in pin mode */}
                {isPinMode && userLocation && (
                  <motion.button
                    initial={{ opacity: 0, scale: 0.8 }}
                    animate={{ opacity: 1, scale: 1 }}
                    whileTap={{ scale: 0.95 }}
                    onClick={handleCenterOnMe}
                    className="absolute bottom-24 right-4 z-50 w-12 h-12 bg-card rounded-full shadow-lg border border-bo...
                    title="Center on my location"
                  >
                    <MyLocationIcon style={{ fontSize: 24 }} className="text-primary" />
                  </motion.button>
                )}

                {/* Confirm Location Button - Bottom of map in pin mode */}
                {isPinMode && (
                  <motion.div
                    initial={{ opacity: 0, y: 20 }}
                    animate={{ opacity: 1, y: 0 }}
                    className="absolute bottom-4 left-4 right-4 z-50"
                  >
                    <Button
                      onClick={handleConfirmLocation}
                      disabled={!pendingLocation || isReverseGeocoding}
                      className="w-full py-6 text-base font-semibold bg-primary hover:bg-primary/90 shadow-lg"
                    >
                      {isReverseGeocoding ? (
                        <span className="flex items-center gap-2">
                          <div className="w-5 h-5 border-2 border-white border-t-transparent rounded-full animate-spin" ...
                          Getting location...
                        </span>
                      ) : (
                        <span className="flex items-center gap-2">
                          <CheckCircleIcon style={{ fontSize: 22 }} />
                          Confirm Location
                        </span>
                      )}
                    </Button>
                  </motion.div>
                )}

                {loadError ? (
                  <div className="w-full h-full flex items-center justify-center bg-muted p-4">
                    <div className="text-center max-w-md">
                      <p className="text-destructive font-semibold mb-2 text-lg">& þ  Error loading Google Maps</p>
                      <p className="text-sm text-muted-foreground mb-3">
                        {!hasApiKey
                          ? "Google Maps API key is missing. Please add VITE_GOOGLE_MAPS_API_KEY to your .env file and r...
                          : loadError.message || "Unable to load Google Maps. Please check the troubleshooting steps bel... </p>
                      {hasApiKey && (
                        <ul className="text-xs text-muted-foreground text-left list-disc list-inside space-y-1 mt-3">
                          <li>API key is valid in Google Cloud Console</li>
                          <li>Maps JavaScript API is enabled</li>
                          <li>API key restrictions allow this domain/localhost</li>
                          <li>Internet connection is active</li>
                          <li>Check browser console for detailed error messages</li>
                        </ul>
                      )}
                      {!hasApiKey && (
                        <div className="mt-4 p-3 bg-primary/10 rounded-lg text-left">
                          <p className="text-xs font-semibold mb-2">Quick Fix:</p>
                          <ol className="text-xs text-muted-foreground list-decimal list-inside space-y-1">
                            <li>Create a <code className="bg-background px-1 py-0.5 rounded">.env</code> file in the pro...
```

```
                          <li>Add: <code className="bg-background px-1 py-0.5 rounded">VITE_GOOGLE_MAPS_API_KEY=your_k
                          <li>Restart your dev server</li>
                        </ol>
                      </div>
                    )}
                  </div>
                </div>
              ) : !isLoaded ? (
                <div className="w-full h-full flex items-center justify-center bg-muted">
                  <div className="text-center">
                    <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary mx-auto mb-4"></div> <p className="text-muted-foreground">Loading map...</p>
                  </div>
                </div>
              ) : (
                <GoogleMap
                  mapContainerStyle={{ width: "100%", height: "100%" }}
                  center={mapCenter}
                  zoom={mapZoom}
                  onLoad={(map) => {
                    setMapRef(map);
                    if (userLocation) {
                      map.panTo({ lat: userLocation.lat, lng: userLocation.lng });
                    }
                  }}
                  onClick={() => {
                    // In center-pin mode, clicking doesn't place markers - user drags map
                    // Just close event details panel when clicking on empty map space
                    if (showEventDetail) {
                      setShowEventDetail(false);
                      setSelectedEvent(null);
                      setShowRoute(false);
                      setRoutePath([]);
                      setRouteDestination(null);
                    }
                  }}
                  onIdle={handleMapIdle}
                  onDragStart={handleMapDragStart}
                  options={{
                    disableDefaultUI: true,
                    zoomControl: false,
                    streetViewControl: false,
                    mapTypeControl: false,
                    fullscreenControl: false,
                    clickableIcons: false, // Disable clicking on places/landmarks to prevent clutter
                    styles: [
                      // Hide/mute small POIs to reduce clutter
                      {
                        featureType: "poi",
                        elementType: "all",
                        stylers: [{ visibility: "off" }]
                      },
                      {
                        featureType: "poi.business",
                        elementType: "all",
                        stylers: [{ visibility: "off" }]
                      }
                    ]
                  }}
                >
                  {/* User Location Marker */}
                  {userLocation && (
                    <Marker
                      position={{ lat: userLocation.lat, lng: userLocation.lng }}
                      icon={{
                        url: "https://maps.google.com/mapfiles/ms/icons/blue-dot.png",
                        scaledSize: new window.google.maps.Size(32, 32),
                      }}
                      title="Your location"
                    />
                  )}
```

```
                    {/* Selected Event Location Marker (shown after location is confirmed) */}
                    {selectedEventLocation && showEventForm && (
                      <Marker
                        position={{ lat: selectedEventLocation.lat, lng: selectedEventLocation.lng }}
                        icon={{
                          url: "https://maps.google.com/mapfiles/ms/icons/red-dot.png",
                          scaledSize: new window.google.maps.Size(48, 48),
                        }}
                        title="New event location"
                        animation={window.google.maps.Animation.DROP}
                      />
                    )}

                    {/* Event Markers */}
                    {sortedEvents.map((event) => {
                      const checkInCount = checkInCounts[event.id] || 0;
                      const countdown = calculateCountdown(event.date, event.time);

                      return (
                        <OverlayView
                          key={event.id}
                          position={{ lat: event.lat, lng: event.lng }}
                          mapPaneName={OverlayView.OVERLAY_MOUSE_TARGET}
                        >
                          <div onClick={(e) => e.stopPropagation()}>
                            <EventMarker
                              event={event}
                              checkInCount={checkInCount}
                              countdown={countdown}
                              onClick={() => {
                                handleEventClick(event);
                              }}
                            />
                          </div>
                        </OverlayView>
                      );
                    })}

                    {/* Route Polyline - Show route from user location to event */}
                    {showRoute && routePath.length > 0 && isLoaded && window.google && (
                      <Polyline
                        path={routePath}
                        options={{
                          strokeColor: "#3b82f6",
                          strokeOpacity: 0.8,
                          strokeWeight: 5,
                          geodesic: true,
                          icons: window.google.maps ? [
                            {
                              icon: {
                                path: window.google.maps.SymbolPath.FORWARD_CLOSED_ARROW,
                                scale: 4,
                                strokeColor: "#3b82f6",
                              },
                              offset: "50%",
                              repeat: "100px",
                            },
                          ] : undefined,
                        }}
                      />
                    )}

                  </GoogleMap>
                )}
              </div>

              {/* Route Controls - Floating panel when route is shown */}
              <AnimatePresence>
                {showRoute && routeDestination && (
                  <motion.div
```

```
                    initial={{ opacity: 0, y: 20 }}
                    animate={{ opacity: 1, y: 0 }}
                    exit={{ opacity: 0, y: 20 }}
                    transition={{ type: "spring", stiffness: 300, damping: 25 }}
                    className="absolute bottom-20 left-0 right-0 z-40 px-4"
                    style={{
                      bottom: `calc(5rem + env(safe-area-inset-bottom, 48px))`,
                    }}
                  >
                    <Card className="bg-card/95 backdrop-blur-md border-border shadow-
elevation-4 p-4 max-w-md mx-auto">
                      <div className="space-y-3">
                        <div className="flex items-center gap-3">
                          {/* Clear Route Button */}
                          <Button
                            variant="outline"
                            size="sm"
                            onClick={clearRoute}
                            className="flex-1 gap-2"
                          >
                            <ClearIcon style={{ fontSize: 18 }} />
                            <span>Clear Route</span>
                          </Button>

                          {/* Open in Google Maps Button */}
                          <Button
                            onClick={openRouteInGoogleMaps}
                            className="flex-1 gap-2 bg-gradient-to-r from-primary via-primary to-
success hover:from-prim...
                          >
                            <LaunchIcon style={{ fontSize: 18 }} />
                            <span>Open in Google Maps</span>
                          </Button>
                        </div>

                        {/* Helper text */}
                        <p className="text-xs text-muted-foreground text-center">
                          Opens Google Maps app or browser for turn-by-turn navigation
                        </p>
                      </div>
                    </Card>
                  </motion.div>
                )}
              </AnimatePresence>

              {/* Middle Right Controls - Events List and Create Event FAB */}
              <div className="fixed top-1/2 -translate-y-1/2 right-4 flex flex-col gap-3 z-50">
                {/* Events Drawer/List Button */}
                <motion.button
                  whileTap={{ scale: 0.95 }}
                  onClick={() => setDrawerOpen(true)}
                  className="touch-target rounded-full shadow-elevation-3 border-2 bg-card/90
backdrop-blur-sm text-fore...
                  style={{ width: 56, height: 56 }}
                  title="Events list"
                >
                  <ViewListIcon style={{ fontSize: 28 }} />
                  {sortedEvents.length > 0 && (
                    <Badge
                      className="absolute -top-1 -right-1 bg-success text-white text-xs min-w-
[20px] h-5 flex items-cent...
                      {sortedEvents.length > 9 ? '9+' : sortedEvents.length}
                    </Badge>
                  )}
                </motion.button>

                {/* Fit All Events Button */}
                {viewMode === "map" && sortedEvents.length > 0 && !isPinMode && (
                  <motion.button
                    whileTap={{ scale: 0.95 }}
                    onClick={handleFitAllEvents}
                    className="touch-target rounded-full shadow-elevation-3 border-2 bg-card/90
backdrop-blur-sm text-fo...
                    title="Fit all events on map"
                  >
```

```
                    <FitScreenIcon style={{ fontSize: 28 }} />
                  </motion.button>
                )}

                {/* Create Event FAB / Cancel Button */}
                <AnimatePresence mode="wait">
                  {!isPinMode && !showEventForm && !showRoute && !showEventDetail && (
                    <motion.button
                      key="create-event"
                      initial={{ opacity: 0, scale: 0.8 }}
                      animate={{ opacity: 1, scale: 1 }}
                      exit={{ opacity: 0, scale: 0.8 }}
                      transition={{ delay: 0.3, duration: 0.2, type: "spring", stiffness: 200 }}
                      whileTap={{ scale: 0.9 }}
                      whileHover={{ scale: 1.05 }}
                      onClick={() => {
                        setIsPinMode(true);
                        setSelectedEventLocation(null);
                        setPinModeAddress("");
                        // Initialize pending location to current map center
                        if (mapRef) {
                          const center = mapRef.getCenter();
                          if (center) {
                            setPendingLocation({ lat: center.lat(), lng: center.lng() });
                          }
                        } else {
                          setPendingLocation(mapCenter);
                        }
                      }}
                      className="w-14 h-14 rounded-full bg-gradient-to-r from-primary via-
primary to-success shadow-elev... title="Create event"
                    >
                      {/* Plus icon */}
                      <AddIcon style={{ fontSize: 28, color: "white" }} />
                    </motion.button>
                  )}
                  {isPinMode && !showEventForm && (
                    <motion.button
                      key="cancel-event"
                      initial={{ opacity: 0, scale: 0.8 }}
                      animate={{ opacity: 1, scale: 1 }}
                      exit={{ opacity: 0, scale: 0.8 }}
                      transition={{ duration: 0.2, type: "spring", stiffness: 200 }}
                      whileTap={{ scale: 0.9 }}
                      whileHover={{ scale: 1.05 }}
                      onClick={() => {
                        handleCancelInlineEvent();
                        setIsPinMode(false);
                      }}
                      className="w-14 h-14 rounded-full bg-destructive hover:bg-destructive/90
shadow-elevation-4 hover:... title="Cancel"
                    >
                      {/* X icon */}
                      <CloseIcon style={{ fontSize: 28, color: "white" }} />
                    </motion.button>
                  )}
                </AnimatePresence>
              </div>
            </motion.div>
          ) : (
            <motion.div
              key="list"
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              exit={{ opacity: 0, y: -20 }}
              transition={{ duration: 0.3 }}
              className="space-y-4"
            >
              {sortedEvents.length === 0 ? (
                <Card className="p-12 text-center shadow-elevation-2">
                  <EventIcon style={{ fontSize: 64 }} className="text-muted-foreground/30 mx-
auto mb-4" />
```

```
                  <h3 className="text-lg font-bold mb-2">No events found</h3>
                  <p className="text-muted-foreground text-sm">
                    Try adjusting your filters to see more events
                  </p>
                </Card>
              ) : (
                sortedEvents.map((event, index) => (
                  <EventCard
                    key={event.id}
                    event={event}
                    index={index}
                    onJoin={handleJoinEvent}
                    onClick={() => {
                      // Switch to map view first (we're in list view)
                      setViewMode("map");
                      // Wait a bit for map to render, then zoom and show details
                      setTimeout(() => {
                        handleEventClick(event);
                      }, 100);
                    }}
                    getActivityIcon={getActivityIcon}
                    getActivityColor={getActivityColor}
                    listView
                  />
                ))
              )}
            </motion.div>
          )}
        </AnimatePresence>
      </div>


      {/* Event Details Panel - Bottom Sliding Panel */}
      <EventDetailsPanel
        event={selectedEvent}
        onClose={() => {
          setShowEventDetail(false);
          setSelectedEvent(null);
          // Only clear route if user manually closes (not when showing directions)
          // Route will be cleared when a new event is selected or map is clicked
        }}
        onJoin={handleJoinEvent}
        onLeave={handleLeaveEvent}
        onEdit={handleEditEvent}
        onDelete={handleDeleteEvent}
        onShowRoute={(lat, lng) => {
          // Show the route on the map
          showDirectionsToEvent(lat, lng);
          // Close the event details panel so user can see the route
          setShowEventDetail(false);
          setSelectedEvent(null);
          // Switch to map view if not already there
          if (viewMode !== "map") {
            setViewMode("map");
          }
        }}
      />

      {/* Inline Event Creation Form - Bottom Sheet */}
      <AnimatePresence>
        {showEventForm && selectedEventLocation && (
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            exit={{ opacity: 0 }}
            transition={{ duration: 0.2 }}
            className="fixed inset-x-0 z-50 bg-card rounded-t-3xl shadow-elevation-4 border-t
border-border"
            style={{
              bottom: `calc(4.5rem + env(safe-area-inset-bottom, 0px))`,
              maxHeight: "calc(100vh - 6rem - env(safe-area-inset-bottom, 0px))",
            }}
```

```
          >
            {/* Handle bar */}
            <div className="flex justify-center pt-3 pb-2">
              <div className="w-12 h-1.5 bg-muted-foreground/30 rounded-full" />
            </div>

            {/* Header */}
            <div className="px-5 pb-3 border-b border-border flex items-center justify-between">
              <div>
                <h2 className="text-lg font-semibold text-foreground">Create Event</h2>
                <p className="text-sm text-muted-foreground truncate max-w-[280px]">
                  <LocationOnIcon style={{ fontSize: 14 }} className="mr-1" />
                  {pinModeAddress || "Selected location"}
                </p>
              </div>
              <Button
                variant="ghost"
                size="icon"
                onClick={handleCancelInlineEvent}
                className="h-9 w-9 rounded-full"
              >
                <CloseIcon style={{ fontSize: 20 }} />
              </Button>
            </div>

            {/* Form Content */}
            <div className="px-5 py-4 overflow-y-auto" style={{ maxHeight: "calc(100vh -
16rem)" }}>
              {/* Activity Type Selection */}
              <div className="mb-4">
                <Label className="text-sm font-medium text-foreground mb-2 block">Activity Type</
label>
                <div className="grid grid-cols-4 gap-2">
                  {(["running", "cycling", "walking", "others"] as const).map((type) => (
                    <button
                      key={type}
                      type="button"
                      onClick={() => setInlineEventForm(prev => ({ ...prev, activityType:
type }))}
                      className={`flex flex-col items-center gap-1 p-3 rounded-xl border-2
transition-all ${
                        inlineEventForm.activityType === type
                          ? "border-primary bg-primary/10"
                          : "border-border bg-muted/30 hover:border-primary/50"
                      }`}
                    >
                      {type === "running" && <DirectionsRunIcon style={{ fontSize: 24 }}
className={inlineEventForm.activ...
                      {type === "cycling" && <DirectionsBikeIcon style={{ fontSize: 24 }}
className={inlineEventForm.activ...
                      {type === "walking" && <DirectionsWalkIcon style={{ fontSize: 24 }}
className={inlineEventForm.activ...
                      {type === "others" && <FitnessCenterIcon style={{ fontSize: 24 }}
className={inlineEventForm.activ...
                      <span.className={`text-xs capitalize ${inlineEventForm.activityType ===
type ? "text-primary font-...  {type}
                      </span>
                    </button>
                  ))}
                </div>
              </div>

              {/* Event Title */}
              <div className="mb-4">
                <Label htmlFor="event-title" className="text-sm font-medium text-foreground mb-2
block">
                  Event Title
                </Label>
                <Input
                  id="event-title"
                  placeholder="e.g., Morning Run at the Park"
                  value={inlineEventForm.title}
                  onChange={(e) => setInlineEventForm(prev => ({ ...prev, title:
e.target.value }))}
                  className="bg-muted/50"
                />
              </div>

              {/* Date and Time */}
              <div className="grid grid-cols-2 gap-3 mb-4">
                <div>
```

```
                  <Label htmlFor="event-date" className="text-sm font-medium text-foreground
mb-2 block">
                    Date
                  </Label>
                  <Input
                    id="event-date"
                    type="date"
                    value={inlineEventForm.date}
                    min={new Date().toISOString().split('T')[0]}
                    onChange={(e) => setInlineEventForm(prev => ({ ...prev, date:
e.target.value }))}
                    className="bg-muted/50"
                  />
                </div>
                <div>
                  <Label htmlFor="event-time" className="text-sm font-medium text-foreground
mb-2 block">
                    Time
                  </Label>
                  <Input
                    id="event-time"
                    type="time"
                    value={inlineEventForm.time}
                    onChange={(e) => setInlineEventForm(prev => ({ ...prev, time:
e.target.value }))}
                    className="bg-muted/50"
                  />
                </div>
              </div>

              {/* Max Participants */}
              <div className="mb-4">
                <Label htmlFor="max-participants" className="text-sm font-medium text-foreground
mb-2 block">
                  Max Participants (Optional)
                </Label>
                <Input
                  id="max-participants"
                  type="number"
                  min={2}
                  max={1000}
                  placeholder="Leave empty for unlimited"
                  value={inlineEventForm.maxParticipants || ""}
                  onChange={(e) => setInlineEventForm(prev => ({
                    ...prev,
                    maxParticipants: e.target.value ? parseInt(e.target.value) : undefined
                  }))}
                  className="bg-muted/50"
                />
                <p className="text-xs text-muted-foreground mt-1">
                  Leave empty for unlimited participants
                </p>
              </div>

              {/* Description */}
              <div className="mb-4">
                <Label htmlFor="event-description" className="text-sm font-medium text-
foreground mb-2 block">
                  Description
                </Label>
                <Textarea
                  id="event-description"
                  placeholder="Tell people what to expect..."
                  value={inlineEventForm.description}
                  onChange={(e) => setInlineEventForm(prev => ({ ...prev, description:
e.target.value }))}
                  className="bg-muted/50 min-h-[80px]"
                />
              </div>
            </div>

            {/* Action Buttons */}
            <div className="px-5 pt-3 pb-5 border-t border-border flex gap-3">
              <Button
                variant="outline"
                className="flex-1"
                onClick={handleCancelInlineEvent}
              >
```

```
                Cancel
              </Button>
              <Button
                className="flex-1 bg-primary hover:bg-primary/90"
                onClick={handleInlineEventSubmit}
                disabled={isSubmittingEvent}
              >
                {isSubmittingEvent ? (
                  <span className="flex items-center gap-2">
                    <div className="w-4 h-4 border-2 border-white border-t-transparent rounded-full animate-spin" />
                    Creating...
                  </span>
                ) : (
                  "Create Event"
                )}
              </Button>
            </div>
          </motion.div>
        )}
      </AnimatePresence>

      {/* Keep Event Detail Modal as fallback for desktop if needed */}
      {showEventDetail && false && (
        <EventDetailModal
          event={selectedEvent}
          onClose={() => {
            setShowEventDetail(false);
            setSelectedEvent(null);
          }}
          onJoin={handleJoinEvent}
          onEdit={handleEditEvent}
          onDelete={handleDeleteEvent}
        />
      )}

      {/* Create Event Modal */}
      {showCreateEvent && (
        <CreateEventModal
          onClose={() => {
            setShowCreateEvent(false);
          }}
          onCreateEvent={handleCreateEvent}
        />
      )}

      {/* Edit Event Modal */}
      {showEditEvent && eventBeingEdited && (
        <CreateEventModal
          mode="edit"
          eventToEdit={eventBeingEdited as FirebaseEvent}
          onClose={handleCloseEditModal}
          onUpdateEvent={handleUpdateEvent}
        />
      )}

      {/* Quick Check-in Modal */}
      {/* Mobile Events Drawer */}
      <Drawer open={drawerOpen} onOpenChange={setDrawerOpen}>
        <DrawerContent className="max-h-[85vh]">
          <DrawerHeader className="border-b border-border pb-4">
            <div className="flex items-center justify-between gap-3">
              <div className="flex items-center gap-3 flex-1 min-w-0">
                <div className="p-2 bg-primary/10 rounded-lg flex-shrink-0">
                  <EventIcon className="text-primary" style={{ fontSize: 24 }} />
                </div>
                <div className="flex-1 min-w-0">
                  <DrawerTitle className="text-xl font-bold">Events</DrawerTitle>
                  <DrawerDescription className="text-xs mt-0.5">
                    {sortedEvents.length} event{sortedEvents.length !== 1 ? "s" : ""} found
                  </DrawerDescription>
                </div>
```

```
1913 |               </div>
1914 |               <Button
1915 |                 variant="ghost"
1916 |                 size="icon"
1917 |                 onClick={() => setDrawerOpen(false)}
1918 |                 className="h-8 w-8 flex-shrink-0"
1919 |                 aria-label="Close events drawer"
1920 |               >
1921 |                 <CloseIcon style={{ fontSize: 20 }} />
1922 |               </Button>
1923 |             </div>
1924 |           </DrawerHeader>
1925 |
1926 |           <div className="overflow-y-auto flex-1">
1927 |             {/* Filters inside drawer - Only activity filters */}
1928 |             <div className="p-4 border-b border-border bg-muted/30">
1929 |               <p className="text-xs font-semibold text-muted-foreground mb-3 uppercase tracking-
wide">
1930 |                 Filter by Activity
1931 |               </p>
1932 |               <div className="grid grid-cols-2 gap-2">
1933 |                 <motion.button
1934 |                   whileTap={{ scale: 0.95 }}
1935 |                   onClick={() => setActivityFilter("all")}
1936 |                   className={`px-4 py-3 rounded-xl text-sm font-medium transition-all flex items-
center justify-center g... activityFilter === "all"
1938 |                       ? "bg-primary text-primary-foreground shadow-elevation-2"
1939 |                       : "bg-background text-foreground hover:bg-accent border border-border"
1940 |                   }`}
1941 |                 >
1942 |                   <FilterListIcon style={{ fontSize: 18 }} />
1943 |                   All Activities
1944 |                 </motion.button>
1945 |                 <motion.button
1946 |                   whileTap={{ scale: 0.95 }}
1947 |                   onClick={() => setActivityFilter("running")}
1948 |                   className={`px-4 py-3 rounded-xl text-sm font-medium transition-all flex items-
center justify-center g... activityFilter === "running"
1950 |                       ? "bg-success text-success-foreground shadow-elevation-2"
1951 |                       : "bg-background text-foreground hover:bg-accent border border-border"
1952 |                   }`}
1953 |                 >
1954 |                   <DirectionsRunIcon style={{ fontSize: 20 }} />
1955 |                   Running
1956 |                 </motion.button>
1957 |                 <motion.button
1958 |                   whileTap={{ scale: 0.95 }}
1959 |                   onClick={() => setActivityFilter("cycling")}
1960 |                   className={`px-4 py-3 rounded-xl text-sm font-medium transition-all flex items-
center justify-center g... activityFilter === "cycling"
1962 |                       ? "bg-primary text-primary-foreground shadow-elevation-2"
1963 |                       : "bg-background text-foreground hover:bg-accent border border-border"
1964 |                   }`}
1965 |                 >
1966 |                   <DirectionsBikeIcon style={{ fontSize: 20 }} />
1967 |                   Cycling
1968 |                 </motion.button>
1969 |                 <motion.button
1970 |                   whileTap={{ scale: 0.95 }}
1971 |                   onClick={() => setActivityFilter("walking")}
1972 |                   className={`px-4 py-3 rounded-xl text-sm font-medium transition-all flex items-
center justify-center g... activityFilter === "walking"
1974 |                       ? "bg-warning text-warning-foreground shadow-elevation-2"
1975 |                       : "bg-background text-foreground hover:bg-accent border border-border"
1976 |                   }`}
1977 |                 >
1978 |                   <DirectionsWalkIcon style={{ fontSize: 20 }} />
1979 |                   Walking
1980 |                 </motion.button>
1981 |                 <motion.button
1982 |                   whileTap={{ scale: 0.95 }}
1983 |                   onClick={() => setActivityFilter("others")}
```

```
                  className={`px-4 py-3 rounded-xl text-sm font-medium transition-all flex items-center justify-center g... activityFilter === "others"
                      ? "bg-secondary text-secondary-foreground shadow-elevation-2"
                      : "bg-background text-foreground hover:bg-accent border border-border"
                  }`}
                >
                  <FitnessCenterIcon style={{ fontSize: 20 }} />
                  Others
                </motion.button>
              </div>
            </div>

            {/* Event List */}
            <div className="p-4 space-y-3">
              {sortedEvents.length === 0 ? (
                <div className="text-center py-12">
                  <EventIcon className="text-muted-foreground/30 mx-auto mb-3" style={{ fontSize: 48 }} />
                  <p className="text-sm font-medium text-muted-foreground">No events found</p>
                  <p className="text-xs text-muted-foreground mt-1">Try adjusting your filters</p>
                </div>
              ) : (
                sortedEvents.map((event, index) => {
                  const checkInCount = checkInCounts[event.id] || 0;
                  const countdown = calculateCountdown(event.date, event.time);

                  return (
                    <motion.div
                      key={event.id}
                      initial={{ opacity: 0, y: 20 }}
                      animate={{ opacity: 1, y: 0 }}
                      transition={{ delay: index * 0.05 }}
                      onClick={() => {
                        // Close drawer immediately first
                        setDrawerOpen(false);

                        // Then handle event click after a brief delay to allow drawer to close
                        setTimeout(() => {
                          if (mapRef) {
                            mapRef.panTo({ lat: event.lat, lng: event.lng });
                            mapRef.setZoom(15);
                          }
                          handleEventClick(event);
                          if (viewMode !== "map") {
                            setViewMode("map");
                          }
                        }, 100);
                      }}
                      className="cursor-pointer"
                    >
                      <Card className="p-4 hover:shadow-elevation-2 transition-all border border-border">
                        <div className="flex items-start gap-3">
                          <Avatar
                            src={event.hostAvatar}
                            alt={event.hostName}
                            sx={{ width: 48, height: 48 }}
                            className="flex-shrink-0"
                          />
                          <div className="flex-1 min-w-0">
                            <div className="flex items-center gap-2 mb-1.5">
                              <div className="flex-shrink-0">
                                {getActivityIcon(event.type)}
                              </div>
                              <h4 className="font-semibold text-sm truncate">{event.title}</h4>
                            </div>
                            <p className="text-xs text-muted-foreground mb-2">
                              {event.date} at {event.time}
                            </p>
                            <div className="flex items-center gap-2 text-xs text-muted-foreground flex-wrap">
                              <Badge variant="secondary" className="text-xs">
                                {countdown}
                              </Badge>
```

```
                              {checkInCount > 0 && (
                                <Badge variant="outline" className="text-xs">
                                  +{checkInCount} checked in
                                </Badge>
                              )}
                              <span className="text-xs">· {event.distance}</span>
                            </div>
                          </div>
                        </div>
                      </Card>
                    </motion.div>
                  );
                })
              )}
            </div>
          </div>
        </DrawerContent>
      </Drawer>

      {/* Quick Check-in, My Events, and View Toggle - Above Bottom Navigation */}
      <AnimatePresence>
        {!showEventDetail && !selectedEvent && !isPinMode && (
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            exit={{ opacity: 0, y: 20 }}
            transition={{ duration: 0.2 }}
            className="fixed left-0 right-0 z-40 px-4 pb-2"
            style={{
              bottom: `calc(5rem + env(safe-area-inset-bottom, 48px))`, // 80px + safe area
            }}
          >
            <div className="bg-card/95 backdrop-blur-md rounded-2xl p-3 shadow-elevation-3 border border-border/50 max-w
              <div className="flex items-center gap-2">
                {/* My Events Button */}
                <Button
                  variant="outline"
                  onClick={() => navigate("/my-events")}
                  className="flex-1 h-10 border-border bg-background hover:bg-secondary"
                >
                  <div className="flex items-center mr-2">
                    <CheckCircleIcon style={{ fontSize: 16 }} />
                  </div>
                  <span className="text-sm font-semibold">My Events</span>
                </Button>

                {/* Center User on Map Button */}
                {viewMode === "map" && (
                  <Button
                    variant="outline"
                    onClick={handleCenterOnMe}
                    className="h-10 border-border bg-background hover:bg-secondary"
                    disabled={isGettingLocation}
                  >
                    <MyLocationIcon style={{ fontSize: 16 }} className="mr-2" />
                    <span className="text-sm font-semibold">Center</span>
                  </Button>
                )}
              </div>
            </div>
          </motion.div>
        )}
      </AnimatePresence>

      {/* Notification Drawer */}
      <AnimatePresence>
        {showNotificationDrawer && (
          <>
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
```

```
              exit={{ opacity: 0 }}
              onClick={() => setShowNotificationDrawer(false)}
              className="fixed inset-0 bg-black/50 z-40"
            />
            <motion.div
              initial={{ y: "100%" }}
              animate={{ y: 0 }}
              exit={{ y: "100%" }}
              transition={{ type: "spring", damping: 25, stiffness: 300 }}
              onClick={(e) => e.stopPropagation()}
              className={`fixed bottom-0 left-0 right-0 z-50 bg-card rounded-t-3xl shadow-elevation-4 p-6 pb-24 border-t...
              style={{
                maxHeight: '85vh',
                minHeight: '200px',
                overflowY: 'auto'
              }}
            >
              {/* Header */}
              <div className="flex items-center justify-between mb-6">
                <div>
                  <h2 className="text-2xl font-bold text-foreground">Notification History</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    {notifications.length > 0
                      ? `${notifications.length} total notification${notifications.length !== 1 ? 's' : ''}${unreadCount...: 'No notifications yet'}
                  </p>
                </div>
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={() => setShowNotificationDrawer(false)}
                  className="rounded-full"
                >
                  <CloseIcon />
                </Button>
              </div>

              {/* Notifications List - Sorted by newest first */}
              {notifications.length > 0 ? (
                <div className="space-y-2">
                  {[...notifications]
                    .sort((a, b) => b.timestamp - a.timestamp) // Newest first
                    .map((notification) => {
                      const getNotificationIcon = () => {
                        switch (notification.type) {
                          case "message":
                            return <MailIcon style={{ fontSize: 20 }} className="text-primary" />;
                          case "message_request":
                            return <ChatBubbleIcon style={{ fontSize: 20 }} className="text-blue-500" />;
                          case "friend_request":
                            return <PersonAddIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "poke":
                            return <TouchAppIcon style={{ fontSize: 20 }} className="text-purple-500" />;
                          case "friend_accepted":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "workout_complete":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "achievement":
                            return <EmojiEventsIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "username_change_required":
                            return <EditIcon style={{ fontSize: 20 }} className="text-warning" />;
                          default:
                            return <NotificationsIcon style={{ fontSize: 20 }} />;
                        }
                      };

                      const getNotificationTitle = () => {
                        switch (notification.type) {
                          case "message":
                            return notification.userName;
                          case "message_request":
```

```
                            return notification.userName;
                          case "friend_request":
                            return notification.userName;
                          case "poke":
                            return notification.userName;
                          case "friend_accepted":
                            return notification.userName;
                          case "workout_complete":
                            return "Workout Completed";
                          case "achievement":
                            return notification.message || "Congrats for a new achievement!";
                          case "username_change_required":
                            return "Username Change Required";
                          default:
                            return notification.userName;
                        }
                      };

                      const getNotificationMessage = () => {
                        switch (notification.type) {
                          case "message":
                            return notification.message || "Sent you a message";
                          case "message_request":
                            return "wants to start a conversation with you";
                          case "friend_request":
                            return "wants to add you as a friend";
                          case "poke":
                            return "poked you! They're interested in matching";
                          case "friend_accepted":
                            return "accepted your friend request";
                          case "workout_complete":
                            return notification.message || "Workout completed successfully!";
                          case "achievement":
                            return notification.message || "Congrats for a new achievement!";
                          case "username_change_required":
                            return notification.message || "Your username has been changed due
to misuse. Please update ...
                          default:
                            return "";
                        }
                      };

                      const formatTimestamp = (timestamp: number) => {
                        const now = Date.now();
                        const diff = now - timestamp;
                        const minutes = Math.floor(diff / 60000);
                        const hours = Math.floor(diff / 3600000);
                        const days = Math.floor(diff / 86400000);

                        if (minutes < 1) return "Just now";
                        if (minutes < 60) return `${minutes}m ago`;
                        if (hours < 24) return `${hours}h ago`;
                        if (days < 7) return `${days}d ago`;
                        return new Date(timestamp).toLocaleDateString();
                      };

                      return (
                        <motion.div
                          key={notification.id}
                          initial={{ opacity: 0, y: 10 }}
                          animate={{ opacity: 1, y: 0 }}
                          className={`p-4 rounded-lg border border-border/50 hover:bg-muted/50
transition-colors cursor-...
                            !notification.read ? 'bg-primary/5 border-primary/30' : ''
                          }`}
                          onClick={() => {
                            handleNotificationTap(notification);
                            setShowNotificationDrawer(false);
                          }}
                        >
                          <div className="flex items-start gap-3">
                            {/* Icon */}
                            <div className={`flex-shrink-0 p-2 rounded-full ${
```

```
                                notification.type === "message"
                                  ? "bg-primary/15"
                                  : notification.type === "message_request"
                                  ? "bg-blue-500/15"
                                  : notification.type === "friend_request"
                                  ? "bg-warning/15"
                                  : notification.type === "poke"
                                  ? "bg-purple-500/15"
                                  : notification.type === "workout_complete"
                                  ? "bg-success/15"
                                  : notification.type === "achievement"
                                  ? "bg-warning/15"
                                  : notification.type === "friend_accepted"
                                  ? "bg-success/15"
                                  : "bg-gray-500/15"
                              }`}>
                                {getNotificationIcon()}
                              </div>

                              {/* Content */}
                              <div className="flex-1 min-w-0">
                                <div className="flex items-center justify-between mb-1">
                                  <p className="font-semibold text-sm text-foreground">
                                    {getNotificationTitle()}
                                  </p>
                                  {!notification.read && (
                                    <span className="w-2 h-2 bg-primary rounded-full flex-
shrink-0"></span>
                                  )}
                                </div>
                                <p className="text-sm text-muted-foreground mb-1">
                                  {getNotificationMessage()}
                                </p>
                                <p className="text-xs text-muted-foreground">
                                  {formatTimestamp(notification.timestamp)}
                                </p>
                              </div>

                              {/* Dismiss button */}
                              <button
                                onClick={(e) => {
                                  e.stopPropagation();
                                  dismissNotification(notification.id);
                                }}
                                className="flex-shrink-0 p-1 hover:bg-accent rounded-full
transition-colors"
                              >
                                <CloseIcon style={{ fontSize: 16 }} className="text-muted-
foreground" />
                              </button>
                            </div>
                          </motion.div>
                        );
                      })}
                    </div>
                  ) : (
                    <div className="py-12 text-center">
                      <NotificationsIcon className="text-muted-foreground mx-auto mb-2"
style={{ fontSize: 48 }} />
                      <p className="text-sm text-muted-foreground">No notifications yet</p>
                    </div>
                  )}
                </motion.div>
              </>
            )}
          </AnimatePresence>

          <BottomNavigation />
        </div>
      );
    };

    // Event Card Component
    interface EventCardProps {
      event: Event;
```

```
  index: number;
  onJoin: (id: string) => void;
  onClick: () => void;
  getActivityIcon: (type: EventType) => JSX.Element;
  getActivityColor: (type: EventType) => string;
  listView?: boolean;
}

const EventCard = ({ event, index, onJoin, onClick, getActivityIcon, getActivityColor, listView }: EventCardProps) => {
  return (
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: index * 0.05 }}
      onClick={onClick}
      className="cursor-pointer"
    >
      <Card className={`overflow-hidden shadow-elevation-2 hover:shadow-elevation-3 transition-all duration-300 border-2 ${event.isCreatedOut
          ? "opacity-50 grayscale border-muted bg-muted/20"
          : "border-border/50 hover:border-primary/30"
      }`}>
        <div className="p-5 space-y-4">
          {/* Header */}
          <div className="flex items-start justify-between gap-3">
            <div className="flex-1 min-w-0">
              <div className="flex items-center gap-2 mb-2">
                {getActivityIcon(event.type)}
                <h3 className="font-bold text-lg truncate">{event.title}</h3>
              </div>
              <p className="text-sm text-muted-foreground line-clamp-2">{event.description}</p>
            </div>
            {event.category === "sponsored" && (
              <Badge variant="secondary" className="flex-shrink-0 bg-warning/10 text-warning border-warning/30">
                <StarIcon style={{ fontSize: 14 }} className="mr-1" />
                Sponsored
              </Badge>
            )}
          </div>

          {/* Event Info */}
          <div className="space-y-2 text-sm">
            <div className="flex items-center gap-2 text-muted-foreground">
              <EventIcon style={{ fontSize: 18 }} />
              <span>{event.date} at {event.time}</span>
            </div>
            <div className="flex items-center gap-2 text-muted-foreground">
              <LocationOnIcon style={{ fontSize: 18 }} />
              <span>{event.location} · {event.distance} away</span>
            </div>
            <div className="flex items-center gap-2 text-muted-foreground">
              <PeopleIcon style={{ fontSize: 18 }} />
              <span>
                {event.participants.length} {event.maxParticipants ? `/ ${event.maxParticipants}` : ""} participants
              </span>
            </div>
          </div>

          {/* Host/Sponsor */}
          {event.category === "user" && event.hostName ? (
            <div className="flex items-center gap-3 pt-2 border-t border-border">
              <Avatar src={event.hostAvatar} alt={event.hostName} sx={{ width: 32, height: 32 }} />
              <div>
                <p className="text-xs text-muted-foreground">Hosted by</p>
                <p className="text-sm font-semibold">{event.hostName}</p>
              </div>
            </div>
          ) : event.sponsorLogo ? (
            <div className="flex items-center gap-3 pt-2 border-t border-border">
              <img src={event.sponsorLogo} alt="Sponsor" className="w-8 h-8 rounded" />
              <p className="text-sm font-semibold text-muted-foreground">Official Sponsor Event</p>
```

```
2410 |             </div>
2411 |           ) : null}
2412 |
2413 |           {/* Action Button */}
2414 |           <Button
2415 |             onClick={(e) => {
2416 |               e.stopPropagation();
2417 |               onJoin(String(event.id));
2418 |             }}
2419 |             className={`w-full h-11 font-semibold ${
2420 |               event.isJoined
2421 |                 ? "bg-success/20 text-success hover:bg-success/30 border-2 border-success"
2422 |                 : ""
2423 |             }`}
2424 |             variant={event.isJoined ? "outline" : "default"}
2425 |           >
2426 |             {event.isJoined ? "'  Joined" : "Join Event"}
2427 |           </Button>
2428 |         </div>
2429 |       </Card>
2430 |     </motion.div>
2431 |   );
2432 | };
2433 |
2434 | export default Events;
2435 |
```

File: src/pages/Friends.tsx

Lines: 1019

```
  1 | import { useState, useMemo, useEffect } from "react";
  2 | import { motion, AnimatePresence } from "framer-motion";
  3 | import { useNavigate, useLocation } from "react-router-dom";
  4 | import { Card } from "@/components/ui/card";
  5 | import { Button } from "@/components/ui/button";
  6 | import { Input } from "@/components/ui/input";
  7 | import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
  8 | import { Avatar } from "@mui/material";
  9 | import { ProfileView } from "@/pages/ProfileView";
 10 | import { useAuth } from "@/hooks/useAuth";
 11 | import { useLocation as useLocationHook } from "@/hooks/useLocation";
 12 | import { useMatching } from "@/hooks/useMatching";
 13 | import { useNearbyUsers } from "@/hooks/useNearbyUsers";
 14 | import { useUser } from "@/contexts/UserContext";
 15 | import { formatDistance, calculateDistance } from "@/utils/distance";
 16 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
 17 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
 18 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
 19 | import TouchAppIcon from "@mui/icons-material/TouchApp";
 20 | import SendIcon from "@mui/icons-material/Send";
 21 | import { sendPoke } from "@/services/pokeService";
 22 | import { isWorkoutActive } from "@/utils/workoutState";
 23 | import {
 24 |   listenToUserFriends,
 25 |   listenToFriendRequests,
 26 |   sendFriendRequest,
 27 |   acceptFriendRequest,
 28 |   declineFriendRequest,
 29 |   cancelFriendRequest,
 30 |   removeFriend,
 31 | } from "@/services/friendService";
 32 | import { getUserData } from "@/services/authService";
 33 | import {
 34 |   listenToEncounteredUsers,
 35 |   cleanupOldEncounters,
 36 |   EncounteredUser,
 37 | } from "@/services/encounteredUsersService";
 38 | import {
 39 |   getLocationSharingSettings,
 40 |   setLocationSharing as saveLocationSharing,
 41 | } from "@/lib/socialStorage";
 42 | import { getDisplayName } from "@/utils/anonymousName";
 43 | import { getProfilePictureUrl } from "@/utils/profilePicture";
 44 | import ArrowBackIcon from "@mui/icons-material/ArrowBack";
 45 | import SearchIcon from "@mui/icons-material/Search";
 46 | import PersonAddIcon from "@mui/icons-material/PersonAdd";
 47 | import ChatIcon from "@mui/icons-material/Chat";
 48 | import CheckIcon from "@mui/icons-material/Check";
 49 | import CloseIcon from "@mui/icons-material/Close";
 50 | import LocationOnIcon from "@mui/icons-material/LocationOn";
 51 | import LocationOffIcon from "@mui/icons-material/LocationOff";
 52 | import { toast } from "sonner";
 53 | import BottomNavigation from "@/components/BottomNavigation";
 54 | import { Switch } from "@/components/ui/switch";
 55 | import {
 56 |   AlertDialog,
 57 |   AlertDialogAction,
 58 |   AlertDialogCancel,
 59 |   AlertDialogContent,
 60 |   AlertDialogDescription,
 61 |   AlertDialogFooter,
 62 |   AlertDialogHeader,
 63 |   AlertDialogTitle,
 64 | } from "@/components/ui/alert-dialog";
 65 |
 66 | const Friends = () => {
```

```
  67 |   const navigate = useNavigate();
  68 |   const location = useLocation();
  69 |   const { user } = useAuth();
  70 |   const { userProfile } = useUser();
  71 |   const [searchQuery, setSearchQuery] = useState("");
  72 |   const [requests, setRequests] = useState<{ incoming: string[]; outgoing: string[] }
>({ incoming: [], outgoing: [] });
  73 |   const [friends, setFriends] = useState<string[]>([]);
  74 |   const [friendsData, setFriendsData] = useState<Record<string, any>>({});
  75 |   const [requestsData, setRequestsData] = useState<Record<string, any>>({});
  76 |   const [selectedUser, setSelectedUser] = useState<any | null>(null);
  77 |   const [locationSharing, setLocationSharing] = useState<Record<string,
boolean>>(getLocationSharingSettings());
  78 |   const [unfriendDialog, setUnfriendDialog] = useState<{ open: boolean; friendId: string | null;
friendName: string }>({
  79 |     open: false,
  80 |     friendId: null,
  81 |     friendName: "",
  82 |   });
  83 |   const [hasPokedUsers, setHasPokedUsers] = useState<Record<string, boolean>>({});
  84 |   const [encounteredUsers, setEncounteredUsers] = useState<Record<string, EncounteredUser>>({});
  85 |   const [encounteredUsersLoading, setEncounteredUsersLoading] = useState(true);
  86 |   const [encounteredUsersData, setEncounteredUsersData] = useState<Record<string, any>>({});
  87 |
  88 |   // Get user location for nearby users
  89 |   const { location: currentLocation } = useLocationHook(user?.uid || null, false, true);
  90 |
  91 |   // Get initial tab from location state, default to "friends"
  92 |   const initialTab = (location.state as { tab?: string })?.tab || "friends";
  93 |
  94 |   // Get nearby users using matching algorithm (for other features)
  95 |   const { matches, loading: matchesLoading } = useMatching({
  96 |     currentUserId: user?.uid || "",
  97 |     currentLocation: currentLocation,
  98 |     activity: userProfile?.activities?.[0] || "running",
  99 |     fitnessLevel: userProfile?.fitnessLevel || "intermediate",
 100 |     pace: userProfile?.pace,
 101 |     visibility: userProfile?.visibility || {
 102 |       visibleToAllLevels: true,
 103 |       allowedLevels: ["beginner", "intermediate", "pro"]
 104 |     },
 105 |     searchFilter: "all",
 106 |     radiusPreference: userProfile?.radiusPreference || "normal"
 107 |   });
 108 |
 109 |   // Get all nearby users (not just matches) for discover tab
 110 |   // This shows all users you've encountered nearby, regardless of matching criteria
 111 |   const maxDistanceKm = userProfile?.radiusPreference === "wide" ? 10 :
 112 |                         userProfile?.radiusPreference === "narrow" ? 3 : 5;
 113 |   const { nearbyUsers: allNearbyUsers, loading: nearbyUsersLoading } = useNearbyUsers(
 114 |     currentLocation,
 115 |     maxDistanceKm,
 116 |     "all", // Show all activities
 117 |     "all", // Show all genders
 118 |     user?.uid || null
 119 |   );
 120 |
 121 |   // Listen to friends list from Firebase
 122 |   useEffect(() => {
 123 |     if (!user?.uid) {
 124 |       setFriends([]);
 125 |       setFriendsData({});
 126 |       return;
 127 |     }
 128 |
 129 |     const unsubscribeFriends = listenToUserFriends(user.uid, async (friendIds) => {
 130 |       setFriends(friendIds);
 131 |
 132 |       // Fetch friend data for each friend
 133 |       const friendsDataMap: Record<string, any> = {};
 134 |       for (const friendId of friendIds) {
 135 |         try {
 136 |           const friendData = await getUserData(friendId);
 137 |           if (friendData) {
```

```
138 |             friendsDataMap[friendId] = friendData;
139 |           }
140 |         } catch (error) {
141 |           console.error(`Error fetching friend data for ${friendId}:`, error);
142 |         }
143 |       }
144 |       setFriendsData(friendsDataMap);
145 |     });
146 |
147 |     return () => unsubscribeFriends();
148 |   }, [user?.uid]);
149 |
150 |   // Listen to friend requests from Firebase
151 |   useEffect(() => {
152 |     if (!user?.uid) {
153 |       setRequests({ incoming: [], outgoing: [] });
154 |       setRequestsData({});
155 |       return;
156 |     }
157 |
158 |     const unsubscribe = listenToFriendRequests(user.uid, async (requestData) => {
159 |       setRequests(requestData);
160 |
161 |       // Fetch user data for incoming and outgoing requests
162 |       const requestsDataMap: Record<string, any> = {};
163 |       const allRequestIds = [...requestData.incoming, ...requestData.outgoing];
164 |
165 |       for (const requestId of allRequestIds) {
166 |         if (!requestsDataMap[requestId]) {
167 |           try {
168 |             const userData = await getUserData(requestId);
169 |             if (userData) {
170 |               requestsDataMap[requestId] = userData;
171 |             }
172 |           } catch (error) {
173 |             console.error(`Error fetching request user data for ${requestId}:`, error);
174 |           }
175 |         }
176 |       }
177 |       setRequestsData(requestsDataMap);
178 |     });
179 |
180 |     return () => unsubscribe();
181 |   }, [user?.uid]);
182 |
183 |   // Listen to encountered users history and run cleanup
184 |   useEffect(() => {
185 |     if (!user?.uid) {
186 |       setEncounteredUsers({});
187 |       setEncounteredUsersLoading(false);
188 |       return;
189 |     }
190 |
191 |     // Run cleanup on mount
192 |     cleanupOldEncounters(user.uid).catch((error) => {
193 |       console.error("Error cleaning up old encounters:", error);
194 |     });
195 |
196 |     const unsubscribe = listenToEncounteredUsers(user.uid, (encounters) => {
197 |       setEncounteredUsers(encounters);
198 |       setEncounteredUsersLoading(false);
199 |
200 |       // Fetch user data for encountered users
201 |       const fetchUserData = async () => {
202 |         const encounteredIds = Object.keys(encounters);
203 |         const dataMap: Record<string, any> = {};
204 |
205 |         for (const encounteredId of encounteredIds) {
206 |           // Skip if already a friend (we have their data) or currently nearby
207 |           if (friends.includes(encounteredId)) continue;
208 |
```

```
209 |           try {
210 |             const userData = await getUserData(encounteredId);
211 |             if (userData) {
212 |               dataMap[encounteredId] = userData;
213 |             }
214 |           } catch (error) {
215 |             // User might not exist anymore - skip silently
216 |           }
217 |         }
218 |
219 |         setEncounteredUsersData(prev => ({ ...prev, ...dataMap }));
220 |       };
221 |
222 |       fetchUserData();
223 |     });
224 |
225 |     return () => unsubscribe();
226 |   }, [user?.uid, friends]);
227 |
228 |   // Format all nearby users for discover tab (exclude friends)
229 |   // Combines currently nearby users with historical encounters
230 |   const nearbyUsersForDiscover = useMemo(() => {
231 |     const friendIds = new Set(friends);
232 |     const currentlyNearbyIds = new Set(allNearbyUsers.map(u => u.id));
233 |     const combinedUsers: Record<string, any> = {};
234 |
235 |     // Add currently nearby users (prioritized)
236 |     allNearbyUsers
237 |       .filter(user => !friendIds.has(user.id))
238 |       .forEach(user => {
239 |         const username = user.name || null;
240 |         const activity = user.activity || null;
241 |         const displayName = getDisplayName(username, user.id, activity);
242 |
243 |         let distanceKm = user.distance;
244 |         if (!distanceKm && currentLocation && user.lat && user.lng) {
245 |           distanceKm = calculateDistance(
246 |             currentLocation.lat,
247 |             currentLocation.lng,
248 |             user.lat,
249 |             user.lng
250 |           );
251 |         }
252 |
253 |         combinedUsers[user.id] = {
254 |           id: user.id,
255 |           name: displayName,
256 |           distance: formatDistance(distanceKm || 0),
257 |           distanceValue: distanceKm || 0,
258 |           activity: user.activity || "Running",
259 |           avatar: getProfilePictureUrl(user.photoURL, user.avatar, displayName),
260 |           lat: user.lat,
261 |           lng: user.lng,
262 |           fitnessLevel: user.fitnessLevel,
263 |           pace: user.pace,
264 |           bio: user.bio || "",
265 |           photos: user.photos || [],
266 |           isCurrentlyNearby: true,
267 |           lastSeenAt: Date.now()
268 |         };
269 |       });
270 |
271 |     // Add historical encounters (not currently nearby)
272 |     Object.entries(encounteredUsers).forEach(([encounteredUserId, encounter]) => {
273 |       // Skip if already in currently nearby or is a friend
274 |       if (currentlyNearbyIds.has(encounteredUserId) || friendIds.has(encounteredUserId)) {
275 |         return;
276 |       }
277 |
278 |       // Get user data if available
279 |       const userData = encounteredUsersData[encounteredUserId];
```

```
      const username = userData?.name || userData?.username || null;
      const activity = userData?.activity || userData?.activities?.[0] || null;
      const displayName = getDisplayName(username, encounteredUserId, activity);

      if (!combinedUsers[encounteredUserId]) {
        combinedUsers[encounteredUserId] = {
          id: encounteredUserId,
          name: displayName,
          distance: formatDistance(encounter.distance),
          distanceValue: encounter.distance,
          activity: activity || "Unknown",
          avatar: getProfilePictureUrl(userData?.photoURL, userData?.avatar, displayName),
          lat: encounter.lat,
          lng: encounter.lng,
          fitnessLevel: userData?.fitnessLevel,
          pace: userData?.pace,
          bio: userData?.bio || "",
          photos: userData?.photos || [],
          isCurrentlyNearby: false,
          lastSeenAt: encounter.lastSeenAt,
          encounterCount: encounter.count
        };
      }
    });

    // Convert to array and sort: currently nearby first, then by most recent encounter
    return Object.values(combinedUsers)
      .sort((a, b) => {
        // Currently nearby users first
        if (a.isCurrentlyNearby && !b.isCurrentlyNearby) return -1;
        if (!a.isCurrentlyNearby && b.isCurrentlyNearby) return 1;
        // Then sort by most recent encounter
        return (b.lastSeenAt || 0) - (a.lastSeenAt || 0);
      });
  }, [allNearbyUsers, encounteredUsers, encounteredUsersData, friends, currentLocation]);

  const handleSendRequest = async (userId: string, username: string) => {
    if (!user?.uid) return;

    try {
      await sendFriendRequest(user.uid, userId);
      toast.success(`Friend request sent to ${username}!`);
    } catch (error) {
      console.error("Error sending friend request:", error);
      toast.error("Failed to send friend request");
    }
  };

  const handleAcceptRequest = async (userId: string) => {
    if (!user?.uid) return;

    try {
      await acceptFriendRequest(user.uid, userId);
      const userData = requestsData[userId];
      toast.success(`You are now friends with ${userData?.name || userData?.username || "user"}!
`);
    } catch (error) {
      console.error("Error accepting friend request:", error);
      toast.error("Failed to accept friend request");
    }
  };

  const handleDeclineRequest = async (userId: string) => {
    if (!user?.uid) return;

    try {
      await declineFriendRequest(user.uid, userId);
      toast.success("Request declined");
    } catch (error) {
      console.error("Error declining friend request:", error);
      toast.error("Failed to decline friend request");
    }
```

```
351 |   };
352 |
353 |   const handleCancelRequest = async (userId: string) => {
354 |     if (!user?.uid) return;
355 |
356 |     try {
357 |       await cancelFriendRequest(user.uid, userId);
358 |       toast.success("Request cancelled");
359 |     } catch (error) {
360 |       console.error("Error cancelling friend request:", error);
361 |       toast.error("Failed to cancel friend request");
362 |     }
363 |   };
364 |
365 |   const handleUnfriend = (friendId: string, friendName: string) => {
366 |     setUnfriendDialog({ open: true, friendId, friendName });
367 |   };
368 |
369 |   const confirmUnfriend = async () => {
370 |     if (!unfriendDialog.friendId || !user?.uid) return;
371 |
372 |     try {
373 |       await removeFriend(user.uid, unfriendDialog.friendId);
374 |       setUnfriendDialog({ open: false, friendId: null, friendName: "" });
375 |       toast.success(`Unfriended ${unfriendDialog.friendName}`);
376 |     } catch (error) {
377 |       console.error("Error removing friend:", error);
378 |       toast.error("Failed to unfriend user");
379 |     }
380 |   };
381 |
382 |   const handleLocationSharingToggle = (friendId: string, enabled: boolean) => {
383 |     // Update local state
384 |     setLocationSharing(prev => {
385 |       const updated = { ...prev };
386 |       if (enabled) {
387 |         updated[friendId] = true;
388 |       } else {
389 |         delete updated[friendId];
390 |       }
391 |       return updated;
392 |     });
393 |     // Save to storage (using friendId as string, but socialStorage expects number - we'll keep this for now)
394 |     // TODO: Migrate location sharing to Firebase in future
395 |     const friendIdNum = parseInt(friendId);
396 |     if (!isNaN(friendIdNum)) {
397 |       saveLocationSharing(friendIdNum, enabled);
398 |     }
399 |     toast.success(
400 |       enabled
401 |         ? "Friend will see your location during workouts"
402 |         : "Friend will no longer see your location during workouts"
403 |     );
404 |   };
405 |
406 |   const handlePoke = async (userId: string) => {
407 |     if (!user?.uid) {
408 |       toast.error("You must be logged in to poke someone");
409 |       return;
410 |     }
411 |
412 |     // Check if user has active workout session
413 |     if (!isWorkoutActive()) {
414 |       toast.error("You must have an active workout session to poke someone");
415 |       return;
416 |     }
417 |
418 |     try {
419 |       await sendPoke(user.uid, userId);
420 |       setHasPokedUsers(prev => ({ ...prev, [userId]: true }));
421 |       toast.success("Poke sent! They'll be notified.");
```

```
    } catch (error) {
      console.error("Error sending poke:", error);
      toast.error("Failed to send poke. Please try again.");
    }
  };

  const handleSendMessageFromDiscover = (user: any) => {
    navigate("/chat", {
      state: {
        user: {
          id: user.id,
          name: user.name,
          avatar: user.avatar
        }
      }
    });
  };

  const getActivityIcon = (activity: string) => {
    switch (activity.toLowerCase()) {
      case "running":
        return <DirectionsRunIcon className="text-success" style={{ fontSize: 18 }} />;
      case "cycling":
        return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 18 }} />;
      case "walking":
        return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 18 }} />;
      default:
        return null;
    }
  };


  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/5 via-background to-success/5 pb-20">
      {/* Header */}
      <motion.div
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b border-border/50"
      >
        <div className="max-w-2xl mx-auto px-6 py-5">
          <h1 className="text-3xl font-bold">Friends</h1>
          <p className="text-sm text-muted-foreground">
            {friends.length} Friends • {requests.incoming.length} Requests
          </p>
        </div>
      </motion.div>

      <div className="max-w-2xl mx-auto px-6 py-6">
        {/* Search Bar */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1 }}
        >
          <div className="relative mb-6">
            <SearchIcon className="absolute left-4 top-1/2 -translate-y-1/2 text-muted-foreground" />
            <Input
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
              placeholder="Search friends..."
              className="pl-12 h-12"
            />
          </div>
        </motion.div>

        {/* Tabs */}
        <Tabs defaultValue={initialTab} className="w-full">
          <TabsList className="w-full grid grid-cols-3 mb-6">
            <TabsTrigger value="friends">Friends</TabsTrigger>
            <TabsTrigger value="requests">
```

```
              Requests
              {requests.incoming.length > 0 && (
                <span className="ml-2 bg-primary text-primary-foreground rounded-full px-2
py-0.5 text-xs">
                  {requests.incoming.length}
                </span>
              )}
            </TabsTrigger>
            <TabsTrigger value="discover">Discover</TabsTrigger>
          </TabsList>

          {/* Friends Tab */}
          <TabsContent value="friends" className="space-y-3">
            {friends.length === 0 ? (
              <Card className="p-12 text-center">
                <p className="text-muted-foreground">No friends yet</p>
                <p className="text-sm text-muted-foreground mt-2">
                  Start adding friends to see them here!
                </p>
              </Card>
            ) : (
              friends
                .filter(friendId => {
                  const friend = friendsData[friendId];
                  if (!friend) return false;
                  const friendName = friend.name || friend.username || "";
                  return !searchQuery ||
friendName.toLowerCase().includes(searchQuery.toLowerCase());
                })
                .map((friendId, index) => {
                  const friend = friendsData[friendId];
                  if (!friend) return null;

                  const username = friend.name || friend.username || null;
                  const activity = friend.activity || null;
                  const friendName = getDisplayName(username, friendId, activity);
                  const friendAvatar = getProfilePictureUrl(friend.photoURL, friend.avatar,
friendName);
                  const friendBio = friend.bio || "Fitness enthusiast";
                  const friendActivities = friend.activities || (friend.activity ?
[friend.activity] : []);

                  return (
                    <motion.div
                      key={friendId}
                      initial={{ opacity: 0, x: -20 }}
                      animate={{ opacity: 1, x: 0 }}
                      transition={{ delay: index * 0.05 }}
                    >
                      <Card
                        className="p-4 hover:shadow-elevation-2 transition-shadow"
                      >
                        <div className="flex items-center gap-3 mb-3">
                          <Avatar
                            src={friendAvatar}
                            alt={friendName}
                            sx={{ width: 56, height: 56 }}
                            className="cursor-pointer"
                            onClick={() => setSelectedUser(friendId)}
                          />
                          <div
                            className="flex-1 cursor-pointer"
                            onClick={() => setSelectedUser(friendId)}
                          >
                            <h3 className="font-bold">{friendName}</h3>
                            <p className="text-sm text-muted-foreground">{friendBio}</p>
                            {friendActivities.length > 0 && (
                              <div className="flex gap-1 mt-1">
                                {friendActivities.map((activity: string) => (
                                  <span
                                    key={activity}
                                    className="text-xs px-2 py-0.5 rounded-full bg-primary/10
text-primary capitalize"
                                  >
                                    {activity}
                                  </span>
```

```
                              ))}
                            </div>
                          )}
                        </div>
                        <Button
                          size="sm"
                          variant="outline"
                          onClick={(e) => {
                            e.stopPropagation();
                            navigate("/chat", {
                              state: {
                                user: {
                                  id: friendId,
                                  name: friendName,
                                  avatar: friendAvatar
                                }
                              }
                            });
                          }}
                        >
                          <ChatIcon style={{ fontSize: 18 }} className="mr-1" />
                          Message
                        </Button>
                      </div>

                      {/* Location Sharing Toggle */}
                      <div className="flex items-center justify-between pt-3 border-t border-
border/50">
                        <div className="flex items-center gap-2 flex-1">
                          {locationSharing[friendId] ? (
                            <LocationOnIcon className="text-success" style={{ fontSize: 20 }} /
>
                          ) : (
                            <LocationOffIcon className="text-muted-foreground"
style={{ fontSize: 20 }} />
                          )}
                          <div className="flex-1">
                            <p className="text-sm font-medium">
                              Share location during workouts
                            </p>
                            <p className="text-xs text-muted-foreground">
                              {locationSharing[friendId]
                                ? "Friend can see your location when you're active"
                                : "Location only shared during active workouts"}
                            </p>
                          </div>
                        </div>
                        <Switch
                          checked={locationSharing[friendId] || false}
                          onCheckedChange={(checked) => handleLocationSharingToggle(friendId,
checked)}
                          onClick={(e) => e.stopPropagation()}
                        />
                      </div>
                    </Card>
                  </motion.div>
                );
              })
            )}
          </TabsContent>

          {/* Requests Tab */}
          <TabsContent value="requests" className="space-y-6">
            {/* Incoming Requests */}
            <div>
              <h3 className="font-bold mb-3">Incoming Requests</h3>
              {requests.incoming.length === 0 ? (
                <Card className="p-8 text-center">
                  <p className="text-sm text-muted-foreground">No incoming requests</p>
                </Card>
              ) : (
                <div className="space-y-3">
                  {requests.incoming.map(userId => {
                    const user = requestsData[userId];
                    if (!user) return null;
```

```
                    const username = user.name || user.username || null;
                    const activity = user.activity || null;
                    const userName = getDisplayName(username, userId, activity);
                    const userAvatar = user.photoURL || `https://ui-avatars.com/api/?name=${encodeURIComponent(userName)}...
                    const userBio = user.bio || "Fitness enthusiast";

                    return (
                      <Card key={userId} className="p-4">
                        <div className="flex items-center gap-3">
                          <Avatar src={userAvatar} alt={userName} sx={{ width: 56, height: 56 }} />
                          <div className="flex-1">
                            <h3 className="font-bold">{userName}</h3>
                            <p className="text-sm text-muted-foreground">{userBio}</p>
                          </div>
                          <div className="flex gap-2">
                            <Button
                              size="sm"
                              onClick={() => handleAcceptRequest(userId)}
                            >
                              <CheckIcon style={{ fontSize: 18 }} />
                            </Button>
                            <Button
                              size="sm"
                              variant="outline"
                              onClick={() => handleDeclineRequest(userId)}
                            >
                              <CloseIcon style={{ fontSize: 18 }} />
                            </Button>
                          </div>
                        </div>
                      </Card>
                    );
                  })}
                </div>
              )}
            </div>

            {/* Outgoing Requests */}
            <div>
              <h3 className="font-bold mb-3">Outgoing Requests</h3>
              {requests.outgoing.length === 0 ? (
                <Card className="p-8 text-center">
                  <p className="text-sm text-muted-foreground">No outgoing requests</p>
                </Card>
              ) : (
                <div className="space-y-3">
                  {requests.outgoing.map(userId => {
                    const user = requestsData[userId];
                    if (!user) return null;

                    const username = user.name || user.username || null;
                    const activity = user.activity || null;
                    const userName = getDisplayName(username, userId, activity);
                    const userAvatar = user.photoURL || `https://ui-avatars.com/api/?name=${encodeURIComponent(userName)}...
                    return (
                      <Card key={userId} className="p-4">
                        <div className="flex items-center gap-3">
                          <Avatar src={userAvatar} alt={userName} sx={{ width: 56, height: 56 }} />
                          <div className="flex-1">
                            <h3 className="font-bold">{userName}</h3>
                            <p className="text-sm text-muted-foreground">Request pending</p>
                          </div>
                          <Button
                            size="sm"
                            variant="outline"
                            onClick={() => handleCancelRequest(userId)}
                          >
                            Cancel
                          </Button>
```

```
                        </div>
                      </Card>
                    );
                  })}
                </div>
              )}
            </div>
          </TabsContent>

          {/* Discover Tab - All Nearby Users You've Encountered */}
          <TabsContent value="discover" className="space-y-3">
            {(nearbyUsersLoading || encounteredUsersLoading) ? (
              <Card className="p-12 text-center">
                <p className="text-muted-foreground">Loading nearby users...</p>
              </Card>
            ) : nearbyUsersForDiscover.length === 0 ? (
              <Card className="p-12 text-center">
                <p className="text-muted-foreground">No nearby users found</p>
                <p className="text-sm text-muted-foreground mt-2">
                  {currentLocation
                    ? "No other users are currently active nearby. Start a workout to discover people!"
                    : "Enable location sharing to discover nearby users"}
                </p>
              </Card>
            ) : (
              nearbyUsersForDiscover
                .filter(user =>
                  !searchQuery ||
                  user.name.toLowerCase().includes(searchQuery.toLowerCase())
                )
                .map((user, index) => {
                  const isRequestPending = requests.outgoing.includes(user.id);
                  const hasPoked = hasPokedUsers[user.id] || false;
                  const isCurrentlyNearby = user.isCurrentlyNearby || false;

                  // Calculate days since last seen
                  const daysSinceLastSeen = user.lastSeenAt
                    ? Math.floor((Date.now() - user.lastSeenAt) / (1000 * 60 * 60 * 24))
                    : null;

                  return (
                    <motion.div
                      key={user.id}
                      initial={{ opacity: 0, x: -20 }}
                      animate={{ opacity: 1, x: 0 }}
                      transition={{ delay: index * 0.05 }}
                    >
                      <Card
                        className="p-4 cursor-pointer hover:shadow-elevation-2 transition-shadow"
                        onClick={() => setSelectedUser({
                          id: user.id,
                          name: user.name,
                          distance: user.distance,
                          activity: user.activity,
                          avatar: user.avatar,
                          photos: user.photos,
                          bio: user.bio
                        })}
                      >
                        <div className="flex items-center gap-3 mb-3">
                          <div className="relative">
                            <Avatar src={user.avatar} alt={user.name} sx={{ width: 56, height: 56 }} />
                            {isCurrentlyNearby && (
                              <div className="absolute -top-1 -right-1 w-4 h-4 bg-success rounded-full border-2 border-c...
                                <div className="w-2 h-2 bg-white rounded-full animate-pulse" />
                              </div>
                            )}
                          </div>
                          <div className="flex-1">
                            <div className="flex items-center gap-2 mb-1">
                              <h3 className="font-bold">{user.name}</h3>
```

```
                              {isCurrentlyNearby && (
                                <span className="text-xs px-2 py-0.5 rounded-full bg-success/20
text-success font-medium...
                                  Nearby
                                </span>
                              )}
                              <span className="text-xs px-2 py-0.5 rounded-full bg-primary/10
text-primary">
                                {user.distance}
                              </span>
                            </div>
                            <div className="flex items-center gap-2 text-sm text-muted-
foreground">
                              {getActivityIcon(user.activity)}
                              <span className="capitalize">{user.activity}</span>
                              {user.fitnessLevel && (
                                <>
                                  <span>•</span>
                                  <span className="capitalize">{user.fitnessLevel}</span>
                                </>
                              )}
                            </div>
                            {!isCurrentlyNearby && daysSinceLastSeen !== null && (
                              <p className="text-xs text-muted-foreground mt-1">
                                Last seen {daysSinceLastSeen === 0
                                  ? "today"
                                  : daysSinceLastSeen === 1
                                  ? "yesterday"
                                  : `${daysSinceLastSeen} days ago`}
                                {user.encounterCount && user.encounterCount > 1 && (
                                  <span> • {user.encounterCount} encounters</span>
                                )}
                              </p>
                            )}
                          </div>
                        </div>

                        {/* Action Buttons */}
                        <div className="flex gap-2 pt-3 border-t border-border/50">
                          {isWorkoutActive() && (
                            <Button
                              size="sm"
                              variant="outline"
                              className="flex-1 h-9 text-xs"
                              onClick={(e) => {
                                e.stopPropagation();
                                handlePoke(user.id);
                              }}
                              disabled={hasPoked}
                            >
                              <TouchAppIcon style={{ fontSize: 16 }} className="mr-1" />
                              {hasPoked ? "Poked" : "Poke"}
                            </Button>
                          )}

                          {isRequestPending ? (
                            <Button
                              size="sm"
                              variant="outline"
                              disabled
                              className="flex-1 h-9 text-xs"
                            >
                              <PersonAddIcon style={{ fontSize: 16 }} className="mr-1" />
                              Pending
                            </Button>
                          ) : (
                            <Button
                              size="sm"
                              variant="outline"
                              className="flex-1 h-9 text-xs"
                              onClick={(e) => {
                                e.stopPropagation();
                                handleSendRequest(user.id, user.name);
                              }}
```

```
                            >
                              <PersonAddIcon style={{ fontSize: 16 }} className="mr-1" />
                              Add
                            </Button>
                          )}

                          <Button
                            size="sm"
                            variant="outline"
                            className="flex-1 h-9 text-xs"
                            onClick={(e) => {
                              e.stopPropagation();
                              handleSendMessageFromDiscover(user);
                            }}
                          >
                            <SendIcon style={{ fontSize: 16 }} className="mr-1" />
                            Message
                          </Button>
                        </div>
                      </Card>
                    </motion.div>
                  );
                })
            )}
          </TabsContent>
        </Tabs>
      </div>

      <BottomNavigation />

      {/* Profile View Modal */}
      <AnimatePresence>
        {selectedUser && (() => {
          // Check if selectedUser is from discover (has id as string) or friends list (string)
          const isDiscoverUser = typeof selectedUser === 'object' && selectedUser.id;
          const isFriendId = typeof selectedUser === 'string';

          let userData: any = null;
          let friendStatus: "not_friends" | "request_pending" | "request_received" | "friends" |
"denied" = "not_friends...
          if (isDiscoverUser) {
            // User from discover tab
            userData = selectedUser;
            if (requests.outgoing.includes(userData.id)) {
              friendStatus = "request_pending";
            } else if (friends.includes(userData.id)) {
              friendStatus = "friends";
            } else if (requests.incoming.includes(userData.id)) {
              friendStatus = "request_received";
            } else {
              friendStatus = "not_friends";
            }
          } else if (isFriendId) {
            // User from friends list
            const friend = friendsData[selectedUser];
            if (!friend) return null;

            const username = friend.name || friend.username || null;
            const activity = friend.activity || null;
            const friendName = getDisplayName(username, selectedUser, activity);
            const friendAvatar = friend.photoURL || `https://ui-avatars.com/api/?
name=${encodeURIComponent(friendName)}&...
            const friendActivities = friend.activities || (friend.activity ? [friend.activity] :
[]);

            userData = {
              id: selectedUser,
              name: friendName,
              distance: friend.lat && friend.lng && currentLocation?.lat && currentLocation?.lng
                ? formatDistance(calculateDistance(
                    currentLocation.lat,
                    currentLocation.lng,
                    friend.lat,
```

```
                    friend.lng
                  ))
                : "Unknown",
              activity: friendActivities.length > 0
                ? friendActivities[0].charAt(0).toUpperCase() + friendActivities[0].slice(1)
                : (friend.activity ? friend.activity.charAt(0).toUpperCase() +
friend.activity.slice(1) : "Running"),
              avatar: friendAvatar,
              photos: friend.photos || [friendAvatar],
              bio: friend.bio,
            };
            friendStatus = "friends";
          }

          if (!userData) return null;

          return (
            <ProfileView
              user={userData}
              friendStatus={friendStatus}
              onClose={() => setSelectedUser(null)}
              onSendMessage={() => {
                if (isDiscoverUser) {
                  handleSendMessageFromDiscover(userData);
                } else {
                  navigate("/chat", {
                    state: {
                      user: {
                        id: userData.id,
                        name: userData.name,
                        avatar: userData.avatar
                      }
                    });
                  }
                  setSelectedUser(null);
              }}
              onAddFriend={() => {
                if (isDiscoverUser) {
                  handleSendRequest(userData.id, userData.name);
                }
              }}
              onAcceptFriend={() => {
                if (isDiscoverUser && requests.incoming.includes(userData.id)) {
                  handleAcceptRequest(userData.id);
                }
              }}
              onDeclineFriend={() => {
                if (isDiscoverUser && requests.incoming.includes(userData.id)) {
                  handleDeclineRequest(userData.id);
                }
              }}
              onUnfriend={() => {
                if (isFriendId) {
                  const friend = friendsData[selectedUser];
                  const username = friend?.name || friend?.username || null;
                  const activity = friend?.activity || null;
                  const friendName = getDisplayName(username, selectedUser, activity);
                  handleUnfriend(selectedUser, friendName);
                }
              }}
              onPoke={isWorkoutActive() ? () => {
                if (isDiscoverUser) {
                  handlePoke(userData.id);
                }
              } : undefined}
              hasPoked={isDiscoverUser ? (hasPokedUsers[userData.id] || false) : false}
              isWorkoutActive={isWorkoutActive()}
            />
          );
        })()}
      </AnimatePresence>
```

```
      {/* Unfriend Confirmation Dialog */}
      <AlertDialog open={unfriendDialog.open} onOpenChange={(open) =>
        setUnfriendDialog(prev => ({ ...prev, open }))
      }>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Unfriend {unfriendDialog.friendName}?</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to unfriend {unfriendDialog.friendName}?
              They will be removed from your friends list and won't be able to see your location during workouts.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel>Cancel</AlertDialogCancel>
            <AlertDialogAction
              onClick={confirmUnfriend}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Unfriend
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>
    </div>
  );
};

export default Friends;
```

File: src/pages/Index.tsx

Lines: 1971

```
import { useState, useEffect, useMemo } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { useNavigate } from "react-router-dom";
import { useUser, FitnessLevel } from "@/contexts/UserContext";
import { useAuth } from "@/hooks/useAuth";
import { useLocation } from "@/hooks/useLocation";
import { useMatching } from "@/hooks/useMatching";
import { listenToWorkoutPosts, WorkoutPost as FirebaseWorkoutPost } from "@/services/feedService";
import { listenToUserFriends, listenToFriendRequests, sendFriendRequest, acceptFriendRequest, declineFriendRequest, removeFriend } from "@/services/friendService";
import { listenToAllUsers, updateUserVisibility } from "@/services/locationService";
import { getUserData, updateUserProfile } from "@/services/authService";
import { reportUser, blockUser } from "@/services/userService";
import { generateDummyWorkoutPosts, ENABLE_DUMMY_DATA } from "@/lib/dummyData";
import { formatDistance, calculateDistance } from "@/utils/distance";
import { WorkoutPost } from "@/components/WorkoutPost";
import { CommentDrawer } from "@/components/CommentDrawer";
import { WorkoutHistoryFeed } from "@/components/WorkoutHistoryFeed";
import { ProfileView } from "@/pages/ProfileView";
import { WorkoutWithUser } from "@/services/workoutService";
import { Button } from "@/components/ui/button";
import { Card } from "@/components/ui/card";
import { Tabs, TabsList, TabsTrigger, TabsContent } from "@/components/ui/tabs";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { Input } from "@/components/ui/input";
import HomeIcon from "@mui/icons-material/Home";
import MapIcon from "@mui/icons-material/Map";
import EventIcon from "@mui/icons-material/Event";
import ChatIcon from "@mui/icons-material/Chat";
import PersonIcon from "@mui/icons-material/Person";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import LocationOffIcon from "@mui/icons-material/LocationOff";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import NotificationsIcon from "@mui/icons-material/Notifications";
import CloseIcon from "@mui/icons-material/Close";
import MailIcon from "@mui/icons-material/Mail";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import EmojiEventsIcon from "@mui/icons-material/EmojiEvents";
import ChatBubbleIcon from "@mui/icons-material/ChatBubble";
import EditIcon from "@mui/icons-material/Edit";
import BottomNavigation from "@/components/BottomNavigation";
import { NotificationBell } from "@/components/NotificationBell";
import { Switch } from "@/components/ui/switch";
import { Avatar } from "@mui/material";
import { toast } from "sonner";
import { useNotificationContext } from "@/contexts/NotificationContext";
import { QuickCheckInModal } from "@/components/QuickCheckInModal";
import { listenToUserVenuePreferences, listenToUsersByVenues, VenueUser } from "@/services/venuePreferenceService";
import { getAllVenues, getVenueById } from "@/services/venueService";
import { getProfilePictureUrl } from "@/utils/profilePicture";
import { WeatherWidget } from "@/components/WeatherWidget";
import { getLocationForWeather } from "@/utils/getLocationForWeather";

type FriendStatus = "not_friends" | "request_pending" | "request_received" | "friends" | "denied";

const Index = () => {
  const navigate = useNavigate();
  const { userProfile, workoutHistory, useMetric } = useUser();
  const { user: currentUser } = useAuth();
  const [selectedTab, setSelectedTab] = useState<string>("all");
  const [activeFriendsTab, setActiveFriendsTab] = useState<"active" | "history">("history");
  const [timeframe, setTimeframe] = useState<"week" | "month" | "all">("all");
```

```
  67 |   const [selectedPost, setSelectedPost] = useState<FirebaseWorkoutPost | null>(null);
  68 |   const [isCommentDrawerOpen, setIsCommentDrawerOpen] = useState(false);
  69 |   const [workoutPosts, setWorkoutPosts] = useState<FirebaseWorkoutPost[]>([]);
  70 |   const [loading, setLoading] = useState(true);
  71 |   const [friendsOnline, setFriendsOnline] = useState(0);
  72 |   const [friendsList, setFriendsList] = useState<string[]>([]);
  73 |   const [friendsData, setFriendsData] = useState<Record<string, any>>({});
  74 |   const [userVisibility, setUserVisibility] = useState(true);
  75 |   const [selectedWorkoutHistoryItem, setSelectedWorkoutHistoryItem] = useState<WorkoutWithUser |
null>(null);
  76 |   const [selectedActiveFriend, setSelectedActiveFriend] = useState<{ id: string; name?: string;
username?: string; photo
>(fr incoming: string[]; outgo
  78 |   const [friendStatuses, setFriendStatuses] = useState<Record<string, { status: FriendStatus;
cooldownUntil?: number }>>(
  79 |     // Dummy data flag -- controlled by environment variable for production
  80 |   // Set VITE_ENABLE_DUMMY_DATA=false in production
  81 |   const [showDummyData, setShowDummyData] = useState(ENABLE_DUMMY_DATA);
  82 |   const [showNotificationDrawer, setShowNotificationDrawer] = useState(false);
  83 |   const [profileVisible, setProfileVisible] = useState(false);
  84 |   const [generalLocation, setGeneralLocation] = useState<string>("");
  85 |   const [isEditingLocation, setIsEditingLocation] = useState(false);
  86 |   const [showQuickCheckIn, setShowQuickCheckIn] = useState(false);
  87 |   const [usersByVenue, setUsersByVenue] = useState<Record<string, VenueUser[]>>({});
  88 |   const [userVenuePreferences, setUserVenuePreferences] = useState<{ venues: string[];
activities: string[] } | null>(nu
  89 |     const [username, setUsername] = useState<string>("");
  90 |   const [activeFriendData, setActiveFriendData] = useState<any>(null);
  91 |   const [loadingActiveFriendData, setLoadingActiveFriendData] = useState(false);
  92 |
  93 |   // Notification context
  94 |   const { unreadCount, notifications, dismissNotification, markAllAsRead,
handleNotificationTap } = useNotificationConte...
  96 |   // Get user location for matching
  97 |   const { location: trackingLocation } = useLocation(currentUser?.uid || null, false, true);
  98 |
  99 |   // Get location for weather widget (one-time request on sign-in)
 100 |   const [weatherLocation, setWeatherLocation] = useState<{ lat: number; lng: number } |
null>(null);
 101 |
 102 |   // Request location permission and get location for weather when user signs in
 103 |   useEffect(() => {
 104 |     if (!currentUser?.uid) {
 105 |       setWeatherLocation(null);
 106 |       return;
 107 |     }
 108 |
 109 |     const fetchWeatherLocation = async () => {
 110 |       // Try to get location for weather widget
 111 |       const loc = await getLocationForWeather();
 112 |       if (loc) {
 113 |         setWeatherLocation(loc);
 114 |       }
 115 |     };
 116 |
 117 |     // Request location after a short delay to ensure user is fully authenticated
 118 |     const timer = setTimeout(() => {
 119 |       fetchWeatherLocation();
 120 |     }, 500);
 121 |
 122 |     return () => clearTimeout(timer);
 123 |   }, [currentUser?.uid]);
 124 |
 125 |   // Use weather location if available, otherwise fall back to tracking location
 126 |   const location = weatherLocation || trackingLocation;
 127 |
 128 |   // Memoize visibility object to prevent infinite loops
 129 |   const defaultVisibility = useMemo(() => ({
 130 |     visibleToAllLevels: true,
 131 |     allowedLevels: ["beginner", "intermediate", "pro"] as FitnessLevel[]
 132 |   }), []);
 133 |
 134 |   const visibility = useMemo(() =>
 135 |     userProfile?.visibility || defaultVisibility,
 136 |     [userProfile?.visibility, defaultVisibility]
 137 |   );
```

```
138 |
139 |   // Get top matches
140 |   const { matches: topMatches } = useMatching({
141 |     currentUserId: currentUser?.uid || "",
142 |     currentLocation: location,
143 |     activity: userProfile?.activities?.[0] || "running",
144 |     fitnessLevel: userProfile?.fitnessLevel || "intermediate",
145 |     pace: userProfile?.pace,
146 |     visibility,
147 |     radiusPreference: userProfile?.radiusPreference || "normal"
148 |   });
149 |
150 |   // Listen to workout posts from Firebase
151 |   useEffect(() => {
152 |     const unsubscribe = listenToWorkoutPosts((posts) => {
153 |       // Add dummy posts if enabled and no real posts exist
154 |       if (ENABLE_DUMMY_DATA && posts.length === 0) {
155 |         const dummyPosts = generateDummyWorkoutPosts();
156 |         setWorkoutPosts(dummyPosts);
157 |       } else {
158 |         setWorkoutPosts(posts);
159 |       }
160 |       setLoading(false);
161 |     });
162 |
163 |     return () => unsubscribe();
164 |   }, []);
165 |
166 |   // Update selectedPost when posts update (for real-time comment/kudos updates)
167 |   useEffect(() => {
168 |     if (selectedPost) {
169 |       const updatedPost = workoutPosts.find(p => p.id === selectedPost.id);
170 |       if (updatedPost) {
171 |         setSelectedPost(updatedPost);
172 |       }
173 |     }
174 |   }, [workoutPosts, selectedPost?.id]);
175 |
176 |   // Get current user's visibility status and profile discovery settings
177 |   useEffect(() => {
178 |     if (!currentUser?.uid) return;
179 |
180 |     const fetchUserSettings = async () => {
181 |       try {
182 |         const userData = await getUserData(currentUser.uid);
183 |         if (userData) {
184 |           setUserVisibility(userData.visible !== false);
185 |           setProfileVisible(userData.profileVisible !== false);
186 |           setGeneralLocation(userData.generalLocation || "");
187 |           // Load username from Firebase (stored as 'name' field)
188 |           if (userData.name) {
189 |             setUsername(userData.name);
190 |           }
191 |         }
192 |       } catch (error) {
193 |         console.error("Error fetching user settings:", error);
194 |       }
195 |     };
196 |
197 |     fetchUserSettings();
198 |   }, [currentUser?.uid]);
199 |
200 |   // Calculate friends online and fetch friends data from Firebase
201 |   useEffect(() => {
202 |     if (!currentUser?.uid) {
203 |       setFriendsOnline(0);
204 |       setFriendsList([]);
205 |       setFriendsData({});
206 |       return;
207 |     }
208 |
```

```
209 |     let friendsListLocal: string[] = [];
210 |     let allUsers: Record<string, any> = {};
211 |
212 |     // Listen to friends list
213 |     const unsubscribeFriends = listenToUserFriends(currentUser.uid, async (friends) => {
214 |       friendsListLocal = friends;
215 |       setFriendsList(friends);
216 |
217 |       // Fetch friend data for each friend
218 |       const friendsDataMap: Record<string, any> = {};
219 |       for (const friendId of friends) {
220 |         try {
221 |           const friendData = await getUserData(friendId);
222 |           if (friendData) {
223 |             friendsDataMap[friendId] = friendData;
224 |           }
225 |         } catch (error) {
226 |           console.error(`Error fetching friend data for ${friendId}:`, error);
227 |         }
228 |       }
229 |       setFriendsData(friendsDataMap);
230 |
231 |       calculateFriendsOnline(friends, allUsers);
232 |     });
233 |
234 |     // Listen to all users to check who's online
235 |     const unsubscribeUsers = listenToAllUsers((users) => {
236 |       allUsers = users;
237 |       calculateFriendsOnline(friendsListLocal, users);
238 |
239 |       // Update friends data with latest location info
240 |       setFriendsData((prev) => {
241 |         const updated = { ...prev };
242 |         friendsListLocal.forEach((friendId) => {
243 |           if (users[friendId]) {
244 |             updated[friendId] = { ...updated[friendId], ...users[friendId] };
245 |           }
246 |         });
247 |         return updated;
248 |       });
249 |     });
250 |
251 |     const calculateFriendsOnline = (friends: string[], users: Record<string, any>) => {
252 |       if (friends.length === 0) {
253 |         setFriendsOnline(0);
254 |         return;
255 |       }
256 |
257 |       const now = Date.now();
258 |       const onlineThreshold = 10 * 60 * 1000; // 10 minutes in milliseconds
259 |
260 |       const onlineCount = friends.filter((friendId) => {
261 |         const user = users[friendId];
262 |         if (!user || !user.visible) return false;
263 |
264 |         // Check if user has recent location update (within last 10 minutes)
265 |         if (user.timestamp) {
266 |           const timeDiff = now - user.timestamp;
267 |           return timeDiff <= onlineThreshold;
268 |         }
269 |
270 |         return false;
271 |       }).length;
272 |
273 |       setFriendsOnline(onlineCount);
274 |     };
275 |
276 |     return () => {
277 |       unsubscribeFriends();
278 |       unsubscribeUsers();
279 |     };
```

```
280 |   }, [currentUser?.uid]);
281 |
282 |   // Listen to friend requests
283 |   useEffect(() => {
284 |     if (!currentUser?.uid) return;
285 |
286 |     const unsubscribe = listenToFriendRequests(currentUser.uid, (requests) => {
287 |       setFriendRequests(requests);
288 |     });
289 |
290 |     return () => unsubscribe();
291 |   }, [currentUser?.uid]);
292 |
293 |   // Mark all notifications as read when notification drawer opens
294 |   useEffect(() => {
295 |     if (showNotificationDrawer && unreadCount > 0) {
296 |       markAllAsRead();
297 |     }
298 |   }, [showNotificationDrawer, unreadCount, markAllAsRead]);
299 |
300 |   // Load user venue preferences in real-time so Meet New People updates without refresh
301 |   useEffect(() => {
302 |     if (!currentUser?.uid) {
303 |       setUsersByVenue({});
304 |       setUserVenuePreferences(null);
305 |       return;
306 |     }
307 |
308 |     const unsubscribe = listenToUserVenuePreferences(currentUser.uid, (preferences) => {
309 |       if (preferences && preferences.venues && preferences.venues.length > 0) {
310 |         setUserVenuePreferences({
311 |           venues: preferences.venues,
312 |           activities: preferences.activities
313 |         });
314 |       } else {
315 |         setUserVenuePreferences(null);
316 |         setUsersByVenue({});
317 |       }
318 |     });
319 |
320 |     return () => {
321 |       if (unsubscribe) {
322 |         unsubscribe();
323 |       }
324 |     };
325 |   }, [currentUser?.uid]);
326 |
327 |   // Fetch active friend data when selected
328 |   useEffect(() => {
329 |     if (!selectedActiveFriend?.id) {
330 |       setActiveFriendData(null);
331 |       return;
332 |     }
333 |
334 |     const fetchFriendData = async () => {
335 |       try {
336 |         setLoadingActiveFriendData(true);
337 |         const data = await getUserData(selectedActiveFriend.id);
338 |         setActiveFriendData(data);
339 |       } catch (error) {
340 |         console.error("Error fetching active friend data:", error);
341 |         toast.error("Failed to load friend profile");
342 |         setActiveFriendData(null);
343 |       } finally {
344 |         setLoadingActiveFriendData(false);
345 |       }
346 |     };
347 |
348 |     fetchFriendData();
349 |   }, [selectedActiveFriend?.id]);
350 |
```

```
  // Listen to users by venues when preferences are set
  useEffect(() => {
    if (!currentUser?.uid || !userVenuePreferences || userVenuePreferences.venues.length === 0) {
      setUsersByVenue({});
      return;
    }

    console.log(`[Index] Setting up listener for venues:`, userVenuePreferences.venues);

    const unsubscribe = listenToUsersByVenues(
      userVenuePreferences.venues,
      (usersByVenueData) => {
        // Force console log - this should always show
        console.log(`[Index] ===== CALLBACK FIRED =====`);
        console.log(`[Index] Received venue users data:`, usersByVenueData);
        console.log(`[Index] Current user ID:`, currentUser?.uid);
        console.log(`[Index] Venue IDs we're listening to:`, userVenuePreferences.venues);
        console.log(`[Index] Data type:`, typeof usersByVenueData);
        console.log(`[Index] Data keys:`, Object.keys(usersByVenueData || {}));

        // Log each venue's user count
        userVenuePreferences.venues.forEach((venueId) => {
          const users = usersByVenueData[venueId] || [];
          console.log(`[Index] Venue ${venueId} has ${users.length} users:`, users);
          if (users.length > 0) {
            console.log(`[Index] First user in ${venueId}:`, users[0]);
          }
        });

        // Keep all users including current user
        setUsersByVenue(usersByVenueData);
      }
    );

    return () => {
      if (unsubscribe) unsubscribe();
    };
  }, [currentUser?.uid, userVenuePreferences]);

  // Get friend status for a user
  const getFriendStatus = (userId: string | number): FriendStatus => {
    const id = typeof userId === "number" ? userId.toString() : userId;

    // Check if already friends
    if (friendsList.includes(id)) {
      return "friends";
    }

    // Check if request pending (outgoing)
    if (friendRequests.outgoing.includes(id)) {
      return "request_pending";
    }

    // Check if request received (incoming)
    if (friendRequests.incoming.includes(id)) {
      return "request_received";
    }

    // Check local state
    const localStatus = friendStatuses[id];
    if (localStatus) {
      return localStatus.status;
    }

    return "not_friends";
  };

  // Handle workout history item click
  const handleWorkoutHistoryClick = (item: WorkoutWithUser) => {
    setSelectedWorkoutHistoryItem(item);
  };
```

```
422 |
423 |   // Handle profile view actions
424 |   const handleAddFriend = async (userId: string) => {
425 |     if (!currentUser?.uid) return;
426 |
427 |     try {
428 |       await sendFriendRequest(currentUser.uid, userId);
429 |       setFriendStatuses((prev) => ({
430 |         ...prev,
431 |         [userId]: { status: "request_pending" },
432 |       }));
433 |       toast.success("Friend request sent!");
434 |     } catch (error) {
435 |       console.error("Error sending friend request:", error);
436 |       toast.error("Failed to send friend request");
437 |     }
438 |   };
439 |
440 |   const handleAcceptFriend = async (userId: string) => {
441 |     if (!currentUser?.uid) return;
442 |
443 |     try {
444 |       await acceptFriendRequest(currentUser.uid, userId);
445 |       setFriendStatuses((prev) => ({
446 |         ...prev,
447 |         [userId]: { status: "friends" },
448 |       }));
449 |       toast.success("Friend request accepted!");
450 |     } catch (error) {
451 |       console.error("Error accepting friend request:", error);
452 |       toast.error("Failed to accept friend request");
453 |     }
454 |   };
455 |
456 |   const handleDeclineFriend = async (userId: string) => {
457 |     if (!currentUser?.uid) return;
458 |
459 |     try {
460 |       await declineFriendRequest(currentUser.uid, userId);
461 |       setFriendStatuses((prev) => {
462 |         const updated = { ...prev };
463 |         delete updated[userId];
464 |         return updated;
465 |       });
466 |       toast.success("Friend request declined");
467 |     } catch (error) {
468 |       console.error("Error declining friend request:", error);
469 |       toast.error("Failed to decline friend request");
470 |     }
471 |   };
472 |
473 |   const handleUnfriend = async (userId: string) => {
474 |     if (!currentUser?.uid) return;
475 |
476 |     try {
477 |       await removeFriend(currentUser.uid, userId);
478 |       setFriendStatuses((prev) => {
479 |         const updated = { ...prev };
480 |         delete updated[userId];
481 |         return updated;
482 |       });
483 |       toast.success("Friend removed");
484 |       setSelectedWorkoutHistoryItem(null);
485 |     } catch (error) {
486 |       console.error("Error removing friend:", error);
487 |       toast.error("Failed to remove friend");
488 |     }
489 |   };
490 |
491 |   const handleReportUser = async (userId: string, reason: string, details?: string) => {
492 |     if (!currentUser?.uid) {
```

```
493 |       toast.error("Unable to report user");
494 |       return;
495 |     }
496 |
497 |     try {
498 |       await reportUser(currentUser.uid, userId, reason, details);
499 |       toast.success("User reported. Thank you for helping keep PaceMatch safe.");
500 |       setSelectedActiveFriend(null);
501 |       setActiveFriendData(null);
502 |       setSelectedWorkoutHistoryItem(null);
503 |     } catch (error: any) {
504 |       console.error("Error reporting user:", error);
505 |       toast.error(error.message || "Failed to report user. Please try again.");
506 |     }
507 |   };
508 |
509 |   const handleBlockUser = async (userId: string) => {
510 |     if (!currentUser?.uid) {
511 |       toast.error("Unable to block user");
512 |       return;
513 |     }
514 |
515 |     try {
516 |       await blockUser(currentUser.uid, userId);
517 |       toast.success("User has been blocked");
518 |       setSelectedActiveFriend(null);
519 |       setActiveFriendData(null);
520 |       setSelectedWorkoutHistoryItem(null);
521 |       // Refresh to remove blocked user from view
522 |       setTimeout(() => {
523 |         window.location.reload();
524 |       }, 1000);
525 |     } catch (error: any) {
526 |       console.error("Error blocking user:", error);
527 |       toast.error(error.message || "Failed to block user. Please try again.");
528 |     }
529 |   };
530 |
531 |   // Use Firebase workout posts
532 |   const allPosts = workoutPosts;
533 |
534 |   // Filter posts based on selected activity
535 |   const filteredPosts = selectedTab === "all"
536 |     ? allPosts
537 |     : allPosts.filter(post => post.workout.activity === selectedTab);
538 |
539 |   // Calculate quick stats
540 |   const thisWeekWorkouts = workoutHistory.filter(w => {
541 |     const weekAgo = new Date();
542 |     weekAgo.setDate(weekAgo.getDate() - 7);
543 |     return w.date >= weekAgo;
544 |   }).length;
545 |
546 |   const totalDistance = workoutHistory.reduce((sum, w) => sum + w.distance, 0);
547 |
548 |   const handleCommentClick = (post: FirebaseWorkoutPost) => {
549 |     setSelectedPost(post);
550 |     setIsCommentDrawerOpen(true);
551 |   };
552 |
553 |   const handleVisibilityToggle = async (checked: boolean) => {
554 |     if (!currentUser?.uid) return;
555 |
556 |     try {
557 |       // Only update visibility status, do NOT share exact GPS coordinates
558 |       await updateUserVisibility(currentUser.uid, checked);
559 |       setUserVisibility(checked);
560 |
561 |       toast.success(checked ? "Active status enabled" : "Active status disabled");
562 |     } catch (error) {
563 |       console.error("Error updating visibility:", error);
```

```
564 |       toast.error("Failed to update active status");
565 |     }
566 |   };
567 |
568 |   const handleLocationSave = async () => {
569 |     if (!currentUser?.uid) return;
570 |
571 |     try {
572 |       await updateUserProfile(currentUser.uid, { generalLocation: generalLocation.trim() ||
null });
573 |       setIsEditingLocation(false);
574 |       toast.success("Location updated successfully");
575 |     } catch (error) {
576 |       console.error("Error updating location:", error);
577 |       toast.error("Failed to update location");
578 |     }
579 |   };
580 |
581 |   /**
582 |    * Opens Google Maps with venue location
583 |    * Validates coordinates and handles cross-platform support (iOS, Android, Web)
584 |    */
585 |   const openVenueInMaps = (venue: { lat: number; lng: number; name: string } | null) => {
586 |     if (!venue) {
587 |       toast.error("Venue location not available");
588 |       return;
589 |     }
590 |
591 |     // Validate coordinates
592 |     const { lat, lng } = venue;
593 |     if (typeof lat !== 'number' || typeof lng !== 'number' || isNaN(lat) || isNaN(lng)) {
594 |       toast.error("Invalid venue coordinates");
595 |       return;
596 |     }
597 |
598 |     // Validate coordinate ranges
599 |     if (lat < -90 || lat > 90 || lng < -180 || lng > 180) {
600 |       toast.error("Venue coordinates are out of valid range");
601 |       return;
602 |     }
603 |
604 |     // Universal Google Maps URL that works on iOS, Android, and Web
605 |     // This URL will open in native Google Maps app if installed, otherwise in browser
606 |     const mapsUrl = `https://www.google.com/maps/search/?api=1&query=${lat},${lng}`;
607 |
608 |     try {
609 |       window.open(mapsUrl, '_blank', 'noopener,noreferrer');
610 |     } catch (error) {
611 |       console.error("Error opening maps:", error);
612 |       toast.error("Failed to open maps. Please try again.");
613 |     }
614 |   };
615 |
616 |   // Dummy data for preview (remove in production)
617 |   const getDummyActiveFriends = () => {
618 |     if (!showDummyData || !location?.lat || !location?.lng) return [];
619 |
620 |     const now = Date.now();
621 |     const dummyFriends = [
622 |       {
623 |         id: "dummy-friend-1",
624 |         name: "Sarah Johnson",
625 |         username: "sarah_runs",
626 |         photoURL: "https://i.pravatar.cc/150?img=47",
627 |         activity: "running",
628 |         lat: location.lat + 0.01, // ~1.1km away
629 |         lng: location.lng + 0.01,
630 |         visible: true,
631 |         timestamp: now - 2 * 60 * 1000, // 2 minutes ago
632 |         fitnessLevel: "intermediate",
633 |         pace: 5.2,
634 |       },
```

```
635 |       {
636 |         id: "dummy-friend-2",
637 |         name: "Mike Chen",
638 |         username: "mike_cycles",
639 |         photoURL: "https://i.pravatar.cc/150?img=33",
640 |         activity: "cycling",
641 |         lat: location.lat - 0.015, // ~1.7km away
642 |         lng: location.lng + 0.008,
643 |         visible: true,
644 |         timestamp: now - 5 * 60 * 1000, // 5 minutes ago
645 |         fitnessLevel: "pro",
646 |         pace: 25.5,
647 |       },
648 |       {
649 |         id: "dummy-friend-3",
650 |         name: "Emma Wilson",
651 |         username: "emma_walks",
652 |         photoURL: "https://i.pravatar.cc/150?img=20",
653 |         activity: "walking",
654 |         lat: location.lat + 0.008, // ~0.9km away
655 |         lng: location.lng - 0.012,
656 |         visible: true,
657 |         timestamp: now - 1 * 60 * 1000, // 1 minute ago
658 |         fitnessLevel: "beginner",
659 |         pace: 7.8,
660 |       },
661 |     ];
662 |
663 |     return dummyFriends.map((friend) => {
664 |       const distanceKm = calculateDistance(
665 |         location.lat,
666 |         location.lng,
667 |         friend.lat,
668 |         friend.lng
669 |       );
670 |
671 |       return {
672 |         ...friend,
673 |         distanceKm,
674 |       };
675 |     }).sort((a, b) => (b.timestamp || 0) - (a.timestamp || 0));
676 |   };
677 |
678 |   // Get active friends (currently on a workout - recent location update)
679 |   const getActiveFriends = () => {
680 |     const now = Date.now();
681 |     const activeThreshold = 10 * 60 * 1000; // 10 minutes
682 |
683 |     const realFriends = friendsList
684 |       .filter((friendId) => {
685 |         const friend = friendsData[friendId];
686 |         if (!friend || !friend.visible) return false;
687 |
688 |         // Check if friend has recent location update
689 |         if (friend.timestamp) {
690 |           const timeDiff = now - friend.timestamp;
691 |           return timeDiff <= activeThreshold;
692 |         }
693 |         return false;
694 |       })
695 |       .map((friendId) => {
696 |         const friend = friendsData[friendId];
697 |
698 |         // Do NOT calculate distance - we're showing active status only, not exact location
699 |         // Return friend data without distance information
700 |         return {
701 |           id: friendId,
702 |           ...friend,
703 |           distanceKm: null, // Explicitly set to null to ensure no distance is shown
704 |         };
705 |       })
```

