# README Documentation

Repository: supriyo08/language-connect  
Files analyzed: 98  
  
Estimated tokens: 79.1k  
  
Directory structure:  
└── supriyo08-language-connect/  
 ├── AGENTS.md  
 ├── components.json  
 ├── index.html  
 ├── netlify.toml  
 ├── package.json  
 ├── postcss.config.js  
 ├── tailwind.config.ts  
 ├── tsconfig.json  
 ├── vite.config.server.ts  
 ├── vite.config.ts  
 ├── .dockerignore  
 ├── .npmrc  
 ├── .prettierrc  
 ├── client/  
 │ ├── App.tsx  
 │ ├── global.css  
 │ ├── vite-env.d.ts  
 │ ├── components/  
 │ │ ├── VideoPlayer.tsx  
 │ │ └── ui/  
 │ │ ├── accordion.tsx  
 │ │ ├── alert-dialog.tsx  
 │ │ ├── alert.tsx  
 │ │ ├── aspect-ratio.tsx  
 │ │ ├── avatar.tsx  
 │ │ ├── badge.tsx  
 │ │ ├── breadcrumb.tsx  
 │ │ ├── button.tsx  
 │ │ ├── calendar.tsx  
 │ │ ├── card.tsx  
 │ │ ├── carousel.tsx  
 │ │ ├── chart.tsx  
 │ │ ├── checkbox.tsx  
 │ │ ├── collapsible.tsx  
 │ │ ├── command.tsx  
 │ │ ├── context-menu.tsx  
 │ │ ├── dialog.tsx  
 │ │ ├── drawer.tsx  
 │ │ ├── dropdown-menu.tsx  
 │ │ ├── form.tsx  
 │ │ ├── hover-card.tsx  
 │ │ ├── input-otp.tsx  
 │ │ ├── input.tsx  
 │ │ ├── label.tsx  
 │ │ ├── menubar.tsx  
 │ │ ├── navigation-menu.tsx  
 │ │ ├── pagination.tsx  
 │ │ ├── popover.tsx  
 │ │ ├── progress.tsx  
 │ │ ├── radio-group.tsx  
 │ │ ├── resizable.tsx  
 │ │ ├── scroll-area.tsx  
 │ │ ├── select.tsx  
 │ │ ├── separator.tsx  
 │ │ ├── sheet.tsx  
 │ │ ├── sidebar.tsx  
 │ │ ├── skeleton.tsx  
 │ │ ├── slider.tsx  
 │ │ ├── sonner.tsx  
 │ │ ├── switch.tsx  
 │ │ ├── table.tsx  
 │ │ ├── tabs.tsx  
 │ │ ├── textarea.tsx  
 │ │ ├── toast.tsx  
 │ │ ├── toaster.tsx  
 │ │ ├── toggle-group.tsx  
 │ │ ├── toggle.tsx  
 │ │ ├── tooltip.tsx  
 │ │ └── use-toast.ts  
 │ ├── contexts/  
 │ │ └── AuthContext.tsx  
 │ ├── hooks/  
 │ │ ├── use-mobile.tsx  
 │ │ └── use-toast.ts  
 │ ├── lib/  
 │ │ ├── api.ts  
 │ │ ├── utils.spec.ts  
 │ │ └── utils.ts  
 │ └── pages/  
 │ ├── Contests.tsx  
 │ ├── Index.tsx  
 │ ├── Leaderboard.tsx  
 │ ├── Login.tsx  
 │ ├── NotFound.tsx  
 │ ├── PlaceholderPage.tsx  
 │ ├── Privacy.tsx  
 │ ├── Referrals.tsx  
 │ ├── Signup.tsx  
 │ └── Terms.tsx  
 ├── netlify/  
 │ └── functions/  
 │ └── api.ts  
 ├── public/  
 │ └── robots.txt  
 ├── server/  
 │ ├── index.ts  
 │ ├── node-build.ts  
 │ ├── db/  
 │ │ ├── connection.ts  
 │ │ └── models.ts  
 │ ├── routes/  
 │ │ ├── auth.ts  
 │ │ ├── contest.ts  
 │ │ ├── demo.ts  
 │ │ ├── raffle.ts  
 │ │ ├── user-profile.ts  
 │ │ └── voting.ts  
 │ └── utils/  
 │ └── helpers.ts  
 ├── shared/  
 │ └── api.ts  
 └── .builder/  
 └── rules/  
 ├── deploy-app.mdc  
 └── organize-ui.mdc  
  
  
================================================  
FILE: AGENTS.md  
================================================  
# Fusion Starter  
  
A production-ready full-stack React application template with integrated Express server, featuring React Router 6 SPA mode, TypeScript, Vitest, Zod and modern tooling.  
  
While the starter comes with a express server, only create endpoint when strictly neccesary, for example to encapsulate logic that must leave in the server, such as private keys handling, or certain DB operations, db...  
  
## Tech Stack  
  
- \*\*Frontend\*\*: React 18 + React Router 6 (spa) + TypeScript + Vite + TailwindCSS 3  
- \*\*Backend\*\*: Express server integrated with Vite dev server  
- \*\*Testing\*\*: Vitest  
- \*\*UI\*\*: Radix UI + TailwindCSS 3 + Lucide React icons  
  
## Project Structure  
  
```  
client/ # React SPA frontend  
├── pages/ # Route components (Index.tsx = home)  
├── components/ui/ # Pre-built UI component library  
├── App.tsx # App entry point and with SPA routing setup  
└── global.css # TailwindCSS 3 theming and global styles  
  
server/ # Express API backend  
├── index.ts # Main server setup (express config + routes)  
└── routes/ # API handlers  
  
shared/ # Types used by both client & server  
└── api.ts # Example of how to share api interfaces  
```  
  
## Key Features  
  
## SPA Routing System  
  
The routing system is powered by React Router 6:  
  
- `client/pages/Index.tsx` represents the home page.  
- Routes are defined in `client/App.tsx` using the `react-router-dom` import  
- Route files are located in the `client/pages/` directory  
  
For example, routes can be defined with:  
  
```typescript  
import { BrowserRouter, Routes, Route } from "react-router-dom";  
  
<Routes>  
 <Route path="/" element={<Index />} />  
 {/\* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "\*" ROUTE \*/}  
 <Route path="\*" element={<NotFound />} />  
</Routes>;  
```  
  
### Styling System  
  
- \*\*Primary\*\*: TailwindCSS 3 utility classes  
- \*\*Theme and design tokens\*\*: Configure in `client/global.css`   
- \*\*UI components\*\*: Pre-built library in `client/components/ui/`  
- \*\*Utility\*\*: `cn()` function combines `clsx` + `tailwind-merge` for conditional classes  
  
```typescript  
// cn utility usage  
className={cn(  
 "base-classes",  
 { "conditional-class": condition },  
 props.className // User overrides  
)}  
```  
  
### Express Server Integration  
  
- \*\*Development\*\*: Single port (8080) for both frontend/backend  
- \*\*Hot reload\*\*: Both client and server code  
- \*\*API endpoints\*\*: Prefixed with `/api/`  
  
#### Example API Routes  
- `GET /api/ping` - Simple ping api  
- `GET /api/demo` - Demo endpoint   
  
### Shared Types  
Import consistent types in both client and server:  
```typescript  
import { DemoResponse } from '@shared/api';  
```  
  
Path aliases:  
- `@shared/\*` - Shared folder  
- `@/\*` - Client folder  
  
## Development Commands  
  
```bash  
npm run dev # Start dev server (client + server)  
npm run build # Production build  
npm run start # Start production server  
npm run typecheck # TypeScript validation  
npm test # Run Vitest tests  
```  
  
## Adding Features  
  
### Add new colors to the theme  
  
Open `client/global.css` and `tailwind.config.ts` and add new tailwind colors.  
  
### New API Route  
1. \*\*Optional\*\*: Create a shared interface in `shared/api.ts`:  
```typescript  
export interface MyRouteResponse {  
 message: string;  
 // Add other response properties here  
}  
```  
  
2. Create a new route handler in `server/routes/my-route.ts`:  
```typescript  
import { RequestHandler } from "express";  
import { MyRouteResponse } from "@shared/api"; // Optional: for type safety  
  
export const handleMyRoute: RequestHandler = (req, res) => {  
 const response: MyRouteResponse = {  
 message: 'Hello from my endpoint!'  
 };  
 res.json(response);  
};  
```  
  
3. Register the route in `server/index.ts`:  
```typescript  
import { handleMyRoute } from "./routes/my-route";  
  
// Add to the createServer function:  
app.get("/api/my-endpoint", handleMyRoute);  
```  
  
4. Use in React components with type safety:  
```typescript  
import { MyRouteResponse } from '@shared/api'; // Optional: for type safety  
  
const response = await fetch('/api/my-endpoint');  
const data: MyRouteResponse = await response.json();  
```  
  
### New Page Route  
1. Create component in `client/pages/MyPage.tsx`  
2. Add route in `client/App.tsx`:  
```typescript  
<Route path="/my-page" element={<MyPage />} />  
```  
  
## Production Deployment  
  
- \*\*Standard\*\*: `npm run build` + `npm start`  
- \*\*Binary\*\*: Self-contained executables (Linux, macOS, Windows)  
- \*\*Cloud Deployment\*\*: Use either Netlify or Vercel via their MCP integrations for easy deployment. Both providers work well with this starter template.  
  
## Architecture Notes  
  
- Single-port development with Vite + Express integration  
- TypeScript throughout (client, server, shared)  
- Full hot reload for rapid development  
- Production-ready with multiple deployment options  
- Comprehensive UI component library included  
- Type-safe API communication via shared interfaces  
  
  
  
================================================  
FILE: components.json  
================================================  
{  
 "$schema": "https://ui.shadcn.com/schema.json",  
 "style": "default",  
 "rsc": false,  
 "tsx": true,  
 "tailwind": {  
 "config": "tailwind.config.ts",  
 "css": "client/index.css",  
 "baseColor": "slate",  
 "cssVariables": true,  
 "prefix": ""  
 },  
 "aliases": {  
 "components": "@/components",  
 "utils": "@/lib/utils",  
 "ui": "@/components/ui",  
 "lib": "@/lib",  
 "hooks": "@/hooks"  
 }  
}  
  
  
  
================================================  
FILE: index.html  
================================================  
<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>Hello world project</title>  
 </head>  
  
 <body>  
 <div id="root"></div>  
  
 <script type="module" src="/client/App.tsx"></script>  
 </body>  
</html>  
  
  
  
================================================  
FILE: netlify.toml  
================================================  
[build]  
 command = "npm run build:client"  
 functions = "netlify/functions"  
 publish = "dist/spa"  
  
  
[functions]  
 external\_node\_modules = ["express"]  
 node\_bundler = "esbuild"  
   
[[redirects]]  
 force = true  
 from = "/api/\*"  
 status = 200  
 to = "/.netlify/functions/api/:splat"  
  
  
  
================================================  
FILE: package.json  
================================================  
{  
 "name": "fusion-starter",  
 "private": true,  
 "type": "module",  
 "pkg": {  
 "assets": [  
 "dist/spa/\*"  
 ],  
 "scripts": [  
 "dist/server/\*\*/\*.js"  
 ]  
 },  
 "scripts": {  
 "dev": "vite",  
 "build": "npm run build:client && npm run build:server",  
 "build:client": "vite build",  
 "build:server": "vite build --config vite.config.server.ts",  
 "start": "node dist/server/node-build.mjs",  
 "test": "vitest --run",  
 "format.fix": "prettier --write .",  
 "typecheck": "tsc"  
 },  
 "dependencies": {  
 "dotenv": "^17.2.0",  
 "express": "^4.18.2",  
 "mongoose": "^8.17.0",  
 "multer": "^2.0.2",  
 "zod": "^3.23.8"  
 },  
 "devDependencies": {  
 "@hookform/resolvers": "^3.9.0",  
 "@radix-ui/react-accordion": "^1.2.0",  
 "@radix-ui/react-alert-dialog": "^1.1.1",  
 "@radix-ui/react-aspect-ratio": "^1.1.0",  
 "@radix-ui/react-avatar": "^1.1.0",  
 "@radix-ui/react-checkbox": "^1.1.1",  
 "@radix-ui/react-collapsible": "^1.1.0",  
 "@radix-ui/react-context-menu": "^2.2.1",  
 "@radix-ui/react-dialog": "^1.1.2",  
 "@radix-ui/react-dropdown-menu": "^2.1.1",  
 "@radix-ui/react-hover-card": "^1.1.1",  
 "@radix-ui/react-label": "^2.1.0",  
 "@radix-ui/react-menubar": "^1.1.1",  
 "@radix-ui/react-navigation-menu": "^1.2.0",  
 "@radix-ui/react-popover": "^1.1.1",  
 "@radix-ui/react-progress": "^1.1.0",  
 "@radix-ui/react-radio-group": "^1.2.0",  
 "@radix-ui/react-scroll-area": "^1.1.0",  
 "@radix-ui/react-select": "^2.1.1",  
 "@radix-ui/react-separator": "^1.1.0",  
 "@radix-ui/react-slider": "^1.2.0",  
 "@radix-ui/react-slot": "^1.1.0",  
 "@radix-ui/react-switch": "^1.1.0",  
 "@radix-ui/react-tabs": "^1.1.0",  
 "@radix-ui/react-toast": "^1.2.1",  
 "@radix-ui/react-toggle": "^1.1.0",  
 "@radix-ui/react-toggle-group": "^1.1.0",  
 "@radix-ui/react-tooltip": "^1.1.4",  
 "@react-three/drei": "^10.1.2",  
 "@react-three/fiber": "^8.18.0",  
 "@swc/core": "^1.11.24",  
 "@tailwindcss/typography": "^0.5.15",  
 "@tanstack/react-query": "^5.56.2",  
 "@types/cors": "^2.8.17",  
 "@types/express": "^4.17.21",  
 "@types/multer": "^2.0.0",  
 "@types/node": "^22.5.5",  
 "@types/react": "^18.3.3",  
 "@types/react-dom": "^18.3.0",  
 "@types/three": "^0.176.0",  
 "@vitejs/plugin-react-swc": "^3.5.0",  
 "autoprefixer": "^10.4.21",  
 "class-variance-authority": "^0.7.1",  
 "clsx": "^2.1.1",  
 "cmdk": "^1.0.0",  
 "cors": "^2.8.5",  
 "date-fns": "^3.6.0",  
 "embla-carousel-react": "^8.3.0",  
 "framer-motion": "^12.6.2",  
 "globals": "^15.9.0",  
 "input-otp": "^1.2.4",  
 "lucide-react": "^0.462.0",  
 "next-themes": "^0.3.0",  
 "postcss": "^8.5.6",  
 "prettier": "^3.5.3",  
 "react": "^18.3.1",  
 "react-day-picker": "^8.10.1",  
 "react-dom": "^18.3.1",  
 "react-hook-form": "^7.53.0",  
 "react-resizable-panels": "^2.1.3",  
 "react-router-dom": "^6.26.2",  
 "recharts": "^2.12.7",  
 "serverless-http": "^3.2.0",  
 "sonner": "^1.5.0",  
 "tailwind-merge": "^2.5.2",  
 "tailwindcss": "^3.4.11",  
 "tailwindcss-animate": "^1.0.7",  
 "three": "^0.176.0",  
 "tsx": "^4.7.0",  
 "typescript": "^5.5.3",  
 "vaul": "^0.9.3",  
 "vite": "^6.2.2",  
 "vitest": "^3.1.4"  
 }  
}  
  