```
      .sort((a, b) => (b.timestamp || 0) - (a.timestamp || 0)); // Most recent first

    // If no real friends and dummy data is enabled, show dummy data
    if (realFriends.length === 0 && showDummyData) {
      return getDummyActiveFriends();
    }

    return realFriends;
  };

  const activeFriends = getActiveFriends();

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/5 via-background to-success/5 pb-20">
      {/* Header */}
      <motion.div
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b border-border/50"
        style={{
          paddingTop: 'env(safe-area-inset-top)',
        }}
      >
        <div className="max-w-2xl mx-auto px-6 py-5">
          <div className="flex items-start justify-between">
            <div>
              <h1 className="text-3xl font-bold mb-1">Welcome back, {userProfile?.username || username || currentUser?.d</h1>
              <p>isplayName || currentUser?.email || 'Athlete'}!</p>
              <p className="text-sm text-muted-foreground">Stay connected with your fitness community</p>
            </div>
            {/* Notification Bell - Yellow when there are notifications */}
            <NotificationBell
              unreadCount={unreadCount}
              onClick={() => setShowNotificationDrawer(true)}
              variant="light"
            />
          </div>
        </div>
      </motion.div>

      {/* Weather Widget */}
      {location && (
        <div className="max-w-4xl mx-auto px-6">
          <WeatherWidget location={location} useMetric={useMetric} />
        </div>
      )}

      <div className="max-w-4xl mx-auto px-6 py-6 space-y-6">
        {/* Quick Stats */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1 }}
        >
          <Card className="p-6 bg-gradient-to-r from-primary/10 to-success/10">
            <div className="grid grid-cols-3 gap-4">
              <div className="text-center">
                <div className="text-3xl font-bold text-primary">{thisWeekWorkouts}</div>
                <div className="text-xs text-muted-foreground mt-1">Workouts this week</div>
              </div>
              <div className="text-center">
                <div className="text-3xl font-bold text-success">
                  {useMetric ? totalDistance.toFixed(0) : (totalDistance * 0.621371).toFixed(0)}
                </div>
                <div className="text-xs text-muted-foreground mt-1">
                  Total {useMetric ? "km" : "mi"}
                </div>
              </div>
              <div className="text-center">
                <div className="text-3xl font-bold text-warning">{friendsOnline}</div>
                <div className="text-xs text-muted-foreground mt-1">Friends online</div>
              </div>
```

```
            </div>
          </Card>
        </motion.div>

        {/* Conditional rendering: Show Friends first if user has venues, otherwise show Meet New People first */}
        {userVenuePreferences && userVenuePreferences.venues.length > 0 ? (
          <>
            {/* Friends Section with Tabs - Show first if user has venues */}
            <motion.div
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ delay: 0.11 }}
            >
            <Card className="p-6">
              <Tabs value={activeFriendsTab} onValueChange={(value) => setActiveFriendsTab(value as "active" | "history")}>
                <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-3 mb-4">
                  <TabsList className="grid grid-cols-2 w-full sm:w-auto flex-shrink-0">
                    <TabsTrigger value="history" className="text-xs sm:text-sm">Active Users</TabsTrigger>
                    <TabsTrigger value="active" className="text-xs sm:text-sm">Active Friends</TabsTrigger>
                  </TabsList>
                  <div className="flex items-center gap-3 flex-shrink-0">
                    {activeFriendsTab === "active" && (
                      <div className="flex items-center justify-between gap-3 w-full">
                        <div className="flex items-center gap-3">
                          {userVisibility ? (
                            <LocationOnIcon className="text-success" style={{ fontSize: 20 }} />
                          ) : (
                            <LocationOffIcon className="text-muted-foreground" style={{ fontSize: 20 }} />
                          )}
                          <div className="flex flex-col">
                            <span className="text-xs sm:text-sm font-medium">
                              Show your active status to friends
                            </span>
                            <span className="text-xs sm:text-sm text-muted-foreground">
                              Friends will see you're working out, but not your exact location
                            </span>
                          </div>
                        </div>
                        <Switch
                          checked={userVisibility}
                          onCheckedChange={handleVisibilityToggle}
                        />
                      </div>
                    )}
                  </div>
                </div>

                <TabsContent value="history" className="mt-0">
                  {!currentUser?.uid ? (
                    <div className="text-center py-8">
                      <p className="text-muted-foreground">Please sign in to view active users</p>
                    </div>
                  ) : !userVenuePreferences || userVenuePreferences.venues.length === 0 ? (
                    <div className="text-center py-8">
                      <LocationOnIcon className="mx-auto text-muted-foreground" style={{ fontSize: 48 }} />
                      <p className="text-muted-foreground mt-4">No venues selected</p>
                      <p className="text-sm text-muted-foreground mt-1">
                        Use the "Add location" button to select venues and discover people nearby
                      </p>
                      <Button
                        onClick={() => setShowQuickCheckIn(true)}
                        className="mt-4"
                      >
                        Add location
                      </Button>
                    </div>
                  ) : (
                    <div className="space-y-6">
                      {userVenuePreferences.venues.map((venueId) => {
                        const venue = getVenueById(venueId);
                        // Include current user so they can confirm their profile is visible
```

```
                        // Users with profileVisible === false are already filtered by the service layer
                        const venueUsers = usersByVenue[venueId] || [];

                        console.log(`[Index] ===== RENDERING VENUE ${venueId} =====`);
                        console.log(`[Index] Venue ${venueId} (${venue?.name}): ${venueUsers.length} users`);
                        if (!venue) return null;

                        return (
                          <div key={venueId} className="space-y-3">
                            <div className="flex items-center gap-2 flex-wrap">
                              <LocationOnIcon className="text-primary" style={{ fontSize: 20 }} />
                              <h3 className="font-semibold text-base">
                                {venue.name}
                              </h3>
                              {venueUsers.length > 0 && (
                                <span className="text-sm text-muted-foreground">
                                  ({venueUsers.length})
                                </span>
                              )}
                              {/* View on Maps button - hidden for now */}
                              {/* <Button
                                size="sm"
                                variant="ghost"
                                onClick={() => openVenueInMaps(venue)}
                                className="h-8 px-2 text-xs ml-auto"
                                title="View on Google Maps"
                              >
                                <MapIcon style={{ fontSize: 16 }} className="mr-1" />
                                <span className="hidden sm:inline">View</span>
                              </Button> */}
                            </div>

                            {venueUsers.length === 0 ? (
                              <div className="text-center py-6 border border-border rounded-lg bg-muted/30">
                                <p className="text-sm text-muted-foreground">
                                  No active users in this venue yet
                                </p>
                                <p className="text-xs text-muted-foreground mt-2">
                                  Other users who have added this venue will appear here
                                </p>
                              </div>
                            ) : (
                              <div className="flex gap-2 overflow-x-auto pb-2 scrollbar-hide">
                                {venueUsers.map((venueUser, index) => {
                                  console.log(`[Index] ===== RENDERING USER CARD ${index} =====`, venueUser);
                                  const activityIcons = {
                                    running: DirectionsRunIcon,
                                    cycling: DirectionsBikeIcon,
                                    walking: DirectionsWalkIcon
                                  };

                                  // Ensure activities is an array
                                  const activities = Array.isArray(venueUser.activities)
                                    ? venueUser.activities
                                    : venueUser.activities
                                      ? [venueUser.activities]
                                      : [];

                                  const isCurrentUser =
                                    String(venueUser.userId || "") === String(currentUser?.uid || "");

                                  console.log(`[Index] User ${venueUser.userId} activities:`, activities);
                                  console.log(`[Index] User ${venueUser.userId} avatar:`, venueUser.avatar);
                                  console.log(`[Index] User ${venueUser.userId} username:`, venueUser.username);

                                  // Get profile picture using utility function
                                  const profilePictureUrl = getProfilePictureUrl(
                                    null, // photoURL not available in VenueUser, it's in avatar field
                                    venueUser.avatar,
                                    venueUser.username
```

```
                                  );

                                  return (
                                    <motion.div
                                      key={venueUser.userId || `user-${index}`}
                                      initial={{ opacity: 0, scale: 0.9 }}
                                      animate={{ opacity: 1, scale: 1 }}
                                      className={`flex flex-col items-center p-2 border border-
border rounded-lg bg-card h...
                                        isCurrentUser ? "ring-2 ring-primary/60" : ""
                                      }`}
                                      onClick={() => {
                                        console.log(`[Index] Clicked on user:`, venueUser.userId);
                                      }}
                                    >
                                      <Avatar
                                        src={profilePictureUrl}
                                        alt={venueUser.username || 'User'}
                                        sx={{ width: 40, height: 40 }}
                                      />
                                      <p className="text-xs font-medium mt-1.5 text-center
truncate w-full max-w-[70px]">
                                        {venueUser.username || 'User'}
                                      </p>
                                      {activities.length > 0 && (
                                        <div className="flex gap-0.5 mt-1">
                                          {activities.map((activity) => {
                                            const ActivityIcon = activityIcons[activity as keyof
typeof activityIcons];
                                            if (!ActivityIcon) return null;
                                            return (
                                              <ActivityIcon
                                                key={activity}
                                                className={
                                                  activity === "running" ? "text-success" :
                                                  activity === "cycling" ? "text-primary" :
                                                  "text-warning"
                                                }
                                                style={{ fontSize: 14 }}
                                              />
                                            );
                                          })}
                                        </div>
                                      )}
                                    </motion.div>
                                  );
                                })}
                              </div>
                            )}
                          </div>
                        );
                      })}
                    </div>
                  )}
                </TabsContent>

                <TabsContent value="active" className="mt-0">
                  {showDummyData && activeFriends.length > 0 &&
activeFriends[0]?.id?.startsWith('dummy-') && (
                    <div className="mb-3">
                      <span className="text-xs px-2 py-0.5 bg-warning/20 text-warning rounded-
full">
                        Demo Data
                      </span>
                    </div>
                  )}
                  {activeFriends.length === 0 ? (
                    <div className="text-center py-8">
                      <FitnessCenterIcon className="mx-auto text-muted-foreground"
style={{ fontSize: 48 }} /><p className="text-muted-foreground mt-4">No active friends right now</p>
                      <p className="text-sm text-muted-foreground mt-1">
                        Friends will appear here when they start a workout
                      </p>
                    </div>
                  ) : (
                    <div className="space-y-3">
```

```
                    {activeFriends.map((friend, index) => {
                      const activityIcon =
                        friend.activity === "running" ? DirectionsRunIcon :
                        friend.activity === "cycling" ? DirectionsBikeIcon :
                        DirectionsWalkIcon;
                      const ActivityIcon = activityIcon;

                      return (
                        <motion.div
                          key={friend.id}
                          initial={{ opacity: 0, x: -20 }}
                          animate={{ opacity: 1, x: 0 }}
                          transition={{ delay: index * 0.05 }}
                          className="flex items-center gap-4 p-4 border border-border rounded-lg
hover:shadow-md transit...     >
                          <div
                            className="relative cursor-pointer"
                            onClick={() => setSelectedActiveFriend({
                              id: friend.id,
                              name: friend.name,
                              username: friend.username,
                              photoURL: friend.photoURL,
                              activity: friend.activity
                            })}
                          >
                            {(() => {
                              // Get profile picture using utility function
                              const profilePictureUrl = getProfilePictureUrl(
                                friend.photoURL,
                                friend.avatar,
                                friend.name || friend.username
                              );
                              return (
                                <Avatar
                                  src={profilePictureUrl}
                                  alt={friend.name || friend.username}
                                  sx={{ width: 56, height: 56 }}
                                />
                              );
                            })()}
                            <div className="absolute -bottom-1 -right-1 w-5 h-5 bg-success
rounded-full border-2 border-...     <div className="w-2 h-2 bg-white rounded-full animate-pulse" />
                            </div>
                          </div>

                          <div
                            className="flex-1 min-w-0 cursor-pointer"
                            onClick={() => setSelectedActiveFriend({
                              id: friend.id,
                              name: friend.name,
                              username: friend.username,
                              photoURL: friend.photoURL,
                              activity: friend.activity
                            })}
                          >
                            <div className="flex items-center gap-2">
                              <h3 className="font-semibold text-sm truncate">
                                {friend.name || friend.username || "Friend"}
                              </h3>
                              <FitnessCenterIcon className="text-success" style={{ fontSize:
16 }} />
                            </div>

                            <div className="flex items-center gap-2 mt-1">
                              <ActivityIcon
                                className={
                                  friend.activity === "running" ? "text-success" :
                                  friend.activity === "cycling" ? "text-primary" :
                                  "text-warning"
                                }
                                style={{ fontSize: 16 }}
                              />
```

```
                            <span className="text-xs text-muted-foreground capitalize">
                              {friend.activity || "working out"}
                            </span>
                          </div>
                        </div>

                        <div className="flex items-center gap-2">
                          <Button
                            variant="outline"
                            size="sm"
                            onClick={(e) => {
                              e.stopPropagation();
                              setSelectedActiveFriend({
                                id: friend.id,
                                name: friend.name,
                                username: friend.username,
                                photoURL: friend.photoURL,
                                activity: friend.activity
                              });
                            }}
                            className="flex items-center gap-1"
                          >
                            <PersonIcon style={{ fontSize: 16 }} />
                            <span className="hidden sm:inline">Profile</span>
                          </Button>
                          <Button
                            variant="default"
                            size="sm"
                            onClick={(e) => {
                              e.stopPropagation();
                              navigate("/messages", {
                                state: {
                                  userId: friend.id,
                                  userName: friend.name || friend.username
                                }
                              });
                            }}
                            className="flex items-center gap-1"
                          >
                            <ChatIcon style={{ fontSize: 16 }} />
                            <span className="hidden sm:inline">Message</span>
                          </Button>
                        </div>
                      </motion.div>
                    );
                  })}
                </div>
              )}
            </TabsContent>
          </Tabs>
        </Card>
      </motion.div>

          {/* Profile Discovery Section - Show below if user has venues */}
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: 0.12 }}
          >
            <Card className="p-6">
              <div className="flex items-center justify-between mb-4">
                <div>
                  <h2 className="text-xl font-bold">Meet New People</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    Add your general location to connect with others
                  </p>
                </div>
              </div>

              {/* General Location Input */}
              <div className="p-4 border border-border rounded-lg space-y-3">
```

```
                  <div className="flex items-center gap-2">
                    <LocationOnIcon className="text-primary" style={{ fontSize: 20 }} />
                    <span className="text-sm font-medium">General Location</span>
                  </div>
                  <p className="text-xs text-muted-foreground">
                    Select venues where you work out to connect with others nearby
                  </p>
                  <div>
                    <span className="text-sm text-muted-foreground">
                      {generalLocation || "No location set"}
                    </span>
                  </div>

                  {/* Quick Check-in Button */}
                  <Button
                    variant="default"
                    onClick={() => setShowQuickCheckIn(true)}
                    className="w-full h-11 border-primary bg-primary hover:bg-primary/90 text-
primary-foreground mt-3 sh...
                    <LocationOnIcon style={{ fontSize: 18 }} className="mr-2" />
                    <span className="text-sm font-semibold">Add venues</span>
                  </Button>
                </div>
              </Card>
            </motion.div>
          </>
        ) : (
          <>
            {/* Profile Discovery Section - Show first if user has no venues */}
            <motion.div
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ delay: 0.11 }}
            >
              <Card className="p-6">
                <div className="flex items-center justify-between mb-4">
                  <div>
                    <h2 className="text-xl font-bold">Meet New People</h2>
                    <p className="text-sm text-muted-foreground mt-1">
                      Add your general location to connect with others
                    </p>
                  </div>
                </div>

                {/* General Location Input */}
                <div className="p-4 border border-border rounded-lg space-y-3">
                  <div className="flex items-center gap-2">
                    <LocationOnIcon className="text-primary" style={{ fontSize: 20 }} />
                    <span className="text-sm font-medium">General Location</span>
                  </div>
                  <p className="text-xs text-muted-foreground">
                    Select venues where you work out to connect with others nearby
                  </p>
                  <div>
                    <span className="text-sm text-muted-foreground">
                      {generalLocation || "No location set"}
                    </span>
                  </div>

                  {/* Quick Check-in Button */}
                  <Button
                    variant="default"
                    onClick={() => setShowQuickCheckIn(true)}
                    className="w-full h-11 border-primary bg-primary hover:bg-primary/90 text-
primary-foreground mt-3 sh...
                    <LocationOnIcon style={{ fontSize: 18 }} className="mr-2" />
                    <span className="text-sm font-semibold">Add venues</span>
                  </Button>
                </div>
              </Card>
            </motion.div>
```

```
1203 |
1204 |             {/* Friends Section with Tabs - Show below if user has no venues */}
1205 |             <motion.div
1206 |               initial={{ opacity: 0, y: 20 }}
1207 |               animate={{ opacity: 1, y: 0 }}
1208 |               transition={{ delay: 0.12 }}
1209 |             >
1210 |               <Card className="p-6">
1211 |                 <Tabs value={activeFriendsTab} onValueChange={(value) =>
1212 |                   setActiveFriendsTab(value as "active" | "history")
1213 |                 }>
1214 |                   <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between
     gap-3 mb-4">
1215 |                     <TabsList className="grid grid-cols-2 w-full sm:w-auto flex-shrink-0">
1216 |                       <TabsTrigger value="history" className="text-xs sm:text-sm">Active Users</
     TabsTrigger>
1217 |                       <TabsTrigger value="active" className="text-xs sm:text-sm">Active Friends</
     TabsTrigger>
1218 |                     </TabsList>
1219 |                     <div className="flex items-center gap-3 flex-shrink-0">
1220 |                       {activeFriendsTab === "active" && (
1221 |                         <div className="flex items-center justify-between gap-3 w-full">
1222 |                           <div className="flex items-center gap-3">
1223 |                             {userVisibility ? (
1224 |                               <LocationOnIcon className="text-success" style={{ fontSize: 20 }} /
     >
1225 |                             ) : (
1226 |                               <LocationOffIcon className="text-muted-foreground"
     style={{ fontSize: 20 }} />
1227 |                             )}
1228 |                             <div className="flex flex-col">
1229 |                               <span className="text-xs sm:text-sm font-medium">
1230 |                                 Show your active status to friends
1231 |                               </span>
1232 |                               <span className="text-xs sm:text-sm text-muted-foreground">
1233 |                                 Friends will see you're working out, but not your exact location
1234 |                               </span>
1235 |                             </div>
1236 |                           </div>
1237 |                           <Switch
1238 |                             checked={userVisibility}
1239 |                             onCheckedChange={handleVisibilityToggle}
1240 |                           />
1241 |                         </div>
1242 |                       )}
1243 |                     </div>
1244 |                   </div>
1245 |
1246 |                   <TabsContent value="history" className="mt-0">
1247 |                     {!currentUser?.uid ? (
1248 |                       <div className="text-center py-8">
1249 |                         <p className="text-muted-foreground">Please sign in to view active
     users</p>
1250 |                       </div>
1251 |                     ) : !userVenuePreferences || userVenuePreferences.venues.length === 0 ? (
1252 |                       <div className="text-center py-8">
1253 |                         <LocationOnIcon className="mx-auto text-muted-foreground"
     style={{ fontSize: 48 }} />
1254 |                         <p className="text-muted-foreground mt-4">No venues selected</p>
1255 |                         <p className="text-sm text-muted-foreground mt-1">
1256 |                           Use the "Add location" button to select venues and discover people
     nearby
1257 |                         </p>
1258 |                         <Button
1259 |                           onClick={() => setShowQuickCheckIn(true)}
1260 |                           className="mt-4"
1261 |                         >
1262 |                           Add location
1263 |                         </Button>
1264 |                       </div>
1265 |                     ) : (
1266 |                       <div className="space-y-6">
1267 |                         {userVenuePreferences.venues.map((venueId) => {
1268 |                           const venue = getVenueById(venueId);
1269 |                           const venueUsers = usersByVenue[venueId] || [];
1270 |
1271 |                           if (!venue) return null;
1272 |
1273 |                           return (
1274 |                             <div key={venueId} className="space-y-3">
1275 |                               <div className="flex items-center gap-2 flex-wrap">
```

```
                                <LocationOnIcon className="text-primary" style={{ fontSize:
20}}/>
                                <h3 className="font-semibold text-base">
                                  {venue.name}
                                </h3>
                                {venueUsers.length > 0 && (
                                  <span className="text-sm text-muted-foreground">
                                    ({venueUsers.length})
                                  </span>
                                )}
                              </div>

                              {venueUsers.length === 0 ? (
                                <div className="text-center py-6 border border-border rounded-lg
bg-muted/30">
                                  <p className="text-sm text-muted-foreground">
                                    No active users in this venue yet
                                  </p>
                                  <p className="text-xs text-muted-foreground mt-2">
                                    Other users who have added this venue will appear here
                                  </p>
                                </div>
                              ) : (
                                <div className="flex gap-2 overflow-x-auto pb-2 scrollbar-hide">
                                  {venueUsers.map((venueUser, index) => {
                                    const activityIcons = {
                                      running: DirectionsRunIcon,
                                      cycling: DirectionsBikeIcon,
                                      walking: DirectionsWalkIcon
                                    };

                                    const activities = Array.isArray(venueUser.activities)
                                      ? venueUser.activities
                                      : venueUser.activities
                                        ? [venueUser.activities]
                                        : [];

                                    const isCurrentUser =
                                      String(venueUser.userId || "") === String(currentUser?.uid
|| "");

                                    // Get profile picture using utility function
                                    const profilePictureUrl = getProfilePictureUrl(
                                      null, // photoURL not available in VenueUser, it's in
avatar field
                                      venueUser.avatar,
                                      venueUser.username
                                    );

                                    return (
                                      <motion.div
                                        key={venueUser.userId || `user-${index}`}
                                        initial={{ opacity: 0, scale: 0.9 }}
                                        animate={{ opacity: 1, scale: 1 }}
                                        className={`flex flex-col items-center p-2 border border-
border rounded-lg bg-ca...
                                          isCurrentUser ? "ring-2 ring-primary/60" : ""
                                        }`}
                                      >
                                        <Avatar
                                          src={profilePictureUrl}
                                          alt={venueUser.username || 'User'}
                                          sx={{ width: 40, height: 40 }}
                                        />
                                        <p className="text-xs font-medium mt-1.5 text-center
truncate w-full max-w-[70px]...
                                          {venueUser.username || 'User'}
                                        </p>
                                        {activities.length > 0 && (
                                          <div className="flex gap-0.5 mt-1">
                                            {activities.map((activity) => {
                                              const ActivityIcon = activityIcons[activity as
keyof typeof activityIcons]...
                                              if (!ActivityIcon) return null;
                                              return (
                                                <ActivityIcon
                                                  key={activity}
                                                  className={
```

```
                                                  activity === "running" ? "text-success" :
                                                  activity === "cycling" ? "text-primary" :
                                                  "text-warning"
                                                }
                                                style={{ fontSize: 14 }}
                                              />
                                            );
                                          })}
                                        </div>
                                      )}
                                    </motion.div>
                                  );
                                })}
                              </div>
                            )}
                          </div>
                        );
                      })}
                    </div>
                  )}
                </TabsContent>

                <TabsContent value="active" className="mt-0">
                  {showDummyData && activeFriends.length > 0 &&
activeFriends[0]?.id?.startsWith('dummy-') && (
                    <div className="mb-3">
                      <span className="text-xs px-2 py-0.5 bg-warning/20 text-warning rounded-
full">
                        Demo Data
                      </span>
                    </div>
                  )}
                  {activeFriends.length === 0 ? (
                    <div className="text-center py-8">
                      <FitnessCenterIcon className="mx-auto text-muted-foreground"
style={{ fontSize: 48 }} />
                      <p className="text-muted-foreground mt-4">No active friends right now</p>
                      <p className="text-sm text-muted-foreground mt-1">
                        Friends will appear here when they start a workout
                      </p>
                    </div>
                  ) : (
                    <div className="space-y-3">
                      {activeFriends.map((friend, index) => {
                        const activityIcon =
                          friend.activity === "running" ? DirectionsRunIcon :
                          friend.activity === "cycling" ? DirectionsBikeIcon :
                          DirectionsWalkIcon;
                        const ActivityIcon = activityIcon;

                        return (
                          <motion.div
                            key={friend.id}
                            initial={{ opacity: 0, x: -20 }}
                            animate={{ opacity: 1, x: 0 }}
                            transition={{ delay: index * 0.05 }}
                            className="flex items-center gap-4 p-4 border border-border
rounded-lg hover:shadow-md tra...  >
                            <div
                              className="relative cursor-pointer"
                              onClick={() => setSelectedActiveFriend({
                                id: friend.id,
                                name: friend.name,
                                username: friend.username,
                                photoURL: friend.photoURL,
                                activity: friend.activity
                              })}
                            >
                              {(() => {
                                // Get profile picture using utility function
                                const profilePictureUrl = getProfilePictureUrl(
                                  friend.photoURL,
                                  friend.avatar,
                                  friend.name || friend.username
```

```
                                    );
                                    return (
                                      <Avatar
                                        src={profilePictureUrl}
                                        alt={friend.name || friend.username}
                                        sx={{ width: 56, height: 56 }}
                                      />
                                    );
                                  })()}
                                  <div classNam="absolute -bottom-1 -right-1 w-5 h-5 bg-success rounded-full border-2 bor...
                                    <div className="w-2 h-2 bg-white rounded-full animate-pulse" />
                                  </div>
                                </div>

                                <div
                                  className="flex-1 min-w-0 cursor-pointer"
                                  onClick={() => setSelectedActiveFriend({
                                    id: friend.id,
                                    name: friend.name,
                                    username: friend.username,
                                    photoURL: friend.photoURL,
                                    activity: friend.activity
                                  })}
                                >
                                  <div className="flex items-center gap-2">
                                    <h3 className="font-semibold text-sm truncate">
                                      {friend.name || friend.username || "Friend"}
                                    </h3>
                                    <FitnessCenterIcon className="text-success" style={{ fontSize: 16 }} />
                                  </div>

                                  <div className="flex items-center gap-2 mt-1">
                                    <ActivityIcon
                                      className={
                                        friend.activity === "running" ? "text-success" :
                                        friend.activity === "cycling" ? "text-primary" :
                                        "text-warning"
                                      }
                                      style={{ fontSize: 16 }}
                                    />
                                    <span className="text-xs text-muted-foreground capitalize">
                                      {friend.activity || "working out"}
                                    </span>
                                  </div>
                                </div>

                                <div className="flex items-center gap-2">
                                  <Button
                                    variant="outline"
                                    size="sm"
                                    onClick={(e) => {
                                      e.stopPropagation();
                                      setSelectedActiveFriend({
                                        id: friend.id,
                                        name: friend.name,
                                        username: friend.username,
                                        photoURL: friend.photoURL,
                                        activity: friend.activity
                                      });
                                    }}
                                    className="flex items-center gap-1"
                                  >
                                    <PersonIcon style={{ fontSize: 16 }} />
                                    <span className="hidden sm:inline">Profile</span>
                                  </Button>
                                  <Button
                                    variant="default"
                                    size="sm"
                                    onClick={(e) => {
                                      e.stopPropagation();
                                      navigate("/messages", {
```

```
                                    state: {
                                      userId: friend.id,
                                      userName: friend.name || friend.username
                                    }
                                  });
                                }}
                                className="flex items-center gap-1"
                              >
                                <ChatIcon style={{ fontSize: 16 }} />
                                <span className="hidden sm:inline">Message</span>
                              </Button>
                            </div>
                          </motion.div>
                        );
                      })}
                    </div>
                  )}
                </TabsContent>
              </Tabs>
            </Card>
          </motion.div>
        </>
      )}

      {/* Top Matches Section */}
      {topMatches.length > 0 && (
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.15 }}
        >
          <Card className="p-6">
            <div className="flex items-center justify-between mb-4">
              <h2 className="text-xl font-bold">Top Matches</h2>
              <Button
                variant="ghost"
                size="sm"
                onClick={() => navigate("/map")}
                className="text-primary"
              >
                View All
              </Button>
            </div>
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
              {topMatches.slice(0, 3).map((match, index) => {
                const user = match.user;
                const distanceKm = match.distance / 1000;
                const score = match.score;

                return (
                  <motion.div
                    key={user.uid}
                    initial={{ opacity: 0, scale: 0.95 }}
                    animate={{ opacity: 1, scale: 1 }}
                    transition={{ delay: index * 0.1 }}
                    className="p-4 border border-border rounded-lg hover:shadow-md transition-
shadow cursor-pointer"
                    onClick={() => navigate("/map")}
                  >
                    <div className="flex items-start gap-3">
                      <div className="relative">
                        {(() => {
                          // Get profile picture using utility function
                          const profilePictureUrl = getProfilePictureUrl(
                            user.photoURL,
                            user.avatar,
                            user.name
                          );
                          return (
                            <img
                              src={profilePictureUrl}
                              alt={user.name}
```

```
                                  className="w-12 h-12 rounded-full border-2 border-primary"
                                />
                              );
                            })()}
                            <div className={`absolute -bottom-1 -right-1 w-5 h-5 rounded-full flex items-center justify-ce...
                              score >= 0.8 ? "bg-success text-white" :
                              score >= 0.6 ? "bg-primary text-white" :
                              "bg-warning text-white"
                            }`}>
                              {Math.round(score * 100)}
                            </div>
                          </div>
                          <div className="flex-1 min-w-0">
                            <h3 className="font-semibold text-sm truncate">{user.name || "User"}</h3>
                            <div className="flex items-center gap-2 text-xs text-muted-foreground mt-1">
                              <span>{formatDistance(distanceKm)}</span>
                              <span>•</span>
                              <span className="capitalize">{user.activity}</span>
                            </div>
                            <div className="flex items-center gap-2 mt-2">
                              <span className="text-xs px-2 py-0.5 bg-muted rounded-full">
                                {user.fitnessLevel}
                              </span>
                              {user.pace > 0 && (
                                <span className="text-xs text-muted-foreground">
                                  {user.activity === "cycling"
                                    ? `${user.pace.toFixed(1)} km/h`
                                    : `${user.pace.toFixed(1)} min/km`}
                                </span>
                              )}
                            </div>
                          </div>
                        </div>
                      </motion.div>
                    );
                  })}
                </div>
              </Card>
            </motion.div>
          )}

          {/* Activity Feed */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.3 }}
            className="space-y-4"
          >
            {loading ? (
              <Card className="p-12 text-center">
                <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary mx-auto"></div>
                <p className="text-muted-foreground mt-4">Loading workouts...</p>
              </Card>
            ) : filteredPosts.length === 0 ? (
              <Card className="p-12 text-center">
                <p className="text-muted-foreground">No workouts to show yet</p>
                <p className="text-sm text-muted-foreground mt-2">Start a workout to see it here!</p>
                <Button onClick={() => navigate("/map")} className="mt-4">
                  Go to Map
                </Button>
              </Card>
            ) : (
              filteredPosts.map((post, index) => (
                <motion.div
                  key={post.id}
                  initial={{ opacity: 0, y: 20 }}
                  animate={{ opacity: 1, y: 0 }}
                  transition={{ delay: index * 0.05 }}
                >
                  <WorkoutPost
                    post={post}
```

```
                  onCommentClick={handleCommentClick}
                  useMetric={useMetric}
                  currentUserId={currentUser?.uid || ""}
                />
              </motion.div>
            ))
          )}
        </motion.div>
      </div>

      {/* Comment Drawer */}
      <CommentDrawer
        isOpen={isCommentDrawerOpen}
        post={selectedPost}
        onClose={() => setIsCommentDrawerOpen(false)}
        currentUserId={currentUser?.uid || ""}
        currentUsername={userProfile?.username || currentUser?.displayName || "You"}
        currentAvatar={currentUser?.photoURL || ""}
      />

      {/* Profile View Modal for Active Friends */}
      <AnimatePresence>
        {selectedActiveFriend && activeFriendData && (() => {
          const friendId = selectedActiveFriend.id;
          const friendStatus = getFriendStatus(friendId);
          const activityLabel = selectedActiveFriend.activity
            ? selectedActiveFriend.activity.charAt(0).toUpperCase() +
selectedActiveFriend.activity.slice(1)
            : "Active";

          return (
            <ProfileView
              user={{
                id: friendId,
                name: activeFriendData?.name || activeFriendData?.username ||
selectedActiveFriend.name || selectedActiv...
                distance: "Active Now", // Since we're not sharing exact location
                activity: activityLabel,
                avatar: activeFriendData?.photoURL || selectedActiveFriend.photoURL || `https://
ui-avatars.com/api/?name=...
                photos: activeFriendData?.photos || [],
                bio: activeFriendData?.bio,
              }}
              friendStatus={friendStatus}
              onClose={() => {
                setSelectedActiveFriend(null);
                setActiveFriendData(null);
              }}
              onSendMessage={() => {
                setSelectedActiveFriend(null);
                setActiveFriendData(null);
                navigate("/messages", {
                  state: {
                    userId: friendId,
                    userName: activeFriendData?.name || activeFriendData?.username ||
selectedActiveFriend.name || selec...
                  });
              }}
              onAddFriend={() => handleAddFriend(friendId)}
              onAcceptFriend={() => handleAcceptFriend(friendId)}
              onDeclineFriend={() => handleDeclineFriend(friendId)}
              onUnfriend={() => handleUnfriend(friendId)}
              onReport={(reason, details) => handleReportUser(friendId, reason, details)}
              onBlock={() => handleBlockUser(friendId)}
            />
          );
        })()}
      </AnimatePresence>

      {/* Profile View Modal for Workout History */}
      <AnimatePresence>
        {selectedWorkoutHistoryItem && (() => {
          const item = selectedWorkoutHistoryItem;
          const user = item.userData;
```

```
1700 |           const workout = item.workout;
1701 |           const userId = item.userId;
1702 |
1703 |           // Calculate distance if possible
1704 |           let distance = "Unknown";
1705 |           if (location?.lat && location?.lng && user?.lat && user?.lng) {
1706 |             const distanceKm = calculateDistance(
1707 |               location.lat,
1708 |               location.lng,
1709 |               user.lat,
1710 |               user.lng
1711 |             );
1712 |             distance = formatDistance(distanceKm);
1713 |           }
1714 |
1715 |           const friendStatus = getFriendStatus(userId);
1716 |           const activityLabel = workout.activity.charAt(0).toUpperCase() +
workout.activity.slice(1);
1717 |
1718 |           return (
1719 |             <ProfileView
1720 |               user={{
1721 |                 id: userId,
1722 |                 name: user?.name || user?.username || "User",
1723 |                 distance: distance,
1724 |                 activity: activityLabel,
1725 |                 avatar: user?.photoURL || `https://ui-avatars.com/api/?name=${user?.name ||
user?.username || 'User'}`,
1726 |                 photos: user?.photos || [],
1727 |                 bio: user?.bio,
1728 |               }}
1729 |               friendStatus={friendStatus}
1730 |               onClose={() => setSelectedWorkoutHistoryItem(null)}
1731 |               onSendMessage={() => {
1732 |                 setSelectedWorkoutHistoryItem(null);
1733 |                 navigate("/messages", {
1734 |                   state: { userId, userName: user?.name || user?.username }
1735 |                 });
1736 |               }}
1737 |               onAddFriend={() => handleAddFriend(userId)}
1738 |               onAcceptFriend={() => handleAcceptFriend(userId)}
1739 |               onDeclineFriend={() => handleDeclineFriend(userId)}
1740 |               onUnfriend={() => handleUnfriend(userId)}
1741 |               onReport={(reason, details) => handleReportUser(userId, reason, details)}
1742 |               onBlock={() => handleBlockUser(userId)}
1743 |             />
1744 |           );
1745 |         })()}
1746 |       </AnimatePresence>
1747 |
1748 |       {/* Notification Drawer */}
1749 |       <AnimatePresence>
1750 |         {showNotificationDrawer && (
1751 |           <>
1752 |             <motion.div
1753 |               initial={{ opacity: 0 }}
1754 |               animate={{ opacity: 1 }}
1755 |               exit={{ opacity: 0 }}
1756 |               onClick={() => setShowNotificationDrawer(false)}
1757 |               className="fixed inset-0 bg-black/50 z-40"
1758 |             />
1759 |             <motion.div
1760 |               initial={{ y: "100%" }}
1761 |               animate={{ y: 0 }}
1762 |               exit={{ y: "100%" }}
1763 |               transition={{ type: "spring", damping: 25, stiffness: 300 }}
1764 |               onClick={(e) => e.stopPropagation()}
1765 |               className={`fixed bottom-0 left-0 right-0 z-50 bg-card rounded-t-3xl shadow-
elevation-4 p-6 pb-24 border-t...
1766 |               style={{
1767 |                 maxHeight: '85vh',
1768 |                 minHeight: '200px',
1769 |                 overflowY: 'auto'
1770 |               }}
```

```
            >
              {/* Header */}
              <div className="flex items-center justify-between mb-6">
                <div>
                  <h2 className="text-2xl font-bold text-foreground">Notifications</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    {unreadCount > 0 ? `${unreadCount} unread notification${unreadCount > 1 ? 's' : ''}` : 'All caught up'}
                  </p>
                </div>
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={() => setShowNotificationDrawer(false)}
                  className="rounded-full"
                >
                  <CloseIcon />
                </Button>
              </div>

              {/* Notifications List - Sorted by newest first */}
              {notifications.length > 0 ? (
                <div className="space-y-2">
                  {[...notifications]
                    .sort((a, b) => b.timestamp - a.timestamp) // Newest first
                    .map((notification) => {
                      const getNotificationIcon = () => {
                        switch (notification.type) {
                          case "message":
                            return <MailIcon style={{ fontSize: 20 }} className="text-primary" />;
                          case "message_request":
                            return <ChatBubbleIcon style={{ fontSize: 20 }} className="text-blue-500" />;
                          case "friend_request":
                            return <PersonAddIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "poke":
                            return <TouchAppIcon style={{ fontSize: 20 }} className="text-purple-500" />;
                          case "friend_accepted":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "workout_complete":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "achievement":
                            return <EmojiEventsIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "username_change_required":
                            return <EditIcon style={{ fontSize: 20 }} className="text-warning" />;
                          default:
                            return <NotificationsIcon style={{ fontSize: 20 }} />;
                        }
                      };

                      const getNotificationTitle = () => {
                        switch (notification.type) {
                          case "message":
                            return notification.userName;
                          case "message_request":
                            return notification.userName;
                          case "friend_request":
                            return notification.userName;
                          case "poke":
                            return notification.userName;
                          case "friend_accepted":
                            return notification.userName;
                          case "workout_complete":
                            return "Workout Completed";
                          case "achievement":
                            return notification.message || "Congrats for a new achievement!";
                          case "username_change_required":
                            return "Username Change Required";
                          default:
                            return notification.userName;
                        }
                      };

```

```
                      const getNotificationMessage = () => {
                        switch (notification.type) {
                          case "message":
                            return notification.message || "Sent you a message";
                          case "message_request":
                            return "wants to start a conversation with you";
                          case "friend_request":
                            return "wants to add you as a friend";
                          case "poke":
                            return "poked you! They're interested in matching";
                          case "friend_accepted":
                            return "accepted your friend request";
                          case "workout_complete":
                            return notification.message || "Workout completed successfully!";
                          case "achievement":
                            return notification.message || "Congrats for a new achievement!";
                          case "username_change_required":
                            return notification.message || "Your username has been changed due
to misuse. Please update ...
                          default:
                            return "";
                        }
                      };

                      const formatTimestamp = (timestamp: number) => {
                        const now = Date.now();
                        const diff = now - timestamp;
                        const minutes = Math.floor(diff / 60000);
                        const hours = Math.floor(diff / 3600000);
                        const days = Math.floor(diff / 86400000);

                        if (minutes < 1) return "Just now";
                        if (minutes < 60) return `${minutes}m ago`;
                        if (hours < 24) return `${hours}h ago`;
                        if (days < 7) return `${days}d ago`;
                        return new Date(timestamp).toLocaleDateString();
                      };

                      return (
                        <motion.div
                          key={notification.id}
                          initial={{ opacity: 0, y: 10 }}
                          animate={{ opacity: 1, y: 0 }}
                          className={`p-4 rounded-lg border border-border/50 hover:bg-muted/50
transition-colors cursor-...
                            !notification.read ? 'bg-primary/5 border-primary/30' : ''
                          }`}
                          onClick={() => {
                            handleNotificationTap(notification);
                            setShowNotificationDrawer(false);
                          }}
                        >
                          <div className="flex items-start gap-3">
                            {/* Icon */}
                            <div className={`flex-shrink-0 p-2 rounded-full ${
                              notification.type === "message"
                                ? "bg-primary/15"
                                : notification.type === "message_request"
                                ? "bg-blue-500/15"
                                : notification.type === "friend_request"
                                ? "bg-warning/15"
                                : notification.type === "poke"
                                ? "bg-purple-500/15"
                                : notification.type === "workout_complete"
                                ? "bg-success/15"
                                : notification.type === "achievement"
                                ? "bg-warning/15"
                                : notification.type === "friend_accepted"
                                ? "bg-success/15"
                                : "bg-gray-500/15"
                            }`}>
                              {getNotificationIcon()}
                            </div>
```

```
                              {/* Content */}
                              <div className="flex-1 min-w-0">
                                <div className="flex items-center justify-between mb-1">
                                  <p className="font-semibold text-sm text-foreground">
                                    {getNotificationTitle()}
                                  </p>
                                  {!notification.read && (
                                    <span className="w-2 h-2 bg-primary rounded-full flex-
shrink-0"></span>
                                  )}
                                </div>
                                <p className="text-sm text-muted-foreground mb-1">
                                  {getNotificationMessage()}
                                </p>
                                <p className="text-xs text-muted-foreground">
                                  {formatTimestamp(notification.timestamp)}
                                </p>
                              </div>

                              {/* Dismiss button */}
                              <button
                                onClick={(e) => {
                                  e.stopPropagation();
                                  dismissNotification(notification.id);
                                }}
                                className="flex-shrink-0 p-1 hover:bg-accent rounded-full
transition-colors"
                              >
                                <CloseIcon style={{ fontSize: 16 }} className="text-muted-
foreground" />
                              </button>
                            </div>
                          </motion.div>
                        );
                      })}
                    </div>
                  ) : (
                    <div className="py-12 text-center">
                      <NotificationsIcon className="text-muted-foreground mx-auto mb-2"
style={{ fontSize: 48 }} />
                      <p className="text-sm text-muted-foreground">No notifications yet</p>
                    </div>
                  )}
                </motion.div>
              </>
            )}
          </AnimatePresence>

          {/* Quick Check-in Modal */}
          {showQuickCheckIn && (
            <QuickCheckInModal
              onClose={() => setShowQuickCheckIn(false)}
            />
          )}

          <BottomNavigation />
        </div>
      );
};

export default Index;
```

File: src/pages/LoginScreen.tsx

Lines: 975

```
import { useState, useEffect, useRef } from "react";
import { motion } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
  DialogDescription,
} from "@/components/ui/dialog";
import { useNavigate, useLocation } from "react-router-dom";
import EmailIcon from "@mui/icons-material/Email";
import PersonIcon from "@mui/icons-material/Person";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import PeopleIcon from "@mui/icons-material/People";
import { signUpWithEmail, checkEmailExists, sendEmailVerificationCode, verifyEmailCode,
sendEmailOTPForSignIn, verifyEma
import { toast } from "sonner";
import { useAuth } from "@/hooks/useAuth";
import { auth } from "@/services/firebase";

const LoginScreen = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { user, loading: authLoading } = useAuth();
  // Email OTP state
  const [email, setEmail] = useState("");
  const [displayName, setDisplayName] = useState("");
  const [otpCode, setOtpCode] = useState("");
  const [isSignUp, setIsSignUp] = useState(false);
  const [otpSent, setOtpSent] = useState(false);
  const [otpVerified, setOtpVerified] = useState(false);
  const [password, setPassword] = useState(""); // Only needed for sign-up final step
  // Common state
  const [loading, setLoading] = useState(false);
  const [sendingCode, setSendingCode] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const hasCheckedRedirect = useRef(false);
  const hasRedirected = useRef(false);
  // Terms and Privacy dialogs
  const [showTerms, setShowTerms] = useState(false);
  const [showPrivacy, setShowPrivacy] = useState(false);

  // Check if mobile device (including Chrome mobile view)
  const isMobile = () => {
    const userAgent = navigator.userAgent.toLowerCase();
    const isMobileUA = /android|webos|iphone|ipad|ipod|blackberry|iemobile|opera mini/
i.test(userAgent);
    const isTouchDevice = 'ontouchstart' in window || navigator.maxTouchPoints > 0;
    const isSmallScreen = window.innerWidth < 768;
    const isMobileView = isSmallScreen || (isTouchDevice && window.innerWidth < 1024);

    // Chrome mobile view often has mobile-like behavior even on desktop
    const isChromeMobileView = window.innerWidth < 768 && /chrome/i.test(userAgent);

    return isMobileUA || isMobileView || isChromeMobileView;
  };

  // Handle Google sign-in redirect result (prevents redirect loop)
  useEffect(() => {
    const checkRedirectResult = async () => {
      // Only check on login page
      if (location.pathname !== "/login") {
        return;
      }

      try {
```

```
 67 |         const result = await handleRedirectResult();
 68 |         if (result) {
 69 |           console.log("'  Google sign-in redirect handled successfully");
 70 |           // User will be automatically redirected by the auth check below
 71 |           // Don't set hasRedirected here - let the auth check handle it
 72 |         }
 73 |       } catch (error) {
 74 |         console.error("Error handling redirect result:", error);
 75 |         // Don't show error - user can try again
 76 |       }
 77 |     };
 78 |
 79 |     // Check redirect result when component mounts or pathname changes
 80 |     checkRedirectResult();
 81 |   }, [location.pathname]);
 82 |
 83 |   // Redirect if user is already authenticated
 84 |   useEffect(() => {
 85 |     // Only redirect if we're actually on the login page
 86 |     if (location.pathname !== "/login") {
 87 |       return;
 88 |     }
 89 |
 90 |     // Prevent multiple redirects
 91 |     if (hasRedirected.current || authLoading || !user) {
 92 |       return;
 93 |     }
 94 |
 95 |     // User is already logged in, redirect them away from login page
 96 |     const checkUserProfile = async () => {
 97 |       try {
 98 |         // Add delay on mobile to ensure authentication state is fully updated
 99 |         const mobileDelay = isMobile() ? 2000 : 0; // 2 seconds on mobile, no delay on desktop
100 |         if (mobileDelay > 0) {
101 |           console.log("Ø=Üñ Mobile detected, waiting", mobileDelay, "ms before redirect...");
102 |           await new Promise(resolve => setTimeout(resolve, mobileDelay));
103 |         }
104 |
105 |         // Double-check user is still authenticated after delay
106 |         if (!user && !auth.currentUser) {
107 |           console.log("& þ  User no longer authenticated after delay, aborting redirect");
108 |           hasRedirected.current = false;
109 |           return;
110 |         }
111 |
112 |         hasRedirected.current = true;
113 |         const { getUserData } = await import("@/services/authService");
114 |         const userData = await getUserData(user.uid);
115 |
116 |         if (userData && userData.activity) {
117 |           navigate("/", { replace: true });
118 |         } else {
119 |           navigate("/profile-setup", { replace: true });
120 |         }
121 |       } catch (err) {
122 |         console.error("Error checking user profile:", err);
123 |         hasRedirected.current = false; // Reset on error
124 |       }
125 |     };
126 |     checkUserProfile();
127 |   }, [user, authLoading, navigate, location.pathname]);
128 |
129 |
130 |   // Handle sending OTP for email (works for both sign-in and sign-up)
131 |   const handleSendOTP = async () => {
132 |     setSendingCode(true);
133 |     setError(null);
134 |
135 |     try {
136 |       // Validate email
137 |       if (!email || !email.includes('@')) {
```

```
138 |         setError("Please enter a valid email address");
139 |         setSendingCode(false);
140 |         return;
141 |       }
142 |
143 |       // For sign-up, also validate display name
144 |       if (isSignUp && (!displayName || displayName.trim().length === 0)) {
145 |         setError("Please enter your name");
146 |         setSendingCode(false);
147 |         return;
148 |       }
149 |
150 |       console.log("Ø=Üç Sending OTP to:", email);
151 |
152 |       if (isSignUp) {
153 |         // For sign-up, use the existing verification code function
154 |         await sendEmailVerificationCode(email);
155 |         toast.success("Verification code sent! Check your email.", {
156 |           description: "Check your inbox for the 6-digit code. If you don't see it, check spam
folder.",
157 |           duration: 10000,
158 |         });
159 |       } else {
160 |         // For sign-in, use the new OTP function
161 |         await sendEmailOTPForSignIn(email);
162 |         toast.success("Verification code sent! Check your email.", {
163 |           description: "Check your inbox for the 6-digit code. If you don't see it, check spam
folder.",
164 |           duration: 10000,
165 |         });
166 |       }
167 |
168 |       setOtpSent(true);
169 |       setSendingCode(false);
170 |     } catch (err: any) {
171 |       console.error("'L Error sending OTP:", err);
172 |       setError(err.message || "Failed to send verification code. Please try again.");
173 |       setSendingCode(false);
174 |     }
175 |   };
176 |
177 |   // Handle verifying OTP
178 |   const handleVerifyOTP = async () => {
179 |     setLoading(true);
180 |     setError(null);
181 |
182 |     try {
183 |       if (!otpCode || otpCode.length !== 6) {
184 |         setError("Please enter the 6-digit verification code");
185 |         setLoading(false);
186 |         return;
187 |       }
188 |
189 |       console.log("Ø=Ý  Verifying OTP code...");
190 |
191 |       let isValid = false;
192 |       if (isSignUp) {
193 |         isValid = await verifyEmailCode(email, otpCode);
194 |       } else {
195 |         isValid = await verifyEmailOTPForSignIn(email, otpCode);
196 |       }
197 |
198 |       if (!isValid) {
199 |         setError("Invalid verification code. Please try again.");
200 |         setLoading(false);
201 |         return;
202 |       }
203 |
204 |       console.log("'  OTP verified successfully!");
205 |       setOtpVerified(true);
206 |
207 |       if (isSignUp) {
208 |         // For sign-up, user needs to set password next
```

```
209 |         toast.success("Email verified! Now create your password.");
210 |       } else {
211 |         // For sign-in, try to sign in
212 |         await handleSignInAfterOTP();
213 |       }
214 |
215 |       setLoading(false);
216 |     } catch (err: any) {
217 |       console.error("'L Error verifying OTP:", err);
218 |       setError(err.message || "Invalid verification code. Please try again.");
219 |       setLoading(false);
220 |     }
221 |   };
222 |
223 |   // Handle sign-in after OTP verification
224 |   const handleSignInAfterOTP = async () => {
225 |     try {
226 |       // Try to sign in with OTP
227 |       // Note: This will send an email link for sign-in (Firebase limitation)
228 |       await signInWithEmailOTP(email);
229 |       // Show success message - user needs to check email for sign-in link
230 |       toast.success("Please check your email!", {
231 |         description: "We've sent you a sign-in link. Click it to complete authentication.",
232 |         duration: 10000,
233 |       });
234 |       // Reset state so user can try again if needed
235 |       setOtpSent(false);
236 |       setOtpVerified(false);
237 |       setOtpCode("");
238 |     } catch (err: any) {
239 |       // If sign-in fails (e.g., user doesn't exist), show error
240 |       if (err.message.includes("No account found")) {
241 |         setError("No account found with this email. Please sign up instead.");
242 |         setIsSignUp(true);
243 |         setOtpVerified(false);
244 |         setOtpCode("");
245 |         setOtpSent(false);
246 |       } else if (err.message.includes("check your email")) {
247 |         // This is expected - user needs to check email for link
248 |         toast.info("Check your email!", {
249 |           description: "We've sent you a sign-in link. Click it to complete authentication.",
250 |           duration: 10000,
251 |         });
252 |         setOtpSent(false);
253 |         setOtpVerified(false);
254 |         setOtpCode("");
255 |       } else {
256 |         setError(err.message || "Failed to sign in. Please try again.");
257 |       }
258 |     }
259 |   };
260 |
261 |   // Handle resending OTP
262 |   const handleResendOTP = async () => {
263 |     setOtpCode("");
264 |     setOtpVerified(false);
265 |     await handleSendOTP();
266 |   };
267 |
268 |   // Handle creating account after OTP verification (for sign-up)
269 |   const handleCreateAccount = async () => {
270 |     setLoading(true);
271 |     setError(null);
272 |
273 |     try {
274 |       // Validate password
275 |       if (!password || password.length < 6) {
276 |         setError("Password must be at least 6 characters");
277 |         setLoading(false);
278 |         return;
279 |       }
```

```
280 |
281 |       console.log("Ø=Üç Creating account with verified email:", email);
282 |       const user = await signUpWithEmail(email, password, displayName);
283 |
284 |       console.log("'  Sign-up successful! User:", user.uid);
285 |
286 |       // Wait for auth state to update
287 |       const delay = isMobile() ? 2000 : 500;
288 |       await new Promise(resolve => setTimeout(resolve, delay));
289 |
290 |       if (!auth.currentUser) {
291 |         setLoading(false);
292 |         return;
293 |       }
294 |
295 |       // Check if user has completed profile setup
296 |       const { getUserData } = await import("@/services/authService");
297 |       const userData = await getUserData(user.uid);
298 |
299 |       hasRedirected.current = true;
300 |       if (userData && userData.activity) {
301 |         navigate("/", { replace: true });
302 |       } else {
303 |         navigate("/profile-setup", { replace: true });
304 |       }
305 |       setLoading(false);
306 |     } catch (err: any) {
307 |       console.error("'L Error signing up:", err);
308 |       setError(err.message || "Failed to create account. Please try again.");
309 |       setLoading(false);
310 |     }
311 |   };
312 |
313 |   const handleGoogleSignIn = async () => {
314 |     setLoading(true);
315 |     setError(null);
316 |
317 |     try {
318 |       console.log("Ø=Ý  Signing in with Google...");
319 |       const userData = await signInWithGoogle();
320 |
321 |       // If redirect was initiated, userData will be null
322 |       // The redirect result will be handled by useEffect
323 |       if (userData) {
324 |         // Popup method - user signed in immediately
325 |         console.log("'  Sign-in successful! User:", userData.uid);
326 |
327 |         // Wait for auth state to update
328 |         const delay = isMobile() ? 2000 : 500;
329 |         await new Promise(resolve => setTimeout(resolve, delay));
330 |
331 |         if (!auth.currentUser) {
332 |           setLoading(false);
333 |           return;
334 |         }
335 |
336 |         // Check if user has completed profile setup
337 |         const { getUserData } = await import("@/services/authService");
338 |         const userDataFromDb = await getUserData(auth.currentUser.uid);
339 |
340 |         hasRedirected.current = true;
341 |         if (userDataFromDb && userDataFromDb.activity) {
342 |           navigate("/", { replace: true });
343 |         } else {
344 |           navigate("/profile-setup", { replace: true });
345 |         }
346 |       } else {
347 |         // Redirect method - will be handled by useEffect
348 |         console.log("Ø=Üñ Redirect initiated, waiting for callback...");
349 |         toast.info("Redirecting to Google sign-in...");
350 |         // Don't set loading to false - let redirect happen
```

```
        return;
      }
      setLoading(false);
    } catch (err: any) {
      console.error("'L Error signing in with Google:", err);
      setError(err.message || "Failed to sign in with Google. Please try again.");
      setLoading(false);
    }
  };

  // Create reCAPTCHA container on mount
  useEffect(() => {
    // Ensure reCAPTCHA container exists and is properly set up
    const setupRecaptchaContainer = () => {
      if (typeof document === 'undefined' || !document.body) {
        return;
      }

      let container = document.getElementById('recaptcha-container');
      if (!container) {
        container = document.createElement('div');
        container.id = 'recaptcha-container';
        // Make it visible but tiny (required for reCAPTCHA to work)
        container.style.cssText = 'position: fixed; bottom: 0; right: 0; width: 1px; height:
1px; overflow: hidden; z-in';
        document.body.appendChild(container);
      }

      // Ensure it's in the DOM
      if (!container.parentElement && document.body) {
        document.body.appendChild(container);
      }
    };

    // Wait for DOM to be ready
    if (document.readyState === 'loading') {
      document.addEventListener('DOMContentLoaded', setupRecaptchaContainer);
    } else {
      setupRecaptchaContainer();
    }

    // Also try after a short delay to ensure everything is ready
    const timeout = setTimeout(setupRecaptchaContainer, 500);

    return () => {
      clearTimeout(timeout);
    };
  }, []);

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/10 via-background to-success/10
flex flex-col items-cent">
      {/* Animated background blobs */}
      <div className="absolute inset-0 overflow-hidden pointer-events-none">
        <motion.div
          animate={{
            x: [0, 100, 0],
            y: [0, -100, 0],
            scale: [1, 1.2, 1],
          }}
          transition={{ duration: 20, repeat: Infinity, ease: "linear" }}
          className="absolute top-0 left-0 w-72 h-72 bg-primary/5 rounded-full blur-3xl"
        />
        <motion.div
          animate={{
            x: [0, -100, 0],
            y: [0, 100, 0],
            scale: [1, 1.3, 1],
          }}
          transition={{ duration: 25, repeat: Infinity, ease: "linear" }}
          className="absolute bottom-0 right-0 w-96 h-96 bg-success/5 rounded-full blur-3xl"
        />
      </div>
```

```
422 |
423 |       <motion.div
424 |         initial={{ opacity: 0, y: 20 }}
425 |         animate={{ opacity: 1, y: 0 }}
426 |         transition={{ duration: 0.6, ease: "easeOut" }}
427 |         className="w-full max-w-md space-y-10 relative z-10"
428 |       >
429 |         {/* Logo and branding */}
430 |         <div className="text-center space-y-6">
431 |           <motion.div
432 |             initial={{ scale: 0.5, opacity: 0 }}
433 |             animate={{ scale: 1, opacity: 1 }}
434 |             transition={{
435 |               duration: 0.6,
436 |               delay: 0.2,
437 |               type: "spring",
438 |               stiffness: 200
439 |             }}
440 |             className="flex justify-center"
441 |           >
442 |             <div className="relative">
443 |               <motion.div
444 |                 animate={{
445 |                   boxShadow: [
446 |                     "0 0 20px rgba(25, 118, 210, 0.3)",
447 |                     "0 0 40px rgba(25, 118, 210, 0.5)",
448 |                     "0 0 20px rgba(25, 118, 210, 0.3)",
449 |                   ]
450 |                 }}
451 |                 transition={{ duration: 2, repeat: Infinity }}
452 |                 className="bg-gradient-to-br from-primary to-primary/80 rounded-full p-8 shadow-
elevation-4"
453 |               >
454 |                 <DirectionsRunIcon className="text-primary-foreground" style={{ fontSize: 56 }} /
>455 |               </motion.div>
456 |             </div>
457 |           </motion.div>
458 |
459 |           <motion.div
460 |             initial={{ opacity: 0, y: 10 }}
461 |             animate={{ opacity: 1, y: 0 }}
462 |             transition={{ duration: 0.5, delay: 0.4 }}
463 |           >
464 |             <h1 className="text-5xl font-extrabold bg-gradient-to-r from-primary via-success to-
primary bg-clip-text tex
                          FadeMatch
466 |             </h1>
467 |             <p className="text-lg text-muted-foreground mt-3 font-medium">
468 |               Connect with active people nearby
469 |             </p>
470 |           </motion.div>
471 |         </div>
472 |
473 |         {/* Error Alert */}
474 |         {error && (
475 |           <motion.div
476 |             initial={{ opacity: 0, y: -10 }}
477 |             animate={{ opacity: 1, y: 0 }}
478 |             className="w-full"
479 |           >
480 |             <div className="bg-destructive/10 border border-destructive/20 text-destructive px-4
py-3 rounded-lg text-sm
481 |               {error}
482 |             </div>
483 |           </motion.div>
484 |         )}
485 |
486 |         {/* Login Method Toggle */}
487 |         <motion.div
488 |           initial={{ opacity: 0, y: 20 }}
489 |           animate={{ opacity: 1, y: 0 }}
490 |           transition={{ duration: 0.5, delay: 0.6 }}
491 |           className="space-y-4"
492 |         >
```

```
          {/* Google Sign-In Button - First Option */}
          <Button
            onClick={handleGoogleSignIn}
            disabled={loading}
            className="w-full h-12 text-base font-semibold shadow-elevation-3 hover:shadow-elevation-4 transition-all du...
          >
            {loading ? (
              <>
                <div className="mr-3 h-5 w-5 border-2 border-gray-900 border-t-transparent rounded-full animate-spin" />
                Signing in...
              </>
            ) : (
              <>
                <svg className="mr-3 h-5 w-5" viewBox="0 0 24 24">
                  <path
                    fill="#4285F4"
                    d="M22.56 12.25c0-.78-.07-1.53-.2-2.25H12v4.26h5.92c-.26 1.37-1.04 2.53-2.21 3.31v2.77h3.57c2.08-1.9...
                  />
                  <path
                    fill="#34A853"
                    d="M12 23c2.97 0 5.46-.98 7.28-2.66l-3.57-2.77c-.98.66-2.23 1.06-3.71 1.06-2.86 0-5.29-1.93-6.16-4.5...
                  />
                  <path
                    fill="#FBBC05"
                    d="M5.84 14.09c-.22-.66-.35-1.36-.35-2.09s.13-1.43.35-2.09V7.07H2.18C1.43 8.55 1 10.22 1 12s.43 3.45...
                  />
                  <path
                    fill="#EA4335"
                    d="M12 5.38c1.62 0 3.06.56 4.21 1.64l3.15-3.15C17.45 2.09 14.97 1 12 1 7.7 1 3.99 3.47 2.18 7.07l3.6...
                  />
                </svg>
                Continue with Google
              </>
            )}
          </Button>

          {/* Divider */}
          <div className="relative my-4">
            <div className="absolute inset-0 flex items-center">
              <div className="w-full border-t border-muted"></div>
            </div>
            <div className="relative flex justify-center text-xs uppercase">
              <span className="bg-background px-2 text-muted-foreground">Or continue with email</span>
            </div>
          </div>

          {/* Email OTP Form */}
          {!otpSent ? (
            /* Step 1: Enter Email (and Name for sign-up) */
            <>
              {isSignUp && (
                <div className="space-y-2">
                  <label htmlFor="name" className="text-sm font-medium text-muted-foreground">
                    Full Name
                  </label>
                  <div className="relative">
                    <PersonIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-foreground" sty...
                    <Input
                      id="name"
                      type="text"
                      placeholder="John Doe"
                      value={displayName}
                      onChange={(e) => setDisplayName(e.target.value)}
                      className="pl-10 h-12 text-base"
                      disabled={loading || sendingCode}
                    />
                  </div>
                </div>
              )}

              <div className="space-y-2">
```

```
                <label htmlFor="email" className="text-sm font-medium text-muted-foreground">
                  Email Address
                </label>
                <div className="relative">
                  <EmailIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style={...
                    <Input
                      id="email"
                      type="email"
                      placeholder="your@email.com"
                      value={email}
                      onChange={(e) => setEmail(e.target.value)}
                      className="pl-10 h-12 text-base"
                      disabled={loading || sendingCode}
                      onKeyDown={(e) => {
                        if (e.key === 'Enter' && !loading && !sendingCode && email.trim() && (!
isSignUp || displayName.tri... handleSendOTP();
                        }
                      }}
                    />
                </div>
                <p className="text-xs text-muted-foreground">
                  We'll send a 6-digit verification code to this email
                </p>
              </div>

              <Button
                onClick={handleSendOTP}
                disabled={loading || sendingCode || !email.trim() || (isSignUp && !
displayName.trim())}
                className="w-full h-12 text-base font-semibold shadow-elevation-3 hover:shadow-
elevation-4 transition-al...
                {sendingCode ? (
                  <>
                    <div className="mr-3 h-5 w-5 border-2 border-primary-foreground border-t-
transparent rounded-full an... Sending Code...
                  </>
                ) : (
                  <>
                    <EmailIcon className="mr-2" style={{ fontSize: 20 }} />
                    Send Verification Code
                  </>
                )}
              </Button>

              <div className="text-center">
                <button
                  onClick={() => {
                    setIsSignUp(!isSignUp);
                    setError(null);
                    setEmail("");
                    setOtpCode("");
                    setOtpSent(false);
                    setOtpVerified(false);
                    setDisplayName("");
                  }}
                  className="text-sm text-muted-foreground hover:text-primary transition-colors"
                >
                  {isSignUp ? "Already have an account? Sign in" : "Don't have an account? Sign
up"}
                </button>
              </div>
            </>
          ) : !otpVerified ? (
            /* Step 2: Verify OTP Code */
            <>
              <div className="space-y-2">
                <label htmlFor="otp" className="text-sm font-medium text-muted-foreground">
                  Enter 6-Digit Verification Code
                </label>
                <Input
                  id="otp"
                  type="text"
                  placeholder="123456"
```

```
                  value={otpCode}
                  onChange={(e) => {
                    const value = e.target.value.replace(/\D/g, '').slice(0, 6);
                    setOtpCode(value);
                  }}
                  className="h-12 text-base text-center text-2xl tracking-widest font-mono"
                  disabled={loading}
                  maxLength={6}
                  autoFocus
                  onKeyDown={(e) => {
                    if (e.key === 'Enter' && !loading && otpCode.length === 6) {
                      handleVerifyOTP();
                    }
                  }}
                />
                <p className="text-xs text-muted-foreground text-center">
                  Check your email ({email}) for the verification code
                </p>
                <p className="text-xs text-muted-foreground/70 text-center italic mt-1">
                  Code expires in 10 minutes
                </p>
              </div>

              <Button
                onClick={handleVerifyOTP}
                disabled={loading || otpCode.length !== 6}
                className="w-full h-12 text-base font-semibold shadow-elevation-3 hover:shadow-
elevation-4 transition-al...
                {loading ? (
                  <>
                    <div className="mr-3 h-5 w-5 border-2 border-primary-foreground border-t-
transparent rounded-full an...
                    Verifying...
                  </>
                ) : (
                  "Verify Code"
                )}
              </Button>

              <Button
                onClick={handleResendOTP}
                variant="outline"
                disabled={sendingCode || loading}
                className="w-full h-10 text-sm"
              >
                {sendingCode ? "Sending..." : "Resend Code"}
              </Button>

              <Button
                onClick={() => {
                  setOtpSent(false);
                  setOtpCode("");
                  setError(null);
                }}
                variant="ghost"
                className="w-full h-10 text-sm"
              >
                Change Email
              </Button>
            </>
          ) : isSignUp ? (
            /* Step 3: Set Password (Sign-up only) */
            <>
              <div className="space-y-2">
                <label htmlFor="password" className="text-sm font-medium text-muted-foreground">
                  Create Password
                </label>
                <div className="relative">
                  <EmailIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-
muted-foreground" style=...
                  <Input
                    id="password"
                    type="password"
```

```
                    placeholder="At least 6 characters"
                    value={password}
                    onChange={(e) => setPassword(e.target.value)}
                    className="pl-10 h-12 text-base"
                    disabled={loading}
                    onKeyDown={(e) => {
                      if (e.key === 'Enter' && !loading && password.trim() && password.length >= 6) {
                        handleCreateAccount();
                      }
                    }}
                  />
                </div>
                <p className="text-xs text-muted-foreground">
                  Password must be at least 6 characters
                </p>
              </div>

              <Button
                onClick={handleCreateAccount}
                disabled={loading || !password.trim() || password.length < 6}
                className="w-full h-12 text-base font-semibold shadow-elevation-3 hover:shadow-elevation-4 transition-al...
                {loading ? (
                  <>
                    <div className="mr-3 h-5 w-5 border-2 border-primary-foreground border-t-transparent rounded-full an...
                    Creating Account...
                  </>
                ) : (
                  "Create Account"
                )}
              </Button>
            </>
          ) : null}

          <p className="text-xs text-center text-muted-foreground px-4 leading-relaxed">
            By continuing, you agree to our{" "}
            <button
              onClick={() => setShowTerms(true)}
              className="underline hover:text-primary transition-colors font-medium"
            >
              Terms of Service
            </button>{" "}
            and{" "}
            <button
              onClick={() => setShowPrivacy(true)}
              className="underline hover:text-primary transition-colors font-medium"
            >
              Privacy Policy
            </button>
          </p>
        </motion.div>

        {/* Terms of Service Dialog */}
        <Dialog open={showTerms} onOpenChange={setShowTerms}>
          <DialogContent className="max-w-lg max-h-[80vh] overflow-y-auto">
            <DialogHeader>
              <DialogTitle className="text-xl font-bold text-primary">
                Terms of Service
              </DialogTitle>
              <DialogDescription className="text-sm text-muted-foreground">
                Research Prototype Study Agreement
              </DialogDescription>
            </DialogHeader>
            <div className="space-y-4 text-sm text-foreground/90 leading-relaxed">
              <section>
                <h3 className="font-semibold text-primary mb-2">1. Research Study Overview</h3>
                <p>
                  <strong>PaceMatch</strong> is a venue-based fitness partner matching system developed as a
                  capstone research project for the Bachelor of Science in Information Technology degree at the
                  Institute of Information and Computing Technology.
                </p>
```

```
                <p className="mt-2">
                  <strong>Researchers:</strong> Roy Vincent R. Pertubal & Mark Anthony A. Mateo
                </p>
                <p>
                  <strong>Research Adviser:</strong> Engr. Klarence M. Baptista, MIT
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">2. Purpose of the Study</h3>
                <p>
                  This study aims to develop and evaluate a venue-based fitness partner matching
                  utilizes real-time GPS location sharing to connect Filipino fitness
                  compatible workout partners in Metro Manila. The system addresses the gap in
                  fitness applications by combining location-based services with fitness-
                  compatibility factors.
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">3. Prototype Nature</h3>
                <p>
                  This application is a <strong>research prototype</strong> developed for
                  It is not a commercial product and is intended solely for research validation
                  Features and functionality may be limited or subject to change as part of the
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">4. User Responsibilities</h3>
                <ul className="list-disc pl-5 space-y-1">
                  <li>Provide accurate profile information</li>
                  <li>Use the application responsibly and respectfully</li>
                  <li>Report any inappropriate behavior or technical issues</li>
                  <li>Respect other users' privacy and safety</li>
                  <li>Understand that meetups arranged through the app are at your own
                </ul>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">5. Disclaimer</h3>
                <p>
                  The researchers and institution are not liable for any incidents arising from
                  meetups arranged through this application. Users are advised to exercise
                  in public, well-lit areas. This is a research prototype and comes with no
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">6. Acceptance</h3>
                <p>
                  By creating an account and using PaceMatch, you acknowledge that you have
                  and agree to these terms. You also consent to participate in this research
                </p>
              </section>
            </div>
            <div className="mt-4 pt-4 border-t">
              <Button onClick={() => setShowTerms(false)} className="w-full">
                I Understand
              </Button>
            </div>
          </DialogContent>
        </Dialog>

        {/* Privacy Policy Dialog */}
        <Dialog open={showPrivacy} onOpenChange={setShowPrivacy}>
          <DialogContent className="max-w-lg max-h-[80vh] overflow-y-auto">
            <DialogHeader>
              <DialogTitle className="text-xl font-bold text-primary">
                Privacy Policy
              </DialogTitle>
```

system that
enthusiasts with
existing
specific

academic purposes.
and testing.
research process.

discretion

in-person
caution and meet
warranties.

read, understood,
study

```
              <DialogDescription className="text-sm text-muted-foreground">
                Data Privacy and Protection Notice
              </DialogDescription>
            </DialogHeader>
            <div className="space-y-4 text-sm text-foreground/90 leading-relaxed">
              <section>
                <h3 className="font-semibold text-primary mb-2">1. Data Privacy Compliance</h3>
                <p>
                  This research study complies with the <strong>Data Privacy Act of 2012
(Republic Act No. 10173)</strong> of the Philippines. Your personal data is collected and processed in
accordance with the
                  principles of transparency, legitimate purpose, and proportionality.
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">2. Data We Collect</h3>
                <ul className="list-disc pl-5 space-y-1">
                  <li><strong>Account Information:</strong> Email address, phone number, display
name</li>
                  <li><strong>Profile Data:</strong> Fitness activity preferences, fitness
level</li>
                  <li><strong>Location Data:</strong> Real-time GPS coordinates (when enabled)</
li>
                  <li><strong>Activity Data:</strong> Workout tracking information, distance,
duration</li>
                  <li><strong>Usage Data:</strong> App interactions for system improvement</li>
                </ul>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">3. How We Use Your Data</h3>
                <ul className="list-disc pl-5 space-y-1">
                  <li>To match you with compatible fitness partners nearby</li>
                  <li>To display your presence on the map (when visibility is enabled)</li>
                  <li>To facilitate communication between matched users</li>
                  <li>To validate research objectives and system functionality</li>
                  <li>For academic analysis and thesis documentation (anonymized)</li>
                </ul>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">4. Privacy Controls</h3>
                <p>
                  PaceMatch provides comprehensive privacy controls as required by Filipino
users' expectations:
                </p>
                <ul className="list-disc pl-5 space-y-1 mt-2">
                  <li><strong>Visibility Toggle:</strong> Control whether you appear on the map</
li>
                  <li><strong>Radius-Based Sharing:</strong> Limit who can see your location</li>
                  <li><strong>Activity Filtering:</strong> Choose what activities to display</li>
                  <li><strong>Ghost Mode:</strong> Hide your location entirely when needed</li>
                </ul>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">5. Data Storage & Security</h3>
                <p>
                  Your data is securely stored in Firebase Realtime Database with encryption in
transit.
                  Location data is processed in real-time and not permanently stored beyond the
current session.
                  Access to research data is limited to the authorized researchers.
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">6. Your Rights</h3>
                <p>Under the Data Privacy Act of 2012, you have the right to:</p>
                <ul className="list-disc pl-5 space-y-1 mt-2">
                  <li>Be informed about how your data is processed</li>
                  <li>Access your personal data</li>
                  <li>Object to data processing</li>
                  <li>Request correction or deletion of your data</li>
                  <li>Withdraw consent at any time by deleting your account</li>
                </ul>
              </section>

              <section>
```

```
                <h3 className="font-semibold text-primary mb-2">7. Research Use</h3>
                <p>
                  Aggregated and anonymized data may be used in the research thesis document and
academic
                  presentations. No personally identifiable information will be published or
shared outside
                  the research team.
                </p>
              </section>

              <section>
                <h3 className="font-semibold text-primary mb-2">8. Contact</h3>
                <p>
                  For questions about data privacy or to exercise your rights, please contact
the research
                  team through the Institute of Information and Computing Technology.
                </p>
              </section>
            </div>
            <div className="mt-4 pt-4 border-t">
              <Button onClick={() => setShowPrivacy(false)} className="w-full">
                I Understand
              </Button>
            </div>
          </DialogContent>
        </Dialog>

        {/* Features preview */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5, delay: 0.8 }}
          className="grid grid-cols-3 gap-6 pt-4"
        >
          {[
            { icon: PeopleIcon, label: "Find Partners", color: "primary", delay: 0 },
            { icon: DirectionsRunIcon, label: "Match Fitness", color: "success", delay: 0.1 },
            { icon: DirectionsRunIcon, label: "Train Together", color: "warning", delay: 0.2 },
          ].map((feature, i) => (
            <motion.div
              key={i}
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ duration: 0.4, delay: 0.9 + feature.delay }}
              className="text-center space-y-3"
            >
              <div className={`bg-${feature.color}/10 backdrop-blur-sm rounded-2xl p-4 mx-auto w-
fit border border-${fea
                <feature.icon className={`text-${feature.color}`} style={{ fontSize: 28 }} />
              </div>
              <p className="text-xs text-muted-foreground font-medium">{feature.label}</p>
            </motion.div>
          ))}
        </motion.div>
      </motion.div>
    </div>
  );
};

export default LoginScreen;
```

File: src/pages/MapScreen.tsx

Lines: 4199

```
   1 | // Main map screen with Waze-like proximity matching
   2 | import { useState, useCallback, useEffect, useRef, useMemo } from "react";
   3 | import { motion, AnimatePresence } from "framer-motion";
   4 | import "../styles/animations.css";
   5 | import {
   6 |   GoogleMap,
   7 |   Marker,
   8 |   InfoWindow,
   9 |   Polyline,
  10 |   Circle,
  11 |   useJsApiLoader
  12 | } from "@react-google-maps/api";
  13 | import {
  14 |   Box,
  15 |   Button as MuiButton,
  16 |   Typography,
  17 |   CircularProgress
  18 | } from "@mui/material";
  19 | import { Button } from "@/components/ui/button";
  20 | import { Avatar, AvatarImage, AvatarFallback } from "@/components/ui/avatar";
  21 | import { FitnessLevelAvatar } from "@/components/FitnessLevelAvatar";
  22 | import { Badge } from "@/components/ui/badge";
  23 | import {
  24 |   AlertDialog,
  25 |   AlertDialogAction,
  26 |   AlertDialogCancel,
  27 |   AlertDialogContent,
  28 |   AlertDialogDescription,
  29 |   AlertDialogFooter,
  30 |   AlertDialogHeader,
  31 |   AlertDialogTitle,
  32 | } from "@/components/ui/alert-dialog";
  33 | import { useNavigate, useLocation as useRouterLocation } from "react-router-dom";
  34 | import { useUser, Activity, RadiusPreference } from "@/contexts/UserContext";
  35 | import { useLocation } from "@/hooks/useLocation";
  36 | import { useNearbyUsers } from "@/hooks/useNearbyUsers";
  37 | import { useMatching } from "@/hooks/useMatching";
  38 | import { useAuth } from "@/hooks/useAuth";
  39 | import { useMovementDetection } from "@/hooks/useMovementDetection";
  40 | import { Geolocation } from "@capacitor/geolocation";
  41 | import { isNativePlatform } from "@/utils/platform";
  42 | import { formatDistance } from "@/utils/distance";
  43 | import { SearchFilter } from "@/services/matchingService";
  44 | import { getDisplayName } from "@/utils/anonymousName";
  45 | import { isWorkoutActive } from "@/utils/workoutState";
  46 | import { createMapIcon, createProfileIcon } from "@/utils/mapIcons";
  47 | import { preventMarkerOverlap, preventMarkerOverlapSimple, AdjustedPosition } from "@/utils/
markerOverlap";
  48 | import { getProfilePictureUrl } from "@/utils/profilePicture";
  49 | import { updateUserProfile } from "@/services/authService";
  50 | import FilterListIcon from "@mui/icons-material/FilterList";
  51 | import { updateUserVisibility, listenToAllUsers } from "@/services/locationService";
  52 | import { saveWorkout } from "@/services/workoutService";
  53 | import { listenToFriendRequests, removeFriend, sendFriendRequest } from "@/services/
friendService";
  54 | import { getUserConversations, markMessagesAsRead } from "@/services/messageService";
  55 | import { listenToPokes, sendPoke, acceptPoke, dismissPoke } from "@/services/pokeService";
  56 | import { reportUser, blockUser } from "@/services/userService";
  57 | import { PokeModal } from "@/components/PokeModal";
  58 | import PlayArrowIcon from "@mui/icons-material/PlayArrow";
  59 | import StopIcon from "@mui/icons-material/Stop";
  60 | import PauseIcon from "@mui/icons-material/Pause";
  61 | import PeopleIcon from "@mui/icons-material/People";
  62 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
  63 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
  64 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
  65 | import NotificationsIcon from "@mui/icons-material/Notifications";
  66 | import {
```

```
  MyLocation as MyLocationMui,
  CenterFocusStrong,
  Close,
  ViewComfy,
  ZoomIn,
  ZoomOut
} from "@mui/icons-material";
import { toast } from "sonner";
import { NotificationBanner } from "@/components/NotificationBanner";
import { MessageModal } from "@/components/MessageModal";
import { FriendRequestModal } from "@/components/FriendRequestModal";
import { ProfileView } from "@/pages/ProfileView";
import { InactivityWarningModal } from "@/components/InactivityWarningModal";
import { useNotificationContext } from "@/contexts/NotificationContext";
import { BadgeCounter } from "@/components/NotificationSystem";
import { NotificationBell } from "@/components/NotificationBell";
import { WorkoutSummaryModal } from "@/components/WorkoutSummaryModal";
import ChatBubbleIcon from "@mui/icons-material/ChatBubble";
import PersonAddIcon from "@mui/icons-material/PersonAdd";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import HourglassEmptyIcon from "@mui/icons-material/HourglassEmpty";
import TouchAppIcon from "@mui/icons-material/TouchApp";
import EmojiEventsIcon from "@mui/icons-material/EmojiEvents";
import MailIcon from "@mui/icons-material/Mail";
import BottomNavigation from "@/components/BottomNavigation";

type FriendStatus = "not_friends" | "request_pending" | "request_received" | "friends" |
"denied";
import SendIcon from "@mui/icons-material/Send";
import CloseIcon from "@mui/icons-material/Close";
import FitScreenIcon from "@mui/icons-material/FitScreen";

const libraries: ("places")[] = ["places"];

// Map container style
const mapContainerStyle = {
  width: "100%",
  height: "100vh"
};

// Default map center (will be overridden by user location)
const defaultCenter = {
  lat: 14.5995,
  lng: 120.9842 // Manila, Philippines
};

// Pokemon Go-style map styling for 3D mode
// Reduces POI clutter and enhances road visibility for better nearby user spotting
const pokemonGoMapStyle: google.maps.MapTypeStyle[] = [
  {
    featureType: "poi",
    elementType: "all",
    stylers: [{ visibility: "off" }]
  },
  {
    featureType: "poi.business",
    elementType: "all",
    stylers: [{ visibility: "off" }]
  },
  {
    featureType: "poi.park",
    elementType: "all",
    stylers: [{ visibility: "simplified" }]
  },
  {
    featureType: "road",
    elementType: "geometry",
    stylers: [{ visibility: "on" }, { saturation: -20 }]
  },
  {
    featureType: "road",
    elementType: "labels",
```

```
    stylers: [{ visibility: "simplified" }]
  },
  {
    featureType: "transit",
    elementType: "all",
    stylers: [{ visibility: "off" }]
  },
  {
    featureType: "water",
    elementType: "geometry",
    stylers: [{ color: "#a0d0ff" }, { saturation: 20 }]
  },
  {
    featureType: "landscape",
    elementType: "geometry",
    stylers: [{ saturation: -30 }, { lightness: 10 }]
  }
];

// Marker colors based on activity
const getMarkerColor = (activity?: string | null): string => {
  switch (activity) {
    case "running":
      return "Ø=ßâ"; // Green for running
    case "cycling":
      return "Ø=Ý5"; // Blue for cycling
    case "walking":
      return "Ø=ßá"; // Yellow for walking
    default:
      return "&ª"; // White/gray for unknown
  }
};

/**
 * Calculate zoom level from distance in meters
 * Aligned with research recommendations: Cycling (12-14), Running (14-16), Walking (16-18)
 */
const calculateZoomFromMeters = (distanceMeters: number): number => {
  if (distanceMeters >= 50000) return 10; // 50km
  if (distanceMeters >= 20000) return 11; // 20km
  if (distanceMeters >= 10000) return 12; // 10km
  if (distanceMeters >= 5000) return 13; // 5km
  if (distanceMeters >= 2000) return 14; // 2km
  if (distanceMeters >= 1000) return 15; // 1km
  if (distanceMeters >= 500) return 16; // 500m
  if (distanceMeters >= 200) return 17; // 200m - walking close zoom
  if (distanceMeters >= 150) return 18; // 150m
  if (distanceMeters >= 100) return 18.5; // 100m
  if (distanceMeters >= 50) return 19; // 50m
  return 19.5; // Very close (25m or less)
};

interface Location {
  lat: number;
  lng: number;
}

interface LocationHistoryPoint {
  lat: number;
  lng: number;
}

interface UserTrails {
  [userId: string]: LocationHistoryPoint[];
}

interface MapRef {
  panTo: (center: { lat: number; lng: number }) => void;
  setZoom: (zoom: number) => void;
  setTilt: (tilt: number) => void;
  setHeading: (heading: number) => void;
```

```
  setCenter: (center: { lat: number; lng: number }) => void;
  setOptions: (options: any) => void;
}

/**
 * MapScreen Component
 * Main screen showing map with user location and nearby users
 */
const MapScreen = () => {
  const navigate = useNavigate();
  const routerLocation = useRouterLocation();
  const { user } = useAuth();
  const { userProfile, useMetric, addWorkout, setUserProfile } = useUser();
  const { addNotification, unreadMessageCount, unreadFriendRequestCount, notifications,
unreadCount, dismissNotification...

  // Get focusFriend from navigation state (when coming from Index page)
  const focusFriend = routerLocation.state?.focusFriend;

  // Google Maps state
  // Visibility: false in Beacon Mode (workout inactive), true in Active Workout (workout active)
  const [visible, setVisible] = useState(false); // Start hidden in Beacon Mode
  const [mapCenter, setMapCenter] = useState(defaultCenter);
  const [mapZoom, setMapZoom] = useState(15);
  const [mapRef, setMapRef] = useState<MapRef | null>(null);
  const [locationHistory, setLocationHistory] = useState<LocationHistoryPoint[]>([]);
  const [userTrails, setUserTrails] = useState<UserTrails>({});
  const [mapTilt, setMapTilt] = useState(0);
  const [mapHeading, setMapHeading] = useState(0);
  const [isWazeMode, setIsWazeMode] = useState(false);
  const [userHeading, setUserHeading] = useState<number | null>(null);
  const [isNavigationStyle, setIsNavigationStyle] = useState(false);
  const [viewDistanceMeters, setViewDistanceMeters] = useState(150);
  const [showViewDistanceControl, setShowViewDistanceControl] = useState(false);
  const [zoomLevel, setZoomLevel] = useState<"close" | "medium" | "far">("medium");

  // Workout tracking state
  const [isActive, setIsActive] = useState(false);
  const [isPaused, setIsPaused] = useState(false);
  const [showSidebar, setShowSidebar] = useState(false); // Sidebar visibility toggle - starts
closed
  const [lastViewedMatchesCount, setLastViewedMatchesCount] = useState(0); // Track viewed
matches for notification
  const hasAutoOpenedRef = useRef(false); // Track if sidebar was auto-opened to prevent re-
opening
  const [showStopConfirmation, setShowStopConfirmation] = useState(false); // Confirmation
dialog for stopping session
  const [showMatchesDrawer, setShowMatchesDrawer] = useState(false); // Matches drawer
visibility toggle
  const [pointsTracked, setPointsTracked] = useState(0);
  const [distance, setDistance] = useState(0);
  const [showNotification, setShowNotification] = useState(false);
  const [elapsedTime, setElapsedTime] = useState(0); // in seconds
  const [currentSpeed, setCurrentSpeed] = useState(0); // km/h
  const [avgSpeed, setAvgSpeed] = useState(0); // km/h
  const [startTime, setStartTime] = useState<Date | null>(null);
  const [pauseStartTime, setPauseStartTime] = useState<Date | null>(null); // Track when pause
started
  const [totalPausedTime, setTotalPausedTime] = useState(0); // Total paused duration in seconds
  const [showSummary, setShowSummary] = useState(false);
  const [showSpeedNotPace, setShowSpeedNotPace] = useState(true);
  const [showNotificationDrawer, setShowNotificationDrawer] = useState(false);
  const [showInactivityWarning, setShowInactivityWarning] = useState(false);
  const [pausedDueToInactivity, setPausedDueToInactivity] = useState(false);
  const lastDistanceRef = useRef(0);
  const lastTimeRef = useRef(0);
  const intervalRef = useRef<NodeJS.Timeout | null>(null);
  const lastLocationRef = useRef<Location | null>(null);
  const processedUserIdsRef = useRef<Set<string>>(new Set());
  const nearbyUsersRef = useRef<any[]>([]);
  const notificationShownRef = useRef(false); // Track if notification was shown in this session
  const notificationCreatedRef = useRef(false); // Track if notification was added to bell in
this session

  // Get user's enabled activities from profile
  // Handle both 'activities' (array) and 'activity' (single value) for backward compatibility
  const userActivities: Activity[] = userProfile?.activities?.length > 0
    ? userProfile.activities
    : (userProfile as any)?.activity
```

```
      ? [(userProfile as any).activity as Activity]
      : (["running", "cycling", "walking"] as Activity[]);
  const [selectedActivity, setSelectedActivity] = useState<"running" | "cycling" | "walking">(
    userActivities[0] as "running" | "cycling" | "walking"
  );

  // Track previous userActivities to detect changes
  const prevUserActivitiesRef = useRef<Activity[]>(userActivities);

  // Update selectedActivity when user profile activities change
  useEffect(() => {
    if (userActivities && userActivities.length > 0) {
      const prevActivities = prevUserActivitiesRef.current;
      const activitiesChanged = JSON.stringify(prevActivities) !==
JSON.stringify(userActivities);
      if (activitiesChanged) {
        // If current selectedActivity is not in new userActivities, switch to first available
        if (!userActivities.includes(selectedActivity)) {
          setSelectedActivity(userActivities[0] as "running" | "cycling" | "walking");
        }
        // Update ref for next comparison
        prevUserActivitiesRef.current = userActivities;
      }
    }
  }, [userActivities, selectedActivity]);

  const [selectedUser, setSelectedUser] = useState<any>(null);
  const [showMessageModal, setShowMessageModal] = useState(false);
  const [showFriendRequestModal, setShowFriendRequestModal] = useState(false);
  const [showProfileView, setShowProfileView] = useState(false);
  const [showFilterModal, setShowFilterModal] = useState(false);
  const [showPokeModal, setShowPokeModal] = useState(false);
  const [searchFilter, setSearchFilter] = useState<SearchFilter>("all");
  const [declinedUsers, setDeclinedUsers] = useState<Record<string, number>>({}); // userId ->
cooldownUntilTimestamp
  const [visibleToFriendsOnly, setVisibleToFriendsOnly] = useState(false);

  // Poke state
  const [pokes, setPokes] = useState<string[]>([]); // Array of user IDs who poked the current
user (for notifications)
  const [hasPokedUsers, setHasPokedUsers] = useState<Record<string, boolean>>({}); // Track
which users we've poked
  const [workoutPokes, setWorkoutPokes] = useState<string[]>([]); // Pokes received during
current workout session
  const [workoutNearbyUsers, setWorkoutNearbyUsers] = useState<any[]>([]); // Nearby users
during workout
  const notifiedPokesRef = useRef<Set<string>>(new Set()); // Track which pokes we've already
notified

  // Activity filter for People Sidebar
  const [activityFilter, setActivityFilter] = useState<"all" | "running" | "cycling" |
"walking">("all");

  // Friend requests and messages for sidebar
  const [friendRequests, setFriendRequests] = useState<{ incoming: string[]; outgoing: string[] }
>({ incoming: [], outgo
  const [messageRequests, setMessageRequests] = useState<any[]>([]);
  const [incomingMessages, setIncomingMessages] = useState<any[]>([]); // New messages from
friends
  const [friendRequestUsers, setFriendRequestUsers] = useState<Record<string, any>>({});
  const [messageRequestUsers, setMessageRequestUsers] = useState<Record<string, any>>({});
  const [incomingMessageUsers, setIncomingMessageUsers] = useState<Record<string, any>>({});
  const [loadingFriendRequestUsers, setLoadingFriendRequestUsers] = useState(false);
  const [loadingMessageRequestUsers, setLoadingMessageRequestUsers] = useState(false);
  const [loadingIncomingMessageUsers, setLoadingIncomingMessageUsers] = useState(false);

  // Load Google Maps API
  const googleMapsApiKey = import.meta.env.VITE_GOOGLE_MAPS_API_KEY;
  const { isLoaded, loadError } = useJsApiLoader({
    id: 'google-map-script',
    googleMapsApiKey: googleMapsApiKey || '',
    libraries: libraries
  });

  // Get initial location on web to center the map (one-time request, not continuous tracking)
  const [initialLocation, setInitialLocation] = useState<Location | null>(null);
  const initialLocationFetchedRef = useRef(false);

  // Get user's current location
  // Beacon Mode: No continuous GPS tracking, but get initial location for map display (user is
hidden)
```

```
351 |   // Active Workout: Continuous GPS tracking enabled (user is visible)
352 |   const { location, error: locationError, isGettingLocation, stopTracking } = useLocation(
353 |     user?.uid || null,
354 |     isActive, // Only track location continuously when workout is active (not in Beacon Mode)
355 |     visible
356 |   );
357 |
358 |   // Use location for nearby users calculation (location in active workout, initialLocation in
beacon mode)
359 |   const effectiveLocation = location || initialLocation;
360 |
361 |   useEffect(() => {
362 |     // Only get initial location on web (not native, as native uses useLocation hook)
363 |     if (isNativePlatform() || !user?.uid || initialLocationFetchedRef.current) {
364 |       return;
365 |     }
366 |
367 |     // Get one-time location for map centering
368 |     if (navigator.geolocation) {
369 |       console.log("Ø=ÜÍ Requesting initial location for map centering...");
370 |       navigator.geolocation.getCurrentPosition(
371 |         (position) => {
372 |           const loc = {
373 |             lat: position.coords.latitude,
374 |             lng: position.coords.longitude,
375 |           };
376 |           setInitialLocation(loc);
377 |           initialLocationFetchedRef.current = true;
378 |           console.log("'  Initial location obtained:", loc);
379 |
380 |           // Center and zoom map on user's location (Pokemon Go style)
381 |           if (mapRef && isLoaded && window.google) {
382 |             const zoomLevel = 17; // Close zoom level for Pokemon Go-like experience
383 |             setMapCenter(loc);
384 |             setMapZoom(zoomLevel);
385 |             mapRef.setCenter(loc);
386 |             mapRef.panTo(loc);
387 |             mapRef.setZoom(zoomLevel);
388 |             console.log("'  Map centered and zoomed to user location");
389 |           } else {
390 |             // If map isn't ready yet, set center and zoom state
391 |             setMapCenter(loc);
392 |             setMapZoom(17);
393 |           }
394 |         },
395 |         (err) => {
396 |           console.warn("& þ  Could not get initial location:", err.message);
397 |           initialLocationFetchedRef.current = true; // Don't retry
398 |         },
399 |         {
400 |           enableHighAccuracy: false, // Use less accurate but faster for initial location
401 |           timeout: 5000,
402 |           maximumAge: 60000 // Accept cached location up to 1 minute old
403 |         }
404 |       );
405 |     }
406 |   }, [user?.uid, mapRef, isLoaded]);
407 |
408 |   // Request location permissions when MapScreen loads (on native platforms)
409 |   useEffect(() => {
410 |     const requestLocationPermissions = async () => {
411 |       // Only request on native platforms (iOS/Android)
412 |       if (!isNativePlatform() || !user?.uid) {
413 |         return;
414 |       }
415 |
416 |       try {
417 |         console.log("Ø=ÜÍ Requesting location permissions...");
418 |         const permissionStatus = await Geolocation.requestPermissions();
419 |
420 |         if (permissionStatus.location === 'granted') {
421 |           console.log("'  Location permission granted");
```

```
422 |         } else if (permissionStatus.location === 'denied') {
423 |           console.warn("& þ  Location permission denied");
424 |         } else {
425 |           console.warn("& þ  Location permission:", permissionStatus.location);
426 |         }
427 |       } catch (error) {
428 |         console.error("'L Error requesting location permissions:", error);
429 |       }
430 |     };
431 |
432 |     // Request permissions when component mounts
433 |     requestLocationPermissions();
434 |   }, [user?.uid]); // Request when user is available
435 |
436 |   // Sync visibility with workout state: Hidden in Beacon Mode, Visible in Active Workout
437 |   // Also ensure GPS tracking is fully stopped whenever workout is inactive
438 |   useEffect(() => {
439 |     if (!isActive) {
440 |       // Beacon Mode: User is hidden, stop GPS tracking
441 |       setVisible(false);
442 |       stopTracking();
443 |
444 |       // Clear location from Firebase when entering Beacon Mode
445 |       if (user?.uid) {
446 |         import("@/services/locationService").then(({ updateUserVisibility }) => {
447 |           updateUserVisibility(user.uid, false).catch((error) => {
448 |             console.error("Error updating visibility:", error);
449 |           });
450 |         });
451 |       }
452 |     } else {
453 |       // Active Workout: User is visible, GPS tracking enabled
454 |       setVisible(true);
455 |
456 |       // Update visibility in Firebase when starting workout
457 |       if (user?.uid) {
458 |         import("@/services/locationService").then(({ updateUserVisibility }) => {
459 |           updateUserVisibility(user.uid, true).catch((error) => {
460 |             console.error("Error updating visibility:", error);
461 |           });
462 |         });
463 |       }
464 |     }
465 |
466 |     // Show error messages to user when workout is active but GPS fails
467 |     if (isActive && locationError) {
468 |       console.error("& þ  Location error detected:", locationError);
469 |       toast.error(`GPS Error: ${locationError}`, {
470 |         duration: 5000,
471 |       });
472 |     }
473 |   }, [isActive, stopTracking, locationError, user?.uid]);
474 |
475 |   // Movement detection for inactivity warning
476 |   const handleStationaryDetected = useCallback(() => {
477 |     // Only show warning if workout is active and not already paused
478 |     if (isActive && !isPaused) {
479 |       setShowInactivityWarning(true);
480 |     }
481 |   }, [isActive, isPaused]);
482 |
483 |   const handleMovementDetected = useCallback(() => {
484 |     // If paused due to inactivity and movement detected, show resume prompt
485 |     if (pausedDueToInactivity && isPaused) {
486 |       toast.success("Movement detected! Tap Resume to continue your workout.", {
487 |         duration: 5000,
488 |       });
489 |     }
490 |     // Reset inactivity flag
491 |     setPausedDueToInactivity(false);
492 |   }, [pausedDueToInactivity, isPaused]);
```

```
493 |
494 |   useMovementDetection({
495 |     location: location ? { lat: location.lat, lng: location.lng } : null,
496 |     isActive,
497 |     isPaused,
498 |     onStationary: handleStationaryDetected,
499 |     onMovementDetected: handleMovementDetected,
500 |     distanceThreshold: 10, // 10 meters
501 |     detectionWindowMinutes: 5, // 5 minutes
502 |   });
503 |
504 |   // Calculate bearing (direction) between two points
505 |   const calculateBearing = (lat1: number, lng1: number, lat2: number, lng2: number): number => {
506 |     const dLng = (lng2 - lng1) * Math.PI / 180;
507 |     const lat1Rad = lat1 * Math.PI / 180;
508 |     const lat2Rad = lat2 * Math.PI / 180;
509 |
510 |     const y = Math.sin(dLng) * Math.cos(lat2Rad);
511 |     const x = Math.cos(lat1Rad) * Math.sin(lat2Rad) -
512 |               Math.sin(lat1Rad) * Math.cos(lat2Rad) * Math.cos(dLng);
513 |
514 |     const bearing = Math.atan2(y, x) * 180 / Math.PI;
515 |     return (bearing + 360) % 360; // Normalize to 0-360
516 |   };
517 |
518 |   // Calculate camera center offset to position user at bottom of screen
519 |   const calculateCameraOffset = (userLat: number, userLng: number, heading: number,
offsetDistance: number = 0.002): Loc
520 |     const headingRad = (heading || 0) * Math.PI / 180;
521 |     const offsetLat = userLat + offsetDistance * Math.cos(headingRad);
522 |     const offsetLng = userLng + offsetDistance * Math.sin(headingRad);
523 |     return { lat: offsetLat, lng: offsetLng };
524 |   };
525 |
526 |   // Load searchFilter and friendsOnly visibility from user profile
527 |   // Note: Visibility is controlled by workout state (isActive), not loaded from profile
528 |   useEffect(() => {
529 |     if (user) {
530 |       import("@/services/authService").then(({ getUserData }) => {
531 |         getUserData(user.uid).then((userData) => {
532 |           if (userData?.searchFilter) {
533 |             setSearchFilter(userData.searchFilter);
534 |           }
535 |           if (userData?.visibleToFriendsOnly !== undefined) {
536 |             setVisibleToFriendsOnly(userData.visibleToFriendsOnly);
537 |           }
538 |           // Visibility is controlled by workout state (isActive), not loaded from profile
539 |           // This ensures Beacon Mode = hidden, Active Workout = visible
540 |         });
541 |       });
542 |     } else if (userProfile && (userProfile as any).searchFilter) {
543 |       setSearchFilter((userProfile as any).searchFilter);
544 |     }
545 |   }, [user, userProfile]);
546 |
547 |   // Note: Visibility is now controlled by workout state (isActive), not by Firebase listener
548 |   // This ensures Beacon Mode = hidden, Active Workout = visible
549 |   // Removed Firebase listener for visibility to prevent conflicts with workout state
550 |
551 |   // Load declined users from localStorage
552 |   useEffect(() => {
553 |     const stored = localStorage.getItem("declinedUsers");
554 |     if (stored) {
555 |       try {
556 |         const parsed = JSON.parse(stored);
557 |         // Filter out expired cooldowns
558 |         const now = Date.now();
559 |         const active = Object.entries(parsed).reduce((acc, [userId, cooldownUntil]) => {
560 |           if (typeof cooldownUntil === "number" && cooldownUntil > now) {
561 |             acc[userId] = cooldownUntil;
562 |           }
563 |           return acc;
```

```
564 |         }, {} as Record<string, number>);
565 |         setDeclinedUsers(active);
566 |         localStorage.setItem("declinedUsers", JSON.stringify(active));
567 |       } catch (e) {
568 |         console.error("Error loading declined users:", e);
569 |       }
570 |     }
571 |   }, []);
572 |
573 |   // Restore activity state from localStorage on mount
574 |   useEffect(() => {
575 |     const stored = localStorage.getItem("activityState");
576 |     if (stored) {
577 |       try {
578 |         const parsed = JSON.parse(stored);
579 |         if (parsed.isActive) {
580 |           setIsActive(true);
581 |           setIsPaused(parsed.isPaused || false);
582 |           setSelectedActivity(parsed.selectedActivity || selectedActivity);
583 |           setDistance(parsed.distance || 0);
584 |           setCurrentSpeed(parsed.currentSpeed || 0);
585 |           setAvgSpeed(parsed.avgSpeed || 0);
586 |           setPointsTracked(parsed.pointsTracked || 0);
587 |
588 |           // Restore startTime and calculate elapsed time
589 |           if (parsed.startTime) {
590 |             const savedStartTime = new Date(parsed.startTime);
591 |             setStartTime(savedStartTime);
592 |             setTotalPausedTime(parsed.totalPausedTime || 0);
593 |             if (parsed.isPaused && parsed.pauseStartTime) {
594 |               setPauseStartTime(new Date(parsed.pauseStartTime));
595 |             }
596 |             // Calculate elapsed time based on saved start time
597 |             const now = new Date();
598 |             const totalElapsed = Math.floor((now.getTime() - savedStartTime.getTime()) / 1000);
599 |             // If paused, use saved elapsed time, otherwise calculate from start time minus
paused time
600 |             setElapsedTime(parsed.isPaused ? (parsed.elapsedTime || 0) : (totalElapsed -
(parsed.totalPausedTime || 0)));...
601 |           }
602 |
603 |           console.log("'  Restored activity state from localStorage");
604 |         }
605 |       } catch (e) {
606 |         console.error("Error loading activity state:", e);
607 |         // Clear corrupted data
608 |         localStorage.removeItem("activityState");
609 |       }
610 |     }
611 |   }, []); // Only run on mount
612 |
613 |   // Save activity state to localStorage whenever it changes
614 |   useEffect(() => {
615 |     if (isActive) {
616 |       const stateToSave = {
617 |         isActive,
618 |         isPaused,
619 |         startTime: startTime?.toISOString() || null,
620 |         elapsedTime,
621 |         totalPausedTime,
622 |         pauseStartTime: pauseStartTime?.toISOString() || null,
623 |         distance,
624 |         currentSpeed,
625 |         avgSpeed,
626 |         pointsTracked,
627 |         selectedActivity
628 |       };
629 |       localStorage.setItem("activityState", JSON.stringify(stateToSave));
630 |     } else {
631 |       // Clear saved state when activity is stopped
632 |       localStorage.removeItem("activityState");
633 |     }
634 |   }, [isActive, isPaused, startTime, elapsedTime, distance, currentSpeed, avgSpeed,
pointsTracked, selectedActivity]);
```

```
635 |
636 |   // Listen to friend requests
637 |   useEffect(() => {
638 |     if (!user?.uid) return;
639 |
640 |     const unsubscribe = listenToFriendRequests(user.uid, (requests) => {
641 |       setFriendRequests(requests);
642 |     });
643 |
644 |     return () => unsubscribe();
645 |   }, [user?.uid]);
646 |
647 |   // Restore poke notifications history so we don't alert repeatedly after reload
648 |   useEffect(() => {
649 |     try {
650 |       const stored = localStorage.getItem("notifiedPokes");
651 |       if (stored) {
652 |         const parsed = JSON.parse(stored);
653 |         if (Array.isArray(parsed)) {
654 |           notifiedPokesRef.current = new Set(parsed);
655 |         }
656 |       }
657 |     } catch (error) {
658 |       console.error("Error restoring notified pokes:", error);
659 |     }
660 |   }, []);
661 |
662 |   const persistNotifiedPokes = () => {
663 |     try {
664 |       localStorage.setItem(
665 |         "notifiedPokes",
666 |         JSON.stringify(Array.from(notifiedPokesRef.current))
667 |       );
668 |     } catch (error) {
669 |       console.error("Error persisting notified pokes:", error);
670 |     }
671 |   };
672 |
673 |   // Listen to pokes
674 |   useEffect(() => {
675 |     if (!user?.uid) return;
676 |
677 |     const unsubscribe = listenToPokes(user.uid, (pokeUserIds) => {
678 |       setPokes(pokeUserIds);
679 |
680 |       const pokeSet = new Set(pokeUserIds);
681 |       let removed = false;
682 |       notifiedPokesRef.current.forEach((id) => {
683 |         if (!pokeSet.has(id)) {
684 |           notifiedPokesRef.current.delete(id);
685 |           removed = true;
686 |         }
687 |       });
688 |       if (removed) {
689 |         persistNotifiedPokes();
690 |       }
691 |
692 |       // Track pokes received during workout (for summary)
693 |       if (pokeUserIds.length > 0) {
694 |         const newPokeUserIds = pokeUserIds.filter(
695 |           (id) => !notifiedPokesRef.current.has(id)
696 |         );
697 |
698 |         // Track pokes received during workout (for summary) but only add new ones
699 |         if (isActive) {
700 |           setWorkoutPokes((prev) => {
701 |             const newPokes = newPokeUserIds.filter((id) => !prev.includes(id));
702 |             return newPokes.length > 0 ? [...prev, ...newPokes] : prev;
703 |           });
704 |         }
705 |
```

```
706 |         // Note: Poke notifications are now created in pokeService when poke is sent
707 |         // No need to create notifications here to avoid duplicates
708 |         if (newPokeUserIds.length > 0) {
709 |           newPokeUserIds.forEach((pokeUserId) => {
710 |             notifiedPokesRef.current.add(pokeUserId);
711 |           });
712 |           persistNotifiedPokes();
713 |         }
714 |       }
715 |     });
716 |
717 |     return () => unsubscribe();
718 |   }, [user?.uid, isActive, addNotification]);
719 |
720 |   // Fetch user data for friend requests
721 |   useEffect(() => {
722 |     if (friendRequests.incoming.length === 0) {
723 |       setFriendRequestUsers({});
724 |       return;
725 |     }
726 |
727 |     const fetchFriendRequestUsers = async () => {
728 |       setLoadingFriendRequestUsers(true);
729 |       try {
730 |         const { getUserData } = await import("@/services/authService");
731 |         const userDataPromises = friendRequests.incoming.map(async (userId) => {
732 |           try {
733 |             const userData = await getUserData(userId);
734 |             return { userId, userData };
735 |           } catch (error) {
736 |             console.error(`Error fetching user data for ${userId}:`, error);
737 |             return { userId, userData: null };
738 |           }
739 |         });
740 |
741 |         const results = await Promise.all(userDataPromises);
742 |         const usersMap: Record<string, any> = {};
743 |         results.forEach(({ userId, userData }) => {
744 |           if (userData) {
745 |             usersMap[userId] = userData;
746 |           }
747 |         });
748 |         setFriendRequestUsers(usersMap);
749 |       } catch (error) {
750 |         console.error("Error fetching friend request users:", error);
751 |       } finally {
752 |         setLoadingFriendRequestUsers(false);
753 |       }
754 |     };
755 |
756 |     fetchFriendRequestUsers();
757 |   }, [friendRequests.incoming]);
758 |
759 |   // Load message requests (conversations with unread messages from non-friends)
760 |   useEffect(() => {
761 |     if (!user?.uid) return;
762 |
763 |     const loadMessageRequests = async () => {
764 |       try {
765 |         const conversations = await getUserConversations(user.uid);
766 |         // Filter for conversations with unread messages from non-friends
767 |         const friends = userProfile?.friends || [];
768 |         const requests = conversations.filter(conv =>
769 |           conv.unreadCount > 0 && !friends.some((f: any) => String(f) === conv.otherUserId)
770 |         );
771 |         setMessageRequests(requests);
772 |       } catch (error) {
773 |         console.error("Error loading message requests:", error);
774 |       }
775 |     };
776 |
```

```
777 |     loadMessageRequests();
778 |     // Refresh every 30 seconds
779 |     const interval = setInterval(loadMessageRequests, 30000);
780 |     return () => clearInterval(interval);
781 |   }, [user?.uid, userProfile?.friends]);
782 |
783 |   // Load incoming messages (conversations with unread messages from friends)
784 |   useEffect(() => {
785 |     if (!user?.uid) return;
786 |
787 |     const loadIncomingMessages = async () => {
788 |       try {
789 |         const conversations = await getUserConversations(user.uid);
790 |         // Filter for conversations with unread messages from friends
791 |         const friends = userProfile?.friends || [];
792 |         const messages = conversations.filter(conv =>
793 |           conv.unreadCount > 0 && friends.some((f: any) => String(f) === conv.otherUserId)
794 |         );
795 |         // Sort by last message time (newest first)
796 |         messages.sort((a, b) => b.lastMessageTime - a.lastMessageTime);
797 |         setIncomingMessages(messages);
798 |       } catch (error) {
799 |         console.error("Error loading incoming messages:", error);
800 |       }
801 |     };
802 |
803 |     loadIncomingMessages();
804 |     // Refresh every 30 seconds
805 |     const interval = setInterval(loadIncomingMessages, 30000);
806 |     return () => clearInterval(interval);
807 |   }, [user?.uid, userProfile?.friends]);
808 |
809 |   // Auto-open sidebar only if there are new friend requests, message requests, or incoming
messages
810 |   useEffect(() => {
811 |     const hasNewRequests = friendRequests.incoming.length > 0 || messageRequests.length > 0 ||
incomingMessages.length > 0;
812 |     if (hasNewRequests && !hasAutoOpenedRef.current) {
813 |       setShowSidebar(true);
814 |       hasAutoOpenedRef.current = true;
815 |     }
816 |   }, [friendRequests.incoming.length, messageRequests.length, incomingMessages.length]);
817 |
818 |   // Fetch user data for message requests
819 |   useEffect(() => {
820 |     if (messageRequests.length === 0) {
821 |       setMessageRequestUsers({});
822 |       return;
823 |     }
824 |
825 |     const fetchMessageRequestUsers = async () => {
826 |       setLoadingMessageRequestUsers(true);
827 |       try {
828 |         const { getUserData } = await import("@/services/authService");
829 |         const uniqueUserIds = [...new Set(messageRequests.map(conv => conv.otherUserId))];
830 |         const userDataPromises = uniqueUserIds.map(async (userId) => {
831 |           try {
832 |             const userData = await getUserData(userId);
833 |             return { userId, userData };
834 |           } catch (error) {
835 |             console.error(`Error fetching user data for ${userId}:`, error);
836 |             return { userId, userData: null };
837 |           }
838 |         });
839 |
840 |         const results = await Promise.all(userDataPromises);
841 |         const usersMap: Record<string, any> = {};
842 |         results.forEach(({ userId, userData }) => {
843 |           if (userData) {
844 |             usersMap[userId] = userData;
845 |           }
846 |         });
847 |         setMessageRequestUsers(usersMap);
```

```
848 |       } catch (error) {
849 |         console.error("Error fetching message request users:", error);
850 |       } finally {
851 |         setLoadingMessageRequestUsers(false);
852 |       }
853 |     };
854 |
855 |     fetchMessageRequestUsers();
856 |   }, [messageRequests]);
857 |
858 |   // Fetch user data for incoming messages
859 |   useEffect(() => {
860 |     if (incomingMessages.length === 0) {
861 |       setIncomingMessageUsers({});
862 |       return;
863 |     }
864 |
865 |     const fetchIncomingMessageUsers = async () => {
866 |       setLoadingIncomingMessageUsers(true);
867 |       try {
868 |         const { getUserData } = await import("@/services/authService");
869 |         const uniqueUserIds = [...new Set(incomingMessages.map(conv => conv.otherUserId))];
870 |         const userDataPromises = uniqueUserIds.map(async (userId) => {
871 |           try {
872 |             const userData = await getUserData(userId);
873 |             return { userId, userData };
874 |           } catch (error) {
875 |             console.error(`Error fetching user data for ${userId}:`, error);
876 |             return { userId, userData: null };
877 |           }
878 |         });
879 |
880 |         const results = await Promise.all(userDataPromises);
881 |         const usersMap: Record<string, any> = {};
882 |         results.forEach(({ userId, userData }) => {
883 |           if (userData) {
884 |             usersMap[userId] = userData;
885 |           }
886 |         });
887 |         setIncomingMessageUsers(usersMap);
888 |       } catch (error) {
889 |         console.error("Error fetching incoming message users:", error);
890 |       } finally {
891 |         setLoadingIncomingMessageUsers(false);
892 |       }
893 |     };
894 |
895 |     fetchIncomingMessageUsers();
896 |   }, [incomingMessages]);
897 |
898 |   // Mark all notifications as read when notification drawer opens
899 |   // This makes the red indicator disappear when user views notifications
900 |   useEffect(() => {
901 |     if (showNotificationDrawer && unreadCount > 0) {
902 |       // Mark all notifications as read when user opens the drawer to view them
903 |       // This removes the red indicator from the notification bell
904 |       markAllAsRead();
905 |     }
906 |   }, [showNotificationDrawer]);
907 |
908 |   // Get matched users using matching algorithm
909 |   // Use effectiveLocation so matching works in both Beacon Mode and Active Workout
910 |   const { matches, loading: matchesLoading } = useMatching({
911 |     currentUserId: user?.uid || "",
912 |     currentLocation: effectiveLocation, // Use effective location (location or initialLocation)
913 |     activity: selectedActivity,
914 |     fitnessLevel: userProfile?.fitnessLevel || "intermediate",
915 |     pace: userProfile?.pace,
916 |     visibility: userProfile?.visibility || {
917 |       visibleToAllLevels: true,
918 |       allowedLevels: ["beginner", "intermediate", "pro"]
```

```
919 |     },
920 |     searchFilter: searchFilter,
921 |     radiusPreference: userProfile?.radiusPreference || "normal"
922 |   });
923 |
924 |   // Update last viewed count when matches change and sidebar is open
925 |   useEffect(() => {
926 |     if (showSidebar && matches.length > 0) {
927 |       setLastViewedMatchesCount(matches.length);
928 |     }
929 |   }, [showSidebar, matches.length]);
930 |
931 |   // Calculate actual matching radius based on activity and radius preference
932 |   // Matches the logic from matchingService.ts computeRadius function
933 |   const calculateMatchingRadius = useCallback((
934 |     activity: Activity,
935 |     radiusPreference: RadiusPreference = "normal"
936 |   ): number => {
937 |     // Radius lookup table: exact distances for each activity and preference combination
938 |     const RADIUS_LOOKUP: Record<Activity, Record<RadiusPreference, number>> = {
939 |       walking: {
940 |         nearby: 100,   // Still Apply (0.5x): 100m
941 |         normal: 200,   // Normal (1x): 200m
942 |         wide: 400      // Wide (2x): 400m
943 |       },
944 |       running: {
945 |         nearby: 200,   // Still Apply (0.5x): 200m (slightly tighter)
946 |         normal: 350,   // Normal (1x): 350m (optimized from 500m)
947 |         wide: 800      // Wide (2x): 800m (more reasonable)
948 |       },
949 |       cycling: {
950 |         nearby: 400,   // Still Apply (0.5x): 400m (tighter)
951 |         normal: 1000,  // Normal (1x): 1km
952 |         wide: 2000     // Wide (2x): 2km
953 |       }
954 |     };
955 |
956 |     return RADIUS_LOOKUP[activity]?.[radiusPreference] || RADIUS_LOOKUP.running.normal;
957 |   }, []);
958 |
959 |   // Calculate matching radius based on selected activity and user preference
960 |   const matchingRadiusMeters = useMemo(() => {
961 |     if (!selectedActivity || !userProfile) {
962 |       return 350; // Default to 350m (running normal)
963 |     }
964 |     return calculateMatchingRadius(
965 |       selectedActivity,
966 |       userProfile.radiusPreference || "normal"
967 |     );
968 |   }, [selectedActivity, userProfile?.radiusPreference, calculateMatchingRadius]);
969 |
970 |   const matchingRadiusKm = matchingRadiusMeters / 1000; // Convert to km
971 |
972 |   // Get nearby users from hook (fallback for non-matched display)
973 |   // Show nearby users when location is available (works in both Beacon Mode and Active Workout)
974 |   // Use activity-based radius instead of hardcoded 50km
975 |   // Use effectiveLocation (location in active workout, initialLocation in beacon mode)
976 |   const { nearbyUsers: nearbyUsersFromHook, loading: usersLoading } = useNearbyUsers(
977 |     effectiveLocation, // Use effective location (location or initialLocation)
978 |     matchingRadiusKm, // Use activity-based radius (varies by preference: normal = cycling 1km,
running 350m, walking 20...
979 |     activityFilter, // Apply activity filter (running/cycling/walking/all)
980 |     "all", // Gender filter (not used currently)
981 |     user?.uid || null,
982 |     true // Always show nearby users when location is available (not just during workout)
983 |   );
984 |
985 |
986 |   // Friend status tracking (mock data - replace with backend later)
987 |   const [friendStatuses, setFriendStatuses] = useState<Record<string, { status: FriendStatus;
cooldownUntil?: number }>>...
988 |
989 |   const getFriendStatus = (userId: string | number): FriendStatus => {
```

```
    const id = typeof userId === "number" ? userId.toString() : userId;

    // Check if already friends
    const friends = userProfile?.friends || [];
    if (friends.some((f: any) => String(f) === id)) {
      return "friends";
    }

    // Check if request pending (outgoing)
    if (friendRequests.outgoing.includes(id)) {
      return "request_pending";
    }

    // Check if request received (incoming)
    if (friendRequests.incoming.includes(id)) {
      return "request_received";
    }

    // Check local state
    const localStatus = friendStatuses[id];
    if (localStatus) {
      return localStatus.status;
    }

    return "not_friends";
  };

  const getCooldownDaysForFriend = (userId: string | number): number => {
    const id = typeof userId === "number" ? userId.toString() : userId;
    const cooldownUntil = friendStatuses[id]?.cooldownUntil;
    if (!cooldownUntil) return 0;
    const now = Date.now();
    const diff = cooldownUntil - now;
    if (diff <= 0) return 0;
    return Math.ceil(diff / (1000 * 60 * 60 * 24));
  };

  const activities = [
    { id: "running", label: "Running", icon: DirectionsRunIcon, color: "success" },
    { id: "cycling", label: "Cycling", icon: DirectionsBikeIcon, color: "primary" },
    { id: "walking", label: "Walking", icon: DirectionsWalkIcon, color: "warning" },
  ] as const;

  // Filter activities based on user's profile
  // Ensure we always have at least the default activities if userActivities is empty
  const availableActivities = userActivities.length > 0
    ? activities.filter(act =>
        userActivities.includes(act.id as "running" | "cycling" | "walking")
      )
    : activities; // Fallback to all activities if userActivities is empty

  // Use matched users if available, otherwise fallback to nearby users
  // Filter out declined users (within cooldown period)
  const now = Date.now();
  const matchedUsersForDisplay = useMemo(() => {
    return matches
      .filter((match) => {
        const userId = match.user.uid;
        const cooldownUntil = declinedUsers[userId];
        return !cooldownUntil || cooldownUntil <= now;
      })
      .map((match) => {
        const username = match.user.name || null;
        const activity = match.user.activity || null;
        const displayName = getDisplayName(username, match.user.uid, activity);
        return {
          id: match.user.uid,
          name: displayName,
          distance: formatDistance(match.distance / 1000),
          distanceValue: match.distance / 1000,
          activity: match.user.activity || "Running",
```

```
1061 |           avatar: getProfilePictureUrl(match.user.photoURL, match.user.avatar, displayName),
1062 |         lat: match.user.location.lat,
1063 |         lng: match.user.location.lng,
1064 |         matchScore: match.score,
1065 |         fitnessLevel: match.user.fitnessLevel,
1066 |         pace: match.user.pace,
1067 |           timestamp: match.user.timestamp || null // Include timestamp for active workout
filtering
1068 |         };
1069 |       });
1070 |   }, [matches, declinedUsers, now]);
1071 |
1072 |   // Use nearby users from hook, or fallback to mock data for UI features
1073 |   const nearbyUsers = useMemo(() => {
1074 |     let result;
1075 |     if (matchedUsersForDisplay.length > 0) {
1076 |       result = matchedUsersForDisplay;
1077 |     } else if (nearbyUsersFromHook.length > 0) {
1078 |       result = nearbyUsersFromHook.map((userData: any) => {
1079 |         const username = userData.name || null;
1080 |         const activity = userData.activity || null;
1081 |         const displayName = getDisplayName(username, userData.id, activity);
1082 |         return {
1083 |           id: userData.id,
1084 |           name: displayName,
1085 |           distance: formatDistance(userData.distance),
1086 |           distanceValue: userData.distance,
1087 |           activity: userData.activity || "Running",
1088 |           avatar: getProfilePictureUrl(userData.photoURL, userData.avatar, displayName),
1089 |         lat: userData.lat,
1090 |         lng: userData.lng,
1091 |           photos: [],
1092 |           bio: "",
1093 |           timestamp: userData.timestamp || null // Include timestamp for active workout filtering
1094 |         };
1095 |       });
1096 |     } else {
1097 |       result = [];
1098 |     }
1099 |
1100 |     // Update ref with latest value
1101 |     nearbyUsersRef.current = result;
1102 |     return result;
1103 |   }, [matchedUsersForDisplay, nearbyUsersFromHook]);
1104 |
1105 |   // Check for nearby active users and show notification when workout is inactive
1106 |   useEffect(() => {
1107 |     // Only check when workout is inactive
1108 |     if (isActive) {
1109 |       // Reset notification state when workout starts
1110 |       notificationShownRef.current = false;
1111 |       notificationCreatedRef.current = false;
1112 |       setShowNotification(false);
1113 |       return;
1114 |     }
1115 |
1116 |     // Don't check if notification was already shown in this session
1117 |     if (notificationShownRef.current) {
1118 |       return;
1119 |     }
1120 |
1121 |     // Check if we have location to calculate distances (use effectiveLocation for beacon mode)
1122 |     if (!effectiveLocation || !effectiveLocation.lat || !effectiveLocation.lng) {
1123 |       return;
1124 |     }
1125 |
1126 |     // Throttle notification checks to every 10 seconds
1127 |     let lastCheckTime = 0;
1128 |     const checkInterval = 10000; // 10 seconds
1129 |
1130 |     // Listen to all users to check for nearby active users
1131 |     const unsubscribe = listenToAllUsers((users) => {
```

```
1132 |       // Only check when workout is inactive (Beacon Mode)
1133 |       if (isActive || notificationShownRef.current) {
1134 |         return;
1135 |       }
1136 |
1137 |       // Throttle checks to every 10 seconds
1138 |       const now = Date.now();
1139 |       if (now - lastCheckTime < checkInterval) {
1140 |         return;
1141 |       }
1142 |       lastCheckTime = now;
1143 |
1144 |       const activeThreshold = 3 * 60 * 1000; // 3 minutes
1145 |
1146 |       // Calculate matching radius based on selected activity and user preference
1147 |       // Use the same radius as the matching system and map circle
1148 |       const maxDistanceKm = selectedActivity && userProfile
1149 |         ? calculateMatchingRadius(
1150 |             selectedActivity,
1151 |             userProfile.radiusPreference || "normal"
1152 |           ) / 1000 // Convert meters to km
1153 |         : 0.35; // Default to 350m (running normal) if activity/preference not available
1154 |
1155 |       // Filter users to find active nearby users
1156 |       const activeNearbyUsers = Object.entries(users || {})
1157 |         .filter(([userId, userData]: [string, any]) => {
1158 |           // Exclude current user
1159 |           if (userId === user?.uid) {
1160 |             return false;
1161 |           }
1162 |
1163 |           // Check if user has location
1164 |           if (!userData.lat || !userData.lng) {
1165 |             return false;
1166 |           }
1167 |
1168 |           // Check if user is visible
1169 |           if (userData.visible === false) {
1170 |             return false;
1171 |           }
1172 |
1173 |           // Check if timestamp is recent (within 3 minutes)
1174 |           if (!userData.timestamp) {
1175 |             return false;
1176 |           }
1177 |
1178 |           const timeDiff = now - userData.timestamp;
1179 |           if (timeDiff > activeThreshold) {
1180 |             return false;
1181 |           }
1182 |
1183 |           // Check distance using activity-based radius (varies by preference: normal = cycling
1184 |           // 1km, running 350m, walki... Uses Haversine formula to calculate distance
1185 |           const R = 6371; // Earth's radius in km
1186 |           const dLat = (userData.lat - effectiveLocation.lat) * Math.PI / 180;
1187 |           const dLng = (userData.lng - effectiveLocation.lng) * Math.PI / 180;
1188 |           const a =
1189 |             Math.sin(dLat / 2) * Math.sin(dLat / 2) +
1190 |             Math.cos(effectiveLocation.lat * Math.PI / 180) * Math.cos(userData.lat * Math.PI /
1191 |             Math.sin(dLng / 2) * Math.sin(dLng / 2);
1192 |           const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
1193 |           const distance = R * c;
1194 |
1195 |           return distance <= maxDistanceKm;
1196 |         });
1197 |
1198 |       // Show notification if there are active nearby users
1199 |       if (activeNearbyUsers.length > 0 && !notificationShownRef.current) {
1200 |         setShowNotification(true);
1201 |         notificationShownRef.current = true;
1202 |
```

```
1203 |         // Also add notification to bell if not already created
1204 |         if (!notificationCreatedRef.current && user?.uid) {
1205 |           // Check if there's already an unread nearby_active_user notification
1206 |           const hasExistingNotification = notifications.some(
1207 |             (notif) => notif.type === "nearby_active_user" && !notif.read
1208 |           );
1209 |
1210 |           if (!hasExistingNotification) {
1211 |             addNotification({
1212 |               type: "nearby_active_user",
1213 |               fromUserId: "system",
1214 |               fromUserName: "PaceMatch",
1215 |               fromUserAvatar: "",
1216 |               message: `${activeNearbyUsers.length} active user${activeNearbyUsers.length > 1 ?
     's' : ''} nearby! Start ...
1217 |             });
1218 |             notificationCreatedRef.current = true;
1219 |           }
1220 |         }
1221 |
1222 |         // Auto-dismiss banner after 5 seconds
1223 |         setTimeout(() => {
1224 |           setShowNotification(false);
1225 |         }, 5000);
1226 |       }
1227 |     });
1228 |
1229 |     return () => {
1230 |       unsubscribe();
1231 |     };
1232 |   }, [isActive, location, user?.uid, selectedActivity, userProfile?.radiusPreference,
     calculateMatchingRadius, notificat...
1234 |   // Apply overlap prevention to nearby users markers
1235 |   const nearbyUsersWithAdjustedPositions = useMemo(() => {
1236 |     if (!nearbyUsers || nearbyUsers.length === 0) {
1237 |       return [];
1238 |     }
1239 |
1240 |     // Filter out current user
1241 |     let otherUsers = nearbyUsers.filter((userData: any) => userData.id !== user?.uid);
1242 |
1243 |     // Additional safety filter: Only show users with active workouts (recent timestamp)
1244 |     // This ensures users are only visible when they have an active workout session
1245 |     // Inactive locations automatically terminate after 3 minutes
1246 |     const now = Date.now();
1247 |     const activeThreshold = 3 * 60 * 1000; // 3 minutes in milliseconds
1248 |
1249 |     otherUsers = otherUsers.filter((userData: any) => {
1250 |       // User must have a timestamp indicating recent location update
1251 |       if (!userData.timestamp) {
1252 |         return false;
1253 |       }
1254 |
1255 |       // Check if timestamp is recent (within 3 minutes)
1256 |       const timeDiff = now - userData.timestamp;
1257 |       return timeDiff <= activeThreshold;
1258 |     });
1259 |
1260 |     if (otherUsers.length === 0) {
1261 |       return [];
1262 |     }
1263 |
1264 |     // Prepare markers for overlap prevention
1265 |     const markersForOverlap = otherUsers.map((u: any) => ({
1266 |       lat: u.lat,
1267 |       lng: u.lng,
1268 |       id: u.id
1269 |     }));
1270 |
1271 |     let adjustedMarkers: AdjustedPosition[] = [];
1272 |
1273 |     // Try to use map instance for accurate pixel calculations
```

```
1274 |     // mapRef should be a google.maps.Map instance, but we need to check if it has the required methods
1275 |     const googleMap = mapRef && isLoaded && window.google && (mapRef as any).getProjection
1276 |       ? (mapRef as any as google.maps.Map)
1277 |       : null;
1278 |
1279 |     if (googleMap) {
1280 |       adjustedMarkers = preventMarkerOverlap(
1281 |         markersForOverlap,
1282 |         googleMap,
1283 |         {
1284 |           minDistancePixels: 50,
1285 |           maxOffsetMeters: 50,
1286 |           iterations: 10
1287 |         }
1288 |       );
1289 |     } else if (location && location.lat && location.lng) {
1290 |       // Fallback: use simple algorithm with zoom and center
1291 |       adjustedMarkers = preventMarkerOverlapSimple(
1292 |         markersForOverlap,
1293 |         mapZoom,
1294 |         { lat: location.lat, lng: location.lng },
1295 |         {
1296 |           minDistancePixels: 50,
1297 |           maxOffsetMeters: 50,
1298 |           iterations: 10
1299 |         }
1300 |       );
1301 |     } else {
1302 |       // If no map or location, return original positions
1303 |       return otherUsers;
1304 |     }
1305 |
1306 |     // Merge adjusted positions with original user data
1307 |     return adjustedMarkers.map((adj: AdjustedPosition) => {
1308 |       // Find the original user data
1309 |       const originalUser = otherUsers.find((u: any) => u.id === adj.id);
1310 |       if (!originalUser) {
1311 |         return {
1312 |           ...adj,
1313 |           lat: adj.adjustedLat,
1314 |           lng: adj.adjustedLng
1315 |         };
1316 |       }
1317 |
1318 |       // Return original user data with adjusted position
1319 |       return {
1320 |         ...originalUser,
1321 |         lat: adj.adjustedLat,
1322 |         lng: adj.adjustedLng
1323 |       };
1324 |     });
1325 |   }, [nearbyUsers, mapRef, isLoaded, mapZoom, location, user?.uid]);
1326 |
1327 |   // Track nearby users during workout - using ref to prevent infinite loops
1328 |   const nearbyUserIdsString = useMemo(() =>
1329 |     nearbyUsers.map(u => u.id).sort().join(','),
1330 |     [nearbyUsers]
1331 |   );
1332 |
1333 |   const previousUserIdsRef = useRef<string>('');
1334 |
1335 |   useEffect(() => {
1336 |     if (!isActive) {
1337 |       // Reset tracking when workout stops
1338 |       processedUserIdsRef.current.clear();
1339 |       previousUserIdsRef.current = '';
1340 |       return;
1341 |     }
1342 |
1343 |     // Check if user IDs have actually changed
1344 |     if (nearbyUserIdsString === previousUserIdsRef.current) {
```

```
1345 |       return; // No change, skip update
1346 |     }
1347 |
1348 |     // Update the ref to current state
1349 |     previousUserIdsRef.current = nearbyUserIdsString;
1350 |
1351 |     // Get current nearby users from ref (always has latest value)
1352 |     const currentNearbyUsers = nearbyUsersRef.current;
1353 |     if (currentNearbyUsers.length === 0) return;
1354 |
1355 |     // Create a set of current user IDs
1356 |     const currentUserIds = new Set(currentNearbyUsers.map(u => u.id));
1357 |
1358 |     // Find users that haven't been processed yet
1359 |     const newUserIds = Array.from(currentUserIds).filter((id: string) => !
processedUserIdsRef.current.has(id));
1360 |
1361 |     if (newUserIds.length === 0) return;
1362 |
1363 |     // Mark these users as processed
1364 |     newUserIds.forEach((id: string) => processedUserIdsRef.current.add(id));
1365 |
1366 |     // Update workout nearby users state with REAL Firebase users
1367 |     // These users come from Firebase via useNearbyUsers hook -> locationService -> Firebase
Realtime Database
1368 |     setWorkoutNearbyUsers(prev => {
1369 |       const existingIds = new Set(prev.map(u => u.id));
1370 |       const newUsers = currentNearbyUsers
1371 |         .filter(user => newUserIds.includes(user.id) && !existingIds.has(user.id))
1372 |         .map(user => {
1373 |           const userActivity = user.activity || "running";
1374 |           const isSameActivity = selectedActivity && userActivity.toLowerCase() ===
selectedActivity.toLowerCase();
1375 |           return {
1376 |             id: user.id,
1377 |             name: user.name,
1378 |             avatar: user.avatar,
1379 |             activity: userActivity,
1380 |             distance: user.distance,
1381 |             distanceValue: user.distanceValue,
1382 |             isSameActivity: isSameActivity
1383 |           };
1384 |         });
1385 |
1386 |       if (newUsers.length === 0) {
1387 |         return prev;
1388 |       }
1389 |
1390 |       console.log(`'  Added ${newUsers.length} real Firebase user(s) to workout tracking:`,
newUsers.map(u => u.name));
1391 |       return [...prev, ...newUsers];
1392 |     });
1393 |   }, [isActive, nearbyUserIdsString, selectedActivity]);
1394 |
1395 |   // Filter users by activity
1396 |   const filteredUsers = activityFilter === "all"
1397 |     ? nearbyUsers
1398 |     : nearbyUsers.filter(user => user.activity.toLowerCase() === activityFilter);
1399 |
1400 |   // Sort by distance
1401 |   const sortedUsers = [...filteredUsers].sort((a, b) => a.distanceValue - b.distanceValue);
1402 |
1403 |   // Track location history for trail when activity is active
1404 |   useEffect(() => {
1405 |     if (isActive && location) {
1406 |       const lastPoint = lastLocationRef.current;
1407 |
1408 |       // Check if location actually changed (avoid duplicate processing)
1409 |       if (lastPoint &&
1410 |           Math.abs(lastPoint.lat - location.lat) < 0.00001 &&
1411 |           Math.abs(lastPoint.lng - location.lng) < 0.00001) {
1412 |         return; // Location hasn't changed significantly, skip update
1413 |       }
1414 |
1415 |       // Calculate heading before updating history (only if we have a previous point)
```

```
1416 |       let newHeading: number | null = null;
1417 |       if (lastPoint) {
1418 |         const distance = Math.sqrt(
1419 |           Math.pow(lastPoint.lat - location.lat, 2) +
1420 |           Math.pow(lastPoint.lng - location.lng, 2)
1421 |         );
1422 |
1423 |         // Only calculate heading if moved significantly
1424 |         if (distance > 0.00001) {
1425 |           newHeading = calculateBearing(
1426 |             lastPoint.lat,
1427 |             lastPoint.lng,
1428 |             location.lat,
1429 |             location.lng
1430 |           );
1431 |         }
1432 |       }
1433 |
1434 |       // Update location history
1435 |       setLocationHistory((prev) => {
1436 |         const newHistory = [...prev, { lat: location.lat, lng: location.lng }];
1437 |         return newHistory.slice(-100);
1438 |       });
1439 |
1440 |       // Update ref to track last location
1441 |       lastLocationRef.current = { lat: location.lat, lng: location.lng };
1442 |
1443 |       // Update heading if calculated
1444 |       if (newHeading !== null) {
1445 |         setUserHeading((prevHeading) => {
1446 |           if (prevHeading === null || Math.abs(prevHeading - newHeading!) > 1) {
1447 |             return newHeading!;
1448 |           }
1449 |           return prevHeading;
1450 |         });
1451 |
1452 |         if (isWazeMode && mapRef && isLoaded && window.google) {
1453 |           mapRef.setHeading(newHeading);
1454 |           setMapHeading(newHeading);
1455 |         }
1456 |       }
1457 |     } else if (!isActive) {
1458 |       setLocationHistory([]);
1459 |       setUserHeading(null);
1460 |       lastLocationRef.current = null;
1461 |     }
1462 |   }, [location, isActive, isWazeMode, mapRef, isLoaded]);
1463 |
1464 |   // Timer interval - runs continuously when active (even when paused, to show paused time)
1465 |   useEffect(() => {
1466 |     if (isActive && startTime) {
1467 |       intervalRef.current = setInterval(() => {
1468 |         const now = new Date();
1469 |         // Calculate total elapsed time from start
1470 |         const totalElapsed = Math.floor((now.getTime() - startTime.getTime()) / 1000);
1471 |
1472 |         // Subtract paused time
1473 |         let currentPausedTime = totalPausedTime;
1474 |         if (isPaused && pauseStartTime) {
1475 |           // Add current pause duration
1476 |           currentPausedTime += Math.floor((now.getTime() - pauseStartTime.getTime()) / 1000);
1477 |         }
1478 |
1479 |         const activeElapsed = totalElapsed - currentPausedTime;
1480 |         setElapsedTime(activeElapsed);
1481 |
1482 |         // Only update distance and speed when not paused and location is available
1483 |         if (!isPaused && location) {
1484 |           setPointsTracked(prev => prev + 1);
1485 |
1486 |           // Calculate distance from location history
```

```
1487 |           if (locationHistory.length > 1) {
1488 |             let totalDistance = 0;
1489 |             for (let i = 1; i < locationHistory.length; i++) {
1490 |               const prev = locationHistory[i - 1];
1491 |               const curr = locationHistory[i];
1492 |               // Haversine formula for distance
1493 |               const R = 6371; // Earth's radius in km
1494 |               const dLat = (curr.lat - prev.lat) * Math.PI / 180;
1495 |               const dLng = (curr.lng - prev.lng) * Math.PI / 180;
1496 |               const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
1497 |                         Math.cos(prev.lat * Math.PI / 180) * Math.cos(curr.lat * Math.PI / 180) *
1498 |                         Math.sin(dLng/2) * Math.sin(dLng/2);
1499 |               const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
1500 |               totalDistance += R * c;
1501 |             }
1502 |             setDistance(totalDistance);
1503 |
1504 |             // Calculate current speed
1505 |             const currentTime = Date.now();
1506 |             const timeDiff = (currentTime - lastTimeRef.current) / 1000; // seconds
1507 |
1508 |             if (timeDiff >= 1 && lastTimeRef.current > 0) {
1509 |               const distanceDiff = totalDistance - lastDistanceRef.current;
1510 |               const speed = (distanceDiff / timeDiff) * 3600; // km/h
1511 |               setCurrentSpeed(speed);
1512 |
1513 |               lastDistanceRef.current = totalDistance;
1514 |               lastTimeRef.current = currentTime;
1515 |             } else if (lastTimeRef.current === 0) {
1516 |               lastTimeRef.current = currentTime;
1517 |               lastDistanceRef.current = totalDistance;
1518 |             }
1519 |           }
1520 |         }
1521 |       }, 1000);
1522 |     } else {
1523 |       if (intervalRef.current) {
1524 |         clearInterval(intervalRef.current);
1525 |         intervalRef.current = null;
1526 |       }
1527 |     }
1528 |
1529 |     return () => {
1530 |       if (intervalRef.current) {
1531 |         clearInterval(intervalRef.current);
1532 |       }
1533 |     };
1534 |   }, [isActive, isPaused, startTime, pauseStartTime, totalPausedTime, location,
locationHistory]);
1536 |
1537 |   // Calculate average speed
1538 |   useEffect(() => {
1539 |     if (elapsedTime > 0 && distance > 0) {
1540 |       const avgSpeedValue = (distance / elapsedTime) * 3600; // km/h
1541 |       setAvgSpeed(avgSpeedValue);
1542 |     }
1543 |   }, [elapsedTime, distance]);
1544 |
1545 |   const handleStartStop = async () => {
1546 |     if (!isActive) {
1547 |       // Start activity - first request GPS permission
1548 |       // Check if geolocation is available
1549 |       if (!isNativePlatform() && !navigator.geolocation) {
1550 |         toast.error("Geolocation is not supported by your browser. Please use a modern
browser.");
1551 |         return;
1552 |       }
1553 |
1554 |       console.log("Ø<ßÃ Starting workout... Requesting GPS permission first...");
1555 |
1556 |       // Request GPS permission before starting workout
1557 |       try {
1558 |         if (isNativePlatform()) {
```

```
1558 |           // For native platforms, request permission explicitly
1559 |           const { Geolocation } = await import("@capacitor/geolocation");
1560 |           const permissionStatus = await Geolocation.requestPermissions();
1561 |
1562 |           if (permissionStatus.location !== 'granted') {
1563 |             toast.error("Location permission is required to start a workout. Please allow
location access in your device...
1564 |             return;
1565 |           }
1566 |
1567 |           console.log("'  Location permission granted (native)");
1568 |         } else {
1569 |           // For web, try to get permission by requesting position
1570 |           // This will trigger the browser's permission prompt
1571 |           console.log("Ø=ÜÍ Requesting GPS permission (web)...");
1572 |
1573 |           try {
1574 |             await new Promise<void>((resolve, reject) => {
1575 |               navigator.geolocation.getCurrentPosition(
1576 |                 (position) => {
1577 |                   console.log("'  GPS permission granted and position obtained (web)");
1578 |                   resolve();
1579 |                 },
1580 |                 (err) => {
1581 |                   // Handle different error codes
1582 |                   switch (err.code) {
1583 |                     case err.PERMISSION_DENIED:
1584 |                       // Permission denied - don't allow workout to start
1585 |                       toast.error("Location permission denied. Please allow location access in
your browser settings to ...
1586 |                       reject(err);
1587 |                       break;
1588 |                     case err.POSITION_UNAVAILABLE:
1589 |                       // Permission might be granted, but GPS signal unavailable
1590 |                       // Allow workout to start - GPS will retry when signal is available
1591 |                       console.warn("& þ  GPS signal unavailable, but permission may be granted.
Starting workout anyway.....
1592 |                       toast.warning("GPS signal unavailable. Workout will start, but location
tracking will begin when G...
1593 |                       resolve(); // Allow workout to start
1594 |                       break;
1595 |                     case err.TIMEOUT:
1596 |                       // Timeout - permission might be granted, just taking too long
1597 |                       // Allow workout to start - GPS will retry
1598 |                       console.warn("& þ  GPS request timeout, but permission may be granted.
Starting workout anyway...");
1599 |                       toast.warning("GPS request timed out. Workout will start, but location
tracking will begin when GP...
1600 |                       resolve(); // Allow workout to start
1601 |                       break;
1602 |                     default:
1603 |                       // Unknown error - be lenient and allow workout to start
1604 |                       console.warn("& þ  GPS error:", err.message, "- Starting workout anyway...");
1605 |                       resolve(); // Allow workout to start
1606 |                   }
1607 |                 },
1608 |                 {
1609 |                   enableHighAccuracy: false, // Use lower accuracy for permission check (faster)
1610 |                   timeout: 5000, // Shorter timeout for permission check
1611 |                   maximumAge: 60000 // Accept cached location
1612 |                 }
1613 |               );
1614 |             });
1615 |           } catch (error: any) {
1616 |             // If it's a permission denied error, don't start workout
1617 |             if (error?.code === 1) { // PERMISSION_DENIED
1618 |               throw error; // Re-throw to be caught by outer catch
1619 |             }
1620 |             // For other errors, allow workout to start
1621 |             console.warn("& þ  GPS error during permission check, but allowing workout to start:",
error);
1622 |           }
1623 |         }
1624 |
1625 |         // Permission granted - now start the workout
1626 |         console.log("'  GPS permission granted. Starting workout...");
1627 |         toast.success("Activity started! GPS tracking enabled.");
1628 |         setIsActive(true);
```

```
1629 |         setIsPaused(false);
1630 |         setPausedDueToInactivity(false); // Reset inactivity state
1631 |         setShowInactivityWarning(false); // Close warning if open
1632 |         setStartTime(new Date());
1633 |         setElapsedTime(0);
1634 |         setDistance(0);
1635 |         setCurrentSpeed(0);
1636 |         setAvgSpeed(0);
1637 |         setPointsTracked(0);
1638 |         setTotalPausedTime(0);
1639 |         setPauseStartTime(null);
1640 |         lastDistanceRef.current = 0;
1641 |         lastTimeRef.current = 0;
1642 |         setShowNotification(false);
1643 |         // Reset workout tracking
1644 |         setWorkoutPokes([]);
1645 |         setWorkoutNearbyUsers([]);
1646 |       } catch (error: any) {
1647 |         // Permission denied or error occurred
1648 |         console.error("'L Failed to get GPS permission:", error);
1649 |         // Error message already shown in the catch block above
1650 |         return;
1651 |       }
1652 |     } else {
1653 |       // Show confirmation dialog before stopping
1654 |       setShowStopConfirmation(true);
1655 |     }
1656 |   };
1657 |
1658 |   const handleConfirmStop = async () => {
1659 |     // Force-stop GPS watcher immediately
1660 |     stopTracking();
1661 |     // Stop activity - show summary
1662 |     setIsActive(false);
1663 |     setIsPaused(false);
1664 |     setPauseStartTime(null);
1665 |     setPausedDueToInactivity(false); // Reset inactivity state
1666 |     setShowStopConfirmation(false);
1667 |     setShowInactivityWarning(false); // Close warning if open
1668 |
1669 |     // Clear location from Firebase when stopping workout (entering Beacon Mode)
1670 |     if (user?.uid) {
1671 |       try {
1672 |         const { clearUserLocation } = await import("@/services/locationService");
1673 |         await clearUserLocation(user.uid);
1674 |         console.log("'  Location cleared from Firebase (entering Beacon Mode)");
1675 |       } catch (error) {
1676 |         console.error("Error clearing location:", error);
1677 |       }
1678 |     }
1679 |
1680 |     // Log real Firebase users detected during workout
1681 |     if (workoutNearbyUsers.length > 0) {
1682 |       console.log(`'  Found ${workoutNearbyUsers.length} real Firebase users during workout:`,
workoutNearbyUsers);
1683 |     } else {
1684 |       console.log("& þ  No real Firebase users detected during workout");
1685 |     }
1686 |
1687 |     // Always show summary when stopping activity (to show nearby people and stats)
1688 |     // Use a delay to ensure the stop confirmation dialog is fully closed first
1689 |     setTimeout(() => {
1690 |       setShowSummary(true);
1691 |     }, 300);
1692 |   };
1693 |
1694 |   const blockingModalOpen =
1695 |     showMessageModal ||
1696 |     showFriendRequestModal ||
1697 |     showPokeModal ||
1698 |     showProfileView ||
1699 |     showMatchesDrawer ||
```

```
    showFilterModal ||
    showNotificationDrawer;

  const handlePause = () => {
    if (!isPaused) {
      // Pausing - record pause start time
      setPauseStartTime(new Date());
      setIsPaused(true);
      toast("Activity paused");
      // Close warning modal if open
      setShowInactivityWarning(false);
    } else {
      // Resuming - calculate and add paused duration
      if (pauseStartTime) {
        const pauseDuration = Math.floor((new Date().getTime() - pauseStartTime.getTime()) /
1000);
        setTotalPausedTime(prev => prev + pauseDuration);
        setPauseStartTime(null);
      }
      setIsPaused(false);
      setPausedDueToInactivity(false); // Reset inactivity flag on manual resume
      toast("Activity resumed");
    }
  };

  const handleInactivityPause = () => {
    // Auto-pause due to inactivity
    setPauseStartTime(new Date());
    setIsPaused(true);
    setPausedDueToInactivity(true);
    setShowInactivityWarning(false);
    toast("Workout paused due to inactivity");
  };

  const handleDismissInactivityWarning = () => {
    // User dismissed warning - reset movement detection
    setShowInactivityWarning(false);
    // The movement detection will continue monitoring
  };

  /**
   * Reverse geocode coordinates to get a readable location name
   * Falls back to a general location description if geocoding fails
   */
  const getLocationName = useCallback(async (lat: number, lng: number): Promise<string> => {
    // Check if Google Maps API is available
    if (!window.google || !window.google.maps || !window.google.maps.Geocoder) {
      // Fallback: Use coordinates formatted as a general location
      return `${lat.toFixed(4)}, ${lng.toFixed(4)}`;
    }

    try {
      const geocoder = new window.google.maps.Geocoder();
      return new Promise((resolve) => {
        geocoder.geocode(
          { location: { lat, lng } },
          (results, status) => {
            if (status === "OK" && results && results.length > 0) {
              const address = results[0].formatted_address;
              // Extract a shorter, more readable version (neighborhood, city, or landmark)
              // Try to get a meaningful location name
              let locationName = address;

              // Try to extract a more user-friendly location name
              // Look for neighborhood, locality, or establishment name
              const addressComponents = results[0].address_components || [];

              // Try to find neighborhood, sublocality, or locality
              const neighborhood = addressComponents.find((comp: any) =>
                comp.types.includes("neighborhood") ||
                comp.types.includes("sublocality") ||
                comp.types.includes("sublocality_level_1")
```

```
1771 |               );
1772 |
1773 |               const locality = addressComponents.find((comp: any) =>
1774 |                 comp.types.includes("locality")
1775 |               );
1776 |
1777 |               const adminArea = addressComponents.find((comp: any) =>
1778 |                 comp.types.includes("administrative_area_level_1")
1779 |               );
1780 |
1781 |               // Build a friendly location name
1782 |               if (neighborhood) {
1783 |                 locationName = `${neighborhood.long_name}${locality ? `, ${locality.long_name}
1784 |"}${adminArea ? `, ${adminArea.short_name}` : ""}`;
1785 |               } else if (locality) {
1786 |                 locationName = `${locality.long_name}${adminArea ? `, ${adminArea.short_name}` :
1787 |"}`;               } else {
1788 |                 // Use the formatted address but shorten it if it's too long
1789 |                 locationName = address.split(',').slice(0, 2).join(',').trim();
1790 |               }
1791 |
1792 |               resolve(locationName);
1793 |             } else {
1794 |               // Fallback to coordinates if geocoding fails
1795 |               console.warn("Reverse geocoding failed:", status);
1796 |               resolve(`${lat.toFixed(4)}, ${lng.toFixed(4)}`);
1797 |             }
1798 |           }
1799 |         );
1800 |       });
1801 |     } catch (error) {
1802 |       console.error("Error during reverse geocoding:", error);
1803 |       // Fallback to coordinates
1804 |       return `${lat.toFixed(4)}, ${lng.toFixed(4)}`;
1805 |     }
1806 |   }, []);
1807 |
1808 |   const handleSaveWorkout = async () => {
1809 |     if (!user?.uid) {
1810 |       toast.error("You must be logged in to save workouts");
1811 |       return;
1812 |     }
1813 |
1814 |     // Use workoutNearbyUsers - all REAL Firebase users detected during the workout session
1815 |     // These are populated from Firebase via useNearbyUsers hook and locationService
1816 |     const activeNearbyUsers = workoutNearbyUsers.map(user => ({
1817 |       id: user.id,
1818 |       name: user.name,
1819 |       avatar: user.avatar,
1820 |       activity: user.activity,
1821 |       distance: user.distance,
1822 |       isSameActivity: user.isSameActivity, // Preserve activity matching flag for summary display
1823 |     }));
1824 |
1825 |     // Log what we're saving
1826 |     if (activeNearbyUsers.length > 0) {
1827 |       console.log(`'  Saving workout with ${activeNearbyUsers.length} real Firebase users`);
1828 |     } else {
1829 |       console.log("!9þ  No nearby users detected during this workout");
1830 |     }
1831 |
1832 |     // Get location name from current location or first location in history
1833 |     let locationName = "Unknown Location";
1834 |     if (location) {
1835 |       locationName = await getLocationName(location.lat, location.lng);
1836 |     } else if (locationHistory.length > 0) {
1837 |       // Use the first location from the workout history as a fallback
1838 |       const firstLocation = locationHistory[0];
1839 |       locationName = await getLocationName(firstLocation.lat, firstLocation.lng);
1840 |     }
1841 |
1842 |     const workoutData = {
```

```
      activity: selectedActivity,
      date: startTime || new Date(),
      duration: elapsedTime,
      distance,
      avgSpeed,
      nearbyUsers: activeNearbyUsers,
      location: locationName,
    };

    try {
      // Save to Firebase Realtime Database
      console.log("Ø=Ü¾ Saving workout to Firebase:", workoutData);
      const workoutId = await saveWorkout(user.uid, workoutData);
      console.log("'  Workout saved successfully to Firebase");

      // Also save to localStorage via context (for offline/fallback)
      addWorkout(workoutData);

      // Add workout completion notification with workout ID
      addNotification({
        type: "workout_complete",
        fromUserId: user.uid, // Use fromUserId for Firebase format
        fromUserName: "Workout Complete",
        fromUserAvatar: user?.photoURL || "",
        message: `Great job! You completed ${distance.toFixed(2)} km of ${selectedActivity}`,
        workoutId: workoutId,
        linkType: "workout" // Link to workout when tapped
      });
      console.log("'  Workout completion notification added. Unread count should update.");

      toast.success(`Workout saved! ${distance.toFixed(2)} km tracked.`, {
        position: "bottom-center"
      });
      setShowSummary(false);
      setPointsTracked(0);
      setDistance(0);
      setElapsedTime(0);
      setCurrentSpeed(0);
      setAvgSpeed(0);
      setStartTime(null);
    } catch (error) {
      console.error("Error saving workout:", error);
      toast.error("Failed to save workout. Please try again.");
    }
  };

  const handleDiscardWorkout = () => {
    toast("Workout discarded");
    setShowSummary(false);
    setPointsTracked(0);
    setDistance(0);
    setElapsedTime(0);
    setCurrentSpeed(0);
    setAvgSpeed(0);
    setStartTime(null);
    setTotalPausedTime(0);
    setPauseStartTime(null);
    setWorkoutPokes([]);
    setWorkoutNearbyUsers([]);
  };

  const formatTime = (seconds: number) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;
    if (hours > 0) {
      return `${hours}:${minutes.toString().padStart(2, "0")}:${secs.toString().padStart(2, "0")}
    }
    return `${minutes}:${secs.toString().padStart(2, "0")}`;
  };

```

```
  const formatPace = (kmh: number) => {
    if (kmh === 0) return "--:--";
    const minPerKm = 60 / kmh;
    const mins = Math.floor(minPerKm);
    const secs = Math.round((minPerKm - mins) * 60);
    return `${mins}:${secs.toString().padStart(2, "0")}`;
  };

  const convertDistance = (km: number) => {
    if (useMetric) return { value: km.toFixed(2), unit: "km" };
    return { value: (km * 0.621371).toFixed(2), unit: "mi" };
  };

  const convertSpeed = (kmh: number) => {
    if (useMetric) return { value: kmh.toFixed(1), unit: "km/h" };
    return { value: (kmh * 0.621371).toFixed(1), unit: "mph" };
  };

  const handleNotificationBannerTap = () => {
    setShowNotification(false);
    notificationShownRef.current = true; // Mark as shown so it doesn't appear again
    handleStartStop();
  };

  // Removed test notification simulation - notifications now come from Firebase events only

  const handleUserMarkerClick = (user: typeof nearbyUsers[0]) => {
    setSelectedUser(user);
    setShowProfileView(true);
  };

  const handleCenterOnUser = () => {
    if (selectedUser) {
      toast.success(`Centering on ${selectedUser.name}`);
      // In real implementation, this would pan the map
    }
  };

  const handleAddFriend = async (userId: string | number, userName?: string) => {
    if (!user?.uid) {
      toast.error("Please sign in to add friends");
      return;
    }

    const id = typeof userId === "number" ? userId.toString() : userId;

    if (!id) {
      toast.error("Invalid user");
      return;
    }

    if (id === user.uid) {
      toast.error("You can't add yourself");
      return;
    }

    // Prevent duplicate requests
    if (
      friendStatuses[id]?.status === "request_pending" ||
      friendRequests.outgoing.includes(id)
    ) {
      toast("Friend request already sent");
      return;
    }

    try {
      await sendFriendRequest(user.uid, id);

      setFriendStatuses(prev => ({
        ...prev,
        [id]: { status: "request_pending" }
```

```
      }));

      toast.success(`Friend request sent to ${userName || selectedUser?.name || "user"}`);
    } catch (error) {
      console.error("Error sending friend request:", error);
      toast.error("Failed to send friend request. Please try again.");
    }
  };

  const handleAcceptFriend = (userId: string | number) => {
    const id = typeof userId === "number" ? userId.toString() : userId;
    setFriendStatuses(prev => ({
      ...prev,
      [id]: { status: "friends" }
    }));
    setShowFriendRequestModal(false);
    toast.success("Friend request accepted!");
  };

  const handleDeclineFriend = (userId: string | number) => {
    const id = typeof userId === "number" ? userId.toString() : userId;
    // Set 7-day cooldown
    const cooldownUntil = Date.now() + (7 * 24 * 60 * 60 * 1000);
    setFriendStatuses(prev => ({
      ...prev,
      [id]: { status: "denied", cooldownUntil }
    }));
    setShowFriendRequestModal(false);
    toast("Friend request declined. The user won't be notified.");
  };

  const handleUnfriend = async (userId: string | number) => {
    if (!user?.uid) {
      toast.error("You must be logged in to unfriend someone");
      return;
    }

    const id = typeof userId === "number" ? userId.toString() : userId;

    try {
      // Remove from Firebase
      await removeFriend(user.uid, id);

      // Update local state
      setFriendStatuses(prev => ({
        ...prev,
        [id]: { status: "not_friends" }
      }));

      // Update userProfile context
      if (userProfile) {
        const updatedFriends = (userProfile.friends || []).filter(
          (f: any) => String(f) !== id
        );
        setUserProfile({
          ...userProfile,
          friends: updatedFriends
        });
      }

      toast.success(`Unfriended ${selectedUser?.name || "user"}`);
      setShowProfileView(false);
      setSelectedUser(null);
    } catch (error) {
      console.error("Error unfriending user:", error);
      toast.error("Failed to unfriend. Please try again.");
    }
  };

  const handleSendMessage = () => {
    setShowMessageModal(true);
```

```
2055 |   };
2056 |
2057 |   const handleMessageSent = (message: string, isTemplate: boolean) => {
2058 |     if (selectedUser) {
2059 |       toast.success(`Message sent to ${selectedUser.name}!`);
2060 |       console.log("Message sent:", { message, isTemplate, to: selectedUser.name });
2061 |       // In real implementation, this would save to backend/Firebase
2062 |     }
2063 |     setShowMessageModal(false);
2064 |     setSelectedUser(null);
2065 |   };
2066 |
2067 |   // Report user handler
2068 |   const handleReportUser = async (reason: string, details?: string) => {
2069 |     if (!user?.uid || !selectedUser?.id) {
2070 |       toast.error("Unable to report user");
2071 |       return;
2072 |     }
2073 |     try {
2074 |       await reportUser(user.uid, String(selectedUser.id), reason, details);
2075 |       toast.success("User reported. Thank you for helping keep PaceMatch safe.");
2076 |       setShowProfileView(false);
2077 |       setSelectedUser(null);
2078 |     } catch (error: any) {
2079 |       console.error("Error reporting user:", error);
2080 |       toast.error(error.message || "Failed to report user. Please try again.");
2081 |     }
2082 |   };
2083 |
2084 |   // Block user handler
2085 |   const handleBlockUser = async () => {
2086 |     if (!user?.uid || !selectedUser?.id) {
2087 |       toast.error("Unable to block user");
2088 |       return;
2089 |     }
2090 |     try {
2091 |       await blockUser(user.uid, String(selectedUser.id));
2092 |       toast.success("User has been blocked");
2093 |       setShowProfileView(false);
2094 |       setSelectedUser(null);
2095 |       // Refresh to remove blocked user from view
2096 |       setTimeout(() => window.location.reload(), 1000);
2097 |     } catch (error: any) {
2098 |       console.error("Error blocking user:", error);
2099 |       toast.error(error.message || "Failed to block user. Please try again.");
2100 |     }
2101 |   };
2102 |
2103 |   // Auto-follow user with camera offset in navigation-style mode
2104 |   useEffect(() => {
2105 |     if (isNavigationStyle && isWazeMode && location && mapRef && isLoaded && window.google) {
2106 |       const updateTimer = setTimeout(() => {
2107 |         const cameraCenter = calculateCameraOffset(
2108 |           location.lat,
2109 |           location.lng,
2110 |           userHeading || 0,
2111 |           0.002
2112 |         );
2113 |
2114 |         mapRef.panTo(cameraCenter);
2115 |         mapRef.setTilt(67.5);
2116 |         if (userHeading !== null) {
2117 |           mapRef.setHeading(userHeading);
2118 |           setMapHeading(userHeading);
2119 |         }
2120 |       }, 1000);
2121 |
2122 |       return () => clearTimeout(updateTimer);
2123 |     }
2124 |   }, [location, isNavigationStyle, isWazeMode, userHeading, mapRef, isLoaded]);
2125 |
```

```
2126 |   // Track other users' movement trails
2127 |   useEffect(() => {
2128 |     if (nearbyUsers.length === 0) return;
2129 |
2130 |     // Batch all trail updates into a single state update
2131 |     setUserTrails((prev) => {
2132 |       const updated = { ...prev };
2133 |       let hasChanges = false;
2134 |
2135 |       nearbyUsers.forEach((userData: any) => {
2136 |         if (userData.lat && userData.lng) {
2137 |           const userId = userData.id;
2138 |           const currentTrail = prev[userId] || [];
2139 |           const lastPoint = currentTrail[currentTrail.length - 1];
2140 |
2141 |           if (!lastPoint ||
2142 |               (Math.abs(lastPoint.lat - userData.lat) > 0.0001 ||
2143 |                Math.abs(lastPoint.lng - userData.lng) > 0.0001)) {
2144 |             const newTrail = [...currentTrail, { lat: userData.lat, lng: userData.lng }];
2145 |             updated[userId] = newTrail.slice(-50);
2146 |             hasChanges = true;
2147 |           }
2148 |         }
2149 |       });
2150 |
2151 |       // Only return new object if there were actual changes
2152 |       return hasChanges ? updated : prev;
2153 |     });
2154 |   }, [nearbyUsers]);
2155 |
2156 |   // Handle focusFriend from navigation (when coming from Index page)
2157 |   useEffect(() => {
2158 |     if (focusFriend && focusFriend.lat && focusFriend.lng && mapRef && isLoaded &&
2159 | window.google) {
2160 |       const friendCenter = { lat: focusFriend.lat, lng: focusFriend.lng };
2161 |       setMapCenter(friendCenter);
2162 |       mapRef.panTo(friendCenter);
2163 |       mapRef.setCenter(friendCenter);
2164 |       setMapZoom(16); // Zoom in a bit to focus on friend
2165 |
2166 |       // Clear the focusFriend from state after handling
2167 |       navigate("/map", { replace: true, state: {} });
2168 |     }
2169 |   }, [focusFriend, mapRef, isLoaded]);
2170 |
2171 |   // Center map on user location when both map and location are ready (Pokemon Go style)
2172 |   // This runs immediately when location becomes available
2173 |   useEffect(() => {
2174 |     const currentLocation = location || initialLocation;
2175 |     if (!currentLocation || !currentLocation.lat || !currentLocation.lng || !mapRef || !isLoaded
2176 | || !window.google) {
2177 |     }
2178 |
2179 |     // Don't update center if we're focusing on a friend
2180 |     if (focusFriend) {
2181 |       return;
2182 |     }
2183 |
2184 |     console.log("Ø=ÜÍ Centering map on user location:", currentLocation);
2185 |
2186 |     // Center and zoom immediately when location is available (Pokemon Go style)
2187 |     const zoomLevel = 17; // Close zoom for Pokemon Go-like experience
2188 |     setMapCenter({ lat: currentLocation.lat, lng: currentLocation.lng });
2189 |     setMapZoom(zoomLevel);
2190 |     mapRef.setCenter({ lat: currentLocation.lat, lng: currentLocation.lng });
2191 |     mapRef.panTo({ lat: currentLocation.lat, lng: currentLocation.lng });
2192 |     mapRef.setZoom(zoomLevel);
2193 |   }, [location, initialLocation, mapRef, isLoaded, focusFriend]);
2194 |
2195 |   // Update map center when location changes (continuous tracking)
2196 |   // When workout is active, always keep map centered on user (locked perspective)
```

```
  useEffect(() => {
    const currentLocation = location || initialLocation;
    if (!currentLocation || !currentLocation.lat || !currentLocation.lng || !mapRef || !isLoaded || !window.google) {
      return;
    }

    // Don't update center if we're focusing on a friend
    if (focusFriend) {
      return;
    }

    // Debounce map updates to prevent excessive re-renders
    const updateTimer = setTimeout(() => {
      if (!isNavigationStyle) {
        // Always update center to user location (Pokemon Go style)
        setMapCenter({ lat: currentLocation.lat, lng: currentLocation.lng });

        // Center map on user - always follow user location
        if (isActive) {
          // When active, always lock to user's location - no dragging allowed
          mapRef.setCenter({ lat: currentLocation.lat, lng: currentLocation.lng });
          mapRef.panTo({ lat: currentLocation.lat, lng: currentLocation.lng });
        } else {
          // When inactive (Beacon Mode), still center on user but allow panning
          // Auto-center on user location (Pokemon Go style)
          mapRef.panTo({ lat: currentLocation.lat, lng: currentLocation.lng });
        }
      } else if (isNavigationStyle) {
        const cameraCenter = calculateCameraOffset(currentLocation.lat, currentLocation.lng, userHeading || 0);
        setMapCenter(cameraCenter);
        if (isActive) {
          // When active, lock to camera center (user position with offset)
          mapRef.setCenter(cameraCenter);
          mapRef.panTo(cameraCenter);
        } else {
          mapRef.panTo(cameraCenter);
        }
      }
    }, 100); // Small delay to batch updates

    return () => clearTimeout(updateTimer);
  }, [location, initialLocation, isNavigationStyle, userHeading, mapRef, isLoaded, isActive, focusFriend]);

  // Get zoom radius for selected activity and zoom level
  // Based on research recommendations and matching service base radii alignment
  // Tighter zoom levels for better detail and closer view
  const getZoomRadiusForActivity = (
    activity: "running" | "cycling" | "walking",
    zoomLevel: "close" | "medium" | "far"
  ): number => {
    const zoomConfig = {
      running: {
        close: 500,    // 500m - very nearby runners, immediate vicinity (Zoom 16)
        medium: 2000,  // 2km - typical running range, matches base radius (Zoom 14)
        far: 5000      // 5km - wider search, neighborhood view (Zoom 13)
      },
      cycling: {
        close: 2500,   // 2.5km - nearby cyclists, immediate area (Zoom 14)
        medium: 10000, // 10km - typical cycling range, matches base radius (Zoom 12)
        far: 20000     // 20km - wide area search, city-wide view (Zoom 11)
      },
      walking: {
        close: 100,    // 100m - immediate vicinity, street-level detail (Zoom 18.5)
        medium: 500,   // 500m - typical walking range, neighborhood view (Zoom 16)
        far: 1000      // 1km - wider walking area, matches base radius (Zoom 15)
      }
    };
    return zoomConfig[activity][zoomLevel];
  };

  // Update zoom level based on activity and zoom level setting
```

```
2268 |   // Use effectiveLocation so zoom works in both Beacon Mode and Active Workout
2269 |   useEffect(() => {
2270 |     const currentLocation = location || initialLocation;
2271 |     if (currentLocation && mapRef && isLoaded && window.google && selectedActivity) {
2272 |       const radius = getZoomRadiusForActivity(selectedActivity, zoomLevel);
2273 |       const calculatedZoom = calculateZoomFromMeters(radius);
2274 |       setMapZoom(calculatedZoom);
2275 |       mapRef.setZoom(calculatedZoom);
2276 |       if (!isNavigationStyle) {
2277 |         mapRef.panTo({ lat: currentLocation.lat, lng: currentLocation.lng });
2278 |       }
2279 |     }
2280 |   }, [selectedActivity, zoomLevel, location, initialLocation, mapRef, isLoaded,
2281 | isNavigationStyle]);
2282 |   // Lock/unlock map when workout is active
2283 |   useEffect(() => {
2284 |     if (mapRef && isLoaded && window.google) {
2285 |       // Lock map when workout is active (isActive = true)
2286 |       // But allow zoom controls so user can zoom in/out on their location
2287 |       mapRef.setOptions({
2288 |         draggable: !isActive, // Disable dragging/panning when workout is active
2289 |         scrollwheel: true, // Allow scroll zoom
2290 |         disableDoubleClickZoom: !!isActive, // Disable double-click zoom when workout is active
2291 |         gestureHandling: isActive ? "cooperative" : "auto", // Allow zoom but not pan when
2292 | workout is active
2293 |         keyboardShortcuts: true, // Allow keyboard zoom
2294 |         zoomControl: false, // Keep zoom controls disabled (we use scrollwheel)
2295 |         panControl: false // Disable pan control buttons
2296 |       });
2297 |
2298 |       // Force center on user when workout is active
2299 |       if (isActive && location && location.lat && location.lng) {
2300 |         mapRef.setCenter({ lat: location.lat, lng: location.lng });
2301 |         mapRef.panTo({ lat: location.lat, lng: location.lng });
2302 |       }
2303 |
2304 |       // Add drag prevention when active
2305 |       if (isActive) {
2306 |         // Remove any existing drag listener
2307 |         if ((mapRef as any)._dragListener) {
2308 |           window.google.maps.event.removeListener((mapRef as any)._dragListener);
2309 |         }
2310 |
2311 |         // Listen to drag events and immediately recenter if user tries to drag
2312 |         // Cast mapRef to google.maps.Map to access addListener method
2313 |         const googleMap = mapRef as any as google.maps.Map;
2314 |         const dragListener = window.google.maps.event.addListener(googleMap, 'drag', () => {
2315 |           if (isActive && location && location.lat && location.lng) {
2316 |             // Immediately snap back to user location
2317 |             mapRef.setCenter({ lat: location.lat, lng: location.lng });
2318 |           }
2319 |         });
2320 |
2321 |         // Store listener for cleanup
2322 |         (mapRef as any)._dragListener = dragListener;
2323 |       } else {
2324 |         // Remove drag listener when inactive
2325 |         if ((mapRef as any)._dragListener) {
2326 |           window.google.maps.event.removeListener((mapRef as any)._dragListener);
2327 |           delete (mapRef as any)._dragListener;
2328 |         }
2329 |       }
2330 |     }
2331 |
2332 |     // Cleanup
2333 |     return () => {
2334 |       if (mapRef && (mapRef as any)._dragListener && window.google) {
2335 |         window.google.maps.event.removeListener((mapRef as any)._dragListener);
2336 |       }
2337 |     };
2338 |   }, [isActive, mapRef, isLoaded, location]);
```

```
2339 |   const onMapLoad = useCallback((map: any) => {
2340 |     setMapRef(map);
2341 |     if (window.google && window.google.maps) {
2342 |       map.setOptions({
2343 |         mapTypeId: window.google.maps.MapTypeId.ROADMAP
2344 |       });
2345 |
2346 |       // Lock map when workout is active (disable dragging but allow zoom)
2347 |       if (isActive) {
2348 |         map.setOptions({
2349 |           draggable: false, // Disable dragging/panning
2350 |           scrollwheel: true, // Allow scroll zoom
2351 |           disableDoubleClickZoom: true, // Disable double-click zoom
2352 |           gestureHandling: "cooperative", // Allow zoom gestures but not pan
2353 |           panControl: false, // Disable pan control buttons
2354 |           keyboardShortcuts: true // Allow keyboard zoom
2355 |         });
2356 |       }
2357 |
2358 |       // Center map on user location when map loads (Pokemon Go style)
2359 |       const currentLocation = location || initialLocation;
2360 |       if (currentLocation && currentLocation.lat && currentLocation.lng) {
2361 |         const zoomLevel = 17; // Close zoom for Pokemon Go-like experience
2362 |         console.log("Ø=Ýúþ  Map loaded, centering on user location:", currentLocation);
2363 |         map.setCenter({ lat: currentLocation.lat, lng: currentLocation.lng });
2364 |         map.panTo({ lat: currentLocation.lat, lng: currentLocation.lng });
2365 |         map.setZoom(zoomLevel);
2366 |         setMapCenter({ lat: currentLocation.lat, lng: currentLocation.lng });
2367 |         setMapZoom(zoomLevel);
2368 |       }
2369 |     }
2370 |   }, [isActive, location, initialLocation]);
2371 |
2372 |   const onMarkerClick = (userData: any) => {
2373 |     // Ensure user data structure matches ProfileView expectations
2374 |     // Use 'name' field from Firebase (username), not displayName from Google
2375 |     setSelectedUser({
2376 |       id: userData.id,
2377 |       name: userData.name || "User", // Always use username from Firebase, never displayName
2378 |       distance: userData.distance || formatDistance(userData.distanceValue || 0),
2379 |       activity: userData.activity || "Running",
2380 |       avatar: getProfilePictureUrl(userData.photoURL, userData.avatar, userData.name),
2381 |       photos: userData.photos || [],
2382 |       bio: userData.bio || ""
2383 |     });
2384 |     setShowProfileView(true); // Open ProfileView directly instead of MatchActionsModal
2385 |   };
2386 |
2387 |   const handleUpdateSearchFilter = async (newFilter: SearchFilter) => {
2388 |     setSearchFilter(newFilter);
2389 |     setShowFilterModal(false);
2390 |
2391 |     // Save to Firebase
2392 |     if (user) {
2393 |       try {
2394 |         await updateUserProfile(user.uid, { searchFilter: newFilter });
2395 |         toast.success(`Now finding: ${newFilter === "all" ? "All Levels" : newFilter}`);
2396 |       } catch (error) {
2397 |         console.error("Error updating search filter:", error);
2398 |         toast.error("Failed to update filter");
2399 |       }
2400 |     }
2401 |   };
2402 |
2403 |   const handleDeclineMatch = (userId: string) => {
2404 |     // Set 3-day cooldown
2405 |     const cooldownUntil = Date.now() + (3 * 24 * 60 * 60 * 1000);
2406 |     const updated = { ...declinedUsers, [userId]: cooldownUntil };
2407 |     setDeclinedUsers(updated);
2408 |     localStorage.setItem("declinedUsers", JSON.stringify(updated));
2409 |     setShowProfileView(false);
```

```
2410 |     setSelectedUser(null);
2411 |   };
2412 |
2413 |   // Poke handlers
2414 |   const handlePoke = async (userId: string) => {
2415 |     if (!user?.uid) {
2416 |       toast.error("You must be logged in to poke someone");
2417 |       return;
2418 |     }
2419 |
2420 |     // Check if user has active workout session (only allow poking during active workout)
2421 |     if (!isActive) {
2422 |       toast.error("You must have an active workout session to poke someone");
2423 |       return;
2424 |     }
2425 |
2426 |     try {
2427 |       await sendPoke(user.uid, userId);
2428 |       setHasPokedUsers(prev => ({ ...prev, [userId]: true }));
2429 |       toast.success("Poke sent! They'll be notified.");
2430 |     } catch (error) {
2431 |       console.error("Error sending poke:", error);
2432 |       toast.error("Failed to send poke. Please try again.");
2433 |     }
2434 |   };
2435 |
2436 |   const handleAcceptPoke = async (fromUserId: string) => {
2437 |     if (!user?.uid) return;
2438 |
2439 |     try {
2440 |       await acceptPoke(user.uid, fromUserId);
2441 |       toast.success("Poke accepted!");
2442 |       setShowPokeModal(false);
2443 |     } catch (error) {
2444 |       console.error("Error accepting poke:", error);
2445 |       toast.error("Failed to accept poke.");
2446 |     }
2447 |   };
2448 |
2449 |   const handleDismissPoke = async (fromUserId: string) => {
2450 |     if (!user?.uid) return;
2451 |
2452 |     try {
2453 |       await dismissPoke(user.uid, fromUserId);
2454 |       toast("Poke dismissed");
2455 |       setShowPokeModal(false);
2456 |     } catch (error) {
2457 |       console.error("Error dismissing poke:", error);
2458 |       toast.error("Failed to dismiss poke.");
2459 |     }
2460 |   };
2461 |
2462 |   const getCooldownDays = (userId: string): number => {
2463 |     const cooldownUntil = declinedUsers[userId];
2464 |     if (!cooldownUntil) return 0;
2465 |     const now = Date.now();
2466 |     const diff = cooldownUntil - now;
2467 |     if (diff <= 0) return 0;
2468 |     return Math.ceil(diff / (1000 * 60 * 60 * 24));
2469 |   };
2470 |
2471 |   const onInfoWindowClose = () => {
2472 |     setSelectedUser(null);
2473 |   };
2474 |
2475 |   const handleCenterOnUserMap = (userData: any) => {
2476 |     if (mapRef && userData) {
2477 |       const newCenter = { lat: userData.lat, lng: userData.lng };
2478 |       mapRef.panTo(newCenter);
2479 |       mapRef.setZoom(16);
2480 |       setTimeout(() => {
```

```
2481 |         setMapCenter(newCenter);
2482 |         setMapZoom(16);
2483 |       }, 100);
2484 |       setSelectedUser(userData);
2485 |     }
2486 |   };
2487 |
2488 |   // Fit all nearby users on the map
2489 |   const handleFitAllUsers = () => {
2490 |     if (!mapRef || !isLoaded || nearbyUsers.length === 0) {
2491 |       toast.info("No nearby users to display");
2492 |       return;
2493 |     }
2494 |
2495 |     try {
2496 |       const bounds = new window.google.maps.LatLngBounds();
2497 |
2498 |       // Add all nearby user locations to bounds
2499 |       nearbyUsersWithAdjustedPositions.forEach((userData: any) => {
2500 |         if (userData.lat && userData.lng) {
2501 |           bounds.extend(new window.google.maps.LatLng(userData.lat, userData.lng));
2502 |         }
2503 |       });
2504 |
2505 |       // Include current user location
2506 |       const currentLocation = location || initialLocation;
2507 |       if (currentLocation && currentLocation.lat && currentLocation.lng) {
2508 |         bounds.extend(new window.google.maps.LatLng(currentLocation.lat, currentLocation.lng));
2509 |       }
2510 |
2511 |       // Fit bounds with padding
2512 |       mapRef.fitBounds(bounds);
2513 |       mapRef.setOptions({
2514 |         padding: { top: 100, right: 100, bottom: 200, left: 100 }
2515 |       });
2516 |
2517 |       toast.success(`Showing ${nearbyUsers.length} nearby user${nearbyUsers.length !== 1 ? 's' :
     '')`);
2518 |     } catch (error) {
2519 |       console.error("Error fitting users to bounds:", error);
2520 |       toast.error("Could not fit all users on map");
2521 |     }
2522 |   };
2523 |
2524 |
2525 |   if (!googleMapsApiKey) {
2526 |     return (
2527 |       <div className="h-screen flex items-center justify-center p-4">
2528 |         <div className="bg-destructive/10 border border-destructive/20 text-destructive px-4
     py-3 rounded-lg text-sm">
2529 |           Google Maps API key is missing. Please check your .env file.
2530 |         </div>
2531 |       </div>
2532 |     );
2533 |   }
2534 |
2535 |   if (loadError) {
2536 |     return (
2537 |       <div className="h-screen flex items-center justify-center p-4">
2538 |         <div className="bg-destructive/10 border border-destructive/20 text-destructive px-4
     py-3 rounded-lg text-sm">
2539 |           Error loading Google Maps: {loadError.message}
2540 |         </div>
2541 |       </div>
2542 |     );
2543 |   }
2544 |
2545 |   if (!isLoaded) {
2546 |     return (
2547 |       <div className="h-screen flex items-center justify-center">
2548 |         <CircularProgress />
2549 |       </div>
2550 |     );
2551 |   }
```

```
2552 |
2553 |   return (
2554 |     <div className="relative h-screen w-full overflow-hidden bg-muted pb-20">
2555 |       {/* People Sidebar - Toggleable */}
2556 |       <AnimatePresence>
2557 |         {showSidebar && (
2558 |           <>
2559 |             {/* Overlay - Click outside to close */}
2560 |             <motion.div
2561 |               initial={{ opacity: 0 }}
2562 |               animate={{ opacity: 1 }}
2563 |               exit={{ opacity: 0 }}
2564 |               onClick={() => setShowSidebar(false)}
2565 |               className="fixed inset-0 bg-black/20 z-20"
2566 |             />
2567 |             <motion.div
2568 |               initial={{ x: -320 }}
2569 |               animate={{ x: 0 }}
2570 |               exit={{ x: -320 }}
2571 |               transition={{ type: "spring", damping: 25, stiffness: 300 }}
2572 |               className="fixed left-0 top-0 bottom-0 w-80 bg-card border-r border-border shadow-
elevation-3 z-30 flex fl...
2573 |               onClick={(e) => e.stopPropagation()}
2574 |             >
2575 |               <div className="p-6 space-y-5 flex-1 overflow-y-auto min-h-0 flex flex-col">
2576 |           {/* Header */}
2577 |           <div className="flex items-center justify-between">
2578 |             <div>
2579 |               <h2 className="text-2xl font-bold text-foreground">Activity Feed</h2>
2580 |               <p className="text-sm text-muted-foreground mt-1">
2581 |                 Requests, messages & matches
2582 |               </p>
2583 |             </div>
2584 |           </div>
2585 |
2586 |           {/* Friend Requests Section */}
2587 |           <div className="space-y-2">
2588 |             <div className="flex items-center justify-between">
2589 |               <h3 className="text-lg font-semibold text-foreground">Friend Requests</h3>
2590 |               <BadgeCounter count={friendRequests.incoming.length} variant="default" size="sm" />
2591 |             </div>
2592 |             {loadingFriendRequestUsers ? (
2593 |               <div className="flex items-center justify-center py-4">
2594 |                 <CircularProgress size={24} />
2595 |               </div>
2596 |             ) : friendRequests.incoming.length > 0 ? (
2597 |               <div className="space-y-2">
2598 |                 {friendRequests.incoming.slice(0, 3).map((userId) => {
2599 |                   const userData = friendRequestUsers[userId];
2600 |                   const userName = userData?.name || "Unknown User";
2601 |                   const userAvatar = userData?.photoURL;
2602 |                   const userActivity = userData?.activity;
2603 |                   const userFitnessLevel = userData?.fitnessLevel;
2604 |
2605 |                   return (
2606 |                     <motion.div
2607 |                       key={userId}
2608 |                       initial={{ opacity: 0, x: -10 }}
2609 |                       animate={{ opacity: 1, x: 0 }}
2610 |                       className="p-3 bg-muted/50 rounded-lg border border-border hover:bg-muted
transition-colors cursor...
2611 |                       onClick={() => {
2612 |                         setShowSidebar(false); // Auto-close sidebar
2613 |                         navigate("/friends", { state: { tab: "requests" } });
2614 |                       }}
2615 |                     >
2616 |                       {/* New badge */}
2617 |                       <span className="absolute top-2 right-2 bg-primary text-primary-foreground
text-[9px] font-bold px...
2618 |                         NEW
2619 |                       </span>
2620 |                       <div className="flex items-center gap-3">
2621 |                         <Avatar className="w-10 h-10">
2622 |                           <AvatarImage src={userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(userNam...
```

```
                          <AvatarFallback>{userName.charAt(0).toUpperCase()}</AvatarFallback>
                        </Avatar>
                        <div className="flex-1 min-w-0 pr-8">
                          <p className="text-sm font-medium truncate">
                            {userName} wants to be friends
                          </p>
                          <div className="flex items-center gap-2 mt-0.5">
                            {userActivity && (
                              <span className="text-xs text-muted-foreground capitalize">
                                {userActivity}
                              </span>
                            )}
                            {userFitnessLevel && (
                              <>
                                {userActivity && <span className="text-xs text-muted-
foreground">•</span>}
                                <span className="text-xs text-muted-foreground capitalize">
                                  {userFitnessLevel}
                                </span>
                              </>
                            )}
                          </div>
                        </div>
                      </div>
                    </motion.div>
                  );
                })}
              </div>
            ) : (
              <div className="py-4 text-center">
                <p className="text-sm text-muted-foreground">No friend requests</p>
              </div>
            )}
          </div>

          {/* Message Requests Section */}
          <div className="space-y-2">
            <div className="flex items-center justify-between">
              <h3 className="text-lg font-semibold text-foreground">Message Requests</h3>
              <BadgeCounter count={messageRequests.length} variant="default" size="sm" />
            </div>
            {loadingMessageRequestUsers ? (
              <div className="flex items-center justify-center py-4">
                <CircularProgress size={24} />
              </div>
            ) : messageRequests.length > 0 ? (
              <div className="space-y-2">
                {messageRequests.slice(0, 3).map((conv) => {
                  const userData = messageRequestUsers[conv.otherUserId];
                  const userName = userData?.name || "Unknown User";
                  const userAvatar = userData?.photoURL;

                  return (
                    <motion.div
                      key={conv.conversationId}
                      initial={{ opacity: 0, x: -10 }}
                      animate={{ opacity: 1, x: 0 }}
                      className="p-3 bg-muted/50 rounded-lg border border-border hover:bg-muted
transition-colors cursor...  onClick={async () => {
                        setShowSidebar(false); // Auto-close sidebar
                        if (userData && user?.uid) {
                          // Automatically accept the message request by marking messages as read
                          try {
                            await markMessagesAsRead(user.uid, conv.otherUserId);
                            // Refresh message requests list immediately
                            const conversations = await getUserConversations(user.uid);
                            const friends = userProfile?.friends || [];
                            const requests = conversations.filter(c =>
                              c.unreadCount > 0 && !friends.some((f: any) => String(f) ===
c.otherUserId)
                            );
                            setMessageRequests(requests);
                          } catch (error) {
```

```
                            console.error("Error accepting message request:", error);
                          }

                          navigate("/chat", {
                            state: {
                              user: {
                                id: conv.otherUserId,
                                name: userData.name || "Unknown",
                                avatar: getProfilePictureUrl(userData.photoURL, userData.avatar,
userData.name)
                              }
                            }
                          });
                        }
                      }}
                    >
                      {/* New badge */}
                      <span className="absolute top-2 right-2 bg-primary text-primary-foreground
text-[9px] font-bold px...
                        NEW
                      </span>
                      <div className="flex items-center gap-3">
                        <Avatar className="w-10 h-10">
                          <AvatarImage src={userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(userNam<AvatarFallback>{userName.charAt(0).toUpperCase()}</AvatarFallback>
                        </Avatar>
                        <div className="flex-1 min-w-0 pr-8">
                          <p className="text-sm font-medium truncate">
                            Message from {userName}
                          </p>
                          <p className="text-xs text-muted-foreground truncate mt-0.5">
                            {conv.lastMessage || "No message preview"}
                          </p>
                        </div>
                        {conv.unreadCount > 0 && (
                          <BadgeCounter count={conv.unreadCount} variant="default" size="sm" />
                        )}
                      </div>
                    </motion.div>
                  );
                })}
              </div>
            ) : (
              <div className="py-4 text-center">
                <p className="text-sm text-muted-foreground">No message requests</p>
              </div>
            )}
          </div>

          {/* Incoming Messages Section */}
          <div className="space-y-2">
            <div className="flex items-center justify-between">
              <h3 className="text-lg font-semibold text-foreground">New Messages</h3>
              <BadgeCounter count={incomingMessages.length} variant="default" size="sm" />
            </div>
            {loadingIncomingMessageUsers ? (
              <div className="flex items-center justify-center py-4">
                <CircularProgress size={24} />
              </div>
            ) : incomingMessages.length > 0 ? (
              <div className="space-y-2">
                {incomingMessages.slice(0, 5).map((conv) => {
                  const userData = incomingMessageUsers[conv.otherUserId];
                  const userName = userData?.name || "Unknown User";
                  const userAvatar = userData?.photoURL;

                  return (
                    <motion.div
                      key={conv.conversationId}
                      initial={{ opacity: 0, x: -10 }}
                      animate={{ opacity: 1, x: 0 }}
                      className="p-3 bg-muted/50 rounded-lg border border-border hover:bg-muted
transition-colors cursor...
                      onClick={() => {
```

```
                        setShowSidebar(false); // Auto-close sidebar
                        if (userData) {
                          navigate("/chat", {
                            state: {
                              user: {
                                id: conv.otherUserId,
                                name: userData.name || "Unknown",
                                avatar: getProfilePictureUrl(userData.photoURL, userData.avatar,
userData.name)
                              }
                            }
                          });
                        }
                      }}
                    >
                      <div className="flex items-center gap-3">
                        <Avatar className="w-10 h-10">
                          <AvatarImage src={userAvatar || `https://ui-avatars.com/api/?
name=${encodeURIComponent(userNam<AvatarFallback>{userName.charAt(0).toUpperCase()}</AvatarFallback>
                        </Avatar>
                        <div className="flex-1 min-w-0 pr-8">
                          <p className="text-sm font-medium truncate">
                            {userName}
                          </p>
                          <p className="text-xs text-muted-foreground truncate mt-0.5">
                            {conv.lastMessage || "No message preview"}
                          </p>
                        </div>
                        {conv.unreadCount > 0 && (
                          <BadgeCounter count={conv.unreadCount} variant="default" size="sm" />
                        )}
                      </div>
                    </motion.div>
                  );
                })}
              </div>
            ) : (
              <div className="py-4 text-center">
                <p className="text-sm text-muted-foreground">No new messages</p>
              </div>
            )}
          </div>

              </div>
            </motion.div>
          </>
        )}
      </AnimatePresence>

      {/* Google Maps */}
      <motion.div
        className="absolute inset-0"
        style={{
          left: showSidebar ? '320px' : '0',
          width: showSidebar ? 'calc(100% - 320px)' : '100%',
          transition: 'left 0.3s ease-in-out, width 0.3s ease-in-out'
        }}
      >
        <GoogleMap
          mapContainerStyle={mapContainerStyle}
          center={mapCenter}
          zoom={mapZoom}
          tilt={(isWazeMode) ? mapTilt : 0}
          heading={(isWazeMode) ? mapHeading : 0}
          onLoad={onMapLoad}
          options={{
            disableDefaultUI: false,
            zoomControl: false, // Zoom controls disabled
            streetViewControl: false,
            mapTypeControl: false, // Map/Satellite toggle disabled
            fullscreenControl: false, // Fullscreen button disabled
            mapTypeId: isLoaded && window.google?.maps ? window.google.maps.MapTypeId.ROADMAP :
undefined,
```

```
2836 |             // Lock map when workout is active (isActive = true)
2837 |             draggable: !isActive, // Disable dragging when workout is active
2838 |             scrollwheel: true, // Allow scroll zoom even when workout is active
2839 |             disableDoubleClickZoom: !!isActive, // Disable double-click zoom when workout is
     active
2840 |             gestureHandling: (isActive ? "cooperative" : "auto") as "cooperative" | "auto" |
     "greedy" | "none", // Allow
2841 |             panControl: false, // Disable pan control buttons
2842 |             keyboardShortcuts: true // Allow keyboard zoom (+/- keys)
2843 |           }}
2844 |         >
2845 |           {/* Current user's activity trail */}
2846 |           {isActive && locationHistory.length > 1 && (
2847 |             <Polyline
2848 |               path={locationHistory}
2849 |               options={{
2850 |                 strokeColor: "#1976d2",
2851 |                 strokeOpacity: 0.6,
2852 |                 strokeWeight: 4,
2853 |                 geodesic: true
2854 |               }}
2855 |             />
2856 |           )}
2857 |
2858 |           {/* Other users' movement trails */}
2859 |           {Object.entries(userTrails).map(([userId, trail]) => {
2860 |             if (trail.length > 1) {
2861 |               return (
2862 |                 <Polyline
2863 |                   key={`trail-${userId}`}
2864 |                   path={trail}
2865 |                   options={{
2866 |                     strokeColor: "#d32f2f",
2867 |                     strokeOpacity: 0.4,
2868 |                     strokeWeight: 3,
2869 |                     geodesic: true
2870 |                   }}
2871 |                 />
2872 |               );
2873 |             }
2874 |             return null;
2875 |           })}
2876 |
2877 |           {/* Current user's marker - Always visible when location is available */}
2878 |           {(location || initialLocation) && (location || initialLocation)?.lat && (location ||
     initialLocation)?.lng && ...
2879 |             <Marker
2880 |               position={{ lat: (location || initialLocation)!.lat, lng: (location ||
     initialLocation)!.lng }}
2881 |               title={isActive ? "You are here (Active)" : "You are here"}
2882 |               zIndex={1000}
2883 |               icon={{
2884 |                 url: isActive
2885 |                   ? "https://maps.google.com/mapfiles/ms/icons/blue-dot.png"
2886 |                   : "https://maps.google.com/mapfiles/ms/icons/blue-dot.png",
2887 |                 scaledSize: isActive
2888 |                   ? new window.google.maps.Size(48, 48)
2889 |                   : new window.google.maps.Size(40, 40),
2890 |                 anchor: new window.google.maps.Point(isActive ? 24 : 20, isActive ? 24 : 20)
2891 |               }}
2892 |               animation={isActive && window.google.maps.Animation ?
     window.google.maps.Animation.BOUNCE : undefined}
2893 |             />
2894 |             {/* Circle around user location for better visibility when active */}
2895 |             {isActive && location && window.google.maps && (
2896 |               <Marker
2897 |                 position={{ lat: location.lat, lng: location.lng }}
2898 |                 icon={{
2899 |                   path: window.google.maps.SymbolPath.CIRCLE,
2900 |                   scale: 8,
2901 |                   fillColor: "#2196F3",
2902 |                   fillOpacity: 0.3,
2903 |                   strokeColor: "#1976D2",
2904 |                   strokeWeight: 2,
2905 |                 }}
2906 |
```

```
                  zIndex={999}
                />
              )}
            </>
          )}

          {/* Transparent radius circle overlay showing matching range - only show when workout is active (beacon mode) */}
          {isActive && location && location.lat && location.lng && isLoaded && window.google && selectedActivity && (() => {
            const matchingRadius = calculateMatchingRadius(
              selectedActivity,
              userProfile.radiusPreference || "normal"
            );

            // Activity-specific colors
            const activityColors = {
              walking: {
                fill: "#FFC107",      // Amber/Yellow
                stroke: "#FF9800"     // Orange
              },
              running: {
                fill: "#4CAF50",      // Green
                stroke: "#388E3C"     // Dark Green
              },
              cycling: {
                fill: "#2196F3",      // Blue
                stroke: "#1976D2"     // Dark Blue
              }
            };

            const colors = activityColors[selectedActivity] || activityColors.running;

            return (
              <Circle
                key={`radius-circle-${selectedActivity}-${userProfile.radiusPreference || "normal"}-${location.lat}-${
                center={{ lat: location.lat, lng: location.lng }}
                radius={matchingRadius}
                options={{
                  fillColor: colors.fill,
                  fillOpacity: 0.12,
                  strokeColor: colors.stroke,
                  strokeOpacity: 0.7,
                  strokeWeight: 2,
                  clickable: false,
                  zIndex: 1
                }}
              />
            );
          })()}

          {/* Nearby users' markers with profile pictures */}
          {nearbyUsersWithAdjustedPositions.map((userData: any) => {
            const hasTrail = userTrails[userData.id] && userTrails[userData.id].length > 1;
            const matchScore = userData.matchScore;
            const scorePercent = matchScore ? Math.round(matchScore * 100) : null;

            // Get profile picture URL using utility function for consistent fallback
            const userName = userData.name || "User";
            const profilePictureUrl = getProfilePictureUrl(
              userData.photoURL,
              userData.avatar,
              userName
            );
            const fitnessLevel = userData.fitnessLevel;
            const userActivity = userData.activity || "running";

            // Check if workout is active and if user's activity matches selected activity
            const isSameActivity = isActive && userActivity.toLowerCase() === selectedActivity.toLowerCase();
            const shouldGreyOut = isActive && !isSameActivity;

```

```
2978 |               // Create profile picture icon with fitness level glow, activity badge, and opacity
2979 |               const baseIconSize = hasTrail ? 56 : 48;
2980 |               const iconSize = baseIconSize;
2981 |
2982 |               // Use createProfileIcon directly
2983 |               const profileIconUrl = isLoaded && window.google
2984 |                 ? createProfileIcon(
2985 |                     profilePictureUrl,
2986 |                     userName,
2987 |                     {
2988 |                       size: iconSize,
2989 |                       fitnessLevel,
2990 |                       opacity: shouldGreyOut ? 0.4 : undefined,
2991 |                       activity: userActivity as "running" | "cycling" | "walking"
2992 |                     }
2993 |                   )
2994 |                 : null;
2995 |
2996 |               // Create Google Maps icon object
2997 |               const profileIcon = profileIconUrl && isLoaded && window.google ? (() => {
2998 |                 const hasGlow = !!getFitnessLevelColors(fitnessLevel);
2999 |                 const badgeSpace = userActivity ? 18 + 8 : 0; // ACTIVITY_BADGE_SIZE +
ACTIVITY_BADGE_OFFSET *3000 |                 const glowSize = hasGlow ? 8 : 0;
3001 |                 const actualSize = iconSize + (glowSize * 2) + badgeSpace;
3002 |
3003 |                 return {
3004 |                   url: profileIconUrl,
3005 |                   scaledSize: new window.google.maps.Size(actualSize, actualSize),
3006 |                   anchor: new window.google.maps.Point(actualSize / 2, actualSize / 2),
3007 |                   origin: new window.google.maps.Point(0, 0)
3008 |                 };
3009 |               })() : null;
3010 |
3011 |               // Set z-index: same-activity users on top, then different-activity, then by trail
3012 |               let markerZIndex = hasTrail ? 100 : 99;
3013 |               if (isActive) {
3014 |                 if (isSameActivity) {
3015 |                   markerZIndex = hasTrail ? 102 : 101; // Same activity users on top
3016 |                 } else {
3017 |                   markerZIndex = hasTrail ? 98 : 97; // Different activity users below
3018 |                 }
3019 |               }
3020 |
3021 |               return isLoaded && window.google && profileIcon ? (
3022 |                 <Marker
3023 |                   key={userData.id}
3024 |                   position={{ lat: userData.lat, lng: userData.lng }}
3025 |                   title={`${userName}${hasTrail ? " (Moving)" : ""}${scorePercent ? ` -
${scorePercent}% match` : ""}`}
3026 |                   onClick={() => onMarkerClick(userData)}
3027 |                   icon={profileIcon}
3028 |                   animation={hasTrail && window.google.maps.Animation ?
window.google.maps.Animation.DROP : undefined}
3029 |                   zIndex={markerZIndex}
3030 |                   optimized={false}
3031 |                 />
3032 |               ) : null;
3033 |             })}
3034 |
3035 |           {/* Info window for selected user */}
3036 |           {selectedUser && selectedUser.lat && selectedUser.lng && (
3037 |             <InfoWindow
3038 |               position={{
3039 |                 lat: selectedUser.lat,
3040 |                 lng: selectedUser.lng
3041 |               }}
3042 |               onCloseClick={onInfoWindowClose}
3043 |             >
3044 |               <Box sx={{ padding: 1, minWidth: 200 }}>
3045 |                 <Typography variant="subtitle1" fontWeight="bold" sx={{ mb: 1 }}>
3046 |                   {selectedUser.name || "User"}
3047 |                 </Typography>
3048 |                 <Typography variant="body2" color="text.secondary" sx={{ mb: 0.5 }}>
```

```
                  {getMarkerColor(selectedUser.activity)} {selectedUser.activity || "Active"}
                </Typography>
                <Typography variant="body2" color="text.secondary" sx={{ mb: 1.5 }}>
                  {formatDistance(selectedUser.distanceValue || selectedUser.distance || 0)} away
                </Typography>
                <MuiButton
                  variant="contained"
                  size="small"
                  fullWidth
                  startIcon={<CenterFocusStrong />}
                  onClick={() => handleCenterOnUserMap(selectedUser)}
                  sx={{ mt: 1 }}
                >
                  Center on Location
                </MuiButton>
              </Box>
            </InfoWindow>
          )}
        </GoogleMap>
      </motion.div>

      {/* Notification Banner */}
      <NotificationBanner
        show={showNotification}
        onDismiss={() => {
          setShowNotification(false);
          notificationShownRef.current = true; // Mark as shown so it doesn't appear again
        }}
        onTap={handleNotificationBannerTap}
      />

      {/* Workout Summary Modal */}
      <WorkoutSummaryModal
        isOpen={showSummary}
        onClose={() => setShowSummary(false)}
        onSave={handleSaveWorkout}
        onDiscard={handleDiscardWorkout}
        activity={selectedActivity}
        duration={elapsedTime}
        distance={distance}
        avgSpeed={avgSpeed}
        useMetric={useMetric}
        nearbyUsers={workoutNearbyUsers}
        pokes={workoutPokes}
        locationHistory={locationHistory}
      />

      {/* Inactivity Warning Modal */}
      <InactivityWarningModal
        open={showInactivityWarning}
        onOpenChange={setShowInactivityWarning}
        onDismiss={handleDismissInactivityWarning}
        onPause={handleInactivityPause}
        autoPauseSeconds={120} // 2 minutes
      />

      {/* Header - Beacon Mode or Active Workout */}
      <motion.div
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b
border-border/50"
        style={{
          paddingTop: 'env(safe-area-inset-top)',
        }}
      >
        <div className="max-w-2xl mx-auto px-6 py-5">
          <div className="flex items-center justify-between">
            <h1 className="text-3xl font-bold">
              {isActive ? "Active Workout" : "Beacon Mode"}
            </h1>
            {/* Notification Bell - Yellow when there are notifications */}
```

```
              <NotificationBell
                unreadCount={unreadCount}
                onClick={() => setShowNotificationDrawer(true)}
                variant="light"
              />
            </div>
          </div>
        </motion.div>

        {/* Right Side Controls - No Background */}
        <div className="absolute top-1/2 -translate-y-1/2 right-4 flex flex-col gap-3 z-10">
          {/* Filter Button */}
          <motion.button
            whileTap={{ scale: 0.95 }}
            onClick={() => setShowFilterModal(true)}
            className={`relative touch-target rounded-full shadow-elevation-3 border-2 ${
              searchFilter !== "all"
                ? "bg-primary text-primary-foreground border-primary"
                : "bg-card text-foreground border-border"
            }`}
            style={{ width: 56, height: 56 }}
            title={`Filter: ${searchFilter === "all" ? "All Levels" :
searchFilter.charAt(0).toUpperCase() + searchFilter....
            >
            <FilterListIcon style={{ fontSize: 28 }} />
            {searchFilter !== "all" && (
              <span className={`absolute -top-1 -right-1 text-white text-[10px] font-bold rounded-
full w-5 h-5 flex items-
                  searchFilter === "beginner"
                    ? "bg-blue-500"
                    : searchFilter === "intermediate"
                    ? "bg-green-500"
                    : "bg-purple-500"
              }`}>
                {searchFilter === "beginner" ? "B" : searchFilter === "intermediate" ? "I" : "P"}
              </span>
            )}
          </motion.button>