  
  
================================================  
FILE: postcss.config.js  
================================================  
export default {  
 plugins: {  
 tailwindcss: {},  
 autoprefixer: {},  
 },  
};  
  
  
  
================================================  
FILE: tailwind.config.ts  
================================================  
import type { Config } from "tailwindcss";  
  
export default {  
 darkMode: ["class"],  
 content: ["./client/\*\*/\*.{ts,tsx}"],  
 prefix: "",  
 theme: {  
 container: {  
 center: true,  
 padding: "2rem",  
 screens: {  
 "2xl": "1400px",  
 },  
 },  
 extend: {  
 colors: {  
 border: "hsl(var(--border))",  
 input: "hsl(var(--input))",  
 ring: "hsl(var(--ring))",  
 background: "hsl(var(--background))",  
 foreground: "hsl(var(--foreground))",  
 primary: {  
 DEFAULT: "hsl(var(--primary))",  
 foreground: "hsl(var(--primary-foreground))",  
 },  
 secondary: {  
 DEFAULT: "hsl(var(--secondary))",  
 foreground: "hsl(var(--secondary-foreground))",  
 },  
 destructive: {  
 DEFAULT: "hsl(var(--destructive))",  
 foreground: "hsl(var(--destructive-foreground))",  
 },  
 muted: {  
 DEFAULT: "hsl(var(--muted))",  
 foreground: "hsl(var(--muted-foreground))",  
 },  
 accent: {  
 DEFAULT: "hsl(var(--accent))",  
 foreground: "hsl(var(--accent-foreground))",  
 },  
 popover: {  
 DEFAULT: "hsl(var(--popover))",  
 foreground: "hsl(var(--popover-foreground))",  
 },  
 card: {  
 DEFAULT: "hsl(var(--card))",  
 foreground: "hsl(var(--card-foreground))",  
 },  
 sidebar: {  
 DEFAULT: "hsl(var(--sidebar-background))",  
 foreground: "hsl(var(--sidebar-foreground))",  
 primary: "hsl(var(--sidebar-primary))",  
 "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))",  
 },  
 // LanguageKonnect brand colors  
 "lk-blue": {  
 50: "#eff6ff",  
 100: "#dbeafe",  
 200: "#bfdbfe",  
 300: "#93c5fd",  
 400: "#60a5fa",  
 500: "#3b82f6",  
 600: "#2563eb",  
 700: "#1d4ed8",  
 800: "#1e40af",  
 900: "#1e3a8a",  
 },  
 "lk-purple": {  
 50: "#faf5ff",  
 100: "#f3e8ff",  
 200: "#e9d5ff",  
 300: "#d8b4fe",  
 400: "#c084fc",  
 500: "#a855f7",  
 600: "#9333ea",  
 700: "#7c3aed",  
 800: "#6b21a8",  
 900: "#581c87",  
 },  
 "lk-success": "#10b981",  
 "lk-warning": "#f59e0b",  
 "lk-error": "#ef4444",  
 },  
 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",  
 },  
 },  
 },  
 animation: {  
 "accordion-down": "accordion-down 0.2s ease-out",  
 "accordion-up": "accordion-up 0.2s ease-out",  
 },  
 },  
 },  
 plugins: [require("tailwindcss-animate")],  
} satisfies Config;  
  
  
  
================================================  
FILE: tsconfig.json  
================================================  
{  
 "compilerOptions": {  
 "target": "ES2020",  
 "useDefineForClassFields": true,  
 "lib": ["ES2020", "DOM", "DOM.Iterable"],  
 "module": "ESNext",  
 "skipLibCheck": true,  
  
 /\* Bundler mode \*/  
 "moduleResolution": "bundler",  
 "allowImportingTsExtensions": true,  
 "isolatedModules": true,  
 "moduleDetection": "force",  
 "noEmit": true,  
 "jsx": "react-jsx",  
 "esModuleInterop": true,  
 "allowSyntheticDefaultImports": true,  
 "forceConsistentCasingInFileNames": true,  
  
 /\* Linting \*/  
 "strict": false,  
 "noUnusedLocals": false,  
 "noUnusedParameters": false,  
 "noImplicitAny": false,  
 "noFallthroughCasesInSwitch": false,  
 "strictNullChecks": false,  
  
 /\* Path mapping \*/  
 "baseUrl": ".",  
 "paths": {  
 "@/\*": ["./client/\*"],  
 "@shared/\*": ["./shared/\*"]  
 }  
 },  
 "include": [  
 "client/\*\*/\*",  
 "server/\*\*/\*",  
 "shared/\*\*/\*",  
 "vite.config.ts",  
 "vite.config.server.ts"  
 ],  
 "exclude": ["node\_modules", "dist"]  
}  
  
  
  
================================================  
FILE: vite.config.server.ts  
================================================  
import { defineConfig } from "vite";  
import path from "path";  
  
// Server build configuration  
export default defineConfig({  
 build: {  
 lib: {  
 entry: path.resolve(\_\_dirname, "server/node-build.ts"),  
 name: "server",  
 fileName: "production",  
 formats: ["es"],  
 },  
 outDir: "dist/server",  
 target: "node22",  
 ssr: true,  
 rollupOptions: {  
 external: [  
 // Node.js built-ins  
 "fs",  
 "path",  
 "url",  
 "http",  
 "https",  
 "os",  
 "crypto",  
 "stream",  
 "util",  
 "events",  
 "buffer",  
 "querystring",  
 "child\_process",  
 // External dependencies that should not be bundled  
 "express",  
 "cors",  
 ],  
 output: {  
 format: "es",  
 entryFileNames: "[name].mjs",  
 },  
 },  
 minify: false, // Keep readable for debugging  
 sourcemap: true,  
 },  
 resolve: {  
 alias: {  
 "@": path.resolve(\_\_dirname, "./client"),  
 "@shared": path.resolve(\_\_dirname, "./shared"),  
 },  
 },  
 define: {  
 "process.env.NODE\_ENV": '"production"',  
 },  
});  
  
  
  
================================================  
FILE: vite.config.ts  
================================================  
import { defineConfig, Plugin } from "vite";  
import react from "@vitejs/plugin-react-swc";  
import path from "path";  
import { createServer } from "./server";  
  
// https://vitejs.dev/config/  
export default defineConfig(({ mode }) => ({  
 server: {  
 host: "::",  
 port: 8080,  
 fs: {  
 allow: ["./client", "./shared"],  
 deny: [".env", ".env.\*", "\*.{crt,pem}", "\*\*/.git/\*\*", "server/\*\*"],  
 },  
 },  
 build: {  
 outDir: "dist/spa",  
 },  
 plugins: [react(), expressPlugin()],  
 resolve: {  
 alias: {  
 "@": path.resolve(\_\_dirname, "./client"),  
 "@shared": path.resolve(\_\_dirname, "./shared"),  
 },  
 },  
}));  
  
function expressPlugin(): Plugin {  
 return {  
 name: "express-plugin",  
 apply: "serve", // Only apply during development (serve mode)  
 configureServer(server) {  
 const app = createServer();  
  
 // Add Express app as middleware to Vite dev server  
 server.middlewares.use(app);  
 },  
 };  
}  
  
  
  
================================================  
FILE: .dockerignore  
================================================  
node\_modules  
dist  
.git  
.gitignore  
README.md  
.env  
.env.local  
.env.development  
.env.test  
.env.production  
npm-debug.log\*  
yarn-debug.log\*  
yarn-error.log\*  
.DS\_Store  
.vscode  
.idea  
\*.swp  
\*.swo  
\*~  
.cursor  
coverage  
.nyc\_output  
.eslintcache  
\*.log  
\*.lock  
\*.tmp  
\*.tmp.\*  
log.txt  
  
.DS\_Store  
node\_modules  
\*\*/node\_modules/\*\*  
build  
data  
.env  
load-ids.txt  
  
server  
tmp  
types  
.git  
.gitignore  
dist  
service  
tests  
fixtures-pages  
fixtures-apps  
  
# Netlify  
.netlify  
packages/ml-air/lib  
packages/ml-air/bin  
packages/ml-air/project  
packages/ml-air/share  
packages/ml-air/random\_forest\_classification/  
packages/ml-air/\_\_pycache\_\_/  
packages/ml-air/app/\_\_pycache\_\_/  
packages/vcp-common/native-bridge/build  
packages/vcp-common/\_tests\_/dataset-ranking.csv  
node\_modules/  
Dockerfile  
.gitignore  
  
  
================================================  
FILE: .npmrc  
================================================  
legacy-peer-deps=true   
  
  
================================================  
FILE: .prettierrc  
================================================  
{  
 "tabWidth": 2,  
 "useTabs": false,  
 "trailingComma": "all"  
}  
  
  
  
================================================  
FILE: client/App.tsx  
================================================  
import "./global.css";  
  
import { Toaster } from "@/components/ui/toaster";  
import { createRoot } from "react-dom/client";  
import { Toaster as Sonner } from "@/components/ui/sonner";  
import { TooltipProvider } from "@/components/ui/tooltip";  
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";  
import { AuthProvider } from "./contexts/AuthContext";  
import { BrowserRouter, Routes, Route } from "react-router-dom";  
import Index from "./pages/Index";  
import NotFound from "./pages/NotFound";  
import Login from "./pages/Login";  
import Signup from "./pages/Signup";  
import Contests from "./pages/Contests";  
import Leaderboard from "./pages/Leaderboard";  
import Referrals from "./pages/Referrals";  
import Terms from "./pages/Terms";  
import Privacy from "./pages/Privacy";  
import PlaceholderPage from "./pages/PlaceholderPage";  
  
const queryClient = new QueryClient();  
  
const App = () => (  
 <QueryClientProvider client={queryClient}>  
 <AuthProvider>  
 <TooltipProvider>  
 <Toaster />  
 <Sonner />  
 <BrowserRouter>  
 <Routes>  
 <Route path="/" element={<Index />} />  
 <Route path="/login" element={<Login />} />  
 <Route path="/signup" element={<Signup />} />  
 <Route path="/contests" element={<Contests />} />  
 <Route path="/leaderboard" element={<Leaderboard />} />  
 <Route path="/referrals" element={<Referrals />} />  
 <Route path="/terms" element={<Terms />} />  
 <Route path="/privacy" element={<Privacy />} />  
 {/\* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "\*" ROUTE \*/}  
 <Route path="\*" element={<NotFound />} />  
 </Routes>  
 </BrowserRouter>  
 </TooltipProvider>  
 </AuthProvider>  
 </QueryClientProvider>  
);  
  
createRoot(document.getElementById("root")!).render(<App />);  
  
  
  
================================================  
FILE: client/global.css  
================================================  
@tailwind base;  
@tailwind components;  
@tailwind utilities;  
  
@layer base {  
 /\*\*  
 \* Tailwind CSS theme  
 \* tailwind.config.ts expects the following color variables to be expressed as HSL values.  
 \* A different format will require also updating the theme in tailwind.config.ts.  
 \*/  
 :root {  
 --background: 0 0% 100%;  
 --foreground: 230 15% 15%;  
  
 --card: 0 0% 100%;  
 --card-foreground: 230 15% 15%;  
  
 --popover: 0 0% 100%;  
 --popover-foreground: 230 15% 15%;  
  
 --primary: 210 100% 56%;  
 --primary-foreground: 0 0% 100%;  
  
 --secondary: 215 25% 97%;  
 --secondary-foreground: 230 15% 15%;  
  
 --muted: 215 25% 97%;  
 --muted-foreground: 230 10% 50%;  
  
 --accent: 270 95% 75%;  
 --accent-foreground: 0 0% 100%;  
  
 --destructive: 0 84.2% 60.2%;  
 --destructive-foreground: 0 0% 100%;  
  
 --border: 215 20% 90%;  
 --input: 215 20% 90%;  
 --ring: 210 100% 56%;  
  
 --radius: 0.5rem;  
  
 --sidebar-background: 0 0% 98%;  
  
 --sidebar-foreground: 240 5.3% 26.1%;  
  
 --sidebar-primary: 240 5.9% 10%;  
  
 --sidebar-primary-foreground: 0 0% 98%;  
  
 --sidebar-accent: 240 4.8% 95.9%;  
  
 --sidebar-accent-foreground: 240 5.9% 10%;  
  
 --sidebar-border: 220 13% 91%;  
  
 --sidebar-ring: 217.2 91.2% 59.8%;  
 }  
  
 .dark {  
 --background: 230 25% 8%;  
 --foreground: 210 40% 98%;  
  
 --card: 230 25% 10%;  
 --card-foreground: 210 40% 98%;  
  
 --popover: 230 25% 10%;  
 --popover-foreground: 210 40% 98%;  
  
 --primary: 210 100% 56%;  
 --primary-foreground: 0 0% 100%;  
  
 --secondary: 230 20% 15%;  
 --secondary-foreground: 210 40% 98%;  
  
 --muted: 230 20% 15%;  
 --muted-foreground: 215 20% 65%;  
  
 --accent: 270 95% 75%;  
 --accent-foreground: 0 0% 100%;  
  
 --destructive: 0 62.8% 30.6%;  
 --destructive-foreground: 0 0% 100%;  
  
 --border: 230 20% 20%;  
 --input: 230 20% 20%;  
 --ring: 210 100% 56%;  
 --sidebar-background: 240 5.9% 10%;  
 --sidebar-foreground: 240 4.8% 95.9%;  
 --sidebar-primary: 224.3 76.3% 48%;  
 --sidebar-primary-foreground: 0 0% 100%;  
 --sidebar-accent: 240 3.7% 15.9%;  
 --sidebar-accent-foreground: 240 4.8% 95.9%;  
 --sidebar-border: 240 3.7% 15.9%;  
 --sidebar-ring: 217.2 91.2% 59.8%;  
 }  
}  
  
@layer base {  
 \* {  
 @apply border-border;  
 }  
  
 body {  
 @apply bg-background text-foreground;  
 }  
}  
  
  
  
================================================  
FILE: client/vite-env.d.ts  
================================================  
/// <reference types="vite/client" />  
  
  
  