          {/* Fit All Users Button */}
          <motion.button
            whileTap={{ scale: 0.95 }}
            onClick={handleFitAllUsers}
            className="touch-target rounded-full shadow-elevation-3 border-2 bg-card text-
foreground border-border hover:b
            style={{ width: 56, height: 56 }}
            title="Fit all users on map"
          >
            <FitScreenIcon style={{ fontSize: 28 }} />
          </motion.button>

          {/* Zoom Level Button - Only visible when workout is active */}
          {isActive && (
            <motion.button
              whileTap={{ scale: 0.95 }}
              onClick={() => {
                // Cycle through zoom levels: close !’ medium !’ far !’ close
                setZoomLevel((current) => {
                  if (current === "close") return "medium";
                  if (current === "medium") return "far";
                  return "close";
                });
              }}
              className={`touch-target rounded-full shadow-elevation-3 border-2 transition-all ${
                zoomLevel !== "medium"
                  ? "bg-primary text-primary-foreground border-primary"
                  : "bg-card text-foreground border-border hover:border-primary"
              }`}
              style={{ width: 56, height: 56 }}
              title={`Zoom: ${zoomLevel === "close" ? "Close" : zoomLevel === "medium" ?
"Medium" : "Far"}`}
              {zoomLevel === "close" ? (
                <ZoomIn style={{ fontSize: 28 }} />
              ) : zoomLevel === "medium" ? (
```

```
              <ViewComfy style={{ fontSize: 28 }} />
            ) : (
              <ZoomOut style={{ fontSize: 28 }} />
            )}
          </motion.button>
        )}

        {/* Poke Notification Button - Only visible when workout is active */}
        {isActive && (
          <motion.button
            whileTap={{ scale: 0.95 }}
            onClick={() => {
              setShowMatchesDrawer(true);
              // Scroll to poked users when opened via poke button
              setTimeout(() => {
                const firstPokedElement = document.querySelector('[data-poked="true"]');
                if (firstPokedElement) {
                  firstPokedElement.scrollIntoView({ behavior: 'smooth', block: 'center' });
                }
              }, 300);
            }}
            className={`relative touch-target rounded-full shadow-elevation-3 border-2 transition-all ${
              showMatchesDrawer && pokes.length > 0
                ? "bg-primary/20 border-primary"
                : "bg-card border-border hover:border-primary"
            }`}
            style={{ width: 56, height: 56 }}
            title={pokes.length > 0 ? `${pokes.length} poke${pokes.length > 1 ? 's' : ''} received` : "No pokes"}
          >
            <NotificationsIcon
              style={{ fontSize: 28 }}
              className="text-foreground"
            />
            {pokes.length > 0 && (
              <span className="absolute -top-1 -right-1 bg-purple-500 text-white text-[10px] font-bold rounded-full w-5 h-5 flex items-center justify-center">
                {pokes.length > 9 ? '9+' : pokes.length}
              </span>
            )}
          </motion.button>
        )}

        {/* People Sidebar Toggle */}
        <motion.button
          whileTap={{ scale: 0.95 }}
          onClick={() => {
            const newState = !showSidebar;
            setShowSidebar(newState);
            // Mark matches as viewed when sidebar is opened
            if (newState) {
              setLastViewedMatchesCount(matches.length);
            }
          }}
          className={`relative touch-target rounded-full shadow-elevation-3 border-2 transition-all ${
            showSidebar
              ? "bg-primary text-primary-foreground border-primary"
              : (unreadMessageCount + friendRequests.incoming.length) > 0
              ? "bg-yellow-400/20 border-yellow-400"
              : "bg-card text-foreground border-border hover:border-primary"
          }`}
          style={{ width: 56, height: 56 }}
          title={showSidebar ? "Hide sidebar" : "Show sidebar"}
        >
          <PeopleIcon style={{ fontSize: 28 }} className={
            !showSidebar && (unreadMessageCount + friendRequests.incoming.length) > 0
              ? "text-yellow-500"
              : ""
          } />
          {/* Badge for unread messages and friend requests */}
          {(unreadMessageCount + friendRequests.incoming.length) > 0 && !showSidebar && (
            <span className="absolute -top-1 -right-1 bg-yellow-500 text-gray-900 text-[10px] font-bold rounded-full min-
              {(unreadMessageCount + friendRequests.incoming.length) > 99 ? '99+' :
(unreadMessageCount + friendRequests...
```

```
3262 |             </span>
3263 |           )}
3264 |         </motion.button>
3265 |
3266 |         {/* Recenter Button - Only visible when workout is not active */}
3267 |         {!isActive && (
3268 |           <motion.button
3269 |             whileTap={{ scale: 0.95 }}
3270 |             onClick={() => {
3271 |               if (location && mapRef && isLoaded && window.google) {
3272 |                 const center = { lat: location.lat, lng: location.lng };
3273 |                 mapRef.panTo(center);
3274 |                 mapRef.setCenter(center);
3275 |                 setMapCenter(center);
3276 |                 toast.success("Map recentered on your location");
3277 |               }
3278 |             }}
3279 |             className="touch-target bg-card text-foreground rounded-full shadow-elevation-3
border-2 border-border hover:
3280 |             style={{ width: 56, height: 56 }}
3281 |             title="Recenter on your location"
3282 |           >
3283 |             <MyLocationMui style={{ fontSize: 28 }} />
3284 |           </motion.button>
3285 |         )}
3286 |       </div>
3287 |
3288 |       {/* Profile View Modal */}
3289 |       <AnimatePresence>
3290 |         {showProfileView && selectedUser && (
3291 |           <ProfileView
3292 |             user={selectedUser}
3293 |             friendStatus={getFriendStatus(selectedUser.id)}
3294 |             cooldownDays={getCooldownDays(selectedUser.id) ||
getCooldownDaysForFriend(selectedUser.id)}
3295 |             onClose={() => {
3296 |               setShowProfileView(false);
3297 |               setSelectedUser(null);
3298 |             }}
3299 |             onSendMessage={() => {
3300 |               setShowProfileView(false);
3301 |               handleSendMessage();
3302 |             }}
3303 |             onAddFriend={() => handleAddFriend(selectedUser.id, selectedUser.name)}
3304 |             onAcceptFriend={() => handleAcceptFriend(selectedUser.id)}
3305 |             onDeclineFriend={() => handleDeclineFriend(selectedUser.id)}
3306 |             onUnfriend={() => handleUnfriend(selectedUser.id)}
3307 |             onPoke={isActive ? () => handlePoke(selectedUser.id) : undefined}
3308 |             hasPoked={hasPokedUsers[selectedUser.id] || false}
3309 |             isWorkoutActive={isActive}
3310 |             onReport={handleReportUser}
3311 |             onBlock={handleBlockUser}
3312 |           />
3313 |         )}
3314 |       </AnimatePresence>
3315 |
3316 |       {/* Message Modal */}
3317 |       <MessageModal
3318 |         show={showMessageModal}
3319 |         userName={selectedUser?.name || ""}
3320 |         onClose={() => setShowMessageModal(false)}
3321 |         onSend={handleMessageSent}
3322 |       />
3323 |
3324 |       {/* Friend Request Modal */}
3325 |       <FriendRequestModal
3326 |         isOpen={showFriendRequestModal}
3327 |         onClose={() => setShowFriendRequestModal(false)}
3328 |         userName={selectedUser?.name || ""}
3329 |         onAccept={() => selectedUser && handleAcceptFriend(selectedUser.id)}
3330 |         onDecline={() => selectedUser && handleDeclineFriend(selectedUser.id)}
3331 |       />
3332 |
```

```
      {/* Poke Modal */}
      {selectedUser && (
        <PokeModal
          isOpen={showPokeModal}
          onClose={() => {
            setShowPokeModal(false);
            setSelectedUser(null);
          }}
          userName={selectedUser.name || "User"}
          userAvatar={selectedUser.avatar}
          userId={selectedUser.id}
          onAccept={() => selectedUser && handleAcceptPoke(selectedUser.id)}
          onDismiss={() => selectedUser && handleDismissPoke(selectedUser.id)}
          onChat={() => {
            handleSendMessage();
            setShowPokeModal(false);
          }}
            onAddFriend={() => {
            handleAddFriend(selectedUser.id, selectedUser.name);
            setShowPokeModal(false);
          }}
        />
      )}

      {/* Stop Activity Confirmation Dialog */}
      <AlertDialog open={showStopConfirmation} onOpenChange={setShowStopConfirmation}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Stop Activity?</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to stop your {selectedActivity} session? Your workout will be saved and you can review it.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel>Cancel</AlertDialogCancel>
            <AlertDialogAction
              onClick={handleConfirmStop}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Stop Activity
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>
      {/* Combined Minimalistic Control - Active State */}
      <AnimatePresence>
        {isActive && !blockingModalOpen && (
          <motion.div
            initial={{ opacity: 0, y: 20, scale: 0.95 }}
            animate={{ opacity: 1, y: 0, scale: 1 }}
            exit={{ opacity: 0, y: 20, scale: 0.95 }}
            transition={{ type: "spring", stiffness: 200, damping: 20 }}
            className="fixed bottom-0 left-0 right-0 px-4 pb-4 z-50"
          >
            <div
              className={`
              bg-card/95 backdrop-blur-md rounded-2xl shadow-elevation-4 border border-border/50 overflow-hidden
              ${
                selectedActivity === "running"
                  ? "border-success/30"
                  : selectedActivity === "cycling"
                  ? "border-primary/30"
                  : "border-warning/30"
              }
            `}
            >
              {/* Stats Row */}
              <div className="px-4 py-2.5 border-b border-border/30">
                <div className="grid grid-cols-3 gap-2">
                  {/* Time */}
                  <div className="text-center">
```

```
                    <div className="text-lg font-bold text-foreground tabular-nums">
                      {formatTime(elapsedTime)}
                    </div>
                    <div className="text-[9px] text-muted-foreground mt-0.5">Time</div>
                  </div>

                  {/* Distance */}
                  <div className="text-center">
                    <div className="text-lg font-bold text-foreground tabular-nums">
                      {convertDistance(distance).value}
                    </div>
                    <div className="text-[9px] text-muted-foreground mt-0.5">{convertDistance(distance).unit}</div>
                  </div>

                  {/* Average Speed */}
                  <div className="text-center">
                    <div className="text-lg font-bold text-foreground tabular-nums">
                      {convertSpeed(avgSpeed).value}
                    </div>
                    <div className="text-[9px] text-muted-foreground mt-0.5">{convertSpeed(avgSpeed).unit}</div>
                  </div>
                </div>

                {/* Buttons Row */}
                <div className="px-3 py-3">
                  <div className="flex items-center justify-center gap-3">
                    {/* Pause Button */}
                    <motion.div whileTap={{ scale: 0.97 }} className="flex-1">
                      <Button
                        onClick={handlePause}
                        className={`
                          w-full h-14 text-base font-extrabold shadow-elevation-4 transition-all duration-300 rounded-xl
                          ${
                            selectedActivity === "running"
                              ? "bg-gradient-to-r from-warning to-warning/90 text-warning-foreground hover:from-warning/90...: selectedActivity === "cycling"
                              ? "bg-gradient-to-r from-secondary to-secondary/90 text-secondary-foreground hover:from-seco...
                              : "bg-gradient-to-r from-secondary to-secondary/90 text-secondary-foreground hover:from-seco...
                          }
                        `}
                      >
                        {isPaused ? (
                          <>
                            <PlayArrowIcon className="mr-2" style={{ fontSize: 24 }} />
                            Resume
                          </>
                        ) : (
                          <>
                            <PauseIcon className="mr-2" style={{ fontSize: 24 }} />
                            Pause
                          </>
                        )}
                      </Button>
                    </motion.div>

                    {/* Stop Button */}
                    <motion.div whileTap={{ scale: 0.97 }} className="flex-1">
                      <Button
                        onClick={() => setShowStopConfirmation(true)}
                        className="w-full h-14 text-base font-extrabold shadow-elevation-4 transition-all duration-300 rou...
                        <StopIcon className="mr-2" style={{ fontSize: 24 }} />
                        Stop
                      </Button>
                    </motion.div>
                  </div>
                </div>
              </div>
            </motion.div>
          )}
```

```
3475 |       </AnimatePresence>
3476 |
3477 |       {/* Combined Minimalistic Control - Inactive State */}
3478 |       <AnimatePresence>
3479 |         {!isActive && (
3480 |           <motion.div
3481 |             initial={{ opacity: 0, y: 20, scale: 0.95 }}
3482 |             animate={{ opacity: 1, y: 0, scale: 1 }}
3483 |             exit={{ opacity: 0, y: 20, scale: 0.95 }}
3484 |             transition={{ type: "spring", stiffness: 200, damping: 20 }}
3485 |             className="absolute left-0 right-0 px-4 z-10"
3486 |             style={{
3487 |               bottom: `calc(5rem + env(safe-area-inset-bottom, 48px))`, // 80px + safe area
3488 |             }}
3489 |           >
3490 |             <div
3491 |               className={`
3492 |               bg-card/95 backdrop-blur-md rounded-2xl shadow-elevation-4 border border-border/50
overflow-hidden
3493 |               ${
3494 |                 selectedActivity === "running"
3495 |                   ? "border-success/30"
3496 |                   : selectedActivity === "cycling"
3497 |                   ? "border-primary/30"
3498 |                   : "border-warning/30"
3499 |               }
3500 |             `}
3501 |             >
3502 |               {/* Stats Row */}
3503 |               <div className="px-4 py-2.5 border-b border-border/30">
3504 |                 <div className="grid grid-cols-3 gap-2">
3505 |                   {/* Time */}
3506 |                   <div className="text-center">
3507 |                     <div className="text-lg font-bold text-foreground tabular-nums">
3508 |                       00:00
3509 |                     </div>
3510 |                     <div className="text-[9px] text-muted-foreground mt-0.5">Time</div>
3511 |                   </div>
3512 |
3513 |                   {/* Average Speed */}
3514 |                   <div className="text-center">
3515 |                     <div className="text-lg font-bold text-foreground tabular-nums">
3516 |                       --
3517 |                     </div>
3518 |                     <div className="text-[9px] text-muted-foreground mt-0.5">Avg. speed (km/h)</
div>
3519 |                   </div>
3520 |
3521 |                   {/* Distance */}
3522 |                   <div className="text-center">
3523 |                     <div className="text-lg font-bold text-foreground tabular-nums">
3524 |                       0.00
3525 |                     </div>
3526 |                     <div className="text-[9px] text-muted-foreground mt-0.5">Distance (km)</div>
3527 |                   </div>
3528 |                 </div>
3529 |               </div>
3530 |
3531 |               {/* Buttons Row */}
3532 |               <div className="px-3 py-3">
3533 |                 <div className="flex items-center justify-center gap-4">
3534 |                   {/* Activity Selector Button (Left) */}
3535 |                   <div className="flex flex-col items-center gap-1.5">
3536 |                     <motion.button
3537 |                       whileTap={{ scale: 0.95 }}
3538 |                       onClick={() => {
3539 |                         // Cycle through available activities
3540 |                         const currentIndex = availableActivities.findIndex(a => a.id ===
selectedActivity);
3541 |                         const nextIndex = (currentIndex + 1) % availableActivities.length;
3542 |                         setSelectedActivity(availableActivities[nextIndex].id as typeof
selectedActivity);
3543 |                       }}
3544 |                       className={`
3545 |                         relative flex flex-col items-center justify-center w-14 h-14 rounded-
full border-2 transition-al...
```

```
                        ${
                          selectedActivity === "running"
                            ? "border-success bg-success/15 shadow-elevation-2"
                            : selectedActivity === "cycling"
                            ? "border-primary bg-primary/15 shadow-elevation-2"
                            : "border-warning bg-warning/15 shadow-elevation-2"
                        }
                      `}
                    >
                      {selectedActivity === "running" && (
                        <DirectionsRunIcon className="text-success" style={{ fontSize: 24 }} />
                      )}
                      {selectedActivity === "cycling" && (
                        <DirectionsBikeIcon className="text-primary" style={{ fontSize: 24 }} />
                      )}
                      {selectedActivity === "walking" && (
                        <DirectionsWalkIcon className="text-warning" style={{ fontSize: 24 }} />
                      )}
                      <CheckCircleIcon
                        className={
                          selectedActivity === "running"
                            ? "text-success"
                            : selectedActivity === "cycling"
                            ? "text-primary"
                            : "text-warning"
                        }
                        style={{ fontSize: 14, position: 'absolute', top: -2, right: -2 }}
                      />
                    </motion.button>
                    <span className="text-[10px] font-medium text-foreground">
                      {availableActivities.find((a) => a.id === selectedActivity)?.label ||
selectedActivity}
                    </span>
                  </div>

                  {/* Start Button (Middle) - Circular */}
                  <div className="flex flex-col items-center gap-1.5">
                    <motion.button
                      whileTap={{ scale: 0.95 }}
                      animate={{ scale: [1, 1.01, 1] }}
                      transition={{ duration: 2, repeat: Infinity, repeatDelay: 3 }}
                      onClick={handleStartStop}
                      className={`
                        flex flex-col items-center justify-center w-16 h-16 rounded-full
border-2 transition-all duratio... ${
                          selectedActivity === "running"
                            ? "bg-gradient-to-r from-success to-success/90 text-success-
foreground border-success hover:...: selectedActivity === "cycling"
                            ? "bg-gradient-to-r from-primary to-primary/90 text-primary-
foreground border-primary hover:...: "bg-gradient-to-r from-warning to-warning/90 text-warning-
foreground border-warning hover:...
                        }
                      `}
                    >
                      <span className="text-base font-extrabold">Start</span>
                    </motion.button>
                  </div>

                  {/* Filter Button (Right) */}
                  <div className="flex flex-col items-center gap-1.5">
                    <motion.button
                      whileTap={{ scale: 0.95 }}
                      onClick={() => setShowFilterModal(true)}
                      className={`
                        relative flex flex-col items-center justify-center w-14 h-14 rounded-
full border-2 transition-al... ${
                          searchFilter !== "all"
                            ? "border-primary bg-primary/15 shadow-elevation-2"
                            : "border-border bg-card/50 hover:bg-secondary"
                        }
                      `}
                      title={`Filter: ${searchFilter === "all" ? "All Levels" :
searchFilter.charAt(0).toUpperCase() + s...
```

```
                      <FilterListIcon
                        className={searchFilter !== "all" ? "text-primary" : "text-muted-
foreground"}
                        style={{ fontSize: 24 }}
                      />
                      {searchFilter !== "all" && (
                        <span className={`absolute -top-1 -right-1 text-white text-[10px] font-
bold rounded-full w-5 h-5...
                          searchFilter === "beginner"
                            ? "bg-blue-500"
                            : searchFilter === "intermediate"
                            ? "bg-green-500"
                            : "bg-purple-500"
                        }`}>
                          {searchFilter === "beginner" ? "B" : searchFilter === "intermediate" ?
"I" : "P"}
                        </span>
                      )}
                    </motion.button>
                    <span className="text-[10px] font-medium text-foreground">Filter</span>
                  </div>
                </div>
              </div>
            </motion.div>
          )}
        </AnimatePresence>



        {/* Nearby Matches Drawer */}
        <AnimatePresence>
          {showMatchesDrawer && (
            <>
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                exit={{ opacity: 0 }}
                onClick={() => setShowMatchesDrawer(false)}
                className="fixed inset-0 z-40 backdrop-blur-sm"
                style={{
                  pointerEvents: 'auto',
                  backgroundColor: 'rgba(0, 0, 0, 0.5)',
                  zIndex: 40
                }}
              />
              <motion.div
                initial={{ y: "100%" }}
                animate={{ y: 0 }}
                exit={{ y: "100%" }}
                transition={{ type: "spring", damping: 25, stiffness: 300 }}
                onClick={(e) => e.stopPropagation()}
                className={`fixed bottom-0 left-0 right-0 z-50 bg-card rounded-t-3xl shadow-
elevation-4 p-6 ${
                  isActive ? 'pb-6' : 'pb-24'
                } border-t border-border`}
                style={{
                  pointerEvents: 'auto',
                  maxHeight: '85vh',
                  minHeight: '200px',
                  overflowY: 'auto'
                }}
              >
                {/* Header */}
                <div className="flex items-center justify-between mb-6">
                  <div className="flex items-center gap-3">
                    <h2 className="text-2xl font-bold text-foreground">Nearby Matches</h2>
                    {matches.length > 0 && (
                      <BadgeCounter count={matches.length} variant="default" size="sm" />
                    )}
                    {pokes.length > 0 && (
                      <span className="bg-purple-500 text-white text-xs font-bold rounded-full
px-2 py-0.5">
                        {pokes.length} poke{pokes.length > 1 ? 's' : ''}
                      </span>
```

```
                  )}
                </div>
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={() => setShowMatchesDrawer(false)}
                  className="rounded-full"
                >
                  <CloseIcon />
                </Button>
              </div>

              {/* Matches List */}
              {matchesLoading ? (
                <div className="flex items-center justify-center py-12">
                  <CircularProgress />
                </div>
              ) : matches.length > 0 ? (
                <div className="space-y-2">
                  {matches
                    .sort((a, b) => {
                      const aPoked = pokes.includes(a.user.uid);
                      const bPoked = pokes.includes(b.user.uid);
                      if (aPoked && !bPoked) return -1;
                      if (!aPoked && bPoked) return 1;
                      return 0;
                    })
                    .map((match, index) => {
                      const user = match.user;
                      const distanceKm = match.distance / 1000;
                      const isPoked = pokes.includes(user.uid);
                      const userData = nearbyUsers.find((u: any) => u.id === user.uid);
                      const fallbackUserData = userData || {
                        id: user.uid,
                        name: user.name || "User",
                        distance: formatDistance(distanceKm),
                        distanceValue: distanceKm,
                        activity: user.activity || "Running",
                        avatar: getProfilePictureUrl(user.photoURL, user.avatar, user.name),
                        photos: user.photos || [],
                        bio: user.bio || "",
                        lat: (user as any)?.location?.lat ?? null,
                        lng: (user as any)?.location?.lng ?? null,
                      };

                      return (
                        <motion.div
                          key={user.uid}
                          data-poked={isPoked}
                          initial={{ opacity: 0, y: 10 }}
                          animate={{ opacity: 1, y: 0 }}
                          transition={{ delay: index * 0.05 }}
                          className={`p-4 rounded-lg border border-border/50 hover:bg-muted/50
transition-colors relativ...      isPoked ? 'bg-purple-50/50 dark:bg-purple-950/20 border-
purple-300/30' : ''               }`}
                        >
                          <div className="flex items-center gap-3">
                            {/* Avatar */}
                            <button
                              onClick={() => {
                                setSelectedUser(fallbackUserData);
                                setShowProfileView(true);
                                setShowMatchesDrawer(false);
                              }}
                              className="cursor-pointer hover:opacity-80 transition-opacity"
                            >
                              <FitnessLevelAvatar
                                photoURL={user.photoURL}
                                name={user.name || "User"}
                                fitnessLevel={user.fitnessLevel}
                                size="md"
```

```
                                    showGlow={true}
                                  />
                                </button>

                                {/* User Info */}
                                <div className="flex-1 min-w-0">
                                  <div className="flex items-center gap-2 mb-1">
                                    <h4 className="font-semibold text-sm truncate">
                                      {user.name || "Unknown User"}
                                    </h4>
                                    {isPoked && (
                                      <Badge className="bg-purple-500 text-white text-[10px] font-
bold px-1.5 py-0.5 rounded...
                                        POKED YOU!
                                      </Badge>
                                    )}
                                    <Badge className="bg-primary text-primary-foreground text-xs
px-1.5 py-0">
                                      {formatDistance(distanceKm)}
                                    </Badge>
                                  </div>

                                  {/* Activity & Fitness Level */}
                                  <div className="flex items-center gap-2 text-xs mb-2">
                                    <div className="flex items-center gap-1">
                                      {user.activity === "running" ? (
                                        <DirectionsRunIcon className="text-success"
style={{ fontSize: 14 }} />
                                      ) : user.activity === "cycling" ? (
                                        <DirectionsBikeIcon className="text-primary"
style={{ fontSize: 14 }} />
                                      ) : (
                                        <DirectionsWalkIcon className="text-warning"
style={{ fontSize: 14 }} />
                                      )}
                                      <span className="capitalize">{user.activity}</span>
                                    </div>
                                    <span>•</span>
                                    <Badge
                                      variant="outline"
                                      className={`${
                                        user.fitnessLevel === "beginner" ? "bg-blue-500" :
                                        user.fitnessLevel === "intermediate" ? "bg-yellow-500" :
                                        "bg-red-500"
                                      } text-white border-0 text-xs px-1.5 py-0`}
                                    >
                                      {user.fitnessLevel}
                                    </Badge>
                                  </div>

                                  {/* Actions */}
                                  <div className="flex items-center gap-2 mt-2">
                                    {isActive && (
                                      <Button
                                        size="sm"
                                        className="flex-1 h-8 text-xs justify-center bg-purple-500
hover:bg-purple-600 text-...
                                        onClick={() => handlePoke(user.uid)}
                                      >
                                        <TouchAppIcon style={{ fontSize: 14 }} className="mr-1" />
                                        Poke
                                      </Button>
                                    )}
                                    <Button
                                      size="sm"
                                      variant="outline"
                                      className="flex-1 h-8 text-xs justify-center"
                                      onClick={() => handleAddFriend(user.uid, user.name)}
                                    >
                                      <PersonAddIcon style={{ fontSize: 14 }} className="mr-1" />
                                      Add
                                    </Button>
                                    <Button
                                      size="sm"
                                      variant="outline"
                                      className="flex-1 h-8 text-xs justify-center"
                                      onClick={() => {
```

```
                                    setSelectedUser(fallbackUserData);
                                    handleSendMessage();
                                    setShowMatchesDrawer(false);
                                  }}
                                >
                                  <SendIcon style={{ fontSize: 14 }} className="mr-1" />
                                  Message
                                </Button>
                              </div>
                            </div>
                          </div>
                        </motion.div>
                      );
                    })}
                  </div>
                ) : (
                  <div className="py-12 text-center">
                    <p className="text-sm text-muted-foreground">No nearby matches found</p>
                  </div>
                )}
              </motion.div>
            </>
          )}
        </AnimatePresence>

        {/* Filter Modal */}
        <AnimatePresence>
          {showFilterModal && (
            <>
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                exit={{ opacity: 0 }}
                onClick={() => setShowFilterModal(false)}
                className="fixed inset-0 bg-black/50 z-40"
              />
              <motion.div
                initial={{ y: "100%" }}
                animate={{ y: 0 }}
                exit={{ y: "100%" }}
                transition={{ type: "spring", damping: 25, stiffness: 300 }}
                className="fixed bottom-20 left-0 right-0 z-50 bg-background rounded-t-3xl shadow-elevation-4 p-6 max-h-[7...
                {/* Header */}
                <div className="flex items-center justify-between mb-6">
                  <div>
                    <h2 className="text-2xl font-bold text-foreground">Who do you want to find?</h2>
                    <p className="text-sm text-muted-foreground mt-1">
                      Filter nearby users by fitness level
                    </p>
                  </div>
                  <Button
                    variant="ghost"
                    size="icon"
                    onClick={() => setShowFilterModal(false)}
                    className="rounded-full"
                  >
                    <CloseIcon />
                  </Button>
                </div>

                {/* Filter Options */}
                <div className="space-y-3">
                  {[
                    {
                      value: "all" as SearchFilter,
                      label: "All Levels",
                      description: "Show everyone nearby",
                      color: "bg-muted text-muted-foreground",
                      selectedColor: "bg-primary text-primary-foreground"
                    },
```

```
                  {
                    value: "beginner" as SearchFilter,
                    label: "Beginner",
                    description: "Just starting out",
                    color: "bg-blue-100 text-blue-800 border-blue-300",
                    selectedColor: "bg-blue-600 text-white border-blue-600"
                  },
                  {
                    value: "intermediate" as SearchFilter,
                    label: "Intermediate",
                    description: "Regular fitness routine",
                    color: "bg-green-100 text-green-800 border-green-300",
                    selectedColor: "bg-green-600 text-white border-green-600"
                  },
                  {
                    value: "pro" as SearchFilter,
                    label: "Pro",
                    description: "Advanced athletes",
                    color: "bg-purple-100 text-purple-800 border-purple-300",
                    selectedColor: "bg-purple-600 text-white border-purple-600"
                  }
                ].map((option) => {
                  const isSelected = searchFilter === option.value;
                  return (
                    <motion.button
                      key={option.value}
                      whileTap={{ scale: 0.98 }}
                      onClick={() => handleUpdateSearchFilter(option.value)}
                      className={`
                        w-full p-4 rounded-xl border-2 transition-all duration-200
                        ${isSelected
                          ? option.selectedColor + " shadow-elevation-2"
                          : option.color + " hover:shadow-md"
                        }
                        flex items-center justify-between
                      `}
                    >
                      <div className="flex items-center gap-3">
                        {/* Visual Indicator */}
                        <div className={`
                          w-3 h-3 rounded-full
                          ${isSelected
                            ? "bg-white/30 ring-2 ring-white/50"
                            : option.value === "all"
                            ? "bg-muted-foreground/30"
                            : option.value === "beginner"
                            ? "bg-blue-600"
                            : option.value === "intermediate"
                            ? "bg-green-600"
                            : "bg-purple-600"
                          }
                        `} />
                        <div className="text-left">
                          <div className={`font-semibold text-base ${isSelected ? "text-white" : "text-
                            {option.label}
                          </div>
                          <div className={`text-xs mt-0.5 ${isSelected ? "text-white/80" : "text-muted-foreground"}`}>
                            {option.description}
                          </div>
                        </div>
                      </div>
                      {isSelected && (
                        <CheckCircleIcon className="text-white" style={{ fontSize: 20 }} />
                      )}
                    </motion.button>
                  );
                })}
              </div>
            </motion.div>
          </>
        )}
```

```
      </AnimatePresence>

      {/* Notification Drawer */}
      <AnimatePresence>
        {showNotificationDrawer && (
          <>
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              exit={{ opacity: 0 }}
              onClick={() => setShowNotificationDrawer(false)}
              className="fixed inset-0 bg-black/50 z-40"
            />
            <motion.div
              initial={{ y: "100%" }}
              animate={{ y: 0 }}
              exit={{ y: "100%" }}
              transition={{ type: "spring", damping: 25, stiffness: 300 }}
              onClick={(e) => e.stopPropagation()}
              className={`fixed bottom-0 left-0 right-0 z-50 bg-card rounded-t-3xl shadow-elevation-4 p-6 ${
                isActive ? 'pb-6' : 'pb-24'
              } border-t border-border`}
              style={{
                maxHeight: '85vh',
                minHeight: '200px',
                overflowY: 'auto'
              }}
            >
              {/* Header */}
              <div className="flex items-center justify-between mb-6">
                <div>
                  <h2 className="text-2xl font-bold text-foreground">Notifications</h2>
                  <p className="text-sm text-muted-foreground mt-1">
                    {unreadCount > 0 ? `${unreadCount} unread notification${unreadCount > 1 ? 's' : ''}` : 'All caught up'}
                  </p>
                </div>
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={() => setShowNotificationDrawer(false)}
                  className="rounded-full"
                >
                  <CloseIcon />
                </Button>
              </div>

              {/* Notifications List - Sorted by newest first */}
              {notifications.length > 0 ? (
                <div className="space-y-2">
                  {[...notifications]
                    .sort((a, b) => b.timestamp - a.timestamp) // Newest first
                    .map((notification) => {
                      const getNotificationIcon = () => {
                        switch (notification.type) {
                          case "message":
                            return <MailIcon style={{ fontSize: 20 }} className="text-primary" />;
                          case "message_request":
                            return <ChatBubbleIcon style={{ fontSize: 20 }} className="text-blue-500" />;
                          case "friend_request":
                            return <PersonAddIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "poke":
                            return <TouchAppIcon style={{ fontSize: 20 }} className="text-purple-500" />;
                          case "friend_accepted":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "workout_complete":
                            return <CheckCircleIcon style={{ fontSize: 20 }} className="text-success" />;
                          case "achievement":
                            return <EmojiEventsIcon style={{ fontSize: 20 }} className="text-warning" />;
                          case "nearby_active_user":
                            return <PeopleIcon style={{ fontSize: 20 }} className="text-warning" />;
                          default:
```

```
4043 |                           return <NotificationsIcon style={{ fontSize: 20 }} />;
4044 |                         }
4045 |                       };
4046 |
4047 |                       const getNotificationTitle = () => {
4048 |                         switch (notification.type) {
4049 |                           case "message":
4050 |                             return notification.userName;
4051 |                           case "message_request":
4052 |                             return notification.userName;
4053 |                           case "friend_request":
4054 |                             return notification.userName;
4055 |                           case "poke":
4056 |                             return notification.userName;
4057 |                           case "friend_accepted":
4058 |                             return notification.userName;
4059 |                           case "workout_complete":
4060 |                             return "Workout Completed";
4061 |                           case "achievement":
4062 |                             return notification.message || "Congrats for a new achievement!";
4063 |                           case "nearby_active_user":
4064 |                             return "Active Users Nearby";
4065 |                           default:
4066 |                             return notification.userName;
4067 |                         }
4068 |                       };
4069 |
4070 |                       const getNotificationMessage = () => {
4071 |                         switch (notification.type) {
4072 |                           case "message":
4073 |                             return notification.message || "Sent you a message";
4074 |                           case "message_request":
4075 |                             return "wants to start a conversation with you";
4076 |                           case "friend_request":
4077 |                             return "wants to add you as a friend";
4078 |                           case "poke":
4079 |                             return "poked you! They're interested in matching";
4080 |                           case "friend_accepted":
4081 |                             return "accepted your friend request";
4082 |                           case "workout_complete":
4083 |                             return notification.message || "Workout completed successfully!";
4084 |                           case "achievement":
4085 |                             return notification.message || "Congrats for a new achievement!";
4086 |                           case "nearby_active_user":
4087 |                             return notification.message || "Active users nearby! Start your
workout to see them.";
4088 |                           default:
4089 |                             return "";
4090 |                         }
4091 |                       };
4092 |
4093 |                       const formatTimestamp = (timestamp: number) => {
4094 |                         const now = Date.now();
4095 |                         const diff = now - timestamp;
4096 |                         const minutes = Math.floor(diff / 60000);
4097 |                         const hours = Math.floor(diff / 3600000);
4098 |                         const days = Math.floor(diff / 86400000);
4099 |
4100 |                         if (minutes < 1) return "Just now";
4101 |                         if (minutes < 60) return `${minutes}m ago`;
4102 |                         if (hours < 24) return `${hours}h ago`;
4103 |                         if (days < 7) return `${days}d ago`;
4104 |                         return new Date(timestamp).toLocaleDateString();
4105 |                       };
4106 |
4107 |                       return (
4108 |                         <motion.div
4109 |                           key={notification.id}
4110 |                           initial={{ opacity: 0, y: 10 }}
4111 |                           animate={{ opacity: 1, y: 0 }}
4112 |                           className={`p-4 rounded-lg border border-border/50 hover:bg-muted/50
transition-colors cursor-...
4113 |                             !notification.read ? 'bg-primary/5 border-primary/30' : ''
```

```
                          }`}
                          onClick={() => {
                            // Special handling for nearby_active_user: start workout
                            if (notification.type === "nearby_active_user") {
                              handleNotificationBannerTap();
                              dismissNotification(notification.id);
                            } else {
                              handleNotificationTap(notification);
                            }
                            setShowNotificationDrawer(false);
                          }}
                        >
                          <div className="flex items-start gap-3">
                            {/* Icon */}
                            <div className={`flex-shrink-0 p-2 rounded-full ${
                              notification.type === "message"
                                ? "bg-primary/15"
                                : notification.type === "message_request"
                                ? "bg-blue-500/15"
                                : notification.type === "friend_request"
                                ? "bg-warning/15"
                                : notification.type === "poke"
                                ? "bg-purple-500/15"
                                : notification.type === "workout_complete"
                                ? "bg-success/15"
                                : notification.type === "achievement"
                                ? "bg-warning/15"
                                : notification.type === "friend_accepted"
                                ? "bg-success/15"
                                : notification.type === "nearby_active_user"
                                ? "bg-warning/15"
                                : "bg-gray-500/15"
                            }`}>
                              {getNotificationIcon()}
                            </div>

                            {/* Content */}
                            <div className="flex-1 min-w-0">
                              <div className="flex items-center justify-between mb-1">
                                <p className="font-semibold text-sm text-foreground">
                                  {getNotificationTitle()}
                                </p>
                                {!notification.read && (
                                  <span className="w-2 h-2 bg-primary rounded-full flex-
shrink-0"></span>
                                )}
                              </div>
                              <p className="text-sm text-muted-foreground mb-1">
                                {getNotificationMessage()}
                              </p>
                              <p className="text-xs text-muted-foreground">
                                {formatTimestamp(notification.timestamp)}
                              </p>
                            </div>

                            {/* Dismiss button */}
                            <button
                              onClick={(e) => {
                                e.stopPropagation();
                                dismissNotification(notification.id);
                              }}
                              className="flex-shrink-0 p-1 hover:bg-accent rounded-full
transition-colors"
                            >
                              <CloseIcon style={{ fontSize: 16 }} className="text-muted-
foreground" />
                            </button>
                          </div>
                        </motion.div>
                      );
                    })}
                  </div>
                ) : (
                  <div className="py-12 text-center">
```

```
4185 |                   <NotificationsIcon className="text-muted-foreground mx-auto mb-2"
4186 | style={{ fontSize: 48 }} />
4187 |                   <p className="text-sm text-muted-foreground">No notifications yet</p>
4188 |                 </div>
4189 |               )}
4190 |             </motion.div>
4191 |           </>
4192 |         )}
4193 |       </AnimatePresence>
4194 |
4195 |       {!isActive && <BottomNavigation />}
4196 |     </div>
4197 |   );
4198 | };
4199 | export default MapScreen;
```

File: src/pages/Messages.tsx

Lines: 480

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { useNavigate } from "react-router-dom";
import { useUser } from "@/contexts/UserContext";
import { useAuth } from "@/hooks/useAuth";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import SearchIcon from "@mui/icons-material/Search";
import CheckIcon from "@mui/icons-material/Check";
import CloseIcon from "@mui/icons-material/Close";
import Avatar from "@mui/material/Avatar";
import { Badge } from "@/components/ui/badge";
import { Tabs, TabsList, TabsTrigger, TabsContent } from "@/components/ui/tabs";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import MailOutlineIcon from "@mui/icons-material/MailOutline";
import BottomNavigation from "@/components/BottomNavigation";
import { getUserConversations } from "@/services/messageService";
import { get, ref } from "firebase/database";
import { database } from "@/services/firebase";
import {
  isUserBlocked,
} from "@/lib/messageStorage";
import { markMessagesAsRead, deleteConversation } from "@/services/messageService";
import { toast } from "sonner";
import { useNotificationContext } from "@/contexts/NotificationContext";
import { getDisplayName } from "@/utils/anonymousName";
import { getProfilePictureUrl } from "@/utils/profilePicture";

interface Conversation {
  conversationId: string;
  otherUserId: string;
  lastMessage: string;
  lastMessageTime: number;
  unreadCount: number;
  userName?: string;
  avatar?: string;
  activity?: string | null;
}

const Messages = () => {
  const navigate = useNavigate();
  const { unreadMessageCount } = useNotificationContext();
  const { userProfile } = useUser();
  const { user: currentUser } = useAuth();
  const [activeTab, setActiveTab] = useState<"chats" | "requests">("chats");
  const [conversations, setConversations] = useState<Conversation[]>([]);
  const [loading, setLoading] = useState(true);
  const [searchQuery, setSearchQuery] = useState("");
  const [showSearch, setShowSearch] = useState(false);

  // Fetch conversations from Firebase
  useEffect(() => {
    if (!currentUser?.uid) {
      setLoading(false);
      return;
    }

    const loadConversations = async () => {
      try {
        const firebaseConversations = await getUserConversations(currentUser.uid);

        // Fetch user data for each conversation
        let conversationsWithUserData = await Promise.all(
          firebaseConversations.map(async (conv) => {
            try {
              const userRef = ref(database, `users/${conv.otherUserId}`);
```

```
 67 |               const userSnapshot = await get(userRef);
 68 |               const userData = userSnapshot.exists() ? userSnapshot.val() : null;
 69 |
 70 |               const username = userData?.name || null;
 71 |               const activity = userData?.activity || null;
 72 |               const displayName = getDisplayName(username, conv.otherUserId, activity);
 73 |               return {
 74 |                 ...conv,
 75 |                 userName: displayName,
 76 |                 avatar: getProfilePictureUrl(userData?.photoURL, userData?.avatar, displayName),
 77 |                 activity: activity,
 78 |               };
 79 |             } catch (error) {
 80 |               console.error(`Error fetching user ${conv.otherUserId}:`, error);
 81 |               const fallbackName = getDisplayName(null, conv.otherUserId, null);
 82 |               return {
 83 |                 ...conv,
 84 |                 userName: fallbackName,
 85 |                 avatar: getProfilePictureUrl(null, null, fallbackName),
 86 |                 activity: null,
 87 |               };
 88 |             }
 89 |           })
 90 |         );
 91 |
 92 |         setConversations(conversationsWithUserData);
 93 |       } catch (error) {
 94 |         console.error("Error loading conversations:", error);
 95 |         toast.error("Failed to load conversations");
 96 |       } finally {
 97 |         setLoading(false);
 98 |       }
 99 |     };
100 |
101 |     loadConversations();
102 |   }, [currentUser?.uid]);
103 |
104 |   // Filter conversations based on friend status and blocked users
105 |   const friends = userProfile?.friends || [];
106 |   const blockedUsers = conversations
107 |     .filter(conv => isUserBlocked(parseInt(conv.otherUserId)))
108 |     .map(conv => conv.otherUserId);
109 |
110 |   // Split conversations into chats and requests
111 |   // Message requests are conversations with non-friends that have messages
112 |   const allConversations = conversations
113 |     .filter(conv => !blockedUsers.includes(conv.otherUserId)); // Hide blocked users
114 |
115 |   // Separate into chats (with friends) and requests (from non-friends)
116 |   // Message requests are ALL conversations from people who aren't your friends
117 |   const chatConversationsFiltered = allConversations.filter(conv =>
118 |     friends.includes(conv.otherUserId)
119 |   );
120 |
121 |   const requestConversationsFiltered = allConversations.filter(conv =>
122 |     !friends.includes(conv.otherUserId)
123 |   );
124 |
125 |   // Filter conversations by search query
126 |   const filterConversations = (convs: Conversation[]) => {
127 |     if (!searchQuery.trim()) return convs;
128 |
129 |     const query = searchQuery.toLowerCase().trim();
130 |     return convs.filter(conv => {
131 |       const userName = (conv.userName || "").toLowerCase();
132 |       const lastMessage = (conv.lastMessage || "").toLowerCase();
133 |       return userName.includes(query) || lastMessage.includes(query);
134 |     });
135 |   };
136 |
137 |   const chatConversations = filterConversations(chatConversationsFiltered);
```

```
138 |   const requestConversations = filterConversations(requestConversationsFiltered);
139 |
140 |   const handleAcceptRequest = async (conversation: Conversation, e: React.MouseEvent) => {
141 |     e.stopPropagation();
142 |     if (!currentUser?.uid) return;
143 |
144 |     try {
145 |       // Mark messages as read to "accept" the request
146 |       await markMessagesAsRead(currentUser.uid, conversation.otherUserId);
147 |
148 |       // Refresh conversations to remove the accepted request from the list
149 |       const firebaseConversations = await getUserConversations(currentUser.uid);
150 |       const conversationsWithUserData = await Promise.all(
151 |         firebaseConversations.map(async (conv) => {
152 |           try {
153 |             const userRef = ref(database, `users/${conv.otherUserId}`);
154 |             const userSnapshot = await get(userRef);
155 |             const userData = userSnapshot.exists() ? userSnapshot.val() : null;
156 |
157 |             const username = userData?.name || null;
158 |             const activity = userData?.activity || null;
159 |             return {
160 |               ...conv,
161 |               userName: getDisplayName(username, conv.otherUserId, activity),
162 |               avatar: getProfilePictureUrl(userData?.photoURL, userData?.avatar,
getDisplayName(username, conv.otherUser...
163 |               activity: activity,
164 |             };
165 |           } catch (error) {
166 |             const fallbackName = getDisplayName(null, conv.otherUserId, null);
167 |             return {
168 |               ...conv,
169 |               userName: fallbackName,
170 |               avatar: getProfilePictureUrl(null, null, fallbackName),
171 |               activity: null,
172 |             };
173 |           }
174 |         })
175 |       );
176 |       setConversations(conversationsWithUserData);
177 |
178 |       toast.success(`Accepted message from ${conversation.userName}`);
179 |
180 |       // Navigate to the chat screen after accepting
181 |       navigate("/chat", {
182 |         state: {
183 |           user: {
184 |             id: conversation.otherUserId,
185 |             name: conversation.userName || "Unknown User",
186 |             avatar: conversation.avatar || "",
187 |           },
188 |         },
189 |       });
190 |     } catch (error) {
191 |       console.error("Error accepting message request:", error);
192 |       toast.error("Failed to accept message request");
193 |     }
194 |   };
195 |
196 |   const handleDeclineRequest = async (conversation: Conversation, e: React.MouseEvent) => {
197 |     e.stopPropagation();
198 |     if (!currentUser?.uid) return;
199 |
200 |     try {
201 |       // Delete the conversation
202 |       await deleteConversation(currentUser.uid, conversation.otherUserId);
203 |       toast.success(`Declined message from ${conversation.userName}`);
204 |       // Refresh conversations to update the list
205 |       const firebaseConversations = await getUserConversations(currentUser.uid);
206 |       const conversationsWithUserData = await Promise.all(
207 |         firebaseConversations.map(async (conv) => {
208 |           try {
```

```
209 |             const userRef = ref(database, `users/${conv.otherUserId}`);
210 |             const userSnapshot = await get(userRef);
211 |             const userData = userSnapshot.exists() ? userSnapshot.val() : null;
212 |
213 |             const username = userData?.name || null;
214 |             const activity = userData?.activity || null;
215 |             return {
216 |               ...conv,
217 |               userName: getDisplayName(username, conv.otherUserId, activity),
218 |               avatar: userData?.photoURL || "",
219 |               activity: activity,
220 |             };
221 |           } catch (error) {
222 |             return {
223 |               ...conv,
224 |               userName: getDisplayName(null, conv.otherUserId, null),
225 |               avatar: "",
226 |               activity: null,
227 |             };
228 |           }
229 |         })
230 |       );
231 |       setConversations(conversationsWithUserData);
232 |     } catch (error) {
233 |       console.error("Error declining message request:", error);
234 |       toast.error("Failed to decline message request");
235 |     }
236 |   };
237 |
238 |   const getRelativeTime = (timestamp: number): string => {
239 |     const now = Date.now();
240 |     const diff = now - timestamp;
241 |     const minutes = Math.floor(diff / (1000 * 60));
242 |     const hours = Math.floor(diff / (1000 * 60 * 60));
243 |     const days = Math.floor(diff / (1000 * 60 * 60 * 24));
244 |
245 |     if (minutes < 1) return "Just now";
246 |     if (minutes < 60) return `${minutes}m ago`;
247 |     if (hours < 24) return `${hours}h ago`;
248 |     if (days < 7) return `${days}d ago`;
249 |
250 |     // Format as date if > 7 days
251 |     const date = new Date(timestamp);
252 |     return date.toLocaleDateString('en-US', { month: 'short', day: 'numeric' });
253 |   };
254 |
255 |   const truncateMessage = (message: string, maxLength: number = 50): string => {
256 |     if (message.length <= maxLength) return message;
257 |     return message.substring(0, maxLength) + "...";
258 |   };
259 |
260 |   const handleConversationClick = (conversation: Conversation) => {
261 |     navigate("/chat", {
262 |       state: {
263 |         user: {
264 |           id: conversation.otherUserId,
265 |           name: conversation.userName || "Unknown User",
266 |           avatar: conversation.avatar || "",
267 |         },
268 |       },
269 |     });
270 |   };
271 |
272 |   // Calculate unread counts from filtered conversations
273 |   const totalUnread = chatConversations.reduce((sum, conv) => sum + conv.unreadCount, 0);
274 |   const requestsCount = requestConversations.length;
275 |
276 |   const renderConversationList = (conversations: Conversation[], showActions = false) => {
277 |     if (conversations.length === 0) {
278 |       return (
279 |         <div className="flex flex-col items-center justify-center h-full px-6 py-12 text-center">
```

```
          <motion.div
            initial={{ opacity: 0, scale: 0.8 }}
            animate={{ opacity: 1, scale: 1 }}
            transition={{ duration: 0.3 }}
          >
            <MailOutlineIcon
              style={{ fontSize: 80 }}
              className="text-muted-foreground/50 mb-4"
            />
            <h2 className="text-xl font-bold text-foreground mb-2">
              {showActions ? "No message requests" : "No messages yet"}
            </h2>
            <p className="text-muted-foreground max-w-xs">
              {showActions
                ? "When someone who isn't your friend messages you, you'll see it here."
                : "Start a conversation by sending a message to someone nearby!"}
            </p>
          </motion.div>
        </div>
      );
    }

    return (
      <div className="divide-y divide-border">
        {conversations.map((conversation, index) => (
          <motion.div
            key={conversation.conversationId || conversation.otherUserId}
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: index * 0.05 }}
            className="relative"
          >
            <button
              onClick={() => handleConversationClick(conversation)}
              className="w-full text-left hover:bg-accent transition-colors"
            >
              <div className="flex items-center gap-3 p-4">
                {/* Avatar */}
                <div className="relative flex-shrink-0">
                  <Avatar
                    src={conversation.avatar}
                    alt={conversation.userName || "User"}
                    sx={{ width: 56, height: 56 }}
                  />
                  {conversation.unreadCount > 0 && (
                    <div className="absolute -top-1 -right-1 w-5 h-5 bg-destructive text-
destructive-foreground text-xs rounded-
                      {conversation.unreadCount}
                    </div>
                  )}
                </div>

                {/* Content */}
                <div className="flex-1 min-w-0">
                  <div className="flex items-baseline justify-between gap-2 mb-1">
                    <h3 className={`font-semibold text-base truncate ${
                      conversation.unreadCount > 0 ? "text-foreground" : "text-foreground"
                    }`}>
                      {conversation.userName || "Unknown User"}
                    </h3>
                    <span className="text-xs text-muted-foreground flex-shrink-0">
                      {getRelativeTime(conversation.lastMessageTime)}
                    </span>
                  </div>
                  <p className={`text-sm truncate ${
                    conversation.unreadCount > 0
                      ? "text-foreground font-medium"
                      : "text-muted-foreground"
                  }`}>
                    {truncateMessage(conversation.lastMessage)}
                  </p>
                </div>
```

```
351 |               </div>
352 |             </button>
353 |
354 |             {/* Action buttons for requests */}
355 |             {showActions && (
356 |               <div className="flex gap-2 px-4 pb-4">
357 |                 <Button
358 |                   onClick={(e) => handleAcceptRequest(conversation, e)}
359 |                   className="flex-1 h-9 bg-primary hover:bg-primary/90"
360 |                   size="sm"
361 |                 >
362 |                   <CheckIcon style={{ fontSize: 18 }} className="mr-1" />
363 |                   Accept
364 |                 </Button>
365 |                 <Button
366 |                   onClick={(e) => handleDeclineRequest(conversation, e)}
367 |                   variant="outline"
368 |                   className="flex-1 h-9"
369 |                   size="sm"
370 |                 >
371 |                   <CloseIcon style={{ fontSize: 18 }} className="mr-1" />
372 |                   Delete
373 |                 </Button>
374 |               </div>
375 |             )}
376 |           </motion.div>
377 |         ))}
378 |       </div>
379 |     );
380 |   };
381 |
382 |   return (
383 |     <div className="min-h-screen bg-background flex flex-col pb-20">
384 |       {/* Header */}
385 |       <motion.div
386 |         initial={{ opacity: 0, y: -20 }}
387 |         animate={{ opacity: 1, y: 0 }}
388 |         className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b
border-border/50"
389 |         style={{
390 |           paddingTop: 'env(safe-area-inset-top)',
391 |         }}
392 |       >
393 |         <div className="max-w-2xl mx-auto px-6 py-5">
394 |           <div className="flex items-center justify-between">
395 |             <h1 className="text-3xl font-bold">Messages</h1>
396 |             {!showSearch ? (
397 |               <motion.button
398 |                 whileTap={{ scale: 0.95 }}
399 |                 onClick={() => setShowSearch(true)}
400 |                 className="touch-target p-2 rounded-full hover:bg-accent transition-colors"
401 |               >
402 |                 <SearchIcon style={{ fontSize: 24 }} className="text-muted-foreground" />
403 |               </motion.button>
404 |             ) : (
405 |               <div className="flex items-center gap-2 flex-1 max-w-xs ml-4">
406 |                 <Input
407 |                   type="text"
408 |                   placeholder="Search conversations..."
409 |                   value={searchQuery}
410 |                   onChange={(e) => setSearchQuery(e.target.value)}
411 |                   className="flex-1"
412 |                   autoFocus
413 |                 />
414 |                 <motion.button
415 |                   whileTap={{ scale: 0.95 }}
416 |                   onClick={() => {
417 |                     setShowSearch(false);
418 |                     setSearchQuery("");
419 |                   }}
420 |                   className="touch-target p-2 rounded-full hover:bg-accent transition-colors"
421 |                 >
```

```
                    <CloseIcon style={{ fontSize: 24 }} className="text-muted-foreground" />
                  </motion.button>
                </div>
              )}
            </div>
          </div>
        </motion.div>

        {/* Tabs */}
        <div className="bg-card/80 backdrop-blur-md sticky top-[calc(env(safe-area-inset-top)+5rem)] z-10 border-b border-
        <div  max-w-2xl mx-auto px-6">
          <Tabs value={activeTab} onValueChange={(v) => setActiveTab(v as "chats" | "requests")} 
<TabsList className="w-full grid grid-cols-2 h-12 bg-muted/30">
            <TabsTrigger value="chats" className="relative data-[state=active]:bg-background">
              Chats
              {totalUnread > 0 && (
                <span className="ml-2 px-1.5 py-0.5 text-xs font-bold bg-destructive text-destructive-foreground round
                  {totalUnread}
                </span>
              )}
            </TabsTrigger>
            <TabsTrigger value="requests" className="relative data-[state=active]:bg-background">
              Requests
              {requestsCount > 0 && (
                <span className="ml-2 px-1.5 py-0.5 text-xs font-bold bg-primary text-primary-foreground rounded-full ...{requestsCount}
                </span>
              )}
            </TabsTrigger>
          </TabsList>
        </Tabs>
      </div>
    </div>

    {/* Content */}
    <div className="flex-1 overflow-y-auto">
      {loading ? (
        <div className="flex items-center justify-center h-full">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
        </div>
      ) : (
        <Tabs value={activeTab} className="w-full h-full">
          <TabsContent value="chats" className="mt-0 h-full">
            {renderConversationList(chatConversations, false)}
          </TabsContent>
          <TabsContent value="requests" className="mt-0 h-full">
            {renderConversationList(requestConversations, true)}
          </TabsContent>
        </Tabs>
      )}
    </div>

    <BottomNavigation />
  </div>
 );
};

export default Messages;
```

File: src/pages/MyEvents.tsx

Lines: 925

```
import { useState, useEffect } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { useNavigate } from "react-router-dom";
import { useAuth } from "@/hooks/useAuth";
import { getUserEvents, Event as FirebaseEvent, leaveEvent, updateEvent, deleteEvent } from "@/services/eventService";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { Card } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Calendar } from "@/components/ui/calendar";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
import EventIcon from "@mui/icons-material/Event";
import LocationOnIcon from "@mui/icons-material/LocationOn";
import PeopleIcon from "@mui/icons-material/People";
import StarIcon from "@mui/icons-material/Star";
import CheckCircleIcon from "@mui/icons-material/CheckCircle";
import EditIcon from "@mui/icons-material/Edit";
import UpcomingIcon from "@mui/icons-material/Upcoming";
import HistoryIcon from "@mui/icons-material/History";
import ExploreIcon from "@mui/icons-material/Explore";
import CalendarMonthIcon from "@mui/icons-material/CalendarMonth";
import { EventDetailModal } from "@/components/EventDetailModal";
import { CreateEventModal, type CreateEventFormData } from "@/components/CreateEventModal";
import { toast } from "sonner";
import { format, isSameDay, parseISO } from "date-fns";
import { generateDummyEvents, ENABLE_DUMMY_DATA } from "@/lib/dummyData";

type EventType = "running" | "cycling" | "walking" | "others";

interface Event {
  id: string;
  title: string;
  description: string;
  type: EventType;
  category: "user" | "sponsored";
  date: string;
  time: string;
  location: string;
  distance: string;
  distanceValue: number;
  participants: string[]; // Array of user IDs
  maxParticipants?: number;
  hostId?: string;
  hostName?: string;
  hostAvatar?: string;
  sponsorLogo?: string;
  lat: number;
  lng: number;
  isJoined?: boolean;
  isPast?: boolean;
  isGreyedOut?: boolean;
}

const MyEvents = () => {
  const navigate = useNavigate();
  const { user: currentUser } = useAuth();
  const [activeTab, setActiveTab] = useState("upcoming");
  const [selectedEvent, setSelectedEvent] = useState<Event | null>(null);
  const [showEventDetail, setShowEventDetail] = useState(false);
  const [showEditEvent, setShowEditEvent] = useState(false);
  const [eventBeingEdited, setEventBeingEdited] = useState<Event | null>(null);
  const [selectedDate, setSelectedDate] = useState<Date | undefined>(new Date());
```

```
  const [joinedEvents, setJoinedEvents] = useState<Event[]>([]);
  const [loading, setLoading] = useState(true);

  // Fetch user events from Firebase
  useEffect(() => {
    if (!currentUser?.uid) {
      setLoading(false);
      return;
    }

    const loadUserEvents = async () => {
      try {
        const firebaseEvents = await getUserEvents(currentUser.uid);

        // Transform and determine if events are past or greyed out (24 hours after event)
        let transformedEvents: Event[] = firebaseEvents
          .filter((event) => event && event.date) // Filter out invalid events
          .map((event) => {
            try {
              // Combine date and time to get full event datetime
              const eventDateTime = new Date(`${event.date}T${event.time || '00:00'}`);
              const now = new Date();

              // Check if date is valid
              if (isNaN(eventDateTime.getTime())) {
                console.warn("Invalid event date:", event.date, event);
                return null;
              }

              // Calculate 24 hours after event
              const eventEndTime = new Date(eventDateTime);
              eventEndTime.setHours(eventEndTime.getHours() + 24);

              // Event is past if it's been more than 24 hours since the event
              const isPast = now > eventEndTime;
              // Event is greyed out if it's been more than 24 hours since the event
              const isGreyedOut = now > eventEndTime;

              return {
                ...event,
                participants: Array.isArray(event.participants) ? event.participants : [],
                isJoined: true, // All events from getUserEvents are joined
                isPast,
                isGreyedOut,
              };
            } catch (error) {
              console.error("Error processing event:", error, event);
              return null;
            }
          })
          .filter((event): event is Event => event !== null); // Remove null entries

        // Add dummy events if enabled and no real events exist
        if (ENABLE_DUMMY_DATA && transformedEvents.length === 0) {
          const dummyEvents = generateDummyEvents();
          // Add current user to some events and filter to only include events where current user is a participant
          transformedEvents = dummyEvents
            .map((event, index) => {
              // Ensure participants is an array
              const participants = Array.isArray(event.participants) ? event.participants : [];
              // Add current user to first 3 events to ensure they show up
              if (index < 3 && !participants.includes(currentUser.uid)) {
                participants.push(currentUser.uid);
              }
              return {
                ...event,
                participants,
              };
            })
            .filter(event => {
              const participants = Array.isArray(event.participants) ? event.participants : [];
```

```
138 |               return participants.includes(currentUser.uid);
139 |             })
140 |             .map((event) => {
141 |               // Combine date and time to get full event datetime
142 |               const eventDateTime = new Date(`${event.date}T${event.time || '00:00'}`);
143 |               const now = new Date();
144 |
145 |               // Calculate 24 hours after event
146 |               const eventEndTime = new Date(eventDateTime);
147 |               eventEndTime.setHours(eventEndTime.getHours() + 24);
148 |
149 |               // Event is past if it's been more than 24 hours since the event
150 |               const isPast = now > eventEndTime;
151 |               const isGreyedOut = now > eventEndTime;
152 |
153 |               return {
154 |                 ...event,
155 |                 participants: Array.isArray(event.participants) ? event.participants : [],
156 |                 isJoined: true,
157 |                 isPast,
158 |                 isGreyedOut,
159 |               };
160 |             });
161 |         }
162 |
163 |         setJoinedEvents(transformedEvents);
164 |       } catch (error) {
165 |         console.error("Error loading user events:", error);
166 |       } finally {
167 |         setLoading(false);
168 |       }
169 |     };
170 |
171 |     loadUserEvents();
172 |   }, [currentUser?.uid]);
173 |
174 |   // Filter events by status with null safety
175 |   const upcomingEvents = (joinedEvents || []).filter((event) => event && !event.isPast);
176 |   const pastEvents = (joinedEvents || []).filter((event) => event && event.isPast);
177 |
178 |   // Helper function to safely parse event dates for calendar
179 |   const getValidEventDates = (events: Event[]) => {
180 |     return events
181 |       .filter((event) => event && event.date)
182 |       .map((event) => {
183 |         try {
184 |           const date = parseISO(event.date);
185 |           return isNaN(date.getTime()) ? null : date;
186 |         } catch {
187 |           return null;
188 |         }
189 |       })
190 |       .filter((date): date is Date => date !== null);
191 |   };
192 |
193 |   // Get valid dates for calendar modifiers
194 |   const upcomingEventsDates = getValidEventDates(upcomingEvents);
195 |   const pastEventsDates = getValidEventDates(pastEvents);
196 |   const allJoinedEventsDates = getValidEventDates(joinedEvents);
197 |
198 |   const handleEventClick = (event: Event) => {
199 |     setSelectedEvent(event);
200 |     setShowEventDetail(true);
201 |   };
202 |
203 |   const handleLeaveEvent = async (eventId: string) => {
204 |     if (!currentUser?.uid) {
205 |       toast.error("Please log in to leave events");
206 |       return;
207 |     }
208 |
```

```
209 |     try {
210 |       await leaveEvent(eventId, currentUser.uid);
211 |       toast.success("You've left the event");
212 |       // Remove from local state
213 |       setJoinedEvents(prev => prev.filter(e => e.id !== eventId));
214 |       // Close modal if open
215 |       if (selectedEvent?.id === eventId) {
216 |         setShowEventDetail(false);
217 |         setSelectedEvent(null);
218 |       }
219 |     } catch (error: any) {
220 |       console.error("Error leaving event:", error);
221 |       toast.error(error.message || "Failed to leave event");
222 |     }
223 |   };
224 |
225 |   const handleDeleteEvent = async (eventId: string) => {
226 |     if (!currentUser?.uid) {
227 |       toast.error("Please log in to delete events");
228 |       return;
229 |     }
230 |
231 |     try {
232 |       await deleteEvent(eventId, currentUser.uid);
233 |       toast.success("Event deleted successfully");
234 |       // Remove from local state
235 |       setJoinedEvents(prev => prev.filter(e => e.id !== eventId));
236 |       // Close modal
237 |       setShowEventDetail(false);
238 |       setSelectedEvent(null);
239 |     } catch (error: any) {
240 |       console.error("Error deleting event:", error);
241 |       toast.error(error.message || "Failed to delete event");
242 |     }
243 |   };
244 |
245 |   const handleEditEvent = (eventId: string) => {
246 |     if (!currentUser?.uid) {
247 |       toast.error("Please log in to edit events");
248 |       return;
249 |     }
250 |
251 |     const targetEvent = joinedEvents.find((event) => event.id === eventId);
252 |     if (!targetEvent) {
253 |       toast.error("Event not found");
254 |       return;
255 |     }
256 |
257 |     if (targetEvent.hostId !== currentUser.uid) {
258 |       toast.error("Only the event creator can edit this event");
259 |       return;
260 |     }
261 |
262 |     setEventBeingEdited(targetEvent);
263 |     setShowEditEvent(true);
264 |   };
265 |
266 |   const handleCloseEditModal = () => {
267 |     setShowEditEvent(false);
268 |     setEventBeingEdited(null);
269 |   };
270 |
271 |   const handleUpdateEvent = async (eventId: string, updatedData: CreateEventFormData) => {
272 |     if (!currentUser?.uid) {
273 |       toast.error("Please log in to edit events");
274 |       return;
275 |     }
276 |
277 |     const existingEvent = joinedEvents.find((event) => event.id === eventId);
278 |
279 |     try {
```

```
280 |       await updateEvent(eventId, currentUser.uid, {
281 |         title: updatedData.title,
282 |         description: updatedData.description,
283 |         type: updatedData.activityType,
284 |         date: updatedData.date,
285 |         time: updatedData.time,
286 |         location: updatedData.location,
287 |         lat: updatedData.lat ?? existingEvent?.lat,
288 |         lng: updatedData.lng ?? existingEvent?.lng,
289 |         maxParticipants: updatedData.maxParticipants,
290 |       });
291 |
292 |       toast.success("Event updated successfully");
293 |
294 |       setJoinedEvents((prev) =>
295 |         prev.map((event) =>
296 |           event.id === eventId
297 |             ? {
298 |                 ...event,
299 |                 title: updatedData.title,
300 |                 description: updatedData.description,
301 |                 type: updatedData.activityType as EventType,
302 |                 date: updatedData.date,
303 |                 time: updatedData.time,
304 |                 location: updatedData.location,
305 |                 lat: updatedData.lat ?? event.lat,
306 |                 lng: updatedData.lng ?? event.lng,
307 |                 maxParticipants: updatedData.maxParticipants,
308 |               }
309 |             : event
310 |         )
311 |       );
312 |
313 |       setSelectedEvent((prev) =>
314 |         prev && prev.id === eventId
315 |           ? {
316 |               ...prev,
317 |               title: updatedData.title,
318 |               description: updatedData.description,
319 |               type: updatedData.activityType as EventType,
320 |               date: updatedData.date,
321 |               time: updatedData.time,
322 |               location: updatedData.location,
323 |               lat: updatedData.lat ?? prev.lat,
324 |               lng: updatedData.lng ?? prev.lng,
325 |               maxParticipants: updatedData.maxParticipants,
326 |             }
327 |           : prev
328 |       );
329 |
330 |       handleCloseEditModal();
331 |     } catch (error: any) {
332 |       console.error("Error updating event:", error);
333 |       toast.error(error.message || "Failed to update event");
334 |     }
335 |   };
336 |
337 |   const getActivityIcon = (type: EventType | string) => {
338 |     switch (type) {
339 |       case "running":
340 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 20 }} />;
341 |       case "cycling":
342 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 20 }} />;
343 |       case "walking":
344 |         return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 20 }} />;
345 |       default:
346 |         // Default to running icon if type is unknown
347 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 20 }} />;
348 |     }
349 |   };
350 |
```

```
351 |   const getActivityColor = (type: EventType | string) => {
352 |     switch (type) {
353 |       case "running":
354 |         return "bg-success";
355 |       case "cycling":
356 |         return "bg-primary";
357 |       case "walking":
358 |         return "bg-warning";
359 |       default:
360 |         return "bg-success";
361 |     }
362 |   };
363 |
364 |   const getEventsForDate = (date: Date) => {
365 |     if (!date || !(date instanceof Date) || isNaN(date.getTime())) {
366 |       return [];
367 |     }
368 |     if (!joinedEvents || joinedEvents.length === 0) {
369 |       return [];
370 |     }
371 |     return joinedEvents.filter((event) => {
372 |       if (!event || !event.date) return false;
373 |       try {
374 |         const eventDate = parseISO(event.date);
375 |         if (isNaN(eventDate.getTime())) return false;
376 |         return isSameDay(eventDate, date);
377 |       } catch (error) {
378 |         console.error("Error parsing event date:", error, event);
379 |         return false;
380 |       }
381 |     });
382 |   };
383 |
384 |   const eventsForSelectedDate = selectedDate ? getEventsForDate(selectedDate) : [];
385 |
386 |   return (
387 |     <div className="min-h-screen bg-gradient-to-br from-primary/10 via-background to-success/10">
388 |       {/* Header */}
389 |       <motion.div
390 |         initial={{ opacity: 0, y: -20 }}
391 |         animate={{ opacity: 1, y: 0 }}
392 |         className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-20 border-b
border-border/50"
393 |         >
394 |         <div className="max-w-7xl mx-auto px-4 sm:px-6 py-4">
395 |           <div className="flex items-center gap-3">
396 |             <motion.button
397 |               whileTap={{ scale: 0.95 }}
398 |               onClick={() => navigate("/events")}
399 |               className="touch-target p-2 hover:bg-secondary rounded-xl transition-all
duration-200"
400 |             >
401 |               <ArrowBackIcon style={{ fontSize: 28 }} />
402 |             </motion.button>
403 |             <div className="flex-1">
404 |               <h1 className="text-2xl sm:text-3xl font-bold">My Events</h1>
405 |               <p className="text-sm text-muted-foreground">
406 |                 {joinedEvents.length} event{joinedEvents.length !== 1 ? "s" : ""} joined
407 |               </p>
408 |             </div>
409 |             <Button
410 |               variant="outline"
411 |               onClick={() => navigate("/events")}
412 |               className="hidden sm:flex h-10"
413 |             >
414 |               <ExploreIcon className="mr-2" style={{ fontSize: 20 }} />
415 |               Explore Events
416 |             </Button>
417 |           </div>
418 |         </div>
419 |       </motion.div>
420 |
421 |       {/* Content */}
```

```
422 |       <div className="max-w-7xl mx-auto px-4 sm:px-6 py-6">
423 |         <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full">
424 |           <TabsList className="grid w-full grid-cols-3 h-auto mb-6">
425 |             <TabsTrigger
426 |               value="upcoming"
427 |               className="py-2 sm:py-3 px-2 sm:px-4 data-[state=active]:bg-primary data-
[state=active]:text>primary-foreg...
428 |
429 |               <UpcomingIcon className="sm:mr-2" style={{ fontSize: 20 }} />
430 |               <span className="hidden sm:inline">Upcoming</span>
431 |               <span className="ml-1 sm:ml-2 text-xs sm:text-sm">({upcomingEvents.length})</span>
432 |             </TabsTrigger>
433 |             <TabsTrigger
434 |               value="calendar"
435 |               className="py-2 sm:py-3 px-2 sm:px-4 data-[state=active]:bg-primary data-
[state=active]:text>primary-foreg...
436 |
437 |               <CalendarMonthIcon className="sm:mr-2" style={{ fontSize: 20 }} />
438 |               <span className="hidden sm:inline">Calendar</span>
439 |             </TabsTrigger>
440 |             <TabsTrigger
441 |               value="past"
442 |               className="py-2 sm:py-3 px-2 sm:px-4 data-[state=active]:bg-primary data-
[state=active]:text>primary-foreg...
443 |
444 |               <HistoryIcon className="sm:mr-2" style={{ fontSize: 20 }} />
445 |               <span className="hidden sm:inline">Past</span>
446 |               <span className="ml-1 sm:ml-2 text-xs sm:text-sm">({pastEvents.length})</span>
447 |             </TabsTrigger>
448 |           </TabsList>
449 |
450 |           {/* Upcoming Events */}
451 |           <TabsContent value="upcoming" className="space-y-4">
452 |             <AnimatePresence mode="wait">
453 |               {loading ? (
454 |                 <motion.div
455 |                   initial={{ opacity: 0, y: 20 }}
456 |                   animate={{ opacity: 1, y: 0 }}
457 |                   exit={{ opacity: 0, y: -20 }}
458 |                 >
459 |                   <Card className="p-12 text-center shadow-elevation-2">
460 |                     <UpcomingIcon
461 |                       style={{ fontSize: 64 }}
462 |                       className="text-muted-foreground/30 mx-auto mb-4 animate-pulse"
463 |                     />
464 |                     <h3 className="text-lg font-bold mb-2">Loading events...</h3>
465 |                   </Card>
466 |                 </motion.div>
467 |               ) : upcomingEvents.length === 0 ? (
468 |                 <motion.div
469 |                   initial={{ opacity: 0, y: 20 }}
470 |                   animate={{ opacity: 1, y: 0 }}
471 |                   exit={{ opacity: 0, y: -20 }}
472 |                 >
473 |                   <Card className="p-12 text-center shadow-elevation-2">
474 |                     <UpcomingIcon
475 |                       style={{ fontSize: 64 }}
476 |                       className="text-muted-foreground/30 mx-auto mb-4"
477 |                     />
478 |                     <h3 className="text-lg font-bold mb-2">No Upcoming Events</h3>
479 |                     <p className="text-muted-foreground text-sm mb-6">
480 |                       Join events to see them here
481 |                     </p>
482 |                     <Button onClick={() => navigate("/events")}>
483 |                       <ExploreIcon className="mr-2" style={{ fontSize: 20 }} />
484 |                       Explore Events
485 |                     </Button>
486 |                   </Card>
487 |                 </motion.div>
488 |               ) : (
489 |                 upcomingEvents.map((event, index) => (
490 |                   <EventCard
491 |                     key={event.id}
492 |                     event={event}
```

```
                    index={index}
                    onClick={() => handleEventClick(event)}
                    onLeave={() => handleLeaveEvent(event.id)}
                    onEdit={() => handleEditEvent(event.id)}
                    getActivityIcon={getActivityIcon}
                    showLeaveButton
                  />
                ))
              )}
            </AnimatePresence>
          </TabsContent>

          {/* Calendar View */}
          <TabsContent value="calendar" className="space-y-4">
            <motion.div
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              className="grid lg:grid-cols-[1fr_400px] gap-6"
            >
              {/* Calendar */}
              <Card className="p-6 shadow-elevation-2">
                <div className="mb-6">
                  <h3 className="text-xl font-bold mb-2">Event Calendar</h3>
                  <div className="flex gap-4 text-sm">
                    <div className="flex items-center gap-2">
                      <div className="w-3 h-3 rounded-full bg-success" />
                      <span className="text-muted-foreground">Running</span>
                    </div>
                    <div className="flex items-center gap-2">
                      <div className="w-3 h-3 rounded-full bg-primary" />
                      <span className="text-muted-foreground">Cycling</span>
                    </div>
                    <div className="flex items-center gap-2">
                      <div className="w-3 h-3 rounded-full bg-warning" />
                      <span className="text-muted-foreground">Walking</span>
                    </div>
                  </div>
                </div>
                <div className="relative">
                  <Calendar
                    mode="single"
                    selected={selectedDate}
                    onSelect={setSelectedDate}
                    className="rounded-md border pointer-events-auto"
                    modifiers={{
                      hasUpcomingEvents: upcomingEventsDates,
                      hasPastEvents: pastEventsDates,
                    }}
                    modifiersClassNames={{
                      hasUpcomingEvents: "has-upcoming-event",
                      hasPastEvents: "has-past-event",
                    }}
                  />
                  <style>{`
                    /* Base styles for dates with events */
                    .rdp-day.has-upcoming-event,
                    .rdp-day.has-past-event {
                      position: relative;
                    }

                    /* Upcoming events - primary/blue dot */
                    .rdp-day.has-upcoming-event .rdp-day_button::after {
                      content: '';
                      position: absolute;
                      bottom: 2px;
                      left: 50%;
                      transform: translateX(-50%);
                      width: 6px;
                      height: 6px;
                      border-radius: 50%;
                      background-color: hsl(var(--primary));
```

```
564 |                     box-shadow: 0 0 0 1px hsl(var(--background));
565 |                   }
566 |
567 |                   /* Past events - orange/amber dot */
568 |                   .rdp-day.has-past-event .rdp-day_button::after {
569 |                     content: '';
570 |                     position: absolute;
571 |                     bottom: 2px;
572 |                     left: 50%;
573 |                     transform: translateX(-50%);
574 |                     width: 6px;
575 |                     height: 6px;
576 |                     border-radius: 50%;
577 |                     background-color: hsl(var(--warning));
578 |                     box-shadow: 0 0 0 1px hsl(var(--background));
579 |                   }
580 |
581 |                   /* Ensure the dot is visible even when date is selected - upcoming */
582 |                   .rdp-day.has-upcoming-event[aria-selected="true"] .rdp-day_button::after {
583 |                     background-color: hsl(var(--primary-foreground));
584 |                     box-shadow: 0 0 0 1px hsl(var(--primary));
585 |                   }
586 |
587 |                   /* Ensure the dot is visible even when date is selected - past */
588 |                   .rdp-day.has-past-event[aria-selected="true"] .rdp-day_button::after {
589 |                     background-color: hsl(var(--warning-foreground, hsl(var(--background))));
590 |                     box-shadow: 0 0 0 1px hsl(var(--warning));
591 |                   }
592 |
593 |                   /* Make the dot more prominent on hover */
594 |                   .rdp-day.has-upcoming-event:hover .rdp-day_button::after,
595 |                   .rdp-day.has-past-event:hover .rdp-day_button::after {
596 |                     width: 7px;
597 |                     height: 7px;
598 |                   }
599 |
600 |                   /* Handle dates with both past and upcoming events (shouldn't happen, but
    just in case) */
601 |                   .rdp-day.has-upcoming-event.has-past-event .rdp-day_button::after {
602 |                     background-color: hsl(var(--primary));
603 |                   }
604 |                 `}</style>
605 |               </div>
606 |             </Card>
607 |
608 |             {/* Selected Date Events */}
609 |             <div className="space-y-4">
610 |               <Card className="p-4 shadow-elevation-2 sticky top-24">
611 |                 <h3 className="font-bold text-lg mb-4">
612 |                   {selectedDate ? format(selectedDate, "MMMM d, yyyy") : "Select a date"}
613 |                 </h3>
614 |                 {eventsForSelectedDate.length === 0 ? (
615 |                   <div className="text-center py-8">
616 |                     <CalendarMonthIcon
617 |                       style={{ fontSize: 48 }}
618 |                       className="text-muted-foreground/30 mx-auto mb-3"
619 |                     />
620 |                     <p className="text-muted-foreground text-sm">
621 |                       No events on this date
622 |                     </p>
623 |                   </div>
624 |                 ) : (
625 |                   <div className="space-y-3 max-h-[500px] overflow-y-auto">
626 |                     {eventsForSelectedDate.map((event) => (
627 |                       <motion.div
628 |                         key={event.id}
629 |                         initial={{ opacity: 0, scale: 0.95 }}
630 |                         animate={{ opacity: 1, scale: 1 }}
631 |                         onClick={() => handleEventClick(event)}
632 |                         className="cursor-pointer"
633 |                       >
634 |                         <Card
```

```
                            className={`p-4 hover:shadow-elevation-2 transition-all duration-200
border-2 ${
                              event.isGreyedOut
                                ? "opacity-50 grayscale border-muted bg-muted/20"
                                : event.isPast
                                ? "border-warning/50 bg-muted/30 opacity-90 hover:border-
warning/70"
                                : "border-border/50 hover:border-primary/30"
                            }`}
                          >
                            <div className="flex items-start gap-3">
                              {getActivityIcon(event.type || "running")}
                              <div className="flex-1 min-w-0">
                                <div className="flex items-center gap-2 mb-1">
                                  <h4 className="font-semibold text-sm truncate">
                                    {event.title || "Untitled Event"}
                                  </h4>
                                  {event.isPast && (
                                    <Badge variant="secondary" className="flex-shrink-0 bg-muted
text-xs">
                                      Completed
                                    </Badge>
                                  )}
                                </div>
                                <div className="flex items-center gap-2 text-xs text-muted-
foreground mb-2">
                                  <EventIcon style={{ fontSize: 14 }} />
                                  <span>{event.time || ""}</span>
                                </div>
                                <div className="flex items-center gap-2 text-xs text-muted-
foreground">
                                  <LocationOnIcon style={{ fontSize: 14 }} />
                                  <span className="truncate">{event.location || ""}</span>
                                </div>
                              </div>
                            </div>
                          </Card>
                        </motion.div>
                      ))}
                    </div>
                  )}
                </Card>
              </div>
            </motion.div>
          </TabsContent>

          {/* Past Events */}
          <TabsContent value="past" className="space-y-4">
            <AnimatePresence mode="wait">
              {loading ? (
                <motion.div
                  initial={{ opacity: 0, y: 20 }}
                  animate={{ opacity: 1, y: 0 }}
                  exit={{ opacity: 0, y: -20 }}
                >
                  <Card className="p-12 text-center shadow-elevation-2">
                    <HistoryIcon
                      style={{ fontSize: 64 }}
                      className="text-muted-foreground/30 mx-auto mb-4 animate-pulse"
                    />
                    <h3 className="text-lg font-bold mb-2">Loading events...</h3>
                  </Card>
                </motion.div>
              ) : pastEvents.length === 0 ? (
                <motion.div
                  initial={{ opacity: 0, y: 20 }}
                  animate={{ opacity: 1, y: 0 }}
                  exit={{ opacity: 0, y: -20 }}
                >
                  <Card className="p-12 text-center shadow-elevation-2">
                    <HistoryIcon
                      style={{ fontSize: 64 }}
                      className="text-muted-foreground/30 mx-auto mb-4"
                    />
                    <h3 className="text-lg font-bold mb-2">No Past Events</h3>
                    <p className="text-muted-foreground text-sm">
```

```
                      Your completed events will appear here
                    </p>
                  </Card>
                </motion.div>
              ) : (
                pastEvents.map((event, index) => (
                  <EventCard
                    key={event.id}
                    event={event}
                    index={index}
                    onClick={() => handleEventClick(event)}
                    onLeave={() => handleLeaveEvent(event.id)}
                    onEdit={() => handleEditEvent(event.id)}
                    getActivityIcon={getActivityIcon}
                    isPast
                  />
                ))
              )}
            </AnimatePresence>
          </TabsContent>
        </Tabs>
      </div>

      {/* Event Detail Modal */}
      {showEventDetail && selectedEvent && (
        <EventDetailModal
          event={selectedEvent}
          onClose={() => {
            setShowEventDetail(false);
            setSelectedEvent(null);
          }}
          onJoin={handleLeaveEvent}
          onEdit={handleEditEvent}
          onDelete={handleDeleteEvent}
        />
      )}

      {showEditEvent && eventBeingEdited && (
        <CreateEventModal
          mode="edit"
          eventToEdit={eventBeingEdited}
          onClose={handleCloseEditModal}
          onUpdateEvent={handleUpdateEvent}
        />
      )}
    </div>
  );
};

// Event Card Component
interface EventCardProps {
  event: Event;
  index: number;
  onClick: () => void;
  onLeave: () => void;
  onEdit?: () => void;
  getActivityIcon: (type: EventType) => JSX.Element;
  showLeaveButton?: boolean;
  isPast?: boolean;
}

const EventCard = ({
  event,
  index,
  onClick,
  onLeave,
  onEdit,
  getActivityIcon,
  showLeaveButton,
  isPast,
}: EventCardProps) => {
```

```
  const { user: currentUser } = useAuth();
  // Check if current user is the event owner
  const isEventOwner = currentUser?.uid && event.hostId === currentUser.uid;

  // Add defensive checks for potentially missing data
  const participantsCount = Array.isArray(event.participants)
    ? event.participants.length
    : 0;
  const eventType = event.type || "running"; // Default to running if type is missing
  const eventTitle = event.title || "Untitled Event";
  const eventDescription = event.description || "";
  const eventDate = event.date || "";
  const eventTime = event.time || "";
  const eventLocation = event.location || "";

  return (
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: index * 0.05 }}
      onClick={onClick}
      className="cursor-pointer"
    >
      <Card
        className={`overflow-hidden shadow-elevation-2 hover:shadow-elevation-3 transition-all duration-300 border-2 ${
          eventIsGreyedOut
            ? "opacity-50 grayscale border-muted bg-muted/20"
            : isPast
            ? "opacity-90 border-warning/50 bg-muted/30 hover:border-warning/70"
            : "border-border/50 hover:border-primary/30"
        }`}
      >
        {/* Gradient Header - Similar to EventDetailsPanel */}
        <div className={`relative p-4 pb-5 rounded-t-lg ${
          isPast
            ? "bg-gradient-to-r from-warning/80 via-warning/70 to-warning/80"
            : "bg-gradient-to-r from-primary via-primary to-success"
        }`}>
          <div className="flex items-start gap-3 pr-16">
            <div className="p-2.5 bg-white/10 backdrop-blur-md rounded-xl border border-white/20 flex-shrink-0">
              {eventType === "running" && <DirectionsRunIcon className="text-white" style={{ fontSize: 20 }} />}
              {eventType === "cycling" && <DirectionsBikeIcon className="text-white" style={{ fontSize: 20 }} />}
              {eventType === "walking" && <DirectionsWalkIcon className="text-white" style={{ fontSize: 20 }} />}
              {eventType === "others" && <FitnessCenterIcon className="text-white" style={{ fontSize: 20 }} />}
            </div>
            <div className="flex-1 min-w-0">
              <div className="flex items-start gap-2 mb-1.5">
                <h3 className="font-bold text-lg sm:text-xl text-white truncate drop-shadow-md">{eventTitle}</h3>
                {event.category === "sponsored" && (
                  <Badge className="flex-shrink-0 bg-warning/20 text-warning border-warning/40">
                    <StarIcon style={{ fontSize: 12 }} className="mr-1" />
                    Sponsored
                  </Badge>
                )}
                {isPast && (
                  <Badge className="flex-shrink-0 bg-white/20 text-white border-white/30">
                    Completed
                  </Badge>
                )}
              </div>
              <p className="text-sm text-white/90 line-clamp-2 drop-shadow-sm">
                {eventDescription}
              </p>
            </div>
          </div>
        </div>

        {/* Event Info Section */}
        <div className="p-5 space-y-4">
          {/* Event Details Grid */}
          <div className="grid grid-cols-1 sm:grid-cols-2 gap-3">
```

```
            <div className="flex items-center gap-2 text-muted-foreground bg-muted/30 rounded-lg p-2.5">
              <EventIcon style={{ fontSize: 18 }} className="text-primary" />
              <div className="flex-1 min-w-0">
                <p className="text-xs text-muted-foreground mb-0.5">Date & Time</p>
                <p className="text-sm font-semibold text-foreground">
                  {eventDate} {eventTime && `at ${eventTime}`}
                </p>
              </div>
            </div>
            <div className="flex items-center gap-2 text-muted-foreground bg-muted/30 rounded-lg p-2.5">
              <LocationOnIcon style={{ fontSize: 18 }} className="text-primary" />
              <div className="flex-1 min-w-0">
                <p className="text-xs text-muted-foreground mb-0.5">Location</p>
                <p className="text-sm font-semibold text-foreground truncate">{eventLocation}</p>
              </div>
            </div>
            <div className="flex items-center gap-2 text-muted-foreground bg-muted/30 rounded-lg p-2.5 sm:col-span-2">
              <PeopleIcon style={{ fontSize: 18 }} className="text-primary" />
              <div className="flex-1 min-w-0">
                <p className="text-xs text-muted-foreground mb-0.5">Participants</p>
                <p className="text-sm font-semibold text-foreground">
                  {participantsCount}
                  {event.maxParticipants ? ` / ${event.maxParticipants}` : ""} joined
                </p>
              </div>
            </div>
          </div>

          {/* Status Badge and Action */}
          <div className="flex items-center justify-between pt-3 border-t border-border">
            <div className="flex items-center gap-2">
              <CheckCircleIcon
                className={isPast ? "text-warning" : "text-success"}
                style={{ fontSize: 20 }}
              />
              <span className={`text-sm font-semibold ${isPast ? "text-warning" : "text-success"}`}>
                {isPast ? "Event Completed" : isEventOwner ? "You're Hosting" : "You're Joined"}
              </span>
            </div>
            {!isPast && (
              <>
                {isEventOwner && onEdit ? (
                  <Button
                    onClick={(e) => {
                      e.stopPropagation();
                      onEdit();
                    }}
                    variant="outline"
                    size="sm"
                    className="h-9 border-2 hover:bg-primary/10 hover:border-primary/50"
                  >
                    <EditIcon className="mr-1" style={{ fontSize: 16 }} />
                    Edit Event
                  </Button>
                ) : showLeaveButton && !isEventOwner ? (
                  <Button
                    onClick={(e) => {
                      e.stopPropagation();
                      onLeave();
                    }}
                    variant="outline"
                    size="sm"
                    className="h-9 border-2 hover:bg-destructive/10 hover:border-destructive/50"
                  >
                    Leave Event
                  </Button>
                ) : null}
              </>
            )}
          </div>
        </div>
```

```
919 |       </Card>
920 |     </motion.div>
921 |   );
922 | };
923 |
924 | export default MyEvents;
925 |
```

File: src/pages/NotFound.tsx

Lines: 25

```
import { useLocation } from "react-router-dom";
import { useEffect } from "react";

const NotFound = () => {
  const location = useLocation();

  useEffect(() => {
    console.error("404 Error: User attempted to access non-existent route:", location.pathname);
  }, [location.pathname]);

  return (
    <div className="flex min-h-screen items-center justify-center bg-muted">
      <div className="text-center">
        <h1 className="mb-4 text-4xl font-bold">404</h1>
        <p className="mb-4 text-xl text-muted-foreground">Oops! Page not found</p>
        <a href="/" className="text-primary underline hover:text-primary/90">
          Return to Home
        </a>
      </div>
    </div>
  );
};

export default NotFound;

```

File: src/pages/PasswordReset.tsx

Lines: 300

```
import { useState, useEffect } from "react";
import { useNavigate, useSearchParams } from "react-router-dom";
import { motion } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import LockIcon from "@mui/icons-material/Lock";
import VisibilityIcon from "@mui/icons-material/Visibility";
import VisibilityOffIcon from "@mui/icons-material/VisibilityOff";
import { confirmPasswordResetCode } from "@/services/authService";
import { toast } from "sonner";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";

/**
 * PasswordReset Page Component
 *
 * This page handles password reset when users click the link from their email.
 * It extracts the reset code (oobCode) from the URL and allows users to set a new password.
 *
 * How it works:
 * 1. User clicks password reset link in email (contains oobCode parameter)
 * 2. Firebase redirects to this page with the code in the URL
 * 3. User enters new password
 * 4. We call confirmPasswordResetCode to complete the reset
 * 5. User is redirected to login page
 */
const PasswordReset = () => {
  const navigate = useNavigate();
  const [searchParams] = useSearchParams();

  // Extract the reset code from URL (Firebase adds it as 'oobCode' parameter)
  const oobCode = searchParams.get("oobCode");
  const mode = searchParams.get("mode"); // Should be "resetPassword"

  const [newPassword, setNewPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [showPassword, setShowPassword] = useState(false);
  const [showConfirmPassword, setShowConfirmPassword] = useState(false);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [success, setSuccess] = useState(false);

  // Check if we have a valid reset code
  useEffect(() => {
    if (!oobCode || mode !== "resetPassword") {
      setError("Invalid or missing reset link. Please request a new password reset email.");
    }
  }, [oobCode, mode]);

  /**
   * Handle password reset submission
   * Validates passwords match and calls Firebase to confirm the reset
   */
  const handleResetPassword = async () => {
    setError(null);

    // Validate inputs
    if (!newPassword) {
      setError("Please enter a new password.");
      return;
    }

    if (newPassword.length < 6) {
      setError("Password must be at least 6 characters long.");
      return;
    }

```

```
    if (newPassword !== confirmPassword) {
      setError("Passwords do not match. Please try again.");
      return;
    }

    if (!oobCode) {
      setError("Invalid reset code. Please use the link from your email.");
      return;
    }

    setLoading(true);

    try {
      // Call the service function to confirm password reset
      await confirmPasswordResetCode(oobCode, newPassword);

      // Success!
      setSuccess(true);
      toast.success("Password reset successful! You can now sign in with your new password.");

      // Redirect to login after 2 seconds
      setTimeout(() => {
        navigate("/login", { replace: true });
      }, 2000);
    } catch (err: any) {
      console.error("'L Error resetting password:", err);
      setError(err.message || "Failed to reset password. Please try again.");
    } finally {
      setLoading(false);
    }
  };

  // If no valid code, show error message
  if (!oobCode || mode !== "resetPassword") {
    return (
      <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-
background via-background to-...
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="w-full max-w-md space-y-6"
        >
          <div className="text-center space-y-4">
            <div className="flex justify-center">
              <DirectionsRunIcon className="text-primary" style={{ fontSize: 64 }} />
            </div>
            <h1 className="text-3xl font-bold">Invalid Reset Link</h1>
            <p className="text-muted-foreground">
              {error || "This password reset link is invalid or has expired."}
            </p>
            <Button
              onClick={() => navigate("/login")}
              className="w-full"
            >
              Back to Login
            </Button>
          </div>
        </motion.div>
      </div>
    );
  }

  // Success state
  if (success) {
    return (
      <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-
background via-background to-...
        <motion.div
          initial={{ opacity: 0, scale: 0.9 }}
          animate={{ opacity: 1, scale: 1 }}
          className="w-full max-w-md space-y-6 text-center"
        >
          <div className="flex justify-center">
```

```
138 |             <motion.div
139 |               initial={{ scale: 0 }}
140 |               animate={{ scale: 1 }}
141 |               transition={{ type: "spring", stiffness: 200 }}
142 |               className="w-20 h-20 rounded-full bg-success/20 flex items-center justify-center"
143 |             >
144 |               <LockIcon className="text-success" style={{ fontSize: 40 }} />
145 |             </motion.div>
146 |           </div>
147 |           <h1 className="text-3xl font-bold">Password Reset Successful!</h1>
148 |           <p className="text-muted-foreground">
149 |             Your password has been reset. Redirecting to login...
150 |           </p>
151 |         </motion.div>
152 |       </div>
153 |     );
154 |   }
155 |
156 |   // Main reset form
157 |   return (
158 |     <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-
background via-background to-mu...
159 |       <motion.div
160 |         initial={{ opacity: 0, y: 20 }}
161 |         animate={{ opacity: 1, y: 0 }}
162 |         className="w-full max-w-md space-y-6"
163 |       >
164 |         {/* Logo and Title */}
165 |         <div className="text-center space-y-4">
166 |           <motion.div
167 |             initial={{ scale: 0 }}
168 |             animate={{ scale: 1 }}
169 |             transition={{ type: "spring", stiffness: 200 }}
170 |             className="flex justify-center"
171 |           >
172 |             <div className="w-20 h-20 rounded-full bg-primary/10 flex items-center justify-
center">
173 |               <LockIcon className="text-primary" style={{ fontSize: 40 }} />
174 |             </div>
175 |           </motion.div>
176 |           <h1 className="text-3xl font-bold">Reset Your Password</h1>
177 |           <p className="text-muted-foreground">
178 |             Enter your new password below
179 |           </p>
180 |         </div>
181 |
182 |         {/* Error Alert */}
183 |         {error && (
184 |           <motion.div
185 |             initial={{ opacity: 0, y: -10 }}
186 |             animate={{ opacity: 1, y: 0 }}
187 |             className="bg-destructive/10 border border-destructive/20 text-destructive px-4 py-3
rounded-lg text-sm"
188 |           >
189 |             {error}
190 |           </motion.div>
191 |         )}
192 |
193 |         {/* Password Reset Form */}
194 |         <motion.div
195 |           initial={{ opacity: 0, y: 20 }}
196 |           animate={{ opacity: 1, y: 0 }}
197 |           transition={{ delay: 0.2 }}
198 |           className="space-y-4"
199 |         >
200 |           {/* New Password Input */}
201 |           <div className="space-y-2">
202 |             <label htmlFor="newPassword" className="text-sm font-medium text-muted-foreground">
203 |               New Password
204 |             </label>
205 |             <div className="relative">
206 |               <LockIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-
foreground" style={{ fo...
207 |                 <Input
208 |                   id="newPassword"
```

```
                type={showPassword ? "text" : "password"}
                placeholder="Enter new password"
                value={newPassword}
                onChange={(e) => setNewPassword(e.target.value)}
                className="pl-10 pr-10 h-12 text-base"
                disabled={loading}
              />
              <button
                type="button"
                onClick={() => setShowPassword(!showPassword)}
                className="absolute right-3 top-1/2 transform -translate-y-1/2 text-muted-foreground hover:text>foregrou...
                {showPassword ? (
                  <VisibilityOffIcon style={{ fontSize: 20 }} />
                ) : (
                  <VisibilityIcon style={{ fontSize: 20 }} />
                )}
              </button>
            </div>
            <p className="text-xs text-muted-foreground">
              Must be at least 6 characters long
            </p>
          </div>

          {/* Confirm Password Input */}
          <div className="space-y-2">
            <label htmlFor="confirmPassword" className="text-sm font-medium text-muted-foreground">
              Confirm New Password
            </label>
            <div className="relative">
              <LockIcon className="absolute left-3 top-1/2 transform -translate-y-1/2 text-muted-foreground" style={{ fo...
              <input
                id="confirmPassword"
                type={showConfirmPassword ? "text" : "password"}
                placeholder="Confirm new password"
                value={confirmPassword}
                onChange={(e) => setConfirmPassword(e.target.value)}
                onKeyPress={(e) => {
                  if (e.key === "Enter") {
                    handleResetPassword();
                  }
                }}
                className="pl-10 pr-10 h-12 text-base"
                disabled={loading}
              />
              <button
                type="button"
                onClick={() => setShowConfirmPassword(!showConfirmPassword)}
                className="absolute right-3 top-1/2 transform -translate-y-1/2 text-muted-foreground hover:text>foregrou...
                {showConfirmPassword ? (
                  <VisibilityOffIcon style={{ fontSize: 20 }} />
                ) : (
                  <VisibilityIcon style={{ fontSize: 20 }} />
                )}
              </button>
            </div>
          </div>

          {/* Reset Button */}
          <Button
            onClick={handleResetPassword}
            disabled={loading || !newPassword || !confirmPassword}
            className="w-full h-12 text-base font-semibold shadow-elevation-3 hover:shadow-elevation-4 transition-all du...
            {loading ? (
              <>
                <div className="mr-3 h-5 w-5 border-2 border-primary-foreground border-t-transparent rounded-full animat...
                Resetting Password...
              </>
            ) : (
```

```
              "Reset Password"
            )}
          </Button>

          {/* Back to Login Link */}
          <Button
            onClick={() => navigate("/login")}
            variant="ghost"
            className="w-full h-10 text-sm"
          >
            Back to Login
          </Button>
        </motion.div>
      </motion.div>
    </div>
  );
};

export default PasswordReset;

```

File: src/pages/ProfileSetup.tsx

Lines: 995

```
   1 | import { useState, useEffect, useRef } from "react";
   2 | import { motion, AnimatePresence } from "framer-motion";
   3 | import { Button } from "@/components/ui/button";
   4 | import { Input } from "@/components/ui/input";
   5 | import { Label } from "@/components/ui/label";
   6 | import { Checkbox } from "@/components/ui/checkbox";
   7 | import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/
select";
   8 | import { useNavigate } from "react-router-dom";
   9 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
  10 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
  11 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
  12 | import CameraAltIcon from "@mui/icons-material/CameraAlt";
  13 | import AddPhotoAlternateIcon from "@mui/icons-material/AddPhotoAlternate";
  14 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
  15 | import PersonIcon from "@mui/icons-material/Person";
  16 | import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";
  17 | import SettingsIcon from "@mui/icons-material/Settings";
  18 | import ArrowBackIcon from "@mui/icons-material/ArrowBack";
  19 | import DeleteIcon from "@mui/icons-material/Delete";
  20 | import Avatar from "@mui/material/Avatar";
  21 | import { toast } from "sonner";
  22 | import { useAuth } from "@/hooks/useAuth";
  23 | import { updateUserProfile } from "@/services/authService";
  24 | import { FitnessLevel, RadiusPreference, VisibilitySettings } from "@/contexts/UserContext";
  25 | import { SearchFilter } from "@/services/matchingService";
  26 | import { DEFAULT_AVATARS, generateUserAvatar } from "@/lib/avatars";
  27 | import { storage } from "@/services/firebase";
  28 | import { ref, uploadBytes, getDownloadURL } from "firebase/storage";
  29 |
  30 | const ProfileSetup = () => {
  31 |   const navigate = useNavigate();
  32 |   const { user, loading: authLoading } = useAuth();
  33 |   const [username, setUsername] = useState("");
  34 |   const [selectedActivities, setSelectedActivities] = useState<("running" | "cycling" |
"walking" | string)[]>(["running"]);
  35 |   const [customActivity, setCustomActivity] = useState("");
  36 |   const [showCustomInput, setShowCustomInput] = useState(false);
  37 |   const [gender, setGender] = useState("");
  38 |   const [fitnessLevel, setFitnessLevel] = useState<FitnessLevel>("intermediate");
  39 |   const [visibleToAllLevels, setVisibleToAllLevels] = useState(true);
  40 |   const [allowedLevels, setAllowedLevels] = useState<FitnessLevel[]>(["beginner",
"intermediate", "pro"]);
  41 |   const [searchFilter, setSearchFilter] = useState<SearchFilter>("all");
  42 |   const [radiusPreference, setRadiusPreference] = useState<RadiusPreference>("normal");
  43 |   const [saving, setSaving] = useState(false);
  44 |
  45 |   // Stepper state
  46 |   const [currentStep, setCurrentStep] = useState(1);
  47 |
  48 |   // Photo/Avatar state
  49 |   const [showAvatarPicker, setShowAvatarPicker] = useState(false);
  50 |   const [selectedPhotoUrl, setSelectedPhotoUrl] = useState<string | null>(null);
  51 |   const [uploadingPhoto, setUploadingPhoto] = useState(false);
  52 |   const fileInputRef = useRef<HTMLInputElement>(null);
  53 |
  54 |   // Additional photos state (up to 3)
  55 |   const [photos, setPhotos] = useState<string[]>([]);
  56 |   const photosInputRef = useRef<HTMLInputElement>(null);
  57 |
  58 |   // Redirect if not authenticated
  59 |   useEffect(() => {
  60 |     if (!authLoading && !user) {
  61 |       navigate("/");
  62 |     }
  63 |   }, [user, authLoading, navigate]);
  64 |
  65 |   // Load existing profile data if available
  66 |   useEffect(() => {
```

```
  67 |     const loadProfile = async () => {
  68 |       if (user) {
  69 |         const { getUserData } = await import("@/services/authService");
  70 |         const userData = await getUserData(user.uid);
  71 |         if (userData) {
  72 |           if (userData.name) setUsername(userData.name);
  73 |           // Load activities - support both old single activity and new array format
  74 |           if (userData.activities && Array.isArray(userData.activities)) {
  75 |             setSelectedActivities(userData.activities);
  76 |           } else if (userData.activity) {
  77 |             setSelectedActivities([userData.activity as "running" | "cycling" | "walking"]);
  78 |           }
  79 |           if (userData.gender) setGender(userData.gender);
  80 |           if (userData.fitnessLevel) setFitnessLevel(userData.fitnessLevel);
  81 |           if (userData.visibility) {
  82 |             setVisibleToAllLevels(userData.visibility.visibleToAllLevels ?? true);
  83 |             setAllowedLevels(userData.visibility.allowedLevels || ["beginner", "intermediate",
"pro"]);
  84 |           }
  85 |           if (userData.searchFilter) setSearchFilter(userData.searchFilter);
  86 |           if (userData.radiusPreference) setRadiusPreference(userData.radiusPreference);
  87 |           // Load existing photo URL
  88 |           if (userData.photoURL) setSelectedPhotoUrl(userData.photoURL);
  89 |           // Load additional photos
  90 |           if (userData.photos && Array.isArray(userData.photos)) {
  91 |             setPhotos(userData.photos);
  92 |           }
  93 |         }
  94 |       }
  95 |     };
  96 |     loadProfile();
  97 |   }, [user]);
  98 |
  99 |   // Handle photo file upload
 100 |   const handlePhotoUpload = async (event: React.ChangeEvent<HTMLInputElement>) => {
 101 |     const file = event.target.files?.[0];
 102 |     if (!file || !user) return;
 103 |
 104 |     // Validate file type
 105 |     if (!file.type.startsWith('image/')) {
 106 |       toast.error("Please select an image file");
 107 |       return;
 108 |     }
 109 |
 110 |     // Validate file size (max 5MB)
 111 |     if (file.size > 5 * 1024 * 1024) {
 112 |       toast.error("Image must be less than 5MB");
 113 |       return;
 114 |     }
 115 |
 116 |     setUploadingPhoto(true);
 117 |     try {
 118 |       // Create a reference to the file in Firebase Storage
 119 |       const storageRef = ref(storage, `profile-photos/${user.uid}/${Date.now()}_${file.name}`);
 120 |
 121 |       // Upload the file
 122 |       const snapshot = await uploadBytes(storageRef, file);
 123 |
 124 |       // Get the download URL
 125 |       const downloadUrl = await getDownloadURL(snapshot.ref);
 126 |
 127 |       setSelectedPhotoUrl(downloadUrl);
 128 |       setShowAvatarPicker(false);
 129 |       toast.success("Photo uploaded successfully!");
 130 |     } catch (error: any) {
 131 |       console.error("Error uploading photo:", error);
 132 |       toast.error("Failed to upload photo. Please try again.");
 133 |     } finally {
 134 |       setUploadingPhoto(false);
 135 |     }
 136 |   };
 137 |
```

```
  // Handle default avatar selection
  const handleAvatarSelect = (avatarUrl: string) => {
    setSelectedPhotoUrl(avatarUrl);
    setShowAvatarPicker(false);
    toast.success("Avatar selected!");
  };

  // Handle additional photos upload (up to 3)
  const handlePhotosUpload = (e: React.ChangeEvent<HTMLInputElement>) => {
    const files = e.target.files;
    if (!files) return;

    if (photos.length >= 3) {
      toast.error("Maximum 3 photos allowed");
      return;
    }

    Array.from(files).forEach(file => {
      if (photos.length >= 3) return;

      // Validate file type
      if (!file.type.startsWith('image/')) {
        toast.error("Please select image files only");
        return;
      }

      // Validate file size (max 5MB)
      if (file.size > 5 * 1024 * 1024) {
        toast.error("Each image must be less than 5MB");
        return;
      }

      const reader = new FileReader();
      reader.onloadend = () => {
        setPhotos(prev => [...prev, reader.result as string]);
      };
      reader.readAsDataURL(file);
    });
  };

  const handleRemovePhoto = (index: number) => {
    setPhotos(prev => prev.filter((_, i) => i !== index));
  };

  // Get current display photo URL
  const getDisplayPhotoUrl = (): string => {
    if (selectedPhotoUrl) return selectedPhotoUrl;
    if (user?.photoURL) return user.photoURL;
    return generateUserAvatar(user?.displayName || user?.email || 'User');
  };

  // Step navigation validation
  const validateStep = (step: number): boolean => {
    if (step === 1) {
      if (!username.trim()) {
        toast.error("Please enter a username");
        return false;
      }
      return true;
    }
    if (step === 2) {
      if (selectedActivities.length === 0) {
        toast.error("Please select at least one preferred activity");
        return false;
      }
      return true;
    }
    return true; // Step 3 has no required fields
  };

  const handleNext = () => {
```

```
209 |     if (validateStep(currentStep)) {
210 |       setCurrentStep(currentStep + 1);
211 |     }
212 |   };
213 |
214 |   const handleBack = () => {
215 |     setCurrentStep(currentStep - 1);
216 |   };
217 |
218 |   const handleComplete = async () => {
219 |     if (!user) {
220 |       toast.error("Please sign in first");
221 |       navigate("/");
222 |       return;
223 |     }
224 |
225 |     if (!username.trim()) {
226 |       toast.error("Please enter a username");
227 |       return;
228 |     }
229 |
230 |     setSaving(true);
231 |     try {
232 |       // Ensure at least one activity is selected
233 |       if (selectedActivities.length === 0) {
234 |         toast.error("Please select at least one preferred activity");
235 |         setSaving(false);
236 |         return;
237 |       }
238 |
239 |       const visibility: VisibilitySettings = {
240 |         visibleToAllLevels: visibleToAllLevels,
241 |         allowedLevels: visibleToAllLevels ? ["beginner", "intermediate", "pro"] : allowedLevels
242 |       };
243 |
244 |       // Get primary activity (first selected, or running if available, otherwise first)
245 |       const primaryActivity = selectedActivities.find(a => ["running", "cycling",
"walking"].includes(a)) as "running" |...
246 |
247 |       await updateUserProfile(user.uid, {
248 |         name: username.trim(),
249 |         activity: primaryActivity, // Keep for backward compatibility (use first standard
activity or first selected)
250 |         activities: selectedActivities.filter(a => a.trim() !== "") as ("running" | "cycling" |
"walking")[], // Save as
251 |         gender: gender || null,
252 |         fitnessLevel: fitnessLevel,
253 |         visibility: visibility,
254 |         searchFilter: searchFilter,
255 |         radiusPreference: radiusPreference,
256 |         photoURL: selectedPhotoUrl || user.photoURL || null, // Save the selected/uploaded photo
257 |         photos: photos.length > 0 ? photos : null // Save additional photos
258 |       });
259 |       toast.success("Profile created successfully!");
260 |       navigate("/", { replace: true });
261 |     } catch (error: any) {
262 |       console.error("Error saving profile:", error);
263 |       toast.error("Failed to save profile. Please try again.");
264 |     } finally {
265 |       setSaving(false);
266 |     }
267 |   };
268 |
269 |   const activities: Array<{
270 |     id: "running" | "cycling" | "walking";
271 |     label: string;
272 |     icon: typeof DirectionsRunIcon;
273 |     color: string;
274 |   }> = [
275 |     { id: "running", label: "Running", icon: DirectionsRunIcon, color: "success" },
276 |     { id: "cycling", label: "Cycling", icon: DirectionsBikeIcon, color: "primary" },
277 |     { id: "walking", label: "Walking", icon: DirectionsWalkIcon, color: "warning" },
278 |   ];
279 |
```

```
  const genderOptions = ["Male", "Female", "Other", "Prefer not to say"];
  const fitnessLevelOptions: { value: FitnessLevel; label: string; description: string; color: string; bgColor: string; ...
    {
      value: "beginner",
      label: "Beginner",
      description: "Just starting out or getting back into fitness. You're building a foundation and learning the basics.",
      color: "text-blue-600 dark:text-blue-400",
      bgColor: "bg-blue-50 dark:bg-blue-950/30",
      borderColor: "border-blue-200 dark:border-blue-800"
    },
    {
      value: "intermediate",
      label: "Intermediate",
      description: "Regular exerciser with a consistent routine. You can handle moderate workouts and are comfortable wi...
      color: "text-green-600 dark:text-green-400",
      bgColor: "bg-green-50 dark:bg-green-950/30",
      borderColor: "border-green-200 dark:border-green-800"
    },
    {
      value: "pro",
      label: "Pro",
      description: "Advanced athlete with high performance goals. You train regularly at high intensity and push your li...
      color: "text-purple-600 dark:text-purple-400",
      bgColor: "bg-purple-50 dark:bg-purple-950/30",
      borderColor: "border-purple-200 dark:border-purple-800"
    }
  ];
  const radiusOptions: { value: RadiusPreference; label: string }[] = [
    { value: "nearby", label: "Nearby" },
    { value: "normal", label: "Normal" },
    { value: "wide", label: "Wide" }
  ];

  const searchFilterOptions: { value: SearchFilter; label: string }[] = [
    { value: "all", label: "All Levels" },
    { value: "beginner", label: "Beginner" },
    { value: "intermediate", label: "Intermediate" },
    { value: "pro", label: "Pro" }
  ];

  const handleLevelToggle = (level: FitnessLevel) => {
    if (allowedLevels.includes(level)) {
      setAllowedLevels(allowedLevels.filter(l => l !== level));
    } else {
      setAllowedLevels([...allowedLevels, level]);
    }
  };

  const handleActivityToggle = (activityId: "running" | "cycling" | "walking") => {
    if (selectedActivities.includes(activityId)) {
      setSelectedActivities(selectedActivities.filter(a => a !== activityId));
    } else {
      setSelectedActivities([...selectedActivities, activityId]);
    }
  };

  const handleAddCustomActivity = () => {
    if (customActivity.trim() && !selectedActivities.includes(customActivity.trim())) {
      setSelectedActivities([...selectedActivities, customActivity.trim()]);
      setCustomActivity("");
      setShowCustomInput(false);
    }
  };

  const handleRemoveActivity = (activityToRemove: string) => {
    setSelectedActivities(selectedActivities.filter(a => a !== activityToRemove));
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/5 via-background to-success/5 flex flex-col items-center...
      <motion.div
```

```
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        className="w-full max-w-md space-y-8"
      >
        {/* Header */}
        <div className="text-center space-y-4">
          <h1 className="text-3xl font-bold text-foreground">Complete Your Profile</h1>

          {/* Step Indicator */}
          <div className="flex items-center justify-center gap-2 mb-2">
            {/* Step 1 */}
            <div className="flex items-center">
              <div className={`flex items-center justify-center w-10 h-10 rounded-full border-2
transition-all ${
                currentStep >= 1 ? 'bg-primary border-primary text-primary-foreground' : 'border-
border text-muted-foreg}>.
                {currentStep > 1 ? <CheckCircleIcon style={{ fontSize: 20 }} /> : <span
className="text-sm font-semibold...
              </div>
              <span className={`ml-2 text-sm font-medium hidden sm:inline ${
                currentStep === 1 ? 'text-foreground' : currentStep > 1 ? 'text-primary' : 'text-
muted-foreground'    }`}>About You</span>
            </div>

            {/* Connector Line */}
            <div className={`h-0.5 w-12 transition-all ${
              currentStep > 1 ? 'bg-primary' : 'bg-border'
            }`} />

            {/* Step 2 */}
            <div className="flex items-center">
              <div className={`flex items-center justify-center w-10 h-10 rounded-full border-2
transition-all ${
                currentStep >= 2 ? 'bg-primary border-primary text-primary-foreground' : 'border-
border text-muted-foreg}>.
                {currentStep > 2 ? <CheckCircleIcon style={{ fontSize: 20 }} /> : <span
className="text-sm font-semibold...
              </div>
              <span className={`ml-2 text-sm font-medium hidden sm:inline ${
                currentStep === 2 ? 'text-foreground' : currentStep > 2 ? 'text-primary' : 'text-
muted-foreground'    }`}>Fitness</span>
            </div>

            {/* Connector Line */}
            <div className={`h-0.5 w-12 transition-all ${
              currentStep > 2 ? 'bg-primary' : 'bg-border'
            }`} />

            {/* Step 3 */}
            <div className="flex items-center">
              <div className={`flex items-center justify-center w-10 h-10 rounded-full border-2
transition-all ${
                currentStep >= 3 ? 'bg-primary border-primary text-primary-foreground' : 'border-
border text-muted-foreg}>.
                  <span className="text-sm font-semibold">3</span>
                </div>
                <span className={`ml-2 text-sm font-medium hidden sm:inline ${
                  currentStep === 3 ? 'text-foreground' : 'text-muted-foreground'
                }`}>Preferences</span>
              </div>
            </div>
          </div>

          {/* Step Content */}
          <AnimatePresence mode="wait">
            {/* Step 1: About You */}
            {currentStep === 1 && (
              <motion.div
                key="step1"
                initial={{ opacity: 0, x: 20 }}
                animate={{ opacity: 1, x: 0 }}
                exit={{ opacity: 0, x: -20 }}
                transition={{ duration: 0.3 }}
                className="space-y-6"
              >
```

```
422 |               {/* Tip Card */}
423 |               <div className="bg-primary/10 border border-primary/20 rounded-xl p-4 flex items-
start gap-3">
424 |                 <PersonIcon className="text-primary flex-shrink-0 mt-0.5" style={{ fontSize:
24 }} />
425 |                 <div>
426 |                   <p className="text-sm font-medium text-foreground">
427 |                     Your profile helps others recognize you when matching nearby!
428 |                   </p>
429 |                 </div>
430 |               </div>
431 |
432 |               {/* Profile Photo - Clickable */}
433 |               <motion.div
434 |                 initial={{ scale: 0.8 }}
435 |                 animate={{ scale: 1 }}
436 |                 transition={{ duration: 0.3 }}
437 |                 className="flex flex-col items-center gap-3"
438 |               >
439 |           <div className="relative group">
440 |             <button
441 |               type="button"
442 |               onClick={() => setShowAvatarPicker(true)}
443 |               className="relative rounded-full focus:outline-none focus:ring-4 focus:ring-
primary/30 transition-all dura...
444 |             >
445 |               <Avatar
446 |                 sx={{ width: 96, height: 96 }}
447 |                 alt="Profile"
448 |                 src={getDisplayPhotoUrl()}
449 |               />
450 |               {/* Camera overlay on hover */}
451 |               <div className="absolute inset-0 bg-black/40 rounded-full flex items-center
justify-center opacity-0 group...
452 |                 <CameraAltIcon className="text-white" style={{ fontSize: 32 }} />
453 |               </div>
454 |               {/* Selected indicator */}
455 |               {selectedPhotoUrl && (
456 |                 <div className="absolute -bottom-1 -right-1 bg-green-500 rounded-full p-1">
457 |                   <CheckCircleIcon className="text-white" style={{ fontSize: 18 }} />
458 |                 </div>
459 |               )}
460 |             </button>
461 |           </div>
462 |           <button
463 |             type="button"
464 |             onClick={() => setShowAvatarPicker(true)}
465 |             className="text-sm text-primary hover:text-primary/80 font-medium transition-colors"
466 |           >
467 |             {selectedPhotoUrl ? "Change Photo" : "Add Photo"}
468 |           </button>
469 |               </motion.div>
470 |
471 |               {/* Avatar Picker Modal */}
472 |               <AnimatePresence>
473 |                 {showAvatarPicker && (
474 |             <motion.div
475 |               initial={{ opacity: 0 }}
476 |               animate={{ opacity: 1 }}
477 |               exit={{ opacity: 0 }}
478 |               className="fixed inset-0 bg-black/60 z-50 flex items-center justify-center p-4"
479 |               onClick={() => setShowAvatarPicker(false)}
480 |             >
481 |               <motion.div
482 |                 initial={{ scale: 0.9, opacity: 0 }}
483 |                 animate={{ scale: 1, opacity: 1 }}
484 |                 exit={{ scale: 0.9, opacity: 0 }}
485 |                 className="bg-card rounded-2xl p-6 max-w-md w-full max-h-[80vh] overflow-y-auto
shadow-xl"
486 |                 onClick={(e) => e.stopPropagation()}
487 |               >
488 |                 <div className="flex justify-between items-center mb-4">
489 |                   <h3 className="text-xl font-bold text-foreground">Choose Your Photo</h3>
490 |                   <button
491 |                     type="button"
492 |                     onClick={() => setShowAvatarPicker(false)}
```

```
                    className="text-muted-foreground hover:text-foreground transition-colors
text-2xl leading-none"
                  >
                    ×
                  </button>
                </div>

                {/* Upload Photo Option */}
                <div className="mb-6">
                  <input
                    ref={fileInputRef}
                    type="file"
                    accept="image/*"
                    onChange={handlePhotoUpload}
                    className="hidden"
                  />
                  <button
                    type="button"
                    onClick={() => fileInputRef.current?.click()}
                    disabled={uploadingPhoto}
                    className="w-full flex items-center justify-center gap-3 p-4 rounded-xl
border-2 border-dashed borde...
                  >
                    {uploadingPhoto ? (
                      <>
                        <div className="w-6 h-6 border-2 border-primary border-t-transparent
rounded-full animate-spin" ... <span className="font-medium text-primary">Uploading...</span>
                      </>
                    ) : (
                      <>
                        <AddPhotoAlternateIcon className="text-primary" style={{ fontSize:
28 }} />
                        <span className="font-medium text-primary">Upload Your Photo</span>
                      </>
                    )}
                  </button>
                  <p className="text-xs text-muted-foreground text-center mt-2">
                    Max 5MB • JPG, PNG, GIF
                  </p>
                </div>

                {/* Divider */}
                <div className="flex items-center gap-3 mb-6">
                  <div className="flex-1 h-px bg-border" />
                  <span className="text-sm text-muted-foreground">or choose an avatar</span>
                  <div className="flex-1 h-px bg-border" />
                </div>

                {/* Default Avatars Grid */}
                <div className="grid grid-cols-4 gap-3">
                  {DEFAULT_AVATARS.map((avatar) => (
                    <button
                      key={avatar.id}
                      type="button"
                      onClick={() => handleAvatarSelect(avatar.url)}
                      className={`relative rounded-xl p-2 transition-all duration-200
hover:scale-105 ${
                        selectedPhotoUrl === avatar.url
                          ? "bg-primary/20 ring-2 ring-primary"
                          : "bg-secondary hover:bg-secondary/80"
                      }`}
                    >
                      <img
                        src={avatar.url}
                        alt={avatar.name}
                        className="w-full aspect-square rounded-lg"
                      />
                      {selectedPhotoUrl === avatar.url && (
                        <div className="absolute -top-1 -right-1 bg-primary rounded-full p-0.5">
                          <CheckCircleIcon className="text-white" style={{ fontSize: 14 }} />
                        </div>
                      )}
                    </button>
                  ))}
                </div>
```

```
                {/* Close button */}
                <Button
                  type="button"
                  onClick={() => setShowAvatarPicker(false)}
                  className="w-full mt-6"
                  variant="outline"
                >
                  Done
                </Button>
              </motion.div>
                </motion.div>
              )}
              </AnimatePresence>

              {/* Username Input */}
              <div className="space-y-2">
                <Label htmlFor="username" className="text-lg font-semibold text-
foreground">Username <span className="te...
                <input
                  id="username"
                  type="text"
                  placeholder="Enter your username (required)"
                  value={username}
                  onChange={(e) => setUsername(e.target.value)}
                  className="h-12 text-base"
                  required
                />
                <p className="text-xs text-muted-foreground">
                  This username will be displayed to other users
                </p>
              </div>

              {/* Gender Selection (Optional) */}
              <div className="space-y-3">
                <Label className="text-lg font-semibold text-foreground">Gender <span
className="text-sm font-normal tex...
                <div className="grid grid-cols-2 gap-3">
                  {genderOptions.map((option) => (
                    <button
                      key={option}
                      onClick={() => setGender(option)}
                      className={`
                        p-3 rounded-lg border-2 text-sm font-medium transition-all duration-300
                        ${
                          gender === option
                            ? "border-primary bg-primary/10 text-primary"
                            : "border-border bg-card text-foreground hover:bg-secondary"
                        }
                      `}
                    >
                      {option}
                    </button>
                  ))}
                </div>
              </div>

              {/* Additional Photos Section (Optional) */}
              <div className="space-y-3">
                <Label className="text-lg font-semibold text-foreground">Additional Photos <span
className="text-sm font-...
                <p className="text-xs text-muted-foreground">Add up to 3 photos to showcase your
fitness journey</p>
                <div className="grid grid-cols-3 gap-3">
                  {photos.map((photo, index) => (
                    <div key={index} className="relative aspect-square rounded-lg overflow-
hidden border-2 border-border">
                      <img src={photo} alt={`Photo ${index + 1}`} className="w-full h-full
object-cover" />
                      <button
                        type="button"
                        onClick={() => handleRemovePhoto(index)}
                        className="absolute top-1 right-1 p-1 bg-destructive text-destructive-
foreground rounded-full ho...>
                        <DeleteIcon style={{ fontSize: 14 }} />
                      </button>
                    </div>
```

```
                  ))}
                  {photos.length < 3 && (
                    <label className="aspect-square border-2 border-dashed border-border rounded-lg flex flex-col items-...
                      <input
                        ref={photosInputRef}
                        type="file"
                        accept="image/*"
                        multiple
                        className="hidden"
                        onChange={handlePhotosUpload}
                      />
                      <AddPhotoAlternateIcon className="text-muted-foreground" style={{ fontSize: 28 }} />
                      <span className="text-xs text-muted-foreground mt-1">Add Photo</span>
                    </label>
                  )}
                </div>
                {photos.length > 0 && photos.length < 3 && (
                  <p className="text-xs text-muted-foreground">
                    You can add {3 - photos.length} more photo{3 - photos.length > 1 ? 's' : ''}
                  </p>
                )}
              </div>
            </motion.div>
          )}

          {/* Step 2: Your Fitness */}
          {currentStep === 2 && (
            <motion.div
              key="step2"
              initial={{ opacity: 0, x: 20 }}
              animate={{ opacity: 1, x: 0 }}
              exit={{ opacity: 0, x: -20 }}
              transition={{ duration: 0.3 }}
              className="space-y-6"
            >
              {/* Tip Card */}
              <div className="bg-primary/10 border border-primary/20 rounded-xl p-4 flex items-start gap-3">
                <FitnessCenterIcon className="text-primary flex-shrink-0 mt-0.5" style={{ fontSize: 24 }} />
                <div>
                  <p className="text-sm font-medium text-foreground">
                    Select activities you enjoy - you'll match with others doing the same!
                  </p>
                </div>
              </div>

              {/* Activity Selection */}
              <div className="space-y-3">
                <Label className="text-lg font-semibold text-foreground">Preferred Activities</Label>
          <div className="grid grid-cols-3 gap-3">
            {activities.map((act) => {
              const Icon = act.icon;
              const isSelected = selectedActivities.includes(act.id);
              return (
                <button
                  key={act.id}
                  type="button"
                  onClick={() => handleActivityToggle(act.id)}
                  className={`
                    flex flex-col items-center justify-center p-4 rounded-lg border-2 transition-all duration-300 relati... ${
                      isSelected
                        ? `border-${act.color} bg-${act.color}/10`
                        : "border-border bg-card hover:bg-secondary"
                    }
                  `}
                >
                  {isSelected && (
                    <div className="absolute top-2 right-2 w-5 h-5 bg-primary rounded-full flex items-center justify-cen...  <span className="text-white text-xs">' </span>
                    </div>
                  )}
```

```
706 |                   <Icon
707 |                     className={isSelected ? `text-${act.color}` : "text-muted-foreground"}
708 |                     style={{ fontSize: 32 }}
709 |                   />
710 |                   <span className={`text-xs mt-2 font-medium ${isSelected ? `text-${act.color}
`711"text-muted-foreground...
711 |                     {act.label}
712 |                   </span>
713 |                 </button>
714 |               );
715 |             })}
716 |           </div>
717 |
718 |           {/* Custom Activity Input */}
719 |           <div className="space-y-2">
720 |             {!showCustomInput ? (
721 |               <button
722 |                 type="button"
723 |                 onClick={() => setShowCustomInput(true)}
724 |                 className="w-full p-3 rounded-lg border-2 border-dashed border-border bg-card
hover:bg-secondary text-mu...
725 |                 >
726 |                 + Add Other Activity
727 |               </button>
728 |             ) : (
729 |               <div className="flex gap-2">
730 |                 <Input
731 |                   type="text"
732 |                   placeholder="Enter activity name"
733 |                   value={customActivity}
734 |                   onChange={(e) => setCustomActivity(e.target.value)}
735 |                   onKeyDown={(e) => {
736 |                     if (e.key === "Enter") {
737 |                       e.preventDefault();
738 |                       handleAddCustomActivity();
739 |                     }
740 |                   }}
741 |                   className="flex-1"
742 |                 />
743 |                 <Button
744 |                   type="button"
745 |                   onClick={handleAddCustomActivity}
746 |                   disabled={!customActivity.trim()}
747 |                   variant="outline"
748 |                 >
749 |                   Add
750 |                 </Button>
751 |                 <Button
752 |                   type="button"
753 |                   onClick={() => {
754 |                     setShowCustomInput(false);
755 |                     setCustomActivity("");
756 |                   }}
757 |                   variant="ghost"
758 |                 >
759 |                   Cancel
760 |                 </Button>
761 |               </div>
762 |             )}
763 |           </div>
764 |
765 |           {/* Display Selected Custom Activities */}
766 |           {selectedActivities.filter(a => !["running", "cycling", "walking"].includes(a)).length
>0 && (
767 |             <div className="flex flex-wrap gap-2 mt-2">
768 |               {selectedActivities
769 |                 .filter(a => !["running", "cycling", "walking"].includes(a))
770 |                 .map((activity) => (
771 |                   <div
772 |                     key={activity}
773 |                     className="flex items-center gap-2 px-3 py-1.5 rounded-full bg-primary/10
border border-primary/20">
774 |                     <span className="text-sm font-medium text-primary">{activity}</span>
775 |                     <button
776 |
```

```
                      type="button"
                      onClick={() => handleRemoveActivity(activity)}
                      className="text-primary hover:text-primary/70 text-sm font-bold"
                    >
                      ×
                    </button>
                  </div>
                ))}
              </div>
              )}
              </div>

              {/* Fitness Level */}
              <div className="space-y-3">
                <Label className="text-lg font-semibold text-foreground">Fitness Level</Label>
                <div className="space-y-3">
                  {fitnessLevelOptions.map((option) => {
                    const isSelected = fitnessLevel === option.value;
                    return (
                      <button
                        key={option.value}
                        onClick={() => setFitnessLevel(option.value)}
                        className={`
                          w-full flex items-start gap-3 p-4 rounded-lg border-2 transition-all duration-300 text-left
                          ${isSelected
                            ? `${option.bgColor} ${option.borderColor} ring-2 ring-offset-2 ring-current`
                            : 'border-border bg-card hover:bg-secondary'
                          }
                        `}
                      >
                        <div className={`w-3 h-3 rounded-full flex-shrink-0 mt-1 ${
                          option.value === "beginner" ? "bg-blue-500" :
                          option.value === "intermediate" ? "bg-green-500" :
                          "bg-purple-500"
                        }`} />
                        <div className="flex-1 min-w-0">
                          <span className={`font-bold text-base block ${isSelected ? option.color : 'text-foreground'}`}>{option.label}
                          </span>
                          <span className="text-xs text-muted-foreground leading-relaxed block mt-1">
                            {option.description}
                          </span>
                        </div>
                      </button>
                    );
                  })}
                </div>
                <p className="text-xs text-muted-foreground">
                  Choose the level that best matches your current fitness and training intensity.
                </p>
              </div>
            </motion.div>
          )}

          {/* Step 3: Preferences */}
          {currentStep === 3 && (
            <motion.div
              key="step3"
              initial={{ opacity: 0, x: 20 }}
              animate={{ opacity: 1, x: 0 }}
              exit={{ opacity: 0, x: -20 }}
              transition={{ duration: 0.3 }}
              className="space-y-6"
            >
              {/* Tip Card */}
              <div className="bg-primary/10 border border-primary/20 rounded-xl p-4 flex items-start gap-3">
                <SettingsIcon className="text-primary flex-shrink-0 mt-0.5" style={{ fontSize: 24 }} />
                <div>
                  <p className="text-sm font-medium text-foreground">
                    Control your privacy - you can change these anytime in settings.
                  </p>
```

```
                </div>
              </div>

              {/* Visibility Settings */}
              <div className="space-y-3">
                <Label className="text-lg font-semibold text-foreground">Who can see you?</Label>
                <div className="flex items-center space-x-2">
                  <Checkbox
                    id="visibleToAll"
                    checked={visibleToAllLevels}
                    onCheckedChange={(checked) => {
                      setVisibleToAllLevels(checked as boolean);
                      if (checked) {
                        setAllowedLevels(["beginner", "intermediate", "pro"]);
                      }
                    }}
                  />
                  <Label htmlFor="visibleToAll" className="text-sm font-normal cursor-pointer">
                    Visible to all fitness levels
                  </Label>
                </div>
                <p className="text-xs text-muted-foreground pl-7">
                  This will help others discover you at venues and connect with workout partners
nearby.
                </p>

                {!visibleToAllLevels && (
                  <div className="space-y-3 pl-6 mt-3">
                    <Label className="text-sm text-muted-foreground block mb-2">Select allowed
levels:</Label>
                    <div className="space-y-2">
                      {fitnessLevelOptions.map((option) => (
                        <div
                          key={option.value}
                          className={`flex items-center space-x-3 p-3 rounded-lg border-2
transition-all cursor-pointer ...  allowedLevels.includes(option.value)
                              ? `${option.bgColor} ${option.borderColor}`
                              : 'border-border/50 bg-card'
                          }`}
                          onClick={() => handleLevelToggle(option.value)}
                        >
                          <Checkbox
                            id={`level-${option.value}`}
                            checked={allowedLevels.includes(option.value)}
                            onCheckedChange={() => handleLevelToggle(option.value)}
                            className="flex-shrink-0"
                          />
                          <div className="flex items-center gap-2 flex-1 min-w-0">
                            <div className={`w-3 h-3 rounded-full flex-shrink-0 ${
                              option.value === "beginner" ? "bg-blue-500" :
                              option.value === "intermediate" ? "bg-green-500" :
                              "bg-purple-500"
                            }`} />
                            <Label
                              htmlFor={`level-${option.value}`}
                              className={`text-sm font-medium cursor-pointer flex-1 ${
                                allowedLevels.includes(option.value) ? option.color : 'text-
foreground'
                              }`}
                            >
                              {option.label}
                            </Label>
                          </div>
                        </div>
                      ))}
                    </div>
                  </div>
                )}
              </div>

              {/* Search Filter - Who do you want to find? */}
              <div className="space-y-2">
                <Label className="text-lg font-semibold text-foreground">Who do you want to find?
</Label>
                <Select value={searchFilter} onValueChange={(value) => setSearchFilter(value as
SearchFilter)}>
```

```
                  <SelectTrigger className="h-12">
                    <SelectValue />
                  </SelectTrigger>
                  <SelectContent>
                    {searchFilterOptions.map((option) => (
                      <SelectItem key={option.value} value={option.value}>
                        {option.label}
                      </SelectItem>
                    ))}
                  </SelectContent>
                </Select>
                <p className="text-xs text-muted-foreground">
                  Filter matches by fitness level. You can change this anytime on the map.
                </p>
              </div>

              {/* Radius Preference */}
              <div className="space-y-2">
                <Label className="text-lg font-semibold text-foreground">Search Radius
Preference</Label>
                <Select value={radiusPreference} onValueChange={(value) =>
setRadiusPreference(value as RadiusPreference)}>
                  <SelectTrigger className="h-12">
                    <SelectValue />
                  </SelectTrigger>
                  <SelectContent>
                    {radiusOptions.map((option) => (
                      <SelectItem key={option.value} value={option.value}>
                        {option.label}
                      </SelectItem>
                    ))}
                  </SelectContent>
                </Select>
                <p className="text-xs text-muted-foreground">
                  Controls how far the app searches for matches
                </p>
              </div>
            </motion.div>
          )}
        </AnimatePresence>

        {/* Navigation Buttons */}
        <div className="flex gap-3 pt-4">
          {currentStep > 1 && (
            <Button
              type="button"
              onClick={handleBack}
              variant="outline"
              className="flex-1 h-14 text-base font-medium"
            >
              <ArrowBackIcon className="mr-2" style={{ fontSize: 20 }} />
              Back
            </Button>
          )}
          {currentStep < 3 ? (
            <Button
              type="button"
              onClick={handleNext}
              className="flex-1 h-14 text-base font-medium bg-primary text-primary-foreground
hover:bg-primary/90"
            >
              Next
            </Button>
          ) : (
            <Button
              onClick={handleComplete}
              disabled={saving || authLoading}
              className="flex-1 h-14 text-base font-medium bg-primary text-primary-foreground
hover:bg-primary/90 shadow...
              {saving ? "Saving..." : "Complete Setup"}
            </Button>
          )}
        </div>
      </motion.div>
```

```
990 |     </div>
991 |   );
992 | };
993 |
994 | export default ProfileSetup;
995 |
```

File: src/pages/ProfileView.tsx

Lines: 406

```
   1 | import { motion } from "framer-motion";
   2 | import { Button } from "@/components/ui/button";
   3 | import Avatar from "@mui/material/Avatar";
   4 | import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
   5 | import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
   6 | import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
   7 | import CloseIcon from "@mui/icons-material/Close";
   8 | import SendIcon from "@mui/icons-material/Send";
   9 | import PersonAddIcon from "@mui/icons-material/PersonAdd";
  10 | import PersonRemoveIcon from "@mui/icons-material/PersonRemove";
  11 | import CheckCircleIcon from "@mui/icons-material/CheckCircle";
  12 | import HourglassEmptyIcon from "@mui/icons-material/HourglassEmpty";
  13 | import TouchAppIcon from "@mui/icons-material/TouchApp";
  14 | import BlockIcon from "@mui/icons-material/Block";
  15 | import ReportIcon from "@mui/icons-material/Report";
  16 | import { Card } from "@/components/ui/card";
  17 | import { useState } from "react";
  18 | import { ReportUserModal } from "@/components/ReportUserModal";
  19 | import {
  20 |   AlertDialog,
  21 |   AlertDialogAction,
  22 |   AlertDialogCancel,
  23 |   AlertDialogContent,
  24 |   AlertDialogDescription,
  25 |   AlertDialogFooter,
  26 |   AlertDialogHeader,
  27 |   AlertDialogTitle,
  28 | } from "@/components/ui/alert-dialog";
  29 |
  30 | type FriendStatus = "not_friends" | "request_pending" | "request_received" | "friends" |
"denied";
  31 |
  32 | interface ProfileViewProps {
  33 |   user: {
  34 |     id: string | number;
  35 |     name: string;
  36 |     distance: string;
  37 |     activity: string;
  38 |     avatar: string;
  39 |     photos?: string[];
  40 |     bio?: string;
  41 |   };
  42 |   friendStatus: FriendStatus;
  43 |   cooldownDays?: number;
  44 |   onClose: () => void;
  45 |   onSendMessage: () => void;
  46 |   onAddFriend: () => void;
  47 |   onAcceptFriend: () => void;
  48 |   onDeclineFriend: () => void;
  49 |   onUnfriend?: () => void;
  50 |   onPoke?: () => void;
  51 |   hasPoked?: boolean;
  52 |   isWorkoutActive?: boolean;
  53 |   onReport?: (reason: string, details?: string) => Promise<void>;
  54 |   onBlock?: () => Promise<void>;
  55 | }
  56 |
  57 | export const ProfileView = ({
  58 |   user,
  59 |   friendStatus,
  60 |   cooldownDays,
  61 |   onClose,
  62 |   onSendMessage,
  63 |   onAddFriend,
  64 |   onAcceptFriend,
  65 |   onDeclineFriend,
  66 |   onUnfriend,
```

```
  67 |   onPoke,
  68 |   hasPoked = false,
  69 |   isWorkoutActive = false,
  70 |   onReport,
  71 |   onBlock,
  72 | }: ProfileViewProps) => {
  73 |   const [selectedPhotoIndex, setSelectedPhotoIndex] = useState(0);
  74 |   const [showUnfriendDialog, setShowUnfriendDialog] = useState(false);
  75 |   const [showReportModal, setShowReportModal] = useState(false);
  76 |   const [showBlockDialog, setShowBlockDialog] = useState(false);
  77 |
  78 |   const displayPhotos = user.photos && user.photos.length > 0
  79 |     ? user.photos
  80 |     : [user.avatar];
  81 |
  82 |   const getActivityIcon = () => {
  83 |     switch (user.activity.toLowerCase()) {
  84 |       case "running":
  85 |         return <DirectionsRunIcon className="text-success" style={{ fontSize: 20 }} />;
  86 |       case "cycling":
  87 |         return <DirectionsBikeIcon className="text-primary" style={{ fontSize: 20 }} />;
  88 |       case "walking":
  89 |         return <DirectionsWalkIcon className="text-warning" style={{ fontSize: 20 }} />;
  90 |       default:
  91 |         return null;
  92 |     }
  93 |   };
  94 |
  95 |   return (
  96 |     <motion.div
  97 |       initial={{ opacity: 0, scale: 0.95 }}
  98 |       animate={{ opacity: 1, scale: 1 }}
  99 |       exit={{ opacity: 0, scale: 0.95 }}
 100 |       transition={{ duration: 0.2 }}
 101 |       className="fixed inset-0 z-50 flex items-center justify-center p-4 bg-background/80
backdrop-blur-sm"
 102 |       onClick={onClose}
 103 |     >
 104 |       <motion.div
 105 |         initial={{ y: 50 }}
 106 |         animate={{ y: 0 }}
 107 |         exit={{ y: 50 }}
 108 |         transition={{ type: "spring", stiffness: 300, damping: 30 }}
 109 |         className="w-full max-w-md"
 110 |         onClick={(e) => e.stopPropagation()}
 111 |       >
 112 |         <Card className="overflow-hidden shadow-elevation-4 border-2 border-border/50 relative">
 113 |           {/* Photo Gallery */}
 114 |           <div className="relative w-full aspect-square bg-muted">
 115 |             {/* Close button */}
 116 |             <button
 117 |               onClick={onClose}
 118 |               className="absolute top-4 right-4 z-10 p-2 bg-red-500 hover:bg-red-600 text-white
rounded-full transition-...
 119 |             >
 120 |               <CloseIcon fontSize="small" />
 121 |             </button>
 122 |             <motion.img
 123 |               key={selectedPhotoIndex}
 124 |               initial={{ opacity: 0 }}
 125 |               animate={{ opacity: 1 }}
 126 |               transition={{ duration: 0.3 }}
 127 |               src={displayPhotos[selectedPhotoIndex]}
 128 |               alt={user.name}
 129 |               className="w-full h-full object-cover"
 130 |             />
 131 |
 132 |             {/* Photo indicators */}
 133 |             {displayPhotos.length > 1 && (
 134 |               <div className="absolute bottom-4 left-1/2 -translate-x-1/2 flex gap-2">
 135 |                 {displayPhotos.map((_, index) => (
 136 |                   <button
 137 |                     key={index}
```

```
                    onClick={() => setSelectedPhotoIndex(index)}
                    className={`w-2 h-2 rounded-full transition-all ${
                      index === selectedPhotoIndex
                        ? "bg-white w-6"
                        : "bg-white/50"
                    }`}
                  />
                ))}
              </div>
            )}

            {/* Navigation arrows for photos */}
            {displayPhotos.length > 1 && (
              <>
                <button
                  onClick={() => setSelectedPhotoIndex((prev) =>
                    prev === 0 ? displayPhotos.length - 1 : prev - 1
                  )}
                  className="absolute left-2 top-1/2 -translate-y-1/2 p-2 bg-background/80 backdrop-blur-sm rounded-full...
                >
                  !•
                </button>
                <button
                  onClick={() => setSelectedPhotoIndex((prev) =>
                    (prev + 1) % displayPhotos.length
                  )}
                  className="absolute right-2 top-1/2 -translate-y-1/2 p-2 bg-background/80 backdrop-blur-sm rounded-ful...
                >
                  !′
                </button>
              </>
            )}
          </div>

          {/* User Info */}
          <div className="p-6 space-y-4">
            <div className="flex items-start gap-4">
              <Avatar
                src={user.avatar}
                alt={user.name}
                sx={{ width: 72, height: 72, border: "3px solid hsl(var(--primary))" }}
              />
              <div className="flex-1">
                <h2 className="text-2xl font-bold">{user.name}</h2>
                <div className="flex items-center gap-2 mt-1">
                  {getActivityIcon()}
                  <span className="text-sm text-muted-foreground">{user.activity}</span>
                </div>
                <p className="text-sm text-muted-foreground mt-1">
                  Ø=ÜÍ {user.distance} away
                </p>
              </div>
            </div>

            {/* Bio */}
            {user.bio && (
              <div className="py-3 border-t border-border">
                <p className="text-sm text-muted-foreground leading-relaxed">{user.bio}</p>
              </div>
            )}

            {/* Friend Status Badge */}
            {friendStatus === "friends" && (
              <div className="flex items-center justify-center gap-2 px-4 py-2 bg-success/10 border-2 border-success rou...
                <CheckCircleIcon className="text-success" style={{ fontSize: 20 }} />
                <span className="text-sm font-semibold text-success">Friends</span>
              </div>
            )}

            {friendStatus === "request_pending" && (
              <div className="flex items-center justify-center gap-2 px-4 py-2 bg-warning/10 border-2 border-warning rou...
```

```
                <HourglassEmptyIcon className="text-warning" style={{ fontSize: 20 }} />
                <span className="text-sm font-semibold text-warning">Friend Request Sent</span>
              </div>
            )}

            {/* Action Buttons */}
            <div className="flex flex-col gap-3 pt-2">
              {friendStatus === "request_received" && (
                <div className="flex gap-2">
                  <Button
                    onClick={onAcceptFriend}
                    className="flex-1 h-12 bg-success hover:bg-success/90 text-success-
foreground"
                  >
                    <CheckCircleIcon className="mr-2" style={{ fontSize: 20 }} />
                    Accept
                  </Button>
                  <Button
                    onClick={onDeclineFriend}
                    variant="outline"
                    className="flex-1 h-12"
                  >
                    Decline
                  </Button>
                </div>
              )}

              {/* Poke Button - Show when not friends and workout is active */}
              {onPoke && friendStatus === "not_friends" && !hasPoked && isWorkoutActive && (
                <Button
                  onClick={onPoke}
                  className="w-full h-12 font-semibold bg-purple-500 hover:bg-purple-600 text-
white"
                >
                  <TouchAppIcon className="mr-2" style={{ fontSize: 20 }} />
                  Poke {user.name}
                </Button>
              )}

              {/* Show disabled poke button when workout is not active */}
              {onPoke && friendStatus === "not_friends" && !hasPoked && !isWorkoutActive && (
                <Button
                  disabled
                  variant="outline"
                  className="w-full h-12 opacity-50"
                  title="You must have an active workout session to poke someone"
                >
                  <TouchAppIcon className="mr-2" style={{ fontSize: 20 }} />
                  Poke {user.name} (Start workout first)
                </Button>
              )}

              {hasPoked && (
                <Button
                  disabled
                  variant="outline"
                  className="w-full h-12 opacity-50"
                >
                  <TouchAppIcon className="mr-2" style={{ fontSize: 20 }} />
                  Poke Sent
                </Button>
              )}

              {(friendStatus === "not_friends" || friendStatus === "denied") && (
                <Button
                  onClick={onAddFriend}
                  disabled={friendStatus === "denied"}
                  variant="outline"
                  className="w-full h-12"
                >
                  {friendStatus === "denied" ? (
                    <>
                      <HourglassEmptyIcon className="mr-2" style={{ fontSize: 20 }} />
```

```
                      Try again in {cooldownDays} day{cooldownDays !== 1 ? "s" : ""}
                    </>
                  ) : (
                    <>
                      <PersonAddIcon className="mr-2" style={{ fontSize: 20 }} />
                      Add Friend
                    </>
                  )}
                </Button>
              )}

              <Button
                onClick={onSendMessage}
                className="w-full h-12 font-semibold"
              >
                <SendIcon className="mr-2" style={{ fontSize: 20 }} />
                Send Message
              </Button>

              {/* Unfriend Button - Only show when friends */}
              {friendStatus === "friends" && onUnfriend && (
                <Button
                  onClick={() => setShowUnfriendDialog(true)}
                  variant="outline"
                  className="w-full h-12 text-destructive hover:text-destructive hover:bg-
destructive/10 border-destruct...
                >
                  <PersonRemoveIcon className="mr-2" style={{ fontSize: 20 }} />
                  Unfriend
                </Button>
              )}

              {/* Report and Block Buttons */}
              <div className="flex gap-2 pt-2 border-t border-border">
                {onReport && (
                  <Button
                    onClick={() => setShowReportModal(true)}
                    variant="outline"
                    className="flex-1 h-11"
                  >
                    <ReportIcon className="mr-2" style={{ fontSize: 18 }} />
                    Report
                  </Button>
                )}
                {onBlock && (
                  <Button
                    onClick={() => setShowBlockDialog(true)}
                    variant="outline"
                    className="flex-1 h-11 text-destructive hover:text-destructive hover:bg-
destructive/10 border-destru...
                  >
                    <BlockIcon className="mr-2" style={{ fontSize: 18 }} />
                    Block
                  </Button>
                )}
              </div>
            </div>
          </div>
        </Card>
      </motion.div>

      {/* Unfriend Confirmation Dialog */}
      <AlertDialog open={showUnfriendDialog} onOpenChange={setShowUnfriendDialog}>
        <AlertDialogContent>
          <AlertDialogHeader>
            <AlertDialogTitle>Unfriend {user.name}?</AlertDialogTitle>
            <AlertDialogDescription>
              Are you sure you want to unfriend {user.name}?
              They will be removed from your friends list and won't be able to see your location
during workouts.
            </AlertDialogDescription>
          </AlertDialogHeader>
          <AlertDialogFooter>
            <AlertDialogCancel>Cancel</AlertDialogCancel>
```

```
            <AlertDialogAction
              onClick={() => {
                if (onUnfriend) {
                  onUnfriend();
                }
                setShowUnfriendDialog(false);
              }}
              className="bg-destructive text-destructive-foreground hover:bg-destructive/90"
            >
              Unfriend
            </AlertDialogAction>
          </AlertDialogFooter>
        </AlertDialogContent>
      </AlertDialog>

      {/* Report User Modal */}
      {onReport && (
        <ReportUserModal
          open={showReportModal}
          onOpenChange={setShowReportModal}
          userName={user.name}
          onReport={onReport}
        />
      )}

      {/* Block User Confirmation Dialog */}
      {onBlock && (
        <AlertDialog open={showBlockDialog} onOpenChange={setShowBlockDialog}>
          <AlertDialogContent>
            <AlertDialogHeader>
              <AlertDialogTitle>Block {user.name}?</AlertDialogTitle>
              <AlertDialogDescription>
                This will prevent {user.name} from sending you messages and you won't see them on the map. You can unblo
              </AlertDialogDescription>
            </AlertDialogHeader>
            <AlertDialogFooter>
              <AlertDialogCancel>Cancel</AlertDialogCancel>
              <AlertDialogAction
                onClick={async () => {
                  if (onBlock) {
                    await onBlock();
                  }
                  setShowBlockDialog(false);
                }}
                className="bg-red-600 hover:bg-red-700"
              >
                Block
              </AlertDialogAction>
            </AlertDialogFooter>
          </AlertDialogContent>
        </AlertDialog>
      )}
    </motion.div>
  );
};

```

File: src/pages/Settings.tsx

Lines: 539

```
import { useState, useEffect } from "react";
import { motion } from "framer-motion";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Checkbox } from "@/components/ui/checkbox";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { useNavigate } from "react-router-dom";
import { FitnessLevel, RadiusPreference, VisibilitySettings } from "@/contexts/UserContext";
import { SearchFilter } from "@/services/matchingService";
import LogoutIcon from "@mui/icons-material/Logout";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import HistoryIcon from "@mui/icons-material/History";
import ChevronRightIcon from "@mui/icons-material/ChevronRight";
import PeopleIcon from "@mui/icons-material/People";
import { toast } from "sonner";
import { Card } from "@/components/ui/card";
import BottomNavigation from "@/components/BottomNavigation";
import { useAuth } from "@/hooks/useAuth";
import { updateUserProfile, signOut, getUserData } from "@/services/authService";

const Settings = () => {
  const navigate = useNavigate();
  const { user, loading: authLoading } = useAuth();
  const [activity, setActivity] = useState<string | null>(null);
  const [fitnessLevel, setFitnessLevel] = useState<FitnessLevel>("intermediate");
  const [pace, setPace] = useState("");
  const [visibleToAllLevels, setVisibleToAllLevels] = useState(true);
  const [allowedLevels, setAllowedLevels] = useState<FitnessLevel[]>(["beginner", "intermediate", "pro"]);
  const [searchFilter, setSearchFilter] = useState<SearchFilter>("all");
  const [radiusPreference, setRadiusPreference] = useState<RadiusPreference>("normal");
  const [saving, setSaving] = useState(false);
  const [isEditingMatching, setIsEditingMatching] = useState(false);


  // Redirect if not authenticated
  useEffect(() => {
    if (!authLoading && !user) {
      navigate("/");
    }
  }, [user, authLoading, navigate]);

  // Load user data
  useEffect(() => {
    const loadUserData = async () => {
      if (user) {
        const userData = await getUserData(user.uid);
        if (userData) {
          setActivity(userData.activity || null);
          setFitnessLevel(userData.fitnessLevel || "intermediate");
          setPace(userData.pace ? userData.pace.toString() : "");
          if (userData.visibility) {
            setVisibleToAllLevels(userData.visibility.visibleToAllLevels ?? true);
            setAllowedLevels(userData.visibility.allowedLevels || ["beginner", "intermediate", "pro"]);
          }
          setSearchFilter(userData.searchFilter || "all");
          setRadiusPreference(userData.radiusPreference || "normal");
        }
      }
    };
    loadUserData();
  }, [user]);

  const handleSignOut = async () => {
```

```
    try {
      await signOut();
      toast.success("Signed out successfully");
      navigate("/");
    } catch (error: any) {
      console.error("Error signing out:", error);
      toast.error("Failed to sign out. Please try again.");
    }
  };

  const handleSaveMatching = async () => {
    if (!user) return;

    setSaving(true);
    try {
      const paceValue = pace ? parseFloat(pace) : null;
      const visibility: VisibilitySettings = {
        visibleToAllLevels: visibleToAllLevels,
        allowedLevels: visibleToAllLevels ? ["beginner", "intermediate", "pro"] : allowedLevels
      };

      await updateUserProfile(user.uid, {
        fitnessLevel: fitnessLevel,
        pace: paceValue,
        visibility: visibility,
        searchFilter: searchFilter,
        radiusPreference: radiusPreference
      });
      setIsEditingMatching(false);
      toast.success("Matching preferences updated successfully!");
    } catch (error: any) {
      console.error("Error saving matching preferences:", error);
      toast.error("Failed to save preferences. Please try again.");
    } finally {
      setSaving(false);
    }
  };

  const handleLevelToggle = (level: FitnessLevel) => {
    if (allowedLevels.includes(level)) {
      setAllowedLevels(allowedLevels.filter(l => l !== level));
    } else {
      setAllowedLevels([...allowedLevels, level]);
    }
  };

  const fitnessLevelOptions: { value: FitnessLevel; label: string; description: string; color:
string; bgColor: string; ...
    {
      value: "beginner",
      label: "Beginner",
      description: "Just starting out or getting back into fitness. You're building a foundation
and learning the basics.",
      color: "text-blue-600 dark:text-blue-400",
      bgColor: "bg-blue-50 dark:bg-blue-950/30",
      borderColor: "border-blue-200 dark:border-blue-800"
    },
    {
      value: "intermediate",
      label: "Intermediate",
      description: "Regular exerciser with a consistent routine. You can handle moderate
workouts and are comfortable wi...
      color: "text-green-600 dark:text-green-400",
      bgColor: "bg-green-50 dark:bg-green-950/30",
      borderColor: "border-green-200 dark:border-green-800"
    },
    {
      value: "pro",
      label: "Pro",
      description: "Advanced athlete with high performance goals. You train regularly at high
intensity and push your li...
      color: "text-purple-600 dark:text-purple-400",
      bgColor: "bg-purple-50 dark:bg-purple-950/30",
      borderColor: "border-purple-200 dark:border-purple-800"
    }
```

```
  ];

  const radiusOptions: { value: RadiusPreference; label: string }[] = [
    { value: "nearby", label: "Nearby" },
    { value: "normal", label: "Normal" },
    { value: "wide", label: "Wide" }
  ];

  const searchFilterOptions: { value: SearchFilter; label: string }[] = [
    { value: "all", label: "All Levels" },
    { value: "beginner", label: "Beginner" },
    { value: "intermediate", label: "Intermediate" },
    { value: "pro", label: "Pro" }
  ];

  const getPaceUnit = () => {
    if (activity === "cycling") return "km/h";
    return "min/km";
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-primary/10 via-background to-success/10 pb-20">
      {/* Header */}
      <motion.div
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-card/80 backdrop-blur-md shadow-elevation-2 sticky top-0 z-10 border-b border-border/50"
      >
        <div className="max-w-2xl mx-auto px-6 py-5">
          <h1 className="text-3xl font-bold">Settings</h1>
        </div>
      </motion.div>

      {/* Content */}
      <div className="max-w-2xl mx-auto p-6 space-y-6 pb-10">
        {/* Quick Links */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.4, delay: 0.1 }}
        >
          <Card className="p-4 shadow-elevation-2 bg-card/50 backdrop-blur-sm space-y-2">
            <button
              onClick={() => navigate("/edit-profile")}
              className="w-full flex items-center justify-between p-4 rounded-xl hover:bg-accent transition-all duration...
              <div className="flex items-center gap-3">
                <div className="p-3 rounded-xl bg-primary/10">
                  <DirectionsRunIcon className="text-primary" style={{ fontSize: 24 }} />
                </div>
                <div className="text-left">
                  <h3 className="font-bold">Edit Profile</h3>
                  <p className="text-xs text-muted-foreground">Update your profile information</p>
                </div>
              </div>
              <ChevronRightIcon className="text-muted-foreground" />
            </button>
            <button
              onClick={() => navigate("/workout-history")}
              className="w-full flex items-center justify-between p-4 rounded-xl hover:bg-accent transition-all duration...
              <div className="flex items-center gap-3">
                <div className="p-3 rounded-xl bg-success/10">
                  <HistoryIcon className="text-success" style={{ fontSize: 24 }} />
                </div>
                <div className="text-left">
                  <h3 className="font-bold">Workout History</h3>
                  <p className="text-xs text-muted-foreground">View all your past workouts</p>
                </div>
              </div>
              <ChevronRightIcon className="text-muted-foreground" />
```

```
            </button>
            <button
              onClick={() => navigate("/friends")}
              className="w-full flex items-center justify-between p-4 rounded-xl hover:bg-accent transition-all duration...
              >
                <div className="flex items-center gap-3">
                  <div className="p-3 rounded-xl bg-warning/10">
                    <PeopleIcon className="text-warning" style={{ fontSize: 24 }} />
                  </div>
                  <div className="text-left">
                    <h3 className="font-bold">Friends</h3>
                    <p className="text-xs text-muted-foreground">Manage your connections</p>
                  </div>
                </div>
                <ChevronRightIcon className="text-muted-foreground" />
              </button>
            </Card>
          </motion.div>

          {/* Matching Preferences Section */}
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ duration: 0.4, delay: 0.25 }}
          >
            <Card className="p-6 space-y-6 shadow-elevation-2 bg-card/50 backdrop-blur-sm">
              <div className="flex items-center justify-between">
                <h2 className="text-2xl font-bold">Matching Preferences</h2>
                {!isEditingMatching && (
                  <Button
                    variant="outline"
                    onClick={() => setIsEditingMatching(true)}
                    className="h-10"
                  >
                    Edit Preferences
                  </Button>
                )}
              </div>

              {!isEditingMatching ? (
                // Display Mode
                <div className="space-y-4">
                  <div className="space-y-2">
                    <Label className="text-sm text-muted-foreground">Fitness Level</Label>
                    <div className="flex items-center gap-3">
                      <div className={`w-4 h-4 rounded-full ${
                        fitnessLevel === "beginner" ? "bg-blue-500" :
                        fitnessLevel === "intermediate" ? "bg-green-500" :
                        "bg-purple-500"
                      }`} />
                      <p className={`text-lg font-semibold capitalize ${
                        fitnessLevel === "beginner" ? "text-blue-600 dark:text-blue-400" :
                        fitnessLevel === "intermediate" ? "text-green-600 dark:text-green-400" :
                        "text-purple-600 dark:text-purple-400"
                      }`}>
                        {fitnessLevel}
                      </p>
                    </div>
                    <p className="text-xs text-muted-foreground mt-1">
                      {fitnessLevelOptions.find(opt => opt.value === fitnessLevel)?.description}
                    </p>
                  </div>

                  <div className="space-y-2">
                    <Label className="text-sm text-muted-foreground">Average Pace</Label>
                    <p className="text-lg font-semibold">
                      {pace ? `${pace} ${getPaceUnit()}` : "Not set"}
                    </p>
                  </div>

                  <div className="space-y-2">
```

```
                  <Label className="text-sm text-muted-foreground">Visibility</Label>
                  <p className="text-lg font-semibold">
                    {visibleToAllLevels ? "Visible to all levels" : `Visible to: ${allowedLevels.join(", ")}`}
                  </p>
                </div>

                <div className="space-y-2">
                  <Label className="text-sm text-muted-foreground">Search Filter</Label>
                  <p className="text-lg font-semibold capitalize">
                    {searchFilter === "all" ? "All Levels" : searchFilter}
                  </p>
                </div>

                <div className="space-y-2">
                  <Label className="text-sm text-muted-foreground">Search Radius</Label>
                  <p className="text-lg font-semibold capitalize">{radiusPreference}</p>
                </div>
              </div>
            ) : (
              // Edit Mode
              <div className="space-y-6">
                <div className="space-y-3">
                  <Label>Fitness Level</Label>
                  <div className="space-y-3">
                    {fitnessLevelOptions.map((option) => {
                      const isSelected = fitnessLevel === option.value;
                      return (
                        <button
                          key={option.value}
                          onClick={() => setFitnessLevel(option.value)}
                          className={`
                            w-full flex items-start gap-3 p-4 rounded-lg border-2 transition-all duration-300 text-left ring-current`
                            ${isSelected
                              ? `${option.bgColor} ${option.borderColor} ring-2 ring-offset-2 ring-offset-background`
                              : 'border-border bg-card hover:bg-secondary'
                            }
                          `}
                        >
                          <div className={`w-3 h-3 rounded-full flex-shrink-0 mt-1 ${
                            option.value === "beginner" ? "bg-blue-500" :
                            option.value === "intermediate" ? "bg-green-500" :
                            "bg-purple-500"
                          }`} />
                          <div className="flex-1 min-w-0">
                            <span className={`font-bold text-base block ${isSelected ? option.color : 'text-foreground'}`}>{option.label}</span>
                            <span className="text-xs text-muted-foreground leading-relaxed block mt-1">
                              {option.description}
                            </span>
                          </div>
                        </button>
                      );
                    })}
                  </div>
                  <p className="text-xs text-muted-foreground">
                    Choose the level that best matches your current fitness and training intensity.
                  </p>
                </div>

                <div className="space-y-2">
                  <Label htmlFor="pace">
                    Average Pace ({getPaceUnit()}) <span className="text-xs text-muted-foreground">(Optional)</span>
                  </Label>
                  <Input
                    id="pace"
                    type="number"
                    step="0.1"
                    placeholder={`e.g., ${activity === "cycling" ? "25" : "5.5"}`}
                    value={pace}
                    onChange={(e) => setPace(e.target.value)}
```

```
                    className="h-12"
                  />
                </div>

                <div className="space-y-3">
                  <Label>Who can see you?</Label>
                  <div className="flex items-center space-x-2">
                    <Checkbox
                      id="visibleToAll"
                      checked={visibleToAllLevels}
                      onCheckedChange={(checked) => {
                        setVisibleToAllLevels(checked as boolean);
                        if (checked) {
                          setAllowedLevels(["beginner", "intermediate", "pro"]);
                        }
                      }}
                    />
                    <Label htmlFor="visibleToAll" className="text-sm font-normal cursor-pointer">
                      Visible to all fitness levels
                    </Label>
                  </div>

                  {!visibleToAllLevels && (
                    <div className="space-y-2 pl-6">
                      <Label className="text-sm text-muted-foreground">Select allowed levels:</
Label>
                      {fitnessLevelOptions.map((option) => (
                        <div
                          key={option.value}
                          className={`flex items-center space-x-3 p-2 rounded-lg border
transition-all ${
                            allowedLevels.includes(option.value)
                              ? `${option.bgColor} ${option.borderColor} border-2`
                              : 'border-border/50'
                          }`}
                        >
                          <Checkbox
                            id={`level-${option.value}`}
                            checked={allowedLevels.includes(option.value)}
                            onCheckedChange={() => handleLevelToggle(option.value)}
                          />
                          <div className="flex items-center gap-2 flex-1">
                            <div className={`w-2.5 h-2.5 rounded-full ${
                              option.value === "beginner" ? "bg-blue-500" :
                              option.value === "intermediate" ? "bg-green-500" :
                              "bg-purple-500"
                            }`} />
                            <Label
                              htmlFor={`level-${option.value}`}
                              className={`text-sm font-medium cursor-pointer ${
                                allowedLevels.includes(option.value) ? option.color : ''
                              }`}
                            >
                              {option.label}
                            </Label>
                          </div>
                        </div>
                      ))}
                    </div>
                  )}
                </div>

                <div className="space-y-2">
                  <Label>Who do you want to find?</Label>
                  <Select value={searchFilter} onValueChange={(value) => setSearchFilter(value
as SearchFilter)}>
                    <SelectTrigger className="h-12">
                      <SelectValue />
                    </SelectTrigger>
                    <SelectContent>
                      {searchFilterOptions.map((option) => (
                        <SelectItem key={option.value} value={option.value}>
                          {option.label}
                        </SelectItem>
```

```
                      ))}
                    </SelectContent>
                  </Select>
                  <p className="text-xs text-muted-foreground">
                    Filter matches by fitness level. You can change this anytime on the map.
                  </p>
                </div>

                <div className="space-y-2">
                  <Label>Search Radius Preference</Label>
                  <Select value={radiusPreference} onValueChange={(value) =>
setRadiusPreference(value as RadiusPreferen
                    <SelectTrigger className="h-12">
                      <SelectValue />
                    </SelectTrigger>
                    <SelectContent>
                      {radiusOptions.map((option) => (
                        <SelectItem key={option.value} value={option.value}>
                          {option.label}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                  <p className="text-xs text-muted-foreground">
                    Controls how far the app searches for matches
                  </p>
                </div>

                {/* Action Buttons */}
                <div className="flex gap-3 pt-2">
                  <Button
                    variant="outline"
                    onClick={() => {
                      setIsEditingMatching(false);
                      // Reset values
                      if (user) {
                        getUserData(user.uid).then(userData => {
                          if (userData) {
                            setFitnessLevel(userData.fitnessLevel || "intermediate");
                            setPace(userData.pace ? userData.pace.toString() : "");
                            if (userData.visibility) {
                              setVisibleToAllLevels(userData.visibility.visibleToAllLevels ??
true);
                              setAllowedLevels(userData.visibility.allowedLevels || ["beginner",
"intermediate", "pro"])...
                            }
                            setSearchFilter(userData.searchFilter || "all");
                            setRadiusPreference(userData.radiusPreference || "normal");
                          }
                        });
                      }
                    }}
                    className="flex-1 h-12"
                  >
                    Cancel
                  </Button>
                  <Button
                    onClick={handleSaveMatching}
                    disabled={saving || authLoading}
                    className="flex-1 h-12 font-semibold"
                  >
                    {saving ? "Saving..." : "Save Changes"}
                  </Button>
                </div>
              </div>
            )}
          </Card>
        </motion.div>

        {/* Privacy & Data Section */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.4, delay: 0.25 }}
```

```
          >
            <Card className="p-6 space-y-4 shadow-elevation-2 bg-card/50 backdrop-blur-sm">
              <h2 className="text-2xl font-bold">Privacy & Data</h2>

              <div className="space-y-3">
                <div className="flex items-center justify-between p-3 hover:bg-secondary rounded-lg transition-colors cursor
                  <p className="font-medium text-foreground">Data Download</p>
                  <Button variant="ghost" size="sm" className="text-primary hover:text-primary">Request</Button>
                </div>
                <div className="flex items-center justify-between p-3 hover:bg-secondary rounded-lg transition-colors cursor
                  <p className="font-medium text-foreground">Delete Account</p>
                  <Button variant="ghost" size="sm" className="text-destructive hover:text-destructive">Delete</Button>
                </div>
              </div>

              <div className="pt-2 px-1">
                <p className="text-xs text-muted-foreground leading-relaxed">
                  Your privacy is important. You can download your data or delete your account at anytime.
                </p>
              </div>
            </Card>
          </motion.div>

          {/* Sign Out */}
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ duration: 0.4, delay: 0.3 }}
          >
            <Button
              onClick={handleSignOut}
              variant="outline"
              className="w-full h-14 text-base font-semibold border-2 hover:bg-destructive/10 hover:border-destructive/30 ...
            >
              <LogoutIcon className="mr-2" style={{ fontSize: 24 }} />
              Sign Out
            </Button>
          </motion.div>
        </div>

        <BottomNavigation />
      </div>
    );
};

export default Settings;
```

File: src/pages/WorkoutHistory.tsx

Lines: 230

```
import { useState, useMemo, useEffect } from "react";
import { motion } from "framer-motion";
import { useNavigate, useLocation } from "react-router-dom";
import { useUser, type WorkoutHistory } from "@/contexts/UserContext";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { WorkoutDetailModal } from "@/components/WorkoutDetailModal";
import ArrowBackIcon from "@mui/icons-material/ArrowBack";
import DirectionsRunIcon from "@mui/icons-material/DirectionsRun";
import DirectionsBikeIcon from "@mui/icons-material/DirectionsBike";
import DirectionsWalkIcon from "@mui/icons-material/DirectionsWalk";
import CalendarTodayIcon from "@mui/icons-material/CalendarToday";
import PeopleIcon from "@mui/icons-material/People";
import { format } from "date-fns";
import BottomNavigation from "@/components/BottomNavigation";
import { generateDummyWorkoutHistory, ENABLE_DUMMY_DATA } from "@/lib/dummyData";

const WorkoutHistoryPage = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { workoutHistory, useMetric } = useUser();
  const [selectedWorkout, setSelectedWorkout] = useState<WorkoutHistory | null>(null);

  // Sort workouts by date (most recent first)
  const sortedWorkouts = useMemo(() => {
    // Add dummy workout history if enabled and no real workouts exist
    let workouts = [...workoutHistory];
    if (ENABLE_DUMMY_DATA && workouts.length === 0) {
      workouts = generateDummyWorkoutHistory();
    }
    return workouts.sort((a, b) => new Date(b.date).getTime() - new Date(a.date).getTime());
  }, [workoutHistory]);

  // Auto-open workout detail modal if workoutId is provided in navigation state
  useEffect(() => {
    const workoutId = (location.state as { workoutId?: string })?.workoutId;
    if (workoutId && sortedWorkouts.length > 0) {
      const workout = sortedWorkouts.find(w => w.id === workoutId);
      if (workout) {
        setSelectedWorkout(workout);
        // Clear the state to prevent reopening on subsequent navigations
        navigate(location.pathname, { replace: true, state: {} });
      }
    }
  }, [location.state, sortedWorkouts, navigate, location.pathname]);


  const formatTime = (seconds: number) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    if (hours > 0) {
      return `${hours}h ${minutes}m`;
    }
    return `${minutes}m`;
  };

  const convertDistance = (km: number) => {
    if (useMetric) return { value: km.toFixed(2), unit: "km" };
    return { value: (km * 0.621371).toFixed(2), unit: "mi" };
  };

  const convertSpeed = (kmh: number) => {
    if (useMetric) return { value: kmh.toFixed(1), unit: "km/h" };
    return { value: (kmh * 0.621371).toFixed(1), unit: "mph" };
  };

```

```
  67 |   const getActivityConfig = (activity: "running" | "cycling" | "walking") => {
  68 |     const configs = {
  69 |       running: { icon: DirectionsRunIcon, color: "success", label: "Running", bgClass: "bg-
success/10", textClass: "text...
  70 |       cycling: { icon: DirectionsBikeIcon, color: "primary", label: "Cycling", bgClass: "bg-
primary/10", textClass: "tex...
  71 |       walking: { icon: DirectionsWalkIcon, color: "warning", label: "Walking", bgClass: "bg-
warning/10", textClass: "tex...
  72 |     };
  73 |     return configs[activity];
  74 |   };
  75 |
  76 |   return (
  77 |     <div className="min-h-screen bg-background pb-20">
  78 |       {/* Header */}
  79 |       <div className="sticky top-0 z-10 bg-background/95 backdrop-blur-sm border-b border-
border">
  80 |         <div className="container mx-auto px-4 py-4">
  81 |           <div className="flex items-center gap-4">
  82 |             <Button
  83 |               variant="ghost"
  84 |               size="icon"
  85 |               onClick={() => navigate(-1)}
  86 |               className="rounded-full"
  87 |             >
  88 |               <ArrowBackIcon />
  89 |             </Button>
  90 |             <div className="flex-1">
  91 |               <h1 className="text-2xl font-bold">Workout History</h1>
  92 |               <p className="text-sm text-muted-foreground">
  93 |                 {sortedWorkouts.length} workout{sortedWorkouts.length !== 1 ? "s" : ""} recorded
  94 |               </p>
  95 |             </div>
  96 |           </div>
  97 |         </div>
  98 |       </div>
  99 |
 100 |       <div className="container mx-auto px-4 py-6 space-y-6">
 101 |         {/* Workouts List */}
 102 |         <Card>
 103 |           <CardHeader>
 104 |             <CardTitle className="flex items-center gap-2">
 105 |               <CalendarTodayIcon />
 106 |               Recent Workouts
 107 |             </CardTitle>
 108 |           </CardHeader>
 109 |           <CardContent>
 110 |             {sortedWorkouts.length === 0 ? (
 111 |               <div className="text-center py-12">
 112 |                 <div className="text-6xl mb-4">Ø<ßÃ</div>
 113 |                 <h3 className="text-lg font-semibold mb-2">No workouts yet</h3>
 114 |                 <p className="text-sm text-muted-foreground mb-6">
 115 |                   Start tracking your activities to see them here
 116 |                 </p>
 117 |                 <Button onClick={() => navigate("/map")}>Start a Workout</Button>
 118 |               </div>
 119 |             ) : (
 120 |               <div className="space-y-3">
 121 |                   {sortedWorkouts.map((workout, index) => {
 122 |                   const config = getActivityConfig(workout.activity);
 123 |                   const Icon = config.icon;
 124 |
 125 |                   return (
 126 |                     <motion.div
 127 |                       key={workout.id}
 128 |                       initial={{ opacity: 0, y: 10 }}
 129 |                       animate={{ opacity: 1, y: 0 }}
 130 |                       transition={{ delay: index * 0.05 }}
 131 |                     >
 132 |                       <Card
 133 |                         className="hover:shadow-elevation-2 transition-shadow cursor-pointer"
 134 |                         onClick={() => setSelectedWorkout(workout)}
 135 |                       >
 136 |                         <CardContent className="p-4">
 137 |                           <div className="flex items-start gap-4">
```

```
                            {/* Activity Icon or Photo Thumbnail */}
                            {(workout as any).photos && (workout as any).photos.length > 0 ? (
                              <div className="w-16 h-16 rounded-xl overflow-hidden flex-
shrink-0">
                                <img
                                  src={(workout as any).photos[0]}
                                  alt="Workout"
                                  className="w-full h-full object-cover"
                                />
                              </div>
                            ) : (
                              <div className={`p-3 rounded-xl ${config.bgClass} flex-shrink-0`}>
                                <Icon className={config.textClass} style={{ fontSize: 28 }} />
                              </div>
                            )}

                            {/* Workout Details */}
                            <div className="flex-1 min-w-0">
                              <div className="flex items-center justify-between mb-2">
                                <h3 className="font-bold text-lg">{config.label}</h3>
                                <span className="text-xs text-muted-foreground">
                                  {format(new Date(workout.date), "MMM d, yyyy")}
                                </span>
                              </div>

                              {/* Stats Grid */}
                              <div className="grid grid-cols-2 sm:grid-cols-4 gap-3 mb-3">
                                <div>
                                  <div className="text-xs text-muted-foreground">Distance</div>
                                  <div className="font-semibold">
                                    {convertDistance(workout.distance).value}{" "}
                                    <span className="text-
xs">{convertDistance(workout.distance).unit}</span>
                                  </div>
                                </div>
                                <div>
                                  <div className="text-xs text-muted-foreground">Duration</div>
                                  <div className="font-semibold">{formatTime(workout.duration)}</
div>
                                </div>
                                <div>
                                  <div className="text-xs text-muted-foreground">Avg Speed</div>
                                  <div className="font-semibold">
                                    {convertSpeed(workout.avgSpeed).value}{" "}
                                    <span className="text-
xs">{convertSpeed(workout.avgSpeed).unit}</span>
                                  </div>
                                </div>

                                {/* Nearby Users Indicator */}
                                {workout.nearbyUsers && workout.nearbyUsers.length > 0 && (
                                  <div className="flex items-center gap-2 text-sm text-muted-
foreground">
                                    <PeopleIcon style={{ fontSize: 16 }} />
                                    <span>
                                      {workout.nearbyUsers.length} {workout.nearbyUsers.length ===
1 ? "person" : "people"...
                                    </span>
                                  </div>
                                )}

                                {/* Caption */}
                                {(workout as any).caption && (
                                  <p className="text-sm text-muted-foreground italic mt-2">
                                    "{(workout as any).caption}"
                                  </p>
                                )}
                              </div>
                            </div>
                          </CardContent>
                        </Card>
                      </motion.div>
                    );
                  })}
                </div>
              )}
```

```
          </CardContent>
        </Card>

      </div>

      {/* Workout Detail Modal */}
      {selectedWorkout && (
        <WorkoutDetailModal
          isOpen={!!selectedWorkout}
          onClose={() => setSelectedWorkout(null)}
          workout={selectedWorkout}
          useMetric={useMetric}
        />
      )}

      <BottomNavigation />
    </div>
  );
};

export default WorkoutHistoryPage;

```

File: src/services/adminService.ts

Lines: 820

```
// Admin service for managing admin functionality
import { ref, get, set, remove, onValue, query, orderByChild, equalTo, startAt, endAt, push } from "firebase/database";
import { database } from "./firebase";
import { AuthUser } from "@/hooks/useAuth";
import { createNotification, NotificationType } from "./notificationService";

const getAdminKey = (email: string) =>
  email.trim().toLowerCase().replace(/[.#$[\]]/g, "_");

/**
 * Check if an email is an admin
 */
export const checkAdminStatus = async (email: string | null): Promise<boolean> => {
  if (!email) return false;

  try {
    const adminRef = ref(database, `adminEmails/${getAdminKey(email)}`);
    const snapshot = await get(adminRef);
    return snapshot.exists() && snapshot.val() === true;
  } catch (error) {
    console.error("Error checking admin status:", error);
    return false;
  }
};

/**
 * Check if current user is admin
 * @param {AuthUser | null} user - Current user object
 * @returns {Promise<boolean>} True if user is admin
 */
export const isAdmin = async (user: AuthUser | null): Promise<boolean> => {
  if (!user || !user.email) return false;
  return await checkAdminStatus(user.email);
};

/**
 * Get list of all admin emails
 * @returns {Promise<string[]>} Array of admin email addresses
 */
export const getAdminEmails = async (): Promise<string[]> => {
  try {
    const adminRef = ref(database, "adminEmails");
    const snapshot = await get(adminRef);

    if (!snapshot.exists()) return [];

    const emails: string[] = [];
    snapshot.forEach((child) => {
      if (child.val() === true) {
        emails.push(child.key || "");
      }
      return false;
    });

    return emails;
  } catch (error) {
    console.error("Error getting admin emails:", error);
    throw error;
  }
};

/**
 * Add new admin email
 * @param {string} email - Email address to add as admin
 * @returns {Promise<void>}
 */
```

```
 67 | export const addAdminEmail = async (email: string): Promise<void> => {
 68 |   try {
 69 |     if (!email || !email.includes("@")) {
 70 |       throw new Error("Invalid email address");
 71 |     }
 72 |
 73 |     const adminRef = ref(database, `adminEmails/${getAdminKey(email)}`);
 74 |     await set(adminRef, true);
 75 |   } catch (error) {
 76 |     console.error("Error adding admin email:", error);
 77 |     throw error;
 78 |   }
 79 | };
 80 |
 81 | /**
 82 |  * Remove admin email
 83 |  * @param {string} email - Email address to remove from admins
 84 |  * @returns {Promise<void>}
 85 |  */
 86 | export const removeAdminEmail = async (email: string): Promise<void> => {
 87 |   try {
 88 |     const adminRef = ref(database, `adminEmails/${getAdminKey(email)}`);
 89 |     await remove(adminRef);
 90 |   } catch (error) {
 91 |     console.error("Error removing admin email:", error);
 92 |     throw error;
 93 |   }
 94 | };
 95 |
 96 | /**
 97 |  * Promote user to admin by their email
 98 |  * @param {string} email - Email address to promote to admin
 99 |  * @returns {Promise<void>}
100 |  */
101 | export const promoteUserToAdmin = async (email: string): Promise<void> => {
102 |   try {
103 |     if (!email || !email.includes("@")) {
104 |       throw new Error("Invalid email address");
105 |     }
106 |     await addAdminEmail(email);
107 |   } catch (error) {
108 |     console.error("Error promoting user to admin:", error);
109 |     throw error;
110 |   }
111 | };
112 |
113 | /**
114 |  * Demote admin (remove admin privileges)
115 |  * @param {string} email - Email address to demote from admin
116 |  * @returns {Promise<void>}
117 |  */
118 | export const demoteAdmin = async (email: string): Promise<void> => {
119 |   try {
120 |     await removeAdminEmail(email);
121 |   } catch (error) {
122 |     console.error("Error demoting admin:", error);
123 |     throw error;
124 |   }
125 | };
126 |
127 | /**
128 |  * Check if a user is an admin by their email
129 |  * @param {string} email - Email address to check
130 |  * @returns {Promise<boolean>} True if user is admin
131 |  */
132 | export const isUserAdmin = async (email: string | null | undefined): Promise<boolean> => {
133 |   if (!email) return false;
134 |   return await checkAdminStatus(email);
135 | };
136 |
137 | /**
```

```
 138 |  * Get all users for management
 139 |  * @returns {Promise<any[]>} Array of all users
 140 |  */
 141 | export const getAllUsers = async (): Promise<any[]> => {
 142 |   try {
 143 |     const usersRef = ref(database, "users");
 144 |     const snapshot = await get(usersRef);
 145 |
 146 |     if (!snapshot.exists()) return [];
 147 |
 148 |     const users: any[] = [];
 149 |     snapshot.forEach((child) => {
 150 |       users.push({
 151 |         uid: child.key,
 152 |         ...child.val()
 153 |       });
 154 |       return false;
 155 |     });
 156 |
 157 |     return users;
 158 |   } catch (error) {
 159 |     console.error("Error getting all users:", error);
 160 |     throw error;
 161 |   }
 162 | };
 163 |
 164 | /**
 165 |  * Suspend user account
 166 |  * @param {string} userId - User ID to suspend
 167 |  * @param {number} durationDays - Duration in days (optional, default 7)
 168 |  * @param {string} reason - Reason for suspension
 169 |  * @returns {Promise<void>}
 170 |  */
 171 | export const suspendUser = async (
 172 |   userId: string,
 173 |   durationDays: number = 7,
 174 |   reason: string = "Violation of terms of service"
 175 | ): Promise<void> => {
 176 |   try {
 177 |     const userRef = ref(database, `users/${userId}`);
 178 |     const snapshot = await get(userRef);
 179 |
 180 |     if (!snapshot.exists()) {
 181 |       throw new Error("User not found");
 182 |     }
 183 |
 184 |     const userData = snapshot.val();
 185 |     const suspendedUntil = Date.now() + (durationDays * 24 * 60 * 60 * 1000);
 186 |
 187 |     await set(userRef, {
 188 |       ...userData,
 189 |       status: "suspended",
 190 |       suspendedUntil,
 191 |       suspendedReason: reason,
 192 |       updatedAt: Date.now()
 193 |     });
 194 |   } catch (error) {
 195 |     console.error("Error suspending user:", error);
 196 |     throw error;
 197 |   }
 198 | };
 199 |
 200 | /**
 201 |  * Ban user account permanently
 202 |  * @param {string} userId - User ID to ban
 203 |  * @param {string} reason - Reason for ban
 204 |  * @returns {Promise<void>}
 205 |  */
 206 | export const banUser = async (userId: string, reason: string = "Violation of terms of service"):
Promise<void> => {
 207 |   try {
 208 |     const userRef = ref(database, `users/${userId}`);
```

```
209 |     const snapshot = await get(userRef);
210 |
211 |     if (!snapshot.exists()) {
212 |       throw new Error("User not found");
213 |     }
214 |
215 |     const userData = snapshot.val();
216 |
217 |     await set(userRef, {
218 |       ...userData,
219 |       status: "banned",
220 |       bannedAt: Date.now(),
221 |       bannedReason: reason,
222 |       updatedAt: Date.now()
223 |     });
224 |   } catch (error) {
225 |     console.error("Error banning user:", error);
226 |     throw error;
227 |   }
228 | };
229 |
230 | /**
231 |  * Request username change due to misuse
232 |  * Changes username to default format and sends notification
233 |  * @param {string} userId - User ID to change username
234 |  * @param {string} reason - Reason for username change
235 |  * @param {string} adminId - Admin user ID
236 |  * @param {string} adminName - Admin name
237 |  * @returns {Promise<void>}
238 |  */
239 | export const requestUsernameChange = async (
240 |   userId: string,
241 |   reason: string = "Inappropriate username",
242 |   adminId: string,
243 |   adminName: string
244 | ): Promise<void> => {
245 |   try {
246 |     const userRef = ref(database, `users/${userId}`);
247 |     const snapshot = await get(userRef);
248 |
249 |     if (!snapshot.exists()) {
250 |       throw new Error("User not found");
251 |     }
252 |
253 |     const userData = snapshot.val();
254 |     const defaultUsername = `user ${userId.substring(0, 8)}`;
255 |
256 |     // Update username to default format
257 |     await set(userRef, {
258 |       ...userData,
259 |       username: defaultUsername,
260 |       usernameChanged: true,
261 |       usernameChangeReason: reason,
262 |       usernameChangedAt: Date.now(),
263 |       usernameChangedBy: adminId,
264 |       updatedAt: Date.now()
265 |     });
266 |
267 |     // Send notification to user
268 |     await createNotification(userId, {
269 |       type: "username_change_required",
270 |       fromUserId: adminId,
271 |       fromUserName: adminName || "Admin",
272 |       fromUserAvatar: "",
273 |       message: `Your username has been changed due to misuse. Please update it to an appropriate
username. Reason: ${rea...
274 |       reason,
275 |       adminId
276 |     });
277 |
278 |     console.log(`'  Username changed for user ${userId} to ${defaultUsername}`);
279 |   } catch (error) {
```

```
280 |     console.error("Error requesting username change:", error);
281 |     throw error;
282 |   }
283 | };
284 |
285 | /**
286 |  * Unban user account
287 |  * @param {string} userId - User ID to unban
288 |  * @returns {Promise<void>}
289 |  */
290 | export const unbanUser = async (userId: string): Promise<void> => {
291 |   try {
292 |     const userRef = ref(database, `users/${userId}`);
293 |     const snapshot = await get(userRef);
294 |
295 |     if (!snapshot.exists()) {
296 |       throw new Error("User not found");
297 |     }
298 |
299 |     const userData = snapshot.val();
300 |
301 |     await set(userRef, {
302 |       ...userData,
303 |       status: "active",
304 |       bannedAt: null,
305 |       bannedReason: null,
306 |       updatedAt: Date.now()
307 |     });
308 |   } catch (error) {
309 |     console.error("Error unbanning user:", error);
310 |     throw error;
311 |   }
312 | };
313 |
314 | /**
315 |  * Unsuspend user account
316 |  * @param {string} userId - User ID to unsuspend
317 |  * @returns {Promise<void>}
318 |  */
319 | export const unsuspendUser = async (userId: string): Promise<void> => {
320 |   try {
321 |     const userRef = ref(database, `users/${userId}`);
322 |     const snapshot = await get(userRef);
323 |
324 |     if (!snapshot.exists()) {
325 |       throw new Error("User not found");
326 |     }
327 |
328 |     const userData = snapshot.val();
329 |
330 |     await set(userRef, {
331 |       ...userData,
332 |       status: "active",
333 |       suspendedUntil: null,
334 |       suspendedReason: null,
335 |       updatedAt: Date.now()
336 |     });
337 |   } catch (error) {
338 |     console.error("Error unsuspending user:", error);
339 |     throw error;
340 |   }
341 | };
342 |
343 | /**
344 |  * Get reports against a specific user
345 |  * @param {string} userId - User ID
346 |  * @returns {Promise<any[]>} Array of reports
347 |  */
348 | export const getUserReports = async (userId: string): Promise<any[]> => {
349 |   try {
350 |     const reportsRef = ref(database, "reports");
```

```
351 |     const snapshot = await get(reportsRef);
352 |
353 |     if (!snapshot.exists()) return [];
354 |
355 |     const reports: any[] = [];
356 |     snapshot.forEach((child) => {
357 |       const report = child.val();
358 |       if (report.reportedUserId === userId) {
359 |         reports.push({
360 |           id: child.key,
361 |           ...report
362 |         });
363 |       }
364 |       return false;
365 |     });
366 |
367 |     return reports.sort((a, b) => (b.reportedAt || 0) - (a.reportedAt || 0));
368 |   } catch (error) {
369 |     console.error("Error getting user reports:", error);
370 |     throw error;
371 |   }
372 | };
373 |
374 | /**
375 |  * Get all reports for moderation
376 |  * @returns {Promise<any[]>} Array of all reports
377 |  */
378 | export const getAllReports = async (): Promise<any[]> => {
379 |   try {
380 |     const reportsRef = ref(database, "reports");
381 |     const snapshot = await get(reportsRef);
382 |
383 |     if (!snapshot.exists()) return [];
384 |
385 |     const reports: any[] = [];
386 |     snapshot.forEach((child) => {
387 |       reports.push({
388 |         id: child.key,
389 |         ...child.val()
390 |       });
391 |       return false;
392 |     });
393 |
394 |     return reports.sort((a, b) => (b.reportedAt || 0) - (a.reportedAt || 0));
395 |   } catch (error) {
396 |     console.error("Error getting all reports:", error);
397 |     throw error;
398 |   }
399 | };
400 |
401 | /**
402 |  * Resolve a report
403 |  * @param {string} reportId - Report ID
404 |  * @param {string} action - Action taken (e.g., "warned", "suspended", "banned", "dismissed")
405 |  * @returns {Promise<void>}
406 |  */
407 | export const resolveReport = async (reportId: string, action: string = "resolved"):
Promise<void> => {
408 |   try {
409 |     const reportRef = ref(database, `reports/${reportId}`);
410 |     const snapshot = await get(reportRef);
411 |
412 |     if (!snapshot.exists()) {
413 |       throw new Error("Report not found");
414 |     }
415 |
416 |     const reportData = snapshot.val();
417 |
418 |     await set(reportRef, {
419 |       ...reportData,
420 |       status: "resolved",
421 |       resolvedAt: Date.now(),
```

```
422 |       resolvedAction: action
423 |     });
424 |   } catch (error) {
425 |     console.error("Error resolving report:", error);
426 |     throw error;
427 |   }
428 | };
429 |
430 | /**
431 |  * Get system statistics
432 |  * @returns {Promise<any>} System stats object
433 |  */
434 | export const getSystemStats = async (): Promise<any> => {
435 |   try {
436 |     // Get all users
437 |     const usersRef = ref(database, "users");
438 |     const usersSnapshot = await get(usersRef);
439 |
440 |     let totalUsers = 0;
441 |     let activeUsers = 0;
442 |     const thirtyDaysAgo = Date.now() - (30 * 24 * 60 * 60 * 1000);
443 |
444 |     if (usersSnapshot.exists()) {
445 |       usersSnapshot.forEach((child) => {
446 |         totalUsers++;
447 |         const userData = child.val();
448 |         const lastActivity = userData.timestamp || userData.createdAt || 0;
449 |         if (lastActivity >= thirtyDaysAgo) {
450 |           activeUsers++;
451 |         }
452 |         return false;
453 |       });
454 |     }
455 |
456 |     // Get all workouts
457 |     const workoutsRef = ref(database, "workouts");
458 |     const workoutsSnapshot = await get(workoutsRef);
459 |     let totalWorkouts = 0;
460 |
461 |     if (workoutsSnapshot.exists()) {
462 |       workoutsSnapshot.forEach((userWorkouts) => {
463 |         userWorkouts.forEach(() => {
464 |           totalWorkouts++;
465 |           return false;
466 |         });
467 |         return false;
468 |       });
469 |     }
470 |
471 |     // Get all events
472 |     const eventsRef = ref(database, "events");
473 |     const eventsSnapshot = await get(eventsRef);
474 |     let totalEvents = 0;
475 |
476 |     if (eventsSnapshot.exists()) {
477 |       eventsSnapshot.forEach(() => {
478 |         totalEvents++;
479 |         return false;
480 |       });
481 |     }
482 |
483 |     // Get pending reports
484 |     const reportsRef = ref(database, "reports");
485 |     const reportsSnapshot = await get(reportsRef);
486 |     let pendingReports = 0;
487 |
488 |     if (reportsSnapshot.exists()) {
489 |       reportsSnapshot.forEach((child) => {
490 |         const report = child.val();
491 |         if (!report.status || report.status === "pending") {
492 |           pendingReports++;
```

```
        }
        return false;
      });
    }

    return {
      totalUsers,
      activeUsers,
      totalWorkouts,
      totalEvents,
      pendingReports
    };
  } catch (error) {
    console.error("Error getting system stats:", error);
    throw error;
  }
};

/**
 * Delete user account
 * @param {string} userId - User ID to delete
 * @returns {Promise<void>}
 */
export const deleteUser = async (userId: string): Promise<void> => {
  try {
    const userRef = ref(database, `users/${userId}`);
    await remove(userRef);

    // Also remove user's workouts
    const workoutsRef = ref(database, `workouts/${userId}`);
    await remove(workoutsRef);

    // Remove from friends lists
    const friendsRef = ref(database, `friends/${userId}`);
    await remove(friendsRef);

    // Remove friend requests
    const friendRequestsRef = ref(database, `friendRequests/${userId}`);
    await remove(friendRequestsRef);

    // Note: We don't delete messages/conversations to preserve chat history
    // Events created by user will remain but hostId will be invalid
  } catch (error) {
    console.error("Error deleting user:", error);
    throw error;
  }
};

/**
 * Log admin action
 * @param {string} adminEmail - Admin email who performed the action
 * @param {string} action - Action type (e.g., "promote_admin", "suspend_user", "delete_user")
 * @param {any} details - Additional details about the action
 * @returns {Promise<void>}
 */
export const logAdminAction = async (
  adminEmail: string,
  action: string,
  details: any = {}
): Promise<void> => {
  try {
    const logsRef = ref(database, "adminLogs");
    const newLogRef = push(logsRef);

    await set(newLogRef, {
      adminEmail,
      action,
      details,
      timestamp: Date.now()
    });
  } catch (error) {
```

```
564 |     console.error("Error logging admin action:", error);
565 |     // Don't throw - logging failures shouldn't break admin actions
566 |   }
567 | };
568 |
569 | /**
570 |  * Get admin action logs
571 |  * @param {number} limit - Maximum number of logs to return (default: 100)
572 |  * @returns {Promise<any[]>} Array of admin logs
573 |  */
574 | export const getAdminLogs = async (limit: number = 100): Promise<any[]> => {
575 |   try {
576 |     const logsRef = ref(database, "adminLogs");
577 |     const snapshot = await get(logsRef);
578 |
579 |     if (!snapshot.exists()) return [];
580 |
581 |     const logs: any[] = [];
582 |     snapshot.forEach((child) => {
583 |       logs.push({
584 |         id: child.key,
585 |         ...child.val()
586 |       });
587 |       return false;
588 |     });
589 |
590 |     // Sort by timestamp (newest first) and limit
591 |     return logs
592 |       .sort((a, b) => (b.timestamp || 0) - (a.timestamp || 0))
593 |       .slice(0, limit);
594 |   } catch (error) {
595 |     console.error("Error getting admin logs:", error);
596 |     throw error;
597 |   }
598 | };
599 |
600 | /**
601 |  * Remove a photo from user's profile
602 |  * @param {string} userId - User ID
603 |  * @param {number} photoIndex - Index of photo to remove from photos array (optional, if not
provided removes all)
604 |  * @returns {Promise<void>}
605 |  */
606 | export const removeUserPhoto = async (userId: string, photoIndex?: number): Promise<void> => {
607 |   try {
608 |     const userRef = ref(database, `users/${userId}`);
609 |     const snapshot = await get(userRef);
610 |
611 |     if (!snapshot.exists()) {
612 |       throw new Error("User not found");
613 |     }
614 |
615 |     const userData = snapshot.val();
616 |     const photos = userData.photos || [];
617 |
618 |     if (photoIndex !== undefined) {
619 |       // Remove specific photo by index
620 |       if (photoIndex >= 0 && photoIndex < photos.length) {
621 |         const updatedPhotos = photos.filter((_: any, index: number) => index !== photoIndex);
622 |         await set(userRef, {
623 |           ...userData,
624 |           photos: updatedPhotos.length > 0 ? updatedPhotos : null,
625 |           updatedAt: Date.now()
626 |         });
627 |       } else {
628 |         throw new Error("Invalid photo index");
629 |       }
630 |     } else {
631 |       // Remove all photos
632 |       await set(userRef, {
633 |         ...userData,
634 |         photos: null,
```

```
635 |         updatedAt: Date.now()
636 |       });
637 |     }
638 |   } catch (error) {
639 |     console.error("Error removing user photo:", error);
640 |     throw error;
641 |   }
642 | };
643 |
644 | /**
645 |  * Remove user's main photoURL (profile picture from auth)
646 |  * @param {string} userId - User ID
647 |  * @returns {Promise<void>}
648 |  */
649 | export const removeUserPhotoURL = async (userId: string): Promise<void> => {
650 |   try {
651 |     const userRef = ref(database, `users/${userId}`);
652 |     const snapshot = await get(userRef);
653 |
654 |     if (!snapshot.exists()) {
655 |       throw new Error("User not found");
656 |     }
657 |
658 |     const userData = snapshot.val();
659 |
660 |     await set(userRef, {
661 |       ...userData,
662 |       photoURL: null,
663 |       updatedAt: Date.now()
664 |     });
665 |   } catch (error) {
666 |     console.error("Error removing user photoURL:", error);
667 |     throw error;
668 |   }
669 | };
670 |
671 | // ============ ADMIN MODERATION NOTIFICATIONS ============
672 |
673 | export type AdminActionType =
674 |   | "comment_deleted"
675 |   | "event_deleted"
676 |   | "event_cancelled"
677 |   | "warning"
678 |   | "comment_suspended"
679 |   | "comment_restored";
680 |
681 | interface AdminNotificationOptions {
682 |   userId: string;
683 |   adminId: string;
684 |   adminName: string;
685 |   actionType: AdminActionType;
686 |   reason: string;
687 |   eventId?: string;
688 |   eventTitle?: string;
689 |   commentText?: string;
690 | }
691 |
692 | /**
693 |  * Send a notification to a user about an admin action
694 |  * This centralizes all admin-to-user notifications
695 |  */
696 | export const sendAdminNotification = async (options: AdminNotificationOptions): Promise<void> =>
{
697 |   const { userId, adminId, adminName, actionType, reason, eventId, eventTitle, commentText } =
options;
698 |
699 |   // Map action types to notification types and messages
700 |   const notificationMap: Record<AdminActionType, { type: NotificationType; message: string }> = {
701 |     comment_deleted: {
702 |       type: "admin_comment_deleted",
703 |       message: eventTitle
704 |         ? `Your comment on "${eventTitle}" was removed by a moderator. Reason: ${reason}`
705 |         : `Your comment was removed by a moderator. Reason: ${reason}`
```

```
706 |     },
707 |     event_deleted: {
708 |       type: "admin_event_deleted",
709 |       message: eventTitle
710 |         ? `Your event "${eventTitle}" was deleted by a moderator. Reason: ${reason}`
711 |         : `Your event was deleted by a moderator. Reason: ${reason}`
712 |     },
713 |     event_cancelled: {
714 |       type: "admin_event_cancelled",
715 |       message: eventTitle
716 |         ? `Your event "${eventTitle}" was cancelled by a moderator. Reason: ${reason}`
717 |         : `Your event was cancelled by a moderator. Reason: ${reason}`
718 |     },
719 |     warning: {
720 |       type: "admin_warning",
721 |       message: `You have received a warning from a moderator: ${reason}`
722 |     },
723 |     comment_suspended: {
724 |       type: "admin_comment_suspended",
725 |       message: `Your commenting privileges have been suspended. Reason: ${reason}`
726 |     },
727 |     comment_restored: {
728 |       type: "admin_comment_restored",
729 |       message: `Your commenting privileges have been restored.`
730 |     }
731 |   };
732 |
733 |   const { type, message } = notificationMap[actionType];
734 |
735 |   try {
736 |     await createNotification(userId, {
737 |       type,
738 |       fromUserId: adminId,
739 |       fromUserName: adminName || "Admin",
740 |       fromUserAvatar: "", // Admin notifications don't need avatar
741 |       message,
742 |       eventId,
743 |       eventTitle,
744 |       reason,
745 |       adminId
746 |     });
747 |
748 |     console.log(`'  Admin notification sent to user ${userId}: ${actionType}`);
749 |   } catch (error) {
750 |     console.error("'L Error sending admin notification:", error);
751 |     // Don't throw - notification failures shouldn't break admin actions
752 |   }
753 | };
754 |
755 | /**
756 |  * Send warning to a user
757 |  */
758 | export const warnUser = async (
759 |   userId: string,
760 |   adminId: string,
761 |   adminName: string,
762 |   reason: string
763 | ): Promise<void> => {
764 |   try {
765 |     // Log the warning in admin logs
766 |     await logAdminAction(adminName, "warn_user", { userId, reason });
767 |
768 |     // Send notification to user
769 |     await sendAdminNotification({
770 |       userId,
771 |       adminId,
772 |       adminName,
773 |       actionType: "warning",
774 |       reason
775 |     });
776 |
```

```
777 |     // Optionally store warning in user's record
778 |     const warningsRef = ref(database, `users/${userId}/warnings`);
779 |     const newWarningRef = push(warningsRef);
780 |     await set(newWarningRef, {
781 |       adminId,
782 |       adminName,
783 |       reason,
784 |       timestamp: Date.now()
785 |     });
786 |
787 |     console.log(`'  Warning sent to user ${userId}`);
788 |   } catch (error) {
789 |     console.error("'L Error warning user:", error);
790 |     throw error;
791 |   }
792 | };
793 |
794 | /**
795 |  * Get warnings for a user
796 |  */
797 | export const getUserWarnings = async (userId: string): Promise<any[]> => {
798 |   try {
799 |     const warningsRef = ref(database, `users/${userId}/warnings`);
800 |     const snapshot = await get(warningsRef);
801 |
802 |     if (!snapshot.exists()) return [];
803 |
804 |     const warnings: any[] = [];
805 |     snapshot.forEach((child) => {
806 |       warnings.push({
807 |         id: child.key,
808 |         ...child.val()
809 |       });
810 |       return false;
811 |     });
812 |
813 |     return warnings.sort((a, b) => (b.timestamp || 0) - (a.timestamp || 0));
814 |   } catch (error) {
815 |     console.error("Error getting user warnings:", error);
816 |     return [];
817 |   }
818 | };
819 |
820 |
```