================================================  
FILE: client/components/VideoPlayer.tsx  
================================================  
import { useState, useRef } from 'react';  
import { Button } from '@/components/ui/button';  
import { Play, Pause, Volume2, VolumeX, Maximize } from 'lucide-react';  
  
interface VideoPlayerProps {  
 src: string;  
 thumbnail?: string;  
 title: string;  
 className?: string;  
 autoPlay?: boolean;  
}  
  
export default function VideoPlayer({   
 src,   
 thumbnail,   
 title,   
 className = "",   
 autoPlay = false   
}: VideoPlayerProps) {  
 const [isPlaying, setIsPlaying] = useState(false);  
 const [isMuted, setIsMuted] = useState(false);  
 const [showControls, setShowControls] = useState(true);  
 const [error, setError] = useState(false);  
 const videoRef = useRef<HTMLVideoElement>(null);  
  
 const togglePlay = () => {  
 if (videoRef.current) {  
 if (isPlaying) {  
 videoRef.current.pause();  
 } else {  
 videoRef.current.play().catch(err => {  
 console.error('Error playing video:', err);  
 setError(true);  
 });  
 }  
 }  
 };  
  
 const toggleMute = () => {  
 if (videoRef.current) {  
 videoRef.current.muted = !isMuted;  
 setIsMuted(!isMuted);  
 }  
 };  
  
 const toggleFullscreen = () => {  
 if (videoRef.current) {  
 if (document.fullscreenElement) {  
 document.exitFullscreen();  
 } else {  
 videoRef.current.requestFullscreen().catch(err => {  
 console.error('Error entering fullscreen:', err);  
 });  
 }  
 }  
 };  
  
 const handleVideoError = () => {  
 setError(true);  
 setIsPlaying(false);  
 };  
  
 const handleVideoPlay = () => {  
 setIsPlaying(true);  
 setError(false);  
 };  
  
 const handleVideoPause = () => {  
 setIsPlaying(false);  
 };  
  
 if (error) {  
 return (  
 <div className={`relative aspect-video bg-muted rounded-lg flex items-center justify-center ${className}`}>  
 <div className="text-center">  
 <div className="w-16 h-16 bg-muted-foreground/20 rounded-full flex items-center justify-center mx-auto mb-2">  
 <Play className="w-8 h-8 text-muted-foreground" />  
 </div>  
 <p className="text-sm text-muted-foreground">Video unavailable</p>  
 <p className="text-xs text-muted-foreground">{title}</p>  
 </div>  
 </div>  
 );  
 }  
  
 return (  
 <div   
 className={`relative aspect-video bg-black rounded-lg overflow-hidden group ${className}`}  
 onMouseEnter={() => setShowControls(true)}  
 onMouseLeave={() => setShowControls(isPlaying ? false : true)}  
 >  
 <video  
 ref={videoRef}  
 className="w-full h-full object-cover"  
 poster={thumbnail}  
 autoPlay={autoPlay}  
 muted={isMuted}  
 onPlay={handleVideoPlay}  
 onPause={handleVideoPause}  
 onError={handleVideoError}  
 onClick={togglePlay}  
 title={title}  
 >  
 <source src={src} type="video/mp4" />  
 Your browser does not support the video tag.  
 </video>  
  
 {/\* Video Controls Overlay \*/}  
 <div className={`absolute inset-0 bg-black/40 transition-opacity duration-300 ${  
 showControls ? 'opacity-100' : 'opacity-0'  
 }`}>  
   
 {/\* Center Play Button \*/}  
 <div className="absolute inset-0 flex items-center justify-center">  
 <Button  
 onClick={togglePlay}  
 size="lg"  
 className="w-16 h-16 rounded-full bg-white/20 hover:bg-white/30 backdrop-blur-sm border-2 border-white/30"  
 aria-label={isPlaying ? 'Pause video' : 'Play video'}  
 >  
 {isPlaying ? (  
 <Pause className="w-8 h-8 text-white" />  
 ) : (  
 <Play className="w-8 h-8 text-white ml-1" />  
 )}  
 </Button>  
 </div>  
  
 {/\* Bottom Controls \*/}  
 <div className="absolute bottom-0 left-0 right-0 p-4">  
 <div className="flex items-center justify-between">  
 <div className="flex items-center space-x-2">  
 <Button  
 onClick={togglePlay}  
 size="sm"  
 variant="ghost"  
 className="text-white hover:bg-white/20"  
 aria-label={isPlaying ? 'Pause video' : 'Play video'}  
 >  
 {isPlaying ? (  
 <Pause className="w-4 h-4" />  
 ) : (  
 <Play className="w-4 h-4" />  
 )}  
 </Button>  
   
 <Button  
 onClick={toggleMute}  
 size="sm"  
 variant="ghost"  
 className="text-white hover:bg-white/20"  
 aria-label={isMuted ? 'Unmute video' : 'Mute video'}  
 >  
 {isMuted ? (  
 <VolumeX className="w-4 h-4" />  
 ) : (  
 <Volume2 className="w-4 h-4" />  
 )}  
 </Button>  
 </div>  
  
 <div className="flex items-center space-x-2">  
 <Button  
 onClick={toggleFullscreen}  
 size="sm"  
 variant="ghost"  
 className="text-white hover:bg-white/20"  
 aria-label="Enter fullscreen"  
 >  
 <Maximize className="w-4 h-4" />  
 </Button>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 {/\* Video Info Overlay \*/}  
 <div className="absolute top-4 left-4 right-4">  
 <div className="bg-black/50 backdrop-blur-sm rounded-lg px-3 py-2">  
 <p className="text-white text-sm font-medium truncate">{title}</p>  
 </div>  
 </div>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/components/ui/accordion.tsx  
================================================  
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]>svg]:rotate-180",  
 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-accordion-down"  
 {...props}  
 >  
 <div className={cn("pb-4 pt-0", className)}>{children}</div>  
 </AccordionPrimitive.Content>  
));  
  
AccordionContent.displayName = AccordionPrimitive.Content.displayName;  
  
export { Accordion, AccordionItem, AccordionTrigger, AccordionContent };  
  
  
  
================================================  
FILE: client/components/ui/alert-dialog.tsx  
================================================  
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-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",  
 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-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[state=closed]:slide-out-to-left-1/2 data-[state=closed]:slide-out-to-top-[48%] data-[state=open]:slide-in-from-left-1/2 data-[state=open]:slide-in-from-top-[48%] sm:rounded-lg",  
 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>  
>(({ className, ...props }, ref) => (  
 <AlertDialogPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
));  
AlertDialogDescription.displayName =  
 AlertDialogPrimitive.Description.displayName;  
  
const AlertDialogAction = React.forwardRef<  
 React.ElementRef<typeof AlertDialogPrimitive.Action>,  
 React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Action>  
>(({ className, ...props }, ref) => (  
 <AlertDialogPrimitive.Action  
 ref={ref}  
 className={cn(buttonVariants(), className)}  
 {...props}  
 />  
));  
AlertDialogAction.displayName = AlertDialogPrimitive.Action.displayName;  
  
const AlertDialogCancel = React.forwardRef<  
 React.ElementRef<typeof AlertDialogPrimitive.Cancel>,  
 React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Cancel>  
>(({ className, ...props }, ref) => (  
 <AlertDialogPrimitive.Cancel  
 ref={ref}  
 className={cn(  
 buttonVariants({ variant: "outline" }),  
 "mt-2 sm:mt-0",  
 className,  
 )}  
 {...props}  
 />  
));  
AlertDialogCancel.displayName = AlertDialogPrimitive.Cancel.displayName;  
  
export {  
 AlertDialog,  
 AlertDialogPortal,  
 AlertDialogOverlay,  
 AlertDialogTrigger,  
 AlertDialogContent,  
 AlertDialogHeader,  
 AlertDialogFooter,  
 AlertDialogTitle,  
 AlertDialogDescription,  
 AlertDialogAction,  
 AlertDialogCancel,  
};  
  
  
  
================================================  
FILE: client/components/ui/alert.tsx  
================================================  
import \* as React from "react";  
import { cva, type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
  
const alertVariants = cva(  
 "relative w-full rounded-lg border p-4 [&>svg~\*]:pl-7 [&>svg+div]:translate-y-[-3px] [&>svg]:absolute [&>svg]:left-4 [&>svg]:top-4 [&>svg]:text-foreground",  
 {  
 variants: {  
 variant: {  
 default: "bg-background text-foreground",  
 destructive:  
 "border-destructive/50 text-destructive dark:border-destructive [&>svg]:text-destructive",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 },  
 },  
);  
  
const Alert = React.forwardRef<  
 HTMLDivElement,  
 React.HTMLAttributes<HTMLDivElement> & VariantProps<typeof alertVariants>  
>(({ className, variant, ...props }, ref) => (  
 <div  
 ref={ref}  
 role="alert"  
 className={cn(alertVariants({ variant }), className)}  
 {...props}  
 />  
));  
Alert.displayName = "Alert";  
  
const AlertTitle = React.forwardRef<  
 HTMLParagraphElement,  
 React.HTMLAttributes<HTMLHeadingElement>  
>(({ className, ...props }, ref) => (  
 <h5  
 ref={ref}  
 className={cn("mb-1 font-medium leading-none tracking-tight", className)}  
 {...props}  
 />  
));  
AlertTitle.displayName = "AlertTitle";  
  
const AlertDescription = React.forwardRef<  
 HTMLParagraphElement,  
 React.HTMLAttributes<HTMLParagraphElement>  
>(({ className, ...props }, ref) => (  
 <div  
 ref={ref}  
 className={cn("text-sm [&\_p]:leading-relaxed", className)}  
 {...props}  
 />  
));  
AlertDescription.displayName = "AlertDescription";  
  
export { Alert, AlertTitle, AlertDescription };  
  
  
  
================================================  
FILE: client/components/ui/aspect-ratio.tsx  
================================================  
import \* as AspectRatioPrimitive from "@radix-ui/react-aspect-ratio";  
  
const AspectRatio = AspectRatioPrimitive.Root;  
  
export { AspectRatio };  
  
  
  
================================================  
FILE: client/components/ui/avatar.tsx  
================================================  
import \* as React from "react";  
import \* as AvatarPrimitive from "@radix-ui/react-avatar";  
  
import { cn } from "@/lib/utils";  
  
const Avatar = React.forwardRef<  
 React.ElementRef<typeof AvatarPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>  
>(({ className, ...props }, ref) => (  
 <AvatarPrimitive.Root  
 ref={ref}  
 className={cn(  
 "relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full",  
 className,  
 )}  
 {...props}  
 />  
));  
Avatar.displayName = AvatarPrimitive.Root.displayName;  
  
const AvatarImage = React.forwardRef<  
 React.ElementRef<typeof AvatarPrimitive.Image>,  
 React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>  
>(({ className, ...props }, ref) => (  
 <AvatarPrimitive.Image  
 ref={ref}  
 className={cn("aspect-square h-full w-full", className)}  
 {...props}  
 />  
));  
AvatarImage.displayName = AvatarPrimitive.Image.displayName;  
  
const AvatarFallback = React.forwardRef<  
 React.ElementRef<typeof AvatarPrimitive.Fallback>,  
 React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>  
>(({ className, ...props }, ref) => (  
 <AvatarPrimitive.Fallback  
 ref={ref}  
 className={cn(  
 "flex h-full w-full items-center justify-center rounded-full bg-muted",  
 className,  
 )}  
 {...props}  
 />  
));  
AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName;  
  
export { Avatar, AvatarImage, AvatarFallback };  
  
  
  
================================================  
FILE: client/components/ui/badge.tsx  
================================================  
import \* as React from "react";  
import { cva, type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
  
const badgeVariants = cva(  
 "inline-flex items-center rounded-full border px-2.5 py-0.5 text-xs font-semibold transition-colors focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2",  
 {  
 variants: {  
 variant: {  
 default:  
 "border-transparent bg-primary text-primary-foreground hover:bg-primary/80",  
 secondary:  
 "border-transparent bg-secondary text-secondary-foreground hover:bg-secondary/80",  
 destructive:  
 "border-transparent bg-destructive text-destructive-foreground hover:bg-destructive/80",  
 outline: "text-foreground",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 },  
 },  
);  
  
export interface BadgeProps  
 extends React.HTMLAttributes<HTMLDivElement>,  
 VariantProps<typeof badgeVariants> {}  
  
function Badge({ className, variant, ...props }: BadgeProps) {  
 return (  
 <div className={cn(badgeVariants({ variant }), className)} {...props} />  
 );  
}  
  
export { Badge, badgeVariants };  
  
  
  
================================================  
FILE: client/components/ui/breadcrumb.tsx  
================================================  
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>  
);  
BreadcrumbSeparator.displayName = "BreadcrumbSeparator";  
  
const BreadcrumbEllipsis = ({  
 className,  
 ...props  
}: React.ComponentProps<"span">) => (  
 <span  
 role="presentation"  
 aria-hidden="true"  
 className={cn("flex h-9 w-9 items-center justify-center", className)}  
 {...props}  
 >  
 <MoreHorizontal className="h-4 w-4" />  
 <span className="sr-only">More</span>  
 </span>  
);  
BreadcrumbEllipsis.displayName = "BreadcrumbElipssis";  
  
export {  
 Breadcrumb,  
 BreadcrumbList,  
 BreadcrumbItem,  
 BreadcrumbLink,  
 BreadcrumbPage,  
 BreadcrumbSeparator,  
 BreadcrumbEllipsis,  
};  
  
  
  
================================================  
FILE: client/components/ui/button.tsx  
================================================  
import \* as React from "react";  
import { Slot } from "@radix-ui/react-slot";  
import { cva, type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
  
const buttonVariants = cva(  
 "inline-flex items-center justify-center gap-2 whitespace-nowrap rounded-md text-sm font-medium ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 [&\_svg]:pointer-events-none [&\_svg]:size-4 [&\_svg]:shrink-0",  
 {  
 variants: {  
 variant: {  
 default: "bg-primary text-primary-foreground hover:bg-primary/90",  
 destructive:  
 "bg-destructive text-destructive-foreground hover:bg-destructive/90",  
 outline:  
 "border border-input bg-background hover:bg-accent hover:text-accent-foreground",  
 secondary:  
 "bg-secondary text-secondary-foreground hover:bg-secondary/80",  
 ghost: "hover:bg-accent hover:text-accent-foreground",  
 link: "text-primary underline-offset-4 hover:underline",  
 },  
 size: {  
 default: "h-10 px-4 py-2",  
 sm: "h-9 rounded-md px-3",  
 lg: "h-11 rounded-md px-8",  
 icon: "h-10 w-10",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 size: "default",  
 },  
 },  
);  
  
export interface ButtonProps  
 extends React.ButtonHTMLAttributes<HTMLButtonElement>,  
 VariantProps<typeof buttonVariants> {  
 asChild?: boolean;  
}  
  
const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(  
 ({ className, variant, size, asChild = false, ...props }, ref) => {  
 const Comp = asChild ? Slot : "button";  
 return (  
 <Comp  
 className={cn(buttonVariants({ variant, size, className }))}  
 ref={ref}  
 {...props}  
 />  
 );  
 },  
);  
Button.displayName = "Button";  
  
export { Button, buttonVariants };  
  
  
  
================================================  
FILE: client/components/ui/calendar.tsx  
================================================  
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].day-outside)]:bg-accent/50 [&: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: client/components/ui/card.tsx  
================================================  
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: client/components/ui/carousel.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"  
 : "-bottom-12 left-1/2 -translate-x-1/2 rotate-90",  
 className,  
 )}  
 disabled={!canScrollNext}  
 onClick={scrollNext}  
 {...props}  
 >  
 <ArrowRight className="h-4 w-4" />  
 <span className="sr-only">Next slide</span>  
 </Button>  
 );  
});  
CarouselNext.displayName = "CarouselNext";  
  
export {  
 type CarouselApi,  
 Carousel,  
 CarouselContent,  
 CarouselItem,  
 CarouselPrevious,  
 CarouselNext,  
};  
  
  
  
================================================  
FILE: client/components/ui/chart.tsx  
================================================  
import \* as React from "react";  
import \* as RechartsPrimitive from "recharts";  
  
import { cn } from "@/lib/utils";  
  
// Format: { THEME\_NAME: CSS\_SELECTOR }  
const THEMES = { light: "", dark: ".dark" } as const;  
  
export type ChartConfig = {  
 [k in string]: {  
 label?: React.ReactNode;  
 icon?: React.ComponentType;  
 } & (  
 | { color?: string; theme?: never }  
 | { color?: never; theme: Record<keyof typeof THEMES, string> }  
 );  
};  
  
type ChartContextProps = {  
 config: ChartConfig;  
};  
  
const ChartContext = React.createContext<ChartContextProps | null>(null);  
  
function useChart() {  
 const context = React.useContext(ChartContext);  
  
 if (!context) {  
 throw new Error("useChart must be used within a <ChartContainer />");  
 }  
  
 return context;  
}  
  
const ChartContainer = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div"> & {  
 config: ChartConfig;  
 children: React.ComponentProps<  
 typeof RechartsPrimitive.ResponsiveContainer  
 >["children"];  
 }  
>(({ id, className, children, config, ...props }, ref) => {  
 const uniqueId = React.useId();  
 const chartId = `chart-${id || uniqueId.replace(/:/g, "")}`;  
  
 return (  
 <ChartContext.Provider value={{ config }}>  
 <div  
 data-chart={chartId}  
 ref={ref}  
 className={cn(  
 "flex aspect-video justify-center text-xs [&\_.recharts-cartesian-axis-tick\_text]:fill-muted-foreground [&\_.recharts-cartesian-grid\_line[stroke='#ccc']]:stroke-border/50 [&\_.recharts-curve.recharts-tooltip-cursor]:stroke-border [&\_.recharts-dot[stroke='#fff']]:stroke-transparent [&\_.recharts-layer]:outline-none [&\_.recharts-polar-grid\_[stroke='#ccc']]:stroke-border [&\_.recharts-radial-bar-background-sector]:fill-muted [&\_.recharts-rectangle.recharts-tooltip-cursor]:fill-muted [&\_.recharts-reference-line\_[stroke='#ccc']]:stroke-border [&\_.recharts-sector[stroke='#fff']]:stroke-transparent [&\_.recharts-sector]:outline-none [&\_.recharts-surface]:outline-none",  
 className,  
 )}  
 {...props}  
 >  
 <ChartStyle id={chartId} config={config} />  
 <RechartsPrimitive.ResponsiveContainer>  
 {children}  
 </RechartsPrimitive.ResponsiveContainer>  
 </div>  
 </ChartContext.Provider>  
 );  
});  
ChartContainer.displayName = "Chart";  
  
const ChartStyle = ({ id, config }: { id: string; config: ChartConfig }) => {  
 const colorConfig = Object.entries(config).filter(  
 ([\_, config]) => config.theme || config.color,  
 );  
  
 if (!colorConfig.length) {  
 return null;  
 }  
  
 return (  
 <style  
 dangerouslySetInnerHTML={{  
 \_\_html: Object.entries(THEMES)  
 .map(  
 ([theme, prefix]) => `  
${prefix} [data-chart=${id}] {  
${colorConfig  
 .map(([key, itemConfig]) => {  
 const color =  
 itemConfig.theme?.[theme as keyof typeof itemConfig.theme] ||  
 itemConfig.color;  
 return color ? ` --color-${key}: ${color};` : null;  
 })  
 .join("\n")}  
}  
`,  
 )  
 .join("\n"),  
 }}  
 />  
 );  
};  
  
const ChartTooltip = RechartsPrimitive.Tooltip;  
  
const ChartTooltipContent = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<typeof RechartsPrimitive.Tooltip> &  
 React.ComponentProps<"div"> & {  
 hideLabel?: boolean;  
 hideIndicator?: boolean;  
 indicator?: "line" | "dot" | "dashed";  
 nameKey?: string;  
 labelKey?: string;  
 }  
>(  
 (  
 {  
 active,  
 payload,  
 className,  
 indicator = "dot",  
 hideLabel = false,  
 hideIndicator = false,  
 label,  
 labelFormatter,  
 labelClassName,  
 formatter,  
 color,  
 nameKey,  
 labelKey,  
 },  
 ref,  
 ) => {  
 const { config } = useChart();  
  
 const tooltipLabel = React.useMemo(() => {  
 if (hideLabel || !payload?.length) {  
 return null;  
 }  
  
 const [item] = payload;  
 const key = `${labelKey || item.dataKey || item.name || "value"}`;  
 const itemConfig = getPayloadConfigFromPayload(config, item, key);  
 const value =  
 !labelKey && typeof label === "string"  
 ? config[label as keyof typeof config]?.label || label  
 : itemConfig?.label;  
  
 if (labelFormatter) {  
 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 shadow-xl",  
 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) {  
 return undefined;  
 }  
  
 const payloadPayload =  
 "payload" in payload &&  
 typeof payload.payload === "object" &&  
 payload.payload !== null  
 ? payload.payload  
 : undefined;  
  
 let configLabelKey: string = key;  
  
 if (  
 key in payload &&  
 typeof payload[key as keyof typeof payload] === "string"  
 ) {  
 configLabelKey = payload[key as keyof typeof payload] as string;  
 } else if (  
 payloadPayload &&  
 key in payloadPayload &&  
 typeof payloadPayload[key as keyof typeof payloadPayload] === "string"  
 ) {  
 configLabelKey = payloadPayload[  
 key as keyof typeof payloadPayload  
 ] as string;  
 }  
  
 return configLabelKey in config  
 ? config[configLabelKey]  
 : config[key as keyof typeof config];  
}  
  
export {  
 ChartContainer,  
 ChartTooltip,  
 ChartTooltipContent,  
 ChartLegend,  
 ChartLegendContent,  
 ChartStyle,  
};  
  
  
  
================================================  
FILE: client/components/ui/checkbox.tsx  
================================================  
import \* as React from "react";  
import \* as CheckboxPrimitive from "@radix-ui/react-checkbox";  
import { Check } from "lucide-react";  
  
import { cn } from "@/lib/utils";  
  
const Checkbox = React.forwardRef<  
 React.ElementRef<typeof CheckboxPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>  
>(({ className, ...props }, ref) => (  
 <CheckboxPrimitive.Root  
 ref={ref}  
 className={cn(  
 "peer h-4 w-4 shrink-0 rounded-sm border border-primary ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=checked]:text-primary-foreground",  
 className,  
 )}  
 {...props}  
 >  
 <CheckboxPrimitive.Indicator  
 className={cn("flex items-center justify-center text-current")}  
 >  
 <Check className="h-4 w-4" />  
 </CheckboxPrimitive.Indicator>  
 </CheckboxPrimitive.Root>  
));  
Checkbox.displayName = CheckboxPrimitive.Root.displayName;  
  
export { Checkbox };  
  
  
  
================================================  
FILE: client/components/ui/collapsible.tsx  
================================================  
import \* as CollapsiblePrimitive from "@radix-ui/react-collapsible";  
  
const Collapsible = CollapsiblePrimitive.Root;  
  
const CollapsibleTrigger = CollapsiblePrimitive.CollapsibleTrigger;  
  
const CollapsibleContent = CollapsiblePrimitive.CollapsibleContent;  
  
export { Collapsible, CollapsibleTrigger, CollapsibleContent };  
  
  
  
================================================  
FILE: client/components/ui/command.tsx  
================================================  
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]]:text-muted-foreground [&\_[cmdk-group]:not([hidden])\_~[cmdk-group]]:pt-0 [&\_[cmdk-group]]:px-2 [&\_[cmdk-input-wrapper]\_svg]:h-5 [&\_[cmdk-input-wrapper]\_svg]:w-5 [&\_[cmdk-input]]:h-12 [&\_[cmdk-item]]:px-2 [&\_[cmdk-item]]:py-3 [&\_[cmdk-item]\_svg]:h-5 [&\_[cmdk-item]\_svg]:w-5">  
 {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:cursor-not-allowed disabled:opacity-50",  
 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}  
 />  
));  
  
CommandList.displayName = CommandPrimitive.List.displayName;  
  
const CommandEmpty = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Empty>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Empty>  
>((props, ref) => (  
 <CommandPrimitive.Empty  
 ref={ref}  
 className="py-6 text-center text-sm"  
 {...props}  
 />  
));  
  
CommandEmpty.displayName = CommandPrimitive.Empty.displayName;  
  
const CommandGroup = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Group>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Group>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Group  
 ref={ref}  
 className={cn(  
 "overflow-hidden p-1 text-foreground [&\_[cmdk-group-heading]]:px-2 [&\_[cmdk-group-heading]]:py-1.5 [&\_[cmdk-group-heading]]:text-xs [&\_[cmdk-group-heading]]:font-medium [&\_[cmdk-group-heading]]:text-muted-foreground",  
 className,  
 )}  
 {...props}  
 />  
));  
  
CommandGroup.displayName = CommandPrimitive.Group.displayName;  
  
const CommandSeparator = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 h-px bg-border", className)}  
 {...props}  
 />  
));  
CommandSeparator.displayName = CommandPrimitive.Separator.displayName;  
  
const CommandItem = React.forwardRef<  
 React.ElementRef<typeof CommandPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof CommandPrimitive.Item>  
>(({ className, ...props }, ref) => (  
 <CommandPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none data-[disabled=true]:pointer-events-none data-[selected='true']:bg-accent data-[selected=true]:text-accent-foreground data-[disabled=true]:opacity-50",  
 className,  
 )}  
 {...props}  
 />  
));  
  
CommandItem.displayName = CommandPrimitive.Item.displayName;  
  
const CommandShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn(  
 "ml-auto text-xs tracking-widest text-muted-foreground",  
 className,  
 )}  
 {...props}  
 />  
 );  
};  
CommandShortcut.displayName = "CommandShortcut";  
  
export {  
 Command,  
 CommandDialog,  
 CommandInput,  
 CommandList,  
 CommandEmpty,  
 CommandGroup,  
 CommandItem,  
 CommandShortcut,  
 CommandSeparator,  
};  
  
  
  
================================================  
FILE: client/components/ui/context-menu.tsx  
================================================  
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 focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 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=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 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 fade-in-80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
 </ContextMenuPrimitive.Portal>  
));  
ContextMenuContent.displayName = ContextMenuPrimitive.Content.displayName;  
  
const ContextMenuItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.Item> & {  
 inset?: boolean;  
 }  
>(({ className, inset, ...props }, ref) => (  
 <ContextMenuPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 inset && "pl-8",  
 className,  
 )}  
 {...props}  
 />  
));  
ContextMenuItem.displayName = ContextMenuPrimitive.Item.displayName;  
  
const ContextMenuCheckboxItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.CheckboxItem>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.CheckboxItem>  
>(({ className, children, checked, ...props }, ref) => (  
 <ContextMenuPrimitive.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 focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className,  
 )}  
 checked={checked}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <ContextMenuPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </ContextMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </ContextMenuPrimitive.CheckboxItem>  
));  
ContextMenuCheckboxItem.displayName =  
 ContextMenuPrimitive.CheckboxItem.displayName;  
  
const ContextMenuRadioItem = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.RadioItem>,  
 React.ComponentPropsWithoutRef<typeof ContextMenuPrimitive.RadioItem>  
>(({ className, children, ...props }, ref) => (  
 <ContextMenuPrimitive.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 focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className,  
 )}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <ContextMenuPrimitive.ItemIndicator>  
 <Circle className="h-2 w-2 fill-current" />  
 </ContextMenuPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </ContextMenuPrimitive.RadioItem>  
));  
ContextMenuRadioItem.displayName = ContextMenuPrimitive.RadioItem.displayName;  
  
const ContextMenuLabel = React.forwardRef<  
 React.ElementRef<typeof ContextMenuPrimitive.Label>,  
 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: client/components/ui/dialog.tsx  
================================================  
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]:fade-out-0 data-[state=open]:fade-in-0",  
 className,  
 )}  
 {...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-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[state=closed]:slide-out-to-left-1/2 data-[state=closed]:slide-out-to-top-[48%] data-[state=open]:slide-in-from-left-1/2 data-[state=open]:slide-in-from-top-[48%] sm:rounded-lg",  
 className,  
 )}  
 {...props}  
 >  
 {children}  
 <DialogPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none data-[state=open]:bg-accent data-[state=open]:text-muted-foreground">  
 <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>  
>(({ className, ...props }, ref) => (  
 <DialogPrimitive.Title  
 ref={ref}  
 className={cn(  
 "text-lg font-semibold leading-none tracking-tight",  
 className,  
 )}  
 {...props}  
 />  
));  
DialogTitle.displayName = DialogPrimitive.Title.displayName;  
  
const DialogDescription = React.forwardRef<  
 React.ElementRef<typeof DialogPrimitive.Description>,  
 React.ComponentPropsWithoutRef<typeof DialogPrimitive.Description>  
>(({ className, ...props }, ref) => (  
 <DialogPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
));  
DialogDescription.displayName = DialogPrimitive.Description.displayName;  
  
export {  
 Dialog,  
 DialogPortal,  
 DialogOverlay,  
 DialogClose,  
 DialogTrigger,  
 DialogContent,  
 DialogHeader,  
 DialogFooter,  
 DialogTitle,  
 DialogDescription,  
};  
  
  
  
================================================  
FILE: client/components/ui/drawer.tsx  
================================================  
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;  
  
const DrawerDescription = React.forwardRef<  
 React.ElementRef<typeof DrawerPrimitive.Description>,  
 React.ComponentPropsWithoutRef<typeof DrawerPrimitive.Description>  
>(({ className, ...props }, ref) => (  
 <DrawerPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
));  
DrawerDescription.displayName = DrawerPrimitive.Description.displayName;  
  
export {  
 Drawer,  
 DrawerPortal,  
 DrawerOverlay,  
 DrawerTrigger,  
 DrawerClose,  
 DrawerContent,  
 DrawerHeader,  
 DrawerFooter,  
 DrawerTitle,  
 DrawerDescription,  
};  
  
  
  