File: src/services/authService.ts

Lines: 1776

```
// Authentication service for Google Sign-In, Phone Auth, and Email/Password
import {
  signInWithPopup,
  signInWithRedirect,
  getRedirectResult,
  signOut as firebaseSignOut,
  onAuthStateChanged,
  User,
  signInWithPhoneNumber,
  ConfirmationResult,
  RecaptchaVerifier,
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  sendPasswordResetEmail,
  confirmPasswordReset,
  ActionCodeSettings,
  updateProfile,
  fetchSignInMethodsForEmail
} from "firebase/auth";
import { Browser } from "@capacitor/browser";
import { Capacitor } from "@capacitor/core";
import { GoogleAuth } from "@codetrix-studio/capacitor-google-auth";
import { signInWithCredential, GoogleAuthProvider } from "firebase/auth";
import { ref, set, get, onValue } from "firebase/database";
import { auth, googleProvider, database } from "./firebase";
import { clearUserLocation } from "./locationService";

// Export ConfirmationResult type for use in components
export type { ConfirmationResult };

/**
 * Validate Firebase configuration
 * Checks if Firebase is properly initialized and configured
 * @returns {Promise<{valid: boolean, errors: string[], warnings: string[]}>} Validation result
 */
export const validateFirebaseConfig = async (): Promise<{
  valid: boolean;
  errors: string[];
  warnings: string[];
}> => {
  const errors: string[] = [];
  const warnings: string[] = [];

  try {
    // Check if auth is initialized
    if (!auth) {
      errors.push("Firebase Auth is not initialized");
      return { valid: false, errors, warnings };
    }

    // Check if app exists
    if (!auth.app) {
      errors.push("Firebase App is not initialized");
      return { valid: false, errors, warnings };
    }

    // Get Firebase config from the app
    const app = auth.app;
    const config = app.options;

    // Validate required config fields
    if (!config.apiKey) {
      errors.push("Firebase API Key is missing");
    }
    if (!config.authDomain) {
      errors.push("Firebase Auth Domain is missing");
```

```
    }
    if (!config.projectId) {
      errors.push("Firebase Project ID is missing");
    }
    if (!config.appId) {
      errors.push("Firebase App ID is missing");
    }

    // Expected values based on project
    const expectedProjectId = "pacematch-gps";
    if (config.projectId !== expectedProjectId) {
      warnings.push(`Project ID is "${config.projectId}", expected "${expectedProjectId}"`);
    }

    const expectedAuthDomain = "pacematch-gps.firebaseapp.com";
    if (config.authDomain !== expectedAuthDomain) {
      warnings.push(`Auth Domain is "${config.authDomain}", expected "${expectedAuthDomain}"`);
    }

    // Try to get current user (test auth is working)
    try {
      const currentUser = auth.currentUser;
      if (currentUser) {
        console.log("'  Firebase Auth is working - user is authenticated:", currentUser.uid);
      } else {
        console.log("!9þ  Firebase Auth is initialized but no user is currently signed in");
      }
    } catch (authError: any) {
      warnings.push(`Could not check auth state: ${authError.message}`);
    }

    // Log configuration for debugging (without sensitive data)
    console.log("Ø=Ý  Firebase Configuration Check:", {
      projectId: config.projectId,
      authDomain: config.authDomain,
      hasApiKey: !!config.apiKey,
      hasAppId: !!config.appId,
      databaseURL: config.databaseURL || "Not configured",
    });

  } catch (error: any) {
    errors.push(`Firebase validation error: ${error.message}`);
  }

  return {
    valid: errors.length === 0,
    errors,
    warnings,
  };
};

/**
 * Check if running in Capacitor native app (iOS/Android)
 * @returns {boolean} True if running in Capacitor native platform
 */
const isCapacitorNative = (): boolean => {
  try {
    // Use the imported Capacitor directly
    const platform = Capacitor.getPlatform();
    const isNative = Capacitor.isNativePlatform();

    console.log(`Ø=Ý  Capacitor detection: platform="${platform}", isNative=${isNative}`);

    return isNative;
  } catch (e) {
    console.error(''L Error checking Capacitor platform:', e);

    // Fallback check
    if (typeof (window as any).Capacitor !== 'undefined') {
      const platform = (window as any).Capacitor.getPlatform?.();
      const isNative = (window as any).Capacitor.isNativePlatform?.() === true;
```

```
138 |       console.log(`Ø=Ý  Capacitor fallback: platform="${platform}", isNative=${isNative}`);
139 |       return isNative;
140 |     }
141 |
142 |     console.log(''L Capacitor not detected - running in web browser');
143 |     return false;
144 |   }
145 | };
146 |
147 | /**
148 |  * Check if running in a storage-partitioned browser environment
149 |  * (e.g., embedded browsers in Messenger, Facebook, etc.)
150 |  * BUT exclude Capacitor apps (which use WebView but work fine)
151 |  * @returns {boolean} True if storage-partitioned environment detected
152 |  */
153 | const isStoragePartitioned = (): boolean => {
154 |   // If it's a Capacitor native app, it's NOT storage-partitioned
155 |   // Capacitor WebView works fine with Firebase Auth redirect
156 |   if (isCapacitorNative()) {
157 |     return false;
158 |   }
159 |
160 |   try {
161 |     // Try to access sessionStorage
162 |     const testKey = '__storage_test__';
163 |     sessionStorage.setItem(testKey, 'test');
164 |     sessionStorage.removeItem(testKey);
165 |
166 |     // Check for common embedded browser indicators (but NOT Capacitor)
167 |     const userAgent = navigator.userAgent.toLowerCase();
168 |     const isEmbedded =
169 |       window.parent !== window || // In iframe (not Capacitor)
170 |       userAgent.includes('messenger') || // Facebook Messenger
171 |       userAgent.includes('fban') || // Facebook App
172 |       userAgent.includes('fbav'); // Facebook App
173 |     // Note: We don't check for 'wv' (WebView) because Capacitor uses WebView
174 |     // and it works fine. Only flag actual embedded browsers like Messenger.
175 |     // Note: navigator.standalone === false is NORMAL for regular Safari browsers,
176 |     // so we don't check it here to avoid false positives
177 |
178 |     return isEmbedded;
179 |   } catch (e) {
180 |     // If sessionStorage access fails, check if it's Capacitor first
181 |     // Capacitor apps can have sessionStorage, so if it fails AND it's not Capacitor,
182 |     // then it's likely storage-partitioned
183 |     return !isCapacitorNative();
184 |   }
185 | };
186 |
187 | export interface UserData {
188 |   uid: string;
189 |   displayName: string | null;
190 |   email: string | null;
191 |   phoneNumber: string | null;
192 |   photoURL: string | null;
193 | }
194 |
195 | /**
196 |  * Sign in with Google using redirect (better for mobile/Capacitor) or popup (desktop)
197 |  * @returns {Promise<UserData | null>} User object with uid, displayName, email, photoURL, or
null if redirect was initi...
198 |  */
199 | export const signInWithGoogle = async (): Promise<UserData | null> => {
200 |   // Check if we're in a storage-partitioned environment BEFORE attempting auth
201 |   if (isStoragePartitioned()) {
202 |     throw new Error("STORAGE_PARTITIONED: Please open this app in your regular browser (Chrome,
Safari, Firefox) instead...
203 |   }
204 |
205 |   // Use native Google Sign-In for Capacitor apps (best UX - no browser)
206 |   // Use popup/redirect for web browsers
207 |   if (isCapacitorNative()) {
208 |     console.log("Ø=Üñ Capacitor native app detected - using native Google Sign-In");
```

```
209 |
210 |     try {
211 |       // Initialize Google Auth plugin
212 |       // The plugin automatically reads serverClientId (Web Client ID) from capacitor.config.ts
213 |       // Web Client ID: 891545961086-cs7aq62rgshps172c95ijdcnh2lsej5r.apps.googleusercontent.com
214 |       console.log("Ø=Ý  Initializing Google Auth plugin...");
215 |       console.log("Ø=ÜË Using Web Client ID from capacitor.config.ts for Firebase token
verification");
216 |       try {
217 |         await GoogleAuth.initialize();
218 |         console.log("'  Google Auth plugin initialized successfully");
219 |       } catch (initError: any) {
220 |         console.error("'L Failed to initialize Google Auth plugin:", initError);
221 |         console.error("'L Init error type:", typeof initError);
222 |         console.error("'L Init error keys:", Object.keys(initError || {}));
223 |         console.error("'L Init error message:", initError?.message);
224 |         console.error("'L Init error code:", initError?.code);
225 |         console.error("'L Init error toString:", String(initError));
226 |         throw new Error(`Failed to initialize Google Auth: ${initError?.message ||
initError?.code || String(initError)}...
227 |       }
228 |
229 |       // Sign in with native Google Auth
230 |       console.log("Ø=Ý  Calling GoogleAuth.signIn()...");
231 |       let result;
232 |       try {
233 |         result = await GoogleAuth.signIn();
234 |         console.log("'  GoogleAuth.signIn() completed");
235 |       } catch (signInError: any) {
236 |         // Log every possible property of the error
237 |         console.error("'L GoogleAuth.signIn() FAILED");
238 |         console.error("'L Raw error:", signInError);
239 |         console.error("'L Error type:", typeof signInError);
240 |
241 |         // Try to extract all properties
242 |         if (signInError) {
243 |           try {
244 |             console.error("'L Error.keys:", Object.keys(signInError));
245 |             for (const key of Object.keys(signInError)) {
246 |               console.error(`'L Error.${key}:`, signInError[key]);
247 |             }
248 |           } catch (e) {
249 |             console.error("'L Could not enumerate error keys:", e);
250 |           }
251 |         }
252 |
253 |         // Check specific properties
254 |         console.error("'L Error.message:", signInError?.message);
255 |         console.error("'L Error.code:", signInError?.code);
256 |         console.error("'L Error.statusCode:", signInError?.statusCode);
257 |         console.error("'L Error.status:", signInError?.status);
258 |         console.error("'L Error.name:", signInError?.name);
259 |         console.error("'L Error.toString():", String(signInError));
260 |         console.error("'L Error JSON:", JSON.stringify(signInError,
Object.getOwnPropertyNames(signInError)));
261 |
262 |         // Check for common error codes
263 |         const errorCode = signInError?.code || signInError?.statusCode || signInError?.status;
264 |         if (errorCode) {
265 |           console.error(`'L ERROR CODE FOUND: ${errorCode}`);
266 |           if (errorCode === 10 || errorCode === '10' || String(errorCode).includes('10')) {
267 |             // If Error Code 10, fallback to web-based sign-in instead of throwing error
268 |             console.log("& þ  Error Code 10 detected - falling back to web-based sign-in");
269 |             throw new Error("FALLBACK_TO_WEB");
270 |           } else if (errorCode === 7 || errorCode === '7') {
271 |             throw new Error("NETWORK_ERROR (Code 7): Check your internet connection");
272 |           } else if (errorCode === 12500 || errorCode === '12500') {
273 |             throw new Error("SIGN_IN_CANCELLED (Code 12500): User cancelled sign-in");
274 |           }
275 |         }
276 |
277 |         // If we can't get a specific code, throw with all available info
278 |         const errorMsg = signInError?.message || String(signInError) || 'Unknown error';
279 |
```

```
        throw new Error(`Google Sign-In failed: ${errorMsg} (Code: ${errorCode || 'unknown'})`);
      }

      console.log("Google Sign-In result:", result);

      if (!result || !result.authentication) {
        throw new Error("Google Sign-In was cancelled or failed - no authentication data");
      }

      const { idToken, accessToken } = result.authentication;

      if (!idToken || !accessToken) {
        throw new Error("Google Sign-In failed - missing tokens");
      }

      console.log("Got Google tokens, exchanging for Firebase credential...");

      // Create Firebase credential from Google token
      const credential = GoogleAuthProvider.credential(idToken, accessToken);

      // Sign in to Firebase with the credential
      const firebaseUserCredential = await signInWithCredential(auth, credential);
      const user = firebaseUserCredential.user;

      console.log("'  Successfully signed in to Firebase:", user.uid);

      // Save user to Firebase Realtime Database
      await saveUserToDatabase(user);

      return {
        uid: user.uid,
        displayName: user.displayName,
        email: user.email,
        photoURL: user.photoURL,
        phoneNumber: user.phoneNumber
      };
    } catch (error: any) {
      console.error("'L Error with native Google Sign-In:", error);

      // If Error Code 10 (Android OAuth Client ID issue), fallback to web-based sign-in
      if (error.message === "FALLBACK_TO_WEB" || error.code === 10 ||
error.message?.includes('DEVELOPER_ERROR')) {
        console.log('  Native sign-in failed (Error Code 10) - falling back to web-based sign-
in");
        // Fall through to web-based sign-in below
      } else {
        // For other errors, throw them
        if (error.message?.includes('cancelled') || error.message?.includes('CANCELED')) {
          throw new Error("Sign-in was cancelled. Please try again.");
        }

        // Provide more helpful error messages based on common issues
        let errorMessage = `Native Google Sign-In failed: ${error.message || 'Unknown error'}`;

        if (error.message?.includes('INVALID_CLIENT')) {
          errorMessage = "Invalid client: Check that Android OAuth Client ID is configured
correctly in Google Cloud Con
        } else if (error.message?.includes('SIGN_IN_REQUIRED')) {
          errorMessage = "Sign-in required: Make sure Google Sign-In services are enabled.";
        }

        console.error(`'L ${errorMessage}`);
        throw new Error(errorMessage);
      }
    }
  }

  // For web (desktop OR mobile browser), or fallback from native sign-in
  // Try popup first (better UX), fallback to redirect
  try {
    const isMobileDevice = () => {
      const userAgent = navigator.userAgent.toLowerCase();
      const isMobile = /android|webos|iphone|ipad|ipod|blackberry|iemobile|opera mini/
i.test(userAgent);
      const isTouchDevice = 'ontouchstart' in window || navigator.maxTouchPoints > 0;
```

```
351 |       const isSmallScreen = window.innerWidth < 768;
352 |       return isMobile || (isTouchDevice && isSmallScreen);
353 |     };
354 |
355 |     // If we're here from Capacitor fallback, use redirect (more reliable in WebView)
356 |     if (isCapacitorNative()) {
357 |       console.log("Ø=Üñ Capacitor fallback - using redirect method for authentication");
358 |       await signInWithRedirect(auth, googleProvider);
359 |       return null; // Will be handled by redirect callback in handleRedirectResult()
360 |     }
361 |
362 |     // For desktop, try popup first (better UX)
363 |     const platform = isMobileDevice() ? "Mobile Browser" : "Desktop";
364 |     console.log(`Ø=Ü» ${platform} detected - using popup method for authentication`);
365 |     const result = await signInWithPopup(auth, googleProvider);
366 |     const user = result.user;
367 |
368 |     // Save user to Firebase Realtime Database
369 |     await saveUserToDatabase(user);
370 |
371 |     return {
372 |       uid: user.uid,
373 |       displayName: user.displayName,
374 |       email: user.email,
375 |       photoURL: user.photoURL,
376 |       phoneNumber: user.phoneNumber
377 |     };
378 |   } catch (error: any) {
379 |     // Handle specific auth errors
380 |     if (error.code === 'auth/popup-closed-by-user') {
381 |       throw new Error("Sign-in was cancelled. Please try again.");
382 |     }
383 |     if (error.code === 'auth/popup-blocked') {
384 |       console.log("Popup blocked, switching to redirect method...");
385 |       // Fallback to redirect
386 |       await signInWithRedirect(auth, googleProvider);
387 |       return null;
388 |     }
389 |
390 |     // Handle disallowed_useragent error - use redirect instead
391 |     if (error.code === 'auth/unauthorized-domain' ||
392 |         error.message?.includes('disallowed_useragent') ||
393 |         error.message?.includes('Use secure browsers')) {
394 |       console.log("Popup authentication blocked, switching to redirect method...");
395 |
396 |       // Check again if storage-partitioned before redirect
397 |       if (isStoragePartitioned()) {
398 |         throw new Error("STORAGE_PARTITIONED: Please open this app in your regular browser
(Chrome, Safari, Firefox) ins...
399 |       }
400 |
401 |       // Use redirect method as fallback
402 |       await signInWithRedirect(auth, googleProvider);
403 |       return null; // Will be handled by redirect callback
404 |     }
405 |
406 |     // Handle storage-partitioned error from redirect
407 |     if (error.message?.includes('missing initial state') ||
408 |         error.message?.includes('sessionStorage') ||
409 |         error.message?.includes('storage-partitioned')) {
410 |       throw new Error("STORAGE_PARTITIONED: Please open this app in your regular browser
(Chrome, Safari, Firefox) inste...
411 |     }
412 |
413 |     // COOP warnings are logged but don't break functionality - sign-in still works
414 |     console.error("Error signing in with Google:", error);
415 |     throw error;
416 |   }
417 | };
418 |
419 | /**
420 |  * Handle redirect result after signInWithRedirect
421 |  * Call this in your App component after page load to check if user returned from redirect
```

```
422 |  * @returns {Promise<UserData | null>} User object if redirect was successful, null otherwise
423 |  */
424 | export const handleRedirectResult = async (): Promise<UserData | null> => {
425 |   try {
426 |     const result = await getRedirectResult(auth);
427 |     if (result && result.user) {
428 |       const user = result.user;
429 |
430 |       // Save user to Firebase Realtime Database
431 |       await saveUserToDatabase(user);
432 |
433 |       return {
434 |         uid: user.uid,
435 |         displayName: user.displayName,
436 |         email: user.email,
437 |         photoURL: user.photoURL,
438 |         phoneNumber: user.phoneNumber
439 |       };
440 |     }
441 |     return null;
442 |   } catch (error: any) {
443 |     // Handle storage-partitioned error
444 |     if (error.message?.includes('missing initial state') ||
445 |         error.message?.includes('sessionStorage') ||
446 |         error.message?.includes('storage-partitioned')) {
447 |       console.error("Storage-partitioned browser detected in redirect result");
448 |       // Don't throw - let the user try again with better guidance
449 |       return null;
450 |     }
451 |     console.error("Error handling redirect result:", error);
452 |     throw error;
453 |   }
454 | };
455 |
456 | /**
457 |  * Sign out current user
458 |  */
459 | export const signOut = async (): Promise<void> => {
460 |   try {
461 |     // Get current user ID before signing out (needed to clear location)
462 |     const currentUser = auth.currentUser;
463 |     const userId = currentUser?.uid;
464 |
465 |     // If using native Google Auth in Capacitor, sign out from Google too
466 |     if (isCapacitorNative()) {
467 |       try {
468 |         await GoogleAuth.signOut();
469 |         console.log("Signed out from native Google Auth");
470 |       } catch (error) {
471 |         console.error("Error signing out from Google Auth:", error);
472 |         // Continue with Firebase sign out even if Google sign out fails
473 |       }
474 |     }
475 |
476 |     // Clean up reCAPTCHA before signing out
477 |     // This allows reCAPTCHA to be properly reinitialized on next login
478 |     cleanupRecaptcha();
479 |
480 |     // Clear user's location data before signing out
481 |     // This ensures the user becomes invisible to others immediately
482 |     if (userId) {
483 |       try {
484 |         await clearUserLocation(userId);
485 |       } catch (locationError) {
486 |         // Log but don't fail sign out if location cleanup fails
487 |         console.error("Error clearing location on sign out:", locationError);
488 |       }
489 |     }
490 |
491 |     // Clear user-specific localStorage data before signing out
492 |     localStorage.removeItem("userProfile");
```

```
493 |     localStorage.removeItem("userProfileUserId");
494 |     localStorage.removeItem("activityState");
495 |
496 |     await firebaseSignOut(auth);
497 |   } catch (error) {
498 |     console.error("Error signing out:", error);
499 |     throw error;
500 |   }
501 | };
502 |
503 | /**
504 |  * Save user data to Firebase Realtime Database
505 |  * @param {User} user - Firebase user object
506 |  */
507 | const saveUserToDatabase = async (user: User): Promise<void> => {
508 |   try {
509 |     const userRef = ref(database, `users/${user.uid}`);
510 |     const snapshot = await get(userRef);
511 |
512 |     // Only update if user doesn't exist or update basic info
513 |     if (!snapshot.exists()) {
514 |       await set(userRef, {
515 |         name: user.displayName || "",
516 |         email: user.email || "",
517 |         phoneNumber: user.phoneNumber || "",
518 |         photoURL: user.photoURL || "",
519 |         createdAt: Date.now(),
520 |         // Profile setup fields (will be updated when user completes profile)
521 |         activity: null,
522 |         gender: null,
523 |         visible: true, // Default to visible when user is created
524 |         lat: null,
525 |         lng: null,
526 |         timestamp: null,
527 |         // Matching preferences (defaults)
528 |         fitnessLevel: "intermediate",
529 |         pace: null, // Will be calculated from workout history
530 |         visibility: {
531 |           visibleToAllLevels: true,
532 |           allowedLevels: ["beginner", "intermediate", "pro"]
533 |         },
534 |         searchFilter: "all", // Who do I want to find? (Beginner/Intermediate/Pro/All)
535 |         radiusPreference: "normal",
536 |         // Profile discovery settings (opt-in by default)
537 |         profileVisible: false, // Keep profile hidden until user enables discovery
538 |         generalLocation: null // General location (e.g., "Pasig", "UP Diliman")
539 |       });
540 |     } else {
541 |       // Update basic info if it changed
542 |       const existingData = snapshot.val();
543 |       await set(userRef, {
544 |         ...existingData,
545 |         name: user.displayName || existingData.name,
546 |         email: user.email || existingData.email,
547 |         phoneNumber: user.phoneNumber || existingData.phoneNumber || "",
548 |         photoURL: user.photoURL || existingData.photoURL,
549 |         // Ensure matching fields exist with defaults if missing
550 |         fitnessLevel: existingData.fitnessLevel || "intermediate",
551 |         visibility: existingData.visibility || {
552 |           visibleToAllLevels: true,
553 |           allowedLevels: ["beginner", "intermediate", "pro"]
554 |         },
555 |         searchFilter: existingData.searchFilter || "all",
556 |         radiusPreference: existingData.radiusPreference || "normal",
557 |         // Ensure profile discovery fields exist with defaults if missing
558 |         profileVisible: existingData.profileVisible !== undefined ?
existingData.profileVisible : false,
559 |         generalLocation: existingData.generalLocation || null
560 |       });
561 |     }
562 |   } catch (error) {
563 |     console.error("Error saving user to database:", error);
```

```
564 |     throw error;
565 |   }
566 | };
567 |
568 | /**
569 |  * Get user data from Firebase
570 |  * @param {string} userId - User ID
571 |  * @returns {Promise<any>} User data object
572 |  */
573 | export const getUserData = async (userId: string): Promise<any> => {
574 |   try {
575 |     const userRef = ref(database, `users/${userId}`);
576 |     const snapshot = await get(userRef);
577 |
578 |     if (snapshot.exists()) {
579 |       return snapshot.val();
580 |     }
581 |     return null;
582 |   } catch (error) {
583 |     console.error("Error getting user data:", error);
584 |     throw error;
585 |   }
586 | };
587 |
588 | /**
589 |  * Subscribe to realtime updates for a user's profile
590 |  * @param {string} userId - User ID
591 |  * @param {(data: any | null) => void} callback - Handler invoked with latest data
592 |  * @returns {() => void} Unsubscribe function
593 |  */
594 | export const listenToUserProfile = (
595 |   userId: string,
596 |   callback: (data: any | null) => void
597 | ): (() => void) => {
598 |   const userRef = ref(database, `users/${userId}`);
599 |
600 |   const unsubscribe = onValue(
601 |     userRef,
602 |     (snapshot) => {
603 |       if (snapshot.exists()) {
604 |         callback(snapshot.val());
605 |       } else {
606 |         callback(null);
607 |       }
608 |     },
609 |     (error) => {
610 |       console.error("Error listening to user profile:", error);
611 |       callback(null);
612 |     }
613 |   );
614 |
615 |   return () => unsubscribe();
616 | };
617 |
618 | /**
619 |  * Update user profile data
620 |  * @param {string} userId - User ID
621 |  * @param {any} profileData - Profile data to update
622 |  */
623 | export const updateUserProfile = async (userId: string, profileData: any): Promise<void> => {
624 |   try {
625 |     const userRef = ref(database, `users/${userId}`);
626 |     const snapshot = await get(userRef);
627 |     const existingData = snapshot.exists() ? snapshot.val() : {};
628 |
629 |     await set(userRef, {
630 |       ...existingData,
631 |       ...profileData,
632 |       updatedAt: Date.now()
633 |     });
634 |   } catch (error) {
```

```
635 |     console.error("Error updating user profile:", error);
636 |     throw error;
637 |   }
638 | };
639 |
640 | /**
641 |  * Listen to authentication state changes
642 |  * @param {Function} callback - Callback function that receives user object or null
643 |  * @returns {Function} Unsubscribe function
644 |  */
645 | export const onAuthStateChange = (callback: (user: User | null) => void) => {
646 |   return onAuthStateChanged(auth, callback);
647 | };
648 |
649 | /**
650 |  * Check if an email address is already associated with an existing account
651 |  * @param {string} email - Email address to check
652 |  * @returns {Promise<boolean>} True if email exists, false otherwise
653 |  */
654 | export const checkEmailExists = async (email: string): Promise<boolean> => {
655 |   try {
656 |     // Validate email format first
657 |     if (!email || !email.includes('@')) {
658 |       return false;
659 |     }
660 |
661 |     // Fetch sign-in methods for the email
662 |     // If any methods are returned, the email is already registered
663 |     const signInMethods = await fetchSignInMethodsForEmail(auth, email);
664 |
665 |     // If signInMethods array has any items, email is already in use
666 |     return signInMethods.length > 0;
667 |   } catch (error: any) {
668 |     // If there's an error (e.g., invalid email format), return false
669 |     // The actual signup will catch and handle the error properly
670 |     console.error("Error checking email existence:", error);
671 |     return false;
672 |   }
673 | };
674 |
675 | /**
676 |  * Generate a 6-digit verification code
677 |  * @returns {string} 6-digit code
678 |  */
679 | const generateVerificationCode = (): string => {
680 |   return Math.floor(100000 + Math.random() * 900000).toString();
681 | };
682 |
683 | /**
684 |  * Send email OTP for sign-in (works for any email, existing or new)
685 |  * @param {string} email - User email address
686 |  * @returns {Promise<string>} Verification code (for development/testing)
687 |  */
688 | export const sendEmailOTPForSignIn = async (email: string): Promise<string> => {
689 |   try {
690 |     // Validate email format
691 |     if (!email || !email.includes('@')) {
692 |       throw new Error("Please enter a valid email address");
693 |     }
694 |
695 |     // Generate 6-digit code
696 |     const code = generateVerificationCode();
697 |
698 |     // Store code in Firebase Realtime Database with 10-minute expiration
699 |     const verificationRef = ref(database, `emailOTPSignIn/${email.replace(/[.#$[\]]/g, '_')}`);
700 |     await set(verificationRef, {
701 |       code: code,
702 |       email: email,
703 |       createdAt: Date.now(),
704 |       expiresAt: Date.now() + 10 * 60 * 1000, // 10 minutes
705 |       attempts: 0
```

```
706 |     });
707 |
708 |     console.log("Ø=Üç OTP code generated for sign-in:", email);
709 |     console.log("Ø=Ý  Code:", code); // For development - also log for debugging
710 |
711 |     // Send email with code via email service
712 |     try {
713 |       // Try Firebase Cloud Functions first (most secure, recommended)
714 |       const { getFunctions, httpsCallable } = await import('firebase/functions');
715 |       const functions = getFunctions();
716 |       const sendOTPEmail = httpsCallable(functions, 'sendOTPEmail');
717 |       await sendOTPEmail({ email, code });
718 |       console.log("'  Email sent successfully via Firebase Cloud Functions");
719 |     } catch (firebaseError: any) {
720 |       // Check if it's a permission/not found error (function not deployed)
721 |       const isFunctionNotDeployed =
722 |         firebaseError?.code === 'functions/not-found' ||
723 |         firebaseError?.code === 'permission-denied' ||
724 |         firebaseError?.message?.includes('permission') ||
725 |         firebaseError?.message?.includes('not found') ||
726 |         firebaseError?.message?.includes('not deployed');
727 |
728 |       if (isFunctionNotDeployed) {
729 |         console.log("!9þ  Firebase Cloud Function not deployed yet, trying fallback...");
730 |       } else {
731 |         console.warn("& þ  Firebase Functions error:", firebaseError.message);
732 |       }
733 |
734 |       // If Firebase Functions fails, try EmailJS (simpler, frontend-only)
735 |       try {
736 |         const { sendOTPEmailSimple } = await import('./emailServiceSimple');
737 |         await sendOTPEmailSimple({
738 |           email,
739 |           code,
740 |         });
741 |         console.log("'  Email sent successfully via EmailJS");
742 |       } catch (emailError: any) {
743 |         // If EmailJS fails, try SendGrid backend endpoint
744 |         try {
745 |           const { sendOTPEmail } = await import('./emailService');
746 |           await sendOTPEmail(email, code);
747 |           console.log("'  Email sent successfully via SendGrid");
748 |         } catch (sendGridError: any) {
749 |           // If all fail, log error but still allow development testing
750 |           console.warn("& þ  Email sending failed. Check email service configuration.");
751 |           if (!isFunctionNotDeployed) {
752 |             console.warn("Firebase Functions error:", firebaseError.message);
753 |           }
754 |           console.warn("EmailJS error:", emailError.message);
755 |           console.warn("SendGrid error:", sendGridError.message);
756 |           console.warn("For development: Code is", code, "- check console");
757 |         }
758 |       }
759 |     }
760 |   } catch (error: any) {
761 |     console.error("'L Error sending OTP for sign-in:", error);
762 |     throw error;
763 |   }
764 | };
765 |
766 | /**
767 |  * Send email verification code (for sign-up only)
768 |  * @param {string} email - User email address
769 |  * @returns {Promise<string>} Verification code (for development/testing)
770 |  */
771 | export const sendEmailVerificationCode = async (email: string): Promise<string> => {
772 |   try {
773 |     // Validate email format
774 |     if (!email || !email.includes('@')) {
775 |       throw new Error("Please enter a valid email address");
776 |     }
```

```
777 |
778 |     // Check if email is already registered
779 |     const emailExists = await checkEmailExists(email);
780 |     if (emailExists) {
781 |       throw new Error("This email is already registered. Please sign in instead.");
782 |     }
783 |
784 |     // Generate 6-digit code
785 |     const code = generateVerificationCode();
786 |
787 |     // Store code in Firebase Realtime Database with 10-minute expiration
788 |     const verificationRef = ref(database, `emailVerifications/${email.replace(/[.#$[\]]/g, '_')}
`789 |     await set(verificationRef, {
790 |       code: code,
791 |       email: email,
792 |       createdAt: Date.now(),
793 |       expiresAt: Date.now() + 10 * 60 * 1000, // 10 minutes
794 |       attempts: 0
795 |     });
796 |
797 |     console.log("Ø=Üç Verification code generated for:", email);
798 |     console.log("Ø=Ý  Code:", code); // For development - also log for debugging
799 |
800 |     // Send email with code via email service
801 |     try {
802 |       // Try Firebase Cloud Functions first (most secure, recommended)
803 |       const { getFunctions, httpsCallable } = await import('firebase/functions');
804 |       const functions = getFunctions();
805 |       const sendOTPEmail = httpsCallable(functions, 'sendOTPEmail');
806 |       await sendOTPEmail({ email, code });
807 |       console.log("'  Email sent successfully via Firebase Cloud Functions");
808 |     } catch (firebaseError: any) {
809 |       // Check if it's a permission/not found error (function not deployed)
810 |       const isFunctionNotDeployed =
811 |         firebaseError?.code === 'functions/not-found' ||
812 |         firebaseError?.code === 'permission-denied' ||
813 |         firebaseError?.message?.includes('permission') ||
814 |         firebaseError?.message?.includes('not found') ||
815 |         firebaseError?.message?.includes('not deployed');
816 |
817 |       if (isFunctionNotDeployed) {
818 |         console.log("!9þ  Firebase Cloud Function not deployed yet, trying fallback...");
819 |       } else {
820 |         console.warn("& þ  Firebase Functions error:", firebaseError.message);
821 |       }
822 |
823 |       // If Firebase Functions fails, try EmailJS (simpler, frontend-only)
824 |       try {
825 |         const { sendOTPEmailSimple } = await import('./emailServiceSimple');
826 |         await sendOTPEmailSimple({
827 |           email,
828 |           code,
829 |         });
830 |         console.log("'  Email sent successfully via EmailJS");
831 |       } catch (emailError: any) {
832 |         // If EmailJS fails, try SendGrid backend endpoint
833 |         try {
834 |           const { sendOTPEmail } = await import('./emailService');
835 |           await sendOTPEmail(email, code);
836 |           console.log("'  Email sent successfully via SendGrid");
837 |         } catch (sendGridError: any) {
838 |           // If all fail, log error but still allow development testing
839 |           console.warn("& þ  Email sending failed. Check email service configuration.");
840 |           if (!isFunctionNotDeployed) {
841 |             console.warn("Firebase Functions error:", firebaseError.message);
842 |           }
843 |           console.warn("EmailJS error:", emailError.message);
844 |           console.warn("SendGrid error:", sendGridError.message);
845 |           console.warn("For development: Code is", code, "- check console");
846 |
847 |           // In production, you might want to throw error here
```

```
848 |           // For development, we'll continue and let user see code in console
849 |           // throw new Error("Failed to send verification email. Please check your email service
configuration.");
850 |         }
851 |       }
852 |     }
853 |
854 |     // Don't return code in production - user should check email
855 |     // For development, still log it but don't return
856 |     // return code; // REMOVED - user must check email
857 |   } catch (error: any) {
858 |     console.error("'L Error sending verification code:", error);
859 |     throw error;
860 |   }
861 | };
862 |
863 | /**
864 |  * Verify email verification code
865 |  * @param {string} email - User email address
866 |  * @param {string} code - 6-digit verification code
867 |  * @returns {Promise<boolean>} True if code is valid, false otherwise
868 |  */
869 | export const verifyEmailCode = async (email: string, code: string): Promise<boolean> => {
870 |   try {
871 |     if (!email || !code || code.length !== 6) {
872 |       return false;
873 |     }
874 |
875 |     const verificationRef = ref(database, `emailVerifications/${email.replace(/[.#$[\]]/g, '_')}
`);
876 |     const snapshot = await get(verificationRef);
877 |
878 |     if (!snapshot.exists()) {
879 |       return false;
880 |     }
881 |
882 |     const verificationData = snapshot.val();
883 |
884 |     // Check if code expired
885 |     if (Date.now() > verificationData.expiresAt) {
886 |       // Clean up expired code
887 |       await set(verificationRef, null);
888 |       throw new Error("Verification code has expired. Please request a new one.");
889 |     }
890 |
891 |     // Check attempt limit (max 5 attempts)
892 |     if (verificationData.attempts >= 5) {
893 |       await set(verificationRef, null);
894 |       throw new Error("Too many failed attempts. Please request a new verification code.");
895 |     }
896 |
897 |     // Verify code
898 |     if (verificationData.code !== code) {
899 |       // Increment attempts
900 |       await set(verificationRef, {
901 |         ...verificationData,
902 |         attempts: verificationData.attempts + 1
903 |       });
904 |       return false;
905 |     }
906 |
907 |     // Code is valid - mark as verified (don't delete yet, will be used during signup)
908 |     await set(verificationRef, {
909 |       ...verificationData,
910 |       verified: true,
911 |       verifiedAt: Date.now()
912 |     });
913 |
914 |     return true;
915 |   } catch (error: any) {
916 |     console.error("'L Error verifying code:", error);
917 |     throw error;
918 |   }
```

```
};

/**
 * Clean up verification code after successful signup
 * @param {string} email - User email address
 */
const cleanupVerificationCode = async (email: string): Promise<void> => {
  try {
    const verificationRef = ref(database, `emailVerifications/${email.replace(/[.#$[\]]/g, '_')}`);
    await set(verificationRef, null);
  } catch (error) {
    console.error("Error cleaning up verification code:", error);
  }
};

/**
 * Sign up with email and password (requires email verification first)
 * @param {string} email - User email address
 * @param {string} password - User password
 * @param {string} displayName - User display name
 * @returns {Promise<UserData>} User object with uid, email, displayName, and other user data
 */
export const signUpWithEmail = async (
  email: string,
  password: string,
  displayName: string
): Promise<UserData> => {
  try {
    // Validate inputs
    if (!email || !email.includes('@')) {
      throw new Error("Please enter a valid email address");
    }
    if (!password || password.length < 6) {
      throw new Error("Password must be at least 6 characters");
    }
    if (!displayName || displayName.trim().length === 0) {
      throw new Error("Please enter your name");
    }

    // Verify that email was verified before allowing signup
    const verificationRef = ref(database, `emailVerifications/${email.replace(/[.#$[\]]/g, '_')}`);
    const snapshot = await get(verificationRef);

    if (!snapshot.exists() || !snapshot.val().verified) {
      throw new Error("Please verify your email address first by entering the 6-digit code sent to your email.");
    }

    console.log("Ø=Üç Creating account with verified email:", email);

    // Create user account
    const userCredential = await createUserWithEmailAndPassword(auth, email, password);
    const user = userCredential.user;

    // Update display name
    if (displayName) {
      await updateProfile(user, { displayName: displayName.trim() });
    }

    console.log("'  Account created successfully! User:", user.uid);

    // Clean up verification code
    await cleanupVerificationCode(email);

    // Save user to Firebase Realtime Database
    await saveUserToDatabase(user);

    return {
      uid: user.uid,
      displayName: user.displayName,
      email: user.email,
      phoneNumber: user.phoneNumber,
```

```
 990 |       photoURL: user.photoURL
 991 |     };
 992 |   } catch (error: any) {
 993 |     console.error("'L Error signing up:", error);
 994 |
 995 |     // Handle specific errors
 996 |     if (error.code === 'auth/email-already-in-use') {
 997 |       throw new Error("This email is already registered. Please sign in instead.");
 998 |     }
 999 |     if (error.code === 'auth/invalid-email') {
1000 |       throw new Error("Invalid email address. Please check and try again.");
1001 |     }
1002 |     if (error.code === 'auth/weak-password') {
1003 |       throw new Error("Password is too weak. Please use a stronger password.");
1004 |     }
1005 |     if (error.code === 'auth/operation-not-allowed') {
1006 |       throw new Error("Email/password authentication is not enabled. Please contact support.");
1007 |     }
1008 |
1009 |     throw error;
1010 |   }
1011 | };
1012 |
1013 | /**
1014 |  * Sign in with email and password
1015 |  * @param {string} email - User email address
1016 |  * @param {string} password - User password
1017 |  * @returns {Promise<UserData>} User object with uid, email, displayName, and other user data
1018 |  */
1019 | export const signInWithEmail = async (
1020 |   email: string,
1021 |   password: string
1022 | ): Promise<UserData> => {
1023 |   try {
1024 |     // Validate inputs
1025 |     if (!email || !email.includes('@')) {
1026 |       throw new Error("Please enter a valid email address");
1027 |     }
1028 |     if (!password) {
1029 |       throw new Error("Please enter your password");
1030 |     }
1031 |
1032 |     console.log("Ø=Üç Signing in with email:", email);
1033 |
1034 |     // Sign in user
1035 |     const userCredential = await signInWithEmailAndPassword(auth, email, password);
1036 |     const user = userCredential.user;
1037 |
1038 |     console.log("'  Sign-in successful! User:", user.uid);
1039 |
1040 |     // Update user data in database if needed
1041 |     await saveUserToDatabase(user);
1042 |
1043 |     return {
1044 |       uid: user.uid,
1045 |       displayName: user.displayName,
1046 |       email: user.email,
1047 |       phoneNumber: user.phoneNumber,
1048 |       photoURL: user.photoURL
1049 |     };
1050 |   } catch (error: any) {
1051 |     console.error("'L Error signing in:", error);
1052 |
1053 |     // Handle specific errors
1054 |     if (error.code === 'auth/user-not-found') {
1055 |       throw new Error("No account found with this email. Please sign up first.");
1056 |     }
1057 |     if (error.code === 'auth/wrong-password') {
1058 |       throw new Error("Incorrect password. Please try again.");
1059 |     }
1060 |     if (error.code === 'auth/invalid-email') {
```

```
1061 |       throw new Error("Invalid email address. Please check and try again.");
1062 |     }
1063 |     if (error.code === 'auth/user-disabled') {
1064 |       throw new Error("This account has been disabled. Please contact support.");
1065 |     }
1066 |     if (error.code === 'auth/too-many-requests') {
1067 |       throw new Error("Too many failed attempts. Please try again later.");
1068 |     }
1069 |
1070 |     throw error;
1071 |   }
1072 | };
1073 |
1074 | /**
1075 |  * Verify email OTP for sign-in
1076 |  * @param {string} email - User email address
1077 |  * @param {string} code - 6-digit verification code
1078 |  * @returns {Promise<boolean>} True if code is valid, false otherwise
1079 |  */
1080 | export const verifyEmailOTPForSignIn = async (email: string, code: string): Promise<boolean> => {
1081 |   try {
1082 |     if (!email || !code || code.length !== 6) {
1083 |       return false;
1084 |     }
1085 |
1086 |     const verificationRef = ref(database, `emailOTPSignIn/${email.replace(/[.#$[\]]/g, '_')}`);
1087 |     const snapshot = await get(verificationRef);
1088 |
1089 |     if (!snapshot.exists()) {
1090 |       return false;
1091 |     }
1092 |
1093 |     const verificationData = snapshot.val();
1094 |
1095 |     // Check if code expired
1096 |     if (Date.now() > verificationData.expiresAt) {
1097 |       // Clean up expired code
1098 |       await set(verificationRef, null);
1099 |       throw new Error("Verification code has expired. Please request a new one.");
1100 |     }
1101 |
1102 |     // Check attempt limit (max 5 attempts)
1103 |     if (verificationData.attempts >= 5) {
1104 |       await set(verificationRef, null);
1105 |       throw new Error("Too many failed attempts. Please request a new verification code.");
1106 |     }
1107 |
1108 |     // Verify code
1109 |     if (verificationData.code !== code) {
1110 |       // Increment attempts
1111 |       await set(verificationRef, {
1112 |         ...verificationData,
1113 |         attempts: verificationData.attempts + 1
1114 |       });
1115 |       return false;
1116 |     }
1117 |
1118 |     // Code is valid - mark as verified
1119 |     await set(verificationRef, {
1120 |       ...verificationData,
1121 |       verified: true,
1122 |       verifiedAt: Date.now()
1123 |     });
1124 |
1125 |     return true;
1126 |   } catch (error: any) {
1127 |     console.error("'L Error verifying OTP for sign-in:", error);
1128 |     throw error;
1129 |   }
1130 | };
1131 |
```

```
/**
 * Sign in with email OTP (passwordless)
 * After OTP verification, signs in user using email link authentication
 * @param {string} email - User email address
 * @returns {Promise<UserData>} User object with uid, email, displayName, and other user data
 */
export const signInWithEmailOTP = async (email: string): Promise<UserData> => {
  try {
    // Validate email
    if (!email || !email.includes('@')) {
      throw new Error("Please enter a valid email address");
    }

    // Check if OTP was verified
    const verificationRef = ref(database, `emailOTPSignIn/${email.replace(/[.#$[\]]/g, '_')}`);
    const snapshot = await get(verificationRef);

    if (!snapshot.exists() || !snapshot.val().verified) {
      throw new Error("Please verify your email address first by entering the 6-digit code sent to your email.");
    }

    // Check if user exists
    const emailExists = await checkEmailExists(email);

    if (!emailExists) {
      // User doesn't exist, they need to sign up
      throw new Error("No account found with this email. Please sign up first.");
    }

    // For existing users, we'll use email link authentication
    // This sends a sign-in link to their email
    console.log("Ø=Üç Sending sign-in link to:", email);

    const actionCodeSettings: ActionCodeSettings = {
      url: `${window.location.origin}/login?email=${encodeURIComponent(email)}`,
      handleCodeInApp: true,
    };

    // Import sendSignInLinkToEmail
    const { sendSignInLinkToEmail } = await import('firebase/auth');
    await sendSignInLinkToEmail(auth, email, actionCodeSettings);

    // Clean up OTP verification
    await set(verificationRef, null);

    // Note: User will need to click the link in their email to complete sign-in
    // This is a limitation of Firebase Auth - we can't sign in without password or link
    throw new Error("Please check your email and click the sign-in link to complete authentication.");
  } catch (error: any) {
    console.error("'L Error signing in with email OTP:", error);
    throw error;
  }
};

/**
 * Send password reset email
 * @param {string} email - User email address
 * @returns {Promise<void>}
 */
export const resetPassword = async (email: string): Promise<void> => {
  try {
    // Validate email
    if (!email || !email.includes('@')) {
      throw new Error("Please enter a valid email address");
    }

    console.log("Ø=Üç Sending password reset email to:", email);

    // Configure action code settings to redirect to our app's password reset page
    const actionCodeSettings: ActionCodeSettings = {
      url: `${window.location.origin}/reset-password`,
```

```
1203 |       handleCodeInApp: false, // Open link in browser, not app
1204 |     };
1205 |
1206 |     // Send password reset email with custom redirect URL
1207 |     await sendPasswordResetEmail(auth, email, actionCodeSettings);
1208 |
1209 |     console.log("'  Password reset email sent successfully");
1210 |   } catch (error: any) {
1211 |     console.error("'L Error sending password reset email:", error);
1212 |
1213 |     // Handle specific errors
1214 |     if (error.code === 'auth/user-not-found') {
1215 |       throw new Error("No account found with this email address.");
1216 |     }
1217 |     if (error.code === 'auth/invalid-email') {
1218 |       throw new Error("Invalid email address. Please check and try again.");
1219 |     }
1220 |
1221 |     throw error;
1222 |   }
1223 | };
1224 |
1225 | /**
1226 |  * Confirm password reset with the code from email link
1227 |  * @param {string} oobCode - The action code from the email link
1228 |  * @param {string} newPassword - The new password to set
1229 |  * @returns {Promise<void>}
1230 |  */
1231 | export const confirmPasswordResetCode = async (
1232 |   oobCode: string,
1233 |   newPassword: string
1234 | ): Promise<void> => {
1235 |   try {
1236 |     // Validate inputs
1237 |     if (!oobCode) {
1238 |       throw new Error("Invalid reset code. Please use the link from your email.");
1239 |     }
1240 |     if (!newPassword || newPassword.length < 6) {
1241 |       throw new Error("Password must be at least 6 characters long.");
1242 |     }
1243 |
1244 |     console.log("Ø=Ý  Confirming password reset...");
1245 |
1246 |     // Confirm the password reset using Firebase's confirmPasswordReset
1247 |     await confirmPasswordReset(auth, oobCode, newPassword);
1248 |
1249 |     console.log("'  Password reset successful!");
1250 |   } catch (error: any) {
1251 |     console.error("'L Error confirming password reset:", error);
1252 |
1253 |     // Handle specific errors
1254 |     if (error.code === 'auth/invalid-action-code') {
1255 |       throw new Error("Invalid or expired reset link. Please request a new password reset
email.");  }
1257 |     if (error.code === 'auth/expired-action-code') {
1258 |       throw new Error("This reset link has expired. Please request a new password reset email.");
1259 |     }
1260 |     if (error.code === 'auth/weak-password') {
1261 |       throw new Error("Password is too weak. Please choose a stronger password.");
1262 |     }
1263 |
1264 |     throw error;
1265 |   }
1266 | };
1267 |
1268 | /**
1269 |  * Initialize reCAPTCHA verifier for phone authentication
1270 |  * @returns {Promise<RecaptchaVerifier>} reCAPTCHA verifier instance
1271 |  */
1272 | let recaptchaVerifier: RecaptchaVerifier | null = null;
1273 |
```

```
/**
 * Clean up reCAPTCHA verifier and container completely
 * This should be called when user logs out to allow reCAPTCHA to be reinitialized
 */
export const cleanupRecaptcha = (): void => {
  console.log("Ø>Ýù Cleaning up reCAPTCHA...");

  // Clear the verifier instance
  if (recaptchaVerifier) {
    try {
      recaptchaVerifier.clear();
      console.log("'  reCAPTCHA verifier cleared");
    } catch (e) {
      console.warn("& þ  Error clearing reCAPTCHA verifier:", e);
    }
    recaptchaVerifier = null;
  }

  // Remove the container completely from DOM
  if (typeof document !== 'undefined') {
    const container = document.getElementById('recaptcha-container');
    if (container) {
      try {
        // Clear all content and attributes
        container.innerHTML = '';
        container.removeAttribute('data-sitekey');
        container.removeAttribute('data-callback');
        container.removeAttribute('data-size');

        // Remove from DOM
        if (container.parentElement) {
          container.parentElement.removeChild(container);
        }
        console.log("'  reCAPTCHA container removed from DOM");
      } catch (e) {
        console.warn("& þ  Error removing reCAPTCHA container:", e);
      }
    }

    // Also clean up any old containers that might exist
    const oldContainers = document.querySelectorAll('[id^="recaptcha-container"]');
    oldContainers.forEach((oldContainer) => {
      try {
        if (oldContainer.parentElement) {
          oldContainer.parentElement.removeChild(oldContainer);
        }
      } catch (e) {
        // Ignore errors
      }
    });
  }

  console.log("'  reCAPTCHA cleanup complete");
};

const getRecaptchaVerifier = async (): Promise<RecaptchaVerifier> => {
  // Always create a new verifier to avoid stale instances
  // Clear existing verifier if it exists
  if (recaptchaVerifier) {
    try {
      recaptchaVerifier.clear();
    } catch (e) {
      // Ignore clear errors
      console.warn("& þ  Error clearing existing reCAPTCHA verifier:", e);
    }
    recaptchaVerifier = null;
  }

  // Wait for DOM to be ready
  if (typeof document === 'undefined' || !document.body) {
    await new Promise<void>(resolve => {
```

```
1345 |       if (document.readyState === 'loading') {
1346 |         document.addEventListener('DOMContentLoaded', () => resolve(), { once: true });
1347 |       } else {
1348 |         resolve();
1349 |       }
1350 |     });
1351 |   }
1352 |
1353 |   // Ensure Firebase Auth is initialized
1354 |   if (!auth) {
1355 |     throw new Error("Firebase Auth is not initialized");
1356 |   }
1357 |
1358 |   // Get or create container element
1359 |   let container: HTMLElement | null = document.getElementById('recaptcha-container');
1360 |
1361 |   // CRITICAL: If container exists and has children or reCAPTCHA attributes, we need to remove
it completely
1362 |   // reCAPTCHA widgets can't be reused in the same container - must create a fresh one
1363 |   if (container && (container.children.length > 0 || container.hasAttribute('data-sitekey'))) {
1364 |     console.log("Ø=Ý  Removing existing reCAPTCHA container (has rendered widget)...");
1365 |     // Remove the container completely from DOM
1366 |     if (container.parentElement) {
1367 |       container.parentElement.removeChild(container);
1368 |     }
1369 |     // Clear the reference
1370 |     container = null;
1371 |     // Wait a bit for DOM cleanup
1372 |     await new Promise(resolve => setTimeout(resolve, 200));
1373 |   }
1374 |
1375 |   // If container doesn't exist, create it
1376 |   if (!container) {
1377 |     if (!document.body) {
1378 |       throw new Error("Document body not available");
1379 |     }
1380 |
1381 |     container = document.createElement('div');
1382 |     container.id = 'recaptcha-container';
1383 |     // Make it visible but tiny (required for reCAPTCHA)
1384 |     container.style.cssText = 'position: fixed; bottom: 0; right: 0; width: 1px; height: 1px;
overflow: hidden; z-index:
1385 |     document.body.appendChild(container);
1386 |
1387 |     // Wait for container to be in DOM
1388 |     await new Promise(resolve => setTimeout(resolve, 100));
1389 |   }
1390 |
1391 |   // Ensure container is in the DOM
1392 |   if (!container.parentElement) {
1393 |     if (!document.body) {
1394 |       throw new Error("Document body not available");
1395 |     }
1396 |     document.body.appendChild(container);
1397 |     await new Promise(resolve => setTimeout(resolve, 100));
1398 |   }
1399 |
1400 |   // Verify container is valid and empty
1401 |   if (!container || !container.parentElement) {
1402 |     throw new Error("reCAPTCHA container is not properly initialized");
1403 |   }
1404 |
1405 |   // Double-check container is empty (required for reCAPTCHA)
1406 |   if (container.children.length > 0) {
1407 |     console.warn("& þ  Container still has children, clearing again...");
1408 |     container.innerHTML = '';
1409 |     await new Promise(resolve => setTimeout(resolve, 100));
1410 |   }
1411 |
1412 |   // Wait for container to be fully ready
1413 |   await new Promise(resolve => {
1414 |     // Use requestAnimationFrame to ensure DOM is ready
1415 |     requestAnimationFrame(() => {
```

```
1416 |       setTimeout(resolve, 200);
1417 |     });
1418 |   });
1419 |
1420 |   try {
1421 |     // Create reCAPTCHA verifier with container
1422 |     // Container is required for proper initialization in many environments
1423 |     recaptchaVerifier = new RecaptchaVerifier(auth, container, {
1424 |       size: 'invisible',
1425 |       callback: () => {
1426 |         console.log("'  reCAPTCHA verified");
1427 |       },
1428 |       'expired-callback': () => {
1429 |         console.log("& þ  reCAPTCHA expired, resetting...");
1430 |         if (recaptchaVerifier) {
1431 |           try {
1432 |             recaptchaVerifier.clear();
1433 |           } catch (e) {
1434 |             // Ignore
1435 |           }
1436 |         }
1437 |         recaptchaVerifier = null;
1438 |       }
1439 |     });
1440 |
1441 |     console.log("'  reCAPTCHA verifier created successfully");
1442 |
1443 |     // For invisible reCAPTCHA, no explicit render() is needed
1444 |     // It will render automatically when used with signInWithPhoneNumber
1445 |
1446 |     return recaptchaVerifier;
1447 |   } catch (error: any) {
1448 |     console.error("'L Error creating reCAPTCHA verifier:", error);
1449 |     console.error("Error details:", {
1450 |       code: error.code,
1451 |       message: error.message,
1452 |       name: error.name,
1453 |       stack: error.stack
1454 |     });
1455 |
1456 |     // Handle "already rendered" error specifically
1457 |     const errorMsg = error.message?.toLowerCase() || '';
1458 |     if (errorMsg.includes('already been rendered') || errorMsg.includes('already rendered')) {
1459 |       console.log("Ø=Ý  reCAPTCHA already rendered - removing container and creating new one...");
1460 |
1461 |       // Remove the existing container completely
1462 |       if (container && container.parentElement) {
1463 |         container.parentElement.removeChild(container);
1464 |       }
1465 |
1466 |       // Create a fresh container with a unique ID
1467 |       const newContainerId = `recaptcha-container-${Date.now()}`;
1468 |       if (!document.body) {
1469 |         throw new Error("Document body not available");
1470 |       }
1471 |
1472 |       const newContainer = document.createElement('div');
1473 |       newContainer.id = newContainerId;
1474 |       newContainer.style.cssText = 'position: fixed; bottom: 0; right: 0; width: 1px; height:
1475 | 1px; overflow: hidden; z-i...
1476 |       document.body.appendChild(newContainer);
1477 |
1478 |       // Update the old container ID reference (for future cleanup)
1479 |       const oldContainer = document.getElementById('recaptcha-container');
1480 |       if (oldContainer) {
1481 |         oldContainer.id = `recaptcha-container-old-${Date.now()}`;
1482 |       }
1483 |
1484 |       // Wait for new container to be ready
1485 |       await new Promise(resolve => setTimeout(resolve, 200));
1486 |
1487 |       // Try again with the new container
```

```
1487 |       try {
1488 |         recaptchaVerifier = new RecaptchaVerifier(auth, newContainer, {
1489 |           size: 'invisible',
1490 |           callback: () => {
1491 |             console.log("'  reCAPTCHA verified (new container)");
1492 |           },
1493 |           'expired-callback': () => {
1494 |             console.log("& þ  reCAPTCHA expired, resetting...");
1495 |             if (recaptchaVerifier) {
1496 |               try {
1497 |                 recaptchaVerifier.clear();
1498 |               } catch (e) {
1499 |                 // Ignore
1500 |               }
1501 |             }
1502 |             recaptchaVerifier = null;
1503 |           }
1504 |         });
1505 |
1506 |         // Update the container ID back to standard for future use
1507 |         newContainer.id = 'recaptcha-container';
1508 |         console.log("'  reCAPTCHA verifier created successfully with new container");
1509 |         return recaptchaVerifier;
1510 |       } catch (retryError: any) {
1511 |         console.error("'L Retry with new container also failed:", retryError);
1512 |         // Continue to fallback
1513 |       }
1514 |     }
1515 |
1516 |     // Clean up on error
1517 |     if (recaptchaVerifier) {
1518 |       try {
1519 |         recaptchaVerifier.clear();
1520 |       } catch (e) {
1521 |         // Ignore
1522 |       }
1523 |     }
1524 |     recaptchaVerifier = null;
1525 |
1526 |     // Try alternative: without container (fallback)
1527 |     try {
1528 |       console.log("Ø=Ý  Trying reCAPTCHA without container as fallback...");
1529 |       recaptchaVerifier = new RecaptchaVerifier(auth, {
1530 |         size: 'invisible',
1531 |         callback: () => {
1532 |           console.log("'  reCAPTCHA verified (fallback)");
1533 |         },
1534 |         'expired-callback': () => {
1535 |           console.log("& þ  reCAPTCHA expired, resetting...");
1536 |           recaptchaVerifier = null;
1537 |         }
1538 |       });
1539 |
1540 |       console.log("'  reCAPTCHA verifier created (fallback method)");
1541 |       return recaptchaVerifier;
1542 |     } catch (fallbackError: any) {
1543 |       console.error("'L Fallback also failed:", fallbackError);
1544 |
1545 |       // Check if Firebase Auth is properly configured
1546 |       if (error.code === 'auth/operation-not-allowed') {
1547 |         throw new Error("Phone authentication is not enabled in Firebase. Please enable it in
Firebase Console.");
1548 |       }
1549 |
1550 |       // Provide helpful error message
1551 |       const errorMsgLower = error.message?.toLowerCase() || '';
1552 |       if (errorMsgLower.includes('container') || errorMsgLower.includes('haschildnodes') ||
errorMsgLower.includes('dom')) {  throw new Error("reCAPTCHA initialization failed. Please ensure Phone Authentication is
1553 | enabled in Firebase Cons...
1554 |       }
1555 |
1556 |       if (errorMsgLower.includes('already been rendered') || errorMsgLower.includes('already
rendered')) {  throw new Error("reCAPTCHA was already rendered. Please refresh the page and try
1557 | again.");
```

```
      }

      throw new Error(`reCAPTCHA initialization failed: ${error.message || 'Unknown error'}. Please check Firebase Conso...
    }
  }
};

/**
 * Send SMS verification code to phone number
 * @param {string} phoneNumber - Phone number in E.164 format (e.g., +1234567890)
 * @returns {Promise<ConfirmationResult>} Confirmation result containing verification ID
 */
export const sendPhoneVerificationCode = async (phoneNumber: string): Promise<ConfirmationResult> => {
  try {
    // Validate phone number format (should start with +)
    if (!phoneNumber.startsWith('+')) {
      throw new Error("Phone number must include country code (e.g., +1234567890)");
    }

    // Get or create reCAPTCHA verifier (async now)
    const verifier = await getRecaptchaVerifier();

    console.log("Ø=Üñ Sending SMS verification code to:", phoneNumber);
    console.log("Ø=Üñ Phone number format check:", {
      startsWithPlus: phoneNumber.startsWith('+'),
      length: phoneNumber.length,
      formatted: phoneNumber
    });

    // Send verification code
    const confirmationResult = await signInWithPhoneNumber(auth, phoneNumber, verifier);

    console.log("'  SMS code sent successfully");
    console.log("'  Confirmation result:", {
      verificationId: confirmationResult?.verificationId ? "Present" : "Missing"
    });

    // Additional diagnostic info
    if (confirmationResult?.verificationId) {
      console.log("'  Verification ID received - SMS should be on its way!");
      console.log("Ø=Üñ Check your phone for the SMS code");
      console.log("& þ  Production Mode Check:");
      console.log("   If SMS doesn't arrive, you may be in TEST MODE.");
      console.log("   To enable PRODUCTION MODE (send SMS to any number):");
      console.log("   1. Go to Firebase Console > Authentication > Settings > Phone numbers for testing");
      console.log("   2. REMOVE all test phone numbers from the list");
      console.log("   3. Production mode activates automatically when test list is empty (requires billing enabled)");
      console.log("   4. Verify billing is enabled: Firebase Console > Usage & Billing");
      console.log("   5. Ensure phone number format is correct (+63XXXXXXXXXX)");
      console.log("   Ø=ÜÖ See PHONE_AUTH_PRODUCTION_SETUP.md for detailed instructions");
    }

    return confirmationResult;
  } catch (error: any) {
    console.error("'L Error sending SMS code:", error);
    console.error("Error code:", error.code);
    console.error("Error message:", error.message);
    console.error("Full error:", JSON.stringify(error, null, 2));

    // Reset verifier on error
    if (recaptchaVerifier) {
      try {
        recaptchaVerifier.clear();
      } catch (e) {
        // Ignore clear errors
      }
      recaptchaVerifier = null;
    }

    // Handle specific errors with more detailed messages
    if (error.code === 'auth/billing-not-enabled') {
```

```
      throw new Error("Billing error detected. Please verify: 1) You're on the Blaze plan (not just Spark). 2) Phone Aut...
    }
    if (error.code === 'auth/invalid-app-credential') {
      // Run validation to get more diagnostic info
      console.error("Ø=Ý  Running Firebase configuration validation...");
      const validation = await validateFirebaseConfig();

      let errorMessage = "Invalid app credentials detected. ";

      // Add specific validation errors if found
      if (validation.errors.length > 0) {
        errorMessage += `\n\n'L Configuration Errors:\n${validation.errors.map(e => `• ${e}`).join('\n')}`;
      }

      if (validation.warnings.length > 0) {
        errorMessage += `\n\n& þ  Configuration Warnings:\n${validation.warnings.map(w => `• ${w}`).join('\n')}`;
      }

      // If validation passed, configuration looks correct - likely billing propagation delay
      if (validation.valid && validation.errors.length === 0) {
        errorMessage += "\n\n'  Configuration validation passed - your Firebase config looks correct.";
        errorMessage += "\n\n#ð Most Likely Cause: Billing/Blaze plan activation delay";
        errorMessage += "\n\nThis error usually means billing changes are still propagating. Even if everything is enabl...
        errorMessage += "\n\nØ=Ý  What to Check:\n";
        errorMessage += "• When did you enable billing? If less than 15-30 minutes ago, wait longer\n";
        errorMessage += "• Go to Firebase Console > Usage & Billing and verify Blaze plan shows as 'Active'\n";
        errorMessage += "• Check Firebase Console > Authentication > Sign-in method > Phone - ensure it's enabled\n";
        errorMessage += "• Try again in 5-10 minutes after waiting\n";
      }

      errorMessage += "\n\nØ=ÜË Complete Troubleshooting Steps:\n";
      errorMessage += "1. #ñþ  Wait 15-30 minutes if you just enabled billing/Blaze plan (activation takes time)\n";
      errorMessage += "2. '  Verify Phone Authentication: Firebase Console > Authentication > Sign-in method > Phone > En...
      errorMessage += "3. Ø=Ý  Verify Firebase config: Firebase Console > Project Settings > Your apps > Check config matc...
      errorMessage += "4. Ø=Ü³ Verify billing status: Firebase Console > Usage & Billing > Blaze plan must show 'Active'\n";
      errorMessage += "5. Ø=Üñ Check authorized domains: Firebase Console > Authentication > Settings > Authorized domains...
      errorMessage += "6. Ø=Ý  Hard refresh: Clear cache (Cmd+Shift+R or Ctrl+Shift+R) and try again\n";
      errorMessage += "\nØ=ÜÖ For detailed steps, see: INVALID_APP_CREDENTIALS_TROUBLESHOOTING.md\n";

      throw new Error(errorMessage);
    }
    if (error.code === 'auth/invalid-phone-number') {
      throw new Error("Invalid phone number. Please check the format (include country code, e.g., +1234567890).");
    }
    if (error.code === 'auth/too-many-requests') {
      throw new Error("Too many requests. Please wait a few minutes before trying again.");
    }
    if (error.code === 'auth/quota-exceeded') {
      throw new Error("SMS quota exceeded. Please try again later or check Firebase Console > Usage & Billing for quota ...
    }
    if (error.code === 'auth/operation-not-allowed') {
      throw new Error("Phone authentication is not enabled. Please enable Phone Authentication in Firebase Console under...
    }
    if (error.code === 'auth/captcha-check-failed') {
      throw new Error("reCAPTCHA verification failed. Please refresh the page and try again.");
    }
    if (error.code === 'auth/missing-phone-number') {
      throw new Error("Phone number is required. Please enter your phone number.");
    }
    if (error.code === 'auth/internal-error') {
      // Provide more context about internal errors
      const errorMsg = error.message?.toLowerCase() || '';
      if (errorMsg.includes('recaptcha') || errorMsg.includes('captcha')) {
        throw new Error("reCAPTCHA error. Please ensure Phone Authentication is enabled in Firebase Console and refresh ...
      }
      if (errorMsg.includes('app verification') || errorMsg.includes('app check')) {
        throw new Error("App verification failed. Please check Firebase Console configuration.");
      }
      // Check if this might be a test mode restriction (SMS not arriving)
      const errorMsgLower = error.message?.toLowerCase() || '';
      if (!errorMsgLower.includes('recaptcha') && !errorMsgLower.includes('captcha')) {
```

```
        throw new Error(`Authentication error: ${error.message || 'Unknown error'}. If SMS code
was sent but not receive...
      }
      throw new Error(`Authentication error: ${error.message || 'Unknown error'}. Please check
that Phone Authentication...
    }

    // Handle reCAPTCHA-related errors
    if (error.message?.includes('recaptcha') || error.message?.includes('captcha')) {
      throw new Error("reCAPTCHA error. Please refresh the page and ensure Phone Authentication
is enabled in Firebase C...
    }

    // Re-throw with user-friendly message if it has a message
    if (error.message && typeof error.message === 'string') {
      // Check if it's already a user-friendly message
      if (error.message.includes('Please') || error.message.includes('must')) {
        throw error;
      }
      throw new Error(`Failed to send verification code: ${error.message}`);
    }

    // Generic error message with production mode guidance
    throw new Error("Failed to send verification code. If SMS code was sent but not received,
check Firebase Console > A...
  }
};

/**
 * Verify SMS code and sign in user
 * @param {ConfirmationResult} confirmationResult - Confirmation result from
sendPhoneVerificationCode
 * @param {string} code - 6-digit verification code from SMS
 * @returns {Promise<UserData>} User object with uid, phoneNumber, and other user data
 */
export const verifyPhoneCode = async (
  confirmationResult: ConfirmationResult,
  code: string
): Promise<UserData> => {
  try {
    // Validate code format (should be 6 digits)
    if (!/^\d{6}$/.test(code)) {
      throw new Error("Verification code must be 6 digits");
    }

    console.log("Ø=Ý  Verifying SMS code...");

    // Verify the code
    const result = await confirmationResult.confirm(code);
    const user = result.user;

    console.log("'  Phone verification successful! User:", user.uid);

    // Save user to Firebase Realtime Database
    await saveUserToDatabase(user);

    return {
      uid: user.uid,
      displayName: user.displayName,
      email: user.email,
      phoneNumber: user.phoneNumber,
      photoURL: user.photoURL
    };
  } catch (error: any) {
    console.error("'L Error verifying SMS code:", error);

    // Handle specific errors
    if (error.code === 'auth/invalid-verification-code') {
      throw new Error("Invalid verification code. Please check the code and try again.");
    }
    if (error.code === 'auth/code-expired') {
      throw new Error("Verification code has expired. Please request a new code.");
    }
    if (error.code === 'auth/session-expired') {
      throw new Error("Session expired. Please start the verification process again.");
    }
```

```
1771 |
1772 |     throw error;
1773 |   }
1774 | };
1775 |
1776 |
```

File: src/services/challengeService.ts

Lines: 424

```
// Challenge service - manages challenge zones, participation, and leaderboards
import { ref, set, get, onValue, off, remove, DataSnapshot, push, runTransaction } from "firebase/database";
import { database } from "./firebase";
import { calculateDistance } from "@/utils/distance";

export interface ChallengeZone {
  id: string;
  name: string;
  description: string;
  lat: number;
  lng: number;
  radius: number; // in meters
  points: number; // points awarded per workout (default 10)
  active: boolean;
  visible: boolean;
  createdAt: number;
}

export interface ChallengeParticipation {
  userId: string;
  zoneId: string;
  date: string; // YYYY-MM-DD format
  workoutId: string;
  timestamp: number;
  points: number;
}

export interface ChallengeLeaderboardEntry {
  userId: string;
  zoneId: string;
  totalPoints: number;
  workoutsCount: number;
  lastWorkoutDate: string; // YYYY-MM-DD format
}

/**
 * Create a new challenge zone (admin only)
 */
export const createChallengeZone = async (
  zone: Omit<ChallengeZone, "id" | "createdAt">
): Promise<string> => {
  try {
    const zonesRef = ref(database, "challengeZones");
    const newZoneRef = push(zonesRef);
    const zoneId = newZoneRef.key!;

    const zoneData: ChallengeZone = {
      ...zone,
      id: zoneId,
      createdAt: Date.now(),
    };

    await set(newZoneRef, zoneData);
    console.log(`'  Challenge zone created: ${zoneId}`);
    return zoneId;
  } catch (error) {
    console.error("'L Error creating challenge zone:", error);
    throw error;
  }
};

/**
 * Update a challenge zone (admin only)
 */
export const updateChallengeZone = async (
  zoneId: string,
```

```
  updates: Partial<Omit<ChallengeZone, "id" | "createdAt">>
): Promise<void> => {
  try {
    const zoneRef = ref(database, `challengeZones/${zoneId}`);
    const snapshot = await get(zoneRef);

    if (!snapshot.exists()) {
      throw new Error("Challenge zone not found");
    }

    const currentData = snapshot.val() as ChallengeZone;
    const updatedData = {
      ...currentData,
      ...updates,
    };

    await set(zoneRef, updatedData);
    console.log(`'  Challenge zone updated: ${zoneId}`);
  } catch (error) {
    console.error("'L Error updating challenge zone:", error);
    throw error;
  }
};

/**
 * Delete a challenge zone (admin only)
 */
export const deleteChallengeZone = async (zoneId: string): Promise<void> => {
  try {
    const zoneRef = ref(database, `challengeZones/${zoneId}`);
    await remove(zoneRef);
    console.log(`'  Challenge zone deleted: ${zoneId}`);
  } catch (error) {
    console.error("'L Error deleting challenge zone:", error);
    throw error;
  }
};

/**
 * Get all challenge zones
 */
export const getAllChallengeZones = async (): Promise<ChallengeZone[]> => {
  try {
    const zonesRef = ref(database, "challengeZones");
    const snapshot = await get(zonesRef);

    if (!snapshot.exists()) {
      return [];
    }

    const zones = snapshot.val();
    return Object.values(zones) as ChallengeZone[];
  } catch (error) {
    console.error("'L Error getting challenge zones:", error);
    throw error;
  }
};

/**
 * Get active and visible challenge zones
 */
export const getActiveChallengeZones = async (): Promise<ChallengeZone[]> => {
  try {
    const zones = await getAllChallengeZones();
    return zones.filter((zone) => zone.active && zone.visible);
  } catch (error) {
    console.error("'L Error getting active challenge zones:", error);
    throw error;
  }
};

```

```
138 | /**
139 |  * Listen to challenge zones in real-time
140 |  */
141 | export const listenToChallengeZones = (
142 |   callback: (zones: ChallengeZone[]) => void
143 | ): (() => void) => {
144 |   const zonesRef = ref(database, "challengeZones");
145 |
146 |   const unsubscribe = onValue(
147 |     zonesRef,
148 |     (snapshot: DataSnapshot) => {
149 |       if (!snapshot.exists()) {
150 |         callback([]);
151 |         return;
152 |       }
153 |
154 |       const zones = snapshot.val();
155 |       const zoneList = Object.values(zones) as ChallengeZone[];
156 |       callback(zoneList);
157 |     },
158 |     (error) => {
159 |       console.error("'L Error listening to challenge zones:", error);
160 |       callback([]);
161 |     }
162 |   );
163 |
164 |   return () => {
165 |     off(zonesRef);
166 |   };
167 | };
168 |
169 | /**
170 |  * Check if a user location is within a challenge zone
171 |  */
172 | export const checkUserInZone = (
173 |   userLat: number,
174 |   userLng: number,
175 |   zone: ChallengeZone
176 | ): boolean => {
177 |   if (!userLat || !userLng) {
178 |     return false;
179 |   }
180 |
181 |   const distanceKm = calculateDistance(userLat, userLng, zone.lat, zone.lng);
182 |   if (distanceKm === null) {
183 |     return false;
184 |   }
185 |
186 |   const distanceMeters = distanceKm * 1000;
187 |   return distanceMeters <= zone.radius;
188 | };
189 |
190 | /**
191 |  * Get today's date string in YYYY-MM-DD format
192 |  */
193 | const getTodayDateString = (): string => {
194 |   const today = new Date();
195 |   const year = today.getFullYear();
196 |   const month = String(today.getMonth() + 1).padStart(2, "0");
197 |   const day = String(today.getDate()).padStart(2, "0");
198 |   return `${year}-${month}-${day}`;
199 | };
200 |
201 | /**
202 |  * Check if user has already earned points today for a zone
203 |  */
204 | export const hasEarnedPointsToday = async (
205 |   userId: string,
206 |   zoneId: string
207 | ): Promise<boolean> => {
208 |   try {
```

```
209 |     const today = getTodayDateString();
210 |     const participationRef = ref(
211 |       database,
212 |       `challengeParticipation/${userId}/${zoneId}/${today}`
213 |     );
214 |     const snapshot = await get(participationRef);
215 |     return snapshot.exists();
216 |   } catch (error) {
217 |     console.error("'L Error checking today's participation:", error);
218 |     return false;
219 |   }
220 | };
221 |
222 | /**
223 |  * Award challenge points to a user (once per day per zone)
224 |  * Uses Firebase transaction to prevent duplicate awards
225 |  */
226 | export const awardChallengePoints = async (
227 |   userId: string,
228 |   zoneId: string,
229 |   workoutId: string,
230 |   zone: ChallengeZone
231 | ): Promise<boolean> => {
232 |   try {
233 |     const today = getTodayDateString();
234 |
235 |     // Check if already earned today
236 |     const alreadyEarned = await hasEarnedPointsToday(userId, zoneId);
237 |     if (alreadyEarned) {
238 |       console.log(
239 |         `#íþ  User ${userId} already earned points for zone ${zoneId} today`
240 |       );
241 |       return false;
242 |     }
243 |
244 |     // Use transaction to prevent race conditions
245 |     const participationRef = ref(
246 |       database,
247 |       `challengeParticipation/${userId}/${zoneId}/${today}`
248 |     );
249 |
250 |     await runTransaction(participationRef, (currentData) => {
251 |       // If data already exists, don't award again
252 |       if (currentData) {
253 |         return currentData;
254 |       }
255 |
256 |       // Create participation record
257 |       const participation: ChallengeParticipation = {
258 |         userId,
259 |         zoneId,
260 |         date: today,
261 |         workoutId,
262 |         timestamp: Date.now(),
263 |         points: zone.points,
264 |       };
265 |
266 |       return participation;
267 |     });
268 |
269 |     // Update leaderboard
270 |     await updateLeaderboard(userId, zoneId, zone.points, today);
271 |
272 |     console.log(
273 |       `'  Awarded ${zone.points} points to user ${userId} for zone ${zoneId}`
274 |     );
275 |     return true;
276 |   } catch (error) {
277 |     console.error("'L Error awarding challenge points:", error);
278 |     throw error;
279 |   }
```

```
};

/**
 * Update leaderboard for a zone
 */
const updateLeaderboard = async (
  userId: string,
  zoneId: string,
  points: number,
  date: string
): Promise<void> => {
  try {
    const leaderboardRef = ref(
      database,
      `challengeLeaderboards/${zoneId}/${userId}`
    );

    await runTransaction(leaderboardRef, (currentData) => {
      if (currentData) {
        // Update existing entry
        return {
          ...currentData,
          totalPoints: (currentData.totalPoints || 0) + points,
          workoutsCount: (currentData.workoutsCount || 0) + 1,
          lastWorkoutDate: date,
        };
      } else {
        // Create new entry
        return {
          userId,
          zoneId,
          totalPoints: points,
          workoutsCount: 1,
          lastWorkoutDate: date,
        };
      }
    });
  } catch (error) {
    console.error("'L Error updating leaderboard:", error);
    // Don't throw - leaderboard update failure shouldn't prevent points award
  }
};

/**
 * Get user's challenge stats for a zone
 */
export const getUserChallengeStats = async (
  userId: string,
  zoneId: string
): Promise<{
  totalPoints: number;
  workoutsCount: number;
  lastWorkoutDate: string | null;
  earnedToday: boolean;
}> => {
  try {
    const leaderboardRef = ref(
      database,
      `challengeLeaderboards/${zoneId}/${userId}`
    );
    const snapshot = await get(leaderboardRef);

    const earnedToday = await hasEarnedPointsToday(userId, zoneId);

    if (!snapshot.exists()) {
      return {
        totalPoints: 0,
        workoutsCount: 0,
        lastWorkoutDate: null,
        earnedToday,
      };
```

```
    }

    const data = snapshot.val() as ChallengeLeaderboardEntry;
    return {
      totalPoints: data.totalPoints || 0,
      workoutsCount: data.workoutsCount || 0,
      lastWorkoutDate: data.lastWorkoutDate || null,
      earnedToday,
    };
  } catch (error) {
    console.error("'L Error getting user challenge stats:", error);
    return {
      totalPoints: 0,
      workoutsCount: 0,
      lastWorkoutDate: null,
      earnedToday: false,
    };
  }
};

/**
 * Get leaderboard for a zone (top N users)
 */
export const getZoneLeaderboard = async (
  zoneId: string,
  limit: number = 20
): Promise<ChallengeLeaderboardEntry[]> => {
  try {
    const leaderboardRef = ref(database, `challengeLeaderboards/${zoneId}`);
    const snapshot = await get(leaderboardRef);

    if (!snapshot.exists()) {
      return [];
    }

    const entries = snapshot.val();
    const leaderboard = Object.values(entries) as ChallengeLeaderboardEntry[];

    // Sort by totalPoints (descending), then by workoutsCount (descending)
    leaderboard.sort((a, b) => {
      if (b.totalPoints !== a.totalPoints) {
        return b.totalPoints - a.totalPoints;
      }
      return b.workoutsCount - a.workoutsCount;
    });

    return leaderboard.slice(0, limit);
  } catch (error) {
    console.error("'L Error getting zone leaderboard:", error);
    return [];
  }
};

/**
 * Get all zones a user is currently in
 */
export const getZonesUserIsIn = (
  userLat: number,
  userLng: number,
  zones: ChallengeZone[]
): ChallengeZone[] => {
  if (!userLat || !userLng) {
    return [];
  }

  return zones.filter((zone) => {
    if (!zone.active) {
      return false;
    }
    return checkUserInZone(userLat, userLng, zone);
  });
```

```
422 | };
423 |
424 |
```

File: src/services/checkInService.ts

Lines: 233

```
   1 | // Check-in service for Firebase - manages venue check-ins
   2 | import { ref, set, get, onValue, off, remove, DataSnapshot, query, orderByChild, equalTo } from
"firebase/database";
   3 | import { database } from "./firebase";
   4 |
   5 | export type Activity = "running" | "cycling" | "walking" | "others";
   6 |
   7 | export interface CheckIn {
   8 |   userId: string;
   9 |   venueId: string;
  10 |   venueName: string;
  11 |   activity: Activity;
  12 |   userName: string;
  13 |   userAvatar: string;
  14 |   timestamp: number;
  15 | }
  16 |
  17 | export interface UserCheckInData {
  18 |   userId: string;
  19 |   userName: string;
  20 |   userAvatar: string;
  21 |   activity: Activity;
  22 | }
  23 |
  24 | export interface VenueData {
  25 |   id: string;
  26 |   name: string;
  27 | }
  28 |
  29 | /**
  30 |  * Check in to a venue
  31 |  */
  32 | export const checkInToVenue = async (
  33 |   userId: string,
  34 |   venueId: string,
  35 |   venueData: VenueData,
  36 |   userData: UserCheckInData
  37 | ): Promise<void> => {
  38 |   try {
  39 |     // First, check out from any previous venue
  40 |     const currentCheckIn = await getUserCheckIn(userId);
  41 |     if (currentCheckIn) {
  42 |       await checkOutFromVenue(userId, currentCheckIn.venueId);
  43 |     }
  44 |
  45 |     // Create check-in record
  46 |     const checkInRef = ref(database, `checkIns/${venueId}/${userId}`);
  47 |     const checkIn: CheckIn = {
  48 |       userId,
  49 |       venueId,
  50 |       venueName: venueData.name,
  51 |       activity: userData.activity,
  52 |       userName: userData.userName,
  53 |       userAvatar: userData.userAvatar || "",
  54 |       timestamp: Date.now()
  55 |     };
  56 |
  57 |     await set(checkInRef, checkIn);
  58 |     console.log(`'  User ${userId} checked in to ${venueData.name}`);
  59 |   } catch (error) {
  60 |     console.error("'L Error checking in to venue:", error);
  61 |     throw error;
  62 |   }
  63 | };
  64 |
  65 | /**
  66 |  * Check out from a venue
```

```
 67 |  */
 68 | export const checkOutFromVenue = async (
 69 |   userId: string,
 70 |   venueId: string
 71 | ): Promise<void> => {
 72 |   try {
 73 |     const checkInRef = ref(database, `checkIns/${venueId}/${userId}`);
 74 |     await remove(checkInRef);
 75 |     console.log(`'  User ${userId} checked out from venue ${venueId}`);
 76 |   } catch (error) {
 77 |     console.error("'L Error checking out from venue:", error);
 78 |     throw error;
 79 |   }
 80 | };
 81 |
 82 | /**
 83 |  * Get all check-ins at a specific venue
 84 |  */
 85 | export const getCheckInsAtVenue = async (venueId: string): Promise<CheckIn[]> => {
 86 |   try {
 87 |     const venueCheckInsRef = ref(database, `checkIns/${venueId}`);
 88 |     const snapshot = await get(venueCheckInsRef);
 89 |
 90 |     if (!snapshot.exists()) {
 91 |       return [];
 92 |     }
 93 |
 94 |     const checkIns = snapshot.val();
 95 |     return Object.values(checkIns) as CheckIn[];
 96 |   } catch (error) {
 97 |     console.error("'L Error getting check-ins at venue:", error);
 98 |     return [];
 99 |   }
100 | };
101 |
102 | /**
103 |  * Listen to check-ins at a venue in real-time
104 |  */
105 | export const listenToVenueCheckIns = (
106 |   venueId: string,
107 |   callback: (checkIns: CheckIn[]) => void
108 | ): (() => void) => {
109 |   const venueCheckInsRef = ref(database, `checkIns/${venueId}`);
110 |
111 |   const unsubscribe = onValue(
112 |     venueCheckInsRef,
113 |     (snapshot: DataSnapshot) => {
114 |       if (!snapshot.exists()) {
115 |         callback([]);
116 |         return;
117 |       }
118 |
119 |       const checkIns = snapshot.val();
120 |       callback(Object.values(checkIns) as CheckIn[]);
121 |     },
122 |     (error) => {
123 |       console.error("'L Error listening to venue check-ins:", error);
124 |       callback([]);
125 |     }
126 |   );
127 |
128 |   return () => {
129 |     off(venueCheckInsRef);
130 |   };
131 | };
132 |
133 | /**
134 |  * Get user's current check-in
135 |  */
136 | export const getUserCheckIn = async (userId: string): Promise<CheckIn | null> => {
137 |   try {
```

```
138 |     const checkInsRef = ref(database, "checkIns");
139 |     const snapshot = await get(checkInsRef);
140 |
141 |     if (!snapshot.exists()) {
142 |       return null;
143 |     }
144 |
145 |     const allCheckIns = snapshot.val();
146 |
147 |     // Search through all venues for this user's check-in
148 |     for (const venueId in allCheckIns) {
149 |       const venueCheckIns = allCheckIns[venueId];
150 |       if (venueCheckIns[userId]) {
151 |         return venueCheckIns[userId] as CheckIn;
152 |       }
153 |     }
154 |
155 |     return null;
156 |   } catch (error) {
157 |     console.error("'L Error getting user check-in:", error);
158 |     return null;
159 |   }
160 | };
161 |
162 | /**
163 |  * Listen to user's current check-in in real-time
164 |  */
165 | export const listenToUserCheckIn = (
166 |   userId: string,
167 |   callback: (checkIn: CheckIn | null) => void
168 | ): (() => void) => {
169 |   const checkInsRef = ref(database, "checkIns");
170 |
171 |   const unsubscribe = onValue(
172 |     checkInsRef,
173 |     (snapshot: DataSnapshot) => {
174 |       if (!snapshot.exists()) {
175 |         callback(null);
176 |         return;
177 |       }
178 |
179 |       const allCheckIns = snapshot.val();
180 |
181 |       // Search through all venues for this user's check-in
182 |       for (const venueId in allCheckIns) {
183 |         const venueCheckIns = allCheckIns[venueId];
184 |         if (venueCheckIns[userId]) {
185 |           callback(venueCheckIns[userId] as CheckIn);
186 |           return;
187 |         }
188 |       }
189 |
190 |       callback(null);
191 |     },
192 |     (error) => {
193 |       console.error("'L Error listening to user check-in:", error);
194 |       callback(null);
195 |     }
196 |   );
197 |
198 |   return () => {
199 |     off(checkInsRef);
200 |   };
201 | };
202 |
203 | /**
204 |  * Get all active check-ins across all venues
205 |  */
206 | export const getAllActiveCheckIns = async (): Promise<CheckIn[]> => {
207 |   try {
208 |     const checkInsRef = ref(database, "checkIns");
```

```
209 |     const snapshot = await get(checkInsRef);
210 |
211 |     if (!snapshot.exists()) {
212 |       return [];
213 |     }
214 |
215 |     const allCheckIns = snapshot.val();
216 |     const checkInsList: CheckIn[] = [];
217 |
218 |     // Flatten all check-ins from all venues
219 |     for (const venueId in allCheckIns) {
220 |       const venueCheckIns = allCheckIns[venueId];
221 |       for (const userId in venueCheckIns) {
222 |         checkInsList.push(venueCheckIns[userId] as CheckIn);
223 |       }
224 |     }
225 |
226 |     return checkInsList;
227 |   } catch (error) {
228 |     console.error("'L Error getting all active check-ins:", error);
229 |     return [];
230 |   }
231 | };
232 |
233 |
```

File: src/services/emailService.ts

Lines: 105

```
/**
 * Email Service for sending OTP codes via SendGrid
 *
 * IMPORTANT SECURITY NOTE:
 * For production, API keys should NEVER be in frontend code.
 * This service is designed to call a backend endpoint that handles SendGrid.
 *
 * For a simple research prototype, see EmailJS option in SENDGRID_EMAIL_OTP_SETUP.md
 */

interface EmailOptions {
  to: string;
  subject: string;
  text: string;
  html: string;
}

/**
 * Send email via SendGrid API
 * This should call a backend endpoint, not SendGrid directly from frontend
 *
 * @param options Email options
 * @returns Promise<boolean> True if email sent successfully
 */
export const sendEmailViaSendGrid = async (options: EmailOptions): Promise<boolean> => {
  try {
    // Option 1: Call backend endpoint (recommended)
    const backendUrl = import.meta.env.VITE_EMAIL_API_URL || 'http://localhost:3001';

    const response = await fetch(`${backendUrl}/api/send-email`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(options),
    });

    if (!response.ok) {
      const error = await response.json();
      throw new Error(error.message || 'Failed to send email');
    }

    return true;
  } catch (error: any) {
    console.error(''L Error sending email via SendGrid:', error);
    throw new Error(`Failed to send email: ${error.message || 'Unknown error'}`);
  }
};

/**
 * Send OTP verification email
 *
 * @param email User email address
 * @param code 6-digit verification code
 * @returns Promise<boolean> True if email sent successfully
 */
export const sendOTPEmail = async (email: string, code: string): Promise<boolean> => {
  const subject = 'PaceMatch - Email Verification Code';
  const text = `Your PaceMatch verification code is: ${code}\n\nThis code expires in 10 minutes.
\n\nIf you didn't reques...
  const html = `
    <!DOCTYPE html>
    <html>
    <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
      <title>PaceMatch Verification Code</title>
    </head>
    <body style="font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif; max-width: 600px; margin: 0 auto; padding: 20px;">
      <div style="background: linear-gradient(135deg, #1976d2 0%, #42a5f5 100%); padding: 30px; text-align: center; border-radius: 10px 10px 0 0;">
        <h1 style="color: white; margin: 0; font-size: 28px;">PaceMatch</h1>
      </div>
      <div style="background: #f5f5f5; padding: 30px; border-radius: 0 0 10px 10px;">
        <h2 style="color: #1976d2; margin-top: 0;">Email Verification</h2>
        <p>Hi there,</p>
        <p>Thank you for signing up for PaceMatch! Use the code below to verify your email address:</p>
        <div style="background: white; border: 2px dashed #1976d2; border-radius: 8px; padding: 20px; text-align: center; margin: 20px 0;">
          <div style="font-size: 36px; font-weight: bold; letter-spacing: 8px; color: #1976d2; font-family: 'Courier New', monospace;">
            ${code}
          </div>
        </div>

        <p style="color: #666; font-size: 14px;">This code will expire in <strong>10 minutes</strong>.</p>
        <p style="color: #666; font-size: 14px;">If you didn't request this code, please ignore this email.</p>

        <hr style="border: none; border-top: 1px solid #ddd; margin: 30px 0;">

        <p style="color: #999; font-size: 12px; text-align: center; margin: 0;">
          This is an automated email from PaceMatch. Please do not reply to this email.
        </p>
      </div>
    </body>
    </html>
  `;

  return await sendEmailViaSendGrid({
    to: email,
    subject,
    text,
    html,
  });
};

```

File: src/services/emailServiceSimple.ts

Lines: 67

```
/**
 * Simple Email Service using EmailJS (Frontend-only solution)
 *
 * This is the EASIEST option for a research prototype - no backend needed!
 * EmailJS handles email sending from the frontend securely.
 *
 * Setup:
 * 1. Sign up at https://www.emailjs.com/ (free tier available)
 * 2. Connect your email service (Gmail, Outlook, etc.)
 * 3. Create email template
 * 4. Add environment variables (see .env.example)
 *
 * Alternative: See emailService.ts for SendGrid backend solution
 */

import emailjs from '@emailjs/browser';

// Initialize EmailJS (call this once in your app, e.g., in main.tsx)
export const initEmailJS = () => {
  const publicKey = import.meta.env.VITE_EMAILJS_PUBLIC_KEY;
  if (publicKey) {
    emailjs.init(publicKey);
  } else {
    console.warn('& þ  EmailJS public key not found. Email sending will not work.');
  }
};

interface SendOTPEmailOptions {
  email: string;
  code: string;
  userName?: string;
}

/**
 * Send OTP verification email via EmailJS
 *
 * @param options Email options
 * @returns Promise<boolean> True if email sent successfully
 */
export const sendOTPEmailSimple = async (options: SendOTPEmailOptions): Promise<boolean> => {
  try {
    const serviceId = import.meta.env.VITE_EMAILJS_SERVICE_ID;
    const templateId = import.meta.env.VITE_EMAILJS_TEMPLATE_ID;
    const publicKey = import.meta.env.VITE_EMAILJS_PUBLIC_KEY;

    if (!serviceId || !templateId || !publicKey) {
      throw new Error('EmailJS configuration missing. Please check your environment variables.');
    }

    const templateParams = {
      to_email: options.email,
      verification_code: options.code,
      user_name: options.userName || 'there',
      app_name: 'PaceMatch',
    };

    await emailjs.send(serviceId, templateId, templateParams, publicKey);

    console.log('  Email sent successfully via EmailJS');
    return true;
  } catch (error: any) {
    console.error('L Error sending email via EmailJS:', error);
    throw new Error(`Failed to send email: ${error.text || error.message || 'Unknown error'}`);
  }
};

```

File: src/services/encounteredUsersService.ts

Lines: 165

```
  1 | // Encountered users service - tracks users you've encountered within discovery radius
  2 | import { ref, set, get, onValue, off, remove, DataSnapshot } from "firebase/database";
  3 | import { database } from "./firebase";
  4 |
  5 | export interface EncounteredUser {
  6 |   userId: string;
  7 |   encounteredAt: number; // First encounter timestamp
  8 |   lastSeenAt: number; // Most recent encounter timestamp
  9 |   distance: number; // Last known distance in km
 10 |   lat: number; // Last known location
 11 |   lng: number; // Last known location
 12 |   count: number; // Total number of encounters
 13 | }
 14 |
 15 | /**
 16 |  * Add or update an encountered user record
 17 |  * @param userId - Current user's ID
 18 |  * @param encounteredUserId - ID of the user encountered
 19 |  * @param distance - Distance in km when encountered
 20 |  * @param location - Location where encounter happened {lat, lng}
 21 |  */
 22 | export const addEncounteredUser = async (
 23 |   userId: string,
 24 |   encounteredUserId: string,
 25 |   distance: number,
 26 |   location: { lat: number; lng: number }
 27 | ): Promise<void> => {
 28 |   try {
 29 |     const encounterRef = ref(database, `encounteredUsers/${userId}/${encounteredUserId}`);
 30 |
 31 |     // Get existing encounter data if it exists
 32 |     const snapshot = await get(encounterRef);
 33 |     const existingData = snapshot.exists() ? snapshot.val() : null;
 34 |
 35 |     const now = Date.now();
 36 |
 37 |     if (existingData) {
 38 |       // Update existing encounter
 39 |       await set(encounterRef, {
 40 |         userId: encounteredUserId,
 41 |         encounteredAt: existingData.encounteredAt, // Keep original encounter time
 42 |         lastSeenAt: now,
 43 |         distance,
 44 |         lat: location.lat,
 45 |         lng: location.lng,
 46 |         count: (existingData.count || 1) + 1
 47 |       });
 48 |     } else {
 49 |       // Create new encounter record
 50 |       await set(encounterRef, {
 51 |         userId: encounteredUserId,
 52 |         encounteredAt: now,
 53 |         lastSeenAt: now,
 54 |         distance,
 55 |         lat: location.lat,
 56 |         lng: location.lng,
 57 |         count: 1
 58 |       });
 59 |     }
 60 |   } catch (error) {
 61 |     console.error("'L Error adding encountered user:", error);
 62 |     // Don't throw - this is a background tracking feature
 63 |   }
 64 | };
 65 |
 66 | /**
```

```
 67 |  * Listen to user's encountered users history
 68 |  * @param userId - Current user's ID
 69 |  * @param callback - Callback function that receives encountered users object
 70 |  * @returns Unsubscribe function
 71 |  */
 72 | export const listenToEncounteredUsers = (
 73 |   userId: string,
 74 |   callback: (encounters: Record<string, EncounteredUser>) => void
 75 | ): (() => void) => {
 76 |   const encountersRef = ref(database, `encounteredUsers/${userId}`);
 77 |
 78 |   const unsubscribe = onValue(
 79 |     encountersRef,
 80 |     (snapshot: DataSnapshot) => {
 81 |       if (!snapshot.exists()) {
 82 |         callback({});
 83 |         return;
 84 |       }
 85 |
 86 |       const encounters = snapshot.val() || {};
 87 |       callback(encounters);
 88 |     },
 89 |     (error) => {
 90 |       console.error("'L Error listening to encountered users:", error);
 91 |       callback({});
 92 |     }
 93 |   );
 94 |
 95 |   return () => {
 96 |     off(encountersRef);
 97 |   };
 98 | };
 99 |
100 | /**
101 |  * Cleanup encounters older than 90 days
102 |  * @param userId - Current user's ID
103 |  */
104 | export const cleanupOldEncounters = async (userId: string): Promise<void> => {
105 |   try {
106 |     const encountersRef = ref(database, `encounteredUsers/${userId}`);
107 |     const snapshot = await get(encountersRef);
108 |
109 |     if (!snapshot.exists()) {
110 |       return;
111 |     }
112 |
113 |     const encounters = snapshot.val() || {};
114 |     const now = Date.now();
115 |     const ninetyDaysAgo = now - (90 * 24 * 60 * 60 * 1000); // 90 days in milliseconds
116 |
117 |     const cleanupPromises: Promise<void>[] = [];
118 |
119 |     Object.entries(encounters).forEach(([encounteredUserId, encounterData]: [string, any]) => {
120 |       // Check if lastSeenAt is older than 90 days
121 |       if (encounterData.lastSeenAt && encounterData.lastSeenAt < ninetyDaysAgo) {
122 |         const encounterRef = ref(database, `encounteredUsers/${userId}/${encounteredUserId}`);
123 |         cleanupPromises.push(remove(encounterRef).then(() => {
124 |           if (process.env.NODE_ENV === 'development') {
125 |             console.log(`Ø>Ýù Cleaned up old encounter: ${encounteredUserId}`);
126 |           }
127 |         }));
128 |       }
129 |     });
130 |
131 |     await Promise.all(cleanupPromises);
132 |
133 |     if (cleanupPromises.length > 0 && process.env.NODE_ENV === 'development') {
134 |       console.log(`'  Cleaned up ${cleanupPromises.length} old encounter(s)`);
135 |     }
136 |   } catch (error) {
137 |     console.error("'L Error cleaning up old encounters:", error);
```

```
138 |     // Don't throw - cleanup is a background operation
139 |   }
140 | };
141 |
142 | /**
143 |  * Get encountered users (one-time fetch, not real-time)
144 |  * @param userId - Current user's ID
145 |  * @returns Promise with encountered users object
146 |  */
147 | export const getEncounteredUsers = async (
148 |   userId: string
149 | ): Promise<Record<string, EncounteredUser>> => {
150 |   try {
151 |     const encountersRef = ref(database, `encounteredUsers/${userId}`);
152 |     const snapshot = await get(encountersRef);
153 |
154 |     if (!snapshot.exists()) {
155 |       return {};
156 |     }
157 |
158 |     return snapshot.val() || {};
159 |   } catch (error) {
160 |     console.error("'L Error getting encountered users:", error);
161 |     return {};
162 |   }
163 | };
164 |
165 |
```

File: src/services/eventService.ts

Lines: 625

```
// Event service for Firebase - manages user-created and sponsored events
import { ref, set, get, onValue, off, push, remove, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { checkInToVenue, checkOutFromVenue, getCheckInsAtVenue, CheckIn, UserCheckInData } from "./checkInService";
import { findVenueForEvent } from "./venueService";
import { isCommentingSuspended } from "./userService";

export interface Event {
  id: string;
  title: string;
  description: string;
  type: "running" | "cycling" | "walking" | "others";
  category: "user" | "sponsored";
  date: string; // ISO date string
  time: string;
  location: string;
  distance: string;
  distanceValue: number;
  lat: number;
  lng: number;
  hostId?: string; // For user-created events
  hostName?: string;
  hostAvatar?: string;
  sponsorLogo?: string; // For sponsored events
  participants: string[]; // Array of user IDs
  maxParticipants?: number;
  createdAt: number;
}

/**
 * Create a new event
 */
export const createEvent = async (
  hostId: string,
  eventData: Omit<Event, "id" | "hostId" | "participants" | "createdAt">
): Promise<string> => {
  try {
    const eventsRef = ref(database, "events");
    const newEventRef = push(eventsRef);
    const eventId = newEventRef.key!;

    // Filter out undefined values to avoid Firebase errors
    const cleanEventData: any = {};
    Object.entries(eventData).forEach(([key, value]) => {
      if (value !== undefined) {
        cleanEventData[key] = value;
      }
    });

    const event: Event = {
      ...cleanEventData,
      id: eventId,
      hostId,
      participants: [hostId], // Host is automatically a participant
      createdAt: Date.now()
    };

    await set(newEventRef, event);
    console.log(`'  Event created: ${eventId}`);
    return eventId;
  } catch (error) {
    console.error("'L Error creating event:", error);
    throw error;
  }
};