================================================  
FILE: client/components/ui/dropdown-menu.tsx  
================================================  
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 focus:bg-accent 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=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 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-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 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-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 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-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 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-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 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>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {  
 inset?: boolean;  
 }  
>(({ className, inset, ...props }, ref) => (  
 <DropdownMenuPrimitive.Label  
 ref={ref}  
 className={cn(  
 "px-2 py-1.5 text-sm font-semibold",  
 inset && "pl-8",  
 className,  
 )}  
 {...props}  
 />  
));  
DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName;  
  
const DropdownMenuSeparator = React.forwardRef<  
 React.ElementRef<typeof DropdownMenuPrimitive.Separator>,  
 React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>  
>(({ className, ...props }, ref) => (  
 <DropdownMenuPrimitive.Separator  
 ref={ref}  
 className={cn("-mx-1 my-1 h-px bg-muted", className)}  
 {...props}  
 />  
));  
DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName;  
  
const DropdownMenuShortcut = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLSpanElement>) => {  
 return (  
 <span  
 className={cn("ml-auto text-xs tracking-widest opacity-60", className)}  
 {...props}  
 />  
 );  
};  
DropdownMenuShortcut.displayName = "DropdownMenuShortcut";  
  
export {  
 DropdownMenu,  
 DropdownMenuTrigger,  
 DropdownMenuContent,  
 DropdownMenuItem,  
 DropdownMenuCheckboxItem,  
 DropdownMenuRadioItem,  
 DropdownMenuLabel,  
 DropdownMenuSeparator,  
 DropdownMenuShortcut,  
 DropdownMenuGroup,  
 DropdownMenuPortal,  
 DropdownMenuSub,  
 DropdownMenuSubContent,  
 DropdownMenuSubTrigger,  
 DropdownMenuRadioGroup,  
};  
  
  
  
================================================  
FILE: client/components/ui/form.tsx  
================================================  
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: client/components/ui/hover-card.tsx  
================================================  
import \* as React from "react";  
import \* as HoverCardPrimitive from "@radix-ui/react-hover-card";  
  
import { cn } from "@/lib/utils";  
  
const HoverCard = HoverCardPrimitive.Root;  
  
const HoverCardTrigger = HoverCardPrimitive.Trigger;  
  
const HoverCardContent = React.forwardRef<  
 React.ElementRef<typeof HoverCardPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof HoverCardPrimitive.Content>  
>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (  
 <HoverCardPrimitive.Content  
 ref={ref}  
 align={align}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 w-64 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
));  
HoverCardContent.displayName = HoverCardPrimitive.Content.displayName;  
  
export { HoverCard, HoverCardTrigger, HoverCardContent };  
  
  
  
================================================  
FILE: client/components/ui/input-otp.tsx  
================================================  
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-sm transition-all first:rounded-l-md first:border-l last:rounded-r-md",  
 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="h-4 w-px animate-caret-blink 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: client/components/ui/input.tsx  
================================================  
import \* as React from "react";  
  
import { cn } from "@/lib/utils";  
  
const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>(  
 ({ className, type, ...props }, ref) => {  
 return (  
 <input  
 type={type}  
 className={cn(  
 "flex h-10 w-full rounded-md border border-input bg-background px-3 py-2 text-base ring-offset-background file:border-0 file:bg-transparent file:text-sm file:font-medium file:text-foreground placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 md:text-sm",  
 className,  
 )}  
 ref={ref}  
 {...props}  
 />  
 );  
 },  
);  
Input.displayName = "Input";  
  
export { Input };  
  
  
  
================================================  
FILE: client/components/ui/label.tsx  
================================================  
import \* as React from "react";  
import \* as LabelPrimitive from "@radix-ui/react-label";  
import { cva, type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
  
const labelVariants = cva(  
 "text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70",  
);  
  
const Label = React.forwardRef<  
 React.ElementRef<typeof LabelPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> &  
 VariantProps<typeof labelVariants>  
>(({ className, ...props }, ref) => (  
 <LabelPrimitive.Root  
 ref={ref}  
 className={cn(labelVariants(), className)}  
 {...props}  
 />  
));  
Label.displayName = LabelPrimitive.Root.displayName;  
  
export { Label };  
  
  
  
================================================  
FILE: client/components/ui/menubar.tsx  
================================================  
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 focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 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 focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",  
 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>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubContent>  
>(({ className, ...props }, ref) => (  
 <MenubarPrimitive.SubContent  
 ref={ref}  
 className={cn(  
 "z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
));  
MenubarSubContent.displayName = MenubarPrimitive.SubContent.displayName;  
  
const MenubarContent = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Content>  
>(  
 (  
 { className, align = "start", alignOffset = -4, sideOffset = 8, ...props },  
 ref,  
 ) => (  
 <MenubarPrimitive.Portal>  
 <MenubarPrimitive.Content  
 ref={ref}  
 align={align}  
 alignOffset={alignOffset}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 min-w-[12rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
 </MenubarPrimitive.Portal>  
 ),  
);  
MenubarContent.displayName = MenubarPrimitive.Content.displayName;  
  
const MenubarItem = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Item> & {  
 inset?: boolean;  
 }  
>(({ className, inset, ...props }, ref) => (  
 <MenubarPrimitive.Item  
 ref={ref}  
 className={cn(  
 "relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 inset && "pl-8",  
 className,  
 )}  
 {...props}  
 />  
));  
MenubarItem.displayName = MenubarPrimitive.Item.displayName;  
  
const MenubarCheckboxItem = React.forwardRef<  
 React.ElementRef<typeof MenubarPrimitive.CheckboxItem>,  
 React.ComponentPropsWithoutRef<typeof MenubarPrimitive.CheckboxItem>  
>(({ className, children, checked, ...props }, ref) => (  
 <MenubarPrimitive.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 focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className,  
 )}  
 checked={checked}  
 {...props}  
 >  
 <span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">  
 <MenubarPrimitive.ItemIndicator>  
 <Check className="h-4 w-4" />  
 </MenubarPrimitive.ItemIndicator>  
 </span>  
 {children}  
 </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 focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 className,  
 )}  
 {...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: client/components/ui/navigation-menu.tsx  
================================================  
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 transition-colors hover:bg-accent hover:text-accent-foreground focus:bg-accent focus:text-accent-foreground focus:outline-none disabled:pointer-events-none disabled:opacity-50 data-[active]:bg-accent/50 data-[state=open]:bg-accent/50",  
);  
  
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 data-[motion^=to-]:fade-out data-[motion=from-end]:slide-in-from-right-52 data-[motion=from-start]:slide-in-from-left-52 data-[motion=to-end]:slide-out-to-right-52 data-[motion=to-start]:slide-out-to-left-52 md:absolute md:w-auto ",  
 className,  
 )}  
 {...props}  
 />  
));  
NavigationMenuContent.displayName = NavigationMenuPrimitive.Content.displayName;  
  
const NavigationMenuLink = NavigationMenuPrimitive.Link;  
  
const NavigationMenuViewport = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Viewport>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Viewport>  
>(({ className, ...props }, ref) => (  
 <div className={cn("absolute left-0 top-full flex justify-center")}>  
 <NavigationMenuPrimitive.Viewport  
 className={cn(  
 "origin-top-center relative mt-1.5 h-[var(--radix-navigation-menu-viewport-height)] w-full overflow-hidden rounded-md border bg-popover text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-90 md:w-[var(--radix-navigation-menu-viewport-width)]",  
 className,  
 )}  
 ref={ref}  
 {...props}  
 />  
 </div>  
));  
NavigationMenuViewport.displayName =  
 NavigationMenuPrimitive.Viewport.displayName;  
  
const NavigationMenuIndicator = React.forwardRef<  
 React.ElementRef<typeof NavigationMenuPrimitive.Indicator>,  
 React.ComponentPropsWithoutRef<typeof NavigationMenuPrimitive.Indicator>  
>(({ className, ...props }, ref) => (  
 <NavigationMenuPrimitive.Indicator  
 ref={ref}  
 className={cn(  
 "top-full z-[1] flex h-1.5 items-end justify-center overflow-hidden data-[state=visible]:animate-in data-[state=hidden]:animate-out data-[state=hidden]:fade-out data-[state=visible]:fade-in",  
 className,  
 )}  
 {...props}  
 >  
 <div className="relative top-[60%] h-2 w-2 rotate-45 rounded-tl-sm bg-border shadow-md" />  
 </NavigationMenuPrimitive.Indicator>  
));  
NavigationMenuIndicator.displayName =  
 NavigationMenuPrimitive.Indicator.displayName;  
  
export {  
 navigationMenuTriggerStyle,  
 NavigationMenu,  
 NavigationMenuList,  
 NavigationMenuItem,  
 NavigationMenuContent,  
 NavigationMenuTrigger,  
 NavigationMenuLink,  
 NavigationMenuIndicator,  
 NavigationMenuViewport,  
};  
  
  
  
================================================  
FILE: client/components/ui/pagination.tsx  
================================================  
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: client/components/ui/popover.tsx  
================================================  
import \* as React from "react";  
import \* as PopoverPrimitive from "@radix-ui/react-popover";  
  
import { cn } from "@/lib/utils";  
  
const Popover = PopoverPrimitive.Root;  
  
const PopoverTrigger = PopoverPrimitive.Trigger;  
  
const PopoverContent = React.forwardRef<  
 React.ElementRef<typeof PopoverPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof PopoverPrimitive.Content>  
>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (  
 <PopoverPrimitive.Portal>  
 <PopoverPrimitive.Content  
 ref={ref}  
 align={align}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 w-72 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
 </PopoverPrimitive.Portal>  
));  
PopoverContent.displayName = PopoverPrimitive.Content.displayName;  
  
export { Popover, PopoverTrigger, PopoverContent };  
  
  
  
================================================  
FILE: client/components/ui/progress.tsx  
================================================  
import \* as React from "react";  
import \* as ProgressPrimitive from "@radix-ui/react-progress";  
  
import { cn } from "@/lib/utils";  
  
const Progress = React.forwardRef<  
 React.ElementRef<typeof ProgressPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>  
>(({ className, value, ...props }, ref) => (  
 <ProgressPrimitive.Root  
 ref={ref}  
 className={cn(  
 "relative h-4 w-full overflow-hidden rounded-full bg-secondary",  
 className,  
 )}  
 {...props}  
 >  
 <ProgressPrimitive.Indicator  
 className="h-full w-full flex-1 bg-primary transition-all"  
 style={{ transform: `translateX(-${100 - (value || 0)}%)` }}  
 />  
 </ProgressPrimitive.Root>  
));  
Progress.displayName = ProgressPrimitive.Root.displayName;  
  
export { Progress };  
  
  
  
================================================  
FILE: client/components/ui/radio-group.tsx  
================================================  
import \* as React from "react";  
import \* as RadioGroupPrimitive from "@radix-ui/react-radio-group";  
import { Circle } from "lucide-react";  
  
import { cn } from "@/lib/utils";  
  
const RadioGroup = React.forwardRef<  
 React.ElementRef<typeof RadioGroupPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Root>  
>(({ className, ...props }, ref) => {  
 return (  
 <RadioGroupPrimitive.Root  
 className={cn("grid gap-2", className)}  
 {...props}  
 ref={ref}  
 />  
 );  
});  
RadioGroup.displayName = RadioGroupPrimitive.Root.displayName;  
  
const RadioGroupItem = React.forwardRef<  
 React.ElementRef<typeof RadioGroupPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Item>  
>(({ className, ...props }, ref) => {  
 return (  
 <RadioGroupPrimitive.Item  
 ref={ref}  
 className={cn(  
 "aspect-square h-4 w-4 rounded-full border border-primary text-primary ring-offset-background focus:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",  
 className,  
 )}  
 {...props}  
 >  
 <RadioGroupPrimitive.Indicator className="flex items-center justify-center">  
 <Circle className="h-2.5 w-2.5 fill-current text-current" />  
 </RadioGroupPrimitive.Indicator>  
 </RadioGroupPrimitive.Item>  
 );  
});  
RadioGroupItem.displayName = RadioGroupPrimitive.Item.displayName;  
  
export { RadioGroup, RadioGroupItem };  
  
  
  
================================================  
FILE: client/components/ui/resizable.tsx  
================================================  
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 after:-translate-x-1/2 focus-visible:outline-none focus-visible:ring-1 focus-visible:ring-ring focus-visible:ring-offset-1 data-[panel-group-direction=vertical]:h-px data-[panel-group-direction=vertical]:w-full data-[panel-group-direction=vertical]:after:left-0 data-[panel-group-direction=vertical]:after:h-1 data-[panel-group-direction=vertical]:after:w-full data-[panel-group-direction=vertical]:after:-translate-y-1/2 data-[panel-group-direction=vertical]:after:translate-x-0 [&[data-panel-group-direction=vertical]>div]:rotate-90",  
 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: client/components/ui/scroll-area.tsx  
================================================  
import \* as React from "react";  
import \* as ScrollAreaPrimitive from "@radix-ui/react-scroll-area";  
  
import { cn } from "@/lib/utils";  
  
const ScrollArea = React.forwardRef<  
 React.ElementRef<typeof ScrollAreaPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root>  
>(({ className, children, ...props }, ref) => (  
 <ScrollAreaPrimitive.Root  
 ref={ref}  
 className={cn("relative overflow-hidden", className)}  
 {...props}  
 >  
 <ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">  
 {children}  
 </ScrollAreaPrimitive.Viewport>  
 <ScrollBar />  
 <ScrollAreaPrimitive.Corner />  
 </ScrollAreaPrimitive.Root>  
));  
ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName;  
  
const ScrollBar = React.forwardRef<  
 React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,  
 React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>  
>(({ className, orientation = "vertical", ...props }, ref) => (  
 <ScrollAreaPrimitive.ScrollAreaScrollbar  
 ref={ref}  
 orientation={orientation}  
 className={cn(  
 "flex touch-none select-none transition-colors",  
 orientation === "vertical" &&  
 "h-full w-2.5 border-l border-l-transparent p-[1px]",  
 orientation === "horizontal" &&  
 "h-2.5 flex-col border-t border-t-transparent p-[1px]",  
 className,  
 )}  
 {...props}  
 >  
 <ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />  
 </ScrollAreaPrimitive.ScrollAreaScrollbar>  
));  
ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName;  
  
export { ScrollArea, ScrollBar };  
  
  
  
================================================  
FILE: client/components/ui/select.tsx  
================================================  
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-offset-background placeholder:text-muted-foreground focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 [&>span]:line-clamp-1",  
 className,  
 )}  
 {...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-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 position === "popper" &&  
 "data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=top]:-translate-y-1",  
 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 focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",  
 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,  
 SelectLabel,  
 SelectItem,  
 SelectSeparator,  
 SelectScrollUpButton,  
 SelectScrollDownButton,  
};  
  
  
  
================================================  
FILE: client/components/ui/separator.tsx  
================================================  
import \* as React from "react";  
import \* as SeparatorPrimitive from "@radix-ui/react-separator";  
  
import { cn } from "@/lib/utils";  
  
const Separator = React.forwardRef<  
 React.ElementRef<typeof SeparatorPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof SeparatorPrimitive.Root>  
>(  
 (  
 { className, orientation = "horizontal", decorative = true, ...props },  
 ref,  
 ) => (  
 <SeparatorPrimitive.Root  
 ref={ref}  
 decorative={decorative}  
 orientation={orientation}  
 className={cn(  
 "shrink-0 bg-border",  
 orientation === "horizontal" ? "h-[1px] w-full" : "h-full w-[1px]",  
 className,  
 )}  
 {...props}  
 />  
 ),  
);  
Separator.displayName = SeparatorPrimitive.Root.displayName;  
  
export { Separator };  
  
  
  
================================================  
FILE: client/components/ui/sheet.tsx  
================================================  
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]:fade-out-0 data-[state=open]:fade-in-0",  
 className,  
 )}  
 {...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]:animate-out data-[state=closed]:duration-300 data-[state=open]:duration-500",  
 {  
 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-from-left sm:max-w-sm",  
 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-from-right sm:max-w-sm",  
 },  
 },  
 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 = "right", className, children, ...props }, ref) => (  
 <SheetPortal>  
 <SheetOverlay />  
 <SheetPrimitive.Content  
 ref={ref}  
 className={cn(sheetVariants({ side }), className)}  
 {...props}  
 >  
 {children}  
 <SheetPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none data-[state=open]:bg-secondary">  
 <X className="h-4 w-4" />  
 <span className="sr-only">Close</span>  
 </SheetPrimitive.Close>  
 </SheetPrimitive.Content>  
 </SheetPortal>  
));  
SheetContent.displayName = SheetPrimitive.Content.displayName;  
  
const SheetHeader = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn(  
 "flex flex-col space-y-2 text-center sm:text-left",  
 className,  
 )}  
 {...props}  
 />  
);  
SheetHeader.displayName = "SheetHeader";  
  
const SheetFooter = ({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) => (  
 <div  
 className={cn(  
 "flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",  
 className,  
 )}  
 {...props}  
 />  
);  
SheetFooter.displayName = "SheetFooter";  
  
const SheetTitle = React.forwardRef<  
 React.ElementRef<typeof SheetPrimitive.Title>,  
 React.ComponentPropsWithoutRef<typeof SheetPrimitive.Title>  
>(({ className, ...props }, ref) => (  
 <SheetPrimitive.Title  
 ref={ref}  
 className={cn("text-lg font-semibold text-foreground", className)}  
 {...props}  
 />  
));  
SheetTitle.displayName = SheetPrimitive.Title.displayName;  
  
const SheetDescription = React.forwardRef<  
 React.ElementRef<typeof SheetPrimitive.Description>,  
 React.ComponentPropsWithoutRef<typeof SheetPrimitive.Description>  
>(({ className, ...props }, ref) => (  
 <SheetPrimitive.Description  
 ref={ref}  
 className={cn("text-sm text-muted-foreground", className)}  
 {...props}  
 />  
));  
SheetDescription.displayName = SheetPrimitive.Description.displayName;  
  
export {  
 Sheet,  
 SheetClose,  
 SheetContent,  
 SheetDescription,  
 SheetFooter,  
 SheetHeader,  
 SheetOverlay,  
 SheetPortal,  
 SheetTitle,  
 SheetTrigger,  
};  
  
  
  
================================================  
FILE: client/components/ui/sidebar.tsx  
================================================  
import \* as React from "react";  
import { Slot } from "@radix-ui/react-slot";  
import { VariantProps, cva } from "class-variance-authority";  
import { PanelLeft } from "lucide-react";  
  
import { useIsMobile } from "@/hooks/use-mobile";  
import { cn } from "@/lib/utils";  
import { Button } from "@/components/ui/button";  
import { Input } from "@/components/ui/input";  
import { Separator } from "@/components/ui/separator";  
import { Sheet, SheetContent } from "@/components/ui/sheet";  
import { Skeleton } from "@/components/ui/skeleton";  
import {  
 Tooltip,  
 TooltipContent,  
 TooltipProvider,  
 TooltipTrigger,  
} from "@/components/ui/tooltip";  
  
const SIDEBAR\_COOKIE\_NAME = "sidebar:state";  
const SIDEBAR\_COOKIE\_MAX\_AGE = 60 \* 60 \* 24 \* 7;  
const SIDEBAR\_WIDTH = "16rem";  
const SIDEBAR\_WIDTH\_MOBILE = "18rem";  
const SIDEBAR\_WIDTH\_ICON = "3rem";  
const SIDEBAR\_KEYBOARD\_SHORTCUT = "b";  
  
type SidebarContext = {  
 state: "expanded" | "collapsed";  
 open: boolean;  
 setOpen: (open: boolean) => void;  
 openMobile: boolean;  
 setOpenMobile: (open: boolean) => void;  
 isMobile: boolean;  
 toggleSidebar: () => void;  
};  
  
const SidebarContext = React.createContext<SidebarContext | null>(null);  
  
function useSidebar() {  
 const context = React.useContext(SidebarContext);  
 if (!context) {  
 throw new Error("useSidebar must be used within a SidebarProvider.");  
 }  
  
 return context;  
}  
  
const SidebarProvider = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div"> & {  
 defaultOpen?: boolean;  
 open?: boolean;  
 onOpenChange?: (open: boolean) => void;  
 }  
>(  
 (  
 {  
 defaultOpen = true,  
 open: openProp,  
 onOpenChange: setOpenProp,  
 className,  
 style,  
 children,  
 ...props  
 },  
 ref,  
 ) => {  
 const isMobile = useIsMobile();  
 const [openMobile, setOpenMobile] = React.useState(false);  
  
 // This is the internal state of the sidebar.  
 // We use openProp and setOpenProp for control from outside the component.  
 const [\_open, \_setOpen] = React.useState(defaultOpen);  
 const open = openProp ?? \_open;  
 const setOpen = React.useCallback(  
 (value: boolean | ((value: boolean) => boolean)) => {  
 const openState = typeof value === "function" ? value(open) : value;  
 if (setOpenProp) {  
 setOpenProp(openState);  
 } else {  
 \_setOpen(openState);  
 }  
  
 // 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  
 }  
 className={cn(  
 "group/sidebar-wrapper flex min-h-svh w-full has-[[data-variant=inset]]:bg-sidebar",  
 className,  
 )}  
 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";  
 }  
>(  
 (  
 {  
 side = "left",  
 variant = "sidebar",  
 collapsible = "offcanvas",  
 className,  
 children,  
 ...props  
 },  
 ref,  
 ) => {  
 const { isMobile, state, openMobile, setOpenMobile } = useSidebar();  
  
 if (collapsible === "none") {  
 return (  
 <div  
 className={cn(  
 "flex h-full w-[--sidebar-width] flex-col bg-sidebar text-sidebar-foreground",  
 className,  
 )}  
 ref={ref}  
 {...props}  
 >  
 {children}  
 </div>  
 );  
 }  
  
 if (isMobile) {  
 return (  
 <Sheet open={openMobile} onOpenChange={setOpenMobile} {...props}>  
 <SheetContent  
 data-sidebar="sidebar"  
 data-mobile="true"  
 className="w-[--sidebar-width] bg-sidebar p-0 text-sidebar-foreground [&>button]:hidden"  
 style={  
 {  
 "--sidebar-width": SIDEBAR\_WIDTH\_MOBILE,  
 } as React.CSSProperties  
 }  
 side={side}  
 >  
 <div className="flex h-full w-full flex-col">{children}</div>  
 </SheetContent>  
 </Sheet>  
 );  
 }  
  
 return (  
 <div  
 ref={ref}  
 className="group peer hidden md:block text-sidebar-foreground"  
 data-state={state}  
 data-collapsible={state === "collapsed" ? collapsible : ""}  
 data-variant={variant}  
 data-side={side}  
 >  
 {/\* This is what handles the sidebar gap on desktop \*/}  
 <div  
 className={cn(  
 "duration-200 relative h-svh w-[--sidebar-width] bg-transparent transition-[width] ease-linear",  
 "group-data-[collapsible=offcanvas]:w-0",  
 "group-data-[side=right]:rotate-180",  
 variant === "floating" || variant === "inset"  
 ? "group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)\_+\_theme(spacing.4))]"  
 : "group-data-[collapsible=icon]:w-[--sidebar-width-icon]",  
 )}  
 />  
 <div  
 className={cn(  
 "duration-200 fixed inset-y-0 z-10 hidden h-svh w-[--sidebar-width] transition-[left,right,width] ease-linear md:flex",  
 side === "left"  
 ? "left-0 group-data-[collapsible=offcanvas]:left-[calc(var(--sidebar-width)\*-1)]"  
 : "right-0 group-data-[collapsible=offcanvas]:right-[calc(var(--sidebar-width)\*-1)]",  
 // Adjust the padding for floating and inset variants.  
 variant === "floating" || variant === "inset"  
 ? "p-2 group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)\_+\_theme(spacing.4)\_+2px)]"  
 : "group-data-[collapsible=icon]:w-[--sidebar-width-icon] group-data-[side=left]:border-r group-data-[side=right]:border-l",  
 className,  
 )}  
 {...props}  
 >  
 <div  
 data-sidebar="sidebar"  
 className="flex h-full w-full flex-col bg-sidebar group-data-[variant=floating]:rounded-lg group-data-[variant=floating]:border group-data-[variant=floating]:border-sidebar-border group-data-[variant=floating]:shadow"  
 >  
 {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 after:left-1/2 after:w-[2px] hover:after:bg-sidebar-border group-data-[side=left]:-right-4 group-data-[side=right]:left-0 sm:flex",  
 "[[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-data-[collapsible=offcanvas]:hover:bg-sidebar",  
 "[[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-[state=collapsed]:peer-data-[variant=inset]:ml-2 md:peer-data-[variant=inset]:ml-0 md:peer-data-[variant=inset]:rounded-xl md:peer-data-[variant=inset]:shadow",  
 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}  
 />  
 );  
});  
SidebarGroup.displayName = "SidebarGroup";  
  
const SidebarGroupLabel = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div"> & { asChild?: boolean }  
>(({ className, asChild = false, ...props }, ref) => {  
 const Comp = asChild ? Slot : "div";  
  
 return (  
 <Comp  
 ref={ref}  
 data-sidebar="group-label"  
 className={cn(  
 "duration-200 flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium text-sidebar-foreground/70 outline-none ring-sidebar-ring transition-[margin,opa] ease-linear focus-visible:ring-2 [&>svg]:size-4 [&>svg]:shrink-0",  
 "group-data-[collapsible=icon]:-mt-8 group-data-[collapsible=icon]:opacity-0",  
 className,  
 )}  
 {...props}  
 />  
 );  
});  
SidebarGroupLabel.displayName = "SidebarGroupLabel";  
  
const SidebarGroupAction = React.forwardRef<  
 HTMLButtonElement,  
 React.ComponentProps<"button"> & { asChild?: boolean }  
>(({ className, asChild = false, ...props }, ref) => {  
 const Comp = asChild ? Slot : "button";  
  
 return (  
 <Comp  
 ref={ref}  
 data-sidebar="group-action"  
 className={cn(  
 "absolute right-3 top-3.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sidebar-foreground outline-none ring-sidebar-ring transition-transform hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 [&>svg]:size-4 [&>svg]:shrink-0",  
 // Increases the hit area of the button on mobile.  
 "after:absolute after:-inset-2 after:md:hidden",  
 "group-data-[collapsible=icon]:hidden",  
 className,  
 )}  
 {...props}  
 />  
 );  
});  
SidebarGroupAction.displayName = "SidebarGroupAction";  
  
const SidebarGroupContent = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div">  
>(({ className, ...props }, ref) => (  
 <div  
 ref={ref}  
 data-sidebar="group-content"  
 className={cn("w-full text-sm", className)}  
 {...props}  
 />  
));  
SidebarGroupContent.displayName = "SidebarGroupContent";  
  
const SidebarMenu = React.forwardRef<  
 HTMLUListElement,  
 React.ComponentProps<"ul">  
>(({ className, ...props }, ref) => (  
 <ul  
 ref={ref}  
 data-sidebar="menu"  
 className={cn("flex w-full min-w-0 flex-col gap-1", className)}  
 {...props}  
 />  
));  
SidebarMenu.displayName = "SidebarMenu";  
  
const SidebarMenuItem = React.forwardRef<  
 HTMLLIElement,  
 React.ComponentProps<"li">  
>(({ className, ...props }, ref) => (  
 <li  
 ref={ref}  
 data-sidebar="menu-item"  
 className={cn("group/menu-item relative", className)}  
 {...props}  
 />  
));  
SidebarMenuItem.displayName = "SidebarMenuItem";  
  
const sidebarMenuButtonVariants = cva(  
 "peer/menu-button flex w-full items-center gap-2 overflow-hidden rounded-md p-2 text-left text-sm outline-none ring-sidebar-ring transition-[width,height,padding] hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50 group-has-[[data-sidebar=menu-action]]/menu-item:pr-8 aria-disabled:pointer-events-none aria-disabled:opacity-50 data-[active=true]:bg-sidebar-accent data-[active=true]:font-medium data-[active=true]:text-sidebar-accent-foreground data-[state=open]:hover:bg-sidebar-accent data-[state=open]:hover:text-sidebar-accent-foreground group-data-[collapsible=icon]:!size-8 group-data-[collapsible=icon]:!p-2 [&>span:last-child]:truncate [&>svg]:size-4 [&>svg]:shrink-0",  
 {  
 variants: {  
 variant: {  
 default: "hover:bg-sidebar-accent hover:text-sidebar-accent-foreground",  
 outline:  
 "bg-background shadow-[0\_0\_0\_1px\_hsl(var(--sidebar-border))] hover:bg-sidebar-accent hover:text-sidebar-accent-foreground hover:shadow-[0\_0\_0\_1px\_hsl(var(--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 outline-none ring-sidebar-ring transition-transform hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 peer-hover/menu-button:text-sidebar-accent-foreground [&>svg]:size-4 [&>svg]:shrink-0",  
 // Increases the hit area of the button on mobile.  
 "after:absolute after:-inset-2 after:md:hidden",  
 "peer-data-[size=sm]/menu-button:top-1",  
 "peer-data-[size=default]/menu-button:top-1.5",  
 "peer-data-[size=lg]/menu-button:top-2.5",  
 "group-data-[collapsible=icon]:hidden",  
 showOnHover &&  
 "group-focus-within/menu-item:opacity-100 group-hover/menu-item:opacity-100 data-[state=open]:opacity-100 peer-data-[active=true]/menu-button:text-sidebar-accent-foreground md:opacity-0",  
 className,  
 )}  
 {...props}  
 />  
 );  
});  
SidebarMenuAction.displayName = "SidebarMenuAction";  
  