```

```
  67 | /**
  68 |  * Join an event
  69 |  */
  70 | export const joinEvent = async (
  71 |   eventId: string,
  72 |   userId: string
  73 | ): Promise<void> => {
  74 |   try {
  75 |     const eventRef = ref(database, `events/${eventId}`);
  76 |     const snapshot = await get(eventRef);
  77 |
  78 |     if (!snapshot.exists()) {
  79 |       throw new Error("Event not found");
  80 |     }
  81 |
  82 |     const event = snapshot.val() as Event;
  83 |
  84 |     // Ensure participants is an array
  85 |     const participants = Array.isArray(event.participants) ? event.participants :
(event.hostId ? [event.hostId] : []);
  86 |
  87 |     // Check if user is already a participant
  88 |     if (participants.includes(userId)) {
  89 |       return;
  90 |     }
  91 |
  92 |     // Check if event is full
  93 |     if (event.maxParticipants && participants.length >= event.maxParticipants) {
  94 |       throw new Error("Event is full");
  95 |     }
  96 |
  97 |     // Add user to participants
  98 |     await set(ref(database, `events/${eventId}/participants`), [
  99 |       ...participants,
 100 |       userId
 101 |     ]);
 102 |
 103 |     console.log(`'  User ${userId} joined event ${eventId}`);
 104 |   } catch (error) {
 105 |     console.error("'L Error joining event:", error);
 106 |     throw error;
 107 |   }
 108 | };
 109 |
 110 | /**
 111 |  * Leave an event
 112 |  */
 113 | export const leaveEvent = async (
 114 |   eventId: string,
 115 |   userId: string
 116 | ): Promise<void> => {
 117 |   try {
 118 |     const eventRef = ref(database, `events/${eventId}`);
 119 |     const snapshot = await get(eventRef);
 120 |
 121 |     if (!snapshot.exists()) {
 122 |       throw new Error("Event not found");
 123 |     }
 124 |
 125 |     const event = snapshot.val() as Event;
 126 |
 127 |     // Ensure participants is an array
 128 |     const participants = Array.isArray(event.participants) ? event.participants :
(event.hostId ? [event.hostId] : []);
 129 |
 130 |     // Remove user from participants
 131 |     const updatedParticipants = participants.filter((id: string) => id !== userId);
 132 |     await set(ref(database, `events/${eventId}/participants`), updatedParticipants);
 133 |
 134 |     console.log(`'  User ${userId} left event ${eventId}`);
 135 |   } catch (error) {
 136 |     console.error("'L Error leaving event:", error);
 137 |     throw error;
```

```
138 |   }
139 | };
140 |
141 | /**
142 |  * Get all events
143 |  */
144 | export const getAllEvents = async (): Promise<Event[]> => {
145 |   try {
146 |     const eventsRef = ref(database, "events");
147 |     const snapshot = await get(eventsRef);
148 |
149 |     if (!snapshot.exists()) {
150 |       return [];
151 |     }
152 |
153 |     const events = snapshot.val();
154 |     return Object.values(events) as Event[];
155 |   } catch (error) {
156 |     console.error("'L Error getting events:", error);
157 |     return [];
158 |   }
159 | };
160 |
161 | /**
162 |  * Listen to all events in real-time
163 |  */
164 | export const listenToEvents = (
165 |   callback: (events: Event[]) => void
166 | ): (() => void) => {
167 |   const eventsRef = ref(database, "events");
168 |
169 |   const unsubscribe = onValue(
170 |     eventsRef,
171 |     (snapshot: DataSnapshot) => {
172 |       if (!snapshot.exists()) {
173 |         callback([]);
174 |         return;
175 |       }
176 |
177 |       const events = snapshot.val();
178 |       // Ensure all events have participants as an array
179 |       const normalizedEvents = Object.values(events).map((event: any) => ({
180 |         ...event,
181 |         participants: Array.isArray(event.participants) ? event.participants : (event.hostId ?
[event.hostId] : [])
182 |       })) as Event[];
183 |       callback(normalizedEvents);
184 |     },
185 |     (error) => {
186 |       console.error("'L Error listening to events:", error);
187 |       callback([]);
188 |     }
189 |   );
190 |
191 |   return () => {
192 |     off(eventsRef);
193 |   };
194 | };
195 |
196 | /**
197 |  * Get events for a specific user (events they created or joined)
198 |  */
199 | export const getUserEvents = async (userId: string): Promise<Event[]> => {
200 |   try {
201 |     const eventsRef = ref(database, "events");
202 |     const snapshot = await get(eventsRef);
203 |
204 |     if (!snapshot.exists()) {
205 |       return [];
206 |     }
207 |
208 |     const events = snapshot.val();
```

```
209 |     const userEvents = Object.values(events).filter((event: any) =>
210 |       event.hostId === userId || event.participants?.includes(userId)
211 |     ) as Event[];
212 |
213 |     return userEvents;
214 |   } catch (error) {
215 |     console.error("'L Error getting user events:", error);
216 |     return [];
217 |   }
218 | };
219 |
220 | /**
221 |  * Update an event (only if user is the host)
222 |  */
223 | export const updateEvent = async (
224 |   eventId: string,
225 |   userId: string,
226 |   eventData: Partial<Omit<Event, "id" | "hostId" | "participants" | "createdAt">>
227 | ): Promise<void> => {
228 |   try {
229 |     const eventRef = ref(database, `events/${eventId}`);
230 |     const snapshot = await get(eventRef);
231 |
232 |     if (!snapshot.exists()) {
233 |       throw new Error("Event not found");
234 |     }
235 |
236 |     const event = snapshot.val() as Event;
237 |
238 |     if (event.hostId !== userId) {
239 |       throw new Error("Only the host can update the event");
240 |     }
241 |
242 |     // Filter out undefined values to avoid Firebase errors
243 |     const cleanEventData: any = {};
244 |     Object.entries(eventData).forEach(([key, value]) => {
245 |       if (value !== undefined) {
246 |         cleanEventData[key] = value;
247 |       }
248 |     });
249 |
250 |     // Update only the provided fields
251 |     const updates: any = {};
252 |     Object.keys(cleanEventData).forEach((key) => {
253 |       updates[`events/${eventId}/${key}`] = cleanEventData[key];
254 |     });
255 |
256 |     await set(ref(database, `events/${eventId}`), {
257 |       ...event,
258 |       ...cleanEventData,
259 |     });
260 |
261 |     console.log(`'  Event updated: ${eventId}`);
262 |   } catch (error) {
263 |     console.error("'L Error updating event:", error);
264 |     throw error;
265 |   }
266 | };
267 |
268 | /**
269 |  * Delete an event (only if user is the host)
270 |  */
271 | export const deleteEvent = async (
272 |   eventId: string,
273 |   userId: string
274 | ): Promise<void> => {
275 |   try {
276 |     const eventRef = ref(database, `events/${eventId}`);
277 |     const snapshot = await get(eventRef);
278 |
279 |     if (!snapshot.exists()) {
```

```
280 |       throw new Error("Event not found");
281 |     }
282 |
283 |     const event = snapshot.val() as Event;
284 |
285 |     if (event.hostId !== userId) {
286 |       throw new Error("Only the host can delete the event");
287 |     }
288 |
289 |     await remove(eventRef);
290 |     console.log(`'  Event deleted: ${eventId}`);
291 |   } catch (error) {
292 |     console.error("'L Error deleting event:", error);
293 |     throw error;
294 |   }
295 | };
296 |
297 | /**
298 |  * Check in to an event's location
299 |  * This will check in to the venue if the event location matches a predefined venue,
300 |  * otherwise it will create a check-in record for the event location itself
301 |  */
302 | export const checkInToEventLocation = async (
303 |   eventId: string,
304 |   userId: string,
305 |   userData: UserCheckInData
306 | ): Promise<void> => {
307 |   try {
308 |     const eventRef = ref(database, `events/${eventId}`);
309 |     const snapshot = await get(eventRef);
310 |
311 |     if (!snapshot.exists()) {
312 |       throw new Error("Event not found");
313 |     }
314 |
315 |     const event = snapshot.val() as Event;
316 |
317 |     // Try to find a matching venue
318 |     const venue = findVenueForEvent(event.lat, event.lng);
319 |
320 |     if (venue) {
321 |       // Check in to the venue
322 |       await checkInToVenue(userId, venue.id, { id: venue.id, name: venue.name }, userData);
323 |     } else {
324 |       // Create event-specific check-in
325 |       const eventCheckInRef = ref(database, `eventCheckIns/${eventId}/${userId}`);
326 |       const checkIn: CheckIn = {
327 |         userId,
328 |         venueId: `event-${eventId}`,
329 |         venueName: event.location,
330 |         activity: userData.activity,
331 |         userName: userData.userName,
332 |         userAvatar: userData.userAvatar || "",
333 |         timestamp: Date.now()
334 |       };
335 |
336 |       await set(eventCheckInRef, checkIn);
337 |       console.log(`'  User ${userId} checked in to event location ${eventId}`);
338 |     }
339 |   } catch (error) {
340 |     console.error("'L Error checking in to event location:", error);
341 |     throw error;
342 |   }
343 | };
344 |
345 | /**
346 |  * Get all check-ins at an event's location
347 |  */
348 | export const getCheckInsAtEventLocation = async (eventId: string): Promise<CheckIn[]> => {
349 |   try {
350 |     const eventRef = ref(database, `events/${eventId}`);
```

```
351 |     const snapshot = await get(eventRef);
352 |
353 |     if (!snapshot.exists()) {
354 |       return [];
355 |     }
356 |
357 |     const event = snapshot.val() as Event;
358 |
359 |     // Try to find a matching venue
360 |     const venue = findVenueForEvent(event.lat, event.lng);
361 |
362 |     if (venue) {
363 |       // Get check-ins from the venue
364 |       return await getCheckInsAtVenue(venue.id);
365 |     } else {
366 |       // Get event-specific check-ins
367 |       const eventCheckInsRef = ref(database, `eventCheckIns/${eventId}`);
368 |       const checkInsSnapshot = await get(eventCheckInsRef);
369 |
370 |       if (!checkInsSnapshot.exists()) {
371 |         return [];
372 |       }
373 |
374 |       const checkIns = checkInsSnapshot.val();
375 |       return Object.values(checkIns) as CheckIn[];
376 |     }
377 |   } catch (error) {
378 |     console.error("'L Error getting check-ins at event location:", error);
379 |     return [];
380 |   }
381 | };
382 |
383 | /**
384 |  * Listen to check-ins at an event's location in real-time
385 |  */
386 | // Comment interface
387 | export interface EventComment {
388 |   id: string;
389 |   userId: string;
390 |   userName: string;
391 |   userAvatar: string;
392 |   text: string;
393 |   timestamp: number;
394 | }
395 |
396 | /**
397 |  * Add a comment to an event
398 |  * Checks if user's commenting privileges are suspended before allowing
399 |  */
400 | export const addComment = async (
401 |   eventId: string,
402 |   userId: string,
403 |   userName: string,
404 |   userAvatar: string,
405 |   text: string
406 | ): Promise<string> => {
407 |   try {
408 |     // Check if user's commenting is suspended
409 |     const suspension = await isCommentingSuspended(userId);
410 |     if (suspension) {
411 |       const suspendedUntilText = suspension.suspendedUntil
412 |         ? ` until ${new Date(suspension.suspendedUntil).toLocaleDateString()}`
413 |         : "";
414 |       throw new Error(`Your commenting privileges are suspended${suspendedUntilText}. Reason:
${suspension.reason}`);
415 |     }
416 |
417 |     const commentsRef = ref(database, `events/${eventId}/comments`);
418 |     const newCommentRef = push(commentsRef);
419 |     const commentId = newCommentRef.key!;
420 |
421 |     const comment: EventComment = {
```

```
422 |       id: commentId,
423 |       userId,
424 |       userName,
425 |       userAvatar: userAvatar || "",
426 |       text,
427 |       timestamp: Date.now()
428 |     };
429 |
430 |     await set(newCommentRef, comment);
431 |     console.log(`'  Comment added to event ${eventId}`);
432 |     return commentId;
433 |   } catch (error) {
434 |     console.error("'L Error adding comment:", error);
435 |     throw error;
436 |   }
437 | };
438 |
439 | /**
440 |  * Listen to comments on an event in real-time
441 |  */
442 | export const listenToComments = (
443 |   eventId: string,
444 |   callback: (comments: EventComment[]) => void
445 | ): (() => void) => {
446 |   const commentsRef = ref(database, `events/${eventId}/comments`);
447 |
448 |   const unsubscribe = onValue(
449 |     commentsRef,
450 |     (snapshot: DataSnapshot) => {
451 |       if (!snapshot.exists()) {
452 |         callback([]);
453 |         return;
454 |       }
455 |
456 |       const comments = snapshot.val();
457 |       const commentsArray = Object.values(comments) as EventComment[];
458 |       // Sort by timestamp (oldest first)
459 |       commentsArray.sort((a, b) => a.timestamp - b.timestamp);
460 |       callback(commentsArray);
461 |     },
462 |     (error) => {
463 |       console.error("'L Error listening to comments:", error);
464 |       callback([]);
465 |     }
466 |   );
467 |
468 |   return () => {
469 |     off(commentsRef);
470 |   };
471 | };
472 |
473 | export const listenToEventCheckIns = (
474 |   eventId: string,
475 |   callback: (checkIns: CheckIn[]) => void
476 | ): (() => void) => {
477 |   let venueUnsubscribe: (() => void) | null = null;
478 |   let eventCheckInsUnsubscribe: (() => void) | null = null;
479 |
480 |   const eventRef = ref(database, `events/${eventId}`);
481 |
482 |   const eventUnsubscribe = onValue(
483 |     eventRef,
484 |     async (snapshot: DataSnapshot) => {
485 |       // Clean up previous listeners
486 |       if (venueUnsubscribe) {
487 |         venueUnsubscribe();
488 |         venueUnsubscribe = null;
489 |       }
490 |       if (eventCheckInsUnsubscribe) {
491 |         eventCheckInsUnsubscribe();
492 |         eventCheckInsUnsubscribe = null;
```

```
493 |       }
494 |
495 |       if (!snapshot.exists()) {
496 |         callback([]);
497 |         return;
498 |       }
499 |
500 |       const event = snapshot.val() as Event;
501 |
502 |       // Try to find a matching venue
503 |       const venue = findVenueForEvent(event.lat, event.lng);
504 |
505 |       if (venue) {
506 |         // Listen to venue check-ins
507 |         venueUnsubscribe = onValue(
508 |           ref(database, `checkIns/${venue.id}`),
509 |           (checkInsSnapshot: DataSnapshot) => {
510 |             if (!checkInsSnapshot.exists()) {
511 |               callback([]);
512 |               return;
513 |             }
514 |             const checkIns = checkInsSnapshot.val();
515 |             callback(Object.values(checkIns) as CheckIn[]);
516 |           },
517 |           (error) => {
518 |             console.error("'L Error listening to venue check-ins:", error);
519 |             callback([]);
520 |           }
521 |         );
522 |       } else {
523 |         // Listen to event-specific check-ins
524 |         const eventCheckInsRef = ref(database, `eventCheckIns/${eventId}`);
525 |         eventCheckInsUnsubscribe = onValue(
526 |           eventCheckInsRef,
527 |           (checkInsSnapshot: DataSnapshot) => {
528 |             if (!checkInsSnapshot.exists()) {
529 |               callback([]);
530 |               return;
531 |             }
532 |             const checkIns = checkInsSnapshot.val();
533 |             callback(Object.values(checkIns) as CheckIn[]);
534 |           },
535 |           (error) => {
536 |             console.error("'L Error listening to event check-ins:", error);
537 |             callback([]);
538 |           }
539 |         );
540 |       }
541 |     },
542 |     (error) => {
543 |       console.error("'L Error listening to event:", error);
544 |       callback([]);
545 |     }
546 |   );
547 |
548 |   return () => {
549 |     if (venueUnsubscribe) venueUnsubscribe();
550 |     if (eventCheckInsUnsubscribe) eventCheckInsUnsubscribe();
551 |     off(eventRef);
552 |   };
553 | };
554 |
555 | // ============ ADDITIONAL COMMENT FUNCTIONS ============
556 |
557 | /**
558 |  * Delete a comment (only the comment author, event host, or admin can delete)
559 |  * @param isAdmin - If true, skips ownership check (for admin moderation)
560 |  */
561 | export const deleteComment = async (
562 |   eventId: string,
563 |   commentId: string,
```

```
564 |   userId: string,
565 |   isAdmin: boolean = false
566 | ): Promise<void> => {
567 |   try {
568 |     // Get the comment to check ownership
569 |     const commentRef = ref(database, `events/${eventId}/comments/${commentId}`);
570 |     const commentSnapshot = await get(commentRef);
571 |
572 |     if (!commentSnapshot.exists()) {
573 |       throw new Error("Comment not found");
574 |     }
575 |
576 |     const comment = commentSnapshot.val() as EventComment;
577 |
578 |     // Skip permission check if admin
579 |     if (!isAdmin) {
580 |       // Get the event to check if user is host
581 |       const eventRef = ref(database, `events/${eventId}`);
582 |       const eventSnapshot = await get(eventRef);
583 |
584 |       if (!eventSnapshot.exists()) {
585 |         throw new Error("Event not found");
586 |       }
587 |
588 |       const event = eventSnapshot.val() as Event;
589 |
590 |       // Only allow deletion by comment author or event host
591 |       if (comment.userId !== userId && event.hostId !== userId) {
592 |         throw new Error("You don't have permission to delete this comment");
593 |       }
594 |     }
595 |
596 |     await remove(commentRef);
597 |     console.log(`'  Comment ${commentId} deleted from event ${eventId}${isAdmin ? ' (by admin)' :
'')}`);
598 |   } catch (error) {
599 |     console.error("'L Error deleting comment:", error);
600 |     throw error;
601 |   }
602 | };
603 |
604 | /**
605 |  * Get all comments for an event (used by admin)
606 |  */
607 | export const getAllEventComments = async (eventId: string): Promise<EventComment[]> => {
608 |   try {
609 |     const commentsRef = ref(database, `events/${eventId}/comments`);
610 |     const snapshot = await get(commentsRef);
611 |
612 |     if (!snapshot.exists()) {
613 |       return [];
614 |     }
615 |
616 |     const comments = snapshot.val();
617 |     const commentsArray = Object.values(comments) as EventComment[];
618 |     return commentsArray.sort((a, b) => b.timestamp - a.timestamp);
619 |   } catch (error) {
620 |     console.error("'L Error getting event comments:", error);
621 |     return [];
622 |   }
623 | };
624 |
625 |
```

File: src/services/feedService.ts

Lines: 220

```
// Feed service for Firebase - manages workout posts and social feed
import { ref, set, push, get, onValue, off, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { WorkoutHistory } from "@/contexts/UserContext";

export interface Comment {
  id: string;
  userId: string;
  username: string;
  avatar: string;
  text: string;
  timestamp: number;
}

export interface WorkoutPost {
  id: string;
  userId: string;
  workout: WorkoutHistory;
  photos?: string[];
  caption?: string;
  kudos: string[]; // Array of user IDs who gave kudos
  comments: Comment[];
  timestamp: number;
}

/**
 * Create a workout post
 */
export const createWorkoutPost = async (
  userId: string,
  workout: WorkoutHistory,
  caption?: string,
  photos?: string[]
): Promise<string> => {
  try {
    const postsRef = ref(database, "workoutPosts");
    const newPostRef = push(postsRef);
    const postId = newPostRef.key!;

    const post: WorkoutPost = {
      id: postId,
      userId,
      workout: {
        ...workout,
        date: workout.date.getTime() // Convert Date to timestamp
      } as any,
      caption,
      photos,
      kudos: [],
      comments: [],
      timestamp: Date.now()
    };

    await set(newPostRef, post);
    console.log(`'  Workout post created: ${postId}`);
    return postId;
  } catch (error) {
    console.error("'L Error creating workout post:", error);
    throw error;
  }
};

/**
 * Get all workout posts (feed)
 */
export const getWorkoutPosts = async (): Promise<WorkoutPost[]> => {
```

```
  try {
    const postsRef = ref(database, "workoutPosts");
    const snapshot = await get(postsRef);

    if (!snapshot.exists()) {
      return [];
    }

    const posts = snapshot.val();
    return Object.values(posts).map((post: any) => ({
      ...post,
      workout: {
        ...post.workout,
        date: new Date(post.workout.date) // Convert timestamp back to Date
      }
    })) as WorkoutPost[];
  } catch (error) {
    console.error("'L Error getting workout posts:", error);
    return [];
  }
};

/**
 * Listen to workout posts in real-time
 */
export const listenToWorkoutPosts = (
  callback: (posts: WorkoutPost[]) => void
): (() => void) => {
  const postsRef = ref(database, "workoutPosts");

  const unsubscribe = onValue(
    postsRef,
    (snapshot: DataSnapshot) => {
      if (!snapshot.exists()) {
        callback([]);
        return;
      }

      const posts = snapshot.val();
      const postList = Object.values(posts).map((post: any) => ({
        ...post,
        workout: {
          ...post.workout,
          date: new Date(post.workout.date)
        }
      })) as WorkoutPost[];

      // Sort by timestamp (newest first)
      callback(postList.sort((a, b) => b.timestamp - a.timestamp));
    },
    (error) => {
      console.error("'L Error listening to workout posts:", error);
      callback([]);
    }
  );

  return () => {
    off(postsRef);
  };
};

/**
 * Toggle kudos on a post
 */
export const toggleKudos = async (
  postId: string,
  userId: string
): Promise<boolean> => {
  try {
    const postRef = ref(database, `workoutPosts/${postId}`);
    const snapshot = await get(postRef);
```

```
138 |
139 |     if (!snapshot.exists()) {
140 |       throw new Error("Post not found");
141 |     }
142 |
143 |     const post = snapshot.val() as WorkoutPost;
144 |     const hasKudos = post.kudos.includes(userId);
145 |
146 |     let updatedKudos: string[];
147 |     if (hasKudos) {
148 |       updatedKudos = post.kudos.filter(id => id !== userId);
149 |     } else {
150 |       updatedKudos = [...post.kudos, userId];
151 |     }
152 |
153 |     await set(ref(database, `workoutPosts/${postId}/kudos`), updatedKudos);
154 |
155 |     return !hasKudos;
156 |   } catch (error) {
157 |     console.error("'L Error toggling kudos:", error);
158 |     throw error;
159 |   }
160 | };
161 |
162 | /**
163 |  * Add a comment to a post
164 |  */
165 | export const addComment = async (
166 |   postId: string,
167 |   comment: Omit<Comment, "id" | "timestamp">
168 | ): Promise<string> => {
169 |   try {
170 |     const commentsRef = ref(database, `workoutPosts/${postId}/comments`);
171 |     const newCommentRef = push(commentsRef);
172 |     const commentId = newCommentRef.key!;
173 |
174 |     const commentData: Comment = {
175 |       ...comment,
176 |       id: commentId,
177 |       timestamp: Date.now()
178 |     };
179 |
180 |     await set(newCommentRef, commentData);
181 |     console.log(`'  Comment added: ${commentId}`);
182 |     return commentId;
183 |   } catch (error) {
184 |     console.error("'L Error adding comment:", error);
185 |     throw error;
186 |   }
187 | };
188 |
189 | /**
190 |  * Delete a comment
191 |  */
192 | export const deleteComment = async (
193 |   postId: string,
194 |   commentId: string,
195 |   userId: string
196 | ): Promise<void> => {
197 |   try {
198 |     const commentRef = ref(database, `workoutPosts/${postId}/comments/${commentId}`);
199 |     const snapshot = await get(commentRef);
200 |
201 |     if (!snapshot.exists()) {
202 |       throw new Error("Comment not found");
203 |     }
204 |
205 |     const comment = snapshot.val() as Comment;
206 |
207 |     // Only allow deletion if user is the comment author
208 |     if (comment.userId !== userId) {
```

```
209 |       throw new Error("You can only delete your own comments");
210 |     }
211 |
212 |     await set(commentRef, null);
213 |     console.log(`'  Comment deleted: ${commentId}`);
214 |   } catch (error) {
215 |     console.error("'L Error deleting comment:", error);
216 |     throw error;
217 |   }
218 | };
219 |
220 |
```

File: src/services/firebase.ts

Lines: 40

```
// Firebase configuration and initialization
import { initializeApp } from "firebase/app";
import { getAuth, GoogleAuthProvider } from "firebase/auth";
import { getDatabase } from "firebase/database";
import { getStorage } from "firebase/storage";
import { getFunctions } from "firebase/functions";

// Your Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyBhEjBsdXq1GYbz9IDkzj2fULj-IrC-PE4",
  authDomain: "pacematch-gps.firebaseapp.com",
  databaseURL: "https://pacematch-gps-default-rtdb.asia-southeast1.firebasedatabase.app/",
  projectId: "pacematch-gps",
  storageBucket: "pacematch-gps.firebasestorage.app",
  messagingSenderId: "891545961086",
  appId: "1:891545961086:web:9842aed06947710b5d0e2f",
  measurementId: "G-EPQ2M5BR6F"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Initialize Firebase services
export const auth = getAuth(app);
export const database = getDatabase(app);
export const storage = getStorage(app);
export const functions = getFunctions(app);
export const googleProvider = new GoogleAuthProvider();

// Configure Google Auth Provider
googleProvider.setCustomParameters({
  prompt: 'select_account'
});

// Note: COOP warnings from Firebase Auth popup are harmless and don't affect functionality
// They occur due to browser security policies but sign-in still works correctly

export default app;

```

File: src/services/friendService.ts

Lines: 280

```
// Friend service for Firebase - manages friend requests and friendships
import { ref, set, get, onValue, off, remove, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { createNotification } from "./notificationService";
import { getUserData } from "./authService";

/**
 * Send a friend request
 */
export const sendFriendRequest = async (
  fromUserId: string,
  toUserId: string
): Promise<void> => {
  try {
    const requestRef = ref(database, `friendRequests/${toUserId}/${fromUserId}`);
    const outgoingRef = ref(database, `friendRequestsOutgoing/${fromUserId}/${toUserId}`);
    await set(requestRef, {
      fromUserId,
      toUserId,
      status: "pending",
      createdAt: Date.now()
    });
    await set(outgoingRef, {
      fromUserId,
      toUserId,
      status: "pending",
      createdAt: Date.now()
    });
    console.log(`'  Friend request sent from ${fromUserId} to ${toUserId}`);

    // Create notification for recipient
    try {
      const fromUserData = await getUserData(fromUserId);
      if (fromUserData) {
        await createNotification(toUserId, {
          type: "friend_request",
          fromUserId,
          fromUserName: fromUserData.name || fromUserData.username || "Someone",
          fromUserAvatar: fromUserData.photoURL || "",
        });
      }
    } catch (notificationError) {
      // Don't fail the friend request if notification fails
      console.error("'L Error creating friend request notification:", notificationError);
    }
  } catch (error) {
    console.error("'L Error sending friend request:", error);
    throw error;
  }
};

/**
 * Accept a friend request
 */
export const acceptFriendRequest = async (
  userId: string,
  fromUserId: string
): Promise<void> => {
  try {
    // Remove the request
    const requestRef = ref(database, `friendRequests/${userId}/${fromUserId}`);
    await remove(requestRef);

    // Add to both users' friends lists
    const userFriendsRef = ref(database, `friends/${userId}/${fromUserId}`);
    const fromUserFriendsRef = ref(database, `friends/${fromUserId}/${userId}`);
```

```
67 |
68 |     await set(userFriendsRef, {
69 |       friendId: fromUserId,
70 |       createdAt: Date.now()
71 |     });
72 |
73 |     await set(fromUserFriendsRef, {
74 |       friendId: userId,
75 |       createdAt: Date.now()
76 |     });
77 |
78 |     // Remove outgoing record for sender
79 |     const outgoingRef = ref(database, `friendRequestsOutgoing/${fromUserId}/${userId}`);
80 |     await remove(outgoingRef);
81 |
82 |     console.log(`'  Friend request accepted: ${userId} and ${fromUserId} are now friends`);
83 |
84 |     // Create notification for requester (the person who sent the request)
85 |     try {
86 |       const accepterData = await getUserData(userId);
87 |       if (accepterData) {
88 |         await createNotification(fromUserId, {
89 |           type: "friend_accepted",
90 |           fromUserId: userId,
91 |           fromUserName: accepterData.name || accepterData.username || "Someone",
92 |           fromUserAvatar: accepterData.photoURL || "",
93 |         });
94 |       }
95 |     } catch (notificationError) {
96 |       // Don't fail the accept if notification fails
97 |       console.error("'L Error creating friend accepted notification:", notificationError);
98 |     }
99 |   } catch (error) {
100 |     console.error("'L Error accepting friend request:", error);
101 |     throw error;
102 |   }
103 | };
104 |
105 | /**
106 |  * Decline a friend request
107 |  */
108 | export const declineFriendRequest = async (
109 |   userId: string,
110 |   fromUserId: string
111 | ): Promise<void> => {
112 |   try {
113 |     const requestRef = ref(database, `friendRequests/${userId}/${fromUserId}`);
114 |     await remove(requestRef);
115 |
116 |     // Remove outgoing record for sender
117 |     const outgoingRef = ref(database, `friendRequestsOutgoing/${fromUserId}/${userId}`);
118 |     await remove(outgoingRef);
119 |     console.log(`'  Friend request declined: ${fromUserId} -> ${userId}`);
120 |   } catch (error) {
121 |     console.error("'L Error declining friend request:", error);
122 |     throw error;
123 |   }
124 | };
125 |
126 | /**
127 |  * Cancel a sent friend request
128 |  */
129 | export const cancelFriendRequest = async (
130 |   fromUserId: string,
131 |   toUserId: string
132 | ): Promise<void> => {
133 |   try {
134 |     const requestRef = ref(database, `friendRequests/${toUserId}/${fromUserId}`);
135 |     await remove(requestRef);
136 |     const outgoingRef = ref(database, `friendRequestsOutgoing/${fromUserId}/${toUserId}`);
137 |     await remove(outgoingRef);
```

```
138 |     console.log(`'  Friend request cancelled: ${fromUserId} -> ${toUserId}`);
139 |   } catch (error) {
140 |     console.error("'L Error cancelling friend request:", error);
141 |     throw error;
142 |   }
143 | };
144 |
145 | /**
146 |  * Get pending friend requests for a user
147 |  */
148 | export const getPendingRequests = async (userId: string): Promise<{
149 |   incoming: string[];
150 |   outgoing: string[];
151 | }> => {
152 |   try {
153 |     // Get incoming requests
154 |     const incomingRef = ref(database, `friendRequests/${userId}`);
155 |     const incomingSnapshot = await get(incomingRef);
156 |     const incoming = incomingSnapshot.exists() ? Object.keys(incomingSnapshot.val()) : [];
157 |
158 |     // Get outgoing requests from dedicated node
159 |     const outgoingRef = ref(database, `friendRequestsOutgoing/${userId}`);
160 |     const outgoingSnapshot = await get(outgoingRef);
161 |     const outgoing = outgoingSnapshot.exists() ? Object.keys(outgoingSnapshot.val()) : [];
162 |
163 |     return { incoming, outgoing };
164 |   } catch (error) {
165 |     console.error("'L Error getting pending requests:", error);
166 |     return { incoming: [], outgoing: [] };
167 |   }
168 | };
169 |
170 | /**
171 |  * Listen to pending friend requests in real-time
172 |  */
173 | export const listenToFriendRequests = (
174 |   userId: string,
175 |   callback: (requests: { incoming: string[]; outgoing: string[] }) => void
176 | ): (() => void) => {
177 |   const incomingRef = ref(database, `friendRequests/${userId}`);
178 |   const outgoingRef = ref(database, `friendRequestsOutgoing/${userId}`);
179 |   let currentIncoming: string[] = [];
180 |   let currentOutgoing: string[] = [];
181 |
182 |   const emit = () => callback({ incoming: currentIncoming, outgoing: currentOutgoing });
183 |   const handleError = (error: Error) => {
184 |     console.error("'L Error listening to friend requests:", error);
185 |     currentIncoming = [];
186 |     currentOutgoing = [];
187 |     callback({ incoming: [], outgoing: [] });
188 |   };
189 |
190 |   const unsubscribeIncoming = onValue(
191 |     incomingRef,
192 |     (snapshot: DataSnapshot) => {
193 |       currentIncoming = snapshot.exists() ? Object.keys(snapshot.val()) : [];
194 |       emit();
195 |     },
196 |     handleError
197 |   );
198 |
199 |   const unsubscribeOutgoing = onValue(
200 |     outgoingRef,
201 |     (snapshot: DataSnapshot) => {
202 |       currentOutgoing = snapshot.exists() ? Object.keys(snapshot.val()) : [];
203 |       emit();
204 |     },
205 |     handleError
206 |   );
207 |
208 |   return () => {
```

```
209 |     unsubscribeIncoming();
210 |     unsubscribeOutgoing();
211 |   };
212 | };
213 |
214 | /**
215 |  * Get user's friends list
216 |  */
217 | export const getUserFriends = async (userId: string): Promise<string[]> => {
218 |   try {
219 |     const friendsRef = ref(database, `friends/${userId}`);
220 |     const snapshot = await get(friendsRef);
221 |
222 |     if (!snapshot.exists()) {
223 |       return [];
224 |     }
225 |
226 |     return Object.keys(snapshot.val());
227 |   } catch (error) {
228 |     console.error("'L Error getting user friends:", error);
229 |     return [];
230 |   }
231 | };
232 |
233 | /**
234 |  * Listen to user's friends list in real-time
235 |  */
236 | export const listenToUserFriends = (
237 |   userId: string,
238 |   callback: (friends: string[]) => void
239 | ): (() => void) => {
240 |   const friendsRef = ref(database, `friends/${userId}`);
241 |
242 |   const unsubscribe = onValue(
243 |     friendsRef,
244 |     (snapshot: DataSnapshot) => {
245 |       const friends = snapshot.exists() ? Object.keys(snapshot.val()) : [];
246 |       callback(friends);
247 |     },
248 |     (error) => {
249 |       console.error("'L Error listening to friends:", error);
250 |       callback([]);
251 |     }
252 |   );
253 |
254 |   return () => {
255 |     off(friendsRef);
256 |   };
257 | };
258 |
259 | /**
260 |  * Remove a friend (unfriend)
261 |  */
262 | export const removeFriend = async (
263 |   userId: string,
264 |   friendId: string
265 | ): Promise<void> => {
266 |   try {
267 |     const userFriendsRef = ref(database, `friends/${userId}/${friendId}`);
268 |     const friendFriendsRef = ref(database, `friends/${friendId}/${userId}`);
269 |
270 |     await remove(userFriendsRef);
271 |     await remove(friendFriendsRef);
272 |
273 |     console.log(`'  Friend removed: ${userId} and ${friendId} are no longer friends`);
274 |   } catch (error) {
275 |     console.error("'L Error removing friend:", error);
276 |     throw error;
277 |   }
278 | };
279 |
```

File: src/services/locationService.ts

Lines: 212

```
// Location service for GPS tracking and Firebase updates
import { ref, set, onValue, off, get, DataSnapshot } from "firebase/database";
import { database } from "./firebase";

/**
 * Update user's location in Firebase
 * @param {string} userId - User ID
 * @param {number} lat - Latitude
 * @param {number} lng - Longitude
 * @param {boolean} visible - Whether user is visible on map
 */
export const updateUserLocation = async (
  userId: string,
  lat: number,
  lng: number,
  visible: boolean = true
): Promise<void> => {
  try {
    const userRef = ref(database, `users/${userId}`);

    // Get existing user data
    const snapshot = await get(userRef);
    const userData = snapshot.exists() ? snapshot.val() : {};

    // Update location while preserving other data
    const updateData = {
      ...userData,
      lat,
      lng,
      visible,
      timestamp: Date.now()
    };

    // Security: Don't log user location data in production
    // Only log in development mode if needed for debugging
    if (process.env.NODE_ENV === 'development') {
      console.log(`Ø=ÜÍ Location update initiated for user`);
    }

    await set(userRef, updateData);

    // Security: Don't log user IDs in production
    if (process.env.NODE_ENV === 'development') {
      console.log(`'  Location updated successfully`);
    }
  } catch (error) {
    console.error("'L Error updating user location:", error);
    throw error;
  }
};

/**
 * Update user visibility
 * @param {string} userId - User ID
 * @param {boolean} visible - Visibility status
 */
export const updateUserVisibility = async (userId: string, visible: boolean): Promise<void> => {
  try {
    const userRef = ref(database, `users/${userId}`);

    // Get existing user data
    const snapshot = await get(userRef);
    const userData = snapshot.exists() ? snapshot.val() : {};

    // Update visibility while preserving other data
    await set(userRef, {
```

```
  67 |       ...userData,
  68 |       visible,
  69 |       timestamp: Date.now()
  70 |     });
  71 |   } catch (error) {
  72 |     console.error("Error updating user visibility:", error);
  73 |     throw error;
  74 |   }
  75 | };
  76 |
  77 | /**
  78 |  * Listen to all users' locations
  79 |  * @param {Function} callback - Callback function that receives users object
  80 |  * @returns {Function} Unsubscribe function
  81 |  */
  82 | export const listenToAllUsers = (callback: (users: Record<string, any>) => void): (() => void)
=> {
  83 |   const usersRef = ref(database, "users");
  84 |
  85 |   // Security: Only log in development mode
  86 |   if (process.env.NODE_ENV === 'development') {
  87 |     console.log("Ø=Ý  Setting up listener for all users...");
  88 |   }
  89 |
  90 |   const unsubscribe = onValue(
  91 |     usersRef,
  92 |     async (snapshot: DataSnapshot) => {
  93 |       if (!snapshot.exists()) {
  94 |         // Security: Only log in development mode
  95 |         if (process.env.NODE_ENV === 'development') {
  96 |           console.log("& þ  No users found in Firebase database");
  97 |         }
  98 |         callback({});
  99 |         return;
 100 |       }
 101 |
 102 |       const users = snapshot.val() || {};
 103 |       const userCount = Object.keys(users).length;
 104 |
 105 |       // Security: Only log count, not individual user data
 106 |       // Logging user names, locations, and IDs is a privacy/security risk
 107 |       if (process.env.NODE_ENV === 'development') {
 108 |         console.log(`Ø=ÜÊ Firebase listener triggered - ${userCount} user(s) in database`);
 109 |       }
 110 |
 111 |       // Note: Cleanup of inactive locations is handled by each user's own client
 112 |       // when they become inactive. We don't clean up other users' locations here
 113 |       // because Firebase security rules only allow users to modify their own data.
 114 |       // The 3-minute timeout is handled by filtering logic in useNearbyUsers hook.
 115 |
 116 |       callback(users);
 117 |     },
 118 |     (error) => {
 119 |       console.error("'L Error listening to users:", error);
 120 |       callback({});
 121 |     }
 122 |   );
 123 |
 124 |   return () => {
 125 |     // Security: Only log in development mode
 126 |     if (process.env.NODE_ENV === 'development') {
 127 |       console.log("Ø=Ý  Unsubscribing from users listener");
 128 |     }
 129 |     off(usersRef);
 130 |   };
 131 | };
 132 |
 133 | /**
 134 |  * Get current user's location from Firebase
 135 |  * @param {string} userId - User ID
 136 |  * @returns {Promise<Object | null>} User location data
 137 |  */
```

```
138 | export const getUserLocation = async (userId: string): Promise<{
139 |   lat: number | null;
140 |   lng: number | null;
141 |   visible: boolean;
142 |   timestamp: number | null;
143 | } | null> => {
144 |   try {
145 |     const userRef = ref(database, `users/${userId}`);
146 |     const snapshot = await get(userRef);
147 |
148 |     if (snapshot.exists()) {
149 |       const data = snapshot.val();
150 |       return {
151 |         lat: data.lat,
152 |         lng: data.lng,
153 |         visible: data.visible,
154 |         timestamp: data.timestamp
155 |       };
156 |     }
157 |     return null;
158 |   } catch (error) {
159 |     console.error("Error getting user location:", error);
160 |     throw error;
161 |   }
162 | };
163 |
164 | /**
165 |  * Clear user's location data (used when user logs out or becomes inactive)
166 |  * This removes location data and sets user as invisible
167 |  * @param {string} userId - User ID
168 |  */
169 | export const clearUserLocation = async (userId: string): Promise<void> => {
170 |   try {
171 |     const userRef = ref(database, `users/${userId}`);
172 |
173 |     // Get existing user data
174 |     const snapshot = await get(userRef);
175 |     if (!snapshot.exists()) {
176 |       return; // User doesn't exist, nothing to clear
177 |     }
178 |
179 |     const userData = snapshot.val();
180 |
181 |     // Remove location data while preserving other user data
182 |     const updateData = {
183 |       ...userData,
184 |       lat: null,
185 |       lng: null,
186 |       visible: false,
187 |       timestamp: null
188 |     };
189 |
190 |     await set(userRef, updateData);
191 |
192 |     if (process.env.NODE_ENV === 'development') {
193 |       console.log(`'  Location cleared for user`);
194 |     }
195 |   } catch (error) {
196 |     console.error("'L Error clearing user location:", error);
197 |     throw error;
198 |   }
199 | };
200 |
201 | /**
202 |  * Note: Cleanup of inactive locations is handled by each user's own client
203 |  * when they become inactive (e.g., when they stop their workout or sign out).
204 |  * We don't clean up other users' locations because Firebase security rules
205 |  * only allow users to modify their own data.
206 |  *
207 |  * The 3-minute timeout for inactive users is handled by filtering logic
208 |  * in the useNearbyUsers hook, which filters out users with timestamps
```

```
209 |  * older than 3 minutes.
210 |  */
211 |
212 |
```

File: src/services/locationSharingService.ts

Lines: 167

```
// Location sharing service for temporary location sharing in chat
import { ref, set, onValue, off, get, DataSnapshot } from "firebase/database";
import { database } from "./firebase";

export interface SharedLocation {
  lat: number;
  lng: number;
  startedAt: number;
  expiresAt: number;
  durationMinutes: number;
}

/**
 * Start sharing location with a friend for a specific duration
 */
export const startLocationSharing = async (
  userId: string,
  friendId: string,
  durationMinutes: 15 | 30 | 60,
  initialLocation: { lat: number; lng: number }
): Promise<void> => {
  try {
    const sharingRef = ref(database, `locationSharing/${userId}/${friendId}`);
    const startedAt = Date.now();
    const expiresAt = startedAt + (durationMinutes * 60 * 1000);

    const sharingData: SharedLocation = {
      lat: initialLocation.lat,
      lng: initialLocation.lng,
      startedAt,
      expiresAt,
      durationMinutes
    };

    await set(sharingRef, sharingData);
    console.log(`'  Location sharing started for ${durationMinutes} minutes`);
  } catch (error) {
    console.error("'L Error starting location sharing:", error);
    throw error;
  }
};

/**
 * Stop location sharing early
 */
export const stopLocationSharing = async (
  userId: string,
  friendId: string
): Promise<void> => {
  try {
    const sharingRef = ref(database, `locationSharing/${userId}/${friendId}`);
    await set(sharingRef, null);
    console.log(`'  Location sharing stopped`);
  } catch (error) {
    console.error("'L Error stopping location sharing:", error);
    throw error;
  }
};

/**
 * Update shared location (called periodically while sharing)
 */
export const updateSharedLocation = async (
  userId: string,
  friendId: string,
  location: { lat: number; lng: number }
```

```
): Promise<void> => {
  try {
    const sharingRef = ref(database, `locationSharing/${userId}/${friendId}`);
    const snapshot = await get(sharingRef);

    if (!snapshot.exists()) {
      // Sharing doesn't exist or expired
      return;
    }

    const currentData = snapshot.val() as SharedLocation;

    // Only update if sharing hasn't expired
    if (Date.now() < currentData.expiresAt) {
      await set(sharingRef, {
        ...currentData,
        lat: location.lat,
        lng: location.lng
      });
    } else {
      // Auto-cleanup expired sharing
      await set(sharingRef, null);
    }
  } catch (error) {
    console.error("'L Error updating shared location:", error);
  }
};

/**
 * Listen to shared location updates
 */
export const listenToSharedLocation = (
  userId: string,
  friendId: string,
  callback: (location: SharedLocation | null) => void
): (() => void) => {
  const sharingRef = ref(database, `locationSharing/${userId}/${friendId}`);

  const unsubscribe = onValue(
    sharingRef,
    (snapshot: DataSnapshot) => {
      if (!snapshot.exists()) {
        callback(null);
        return;
      }

      const data = snapshot.val() as SharedLocation;

      // Check if expired
      if (Date.now() >= data.expiresAt) {
        // Auto-cleanup
        set(sharingRef, null).catch(console.error);
        callback(null);
        return;
      }

      callback(data);
    },
    (error) => {
      console.error("'L Error listening to shared location:", error);
      callback(null);
    }
  );

  return () => {
    off(sharingRef);
  };
};

/**
 * Get active location sharing
```

```
 */
export const getActiveLocationSharing = async (
  userId: string,
  friendId: string
): Promise<SharedLocation | null> => {
  try {
    const sharingRef = ref(database, `locationSharing/${userId}/${friendId}`);
    const snapshot = await get(sharingRef);

    if (!snapshot.exists()) {
      return null;
    }

    const data = snapshot.val() as SharedLocation;

    // Check if expired
    if (Date.now() >= data.expiresAt) {
      // Auto-cleanup
      await set(sharingRef, null);
      return null;
    }

    return data;
  } catch (error) {
    console.error("'L Error getting active location sharing:", error);
    return null;
  }
};

```

File: src/services/matchingService.ts

Lines: 323

```
  1 | // Matching service for PACE-MATCH v1 algorithm
  2 | import { filterUsersByDistance } from "@/utils/distance";
  3 | import { calculateDistance } from "@/utils/distance";
  4 |
  5 | export type Activity = "running" | "cycling" | "walking";
  6 | export type FitnessLevel = "beginner" | "intermediate" | "pro";
  7 | export type RadiusPreference = "nearby" | "normal" | "wide";
  8 |
  9 | export interface VisibilitySettings {
 10 |   visibleToAllLevels: boolean;
 11 |   allowedLevels: FitnessLevel[];
 12 | }
 13 |
 14 | export type SearchFilter = FitnessLevel | "all";
 15 |
 16 | export interface MatchingUser {
 17 |   uid: string;
 18 |   location: { lat: number; lng: number };
 19 |   activity: Activity;
 20 |   fitnessLevel: FitnessLevel;
 21 |   pace: number; // min/km for running/walking, km/h for cycling
 22 |   visibility: VisibilitySettings;
 23 |   searchFilter?: SearchFilter; // Who do I want to find? (Beginner/Intermediate/Pro/All)
 24 |   radiusPreference?: RadiusPreference;
 25 |   profileVisible?: boolean;
 26 |   [key: string]: any; // Allow other user properties
 27 | }
 28 |
 29 | export interface MatchResult {
 30 |   user: MatchingUser;
 31 |   score: number;
 32 |   distance: number; // in meters
 33 | }
 34 |
 35 | // Radius lookup table: exact distances for each activity and preference combination
 36 | // Still Apply (nearby 0.5x), Normal (1x), Wide (2x)
 37 | const RADIUS_LOOKUP: Record<Activity, Record<RadiusPreference, number>> = {
 38 |   walking: {
 39 |     nearby: 100,   // Still Apply (0.5x): 100m
 40 |     normal: 200,   // Normal (1x): 200m
 41 |     wide: 400      // Wide (2x): 400m
 42 |   },
 43 |   running: {
 44 |     nearby: 200,   // Still Apply (0.5x): 200m (slightly tighter)
 45 |     normal: 350,   // Normal (1x): 350m (optimized from 500m)
 46 |     wide: 800      // Wide (2x): 800m (more reasonable)
 47 |   },
 48 |   cycling: {
 49 |     nearby: 400,   // Still Apply (0.5x): 400m (tighter)
 50 |     normal: 1000,  // Normal (1x): 1km
 51 |     wide: 2000     // Wide (2x): 2km
 52 |   }
 53 | };
 54 |
 55 | /**
 56 |  * Get radius based on activity type and user preference
 57 |  * Uses direct lookup table for exact distance control
 58 |  * Still Apply (nearby): Walking 100m, Running 200m, Cycling 400m
 59 |  * Normal: Walking 200m, Running 350m, Cycling 1km
 60 |  * Wide: Walking 400m, Running 800m, Cycling 2km
 61 |  */
 62 | export function computeRadius(
 63 |   user: MatchingUser,
 64 |   nearbyCount?: number
 65 | ): number {
 66 |   const activity = user.activity || "running";
```

```
  const preference = user.radiusPreference || "normal";

  // Return exact radius from lookup table
  return RADIUS_LOOKUP[activity]?.[preference] || RADIUS_LOOKUP.running.normal;
}

/**
 * Check if fitness levels are compatible based on visibility settings
 */
export function fitnessAllowed(
  myLevel: FitnessLevel,
  candidateLevel: FitnessLevel,
  visibility: VisibilitySettings
): boolean {
  if (visibility.visibleToAllLevels) return true;
  return visibility.allowedLevels.includes(candidateLevel);
}

/**
 * Check if paces are compatible (30% difference tolerance)
 */
export function paceCompatible(myPace: number, candidatePace: number): boolean {
  // Allow if pace not set (null, undefined, 0, or NaN)
  if (!myPace || !candidatePace || isNaN(myPace) || isNaN(candidatePace)) return true;

  // Prevent division by zero or very small numbers
  if (myPace <= 0.001) return true;

  let diff = Math.abs(myPace - candidatePace);
  let percent = diff / myPace;
  return percent <= 0.30; // Allow 30% pace difference
}

/**
 * Calculate distance between two points in meters
 */
function distanceInMeters(
  lat1: number,
  lng1: number,
  lat2: number,
  lng2: number
): number {
  const distKm = calculateDistance(lat1, lng1, lat2, lng2);
  return distKm ? distKm * 1000 : Infinity;
}

/**
 * Score a candidate match
 * Distance: 50%, Pace: 30%, Level: 20%
 */
export function scoreCandidate(
  user: MatchingUser,
  candidate: MatchingUser,
  radius: number
): number {
  const dist = distanceInMeters(
    user.location.lat,
    user.location.lng,
    candidate.location.lat,
    candidate.location.lng
  );

  // Distance score (closer is better, normalized to 0-1)
  const distanceScore = Math.max(0, 1 - (dist / radius));

  // Pace score (closer pace is better)
  let paceScore = 1;
  if (user.pace && candidate.pace && user.pace > 0.001 && candidate.pace > 0.001) {
    const paceDiff = Math.abs(user.pace - candidate.pace);
    const pacePercent = paceDiff / user.pace;
    paceScore = Math.max(0, 1 - pacePercent); // Normalize to 0-1
```

```
  }

  // Level score (same level = 1, different = 0.5)
  const levelScore = user.fitnessLevel === candidate.fitnessLevel ? 1 : 0.5;

  // Weighted combination
  return (distanceScore * 0.5) + (paceScore * 0.3) + (levelScore * 0.2);
}

/**
 * Main matching function
 * Returns top 5 matches ranked by score
 */
export function matchUsers(
  user: MatchingUser,
  allUsers: Record<string, any>,
  nearbyCount?: number
): MatchResult[] {
  if (user.profileVisible === false) {
    return [];
  }

  // Get fixed radius based on activity type
  const radius = computeRadius(user, nearbyCount);
  const radiusKm = radius / 1000; // Convert to km for filterUsersByDistance

  // Get nearby users within radius
  const candidates = filterUsersByDistance(
    allUsers,
    user.location.lat,
    user.location.lng,
    radiusKm,
    user.uid
  );

  // Filter for active users only (users with recent location updates within 3 minutes)
  // This ensures we only match with users who are currently working out
  const now = Date.now();
  const activeThreshold = 3 * 60 * 1000; // 3 minutes in milliseconds
  const activeCandidates = candidates.filter(candidate => {
    // User must have a timestamp indicating recent location update
    if (!candidate.timestamp) {
      console.log(`Match candidate ${candidate.id} filtered out - no timestamp (not actively tracking)`); return false;
    }

    // Check if timestamp is recent (within 3 minutes)
    const timeDiff = now - candidate.timestamp;
    const isActive = timeDiff <= activeThreshold;

    if (!isActive) {
      console.log(`Match candidate ${candidate.id} filtered out - timestamp too old (${Math.round(timeDiff / 1000)}s ago...
    }

    return isActive;
  });

  console.log(`Matching: ${candidates.length} nearby users !’ ${activeCandidates.length} with active workouts`);

  // Filter by fitness level and pace compatibility
  const filtered = activeCandidates
    .map(candidate => {
      // Ensure candidate has required fields
      const candidateFitnessLevel = candidate.fitnessLevel || "intermediate";
      const candidatePace = candidate.pace || null;
      const candidateActivity = candidate.activity || user.activity;
      const candidateVisibility = candidate.visibility || {
        visibleToAllLevels: true,
        allowedLevels: ["beginner", "intermediate", "pro"]
      };
      const candidateProfileVisible = candidate.profileVisible !== false;
```

```
209 |       const candidateSearchFilter: SearchFilter = candidate.searchFilter || "all";
210 |
211 |       // Only match users with same activity
212 |       if (candidateActivity !== user.activity) {
213 |         return null;
214 |       }
215 |
216 |       // Check if user's search filter matches candidate's fitness level
217 |       const userSearchFilter = user.searchFilter || "all";
218 |       if (userSearchFilter !== "all" && candidateFitnessLevel !== userSearchFilter) {
219 |         return null;
220 |       }
221 |
222 |       if (!candidateProfileVisible) {
223 |         return null;
224 |       }
225 |
226 |       if (candidateSearchFilter !== "all" && candidateSearchFilter !== user.fitnessLevel) {
227 |         return null;
228 |       }
229 |
230 |       // Check if candidate's visibility filter allows user's fitness level
231 |       if (!fitnessAllowed(user.fitnessLevel, candidateFitnessLevel, candidateVisibility)) {
232 |         return null;
233 |       }
234 |
235 |       // Check pace compatibility
236 |       if (user.pace && candidatePace && !paceCompatible(user.pace, candidatePace)) {
237 |         return null;
238 |       }
239 |
240 |       return {
241 |         ...candidate,
242 |         fitnessLevel: candidateFitnessLevel,
243 |         pace: candidatePace,
244 |         activity: candidateActivity,
245 |         visibility: candidateVisibility
246 |       };
247 |     })
248 |     .filter((c): c is MatchingUser & { id: string; distance: number } => c !== null);
249 |
250 |   // Score and rank candidates
251 |   const ranked = filtered
252 |     .map(candidate => {
253 |       const matchUser: MatchingUser = {
254 |         uid: candidate.id,
255 |         location: { lat: candidate.lat, lng: candidate.lng },
256 |         activity: candidate.activity,
257 |         fitnessLevel: candidate.fitnessLevel,
258 |         pace: candidate.pace || 0,
259 |         visibility: candidate.visibility || {
260 |           visibleToAllLevels: true,
261 |           allowedLevels: ["beginner", "intermediate", "pro"]
262 |         },
263 |         profileVisible: candidate.profileVisible !== false,
264 |         radiusPreference: candidate.radiusPreference,
265 |         ...candidate
266 |       };
267 |
268 |       return {
269 |         user: matchUser,
270 |         score: scoreCandidate(user, matchUser, radius),
271 |         distance: candidate.distance * 1000 // Convert km to meters
272 |       };
273 |     })
274 |     .sort((a, b) => b.score - a.score); // Sort by score descending
275 |
276 |   // Return top 5 matches
277 |   return ranked.slice(0, 5);
278 | }
279 |
```

```
/**
 * Calculate average pace from workout history
 * @param workouts - Array of workout history items
 * @param activity - Activity type to filter by
 * @returns Average pace (min/km for running/walking, km/h for cycling)
 */
export function calculatePaceFromWorkouts(
  workouts: Array<{ activity: Activity; duration: number; distance: number; avgSpeed?: number }>,
  activity: Activity
): number | null {
  // Filter workouts by activity and get most recent 5-10
  const relevantWorkouts = workouts
    .filter(w => w.activity === activity)
    .slice(-10) // Last 10 workouts
    .filter(w => w.distance > 0 && w.duration > 0);

  if (relevantWorkouts.length === 0) {
    return null;
  }

  if (activity === "cycling") {
    // For cycling, use average speed (km/h)
    const speeds = relevantWorkouts
      .map(w => w.avgSpeed || (w.distance / (w.duration / 3600)))
      .filter(s => s > 0);

    if (speeds.length === 0) return null;
    return speeds.reduce((a, b) => a + b, 0) / speeds.length;
  } else {
    // For running/walking, calculate min/km
    const paces = relevantWorkouts
      .map(w => {
        const distanceKm = w.distance;
        const durationMinutes = w.duration / 60;
        return durationMinutes / distanceKm;
      })
      .filter(p => p > 0 && p < 30); // Reasonable pace range (0-30 min/km)

    if (paces.length === 0) return null;
    return paces.reduce((a, b) => a + b, 0) / paces.length;
  }
}

```

File: src/services/messageService.ts

Lines: 399

```
// Message service for Firebase - real-time messaging
import { ref, set, push, onValue, off, get, query, orderByChild, remove, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { createNotification } from "./notificationService";
import { getUserFriends } from "./friendService";
import { getUserData } from "./authService";
import { isUserBlocked } from "./userService";

export interface Message {
  id: string;
  senderId: string;
  receiverId: string;
  content: string;
  timestamp: number;
  isRead?: boolean;
  type?: 'text' | 'location';
  location?: { lat: number; lng: number };
  locationSharingId?: string;
  expiresAt?: number;
}

/**
 * Send a message
 */
export const sendMessage = async (
  senderId: string,
  receiverId: string,
  content: string
): Promise<string> => {
  try {
    // Check if sender is blocked by receiver
    const isBlocked = await isUserBlocked(receiverId, senderId);
    if (isBlocked) {
      throw new Error("You cannot send messages to this user. You have been blocked.");
    }

    // Create message in both users' conversation threads
    const conversationId = [senderId, receiverId].sort().join("_");
    const messagesRef = ref(database, `messages/${conversationId}`);
    const newMessageRef = push(messagesRef);
    const messageId = newMessageRef.key!;
    const timestamp = Date.now();

    const messageData: Message = {
      id: messageId,
      senderId,
      receiverId,
      content,
      timestamp,
      isRead: false
    };

    // IMPORTANT: Create conversation metadata FIRST so that security rules can verify participants
    // The rules check if conversations/{id}/participants/{userId} exists before allowing writes
    const conversationMetaRef = ref(database, `conversations/${conversationId}`);
    const participantMap: Record<string, boolean> = {
      [senderId]: true,
      [receiverId]: true
    };
    await set(conversationMetaRef, {
      participants: participantMap,
      participantList: Object.keys(participantMap),
      lastMessage: content,
      lastMessageTime: timestamp,
      lastMessageSender: senderId,
      updatedAt: timestamp
```

```
    });

    // Now write the message (conversation exists, so rules will pass)
    await set(newMessageRef, messageData);

    // Maintain lightweight per-user summaries for fast lookups
    await updateUserConversationSummaries(conversationId, senderId, receiverId, messageData);

    // Check if this is the first message to a non-friend (message request)
    // Also check if chat is muted before creating notification
    try {
      const senderFriends = await getUserFriends(senderId);
      const isFriends = senderFriends.includes(receiverId);

      // Check if conversation existed before (if this is first message)
      const conversationSnapshot = await get(ref(database, `messages/${conversationId}`));
      const isFirstMessage = !conversationSnapshot.exists() ||
        Object.keys(conversationSnapshot.val() || {}).length === 1; // Only the message we just added

      // Check if receiver has muted this chat (client-side check would be better, but this works)
      // For now, we'll create the notification and let the client-side handle muting

      if (!isFriends && isFirstMessage) {
        // Create message request notification
        const senderData = await getUserData(senderId);
        if (senderData) {
          await createNotification(receiverId, {
            type: "message_request",
            fromUserId: senderId,
            fromUserName: senderData.name || senderData.username || "Someone",
            fromUserAvatar: senderData.photoURL || "",
            message: content.length > 50 ? content.substring(0, 50) + "..." : content,
          });
        }
      }
    } catch (notificationError) {
      // Don't fail message send if notification fails
      console.error("'L Error creating message request notification:", notificationError);
    }

    console.log(`'  Message sent: ${messageId}`);
    return messageId;
  } catch (error) {
    console.error("'L Error sending message:", error);
    throw error;
  }
};

/**
 * Listen to messages in a conversation
 */
export const listenToMessages = (
  userId1: string,
  userId2: string,
  callback: (messages: Message[]) => void
): (() => void) => {
  const conversationId = [userId1, userId2].sort().join("_");
  const messagesRef = ref(database, `messages/${conversationId}`);

  const unsubscribe = onValue(
    query(messagesRef, orderByChild("timestamp")),
    (snapshot: DataSnapshot) => {
      if (!snapshot.exists()) {
        callback([]);
        return;
      }

      const messages = snapshot.val();
      const messageList = Object.values(messages) as Message[];
      callback(messageList.sort((a, b) => a.timestamp - b.timestamp));
    },
```

```
138 |     (error) => {
139 |       console.error("'L Error listening to messages:", error);
140 |       callback([]);
141 |     }
142 |   );
143 |
144 |   return () => {
145 |     off(messagesRef);
146 |   };
147 | };
148 |
149 | /**
150 |  * Get all conversations for a user
151 |  */
152 | export const getUserConversations = async (userId: string): Promise<Array<{
153 |   conversationId: string;
154 |   otherUserId: string;
155 |   lastMessage: string;
156 |   lastMessageTime: number;
157 |   unreadCount: number;
158 | }>> => {
159 |   try {
160 |     const userConversationsRef = ref(database, `userConversations/${userId}`);
161 |     const snapshot = await get(userConversationsRef);
162 |
163 |     if (!snapshot.exists()) {
164 |       return [];
165 |     }
166 |
167 |     const conversations = snapshot.val();
168 |     const userConversations: Array<{
169 |       conversationId: string;
170 |       otherUserId: string;
171 |       lastMessage: string;
172 |       lastMessageTime: number;
173 |       unreadCount: number;
174 |     }> = [];
175 |
176 |     Object.entries(conversations).forEach(([conversationId, data]: [string, any]) => {
177 |       const otherUserId = data.otherUserId || "";
178 |       userConversations.push({
179 |         conversationId,
180 |         otherUserId,
181 |         lastMessage: data.lastMessage || "",
182 |         lastMessageTime: data.lastMessageTime || 0,
183 |         unreadCount: data.unreadCount || 0
184 |       });
185 |     });
186 |
187 |     // Sort by last message time
188 |     return userConversations.sort((a, b) => b.lastMessageTime - a.lastMessageTime);
189 |   } catch (error) {
190 |     console.error("'L Error getting user conversations:", error);
191 |     return [];
192 |   }
193 | };
194 |
195 | /**
196 |  * Mark messages as read
197 |  */
198 | export const markMessagesAsRead = async (
199 |   userId1: string,
200 |   userId2: string
201 | ): Promise<void> => {
202 |   try {
203 |     const conversationId = [userId1, userId2].sort().join("_");
204 |     const messagesRef = ref(database, `messages/${conversationId}`);
205 |     const snapshot = await get(query(messagesRef, orderByChild("timestamp")));
206 |
207 |     if (!snapshot.exists()) {
208 |       return;
```

```
209 |     }
210 |
211 |     const messages = snapshot.val();
212 |     const unreadMessages = Object.entries(messages).filter(
213 |       ([, message]: [string, any]) => message.receiverId === userId1 && !message.isRead
214 |     );
215 |
216 |     if (unreadMessages.length > 0) {
217 |       await Promise.all(
218 |         unreadMessages.map(([messageId, message]: [string, any]) => {
219 |           const messageRef = ref(database, `messages/${conversationId}/${messageId}`);
220 |           return set(messageRef, { ...message, isRead: true });
221 |         })
222 |       );
223 |     }
224 |
225 |     // Reset unread count for the viewer
226 |     const userConversationRef = ref(database, `userConversations/${userId1}/${conversationId}`);
227 |     const userConversationSnapshot = await get(userConversationRef);
228 |     if (userConversationSnapshot.exists()) {
229 |       await set(userConversationRef, {
230 |         ...userConversationSnapshot.val(),
231 |         unreadCount: 0
232 |       });
233 |     }
234 |   } catch (error) {
235 |     console.error("'L Error marking messages as read:", error);
236 |   }
237 | };
238 |
239 | /**
240 |  * Update per-user conversation summaries used for fast lookups and rule-friendly reads.
241 |  * Note: We can't read the receiver's current unread count (security rules), so we use
242 |  * a special field that the receiver's client will merge with their local count.
243 |  */
244 | const updateUserConversationSummaries = async (
245 |   conversationId: string,
246 |   senderId: string,
247 |   receiverId: string,
248 |   messageData: Message
249 | ): Promise<void> => {
250 |   const senderSummaryRef = ref(database, `userConversations/${senderId}/${conversationId}`);
251 |   const receiverSummaryRef = ref(database, `userConversations/${receiverId}/${conversationId}`);
252 |
253 |   const senderSummary = {
254 |     conversationId,
255 |     otherUserId: receiverId,
256 |     lastMessage: messageData.content,
257 |     lastMessageTime: messageData.timestamp,
258 |     lastMessageSender: messageData.senderId,
259 |     unreadCount: 0
260 |   };
261 |
262 |   // For receiver, we set hasNewMessage flag and lastMessageTime
263 |   // The receiver's client will handle unread count when they load their conversations
264 |   const receiverSummary = {
265 |     conversationId,
266 |     otherUserId: senderId,
267 |     lastMessage: messageData.content,
268 |     lastMessageTime: messageData.timestamp,
269 |     lastMessageSender: messageData.senderId,
270 |     hasNewMessage: true
271 |   };
272 |
273 |   await Promise.all([
274 |     set(senderSummaryRef, senderSummary),
275 |     set(receiverSummaryRef, receiverSummary)
276 |   ]);
277 | };
278 |
279 | /**
```

```
 * Send a location message
 */
export const sendLocationMessage = async (
  senderId: string,
  receiverId: string,
  location: { lat: number; lng: number },
  locationSharingId?: string,
  expiresAt?: number
): Promise<string> => {
  try {
    // Check if sender is blocked by receiver
    const isBlocked = await isUserBlocked(receiverId, senderId);
    if (isBlocked) {
      throw new Error("You cannot share location with this user. You have been blocked.");
    }

    const conversationId = [senderId, receiverId].sort().join("_");
    const messagesRef = ref(database, `messages/${conversationId}`);
    const newMessageRef = push(messagesRef);
    const messageId = newMessageRef.key!;
    const timestamp = Date.now();

    const messageData: Message = {
      id: messageId,
      senderId,
      receiverId,
      content: "Ø=ÜÍ Shared location",
      timestamp,
      isRead: false,
      type: 'location',
      location,
      ...(locationSharingId && { locationSharingId }),
      ...(expiresAt && { expiresAt })
    };

    // IMPORTANT: Create conversation metadata FIRST so that security rules can verify participants
    const conversationMetaRef = ref(database, `conversations/${conversationId}`);
    const participantMap: Record<string, boolean> = {
      [senderId]: true,
      [receiverId]: true
    };
    await set(conversationMetaRef, {
      participants: participantMap,
      participantList: Object.keys(participantMap),
      lastMessage: "Ø=ÜÍ Shared location",
      lastMessageTime: timestamp,
      lastMessageSender: senderId,
      updatedAt: timestamp
    });

    // Now write the message (conversation exists, so rules will pass)
    await set(newMessageRef, messageData);

    // Maintain lightweight per-user summaries
    await updateUserConversationSummaries(conversationId, senderId, receiverId, messageData);

    console.log(`'  Location message sent: ${messageId}`);
    return messageId;
  } catch (error) {
    console.error("'L Error sending location message:", error);
    throw error;
  }
};

/**
 * Delete a conversation (all messages between two users)
 * @param requesterId - The user requesting the deletion (must be one of the participants)
 * @param otherUserId - The other participant in the conversation
 */
export const deleteConversation = async (
  requesterId: string,
```

```
  otherUserId: string
): Promise<void> => {
  try {
    // Verify requester is a participant
    if (!requesterId || !otherUserId) {
      throw new Error("Both user IDs are required to delete a conversation");
    }

    const conversationId = [requesterId, otherUserId].sort().join("_");

    // Verify conversation exists and requester is a participant
    const conversationRef = ref(database, `conversations/${conversationId}`);
    const conversationSnapshot = await get(conversationRef);

    if (!conversationSnapshot.exists()) {
      // Conversation doesn't exist, nothing to delete
      console.log(`!9þ  Conversation ${conversationId} doesn't exist, nothing to delete`);
      return;
    }

    const conversationData = conversationSnapshot.val();
    if (!conversationData.participants || !conversationData.participants[requesterId]) {
      throw new Error("You are not authorized to delete this conversation");
    }

    // Delete all messages in the conversation
    const messagesRef = ref(database, `messages/${conversationId}`);
    await remove(messagesRef);

    // Delete conversation metadata
    await remove(conversationRef);

    // Delete user conversation summaries for both users
    const user1SummaryRef = ref(database, `userConversations/${requesterId}/${conversationId}`);
    const user2SummaryRef = ref(database, `userConversations/${otherUserId}/${conversationId}`);

    await Promise.all([
      remove(user1SummaryRef),
      remove(user2SummaryRef)
    ]);

    console.log(`'  Conversation ${conversationId} deleted by ${requesterId}`);
  } catch (error) {
    console.error("'L Error deleting conversation:", error);
    throw error;
  }
};

```

File: src/services/notificationService.ts

Lines: 195

```
// Notification service for Firebase - manages persistent notifications
import { ref, set, get, onValue, off, update, DataSnapshot, push, remove } from "firebase/database";
import { database } from "./firebase";

export type NotificationType =
  | "message"
  | "message_request"
  | "friend_request"
  | "friend_accepted"
  | "poke"
  | "workout_complete"
  | "achievement"
  | "report_submitted"
  | "nearby_active_user" // Notification when active users are nearby
  // Admin moderation notification types
  | "admin_comment_deleted"
  | "admin_event_deleted"
  | "admin_event_cancelled"
  | "admin_warning"
  | "admin_comment_suspended"
  | "admin_comment_restored"
  | "username_change_required"; // Admin requested username change due to misuse

export interface Notification {
  id: string;
  type: NotificationType;
  fromUserId: string;
  fromUserName: string;
  fromUserAvatar: string;
  message?: string;
  timestamp: number;
  read: boolean;
  workoutId?: string; // For poke notifications linking to workout
  linkType?: string; // For navigation context
  // Admin moderation fields
  eventId?: string; // For event-related notifications
  eventTitle?: string; // Event title for context
  reason?: string; // Admin action reason
  adminId?: string; // Admin who took action
}

/**
 * Create a notification in Firebase
 */
export const createNotification = async (
  userId: string,
  notification: Omit<Notification, "id" | "timestamp" | "read">
): Promise<string> => {
  try {
    const notificationsRef = ref(database, `notifications/${userId}`);
    const newNotificationRef = push(notificationsRef);
    const notificationId = newNotificationRef.key!;

    const notificationData: Notification = {
      ...notification,
      id: notificationId,
      timestamp: Date.now(),
      read: false,
    };

    await set(newNotificationRef, notificationData);
    console.log(`'  Notification created for user ${userId}: ${notification.type}`);
    return notificationId;
  } catch (error) {
    console.error("'L Error creating notification:", error);
    throw error;
```

```
  }
};

/**
 * Listen to user's notifications in real-time
 */
export const listenToNotifications = (
  userId: string,
  callback: (notifications: Notification[]) => void
): (() => void) => {
  const notificationsRef = ref(database, `notifications/${userId}`);

  const unsubscribe = onValue(
    notificationsRef,
    (snapshot: DataSnapshot) => {
      if (!snapshot.exists()) {
        callback([]);
        return;
      }

      const notificationsData = snapshot.val();
      const notifications: Notification[] = Object.entries(notificationsData)
        .map(([id, data]: [string, any]) => ({
          ...data,
          id,
        }))
        .sort((a, b) => b.timestamp - a.timestamp); // Sort by newest first

      callback(notifications);
    },
    (error) => {
      console.error("'L Error listening to notifications:", error);
      callback([]);
    }
  );

  return () => {
    off(notificationsRef);
  };
};

/**
 * Mark a notification as read
 */
export const markNotificationAsRead = async (
  userId: string,
  notificationId: string
): Promise<void> => {
  try {
    const notificationRef = ref(database, `notifications/${userId}/${notificationId}`);
    await update(notificationRef, { read: true });
    console.log(`'  Notification ${notificationId} marked as read`);
  } catch (error) {
    console.error("'L Error marking notification as read:", error);
    throw error;
  }
};

/**
 * Mark all notifications as read for a user
 */
export const markAllNotificationsAsRead = async (userId: string): Promise<void> => {
  try {
    const notificationsRef = ref(database, `notifications/${userId}`);
    const snapshot = await get(notificationsRef);

    if (!snapshot.exists()) {
      return;
    }

    const notificationsData = snapshot.val();
```

```
138 |     const updates: Record<string, boolean> = {};
139 |
140 |     Object.keys(notificationsData).forEach((notificationId) => {
141 |       if (!notificationsData[notificationId].read) {
142 |         updates[`${notificationId}/read`] = true;
143 |       }
144 |     });
145 |
146 |     if (Object.keys(updates).length > 0) {
147 |       await update(notificationsRef, updates);
148 |       console.log(`'  Marked all notifications as read for user ${userId}`);
149 |     }
150 |   } catch (error) {
151 |     console.error("'L Error marking all notifications as read:", error);
152 |     throw error;
153 |   }
154 | };
155 |
156 | /**
157 |  * Delete a notification (optional - for user cleanup)
158 |  */
159 | export const deleteNotification = async (
160 |   userId: string,
161 |   notificationId: string
162 | ): Promise<void> => {
163 |   try {
164 |     const notificationRef = ref(database, `notifications/${userId}/${notificationId}`);
165 |     await remove(notificationRef);
166 |     console.log(`'  Notification ${notificationId} deleted`);
167 |   } catch (error) {
168 |     console.error("'L Error deleting notification:", error);
169 |     throw error;
170 |   }
171 | };
172 |
173 | /**
174 |  * Get unread notification count
175 |  */
176 | export const getUnreadNotificationCount = async (userId: string): Promise<number> => {
177 |   try {
178 |     const notificationsRef = ref(database, `notifications/${userId}`);
179 |     const snapshot = await get(notificationsRef);
180 |
181 |     if (!snapshot.exists()) {
182 |       return 0;
183 |     }
184 |
185 |     const notificationsData = snapshot.val();
186 |     return Object.values(notificationsData).filter(
187 |       (notification: any) => !notification.read
188 |     ).length;
189 |   } catch (error) {
190 |     console.error("'L Error getting unread notification count:", error);
191 |     return 0;
192 |   }
193 | };
194 |
195 |
```

File: src/services/pokeService.ts

Lines: 194

```
// Poke service for Firebase - manages pokes between users
import { ref, set, get, onValue, off, remove, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { createNotification } from "./notificationService";
import { getUserData } from "./authService";

/**
 * Check if user has an active workout session
 * This checks if user has recent location updates (within last 3 minutes)
 * which indicates they're actively tracking a workout
 * Inactive locations automatically terminate after 3 minutes
 */
const isUserActive = async (userId: string): Promise<boolean> => {
  try {
    const userRef = ref(database, `users/${userId}`);
    const snapshot = await get(userRef);

    if (!snapshot.exists()) {
      return false;
    }

    const userData = snapshot.val();
    const timestamp = userData.timestamp;

    if (!timestamp) {
      return false;
    }

    // Consider user active if location was updated within last 3 minutes
    const threeMinutesAgo = Date.now() - (3 * 60 * 1000);
    return timestamp > threeMinutesAgo;
  } catch (error) {
    console.error("'L Error checking user activity:", error);
    // If we can't check, allow the poke (fail open, frontend validation is primary)
    return true;
  }
};

/**
 * Send a poke to another user
 * A poke is a lightweight way to show interest in matching with someone
 * Note: Frontend validation should check workout state, this is defense in depth
 */
export const sendPoke = async (
  fromUserId: string,
  toUserId: string,
  workoutId?: string
): Promise<void> => {
  try {
    // Optional backend validation: Check if sender has active workout
    // Frontend validation is primary, this is defense in depth
    const isActive = await isUserActive(fromUserId);
    if (!isActive) {
      throw new Error("You must have an active workout session to poke someone");
    }

    const pokeRef = ref(database, `pokes/${toUserId}/${fromUserId}`);
    await set(pokeRef, {
      fromUserId,
      toUserId,
      status: "pending",
      createdAt: Date.now(),
      workoutId: workoutId || null
    });
    console.log(`'  Poke sent from ${fromUserId} to ${toUserId}`);

```

```
    // Create notification for recipient
    try {
      const fromUserData = await getUserData(fromUserId);
      if (fromUserData) {
        await createNotification(toUserId, {
          type: "poke",
          fromUserId,
          fromUserName: fromUserData.name || fromUserData.username || "Someone",
          fromUserAvatar: fromUserData.photoURL || "",
          message: "poked you! They're interested in matching",
          workoutId: workoutId,
          linkType: workoutId ? "workout" : "map",
        });
      }
    } catch (notificationError) {
      // Don't fail the poke if notification fails
      console.error("'L Error creating poke notification:", notificationError);
    }
  } catch (error) {
    console.error("'L Error sending poke:", error);
    throw error;
  }
};

/**
 * Accept a poke (user acknowledges the poke)
 * This could lead to opening a chat or showing interest back
 */
export const acceptPoke = async (
  userId: string,
  fromUserId: string
): Promise<void> => {
  try {
    // Remove the poke
    const pokeRef = ref(database, `pokes/${userId}/${fromUserId}`);
    await remove(pokeRef);

    // Optionally, you could create a "mutual interest" record here
    // For now, we just remove the poke and let users interact normally

    console.log(`'  Poke accepted: ${userId} acknowledged poke from ${fromUserId}`);
  } catch (error) {
    console.error("'L Error accepting poke:", error);
    throw error;
  }
};

/**
 * Dismiss/ignore a poke
 */
export const dismissPoke = async (
  userId: string,
  fromUserId: string
): Promise<void> => {
  try {
    const pokeRef = ref(database, `pokes/${userId}/${fromUserId}`);
    await remove(pokeRef);
    console.log(`'  Poke dismissed: ${fromUserId} -> ${userId}`);
  } catch (error) {
    console.error("'L Error dismissing poke:", error);
    throw error;
  }
};

/**
 * Get pending pokes for a user
 */
export const getPendingPokes = async (userId: string): Promise<string[]> => {
  try {
    const pokesRef = ref(database, `pokes/${userId}`);
    const snapshot = await get(pokesRef);
```

```
138 |
139 |     if (!snapshot.exists()) {
140 |       return [];
141 |     }
142 |
143 |     return Object.keys(snapshot.val());
144 |   } catch (error) {
145 |     console.error("'L Error getting pending pokes:", error);
146 |     return [];
147 |   }
148 | };
149 |
150 | /**
151 |  * Listen to pending pokes in real-time
152 |  */
153 | export const listenToPokes = (
154 |   userId: string,
155 |   callback: (pokeUserIds: string[]) => void
156 | ): (() => void) => {
157 |   const pokesRef = ref(database, `pokes/${userId}`);
158 |
159 |   const unsubscribe = onValue(
160 |     pokesRef,
161 |     (snapshot: DataSnapshot) => {
162 |       const pokeUserIds = snapshot.exists() ? Object.keys(snapshot.val()) : [];
163 |       callback(pokeUserIds);
164 |     },
165 |     (error) => {
166 |       console.error("'L Error listening to pokes:", error);
167 |       callback([]);
168 |     }
169 |   );
170 |
171 |   return () => {
172 |     off(pokesRef);
173 |   };
174 | };
175 |
176 | /**
177 |  * Check if user has poked another user (for outgoing pokes)
178 |  * This checks if the current user has sent a poke to someone
179 |  */
180 | export const hasPokedUser = async (
181 |   fromUserId: string,
182 |   toUserId: string
183 | ): Promise<boolean> => {
184 |   try {
185 |     const pokeRef = ref(database, `pokes/${toUserId}/${fromUserId}`);
186 |     const snapshot = await get(pokeRef);
187 |     return snapshot.exists();
188 |   } catch (error) {
189 |     console.error("'L Error checking poke status:", error);
190 |     return false;
191 |   }
192 | };
193 |
194 |
```

File: src/services/userService.ts

Lines: 226

```
  1 | // User service for blocking, reporting, and moderation functions
  2 | import { ref, set, get, remove, update } from "firebase/database";
  3 | import { database } from "./firebase";
  4 | import { getAdminEmails } from "./adminService";
  5 | import { createNotification } from "./notificationService";
  6 | import { getUserData } from "./authService";
  7 |
  8 | // ============ COMMENTING SUSPENSION ============
  9 |
 10 | export interface CommentingSuspension {
 11 |   suspended: boolean;
 12 |   suspendedAt: number;
 13 |   suspendedUntil?: number; // Optional: for temporary suspensions
 14 |   reason: string;
 15 |   adminId: string;
 16 | }
 17 |
 18 | /**
 19 |  * Suspend a user's commenting privileges
 20 |  */
 21 | export const suspendCommenting = async (
 22 |   userId: string,
 23 |   adminId: string,
 24 |   reason: string,
 25 |   suspendedUntil?: number
 26 | ): Promise<void> => {
 27 |   try {
 28 |     const suspensionRef = ref(database, `users/${userId}/commentingSuspension`);
 29 |     const suspension: CommentingSuspension = {
 30 |       suspended: true,
 31 |       suspendedAt: Date.now(),
 32 |       suspendedUntil: suspendedUntil || undefined,
 33 |       reason,
 34 |       adminId
 35 |     };
 36 |
 37 |     await set(suspensionRef, suspension);
 38 |     console.log(`'  Commenting suspended for user ${userId}`);
 39 |   } catch (error) {
 40 |     console.error("'L Error suspending commenting:", error);
 41 |     throw error;
 42 |   }
 43 | };
 44 |
 45 | /**
 46 |  * Restore a user's commenting privileges
 47 |  */
 48 | export const restoreCommenting = async (userId: string): Promise<void> => {
 49 |   try {
 50 |     const suspensionRef = ref(database, `users/${userId}/commentingSuspension`);
 51 |     await remove(suspensionRef);
 52 |     console.log(`'  Commenting restored for user ${userId}`);
 53 |   } catch (error) {
 54 |     console.error("'L Error restoring commenting:", error);
 55 |     throw error;
 56 |   }
 57 | };
 58 |
 59 | /**
 60 |  * Check if a user's commenting is suspended
 61 |  * Returns null if not suspended, or the suspension details if suspended
 62 |  */
 63 | export const isCommentingSuspended = async (userId: string): Promise<CommentingSuspension |
null> => {
 64 |   try {
 65 |     const suspensionRef = ref(database, `users/${userId}/commentingSuspension`);
 66 |     const snapshot = await get(suspensionRef);
```

```
 67 |
 68 |     if (!snapshot.exists()) {
 69 |       return null;
 70 |     }
 71 |
 72 |     const suspension = snapshot.val() as CommentingSuspension;
 73 |
 74 |     // Check if temporary suspension has expired
 75 |     if (suspension.suspendedUntil && suspension.suspendedUntil < Date.now()) {
 76 |       // Auto-restore if suspension period has passed
 77 |       await restoreCommenting(userId);
 78 |       return null;
 79 |     }
 80 |
 81 |     return suspension.suspended ? suspension : null;
 82 |   } catch (error) {
 83 |     console.error("'L Error checking commenting suspension:", error);
 84 |     return null;
 85 |   }
 86 | };
 87 |
 88 | // ============ USER BLOCKING ============
 89 |
 90 | /**
 91 |  * Block a user - prevents them from messaging you and hides them from your view
 92 |  * Also checks for bidirectional blocking
 93 |  */
 94 | export const blockUser = async (userId: string, blockedUserId: string): Promise<void> => {
 95 |   try {
 96 |     // Check if the user to be blocked has already blocked the requester (bidirectional check)
 97 |     const reverseBlockRef = ref(database, `blockedUsers/${blockedUserId}/${userId}`);
 98 |     const reverseBlockSnapshot = await get(reverseBlockRef);
 99 |     const isMutuallyBlocked = reverseBlockSnapshot.exists();
100 |
101 |     // Block the user
102 |     const blockRef = ref(database, `blockedUsers/${userId}/${blockedUserId}`);
103 |     await set(blockRef, {
104 |       blockedAt: Date.now(),
105 |       blockedUserId,
106 |       isMutual: isMutuallyBlocked
107 |     });
108 |
109 |     // If reverse block exists, update it to mark as mutual
110 |     if (isMutuallyBlocked) {
111 |       const reverseBlockData = reverseBlockSnapshot.val();
112 |       await set(reverseBlockRef, {
113 |         ...reverseBlockData,
114 |         isMutual: true
115 |       });
116 |     }
117 |
118 |     console.log(`'  User ${blockedUserId} blocked by ${userId}${isMutuallyBlocked ? ' (mutual
    blocking)' : ''}`);
119 |   } catch (error) {
120 |     console.error("'L Error blocking user:", error);
121 |     throw error;
122 |   }
123 | };
124 |
125 | /**
126 |  * Unblock a user
127 |  */
128 | export const unblockUser = async (userId: string, blockedUserId: string): Promise<void> => {
129 |   try {
130 |     const blockRef = ref(database, `blockedUsers/${userId}/${blockedUserId}`);
131 |     await remove(blockRef);
132 |     console.log(`'  User ${blockedUserId} unblocked by ${userId}`);
133 |   } catch (error) {
134 |     console.error("'L Error unblocking user:", error);
135 |     throw error;
136 |   }
137 | };
```

```
138 |
139 | /**
140 |  * Check if a user is blocked
141 |  */
142 | export const isUserBlocked = async (userId: string, otherUserId: string): Promise<boolean> => {
143 |   try {
144 |     const blockRef = ref(database, `blockedUsers/${userId}/${otherUserId}`);
145 |     const snapshot = await get(blockRef);
146 |     return snapshot.exists();
147 |   } catch (error) {
148 |     console.error("'L Error checking if user is blocked:", error);
149 |     return false;
150 |   }
151 | };
152 |
153 | /**
154 |  * Get list of blocked user IDs for a user
155 |  */
156 | export const getBlockedUsers = async (userId: string): Promise<string[]> => {
157 |   try {
158 |     const blockedRef = ref(database, `blockedUsers/${userId}`);
159 |     const snapshot = await get(blockedRef);
160 |
161 |     if (!snapshot.exists()) {
162 |       return [];
163 |     }
164 |
165 |     return Object.keys(snapshot.val());
166 |   } catch (error) {
167 |     console.error("'L Error getting blocked users:", error);
168 |     return [];
169 |   }
170 | };
171 |
172 | /**
173 |  * Report a user
174 |  */
175 | export const reportUser = async (
176 |   reporterId: string,
177 |   reportedUserId: string,
178 |   reason: string,
179 |   details?: string
180 | ): Promise<void> => {
181 |   try {
182 |     const reportRef = ref(database, `reports/${Date.now()}_${reporterId}_${reportedUserId}`);
183 |     await set(reportRef, {
184 |       reporterId,
185 |       reportedUserId,
186 |       reason,
187 |       details: details || "",
188 |       reportedAt: Date.now()
189 |     });
190 |     console.log(`'  User ${reportedUserId} reported by ${reporterId}`);
191 |
192 |     // Notify all admins about the new report
193 |     try {
194 |       const adminEmails = await getAdminEmails();
195 |       const reporterData = await getUserData(reporterId);
196 |       const reportedUserData = await getUserData(reportedUserId);
197 |
198 |       // Create notifications for all admins
199 |       // Note: We need to get admin user IDs from emails - for now, we'll store in a special
admin_notifications node
200 |       // Or we can create a notification system that admins check
201 |       const adminNotificationsRef = ref(database, `adminNotifications`);
202 |       const notificationRef = ref(database, `adminNotifications/${Date.now()}_${reporterId}
_${reportedUserId}`);
203 |       await set(notificationRef, {
204 |         type: "report_submitted",
205 |         reporterId,
206 |         reportedUserId,
207 |         reporterName: reporterData?.name || reporterData?.username || "Unknown",
208 |         reportedUserName: reportedUserData?.name || reportedUserData?.username || "Unknown",
```

```
209 |         reason,
210 |         details: details || "",
211 |         timestamp: Date.now(),
212 |         read: false
213 |       });
214 |
215 |       console.log(`'  Admin notification created for report`);
216 |     } catch (notificationError) {
217 |       // Don't fail the report if notification fails
218 |       console.error("'L Error creating admin notification:", notificationError);
219 |     }
220 |   } catch (error) {
221 |     console.error("'L Error reporting user:", error);
222 |     throw error;
223 |   }
224 | };
225 |
226 |
```

File: src/services/venuePreferenceService.ts

Lines: 301

```
  1 | // Venue preference service - manages user venue and activity preferences
  2 | import { ref, set, get, onValue, off, DataSnapshot } from "firebase/database";
  3 | import { database } from "./firebase";
  4 | import { getUserData, updateUserProfile } from "./authService";
  5 |
  6 | export type Activity = "running" | "cycling" | "walking";
  7 |
  8 | export interface UserVenuePreferences {
  9 |   userId: string;
 10 |   venues: string[]; // Array of venue IDs
 11 |   activities: Activity[];
 12 |   updatedAt: number;
 13 | }
 14 |
 15 | export interface VenueUser {
 16 |   userId: string;
 17 |   username: string;
 18 |   avatar: string;
 19 |   activities: Activity[];
 20 | }
 21 |
 22 | /**
 23 |  * Save user venue preferences
 24 |  */
 25 | export const saveUserVenuePreferences = async (
 26 |   userId: string,
 27 |   venues: string[],
 28 |   activities: Activity[]
 29 | ): Promise<void> => {
 30 |   try {
 31 |     const preferencesRef = ref(database, `userVenuePreferences/${userId}`);
 32 |     const preferences: UserVenuePreferences = {
 33 |       userId,
 34 |       venues,
 35 |       activities,
 36 |       updatedAt: Date.now()
 37 |     };
 38 |
 39 |     await set(preferencesRef, preferences);
 40 |     console.log(`'  Saved venue preferences for user ${userId}`);
 41 |
 42 |     // Automatically enable profile visibility when user adds venues
 43 |     // This makes sense because adding venues indicates they want to be discoverable
 44 |     try {
 45 |       await updateUserProfile(userId, { profileVisible: true });
 46 |       console.log(`'  Enabled profile visibility for user ${userId}`);
 47 |     } catch (profileError) {
 48 |       // Log error but don't fail the entire operation if profile update fails
 49 |       console.error("'L Error enabling profile visibility:", profileError);
 50 |     }
 51 |   } catch (error) {
 52 |     console.error("'L Error saving venue preferences:", error);
 53 |     throw error;
 54 |   }
 55 | };
 56 |
 57 | /**
 58 |  * Get user venue preferences
 59 |  */
 60 | export const getUserVenuePreferences = async (
 61 |   userId: string
 62 | ): Promise<UserVenuePreferences | null> => {
 63 |   try {
 64 |     const preferencesRef = ref(database, `userVenuePreferences/${userId}`);
 65 |     const snapshot = await get(preferencesRef);
 66 |
```

```
 67 |     if (!snapshot.exists()) {
 68 |       return null;
 69 |     }
 70 |
 71 |     return snapshot.val() as UserVenuePreferences;
 72 |   } catch (error) {
 73 |     console.error("'L Error getting venue preferences:", error);
 74 |     throw error;
 75 |   }
 76 | };
 77 |
 78 | /**
 79 |  * Subscribe to a user's venue preferences in real-time
 80 |  */
 81 | export const listenToUserVenuePreferences = (
 82 |   userId: string,
 83 |   callback: (preferences: UserVenuePreferences | null) => void
 84 | ): (() => void) => {
 85 |   const preferencesRef = ref(database, `userVenuePreferences/${userId}`);
 86 |
 87 |   const handleValue = (snapshot: DataSnapshot) => {
 88 |     if (snapshot.exists()) {
 89 |       callback(snapshot.val() as UserVenuePreferences);
 90 |     } else {
 91 |       callback(null);
 92 |     }
 93 |   };
 94 |
 95 |   const handleError = (error: Error) => {
 96 |     console.error("'L Error listening to venue preferences:", error);
 97 |     callback(null);
 98 |   };
 99 |
100 |   onValue(preferencesRef, handleValue, handleError);
101 |
102 |   return () => {
103 |     off(preferencesRef, "value", handleValue);
104 |   };
105 | };
106 |
107 | /**
108 |  * Get users who have selected specific venues
109 |  */
110 | export const getUsersByVenues = async (
111 |   venueIds: string[]
112 | ): Promise<Record<string, VenueUser[]>> => {
113 |   try {
114 |     const preferencesRef = ref(database, `userVenuePreferences`);
115 |     const snapshot = await get(preferencesRef);
116 |
117 |     if (!snapshot.exists()) {
118 |       return {};
119 |     }
120 |
121 |     const allPreferences = snapshot.val() as Record<string, UserVenuePreferences>;
122 |     const usersByVenue: Record<string, VenueUser[]> = {};
123 |
124 |     // Initialize venue arrays
125 |     venueIds.forEach(venueId => {
126 |       usersByVenue[venueId] = [];
127 |     });
128 |
129 |     // Process each user's preferences
130 |     for (const [userId, preferences] of Object.entries(allPreferences)) {
131 |       if (!preferences || !preferences.venues || !preferences.activities) {
132 |         continue;
133 |       }
134 |
135 |       // Check which venues this user has selected
136 |       for (const venueId of preferences.venues) {
137 |         if (venueIds.includes(venueId)) {
```

```
          // Get user data for username and avatar
          try {
            const userData = await getUserData(userId);
            if (userData) {
              // Filter out users with profileVisible === false (default to visible if not set)
              if (userData.profileVisible === false) {
                console.log(`User ${userId} filtered out from venue ${venueId} - profileVisible:
false`);
                continue;
              }

              const venueUser: VenueUser = {
                userId,
                username: userData.username || userData.name || "User",
                avatar: userData.photoURL || userData.avatar || "",
                activities: preferences.activities
              };

              if (!usersByVenue[venueId]) {
                usersByVenue[venueId] = [];
              }
              usersByVenue[venueId].push(venueUser);
            }
          } catch (error) {
            console.error(`Error fetching user data for ${userId}:`, error);
          }
        }
      }
    }

    return usersByVenue;
  } catch (error) {
    console.error("'L Error getting users by venues:", error);
    throw error;
  }
};

/**
 * Listen to users by venues in real-time
 * Also listens to user profile changes to refresh when profileVisible changes
 */
export const listenToUsersByVenues = (
  venueIds: string[],
  callback: (usersByVenue: Record<string, VenueUser[]>) => void
): (() => void) => {
  try {
    const preferencesRef = ref(database, `userVenuePreferences`);
    const usersRef = ref(database, `users`);

    // Helper function to fetch and filter venue users
    const fetchVenueUsers = async (allPreferences: Record<string, UserVenuePreferences>):
Promise<void> => {
      const usersByVenue: Record<string, VenueUser[]> = {};

      // Initialize venue arrays
      venueIds.forEach(venueId => {
        usersByVenue[venueId] = [];
      });

      console.log(`[VenueService] Fetching venue users for ${venueIds.length} venues`);
      console.log(`[VenueService] Total users with preferences:
${Object.keys(allPreferences).length}`);

      // Process each user's preferences
      const userDataPromises: Promise<void>[] = [];

      for (const [userId, preferences] of Object.entries(allPreferences)) {
        if (!preferences || !preferences.venues || !preferences.activities) {
          continue;
        }

        // Check which venues this user has selected
        for (const venueId of preferences.venues) {
          if (venueIds.includes(venueId)) {
```

```
209 |             // Get user data for username and avatar
210 |             const promise = getUserData(userId)
211 |               .then((userData) => {
212 |                 if (userData) {
213 |                   // Filter out users with profileVisible === false (default to visible if not
    set)
214 |                   if (userData.profileVisible === false) {
215 |                     console.log(`[VenueService] User ${userId} filtered out from venue
    ${venueId} - profileVisible: fals...
216 |                     return;
217 |                   }
218 |
219 |                   console.log(`[VenueService] Adding user ${userId} (${userData.username ||
    userData.name}) to venue ${v...
220 |
221 |                   const venueUser: VenueUser = {
222 |                     userId,
223 |                     username: userData.username || userData.name || "User",
224 |                     avatar: userData.photoURL || userData.avatar || "",
225 |                     activities: preferences.activities
226 |                   };
227 |
228 |                   if (!usersByVenue[venueId]) {
229 |                     usersByVenue[venueId] = [];
230 |                   }
231 |                   usersByVenue[venueId].push(venueUser);
232 |                 } else {
233 |                   console.log(`[VenueService] No user data found for ${userId}`);
234 |                 }
235 |               })
236 |               .catch((error) => {
237 |                 console.error(`[VenueService] Error fetching user data for ${userId}:`, error);
238 |               });
239 |
240 |             userDataPromises.push(promise);
241 |           }
242 |         }
243 |       }
244 |
245 |       // Wait for all user data to be fetched
246 |       await Promise.all(userDataPromises);
247 |
248 |       // Log final counts per venue
249 |       venueIds.forEach(venueId => {
250 |         const count = usersByVenue[venueId]?.length || 0;
251 |         console.log(`[VenueService] Venue ${venueId} final count: ${count} users`);
252 |       });
253 |
254 |       console.log(`[VenueService] Final venue users object:`, usersByVenue);
255 |       callback(usersByVenue);
256 |     };
257 |
258 |     // Listener for user profile changes (to refresh when profileVisible changes)
259 |     let currentPreferences: Record<string, UserVenuePreferences> | null = null;
260 |
261 |     // Listener for venue preferences changes
262 |     const unsubscribePreferences = onValue(preferencesRef, async (snapshot: DataSnapshot) => {
263 |       if (!snapshot.exists()) {
264 |         console.log(`[VenueService] No venue preferences found`);
265 |         currentPreferences = null;
266 |         callback({});
267 |         return;
268 |       }
269 |
270 |       const allPreferences = snapshot.val() as Record<string, UserVenuePreferences>;
271 |       // Cache preferences for user profile change listener
272 |       currentPreferences = allPreferences;
273 |       console.log(`[VenueService] Venue preferences changed, fetching users...`);
274 |       await fetchVenueUsers(allPreferences);
275 |     }, (error) => {
276 |       console.error("'L Error listening to venue preferences:", error);
277 |       currentPreferences = null;
278 |       callback({});
279 |     });
```

```
280 |
281 |     const unsubscribeUsers = onValue(usersRef, async (snapshot: DataSnapshot) => {
282 |       // Only refresh if we have current preferences cached
283 |       if (currentPreferences) {
284 |         console.log(`[VenueService] User profile changed, refreshing venue users...`);
285 |         await fetchVenueUsers(currentPreferences);
286 |       }
287 |     }, (error) => {
288 |       console.error("'L Error listening to user profiles:", error);
289 |     });
290 |
291 |     return () => {
292 |       off(preferencesRef);
293 |       off(usersRef);
294 |     };
295 |   } catch (error) {
296 |     console.error("'L Error setting up listener:", error);
297 |     return () => {}; // Return empty unsubscribe function
298 |   }
299 | };
300 |
301 |
```

File: src/services/venueRequestService.ts

Lines: 121

```
  1 | // Venue request service - manages user venue requests
  2 | import { ref, set, push, get, onValue, off, update, DataSnapshot } from "firebase/database";
  3 | import { database } from "./firebase";
  4 |
  5 | export type VenueRequestStatus = "pending" | "approved" | "rejected";
  6 |
  7 | export interface VenueRequest {
  8 |   id: string;
  9 |   userId: string;
 10 |   venueName: string;
 11 |   location: string;
 12 |   status: VenueRequestStatus;
 13 |   createdAt: number;
 14 |   reviewedAt?: number;
 15 |   reviewedBy?: string;
 16 | }
 17 |
 18 | /**
 19 |  * Submit a venue request from a user
 20 |  */
 21 | export const submitVenueRequest = async (
 22 |   userId: string,
 23 |   venueName: string,
 24 |   location: string
 25 | ): Promise<string> => {
 26 |   try {
 27 |     const requestsRef = ref(database, `venueRequests`);
 28 |     const newRequestRef = push(requestsRef);
 29 |     const requestId = newRequestRef.key!;
 30 |
 31 |     const requestData: VenueRequest = {
 32 |       id: requestId,
 33 |       userId,
 34 |       venueName: venueName.trim(),
 35 |       location: location.trim(),
 36 |       status: "pending",
 37 |       createdAt: Date.now(),
 38 |     };
 39 |
 40 |     await set(newRequestRef, requestData);
 41 |     console.log(`'  Venue request submitted by user ${userId}: ${venueName}`);
 42 |     return requestId;
 43 |   } catch (error) {
 44 |     console.error("'L Error submitting venue request:", error);
 45 |     throw error;
 46 |   }
 47 | };
 48 |
 49 | /**
 50 |  * Get all venue requests (admin only)
 51 |  */
 52 | export const getAllVenueRequests = async (): Promise<VenueRequest[]> => {
 53 |   try {
 54 |     const requestsRef = ref(database, `venueRequests`);
 55 |     const snapshot = await get(requestsRef);
 56 |
 57 |     if (!snapshot.exists()) {
 58 |       return [];
 59 |     }
 60 |
 61 |     const requests = snapshot.val() as Record<string, VenueRequest>;
 62 |     return Object.values(requests).sort((a, b) => b.createdAt - a.createdAt);
 63 |   } catch (error) {
 64 |     console.error("'L Error getting venue requests:", error);
 65 |     throw error;
 66 |   }
```

```
};

/**
 * Listen to venue requests in real-time (admin only)
 */
export const listenToVenueRequests = (
  callback: (requests: VenueRequest[]) => void
): (() => void) => {
  const requestsRef = ref(database, `venueRequests`);

  const handleValue = (snapshot: DataSnapshot) => {
    if (snapshot.exists()) {
      const requests = snapshot.val() as Record<string, VenueRequest>;
      const requestsArray = Object.values(requests).sort((a, b) => b.createdAt - a.createdAt);
      callback(requestsArray);
    } else {
      callback([]);
    }
  };

  const handleError = (error: Error) => {
    console.error("'L Error listening to venue requests:", error);
    callback([]);
  };

  onValue(requestsRef, handleValue, handleError);

  return () => {
    off(requestsRef, "value", handleValue);
  };
};

/**
 * Update venue request status (admin only)
 */
export const updateVenueRequestStatus = async (
  requestId: string,
  status: VenueRequestStatus,
  reviewedBy: string
): Promise<void> => {
  try {
    const requestRef = ref(database, `venueRequests/${requestId}`);
    await update(requestRef, {
      status,
      reviewedAt: Date.now(),
      reviewedBy,
    });
    console.log(`'  Venue request ${requestId} updated to ${status}`);
  } catch (error) {
    console.error("'L Error updating venue request:", error);
    throw error;
  }
};