const SidebarMenuBadge = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div">  
>(({ className, ...props }, ref) => (  
 <div  
 ref={ref}  
 data-sidebar="menu-badge"  
 className={cn(  
 "absolute right-1 flex h-5 min-w-5 items-center justify-center rounded-md px-1 text-xs font-medium tabular-nums text-sidebar-foreground select-none pointer-events-none",  
 "peer-hover/menu-button:text-sidebar-accent-foreground peer-data-[active=true]/menu-button:text-sidebar-accent-foreground",  
 "peer-data-[size=sm]/menu-button:top-1",  
 "peer-data-[size=default]/menu-button:top-1.5",  
 "peer-data-[size=lg]/menu-button:top-2.5",  
 "group-data-[collapsible=icon]:hidden",  
 className,  
 )}  
 {...props}  
 />  
));  
SidebarMenuBadge.displayName = "SidebarMenuBadge";  
  
const SidebarMenuSkeleton = React.forwardRef<  
 HTMLDivElement,  
 React.ComponentProps<"div"> & {  
 showIcon?: boolean;  
 }  
>(({ className, showIcon = false, ...props }, ref) => {  
 // Random width between 50 to 90%.  
 const width = React.useMemo(() => {  
 return `${Math.floor(Math.random() \* 40) + 50}%`;  
 }, []);  
  
 return (  
 <div  
 ref={ref}  
 data-sidebar="menu-skeleton"  
 className={cn("rounded-md h-8 flex gap-2 px-2 items-center", className)}  
 {...props}  
 >  
 {showIcon && (  
 <Skeleton  
 className="size-4 rounded-md"  
 data-sidebar="menu-skeleton-icon"  
 />  
 )}  
 <Skeleton  
 className="h-4 flex-1 max-w-[--skeleton-width]"  
 data-sidebar="menu-skeleton-text"  
 style={  
 {  
 "--skeleton-width": width,  
 } as React.CSSProperties  
 }  
 />  
 </div>  
 );  
});  
SidebarMenuSkeleton.displayName = "SidebarMenuSkeleton";  
  
const SidebarMenuSub = React.forwardRef<  
 HTMLUListElement,  
 React.ComponentProps<"ul">  
>(({ className, ...props }, ref) => (  
 <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 outline-none ring-sidebar-ring hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50 aria-disabled:pointer-events-none aria-disabled:opacity-50 [&>span:last-child]:truncate [&>svg]:size-4 [&>svg]:shrink-0 [&>svg]:text-sidebar-accent-foreground",  
 "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,  
 SidebarTrigger,  
 useSidebar,  
};  
  
  
  
================================================  
FILE: client/components/ui/skeleton.tsx  
================================================  
import { cn } from "@/lib/utils";  
  
function Skeleton({  
 className,  
 ...props  
}: React.HTMLAttributes<HTMLDivElement>) {  
 return (  
 <div  
 className={cn("animate-pulse rounded-md bg-muted", className)}  
 {...props}  
 />  
 );  
}  
  
export { Skeleton };  
  
  
  
================================================  
FILE: client/components/ui/slider.tsx  
================================================  
import \* as React from "react";  
import \* as SliderPrimitive from "@radix-ui/react-slider";  
  
import { cn } from "@/lib/utils";  
  
const Slider = React.forwardRef<  
 React.ElementRef<typeof SliderPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof SliderPrimitive.Root>  
>(({ className, ...props }, ref) => (  
 <SliderPrimitive.Root  
 ref={ref}  
 className={cn(  
 "relative flex w-full touch-none select-none items-center",  
 className,  
 )}  
 {...props}  
 >  
 <SliderPrimitive.Track className="relative h-2 w-full grow overflow-hidden rounded-full bg-secondary">  
 <SliderPrimitive.Range className="absolute h-full bg-primary" />  
 </SliderPrimitive.Track>  
 <SliderPrimitive.Thumb className="block h-5 w-5 rounded-full border-2 border-primary bg-background ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50" />  
 </SliderPrimitive.Root>  
));  
Slider.displayName = SliderPrimitive.Root.displayName;  
  
export { Slider };  
  
  
  
================================================  
FILE: client/components/ui/sonner.tsx  
================================================  
import { useTheme } from "next-themes";  
import { Toaster as Sonner } from "sonner";  
  
type ToasterProps = React.ComponentProps<typeof Sonner>;  
  
const Toaster = ({ ...props }: ToasterProps) => {  
 const { theme = "system" } = useTheme();  
  
 return (  
 <Sonner  
 theme={theme as ToasterProps["theme"]}  
 className="toaster group"  
 toastOptions={{  
 classNames: {  
 toast:  
 "group toast group-[.toaster]:bg-background group-[.toaster]:text-foreground group-[.toaster]:border-border group-[.toaster]:shadow-lg",  
 description: "group-[.toast]:text-muted-foreground",  
 actionButton:  
 "group-[.toast]:bg-primary group-[.toast]:text-primary-foreground",  
 cancelButton:  
 "group-[.toast]:bg-muted group-[.toast]:text-muted-foreground",  
 },  
 }}  
 {...props}  
 />  
 );  
};  
  
export { Toaster };  
  
  
  
================================================  
FILE: client/components/ui/switch.tsx  
================================================  
import \* as React from "react";  
import \* as SwitchPrimitives from "@radix-ui/react-switch";  
  
import { cn } from "@/lib/utils";  
  
const Switch = React.forwardRef<  
 React.ElementRef<typeof SwitchPrimitives.Root>,  
 React.ComponentPropsWithoutRef<typeof SwitchPrimitives.Root>  
>(({ className, ...props }, ref) => (  
 <SwitchPrimitives.Root  
 className={cn(  
 "peer inline-flex h-6 w-11 shrink-0 cursor-pointer items-center rounded-full border-2 border-transparent transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 focus-visible:ring-offset-background disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=unchecked]:bg-input",  
 className,  
 )}  
 {...props}  
 ref={ref}  
 >  
 <SwitchPrimitives.Thumb  
 className={cn(  
 "pointer-events-none block h-5 w-5 rounded-full bg-background shadow-lg ring-0 transition-transform data-[state=checked]:translate-x-5 data-[state=unchecked]:translate-x-0",  
 )}  
 />  
 </SwitchPrimitives.Root>  
));  
Switch.displayName = SwitchPrimitives.Root.displayName;  
  
export { Switch };  
  
  
  
================================================  
FILE: client/components/ui/table.tsx  
================================================  
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 hover:bg-muted/50 data-[state=selected]:bg-muted",  
 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) => (  
 <caption  
 ref={ref}  
 className={cn("mt-4 text-sm text-muted-foreground", className)}  
 {...props}  
 />  
));  
TableCaption.displayName = "TableCaption";  
  
export {  
 Table,  
 TableHeader,  
 TableBody,  
 TableFooter,  
 TableHead,  
 TableRow,  
 TableCell,  
 TableCaption,  
};  
  
  
  
================================================  
FILE: client/components/ui/tabs.tsx  
================================================  
import \* as React from "react";  
import \* as TabsPrimitive from "@radix-ui/react-tabs";  
  
import { cn } from "@/lib/utils";  
  
const Tabs = TabsPrimitive.Root;  
  
const TabsList = React.forwardRef<  
 React.ElementRef<typeof TabsPrimitive.List>,  
 React.ComponentPropsWithoutRef<typeof TabsPrimitive.List>  
>(({ className, ...props }, ref) => (  
 <TabsPrimitive.List  
 ref={ref}  
 className={cn(  
 "inline-flex h-10 items-center justify-center rounded-md bg-muted p-1 text-muted-foreground",  
 className,  
 )}  
 {...props}  
 />  
));  
TabsList.displayName = TabsPrimitive.List.displayName;  
  
const TabsTrigger = React.forwardRef<  
 React.ElementRef<typeof TabsPrimitive.Trigger>,  
 React.ComponentPropsWithoutRef<typeof TabsPrimitive.Trigger>  
>(({ className, ...props }, ref) => (  
 <TabsPrimitive.Trigger  
 ref={ref}  
 className={cn(  
 "inline-flex items-center justify-center whitespace-nowrap rounded-sm px-3 py-1.5 text-sm font-medium ring-offset-background transition-all focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 data-[state=active]:bg-background data-[state=active]:text-foreground data-[state=active]:shadow-sm",  
 className,  
 )}  
 {...props}  
 />  
));  
TabsTrigger.displayName = TabsPrimitive.Trigger.displayName;  
  
const TabsContent = React.forwardRef<  
 React.ElementRef<typeof TabsPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof TabsPrimitive.Content>  
>(({ className, ...props }, ref) => (  
 <TabsPrimitive.Content  
 ref={ref}  
 className={cn(  
 "mt-2 ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2",  
 className,  
 )}  
 {...props}  
 />  
));  
TabsContent.displayName = TabsPrimitive.Content.displayName;  
  
export { Tabs, TabsList, TabsTrigger, TabsContent };  
  
  
  
================================================  
FILE: client/components/ui/textarea.tsx  
================================================  
import \* as React from "react";  
  
import { cn } from "@/lib/utils";  
  
export interface TextareaProps  
 extends React.TextareaHTMLAttributes<HTMLTextAreaElement> {}  
  
const Textarea = React.forwardRef<HTMLTextAreaElement, TextareaProps>(  
 ({ className, ...props }, ref) => {  
 return (  
 <textarea  
 className={cn(  
 "flex min-h-[80px] w-full rounded-md border border-input bg-background px-3 py-2 text-sm ring-offset-background placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",  
 className,  
 )}  
 ref={ref}  
 {...props}  
 />  
 );  
 },  
);  
Textarea.displayName = "Textarea";  
  
export { Textarea };  
  
  
  
================================================  
FILE: client/components/ui/toast.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 md:max-w-[420px]",  
 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 border p-6 pr-8 shadow-lg transition-all data-[swipe=cancel]:translate-x-0 data-[swipe=end]:translate-x-[var(--radix-toast-swipe-end-x)] data-[swipe=move]:translate-x-[var(--radix-toast-swipe-move-x)] data-[swipe=move]:transition-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[swipe=end]:animate-out data-[state=closed]:fade-out-80 data-[state=closed]:slide-out-to-right-full data-[state=open]:slide-in-from-top-full data-[state=open]:sm:slide-in-from-bottom-full",  
 {  
 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 ring-offset-background transition-colors hover:bg-secondary focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 group-[.destructive]:border-muted/40 group-[.destructive]:hover:border-destructive/30 group-[.destructive]:hover:bg-destructive group-[.destructive]:hover:text-destructive-foreground group-[.destructive]:focus:ring-destructive",  
 className,  
 )}  
 {...props}  
 />  
));  
ToastAction.displayName = ToastPrimitives.Action.displayName;  
  
const ToastClose = React.forwardRef<  
 React.ElementRef<typeof ToastPrimitives.Close>,  
 React.ComponentPropsWithoutRef<typeof ToastPrimitives.Close>  
>(({ className, ...props }, ref) => (  
 <ToastPrimitives.Close  
 ref={ref}  
 className={cn(  
 "absolute right-2 top-2 rounded-md p-1 text-foreground/50 opacity-0 transition-opacity hover:text-foreground focus:opacity-100 focus:outline-none focus:ring-2 group-hover:opacity-100 group-[.destructive]:text-red-300 group-[.destructive]:hover:text-red-50 group-[.destructive]:focus:ring-red-400 group-[.destructive]:focus:ring-offset-red-600",  
 className,  
 )}  
 toast-close=""  
 {...props}  
 >  
 <X className="h-4 w-4" />  
 </ToastPrimitives.Close>  
));  
ToastClose.displayName = ToastPrimitives.Close.displayName;  
  
const ToastTitle = React.forwardRef<  
 React.ElementRef<typeof ToastPrimitives.Title>,  
 React.ComponentPropsWithoutRef<typeof ToastPrimitives.Title>  
>(({ className, ...props }, ref) => (  
 <ToastPrimitives.Title  
 ref={ref}  
 className={cn("text-sm font-semibold", className)}  
 {...props}  
 />  
));  
ToastTitle.displayName = ToastPrimitives.Title.displayName;  
  
const ToastDescription = React.forwardRef<  
 React.ElementRef<typeof ToastPrimitives.Description>,  
 React.ComponentPropsWithoutRef<typeof ToastPrimitives.Description>  
>(({ className, ...props }, ref) => (  
 <ToastPrimitives.Description  
 ref={ref}  
 className={cn("text-sm opacity-90", className)}  
 {...props}  
 />  
));  
ToastDescription.displayName = ToastPrimitives.Description.displayName;  
  
type ToastProps = React.ComponentPropsWithoutRef<typeof Toast>;  
  
type ToastActionElement = React.ReactElement<typeof ToastAction>;  
  
export {  
 type ToastProps,  
 type ToastActionElement,  
 ToastProvider,  
 ToastViewport,  
 Toast,  
 ToastTitle,  
 ToastDescription,  
 ToastClose,  
 ToastAction,  
};  
  
  
  
================================================  
FILE: client/components/ui/toaster.tsx  
================================================  
import { useToast } from "@/hooks/use-toast";  
import {  
 Toast,  
 ToastClose,  
 ToastDescription,  
 ToastProvider,  
 ToastTitle,  
 ToastViewport,  
} from "@/components/ui/toast";  
  
export function Toaster() {  
 const { toasts } = useToast();  
  
 return (  
 <ToastProvider>  
 {toasts.map(function ({ id, title, description, action, ...props }) {  
 return (  
 <Toast key={id} {...props}>  
 <div className="grid gap-1">  
 {title && <ToastTitle>{title}</ToastTitle>}  
 {description && (  
 <ToastDescription>{description}</ToastDescription>  
 )}  
 </div>  
 {action}  
 <ToastClose />  
 </Toast>  
 );  
 })}  
 <ToastViewport />  
 </ToastProvider>  
 );  
}  
  
  
  
================================================  
FILE: client/components/ui/toggle-group.tsx  
================================================  
import \* as React from "react";  
import \* as ToggleGroupPrimitive from "@radix-ui/react-toggle-group";  
import { type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
import { toggleVariants } from "@/components/ui/toggle";  
  
const ToggleGroupContext = React.createContext<  
 VariantProps<typeof toggleVariants>  
>({  
 size: "default",  
 variant: "default",  
});  
  
const ToggleGroup = React.forwardRef<  
 React.ElementRef<typeof ToggleGroupPrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Root> &  
 VariantProps<typeof toggleVariants>  
>(({ className, variant, size, children, ...props }, ref) => (  
 <ToggleGroupPrimitive.Root  
 ref={ref}  
 className={cn("flex items-center justify-center gap-1", className)}  
 {...props}  
 >  
 <ToggleGroupContext.Provider value={{ variant, size }}>  
 {children}  
 </ToggleGroupContext.Provider>  
 </ToggleGroupPrimitive.Root>  
));  
  