```

File: src/services/venueService.ts

Lines: 479

```
   1 | // Venue service - manages predefined venues for check-ins
   2 | import { ref, set, get, onValue, off, remove, DataSnapshot, push } from "firebase/database";
   3 | import { database } from "./firebase";
   4 |
   5 | export type VenueCategory = "park" | "university" | "commercial" | "sports" | "other";
   6 |
   7 | export interface Venue {
   8 |   id: string;
   9 |   name: string;
  10 |   description: string;
  11 |   lat: number;
  12 |   lng: number;
  13 |   radius: number; // in meters
  14 |   category: VenueCategory;
  15 |   city: string;
  16 | }
  17 |
  18 | /**
  19 |  * Predefined venues in the Philippines
  20 |  * These are popular locations for fitness activities
  21 |  */
  22 | const PREDEFINED_VENUES: Venue[] = [
  23 |   {
  24 |     id: "up-diliman",
  25 |     name: "UP Diliman",
  26 |     description: "2.2km Academic Oval loop with shaded tree-lined roads, popular for running,
cycling, and campus-based ...
  27 |     lat: 14.65527,
  28 |     lng: 121.0689,
  29 |     radius: 2200, // 2.2km loop
  30 |     category: "university",
  31 |     city: "Quezon City"
  32 |   },
  33 |   {
  34 |     id: "bgc",
  35 |     name: "Bonifacio Global City (BGC)",
  36 |     description: "Mixed-use district with clean, well-lit sidewalks, bike lanes, and pocket
parks, a major hub for runni...
  37 |     lat: 14.5547,
  38 |     lng: 121.0244,
  39 |     radius: 2000, // Large district
  40 |     category: "commercial",
  41 |     city: "Taguig"
  42 |   },
  43 |   {
  44 |     id: "ayala-triangle",
  45 |     name: "Ayala Triangle Gardens",
  46 |     description: "Small but scenic triangular park in Makati with ~1.3km runnable loop, safe and
busy even at night, ide...
  47 |     lat: 14.5564,
  48 |     lng: 121.0236,
  49 |     radius: 700, // ~1.3km loop
  50 |     category: "park",
  51 |     city: "Makati"
  52 |   },
  53 |   {
  54 |     id: "rizal-park",
  55 |     name: "Rizal Park / Luneta Park",
  56 |     description: "Historic urban park with ~2km loop and wide flat walkways, great for light
runs, walking, and casual c...
  57 |     lat: 14.58312,
  58 |     lng: 120.9794,
  59 |     radius: 1000, // ~2km loop
  60 |     category: "park",
  61 |     city: "Manila"
  62 |   },
  63 |   {
  64 |     id: "moa-complex",
  65 |     name: "MOA Complex / SM Mall of Asia By the Bay",
  66 |     description: "~2km seaside boulevard running and cycling route with Manila Bay views, flat
and bike-friendly roads t...
```

```
    lat: 14.5355,
    lng: 120.9820,
    radius: 2000, // ~2km route, can extend
    category: "commercial",
    city: "Pasay"
  },
  {
    id: "ccp-complex",
    name: "CCP Complex / Baywalk",
    description: "Part of the Manila Bay corridor with a dedicated Green Lane for runners and cyclists, popular for sunr...
    lat: 14.5600,
    lng: 120.9800,
    radius: 1500, // Part of Manila Bay corridor
    category: "other",
    city: "Pasay"
  },
  {
    id: "quezon-memorial",
    name: "Quezon Memorial Circle",
    description: "~2km perimeter path with lanes for joggers and cyclists, open green spaces, fitness stations, and freq...
    lat: 14.6506,
    lng: 121.0497,
    radius: 1000, // ~2km perimeter
    category: "park",
    city: "Quezon City"
  },
  {
    id: "ninoy-aquino-parks",
    name: "Ninoy Aquino Parks and Wildlife Center",
    description: "Nature-focused park in QC with shaded paths and quiet areas, suited for easy runs, walks, and casual c...
    lat: 14.6500,
    lng: 121.0500,
    radius: 1000, // Large nature park
    category: "park",
    city: "Quezon City"
  },
  {
    id: "marikina-river-park",
    name: "Marikina River Park",
    description: "Long riverside corridor with jogging and bike lanes, flat and peaceful, good for long easy runs, cycli...
    lat: 14.6500,
    lng: 121.1000,
    radius: 2000, // Long corridor
    category: "park",
    city: "Marikina"
  },
  {
    id: "filinvest-city",
    name: "Filinvest City / Spectrum Linear Park (Alabang)",
    description: "Tree-lined business district with wide roads and designated paths, used by both runners and cyclists f...
    lat: 14.4200,
    lng: 121.0400,
    radius: 1500, // Business district
    category: "commercial",
    city: "Muntinlupa"
  },
  {
    id: "commonwealth-avenue",
    name: "Commonwealth Avenue Bike Lane Network",
    description: "Major QC corridor with extensive protected bike lanes, used by cycling groups for distance rides while...
    lat: 14.7000,
    lng: 121.0800,
    radius: 3000, // Long corridor network
    category: "other",
    city: "Quezon City"
  },
  {
    id: "bgc-greenway-park",
    name: "BGC Greenway Park",
    description: "1.6km linear park with separate running and cycling lanes, the longest urban linear park in Metro Mani...
    lat: 14.5550,
```

```
    lng: 121.0250,
    radius: 800, // 1.6km linear park
    category: "park",
    city: "Taguig"
  },
  {
    id: "track-30th",
    name: "Track 30th (BGC)",
    description: "~310m soft-surface jogging loop with exercise lawns and meditation areas,
perfect for intervals or runni...
    lat: 14.5550,
    lng: 121.0250,
    radius: 200, // Small track
    category: "park",
    city: "Taguig"
  },
  {
    id: "terra-28th",
    name: "Terra 28th (BGC)",
    description: "Park with running paths, play areas, and outdoor bodyweight exercise stations,
ideal for runners who m...
    lat: 14.5550,
    lng: 121.0250,
    radius: 300, // Small park
    category: "park",
    city: "Taguig"
  },
  {
    id: "legaspi-active-park",
    name: "Legaspi Active Park (Makati)",
    description: "Pocket park with a short jogging loop and open green area, popular for morning
runs, yoga, and bodywei...
    lat: 14.5550,
    lng: 121.0200,
    radius: 300, // Pocket park
    category: "park",
    city: "Makati"
  },
  {
    id: "salcedo-park",
    name: "Salcedo Park / Jaime C. Velasquez Park (Makati)",
    description: "7,000sqm neighborhood park with walking/jogging paths and some exercise-
friendly space, good for easy ...
    lat: 14.5550,
    lng: 121.0200,
    radius: 150, // 7,000sqm park
    category: "park",
    city: "Makati"
  },
  {
    id: "greenbelt-park",
    name: "Greenbelt Park",
    description: "Garden oasis inside Greenbelt mall complex, better for light walks, steps, and
casual movement between...
    lat: 14.5544,
    lng: 121.0244,
    radius: 300, // Small garden oasis
    category: "park",
    city: "Makati"
  },
  {
    id: "arca-south",
    name: "Arca South (Taguig)",
    description: "Emerging mixed-use district with planned jogging paths, bike lanes, and open
spaces, used by runners a...
    lat: 14.5200,
    lng: 121.0500,
    radius: 1500, // Emerging district
    category: "commercial",
    city: "Taguig"
  },
  {
    id: "bridgetowne",
    name: "Bridgetowne (QC–Pasig)",
    description: "New destination estate with wide streets, riverfront areas, and an obstacle
park, used for fun runs an...
    lat: 14.6000,
    lng: 121.0700,
```

```
    radius: 1000, // Destination estate
    category: "commercial",
    city: "Quezon City"
  },
  {
    id: "philsports-arena",
    name: "PhilSports Arena (ULTRA)",
    description: "Professional 400m track in Pasig open at designated hours, best for structured speedwork, intervals a...
    lat: 14.5700,
    lng: 121.0700,
    radius: 500, // Sports complex
    category: "sports",
    city: "Pasig"
  },
  {
    id: "rizal-memorial-sports",
    name: "Rizal Memorial Sports Complex",
    description: "Historic sports complex in Malate with an oval track and facilities, suited for runners who prefer a c...
    lat: 14.5800,
    lng: 120.9800,
    radius: 800, // Sports complex
    category: "sports",
    city: "Manila"
  },
  {
    id: "fort-santiago",
    name: "Fort Santiago / Intramuros",
    description: "Historic walled city area with walkable/runnable cobblestone streets and ramparts, good for short runn...
    lat: 14.5900,
    lng: 120.9700,
    radius: 1000, // Historic area
    category: "other",
    city: "Manila"
  },
  {
    id: "pasig-river-esplanade",
    name: "Pasig River Esplanade",
    description: "Newly developed ~500m walkable and bikeable corridor along Pasig River (expanding to 26km total), scen...
    lat: 14.5800,
    lng: 121.0000,
    radius: 500, // Currently ~500m, expanding
    category: "other",
    city: "Manila"
  },
  {
    id: "eastwood-city",
    name: "Eastwood City (Libis, QC)",
    description: "Emerging commercial area in QC with dedicated jogging spots and cycling paths, good for casual runs an...
    lat: 14.6100,
    lng: 121.0800,
    radius: 1000, // Commercial area
    category: "commercial",
    city: "Quezon City"
  }
];

// Cache for venues from Firebase
let cachedVenues: Venue[] | null = null;
let venuesListener: (() => void) | null = null;

/**
 * Get all venues from Firebase, fallback to hardcoded list
 */
export const getAllVenues = async (): Promise<Venue[]> => {
  try {
    const venuesRef = ref(database, `venues`);
    const snapshot = await get(venuesRef);

    if (snapshot.exists()) {
      const venues = snapshot.val() as Record<string, Venue>;
      const venuesArray = Object.values(venues);
```

```
280 |       cachedVenues = venuesArray;
281 |       return venuesArray;
282 |     }
283 |   } catch (error) {
284 |     console.error("'L Error fetching venues from Firebase:", error);
285 |   }
286 |
287 |   // Fallback to hardcoded venues
288 |   return PREDEFINED_VENUES;
289 | };
290 |
291 | /**
292 |  * Get all venues synchronously (uses cache or hardcoded)
293 |  * For components that need immediate access
294 |  */
295 | export const getAllVenuesSync = (): Venue[] => {
296 |   return cachedVenues || PREDEFINED_VENUES;
297 | };
298 |
299 | /**
300 |  * Listen to venues in real-time
301 |  */
302 | export const listenToVenues = (
303 |   callback: (venues: Venue[]) => void
304 | ): (() => void) => {
305 |   const venuesRef = ref(database, `venues`);
306 |
307 |   const handleValue = (snapshot: DataSnapshot) => {
308 |     if (snapshot.exists()) {
309 |       const venues = snapshot.val() as Record<string, Venue>;
310 |       const venuesArray = Object.values(venues);
311 |       cachedVenues = venuesArray;
312 |       callback(venuesArray);
313 |     } else {
314 |       // Fallback to hardcoded venues
315 |       cachedVenues = PREDEFINED_VENUES;
316 |       callback(PREDEFINED_VENUES);
317 |     }
318 |   };
319 |
320 |   const handleError = (error: Error) => {
321 |     console.error("'L Error listening to venues:", error);
322 |     cachedVenues = PREDEFINED_VENUES;
323 |     callback(PREDEFINED_VENUES);
324 |   };
325 |
326 |   onValue(venuesRef, handleValue, handleError);
327 |
328 |   const unsubscribe = () => {
329 |     off(venuesRef, "value", handleValue);
330 |   };
331 |
332 |   venuesListener = unsubscribe;
333 |   return unsubscribe;
334 | };
335 |
336 | /**
337 |  * Get a venue by ID (uses cache or hardcoded)
338 |  */
339 | export const getVenueById = (id: string): Venue | undefined => {
340 |   const venues = cachedVenues || PREDEFINED_VENUES;
341 |   return venues.find(venue => venue.id === id);
342 | };
343 |
344 | /**
345 |  * Search venues by name (case-insensitive)
346 |  */
347 | export const searchVenues = (query: string): Venue[] => {
348 |   const venues = cachedVenues || PREDEFINED_VENUES;
349 |
350 |   if (!query.trim()) {
```

```
    return venues;
  }

  const lowerQuery = query.toLowerCase();
  return venues.filter(venue =>
    venue.name.toLowerCase().includes(lowerQuery) ||
    venue.description.toLowerCase().includes(lowerQuery) ||
    venue.city.toLowerCase().includes(lowerQuery)
  );
};

/**
 * Calculate distance between two points in kilometers
 */
function calculateDistance(lat1: number, lng1: number, lat2: number, lng2: number): number {
  const R = 6371; // Earth's radius in kilometers
  const dLat = (lat2 - lat1) * Math.PI / 180;
  const dLng = (lng2 - lng1) * Math.PI / 180;
  const a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
    Math.sin(dLng / 2) * Math.sin(dLng / 2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  return R * c;
}

/**
 * Get venues near a location within a specified radius (in kilometers)
 */
export const getVenuesNearby = (
  lat: number,
  lng: number,
  radiusKm: number = 10
): Venue[] => {
  const venues = cachedVenues || PREDEFINED_VENUES;
  return venues
    .map(venue => ({
      venue,
      distance: calculateDistance(lat, lng, venue.lat, venue.lng)
    }))
    .filter(({ distance }) => distance <= radiusKm)
    .sort((a, b) => a.distance - b.distance)
    .map(({ venue }) => venue);
};

/**
 * Find venue that matches an event location (by coordinates)
 * Returns venue if event is within venue radius
 */
export const findVenueForEvent = (
  eventLat: number,
  eventLng: number
): Venue | null => {
  const venues = cachedVenues || PREDEFINED_VENUES;
  for (const venue of venues) {
    const distance = calculateDistance(eventLat, eventLng, venue.lat, venue.lng);
    const distanceMeters = distance * 1000; // Convert to meters

    // Check if event is within venue radius
    if (distanceMeters <= venue.radius) {
      return venue;
    }
  }

  return null;
};

/**
 * Admin functions for managing venues
 */
```

```
422 | /**
423 |  * Add a new venue (admin only)
424 |  */
425 | export const addVenue = async (
426 |   venue: Omit<Venue, "id">,
427 |   createdBy: string
428 | ): Promise<string> => {
429 |   try {
430 |     const venuesRef = ref(database, `venues`);
431 |     const newVenueRef = push(venuesRef);
432 |     const venueId = newVenueRef.key!;
433 |
434 |     const venueData: Venue = {
435 |       ...venue,
436 |       id: venueId,
437 |     };
438 |
439 |     await set(newVenueRef, venueData);
440 |     console.log(`'  Venue added: ${venue.name}`);
441 |     return venueId;
442 |   } catch (error) {
443 |     console.error("'L Error adding venue:", error);
444 |     throw error;
445 |   }
446 | };
447 |
448 | /**
449 |  * Update an existing venue (admin only)
450 |  */
451 | export const updateVenue = async (
452 |   venueId: string,
453 |   updates: Partial<Venue>
454 | ): Promise<void> => {
455 |   try {
456 |     const venueRef = ref(database, `venues/${venueId}`);
457 |     await set(venueRef, { ...updates, id: venueId }, { merge: true });
458 |     console.log(`'  Venue updated: ${venueId}`);
459 |   } catch (error) {
460 |     console.error("'L Error updating venue:", error);
461 |     throw error;
462 |   }
463 | };
464 |
465 | /**
466 |  * Delete a venue (admin only)
467 |  */
468 | export const deleteVenue = async (venueId: string): Promise<void> => {
469 |   try {
470 |     const venueRef = ref(database, `venues/${venueId}`);
471 |     await remove(venueRef);
472 |     console.log(`'  Venue deleted: ${venueId}`);
473 |   } catch (error) {
474 |     console.error("'L Error deleting venue:", error);
475 |     throw error;
476 |   }
477 | };
478 |
479 |
```

File: src/services/weatherService.ts

Lines: 240

```
  1 | /**
  2 |  * Weather Service
  3 |  * Fetches weather data from OpenWeatherMap API
  4 |  */
  5 |
  6 | export interface WeatherData {
  7 |   temperature: number;
  8 |   condition: string;
  9 |   description: string;
 10 |   icon: string;
 11 |   humidity?: number;
 12 |   windSpeed?: number;
 13 |   location?: string;
 14 |   feelsLike?: number;
 15 | }
 16 |
 17 | export interface ForecastData {
 18 |   high: number;
 19 |   low: number;
 20 |   condition: string;
 21 |   description: string;
 22 |   icon: string;
 23 | }
 24 |
 25 | // Cache for weather data to reduce API calls
 26 | const weatherCache = new Map<string, { data: WeatherData; timestamp: number }>();
 27 | const forecastCache = new Map<string, { data: ForecastData; timestamp: number }>();
 28 | const CACHE_DURATION = 10 * 60 * 1000; // 10 minutes
 29 |
 30 | /**
 31 |  * Get cache key from coordinates
 32 |  */
 33 | const getCacheKey = (lat: number, lng: number): string => {
 34 |   // Round to 2 decimal places for cache key (approx 1km precision)
 35 |   return `${lat.toFixed(2)}_${lng.toFixed(2)}`;
 36 | };
 37 |
 38 | /**
 39 |  * Get current weather conditions
 40 |  */
 41 | export const getCurrentWeather = async (
 42 |   lat: number,
 43 |   lng: number
 44 | ): Promise<WeatherData> => {
 45 |   const apiKey = import.meta.env.VITE_OPENWEATHER_API_KEY;
 46 |
 47 |   if (!apiKey) {
 48 |     throw new Error("OpenWeatherMap API key is not configured");
 49 |   }
 50 |
 51 |   // Check cache
 52 |   const cacheKey = getCacheKey(lat, lng);
 53 |   const cached = weatherCache.get(cacheKey);
 54 |   if (cached && Date.now() - cached.timestamp < CACHE_DURATION) {
 55 |     return cached.data;
 56 |   }
 57 |
 58 |   try {
 59 |     const url = `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lng}
&appid=${apiKey}&units=metric`;
 60 |     console.log("Fetching weather from OpenWeatherMap API...");
 61 |     const response = await fetch(url);
 62 |
 63 |     if (!response.ok) {
 64 |       const errorText = await response.text();
 65 |       console.error("'L Weather API error:", response.status, errorText);
 66 |
```

```
      if (response.status === 401) {
        throw new Error("Invalid API key. Please check your OpenWeatherMap API key.");
      }
      if (response.status === 429) {
        throw new Error("API rate limit exceeded. Please try again later.");
      }
      throw new Error(`Weather API error (${response.status}): ${response.statusText}`);
    }

    const data = await response.json();

    const weatherData: WeatherData = {
      temperature: Math.round(data.main.temp),
      condition: data.weather[0].main,
      description: data.weather[0].description,
      icon: data.weather[0].icon,
      humidity: data.main.humidity,
      windSpeed: data.wind?.speed ? Math.round(data.wind.speed * 3.6) : undefined, // Convert m/s to km/h
      feelsLike: Math.round(data.main.feels_like),
    };

    // Cache the result
    weatherCache.set(cacheKey, {
      data: weatherData,
      timestamp: Date.now(),
    });

    return weatherData;
  } catch (error: any) {
    console.error("Error fetching current weather:", error);
    throw error;
  }
};

/**
 * Get today's weather forecast
 */
export const getTodayForecast = async (
  lat: number,
  lng: number
): Promise<ForecastData> => {
  const apiKey = import.meta.env.VITE_OPENWEATHER_API_KEY;

  if (!apiKey) {
    throw new Error("OpenWeatherMap API key is not configured");
  }

  // Check cache
  const cacheKey = getCacheKey(lat, lng);
  const cached = forecastCache.get(cacheKey);
  if (cached && Date.now() - cached.timestamp < CACHE_DURATION) {
    return cached.data;
  }

  try {
    const url = `https://api.openweathermap.org/data/2.5/forecast?lat=${lat}&lon=${lng}&appid=${apiKey}&units=metric&cnt=
    console.log("🌤️ Fetching weather forecast from OpenWeatherMap API...");
    const response = await fetch(url);

    if (!response.ok) {
      const errorText = await response.text();
      console.error("'L Weather Forecast API error:", response.status, errorText);

      if (response.status === 401) {
        throw new Error("Invalid API key. Please check your OpenWeatherMap API key.");
      }
      if (response.status === 429) {
        throw new Error("API rate limit exceeded. Please try again later.");
      }
      throw new Error(`Weather API error (${response.status}): ${response.statusText}`);
    }
```

```
138 |
139 |     const data = await response.json();
140 |
141 |     // Get today's forecasts (next 24 hours)
142 |     const today = new Date();
143 |     today.setHours(0, 0, 0, 0);
144 |     const tomorrow = new Date(today);
145 |     tomorrow.setDate(tomorrow.getDate() + 1);
146 |
147 |     const todayForecasts = data.list.filter((item: any) => {
148 |       const forecastDate = new Date(item.dt * 1000);
149 |       return forecastDate >= today && forecastDate < tomorrow;
150 |     });
151 |
152 |     if (todayForecasts.length === 0) {
153 |       // Fallback to first forecast if no today forecasts
154 |       const firstForecast = data.list[0];
155 |       const forecastData: ForecastData = {
156 |         high: Math.round(firstForecast.main.temp_max),
157 |         low: Math.round(firstForecast.main.temp_min),
158 |         condition: firstForecast.weather[0].main,
159 |         description: firstForecast.weather[0].description,
160 |         icon: firstForecast.weather[0].icon,
161 |       };
162 |
163 |       forecastCache.set(cacheKey, {
164 |         data: forecastData,
165 |         timestamp: Date.now(),
166 |       });
167 |
168 |       return forecastData;
169 |     }
170 |
171 |     // Calculate high and low from today's forecasts
172 |     const temps = todayForecasts.map((item: any) => item.main.temp);
173 |     const high = Math.round(Math.max(...temps));
174 |     const low = Math.round(Math.min(...temps));
175 |
176 |     // Use the most representative forecast (middle of day or most common condition)
177 |     const midDayForecast = todayForecasts[Math.floor(todayForecasts.length / 2)] ||
todayForecasts[0];
178 |
179 |     const forecastData: ForecastData = {
180 |       high,
181 |       low,
182 |       condition: midDayForecast.weather[0].main,
183 |       description: midDayForecast.weather[0].description,
184 |       icon: midDayForecast.weather[0].icon,
185 |     };
186 |
187 |     // Cache the result
188 |     forecastCache.set(cacheKey, {
189 |       data: forecastData,
190 |       timestamp: Date.now(),
191 |     });
192 |
193 |     return forecastData;
194 |   } catch (error: any) {
195 |     console.error("Error fetching weather forecast:", error);
196 |     throw error;
197 |   }
198 | };
199 |
200 | /**
201 |  * Convert temperature based on unit preference
202 |  */
203 | export const convertTemperature = (celsius: number, useMetric: boolean): { value: number; unit:
string } => {
204 |   if (useMetric) {
205 |     return { value: Math.round(celsius), unit: "°C" };
206 |   }
207 |   const fahrenheit = (celsius * 9) / 5 + 32;
208 |   return { value: Math.round(fahrenheit), unit: "°F" };
```

```
};

/**
 * Get weather icon component name or emoji based on condition
 */
export const getWeatherIcon = (condition: string, iconCode?: string): string => {
  const conditionLower = condition.toLowerCase();

  // Use OpenWeatherMap icon codes if available
  if (iconCode) {
    // Icon codes: 01d/01n = clear, 02d/02n = few clouds, etc.
    if (iconCode.startsWith("01")) return "& þ "; // Clear sky
    if (iconCode.startsWith("02")) return "&Å"; // Few clouds
    if (iconCode.startsWith("03") || iconCode.startsWith("04")) return "& þ "; // Clouds
    if (iconCode.startsWith("09") || iconCode.startsWith("10")) return "Ø<ß'þ "; // Rain
    if (iconCode.startsWith("11")) return "&Èþ "; // Thunderstorm
    if (iconCode.startsWith("13")) return "'Dþ "; // Snow
    if (iconCode.startsWith("50")) return "Ø<ß+þ "; // Mist
  }

  // Fallback to condition name
  if (conditionLower.includes("clear") || conditionLower.includes("sunny")) return "& þ ";
  if (conditionLower.includes("cloud")) return "& þ ";
  if (conditionLower.includes("rain") || conditionLower.includes("drizzle")) return "Ø<ß'þ ";
  if (conditionLower.includes("thunder") || conditionLower.includes("storm")) return "&Èþ ";
  if (conditionLower.includes("snow")) return "'Dþ ";
  if (conditionLower.includes("mist") || conditionLower.includes("fog")) return "Ø<ß+þ ";

  return "Ø<ß$þ "; // Default
};

```

File: src/services/workoutService.ts

Lines: 260

```
// Workout service for Firebase - saves and retrieves workout history
import { ref, set, get, onValue, off, push, DataSnapshot } from "firebase/database";
import { database } from "./firebase";
import { WorkoutHistory } from "@/contexts/UserContext";
import { getUserData } from "./authService";

export interface WorkoutWithUser {
  workout: WorkoutHistory;
  userId: string;
  userData: any;
}

/**
 * Save a workout to Firebase
 */
export const saveWorkout = async (
  userId: string,
  workout: Omit<WorkoutHistory, "id">
): Promise<string> => {
  try {
    const workoutsRef = ref(database, `workouts/${userId}`);
    const newWorkoutRef = push(workoutsRef);
    const workoutId = newWorkoutRef.key!;

    const workoutData = {
      ...workout,
      id: workoutId,
      date: workout.date.getTime(), // Convert Date to timestamp
      createdAt: Date.now()
    };

    await set(newWorkoutRef, workoutData);
    console.log(`'  Workout saved: ${workoutId}`);
    return workoutId;
  } catch (error) {
    console.error("'L Error saving workout:", error);
    throw error;
  }
};

/**
 * Get all workouts for a user
 */
export const getUserWorkouts = async (userId: string): Promise<WorkoutHistory[]> => {
  try {
    const workoutsRef = ref(database, `workouts/${userId}`);
    const snapshot = await get(workoutsRef);

    if (!snapshot.exists()) {
      return [];
    }

    const workouts = snapshot.val();
    return Object.values(workouts).map((w: any) => ({
      ...w,
      date: new Date(w.date), // Convert timestamp back to Date
    })) as WorkoutHistory[];
  } catch (error) {
    console.error("'L Error getting user workouts:", error);
    throw error;
  }
};

/**
 * Listen to user's workouts in real-time
 */
```

```
67 | export const listenToUserWorkouts = (
68 |   userId: string,
69 |   callback: (workouts: WorkoutHistory[]) => void
70 | ): (() => void) => {
71 |   const workoutsRef = ref(database, `workouts/${userId}`);
72 |
73 |   const unsubscribe = onValue(
74 |     workoutsRef,
75 |     (snapshot: DataSnapshot) => {
76 |       if (!snapshot.exists()) {
77 |         callback([]);
78 |         return;
79 |       }
80 |
81 |       const workouts = snapshot.val();
82 |       const workoutList = Object.values(workouts).map((w: any) => ({
83 |         ...w,
84 |         date: new Date(w.date),
85 |       })) as WorkoutHistory[];
86 |
87 |       callback(workoutList);
88 |     },
89 |     (error) => {
90 |       console.error("'L Error listening to workouts:", error);
91 |       callback([]);
92 |     }
93 |   );
94 |
95 |   return () => {
96 |     off(workoutsRef);
97 |   };
98 | };
99 |
100 | /**
101 |  * Get all workouts from all users
102 |  * @param currentUserId - Current user ID to filter out
103 |  * @returns Promise of workouts with user information
104 |  */
105 | export const getAllUsersWorkouts = async (
106 |   currentUserId: string
107 | ): Promise<WorkoutWithUser[]> => {
108 |   try {
109 |     const workoutsRef = ref(database, "workouts");
110 |     const snapshot = await get(workoutsRef);
111 |
112 |     if (!snapshot.exists()) {
113 |       return [];
114 |     }
115 |
116 |     const allWorkouts: WorkoutWithUser[] = [];
117 |     const workoutsData = snapshot.val();
118 |
119 |     // Iterate through all users' workouts
120 |     for (const userId in workoutsData) {
121 |       // Skip current user's workouts
122 |       if (userId === currentUserId) continue;
123 |
124 |       const userWorkouts = workoutsData[userId];
125 |       if (!userWorkouts) continue;
126 |
127 |       // Get user data
128 |       let userData: any = null;
129 |       try {
130 |         userData = await getUserData(userId);
131 |       } catch (error) {
132 |         console.error(`Error fetching user data for ${userId}:`, error);
133 |         continue;
134 |       }
135 |
136 |       if (!userData) continue;
137 |
```

```
138 |       // Process each workout
139 |       const workouts = Object.values(userWorkouts) as any[];
140 |       for (const workout of workouts) {
141 |         allWorkouts.push({
142 |           workout: {
143 |             ...workout,
144 |             date: new Date(workout.date), // Convert timestamp back to Date
145 |           } as WorkoutHistory,
146 |           userId,
147 |           userData,
148 |         });
149 |       }
150 |     }
151 |
152 |     // Sort by date (most recent first)
153 |     return allWorkouts.sort(
154 |       (a, b) => b.workout.date.getTime() - a.workout.date.getTime()
155 |     );
156 |   } catch (error) {
157 |     console.error("'L Error getting all users' workouts:", error);
158 |     return [];
159 |   }
160 | };
161 |
162 | /**
163 |  * Listen to all users' workouts in real-time
164 |  * @param currentUserId - Current user ID to filter out
165 |  * @param callback - Callback function that receives workouts with user data
166 |  * @returns Unsubscribe function
167 |  */
168 | export const listenToAllUsersWorkouts = (
169 |   currentUserId: string,
170 |   callback: (workouts: WorkoutWithUser[]) => void
171 | ): (() => void) => {
172 |   const workoutsRef = ref(database, "workouts");
173 |   const userDataCache: Record<string, any> = {};
174 |
175 |   const unsubscribe = onValue(
176 |     workoutsRef,
177 |     async (snapshot: DataSnapshot) => {
178 |       if (!snapshot.exists()) {
179 |         callback([]);
180 |         return;
181 |       }
182 |
183 |       const allWorkouts: WorkoutWithUser[] = [];
184 |       const workoutsData = snapshot.val();
185 |
186 |       // Collect all user IDs we need to fetch
187 |       const userIdsToFetch = new Set<string>();
188 |       for (const userId in workoutsData) {
189 |         if (userId === currentUserId) continue;
190 |         userIdsToFetch.add(userId);
191 |       }
192 |
193 |       // Fetch user data for all users (use cache when available)
194 |       const userDataPromises = Array.from(userIdsToFetch).map(async (userId) => {
195 |         if (userDataCache[userId]) {
196 |           return { userId, userData: userDataCache[userId] };
197 |         }
198 |
199 |         try {
200 |           const userData = await getUserData(userId);
201 |           if (userData) {
202 |             userDataCache[userId] = userData;
203 |             return { userId, userData };
204 |           }
205 |         } catch (error) {
206 |           console.error(`Error fetching user data for ${userId}:`, error);
207 |         }
208 |         return null;
```

```
209 |       });
210 |
211 |       const userDataResults = await Promise.all(userDataPromises);
212 |       const userDataMap = new Map<string, any>();
213 |       userDataResults.forEach((result) => {
214 |         if (result) {
215 |           userDataMap.set(result.userId, result.userData);
216 |         }
217 |       });
218 |
219 |       // Process workouts
220 |       for (const userId in workoutsData) {
221 |         if (userId === currentUserId) continue;
222 |
223 |         const userWorkouts = workoutsData[userId];
224 |         if (!userWorkouts) continue;
225 |
226 |         const userData = userDataMap.get(userId);
227 |         if (!userData) continue;
228 |
229 |         const workouts = Object.values(userWorkouts) as any[];
230 |         for (const workout of workouts) {
231 |           allWorkouts.push({
232 |             workout: {
233 |               ...workout,
234 |               date: new Date(workout.date),
235 |             } as WorkoutHistory,
236 |             userId,
237 |             userData,
238 |           });
239 |         }
240 |       }
241 |
242 |       // Sort by date (most recent first)
243 |       const sorted = allWorkouts.sort(
244 |         (a, b) => b.workout.date.getTime() - a.workout.date.getTime()
245 |       );
246 |
247 |       callback(sorted);
248 |     },
249 |     (error) => {
250 |       console.error("'L Error listening to all users' workouts:", error);
251 |       callback([]);
252 |     }
253 |   );
254 |
255 |   return () => {
256 |     off(workoutsRef);
257 |   };
258 | };
259 |
260 |
```

File: src/styles/animations.css

Lines: 57

```
/* Custom animations for PaceMatch */

/* Pulsing animation for activity badge */
@keyframes pulse {
  0%, 100% {
    opacity: 1;
  }
  50% {
    opacity: 0.5;
  }
}

.animate-pulse-slow {
  animation: pulse 2s infinite;
}

/* Glow animation for fitness level avatars */
@keyframes glow-pulse {
  0%, 100% {
    box-shadow: 0 0 12px rgba(59, 130, 246, 0.6), 0 0 24px rgba(59, 130, 246, 0.3);
  }
  50% {
    box-shadow: 0 0 16px rgba(59, 130, 246, 0.8), 0 0 32px rgba(59, 130, 246, 0.5);
  }
}

@keyframes glow-pulse-green {
  0%, 100% {
    box-shadow: 0 0 12px rgba(34, 197, 94, 0.6), 0 0 24px rgba(34, 197, 94, 0.3);
  }
  50% {
    box-shadow: 0 0 16px rgba(34, 197, 94, 0.8), 0 0 32px rgba(34, 197, 94, 0.5);
  }
}

@keyframes glow-pulse-purple {
  0%, 100% {
    box-shadow: 0 0 12px rgba(168, 85, 247, 0.6), 0 0 24px rgba(168, 85, 247, 0.3);
  }
  50% {
    box-shadow: 0 0 16px rgba(168, 85, 247, 0.8), 0 0 32px rgba(168, 85, 247, 0.5);
  }
}

.animate-glow-pulse-blue {
  animation: glow-pulse 3s ease-in-out infinite;
}

.animate-glow-pulse-green {
  animation: glow-pulse-green 3s ease-in-out infinite;
}

.animate-glow-pulse-purple {
  animation: glow-pulse-purple 3s ease-in-out infinite;
}

```

File: src/utils/anonymousName.ts

Lines: 43

```
  1 | /**
  2 |  * Get display name for a user - returns username if set, otherwise generates anonymous fallback
  3 |  * @param username - User's chosen username (from profile)
  4 |  * @param uid - Firebase user ID
  5 |  * @param activity - User's primary activity (running, cycling, walking)
  6 |  * @returns Display name to show to other users
  7 |  */
  8 | export function getDisplayName(
  9 |   username: string | null | undefined,
 10 |   uid: string,
 11 |   activity: string | null | undefined
 12 | ): string {
 13 |   // If username is set and not empty, use it
 14 |   if (username && username.trim()) {
 15 |     return username.trim();
 16 |   }
 17 |
 18 |   // Generate anonymous fallback name based on activity
 19 |   // Use last 4 characters of UID (converted to number) for uniqueness
 20 |   const uidSuffix = uid.slice(-4);
 21 |   // Convert hex characters to numbers (0-9, a-f -> 0-15)
 22 |   let numericSuffix = 0;
 23 |   for (let i = 0; i < uidSuffix.length; i++) {
 24 |     const char = uidSuffix[i];
 25 |     const value = parseInt(char, 16) || 0;
 26 |     numericSuffix = numericSuffix * 16 + value;
 27 |   }
 28 |   // Ensure it's a 3-digit number (0-999)
 29 |   numericSuffix = numericSuffix % 1000;
 30 |
 31 |   // Map activity to prefix
 32 |   const activityPrefix = (() => {
 33 |     const act = activity?.toLowerCase();
 34 |     if (act === "running") return "Runner";
 35 |     if (act === "cycling") return "Cyclist";
 36 |     if (act === "walking") return "Walker";
 37 |     return "Athlete";
 38 |   })();
 39 |
 40 |   return `${activityPrefix}${numericSuffix.toString().padStart(3, "0")}`;
 41 | }
 42 |
 43 |
```

File: src/utils/distance.ts

Lines: 165

```
  1 | // Distance calculation utilities using Haversine formula
  2 | import { getDistance } from 'geolib';
  3 |
  4 | export interface UserWithLocation {
  5 |   lat: number;
  6 |   lng: number;
  7 |   visible?: boolean;
  8 |   [key: string]: any;
  9 | }
 10 |
 11 | /**
 12 |  * Calculate distance between two coordinates in kilometers
 13 |  * @param {number} lat1 - Latitude of first point
 14 |  * @param {number} lng1 - Longitude of first point
 15 |  * @param {number} lat2 - Latitude of second point
 16 |  * @param {number} lng2 - Longitude of second point
 17 |  * @returns {number | null} Distance in kilometers
 18 |  */
 19 | export const calculateDistance = (
 20 |   lat1: number,
 21 |   lng1: number,
 22 |   lat2: number,
 23 |   lng2: number
 24 | ): number | null => {
 25 |   if (!lat1 || !lng1 || !lat2 || !lng2) {
 26 |     return null;
 27 |   }
 28 |
 29 |   // geolib uses meters, convert to kilometers
 30 |   const distanceInMeters = getDistance(
 31 |     { latitude: lat1, longitude: lng1 },
 32 |     { latitude: lat2, longitude: lng2 }
 33 |   );
 34 |
 35 |   return distanceInMeters / 1000; // Convert to kilometers
 36 | };
 37 |
 38 | /**
 39 |  * Format distance for display
 40 |  * @param {number | null} distanceKm - Distance in kilometers
 41 |  * @returns {string} Formatted distance string
 42 |  */
 43 | export const formatDistance = (distanceKm: number | null | undefined): string => {
 44 |   if (distanceKm === null || distanceKm === undefined) {
 45 |     return "Unknown";
 46 |   }
 47 |
 48 |   if (distanceKm < 1) {
 49 |     return `${Math.round(distanceKm * 1000)}m`;
 50 |   } else {
 51 |     return `${distanceKm.toFixed(1)}km`;
 52 |   }
 53 | };
 54 |
 55 | /**
 56 |  * Filter users by distance
 57 |  * @param {Record<string, UserWithLocation> | UserWithLocation[]} users - Object or array of
user objects with lat/lng
 58 |  * @param {number} userLat - Current user's latitude
 59 |  * @param {number} userLng - Current user's longitude
 60 |  * @param {number} maxDistanceKm - Maximum distance in kilometers
 61 |  * @param {string | null} excludeUserId - User ID to exclude from results (optional)
 62 |  * @param {boolean} requireActive - If true, only include users with active workouts (recent
timestamps within 3 min)
 63 |  * @returns {Array} Filtered users with distance property
 64 |  */
 65 | export const filterUsersByDistance = (
 66 |   users: Record<string, UserWithLocation> | UserWithLocation[],
```

```
  userLat: number,
  userLng: number,
  maxDistanceKm: number,
  excludeUserId: string | null = null,
  requireActive: boolean = false
): Array<UserWithLocation & { id: string; distance: number }> => {
  if (!userLat || !userLng) {
    console.log("& þ  No user location provided for distance filter");
    return [];
  }

  // Active workout threshold: 3 minutes
  const now = Date.now();
  const activeThreshold = 3 * 60 * 1000; // 3 minutes in milliseconds

  if (!users || (typeof users === 'object' && !Array.isArray(users) && Object.keys(users).length ===0)) {  console.log("& þ  No users provided to filter");
    return [];
  }

  // Convert object to entries if needed
  const userEntries: [string, UserWithLocation][] = Array.isArray(users)
    ? users.map((user, index) => [String(index), user])
    : Object.entries(users);

  console.log(`Filtering ${userEntries.length} users...`);

  const filtered = userEntries
    .map(([userId, userData]) => {
      // Exclude current user
      if (excludeUserId && userId === excludeUserId) {
        return null;
      }

      // Check if user has location
      if (!userData.lat || !userData.lng) {
        console.log(`User ${userId} filtered out - no location (lat: ${userData.lat}, lng: ${userData.lng})`);        return null;
      }

      // Only hide if visible is explicitly false (default to visible if not set)
      // This checks workout location visibility
      if (userData.visible === false) {
        console.log(`User ${userId} filtered out - visible: false`);
        return null;
      }

      // Check profile visibility for discovery (default to visible if not set)
      if (userData.profileVisible === false) {
        console.log(`User ${userId} filtered out - profileVisible: false`);
        return null;
      }

      // Check active workout status if required
      if (requireActive) {
        if (!userData.timestamp) {
          console.log(`User ${userId} filtered out - no timestamp (not actively tracking)`);
          return null;
        }

        const timeDiff = now - userData.timestamp;
        if (timeDiff > activeThreshold) {
          console.log(`User ${userId} filtered out - timestamp too old (${Math.round(timeDiff / 1000)}s ago, threshold: ${activeThreshold / 1000}s)`);          return null;
        }
      }

      const distance = calculateDistance(
        userLat,
        userLng,
        userData.lat,
```

```
138 |         userData.lng
139 |       );
140 |
141 |       if (distance === null) {
142 |         console.log(`User ${userId} - distance calculation failed`);
143 |         return null;
144 |       }
145 |
146 |       if (distance > maxDistanceKm) {
147 |         console.log(`User ${userId} filtered out - too far: ${distance.toFixed(2)}km (max:
${maxDistanceKm}km)`);
148 |         return null;
149 |       }
150 |
151 |       console.log(`'  User ${userId} included - distance: ${distance.toFixed(2)}km, visible:
${userData.visible}`);
152 |       return {
153 |         id: userId,
154 |         ...userData,
155 |         distance
156 |       };
157 |     })
158 |     .filter((user): user is UserWithLocation & { id: string; distance: number } => user !== null)
159 |     .sort((a, b) => a.distance - b.distance); // Sort by distance
160 |
161 |   console.log(`Distance filter result: ${filtered.length} users within ${maxDistanceKm}km`);
162 |   return filtered;
163 | };
164 |
165 |
```

File: src/utils/getLocationForWeather.ts

Lines: 77

```
/**
 * Utility function to get a one-time location for weather widget
 * Requests location permission and gets current position
 */
import { Geolocation } from "@capacitor/geolocation";
import { isNativePlatform } from "./platform";

export interface Location {
  lat: number;
  lng: number;
}

/**
 * Get current location for weather widget
 * Requests permission if needed and returns location
 */
export const getLocationForWeather = async (): Promise<Location | null> => {
  try {
    if (isNativePlatform()) {
      // Native platform (iOS/Android) - use Capacitor Geolocation
      try {
        // Request permissions first
        const permissionStatus = await Geolocation.requestPermissions();

        if (permissionStatus.location !== 'granted') {
          console.warn("& þ  Location permission denied for weather widget");
          return null;
        }

        // Get current position
        const position = await Geolocation.getCurrentPosition({
          enableHighAccuracy: false, // Use lower accuracy for faster response (weather doesn't need precise location)
          timeout: 10000
        });

        return {
          lat: position.coords.latitude,
          lng: position.coords.longitude,
        };
      } catch (error) {
        console.error("Error getting location (native):", error);
        return null;
      }
    } else {
      // Web platform - use browser Geolocation API
      if (!navigator.geolocation) {
        console.warn("& þ  Geolocation not supported in this browser");
        return null;
      }

      return new Promise<Location | null>((resolve) => {
        navigator.geolocation.getCurrentPosition(
          (position) => {
            resolve({
              lat: position.coords.latitude,
              lng: position.coords.longitude,
            });
          },
          (error) => {
            console.warn("& þ  Could not get location for weather widget:", error.message);
            resolve(null);
          },
          {
            enableHighAccuracy: false, // Use lower accuracy for faster response
            timeout: 10000,
            maximumAge: 300000 // Accept cached location up to 5 minutes old
```

```
67 |           }
68 |         );
69 |       });
70 |     }
71 |   } catch (error) {
72 |     console.error("Error in getLocationForWeather:", error);
73 |     return null;
74 |   }
75 | };
76 |
77 |
```

File: src/utils/mapIcons.ts

Lines: 755

```
/**
 * Utility functions for creating custom map marker icons from profile pictures
 */

export type FitnessLevel = "beginner" | "intermediate" | "pro";

interface IconOptions {
  size?: number;
  borderWidth?: number;
  borderColor?: string;
  shadow?: boolean;
  fitnessLevel?: FitnessLevel | string;
  opacity?: number; // Opacity for the entire icon (0.0 to 1.0)
  activity?: "running" | "cycling" | "walking"; // Activity type for badge
  enhancedGlow?: boolean; // Enhanced glow effect for 3D mode
}

const DEFAULT_SIZE = 48;
const DEFAULT_BORDER_WIDTH = 3;
const DEFAULT_BORDER_COLOR = '#ffffff';
const CACHE_SIZE = 100; // Limit cache size to prevent memory issues
const GLOW_SIZE = 8; // Additional pixels for glow effect around the avatar
const ACTIVITY_BADGE_SIZE = 18; // Size of activity icon badge
const ACTIVITY_BADGE_OFFSET = 4; // Offset from edge for badge positioning

// Cache for generated icons to avoid regenerating the same icon
const iconCache = new Map<string, string>();

/**
 * Gets the glow color for a fitness level
 * @param fitnessLevel - User's fitness level
 * @returns Object with glow color and border color
 */
function getFitnessLevelColors(fitnessLevel?: FitnessLevel | string): { glowColor: string; borderColor: string } | null {
  if (!fitnessLevel) return null;

  const normalizedLevel = (fitnessLevel as string).toLowerCase();

  switch (normalizedLevel) {
    case "beginner":
      return {
        glowColor: 'rgba(59, 130, 246, 0.6)', // Blue
        borderColor: '#3b82f6' // Blue border
      };
    case "intermediate":
      return {
        glowColor: 'rgba(34, 197, 94, 0.6)', // Green
        borderColor: '#22c55e' // Green border
      };
    case "pro":
      return {
        glowColor: 'rgba(168, 85, 247, 0.6)', // Purple
        borderColor: '#a855f7' // Purple border
      };
    default:
      return null;
  }
}

/**
 * Gets the color for an activity type
 * @param activity - Activity type
 * @returns Color hex code
 */
function getActivityColor(activity?: "running" | "cycling" | "walking"): string {
  switch (activity) {
```

```
    case "running":
      return "#22c55e"; // Green
    case "cycling":
      return "#3b82f6"; // Blue
    case "walking":
      return "#eab308"; // Yellow
    default:
      return "#6b7280"; // Gray
  }
}

/**
 * Draws an activity icon badge in the top-right corner
 * @param ctx - Canvas context
 * @param canvasSize - Total canvas size
 * @param avatarOffset - Offset for avatar positioning
 * @param activity - Activity type
 */
function drawActivityBadge(
  ctx: CanvasRenderingContext2D,
  canvasSize: number,
  avatarOffset: number,
  activity: "running" | "cycling" | "walking"
): void {
  const badgeSize = ACTIVITY_BADGE_SIZE;
  const badgeX = canvasSize - avatarOffset - ACTIVITY_BADGE_OFFSET - badgeSize / 2;
  const badgeY = avatarOffset + ACTIVITY_BADGE_OFFSET + badgeSize / 2;
  const badgeRadius = badgeSize / 2;
  const activityColor = getActivityColor(activity);

  // Draw white background circle
  ctx.beginPath();
  ctx.arc(badgeX, badgeY, badgeRadius, 0, Math.PI * 2);
  ctx.fillStyle = '#ffffff';
  ctx.fill();
  ctx.strokeStyle = activityColor;
  ctx.lineWidth = 2;
  ctx.stroke();

  // Draw activity icon based on type
  ctx.fillStyle = activityColor;
  ctx.strokeStyle = activityColor;
  ctx.lineWidth = 2;
  ctx.lineCap = 'round';
  ctx.lineJoin = 'round';

  switch (activity) {
    case "running":
      // Draw running person icon (simplified)
      drawRunningIcon(ctx, badgeX, badgeY, badgeRadius * 0.7);
      break;
    case "cycling":
      // Draw bike icon (simplified)
      drawBikeIcon(ctx, badgeX, badgeY, badgeRadius * 0.7);
      break;
    case "walking":
      // Draw walking person icon (simplified)
      drawWalkingIcon(ctx, badgeX, badgeY, badgeRadius * 0.7);
      break;
  }
}

/**
 * Draws a simplified running person icon
 */
function drawRunningIcon(ctx: CanvasRenderingContext2D, x: number, y: number, size: number):
void {
  // Head (circle)
  ctx.beginPath();
  ctx.arc(x, y - size * 0.4, size * 0.15, 0, Math.PI * 2);
  ctx.fill();

```

```
  // Body (line)
  ctx.beginPath();
  ctx.moveTo(x, y - size * 0.25);
  ctx.lineTo(x, y + size * 0.2);
  ctx.stroke();

  // Arms (running motion)
  ctx.beginPath();
  ctx.moveTo(x, y - size * 0.1);
  ctx.lineTo(x - size * 0.25, y + size * 0.1);
  ctx.moveTo(x, y - size * 0.1);
  ctx.lineTo(x + size * 0.25, y - size * 0.1);
  ctx.stroke();

  // Legs (running motion)
  ctx.beginPath();
  ctx.moveTo(x, y + size * 0.2);
  ctx.lineTo(x - size * 0.2, y + size * 0.5);
  ctx.moveTo(x, y + size * 0.2);
  ctx.lineTo(x + size * 0.2, y + size * 0.5);
  ctx.stroke();
}

/**
 * Draws a simplified bike icon
 */
function drawBikeIcon(ctx: CanvasRenderingContext2D, x: number, y: number, size: number): void {
  // Wheels (two circles)
  ctx.beginPath();
  ctx.arc(x - size * 0.2, y, size * 0.25, 0, Math.PI * 2);
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(x + size * 0.2, y, size * 0.25, 0, Math.PI * 2);
  ctx.stroke();

  // Frame (triangle-like shape)
  ctx.beginPath();
  ctx.moveTo(x - size * 0.2, y);
  ctx.lineTo(x, y - size * 0.3);
  ctx.lineTo(x + size * 0.2, y);
  ctx.stroke();
}

/**
 * Draws a simplified walking person icon
 */
function drawWalkingIcon(ctx: CanvasRenderingContext2D, x: number, y: number, size: number):
void {
  // Head (circle)
  ctx.beginPath();
  ctx.arc(x, y - size * 0.4, size * 0.15, 0, Math.PI * 2);
  ctx.fill();

  // Body (line)
  ctx.beginPath();
  ctx.moveTo(x, y - size * 0.25);
  ctx.lineTo(x, y + size * 0.2);
  ctx.stroke();

  // Arms (relaxed)
  ctx.beginPath();
  ctx.moveTo(x, y - size * 0.1);
  ctx.lineTo(x - size * 0.2, y);
  ctx.moveTo(x, y - size * 0.1);
  ctx.lineTo(x + size * 0.2, y);
  ctx.stroke();

  // Legs (walking)
  ctx.beginPath();
  ctx.moveTo(x, y + size * 0.2);
  ctx.lineTo(x - size * 0.15, y + size * 0.45);
  ctx.moveTo(x, y + size * 0.2);
```

```
  ctx.lineTo(x + size * 0.15, y + size * 0.45);
  ctx.stroke();
}

/**
 * Creates a circular profile picture icon for Google Maps markers
 * This is an async function that loads the image and returns a data URL
 * @param photoURL - URL of the profile picture, or null/undefined for fallback
 * @param name - User's name for fallback avatar generation
 * @param options - Optional styling options
 * @returns Promise that resolves to data URL of the generated icon
 */
export async function createProfileIconAsync(
  photoURL: string | null | undefined,
  name: string = 'User',
  options: IconOptions = {}
): Promise<string> {
  const {
    size = DEFAULT_SIZE,
    borderWidth = DEFAULT_BORDER_WIDTH,
    borderColor = DEFAULT_BORDER_COLOR,
    shadow = true,
    fitnessLevel,
    opacity = 1.0,
    activity,
    enhancedGlow = false
  } = options;

  // Get fitness level colors
  const fitnessColors = getFitnessLevelColors(fitnessLevel);
  const hasGlow = !!fitnessColors;

  // Increase canvas size to accommodate glow and activity badge
  // Enhanced glow in 3D mode uses larger glow size
  const glowSizeMultiplier = enhancedGlow ? 1.5 : 1;
  const effectiveGlowSize = enhancedGlow || hasGlow ? GLOW_SIZE * glowSizeMultiplier : 0;
  const badgeSpace = activity ? ACTIVITY_BADGE_SIZE + ACTIVITY_BADGE_OFFSET * 2 : 0;
  const canvasSize = (hasGlow || enhancedGlow) ? size + effectiveGlowSize * 2 + badgeSpace :
size + badgeSpace;
  const avatarOffset = (hasGlow || enhancedGlow) ? effectiveGlowSize : 0;
  // Avatar center is in the middle of the avatar area (not including badge space)
  const avatarCenter = avatarOffset + size / 2;

  // Create cache key including opacity and activity
  const cacheKey = `${photoURL || name}-${size}-${borderWidth}-${borderColor}-${shadow}-
${fitnessLevel || 'none'}-${opac...
  // Check cache first
  if (iconCache.has(cacheKey)) {
    return iconCache.get(cacheKey)!;
  }

  // Limit cache size - remove oldest entries if needed
  if (iconCache.size >= CACHE_SIZE) {
    const firstKey = iconCache.keys().next().value;
    iconCache.delete(firstKey);
  }

  const canvas = document.createElement('canvas');
  canvas.width = canvasSize;
  canvas.height = canvasSize;
  const ctx = canvas.getContext('2d');

  if (!ctx) {
    // Fallback to default icon if canvas is not supported
    return 'https://maps.google.com/mapfiles/ms/icons/red-dot.png';
  }

  // Apply opacity to entire icon if specified
  if (opacity < 1.0) {
    ctx.globalAlpha = opacity;
  }

```

```
280 |   // Draw fitness level glow if available (drawn first, behind everything)
281 |   if (hasGlow && fitnessColors) {
282 |     drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, fitnessColors.glowColor,
enhancedGlow);
283 |     } else if (enhancedGlow) {
284 |     // Even without fitness level, add white glow in 3D mode
285 |     drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, 'rgba(255, 255, 255, 0.6)', true);
286 |   }
287 |
288 |   // Draw shadow if enabled
289 |   if (shadow) {
290 |     ctx.shadowBlur = 4;
291 |     ctx.shadowOffsetX = 2;
292 |     ctx.shadowOffsetY = 2;
293 |     ctx.shadowColor = 'rgba(0, 0, 0, 0.3)';
294 |   }
295 |
296 |   // Draw border circle with fitness level color if available
297 |   const finalBorderColor = fitnessColors?.borderColor || borderColor;
298 |   ctx.beginPath();
299 |   ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth / 2, 0, Math.PI * 2);
300 |   ctx.fillStyle = finalBorderColor;
301 |   ctx.fill();
302 |
303 |   // Create clipping path for circular image
304 |   ctx.save();
305 |   ctx.beginPath();
306 |   ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth, 0, Math.PI * 2);
307 |   ctx.clip();
308 |
309 |   // If we have a photo URL, try to load and draw it
310 |   if (photoURL) {
311 |     try {
312 |       const img = await new Promise<HTMLImageElement>((resolve, reject) => {
313 |         const image = new Image();
314 |         image.crossOrigin = 'anonymous';
315 |         image.onload = () => resolve(image);
316 |         image.onerror = () => reject(new Error('Failed to load image'));
317 |         image.src = photoURL;
318 |       });
319 |
320 |       // Draw image centered and scaled to fill the circular area
321 |       const imgAspect = img.width / img.height;
322 |       const circleSize = size;
323 |       let drawWidth = circleSize;
324 |       let drawHeight = circleSize;
325 |       let drawX = avatarOffset;
326 |       let drawY = avatarOffset;
327 |
328 |       // Scale to cover the circle (maintain aspect ratio, fill entire circle)
329 |       if (imgAspect > 1) {
330 |         // Image is wider - scale height to circle, width extends
331 |         drawHeight = circleSize;
332 |         drawWidth = circleSize * imgAspect;
333 |         drawX = avatarOffset - (drawWidth - circleSize) / 2;
334 |       } else {
335 |         // Image is taller or square - scale width to circle, height extends
336 |         drawWidth = circleSize;
337 |         drawHeight = circleSize / imgAspect;
338 |         drawY = avatarOffset - (drawHeight - circleSize) / 2;
339 |       }
340 |
341 |       // Draw the image - clipping path will make it circular
342 |       ctx.drawImage(img, drawX, drawY, drawWidth, drawHeight);
343 |     } catch (error) {
344 |       // If image fails to load, use fallback
345 |       console.warn('Error loading profile image, using fallback:', error);
346 |       drawFallbackAvatar(ctx, name, size, avatarOffset, avatarOffset);
347 |     }
348 |   } else {
349 |     // No photo URL, use fallback avatar
350 |     drawFallbackAvatar(ctx, name, size, avatarOffset, avatarOffset);
```

```
  }

  ctx.restore();

  // Reset globalAlpha before drawing activity badge (badge should be fully opaque)
  if (opacity < 1.0) {
    ctx.globalAlpha = 1.0;
  }

  // Draw activity badge in top-right corner if activity is specified
  if (activity) {
    drawActivityBadge(ctx, canvasSize, avatarOffset, activity);
  }

  // Reset globalAlpha if it was changed
  if (opacity < 1.0) {
    ctx.globalAlpha = 1.0;
  }

  const dataUrl = canvas.toDataURL('image/png');
  iconCache.set(cacheKey, dataUrl);
  return dataUrl;
}

/**
 * Synchronous version that returns fallback immediately and updates when image loads
 * Use this for immediate rendering with fallback
 */
export function createProfileIcon(
  photoURL: string | null | undefined,
  name: string = 'User',
  options: IconOptions = {}
): string {
  const {
    size = DEFAULT_SIZE,
    borderWidth = DEFAULT_BORDER_WIDTH,
    borderColor = DEFAULT_BORDER_COLOR,
    fitnessLevel,
    opacity = 1.0,
    activity,
    enhancedGlow = false
  } = options;

  // Get fitness level colors
  const fitnessColors = getFitnessLevelColors(fitnessLevel);
  const hasGlow = !!fitnessColors;

  // Increase canvas size to accommodate glow and activity badge
  // Enhanced glow in 3D mode uses larger glow size
  const glowSizeMultiplier = enhancedGlow ? 1.5 : 1;
  const effectiveGlowSize = enhancedGlow || hasGlow ? GLOW_SIZE * glowSizeMultiplier : 0;
  const badgeSpace = activity ? ACTIVITY_BADGE_SIZE + ACTIVITY_BADGE_OFFSET * 2 : 0;
  const canvasSize = (hasGlow || enhancedGlow) ? size + effectiveGlowSize * 2 + badgeSpace :
size + badgeSpace;
  const avatarOffset = (hasGlow || enhancedGlow) ? effectiveGlowSize : 0;
  // Avatar center is in the middle of the avatar area (not including badge space)
  const avatarCenter = avatarOffset + size / 2;

  // Create cache key including opacity and activity
  const cacheKey = `${photoURL || name}-${size}-${borderWidth}-${borderColor}-${fitnessLevel ||
'none'}-${opacity}-${act...

  // Check cache first
  if (iconCache.has(cacheKey)) {
    return iconCache.get(cacheKey)!;
  }

  const canvas = document.createElement('canvas');
  canvas.width = canvasSize;
  canvas.height = canvasSize;
  const ctx = canvas.getContext('2d');

  if (!ctx) {
```

```
    return 'https://maps.google.com/mapfiles/ms/icons/red-dot.png';
  }

  // Apply opacity to entire icon if specified
  if (opacity < 1.0) {
    ctx.globalAlpha = opacity;
  }

  // Draw fitness level glow if available (drawn first, behind everything)
  if (hasGlow && fitnessColors) {
    drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, fitnessColors.glowColor,
enhancedGlow);
  } else if (enhancedGlow) {
    // Even without fitness level, add white glow in 3D mode
    drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, 'rgba(255, 255, 255, 0.6)', true);
  }

  // Draw border circle with fitness level color if available
  const finalBorderColor = fitnessColors?.borderColor || borderColor;
  ctx.beginPath();
  ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth / 2, 0, Math.PI * 2);
  ctx.fillStyle = finalBorderColor;
  ctx.fill();

  // Create clipping path for circular image
  ctx.save();
  ctx.beginPath();
  ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth, 0, Math.PI * 2);
  ctx.clip();

  // Draw fallback immediately
  drawFallbackAvatar(ctx, name, size, avatarOffset, avatarOffset);

  // If we have a photo URL, load it asynchronously and update cache
  // Note: We need to recreate the full drawing when image loads since context state is lost
  if (photoURL) {
    // Store canvas and context for async update
    const updateCanvas = async () => {
      try {
        // Recreate the full drawing with the loaded image
        const img = new Image();
        img.crossOrigin = 'anonymous';

        await new Promise<void>((resolve, reject) => {
          img.onload = () => resolve();
          img.onerror = () => reject(new Error('Failed to load image'));
          img.src = photoURL;
        });

        // Clear and redraw everything
        ctx.clearRect(0, 0, canvasSize, canvasSize);

        // Apply opacity to entire icon if specified
        if (opacity < 1.0) {
          ctx.globalAlpha = opacity;
        }

        // Redraw glow
        if (hasGlow && fitnessColors) {
          drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, fitnessColors.glowColor,
enhancedGlow); } else if (enhancedGlow) {
          // Even without fitness level, add white glow in 3D mode
          drawGlowEffect(ctx, avatarCenter, avatarCenter, size / 2, 'rgba(255, 255, 255, 0.6)',
true);
        }

        // Redraw border
        ctx.beginPath();
        ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth / 2, 0, Math.PI * 2);
        ctx.fillStyle = finalBorderColor;
        ctx.fill();

        // Recreate clipping path
```

```
493 |         ctx.beginPath();
494 |         ctx.arc(avatarCenter, avatarCenter, size / 2 - borderWidth, 0, Math.PI * 2);
495 |         ctx.clip();
496 |
497 |         // Draw image centered and scaled to fill circle
498 |         const imgAspect = img.width / img.height;
499 |         const circleSize = size;
500 |         let drawWidth = circleSize;
501 |         let drawHeight = circleSize;
502 |         let drawX = avatarOffset;
503 |         let drawY = avatarOffset;
504 |
505 |         if (imgAspect > 1) {
506 |           drawHeight = circleSize;
507 |           drawWidth = circleSize * imgAspect;
508 |           drawX = avatarOffset - (drawWidth - circleSize) / 2;
509 |         } else {
510 |           drawWidth = circleSize;
511 |           drawHeight = circleSize / imgAspect;
512 |           drawY = avatarOffset - (drawHeight - circleSize) / 2;
513 |         }
514 |
515 |         ctx.drawImage(img, drawX, drawY, drawWidth, drawHeight);
516 |         ctx.restore();
517 |
518 |         // Reset globalAlpha before drawing activity badge (badge should be fully opaque)
519 |         if (opacity < 1.0) {
520 |           ctx.globalAlpha = 1.0;
521 |         }
522 |
523 |         // Draw activity badge in top-right corner if activity is specified
524 |         if (activity) {
525 |           drawActivityBadge(ctx, canvasSize, avatarOffset, activity);
526 |         }
527 |
528 |         // Reset globalAlpha if it was changed
529 |         if (opacity < 1.0) {
530 |           ctx.globalAlpha = 1.0;
531 |         }
532 |
533 |         // Update cache with final image
534 |         const dataUrl = canvas.toDataURL('image/png');
535 |         iconCache.set(cacheKey, dataUrl);
536 |       } catch (error) {
537 |         // Keep fallback, just cache it
538 |         const dataUrl = canvas.toDataURL('image/png');
539 |         iconCache.set(cacheKey, dataUrl);
540 |       }
541 |     };
542 |
543 |     updateCanvas();
544 |   }
545 |
546 |   ctx.restore();
547 |
548 |   // Reset globalAlpha before drawing activity badge (badge should be fully opaque)
549 |   if (opacity < 1.0) {
550 |     ctx.globalAlpha = 1.0;
551 |   }
552 |
553 |   // Draw activity badge in top-right corner if activity is specified
554 |   if (activity) {
555 |     drawActivityBadge(ctx, canvasSize, avatarOffset, activity);
556 |   }
557 |
558 |   // Reset globalAlpha if it was changed
559 |   if (opacity < 1.0) {
560 |     ctx.globalAlpha = 1.0;
561 |   }
562 |
563 |   const dataUrl = canvas.toDataURL('image/png');
```

```
564 |   iconCache.set(cacheKey, dataUrl);
565 |   return dataUrl;
566 | }
567 |
568 | /**
569 |  * Draws a glow effect around a circle
570 |  * @param ctx - Canvas context
571 |  * @param x - Center X coordinate
572 |  * @param y - Center Y coordinate
573 |  * @param radius - Radius of the circle to glow around
574 |  * @param glowColor - Color of the glow (with alpha)
575 |  * @param enhanced - If true, adds enhanced glow with white outline for 3D mode
576 |  */
577 | function drawGlowEffect(
578 |   ctx: CanvasRenderingContext2D,
579 |   x: number,
580 |   y: number,
581 |   radius: number,
582 |   glowColor: string,
583 |   enhanced: boolean = false
584 | ): void {
585 |   // Draw multiple concentric circles with decreasing opacity for smooth glow
586 |   const glowLayers = enhanced ? 6 : 4; // More layers for enhanced glow
587 |   const maxGlowRadius = radius + (enhanced ? GLOW_SIZE * 1.5 : GLOW_SIZE);
588 |
589 |   // If enhanced, draw white outline first for better visibility in 3D mode
590 |   if (enhanced) {
591 |     ctx.beginPath();
592 |     ctx.arc(x, y, radius + GLOW_SIZE * 0.3, 0, Math.PI * 2);
593 |     ctx.strokeStyle = 'rgba(255, 255, 255, 0.9)';
594 |     ctx.lineWidth = 3;
595 |     ctx.stroke();
596 |
597 |     // Add outer white glow
598 |     for (let i = 0; i < 3; i++) {
599 |       const outlineRadius = radius + GLOW_SIZE * 0.5 + i * 2;
600 |       const outlineOpacity = 0.4 * (1 - i / 3);
601 |       ctx.beginPath();
602 |       ctx.arc(x, y, outlineRadius, 0, Math.PI * 2);
603 |       ctx.strokeStyle = `rgba(255, 255, 255, ${outlineOpacity})`;
604 |       ctx.lineWidth = 2;
605 |       ctx.stroke();
606 |     }
607 |   }
608 |
609 |   for (let i = 0; i < glowLayers; i++) {
610 |     const layerRadius = radius + (maxGlowRadius - radius) * (i + 1) / glowLayers;
611 |     const opacity = enhanced ? 0.8 * (1 - i / glowLayers) : 0.6 * (1 - i / glowLayers);
612 |
613 |     // Extract RGB from rgba string and create new rgba with adjusted opacity
614 |     const colorMatch = glowColor.match(/rgba?\((\d+),\s*(\d+),\s*(\d+)/);
615 |     if (colorMatch) {
616 |       const r = parseInt(colorMatch[1]);
617 |       const g = parseInt(colorMatch[2]);
618 |       const b = parseInt(colorMatch[3]);
619 |       const layerColor = `rgba(${r}, ${g}, ${b}, ${opacity})`;
620 |
621 |       ctx.beginPath();
622 |       ctx.arc(x, y, layerRadius, 0, Math.PI * 2);
623 |       ctx.fillStyle = layerColor;
624 |       ctx.fill();
625 |     }
626 |   }
627 | }
628 |
629 | /**
630 |  * Loads an image and draws it to the canvas context within a circular area
631 |  * The image will be centered and scaled to fill the circular clipping area
632 |  */
633 | function loadAndDrawImage(
634 |   ctx: CanvasRenderingContext2D,
```

```
635 |   photoURL: string,
636 |   size: number,
637 |   offsetX: number = 0,
638 |   offsetY: number = 0
639 | ): Promise<void> {
640 |   return new Promise((resolve, reject) => {
641 |     const img = new Image();
642 |     img.crossOrigin = 'anonymous';
643 |
644 |     img.onload = () => {
645 |       try {
646 |         // Calculate dimensions to fill the circle while maintaining aspect ratio
647 |         const imgAspect = img.width / img.height;
648 |         const circleSize = size;
649 |
650 |         let drawWidth = circleSize;
651 |         let drawHeight = circleSize;
652 |         let drawX = offsetX;
653 |         let drawY = offsetY;
654 |
655 |         // Scale image to cover the entire circle (cover behavior)
656 |         if (imgAspect > 1) {
657 |           // Image is wider than tall
658 |           drawHeight = circleSize;
659 |           drawWidth = circleSize * imgAspect;
660 |           drawX = offsetX - (drawWidth - circleSize) / 2;
661 |         } else {
662 |           // Image is taller than wide or square
663 |           drawWidth = circleSize;
664 |           drawHeight = circleSize / imgAspect;
665 |           drawY = offsetY - (drawHeight - circleSize) / 2;
666 |         }
667 |
668 |         // Draw the image centered and scaled to fill the circle
669 |         // The clipping path will make it circular
670 |         ctx.drawImage(img, drawX, drawY, drawWidth, drawHeight);
671 |         resolve();
672 |       } catch (error) {
673 |         reject(error);
674 |       }
675 |     };
676 |
677 |     img.onerror = () => {
678 |       reject(new Error('Failed to load image'));
679 |     };
680 |
681 |     img.src = photoURL;
682 |   });
683 | }
684 |
685 | /**
686 |  * Draws a fallback avatar with user's initials
687 |  * Note: This is called within a circular clipping path, so the shape will be circular
688 |  */
689 | function drawFallbackAvatar(
690 |   ctx: CanvasRenderingContext2D,
691 |   name: string,
692 |   size: number,
693 |   offsetX: number = 0,
694 |   offsetY: number = 0
695 | ): void {
696 |   // Background gradient - will be clipped to circle by the active clipping path
697 |   const centerX = offsetX + size / 2;
698 |   const centerY = offsetY + size / 2;
699 |   const radius = size / 2;
700 |
701 |   const gradient = ctx.createRadialGradient(centerX, centerY, 0, centerX, centerY, radius);
702 |   gradient.addColorStop(0, '#4f46e5'); // indigo
703 |   gradient.addColorStop(1, '#7c3aed'); // purple
704 |   ctx.fillStyle = gradient;
705 |
```

```
706 |   // Fill the entire area - clipping path will make it circular
707 |   ctx.fillRect(offsetX, offsetY, size, size);
708 |
709 |   // Draw initials
710 |   const initials = name
711 |     .split(' ')
712 |     .map(word => word.charAt(0))
713 |     .join('')
714 |     .toUpperCase()
715 |     .substring(0, 2);
716 |
717 |   ctx.fillStyle = '#ffffff';
718 |   ctx.font = `bold ${size * 0.4}px Arial`;
719 |   ctx.textAlign = 'center';
720 |   ctx.textBaseline = 'middle';
721 |   ctx.fillText(initials, centerX, centerY);
722 | }
723 |
724 | /**
725 |  * Creates a Google Maps icon object from a profile picture
726 |  * @param photoURL - URL of the profile picture
727 |  * @param name - User's name for fallback
728 |  * @param size - Size of the icon in pixels
729 |  * @param fitnessLevel - User's fitness level for glow effect
730 |  * @param opacity - Opacity for the entire icon (0.0 to 1.0)
731 |  * @param activity - Activity type for badge display
732 |  * @returns Google Maps icon configuration object
733 |  */
734 | export function createMapIcon(
735 |   photoURL: string | null | undefined,
736 |   name: string = 'User',
737 |   size: number = DEFAULT_SIZE,
738 |   fitnessLevel?: FitnessLevel | string,
739 |   opacity?: number,
740 |   activity?: "running" | "cycling" | "walking"
741 | ): google.maps.Icon {
742 |   const hasGlow = !!getFitnessLevelColors(fitnessLevel);
743 |   const badgeSpace = activity ? ACTIVITY_BADGE_SIZE + ACTIVITY_BADGE_OFFSET * 2 : 0;
744 |   const actualSize = hasGlow ? size + GLOW_SIZE * 2 + badgeSpace : size + badgeSpace;
745 |   const iconUrl = createProfileIcon(photoURL, name, { size, fitnessLevel, opacity, activity });
746 |
747 |   return {
748 |     url: iconUrl,
749 |     scaledSize: new google.maps.Size(actualSize, actualSize),
750 |     anchor: new google.maps.Point(actualSize / 2, actualSize / 2),
751 |     origin: new google.maps.Point(0, 0)
752 |   };
753 | }
754 |
755 |
```

File: src/utils/markerOverlap.ts

Lines: 352

```
  1 | /**
  2 |  * Utility functions for preventing marker overlaps on maps
  3 |  */
  4 |
  5 | export interface MarkerPosition {
  6 |   lat: number;
  7 |   lng: number;
  8 |   id: string;
  9 |   [key: string]: any; // Allow additional properties
 10 | }
 11 |
 12 | export interface AdjustedPosition extends MarkerPosition {
 13 |   adjustedLat: number;
 14 |   adjustedLng: number;
 15 | }
 16 |
 17 | /**
 18 |  * Configuration for overlap prevention
 19 |  */
 20 | interface OverlapConfig {
 21 |   minDistancePixels?: number; // Minimum distance between markers in pixels
 22 |   maxOffsetMeters?: number; // Maximum offset from original position in meters
 23 |   iterations?: number; // Number of iterations for the algorithm
 24 | }
 25 |
 26 | const DEFAULT_MIN_DISTANCE_PIXELS = 50; // 50 pixels minimum separation
 27 | const DEFAULT_MAX_OFFSET_METERS = 50; // Don't move markers more than 50 meters
 28 | const DEFAULT_ITERATIONS = 10; // Number of iterations for convergence
 29 |
 30 | /**
 31 |  * Converts a lat/lng coordinate to pixel coordinates on the map
 32 |  * @param lat - Latitude
 33 |  * @param lng - Longitude
 34 |  * @param map - Google Maps map instance
 35 |  * @returns Pixel coordinates {x, y} or null if map is not available
 36 |  */
 37 | function latLngToPixel(
 38 |   lat: number,
 39 |   lng: number,
 40 |   map: google.maps.Map | null
 41 | ): { x: number; y: number } | null {
 42 |   if (!map) return null;
 43 |
 44 |   const projection = map.getProjection();
 45 |   if (!projection) return null;
 46 |
 47 |   const scale = Math.pow(2, map.getZoom() || 15);
 48 |   const worldCoordinate = projection.fromLatLngToPoint(
 49 |     new google.maps.LatLng(lat, lng)
 50 |   );
 51 |
 52 |   const pixelCoordinate = {
 53 |     x: worldCoordinate.x * scale,
 54 |     y: worldCoordinate.y * scale
 55 |   };
 56 |
 57 |   return pixelCoordinate;
 58 | }
 59 |
 60 | /**
 61 |  * Converts pixel coordinates back to lat/lng
 62 |  * @param x - X pixel coordinate
 63 |  * @param y - Y pixel coordinate
 64 |  * @param map - Google Maps map instance
 65 |  * @returns Lat/lng coordinates or null if map is not available
 66 |  */
```

```
function pixelToLatLng(
  x: number,
  y: number,
  map: google.maps.Map | null
): { lat: number; lng: number } | null {
  if (!map) return null;

  const projection = map.getProjection();
  if (!projection) return null;

  const scale = Math.pow(2, map.getZoom() || 15);
  const worldCoordinate = {
    x: x / scale,
    y: y / scale
  };

  const latLng = projection.fromPointToLatLng(
    new google.maps.Point(worldCoordinate.x, worldCoordinate.y)
  );

  return {
    lat: latLng.lat(),
    lng: latLng.lng()
  };
}

/**
 * Calculates distance between two pixel coordinates
 */
function pixelDistance(
  p1: { x: number; y: number },
  p2: { x: number; y: number }
): number {
  const dx = p2.x - p1.x;
  const dy = p2.y - p1.y;
  return Math.sqrt(dx * dx + dy * dy);
}

/**
 * Calculates distance between two lat/lng points in meters
 */
function haversineDistance(
  lat1: number,
  lng1: number,
  lat2: number,
  lng2: number
): number {
  const R = 6371000; // Earth's radius in meters
  const dLat = ((lat2 - lat1) * Math.PI) / 180;
  const dLng = ((lng2 - lng1) * Math.PI) / 180;
  const a =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos((lat1 * Math.PI) / 180) *
      Math.cos((lat2 * Math.PI) / 180) *
      Math.sin(dLng / 2) *
      Math.sin(dLng / 2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  return R * c;
}

/**
 * Prevents marker overlaps by adjusting positions
 * Uses a simple repulsion algorithm: markers that are too close push each other apart
 * @param markers - Array of marker positions
 * @param map - Google Maps map instance
 * @param config - Configuration options
 * @returns Array of markers with adjusted positions
 */
export function preventMarkerOverlap(
  markers: MarkerPosition[],
  map: google.maps.Map | null,
```

```
138 |   config: OverlapConfig = {}
139 | ): AdjustedPosition[] {
140 |   if (!map || markers.length === 0) {
141 |     return markers.map(m => ({
142 |       ...m,
143 |       adjustedLat: m.lat,
144 |       adjustedLng: m.lng
145 |     }));
146 |   }
147 |
148 |   const {
149 |     minDistancePixels = DEFAULT_MIN_DISTANCE_PIXELS,
150 |     maxOffsetMeters = DEFAULT_MAX_OFFSET_METERS,
151 |     iterations = DEFAULT_ITERATIONS
152 |   } = config;
153 |
154 |   // Convert all markers to pixel coordinates
155 |   const pixelMarkers = markers.map(marker => {
156 |     const pixel = latLngToPixel(marker.lat, marker.lng, map);
157 |     return {
158 |       ...marker,
159 |       pixelX: pixel?.x ?? 0,
160 |       pixelY: pixel?.y ?? 0,
161 |       adjustedX: pixel?.x ?? 0,
162 |       adjustedY: pixel?.y ?? 0
163 |     };
164 |   });
165 |
166 |   // Apply repulsion algorithm iteratively
167 |   for (let iter = 0; iter < iterations; iter++) {
168 |     for (let i = 0; i < pixelMarkers.length; i++) {
169 |       for (let j = i + 1; j < pixelMarkers.length; j++) {
170 |         const marker1 = pixelMarkers[i];
171 |         const marker2 = pixelMarkers[j];
172 |
173 |         const distance = pixelDistance(
174 |           { x: marker1.adjustedX, y: marker1.adjustedY },
175 |           { x: marker2.adjustedX, y: marker2.adjustedY }
176 |         );
177 |
178 |         // If markers are too close, push them apart
179 |         if (distance < minDistancePixels && distance > 0) {
180 |           const overlap = minDistancePixels - distance;
181 |           const pushDistance = overlap / 2; // Split the push between both markers
182 |
183 |           // Calculate direction vector
184 |           const dx = marker2.adjustedX - marker1.adjustedX;
185 |           const dy = marker2.adjustedY - marker1.adjustedY;
186 |           const length = Math.sqrt(dx * dx + dy * dy);
187 |
188 |           if (length > 0) {
189 |             // Normalize direction
190 |             const unitX = dx / length;
191 |             const unitY = dy / length;
192 |
193 |             // Push markers apart
194 |             marker1.adjustedX -= unitX * pushDistance;
195 |             marker1.adjustedY -= unitY * pushDistance;
196 |             marker2.adjustedX += unitX * pushDistance;
197 |             marker2.adjustedY += unitY * pushDistance;
198 |           }
199 |         }
200 |       }
201 |     }
202 |   }
203 |
204 |   // Convert back to lat/lng and apply distance constraint
205 |   const result: AdjustedPosition[] = pixelMarkers.map(marker => {
206 |     const originalLatLng = pixelToLatLng(
207 |       marker.pixelX,
208 |       marker.pixelY,
```

```
209 |       map
210 |     );
211 |     const adjustedLatLng = pixelToLatLng(
212 |       marker.adjustedX,
213 |       marker.adjustedY,
214 |       map
215 |     );
216 |
217 |     if (!originalLatLng || !adjustedLatLng) {
218 |       return {
219 |         ...marker,
220 |         adjustedLat: marker.lat,
221 |         adjustedLng: marker.lng
222 |       };
223 |     }
224 |
225 |     // Check if adjustment is within max offset
226 |     const offsetMeters = haversineDistance(
227 |       originalLatLng.lat,
228 |       originalLatLng.lng,
229 |       adjustedLatLng.lat,
230 |       adjustedLatLng.lng
231 |     );
232 |
233 |     // If offset is too large, use original position
234 |     if (offsetMeters > maxOffsetMeters) {
235 |       return {
236 |         ...marker,
237 |         adjustedLat: marker.lat,
238 |         adjustedLng: marker.lng
239 |       };
240 |     }
241 |
242 |     return {
243 |       ...marker,
244 |       adjustedLat: adjustedLatLng.lat,
245 |       adjustedLng: adjustedLatLng.lng
246 |     };
247 |   });
248 |
249 |   return result;
250 | }
251 |
252 | /**
253 |  * Simplified version that works with zoom level instead of map instance
254 |  * Useful when map instance is not immediately available
255 |  */
256 | export function preventMarkerOverlapSimple(
257 |   markers: MarkerPosition[],
258 |   zoom: number,
259 |   center: { lat: number; lng: number },
260 |   config: OverlapConfig = {}
261 | ): AdjustedPosition[] {
262 |   if (markers.length === 0) {
263 |     return markers.map(m => ({
264 |       ...m,
265 |       adjustedLat: m.lat,
266 |       adjustedLng: m.lng
267 |     }));
268 |   }
269 |
270 |   const {
271 |     minDistancePixels = DEFAULT_MIN_DISTANCE_PIXELS,
272 |     maxOffsetMeters = DEFAULT_MAX_OFFSET_METERS,
273 |     iterations = DEFAULT_ITERATIONS
274 |   } = config;
275 |
276 |   // Approximate pixel conversion using zoom level
277 |   // At zoom level z, 1 degree latitude "H 256 * 2^z pixels
278 |   const pixelsPerDegree = 256 * Math.pow(2, zoom);
279 |   const metersPerDegreeLat = 111320; // Approximately constant
```

```
  const metersPerDegreeLng = 111320 * Math.cos((center.lat * Math.PI) / 180);

  // Convert to pixel space
  const pixelMarkers = markers.map(marker => {
    const pixelX = (marker.lng - center.lng) * pixelsPerDegree * Math.cos((center.lat * Math.PI) / 180);
    const pixelY = (marker.lat - center.lat) * pixelsPerDegree;
    return {
      ...marker,
      pixelX,
      pixelY,
      adjustedX: pixelX,
      adjustedY: pixelY
    };
  });

  // Apply repulsion
  for (let iter = 0; iter < iterations; iter++) {
    for (let i = 0; i < pixelMarkers.length; i++) {
      for (let j = i + 1; j < pixelMarkers.length; j++) {
        const marker1 = pixelMarkers[i];
        const marker2 = pixelMarkers[j];

        const dx = marker2.adjustedX - marker1.adjustedX;
        const dy = marker2.adjustedY - marker1.adjustedY;
        const distance = Math.sqrt(dx * dx + dy * dy);

        if (distance < minDistancePixels && distance > 0) {
          const overlap = minDistancePixels - distance;
          const pushDistance = overlap / 2;
          const unitX = dx / distance;
          const unitY = dy / distance;

          marker1.adjustedX -= unitX * pushDistance;
          marker1.adjustedY -= unitY * pushDistance;
          marker2.adjustedX += unitX * pushDistance;
          marker2.adjustedY += unitY * pushDistance;
        }
      }
    }
  }

  // Convert back to lat/lng
  const result: AdjustedPosition[] = pixelMarkers.map(marker => {
    const adjustedLat = center.lat + marker.adjustedY / pixelsPerDegree;
    const adjustedLng = center.lng + marker.adjustedX / (pixelsPerDegree * Math.cos((center.lat * Math.PI) / 180));
    // Check offset constraint
    const offsetMeters = haversineDistance(
      marker.lat,
      marker.lng,
      adjustedLat,
      adjustedLng
    );

    if (offsetMeters > maxOffsetMeters) {
      return {
        ...marker,
        adjustedLat: marker.lat,
        adjustedLng: marker.lng
      };
    }

    return {
      ...marker,
      adjustedLat,
      adjustedLng
    };
  });

  return result;
}
```

351 |
352 |

File: src/utils/navigation.ts

Lines: 40

```
// Navigation utility for opening Google Maps

/**
 * Open Google Maps navigation to a specific location
 * Opens the Google Maps app externally (outside the main app)
 * Works on both mobile (opens app) and web (opens in new tab)
 *
 * Uses universal Google Maps URL that works reliably across all platforms.
 * On mobile devices, this will open in the Google Maps app if installed,
 * otherwise it opens in the browser.
 *
 * @param lat - Latitude
 * @param lng - Longitude
 */
export const openGoogleMapsNavigation = (lat: number, lng: number): void => {
  try {
    // Universal Google Maps URL that works across all platforms
    // This URL will:
    // - Open in Google Maps app if installed (on mobile)
    // - Open in browser if app not installed
    // - Work on iOS, Android, and Web without errors
    const mapsUrl = `https://www.google.com/maps/dir/?api=1&destination=${lat},${lng}`;

    // Open in new window/tab (or app if available on mobile)
    // Using _blank ensures it opens externally, not in the current app
    window.open(mapsUrl, '_blank', 'noopener,noreferrer');
  } catch (error) {
    // Silent fallback - don't log errors to avoid console noise
    // If window.open fails, try direct navigation
    try {
      const fallbackUrl = `https://www.google.com/maps/dir/?api=1&destination=${lat},${lng}`;
      window.location.href = fallbackUrl;
    } catch (fallbackError) {
      // Last resort: silently fail (user can manually open maps)
      // Don't log to avoid console errors
    }
  }
};

```

File: src/utils/platform.ts

Lines: 48

```
  1 | /**
  2 |  * Platform detection utilities for Capacitor
  3 |  * Helps determine if the app is running on native mobile or web
  4 |  */
  5 |
  6 | import { Capacitor } from '@capacitor/core';
  7 |
  8 | /**
  9 |  * Check if the app is running on a native mobile platform (iOS or Android)
 10 |  * @returns {boolean} True if running on native platform
 11 |  */
 12 | export const isNativePlatform = (): boolean => {
 13 |   return Capacitor.isNativePlatform();
 14 | };
 15 |
 16 | /**
 17 |  * Get the current platform name
 18 |  * @returns {'ios' | 'android' | 'web'} The platform name
 19 |  */
 20 | export const getPlatform = (): 'ios' | 'android' | 'web' => {
 21 |   return Capacitor.getPlatform() as 'ios' | 'android' | 'web';
 22 | };
 23 |
 24 | /**
 25 |  * Check if running on iOS
 26 |  * @returns {boolean} True if running on iOS
 27 |  */
 28 | export const isIOS = (): boolean => {
 29 |   return getPlatform() === 'ios';
 30 | };
 31 |
 32 | /**
 33 |  * Check if running on Android
 34 |  * @returns {boolean} True if running on Android
 35 |  */
 36 | export const isAndroid = (): boolean => {
 37 |   return getPlatform() === 'android';
 38 | };
 39 |
 40 | /**
 41 |  * Check if running on web browser
 42 |  * @returns {boolean} True if running on web
 43 |  */
 44 | export const isWeb = (): boolean => {
 45 |   return getPlatform() === 'web';
 46 | };
 47 |
 48 |
```

File: src/utils/profilePicture.ts

Lines: 55

```
  1 | /**
  2 |  * Profile Picture Utility
  3 |  * Provides consistent profile picture retrieval across the app
  4 |  * Prioritizes: photoURL !’ avatar !’ generated avatar
  5 |  */
  6 |
  7 | import { generateUserAvatar } from "@/lib/avatars";
  8 |
  9 | /**
 10 |  * Get profile picture URL with consistent fallback logic
 11 |  * @param photoURL - Primary profile picture URL from Firebase (photoURL field)
 12 |  * @param avatar - Secondary avatar URL (avatar field)
 13 |  * @param name - User's name or username for fallback avatar generation
 14 |  * @returns Profile picture URL (never returns empty string)
 15 |  */
 16 | export function getProfilePictureUrl(
 17 |   photoURL?: string | null,
 18 |   avatar?: string | null,
 19 |   name?: string | null
 20 | ): string {
 21 |   // Priority 1: Use photoURL if available
 22 |   if (photoURL && photoURL.trim() !== "") {
 23 |     return photoURL;
 24 |   }
 25 |
 26 |   // Priority 2: Use avatar if available
 27 |   if (avatar && avatar.trim() !== "") {
 28 |     return avatar;
 29 |   }
 30 |
 31 |   // Priority 3: Generate avatar from name/username
 32 |   const displayName = name || "User";
 33 |   return generateUserAvatar(displayName);
 34 | }
 35 |
 36 | /**
 37 |  * Get profile picture URL from user data object
 38 |  * Handles various user data structures used throughout the app
 39 |  * @param userData - User data object (can have photoURL, avatar, name, username fields)
 40 |  * @returns Profile picture URL (never returns empty string)
 41 |  */
 42 | export function getProfilePictureFromUserData(userData: {
 43 |   photoURL?: string | null;
 44 |   avatar?: string | null;
 45 |   name?: string | null;
 46 |   username?: string | null;
 47 | }): string {
 48 |   return getProfilePictureUrl(
 49 |     userData.photoURL,
 50 |     userData.avatar,
 51 |     userData.name || userData.username || undefined
 52 |   );
 53 | }
 54 |
 55 |
```

File: src/utils/safeArea.ts

Lines: 97

```
  1 | /**
  2 |  * Safe Area utilities for mobile devices
  3 |  * Provides JavaScript-based safe area insets as fallback when CSS env() doesn't work
  4 |  */
  5 |
  6 | import React from 'react';
  7 | import { Capacitor } from '@capacitor/core';
  8 | import { StatusBar, Style } from '@capacitor/status-bar';
  9 |
 10 | /**
 11 |  * Initialize StatusBar for edge-to-edge display
 12 |  * This enables safe area insets to work properly
 13 |  */
 14 | export const initializeStatusBar = async () => {
 15 |   if (!Capacitor.isNativePlatform()) {
 16 |     return; // Only needed on native platforms
 17 |   }
 18 |
 19 |   try {
 20 |     // Set status bar to overlay content (edge-to-edge)
 21 |     await StatusBar.setOverlaysWebView({ overlay: true });
 22 |
 23 |     // Set status bar style (light content for dark backgrounds, dark for light)
 24 |     // You can customize this based on your app's theme
 25 |     await StatusBar.setStyle({ style: Style.Light });
 26 |
 27 |     // Set background color (transparent for edge-to-edge)
 28 |     await StatusBar.setBackgroundColor({ color: '#00000000' });
 29 |   } catch (error) {
 30 |     console.warn('Failed to initialize StatusBar:', error);
 31 |   }
 32 | };
 33 |
 34 | /**
 35 |  * Get safe area insets using JavaScript
 36 |  * Falls back to CSS env() variables if available
 37 |  */
 38 | export const getSafeAreaInsets = () => {
 39 |   if (!Capacitor.isNativePlatform()) {
 40 |     // On web, try to get from CSS custom properties
 41 |     const root = document.documentElement;
 42 |     const top = getComputedStyle(root).getPropertyValue('--safe-area-inset-top') || '0px';
 43 |     const bottom = getComputedStyle(root).getPropertyValue('--safe-area-inset-bottom') || '0px';
 44 |     const left = getComputedStyle(root).getPropertyValue('--safe-area-inset-left') || '0px';
 45 |     const right = getComputedStyle(root).getPropertyValue('--safe-area-inset-right') || '0px';
 46 |
 47 |     return {
 48 |       top: parseFloat(top) || 0,
 49 |       bottom: parseFloat(bottom) || 0,
 50 |       left: parseFloat(left) || 0,
 51 |       right: parseFloat(right) || 0,
 52 |     };
 53 |   }
 54 |
 55 |   // On native, use CSS env() if available, otherwise use defaults
 56 |   // Android typically has ~24px status bar and ~48px navigation bar
 57 |   const defaultTop = 24; // Typical Android status bar height
 58 |   const defaultBottom = 48; // Typical Android navigation bar height
 59 |
 60 |   return {
 61 |     top: defaultTop,
 62 |     bottom: defaultBottom,
 63 |     left: 0,
 64 |     right: 0,
 65 |   };
 66 | };
```

```
67 |
68 | /**
69 |  * React hook to get safe area insets
70 |  * Updates when window resizes
71 |  */
72 | export const useSafeAreaInsets = () => {
73 |   const [insets, setInsets] = React.useState(getSafeAreaInsets());
74 |
75 |   React.useEffect(() => {
76 |     const updateInsets = () => {
77 |       setInsets(getSafeAreaInsets());
78 |     };
79 |
80 |     // Update on resize
81 |     window.addEventListener('resize', updateInsets);
82 |     window.addEventListener('orientationchange', updateInsets);
83 |
84 |     // Initial update after a short delay to ensure CSS is applied
85 |     const timeout = setTimeout(updateInsets, 100);
86 |
87 |     return () => {
88 |       window.removeEventListener('resize', updateInsets);
89 |       window.removeEventListener('orientationchange', updateInsets);
90 |       clearTimeout(timeout);
91 |     };
92 |   }, []);
93 |
94 |   return insets;
95 | };
96 |
97 |
```

File: src/utils/workoutState.ts

Lines: 25

```
 1 | /**
 2 |  * Utility functions for checking workout state
 3 |  * Workout state is stored in localStorage as 'activityState'
 4 |  */
 5 |
 6 | /**
 7 |  * Check if user has an active workout session
 8 |  * @returns {boolean} True if user has an active workout, false otherwise
 9 |  */
10 | export const isWorkoutActive = (): boolean => {
11 |   try {
12 |     const stored = localStorage.getItem("activityState");
13 |     if (!stored) {
14 |       return false;
15 |     }
16 |
17 |     const parsed = JSON.parse(stored);
18 |     return parsed.isActive === true;
19 |   } catch (error) {
20 |     console.error("Error checking workout state:", error);
21 |     return false;
22 |   }
23 | };
24 |
25 |
```

File: src/vite-env.d.ts

Lines: 2

```
1 | /// <reference types="vite/client" />
2 |
```

File: tailwind.config.ts

Lines: 167

```
  1 | import type { Config } from "tailwindcss";
  2 |
  3 | export default {
  4 |   darkMode: ["class"],
  5 |   content: ["./pages/**/*.{ts,tsx}", "./components/**/*.{ts,tsx}", "./app/**/*.{ts,tsx}", "./src/
**/*.{ts,tsx}"],
  6 |   prefix: "",
  7 |   theme: {
  8 |     container: {
  9 |       center: true,
 10 |       padding: "2rem",
 11 |       screens: {
 12 |         "2xl": "1400px",
 13 |       },
 14 |     },
 15 |     extend: {
 16 |       colors: {
 17 |         border: "hsl(var(--border))",
 18 |         input: "hsl(var(--input))",
 19 |         ring: "hsl(var(--ring))",
 20 |         background: "hsl(var(--background))",
 21 |         foreground: "hsl(var(--foreground))",
 22 |         primary: {
 23 |           DEFAULT: "hsl(var(--primary))",
 24 |           foreground: "hsl(var(--primary-foreground))",
 25 |         },
 26 |         success: {
 27 |           DEFAULT: "hsl(var(--success))",
 28 |           foreground: "hsl(var(--success-foreground))",
 29 |         },
 30 |         warning: {
 31 |           DEFAULT: "hsl(var(--warning))",
 32 |           foreground: "hsl(var(--warning-foreground))",
 33 |         },
 34 |         purple: {
 35 |           DEFAULT: "hsl(var(--purple))",
 36 |           foreground: "hsl(var(--purple-foreground))",
 37 |         },
 38 |         secondary: {
 39 |           DEFAULT: "hsl(var(--secondary))",
 40 |           foreground: "hsl(var(--secondary-foreground))",
 41 |         },
 42 |         destructive: {
 43 |           DEFAULT: "hsl(var(--destructive))",
 44 |           foreground: "hsl(var(--destructive-foreground))",
 45 |         },
 46 |         muted: {
 47 |           DEFAULT: "hsl(var(--muted))",
 48 |           foreground: "hsl(var(--muted-foreground))",
 49 |         },
 50 |         accent: {
 51 |           DEFAULT: "hsl(var(--accent))",
 52 |           foreground: "hsl(var(--accent-foreground))",
 53 |         },
 54 |         popover: {
 55 |           DEFAULT: "hsl(var(--popover))",
 56 |           foreground: "hsl(var(--popover-foreground))",
 57 |         },
 58 |         card: {
 59 |           DEFAULT: "hsl(var(--card))",
 60 |           foreground: "hsl(var(--card-foreground))",
 61 |         },
 62 |         sidebar: {
 63 |           DEFAULT: "hsl(var(--sidebar-background))",
 64 |           foreground: "hsl(var(--sidebar-foreground))",
 65 |           primary: "hsl(var(--sidebar-primary))",
 66 |           "primary-foreground": "hsl(var(--sidebar-primary-foreground))",
```

```
          accent: "hsl(var(--sidebar-accent))",
          "accent-foreground": "hsl(var(--sidebar-accent-foreground))",
          border: "hsl(var(--sidebar-border))",
          ring: "hsl(var(--sidebar-ring))",
        },
        activity: {
          running: "hsl(var(--activity-running))",
          cycling: "hsl(var(--activity-cycling))",
          walking: "hsl(var(--activity-walking))",
        },
      },
      borderRadius: {
        lg: "var(--radius)",
        md: "calc(var(--radius) - 2px)",
        sm: "calc(var(--radius) - 4px)",
      },
      keyframes: {
        "accordion-down": {
          from: {
            height: "0",
          },
          to: {
            height: "var(--radix-accordion-content-height)",
          },
        },
        "accordion-up": {
          from: {
            height: "var(--radix-accordion-content-height)",
          },
          to: {
            height: "0",
          },
        },
        "fade-in": {
          "0%": {
            opacity: "0",
            transform: "translateY(10px)",
          },
          "100%": {
            opacity: "1",
            transform: "translateY(0)",
          },
        },
        "fade-out": {
          "0%": {
            opacity: "1",
            transform: "translateY(0)",
          },
          "100%": {
            opacity: "0",
            transform: "translateY(10px)",
          },
        },
        "slide-up": {
          "0%": {
            transform: "translateY(100%)",
          },
          "100%": {
            transform: "translateY(0)",
          },
        },
        "slide-down": {
          "0%": {
            transform: "translateY(0)",
          },
          "100%": {
            transform: "translateY(100%)",
          },
        },
        "bounce-marker": {
          "0%, 100%": {
```

```
            transform: "translateY(0)",
          },
          "50%": {
            transform: "translateY(-10px)",
          },
        },
        "pulse-slow": {
          "0%, 100%": {
            opacity: "1",
          },
          "50%": {
            opacity: "0.5",
          },
        },
      },
      animation: {
        "accordion-down": "accordion-down 0.2s ease-out",
        "accordion-up": "accordion-up 0.2s ease-out",
        "fade-in": "fade-in 0.3s ease-out",
        "fade-out": "fade-out 0.3s ease-out",
        "slide-up": "slide-up 0.3s ease-out",
        "slide-down": "slide-down 0.3s ease-out",
        "bounce-marker": "bounce-marker 1s ease-in-out infinite",
        "pulse-slow": "pulse-slow 2s cubic-bezier(0.4, 0, 0.6, 1) infinite",
      },
    },
  },
  plugins: [require("tailwindcss-animate")],
} satisfies Config;

```

File: tsconfig.app.json

Lines: 31

```
  1 | {
  2 |   "compilerOptions": {
  3 |     "target": "ES2020",
  4 |     "useDefineForClassFields": true,
  5 |     "lib": ["ES2020", "DOM", "DOM.Iterable"],
  6 |     "module": "ESNext",
  7 |     "skipLibCheck": true,
  8 |
  9 |     /* Bundler mode */
 10 |     "moduleResolution": "bundler",
 11 |     "allowImportingTsExtensions": true,
 12 |     "isolatedModules": true,
 13 |     "moduleDetection": "force",
 14 |     "noEmit": true,
 15 |     "jsx": "react-jsx",
 16 |
 17 |     /* Linting */
 18 |     "strict": false,
 19 |     "noUnusedLocals": false,
 20 |     "noUnusedParameters": false,
 21 |     "noImplicitAny": false,
 22 |     "noFallthroughCasesInSwitch": false,
 23 |
 24 |     "baseUrl": ".",
 25 |     "paths": {
 26 |       "@/*": ["./src/*"]
 27 |     }
 28 |   },
 29 |   "include": ["src"]
 30 | }
 31 |
```

File: tsconfig.json

Lines: 17

```
  1 | {
  2 |   "files": [],
  3 |   "references": [{ "path": "./tsconfig.app.json" }, { "path": "./tsconfig.node.json" }],
  4 |   "compilerOptions": {
  5 |     "baseUrl": ".",
  6 |     "paths": {
  7 |       "@/*": ["./src/*"]
  8 |     },
  9 |     "noImplicitAny": false,
 10 |     "noUnusedParameters": false,
 11 |     "skipLibCheck": true,
 12 |     "allowJs": true,
 13 |     "noUnusedLocals": false,
 14 |     "strictNullChecks": false
 15 |   }
 16 | }
 17 |
```

File: tsconfig.node.json

Lines: 23

```
 1 | {
 2 |   "compilerOptions": {
 3 |     "target": "ES2022",
 4 |     "lib": ["ES2023"],
 5 |     "module": "ESNext",
 6 |     "skipLibCheck": true,
 7 |
 8 |     /* Bundler mode */
 9 |     "moduleResolution": "bundler",
10 |     "allowImportingTsExtensions": true,
11 |     "isolatedModules": true,
12 |     "moduleDetection": "force",
13 |     "noEmit": true,
14 |
15 |     /* Linting */
16 |     "strict": true,
17 |     "noUnusedLocals": false,
18 |     "noUnusedParameters": false,
19 |     "noFallthroughCasesInSwitch": true
20 |   },
21 |   "include": ["vite.config.ts"]
22 | }
23 |
```

File: vite.config.ts

Lines: 44

```
   1 | import { defineConfig } from "vite";
   2 | import react from "@vitejs/plugin-react-swc";
   3 | import path from "path";
   4 | import { componentTagger } from "lovable-tagger";
   5 |
   6 | // CSP policy for Firebase Realtime Database
   7 | const cspPolicy = [
   8 |   "script-src 'self' 'unsafe-inline' 'unsafe-eval' https://apis.google.com https://
     | www.googleapis.com https://www.gstatic.com",
   9 |   "connect-src 'self' https://*.googleapis.com https://*.gstatic.com https://*.firebaseio.com
     | https://*.firebasedatabase.app",
  10 |   "frame-src 'self' https://accounts.google.com https://www.google.com https://*.google.com
     | https://*.firebaseapp.com",
  11 |   "img-src 'self' https: data:",
  12 |   "style-src 'self' 'unsafe-inline' https://fonts.googleapis.com",
  13 |   "font-src 'self' https://fonts.gstatic.com data:"
  14 | ].join("; ");
  15 |
  16 | // Vite plugin to set CSP header
  17 | const cspPlugin = () => ({
  18 |   name: "csp-headers",
  19 |   configureServer(server) {
  20 |     server.middlewares.use((_req, res, next) => {
  21 |       res.setHeader("Content-Security-Policy", cspPolicy);
  22 |       next();
  23 |     });
  24 |   },
  25 | });
  26 |
  27 | // https://vitejs.dev/config/
  28 | export default defineConfig(({ mode }) => ({
  29 |   server: {
  30 |     host: "::",
  31 |     port: 8080,
  32 |   },
  33 |   plugins: [
  34 |     react(),
  35 |     cspPlugin(),
  36 |     mode === "development" && componentTagger()
  37 |   ].filter(Boolean),
  38 |   resolve: {
  39 |     alias: {
  40 |       "@": path.resolve(__dirname, "./src"),
  41 |     },
  42 |   },
  43 | }));
  44 |
```