ToggleGroup.displayName = ToggleGroupPrimitive.Root.displayName;  
  
const ToggleGroupItem = React.forwardRef<  
 React.ElementRef<typeof ToggleGroupPrimitive.Item>,  
 React.ComponentPropsWithoutRef<typeof ToggleGroupPrimitive.Item> &  
 VariantProps<typeof toggleVariants>  
>(({ className, children, variant, size, ...props }, ref) => {  
 const context = React.useContext(ToggleGroupContext);  
  
 return (  
 <ToggleGroupPrimitive.Item  
 ref={ref}  
 className={cn(  
 toggleVariants({  
 variant: context.variant || variant,  
 size: context.size || size,  
 }),  
 className,  
 )}  
 {...props}  
 >  
 {children}  
 </ToggleGroupPrimitive.Item>  
 );  
});  
  
ToggleGroupItem.displayName = ToggleGroupPrimitive.Item.displayName;  
  
export { ToggleGroup, ToggleGroupItem };  
  
  
  
================================================  
FILE: client/components/ui/toggle.tsx  
================================================  
import \* as React from "react";  
import \* as TogglePrimitive from "@radix-ui/react-toggle";  
import { cva, type VariantProps } from "class-variance-authority";  
  
import { cn } from "@/lib/utils";  
  
const toggleVariants = cva(  
 "inline-flex items-center justify-center rounded-md text-sm font-medium ring-offset-background transition-colors hover:bg-muted hover:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 data-[state=on]:bg-accent data-[state=on]:text-accent-foreground",  
 {  
 variants: {  
 variant: {  
 default: "bg-transparent",  
 outline:  
 "border border-input bg-transparent hover:bg-accent hover:text-accent-foreground",  
 },  
 size: {  
 default: "h-10 px-3",  
 sm: "h-9 px-2.5",  
 lg: "h-11 px-5",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 size: "default",  
 },  
 },  
);  
  
const Toggle = React.forwardRef<  
 React.ElementRef<typeof TogglePrimitive.Root>,  
 React.ComponentPropsWithoutRef<typeof TogglePrimitive.Root> &  
 VariantProps<typeof toggleVariants>  
>(({ className, variant, size, ...props }, ref) => (  
 <TogglePrimitive.Root  
 ref={ref}  
 className={cn(toggleVariants({ variant, size, className }))}  
 {...props}  
 />  
));  
  
Toggle.displayName = TogglePrimitive.Root.displayName;  
  
export { Toggle, toggleVariants };  
  
  
  
================================================  
FILE: client/components/ui/tooltip.tsx  
================================================  
import \* as React from "react";  
import \* as TooltipPrimitive from "@radix-ui/react-tooltip";  
  
import { cn } from "@/lib/utils";  
  
const TooltipProvider = TooltipPrimitive.Provider;  
  
const Tooltip = TooltipPrimitive.Root;  
  
const TooltipTrigger = TooltipPrimitive.Trigger;  
  
const TooltipContent = React.forwardRef<  
 React.ElementRef<typeof TooltipPrimitive.Content>,  
 React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content>  
>(({ className, sideOffset = 4, ...props }, ref) => (  
 <TooltipPrimitive.Content  
 ref={ref}  
 sideOffset={sideOffset}  
 className={cn(  
 "z-50 overflow-hidden rounded-md border bg-popover px-3 py-1.5 text-sm text-popover-foreground shadow-md animate-in fade-in-0 zoom-in-95 data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=closed]:zoom-out-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",  
 className,  
 )}  
 {...props}  
 />  
));  
TooltipContent.displayName = TooltipPrimitive.Content.displayName;  
  
export { Tooltip, TooltipTrigger, TooltipContent, TooltipProvider };  
  
  
  
================================================  
FILE: client/components/ui/use-toast.ts  
================================================  
import { useToast, toast } from "@/hooks/use-toast";  
  
export { useToast, toast };  
  
  
  
================================================  
FILE: client/contexts/AuthContext.tsx  
================================================  
import { createContext, useContext, useState, useEffect, ReactNode } from 'react';  
  
export interface User {  
 id: string;  
 name: string;  
 email: string;  
 nativeLanguages: string[];  
 targetLanguages: string[];  
 experienceLevel: string;  
 profilePicture?: string;  
 videoIntro?: string;  
 referralId: string;  
 joinedAt: Date;  
}  
  
interface AuthContextType {  
 user: User | null;  
 login: (userData: User) => void;  
 logout: () => void;  
 updateUser: (userData: Partial<User>) => void;  
 isAuthenticated: boolean;  
}  
  
const AuthContext = createContext<AuthContextType | undefined>(undefined);  
  
export const useAuth = () => {  
 const context = useContext(AuthContext);  
 if (context === undefined) {  
 throw new Error('useAuth must be used within an AuthProvider');  
 }  
 return context;  
};  
  
interface AuthProviderProps {  
 children: ReactNode;  
}  
  
export const AuthProvider = ({ children }: AuthProviderProps) => {  
 const [user, setUser] = useState<User | null>(null);  
  
 // Load user from localStorage on mount  
 useEffect(() => {  
 const savedUser = localStorage.getItem('languagekonnect\_user');  
 if (savedUser) {  
 try {  
 const userData = JSON.parse(savedUser);  
 setUser(userData);  
 } catch (error) {  
 console.error('Error parsing saved user data:', error);  
 localStorage.removeItem('languagekonnect\_user');  
 }  
 }  
 }, []);  
  
 const login = (userData: User) => {  
 setUser(userData);  
 localStorage.setItem('languagekonnect\_user', JSON.stringify(userData));  
 };  
  
 const logout = () => {  
 setUser(null);  
 localStorage.removeItem('languagekonnect\_user');  
 };  
  
 const updateUser = (userData: Partial<User>) => {  
 if (user) {  
 const updatedUser = { ...user, ...userData };  
 setUser(updatedUser);  
 localStorage.setItem('languagekonnect\_user', JSON.stringify(updatedUser));  
 }  
 };  
  
 const value = {  
 user,  
 login,  
 logout,  
 updateUser,  
 isAuthenticated: !!user,  
 };  
  
 return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;  
};  
  
  
  
================================================  
FILE: client/hooks/use-mobile.tsx  
================================================  
import \* as React from "react";  
  
const MOBILE\_BREAKPOINT = 768;  
  
export function useIsMobile() {  
 const [isMobile, setIsMobile] = React.useState<boolean | undefined>(  
 undefined,  
 );  
  
 React.useEffect(() => {  
 const mql = window.matchMedia(`(max-width: ${MOBILE\_BREAKPOINT - 1}px)`);  
 const onChange = () => {  
 setIsMobile(window.innerWidth < MOBILE\_BREAKPOINT);  
 };  
 mql.addEventListener("change", onChange);  
 setIsMobile(window.innerWidth < MOBILE\_BREAKPOINT);  
 return () => mql.removeEventListener("change", onChange);  
 }, []);  
  
 return !!isMobile;  
}  
  
  
  
================================================  
FILE: client/hooks/use-toast.ts  
================================================  
import \* as React from "react";  
  
import type { ToastActionElement, ToastProps } from "@/components/ui/toast";  
  
const TOAST\_LIMIT = 1;  
const TOAST\_REMOVE\_DELAY = 1000000;  
  
type ToasterToast = ToastProps & {  
 id: string;  
 title?: React.ReactNode;  
 description?: React.ReactNode;  
 action?: ToastActionElement;  
};  
  
const actionTypes = {  
 ADD\_TOAST: "ADD\_TOAST",  
 UPDATE\_TOAST: "UPDATE\_TOAST",  
 DISMISS\_TOAST: "DISMISS\_TOAST",  
 REMOVE\_TOAST: "REMOVE\_TOAST",  
} as const;  
  
let count = 0;  
  
function genId() {  
 count = (count + 1) % Number.MAX\_SAFE\_INTEGER;  
 return count.toString();  
}  
  
type ActionType = typeof actionTypes;  
  
type Action =  
 | {  
 type: ActionType["ADD\_TOAST"];  
 toast: ToasterToast;  
 }  
 | {  
 type: ActionType["UPDATE\_TOAST"];  
 toast: Partial<ToasterToast>;  
 }  
 | {  
 type: ActionType["DISMISS\_TOAST"];  
 toastId?: ToasterToast["id"];  
 }  
 | {  
 type: ActionType["REMOVE\_TOAST"];  
 toastId?: ToasterToast["id"];  
 };  
  
interface State {  
 toasts: ToasterToast[];  
}  
  
const toastTimeouts = new Map<string, ReturnType<typeof setTimeout>>();  
  
const addToRemoveQueue = (toastId: string) => {  
 if (toastTimeouts.has(toastId)) {  
 return;  
 }  
  
 const timeout = setTimeout(() => {  
 toastTimeouts.delete(toastId);  
 dispatch({  
 type: "REMOVE\_TOAST",  
 toastId: toastId,  
 });  
 }, TOAST\_REMOVE\_DELAY);  
  
 toastTimeouts.set(toastId, timeout);  
};  
  
export const reducer = (state: State, action: Action): State => {  
 switch (action.type) {  
 case "ADD\_TOAST":  
 return {  
 ...state,  
 toasts: [action.toast, ...state.toasts].slice(0, TOAST\_LIMIT),  
 };  
  
 case "UPDATE\_TOAST":  
 return {  
 ...state,  
 toasts: state.toasts.map((t) =>  
 t.id === action.toast.id ? { ...t, ...action.toast } : t,  
 ),  
 };  
  
 case "DISMISS\_TOAST": {  
 const { toastId } = action;  
  
 // ! Side effects ! - This could be extracted into a dismissToast() action,  
 // but I'll keep it here for simplicity  
 if (toastId) {  
 addToRemoveQueue(toastId);  
 } else {  
 state.toasts.forEach((toast) => {  
 addToRemoveQueue(toast.id);  
 });  
 }  
  
 return {  
 ...state,  
 toasts: state.toasts.map((t) =>  
 t.id === toastId || toastId === undefined  
 ? {  
 ...t,  
 open: false,  
 }  
 : t,  
 ),  
 };  
 }  
 case "REMOVE\_TOAST":  
 if (action.toastId === undefined) {  
 return {  
 ...state,  
 toasts: [],  
 };  
 }  
 return {  
 ...state,  
 toasts: state.toasts.filter((t) => t.id !== action.toastId),  
 };  
 }  
};  
  
const listeners: Array<(state: State) => void> = [];  
  
let memoryState: State = { toasts: [] };  
  
function dispatch(action: Action) {  
 memoryState = reducer(memoryState, action);  
 listeners.forEach((listener) => {  
 listener(memoryState);  
 });  
}  
  
type Toast = Omit<ToasterToast, "id">;  
  
function toast({ ...props }: Toast) {  
 const id = genId();  
  
 const update = (props: ToasterToast) =>  
 dispatch({  
 type: "UPDATE\_TOAST",  
 toast: { ...props, id },  
 });  
 const dismiss = () => dispatch({ type: "DISMISS\_TOAST", toastId: id });  
  
 dispatch({  
 type: "ADD\_TOAST",  
 toast: {  
 ...props,  
 id,  
 open: true,  
 onOpenChange: (open) => {  
 if (!open) dismiss();  
 },  
 },  
 });  
  
 return {  
 id: id,  
 dismiss,  
 update,  
 };  
}  
  
function useToast() {  
 const [state, setState] = React.useState<State>(memoryState);  
  
 React.useEffect(() => {  
 listeners.push(setState);  
 return () => {  
 const index = listeners.indexOf(setState);  
 if (index > -1) {  
 listeners.splice(index, 1);  
 }  
 };  
 }, [state]);  
  
 return {  
 ...state,  
 toast,  
 dismiss: (toastId?: string) => dispatch({ type: "DISMISS\_TOAST", toastId }),  
 };  
}  
  
export { useToast, toast };  
  
  
  
================================================  
FILE: client/lib/api.ts  
================================================  
// API configuration for LanguageKonnect  
export const API\_CONFIG = {  
 BASE\_URL: '', // Use relative URLs for local API  
 ENDPOINTS: {  
 LOGIN: '/api/auth/login',  
 GET\_PROFILE: '/api/auth/profile',  
 USER\_PROFILE: '/api/user/profile',  
 CONTEST\_ENTRY: '/api/contest/entry',  
 CONTEST\_LEADERBOARD: '/api/contest/leaderboard',  
 CONTEST\_VOTE: '/api/contest/vote',  
 CONTEST\_VOTES: '/api/contest/votes',  
 RAFFLE\_ENTRY: '/api/raffle-entry',  
 REFERRAL\_LEADERBOARD: '/api/referral/leaderboard',  
 },  
 // Auth token should be obtained from Slack - placeholder for now  
 AUTH\_TOKEN: import.meta.env.VITE\_API\_TOKEN || 'your-token-here',  
 CACHE\_DURATION: 60000, // 60 seconds  
};  
  
// Response types based on the API specification  
export interface UserProfile {  
 name: string;  
 email: string;  
 nativeLanguages: string[];  
 targetLanguages: string[];  
 experienceLevel: string;  
 profilePicture?: string;  
 videoIntro?: string;  
}  
  
export interface ContestEntry {  
 id: string;  
 language: string;  
 region: string;  
 caption: string;  
 videoUrl: string;  
 votes: number;  
 createdAt: string;  
 userId: string;  
}  
  
export interface LeaderboardEntry {  
 id: string;  
 userName: string;  
 score: number;  
 rank: number;  
 language?: string;  
 region?: string;  
}  
  
export interface RaffleEntry {  
 userId: string;  
 referralId: string;  
 ticketCount: number;  
}  
  
export interface ReferralStats {  
 userId: string;  
 referralId: string;  
 totalReferrals: number;  
 ticketsEarned: number;  
 rank: number;  
}  
  
// API utility class  
class LanguageKonnectAPI {  
 private baseUrl: string;  
 private authToken: string;  
  
 constructor() {  
 this.baseUrl = API\_CONFIG.BASE\_URL;  
 this.authToken = API\_CONFIG.AUTH\_TOKEN;  
 }  
  
 // Helper method to make API requests  
 private async makeRequest<T>(  
 endpoint: string,  
 options: RequestInit = {}  
 ): Promise<T> {  
 const url = `${this.baseUrl}${endpoint}`;  
  
 const defaultHeaders: Record<string, string> = {};  
  
 // Only set Content-Type for non-FormData requests  
 if (!(options.body instanceof FormData)) {  
 defaultHeaders['Content-Type'] = 'application/json';  
 }  
  
 // Only add auth header if we have a valid token  
 if (this.authToken && this.authToken !== 'your-token-here') {  
 defaultHeaders['Authorization'] = `Bearer ${this.authToken}`;  
 }  
  
 const response = await fetch(url, {  
 ...options,  
 headers: {  
 ...defaultHeaders,  
 ...options.headers,  
 },  
 });  
  
 if (!response.ok) {  
 let errorText: string;  
 try {  
 errorText = await response.text();  
 } catch {  
 errorText = `HTTP ${response.status} ${response.statusText}`;  
 }  
 throw new Error(`API Error ${response.status}: ${errorText}`);  
 }  
  
 // Handle empty responses - check content type before reading body  
 const contentType = response.headers.get('content-type');  
 if (contentType && contentType.includes('application/json')) {  
 return response.json();  
 }  
  
 return response.text() as T;  
 }  
  
 // Voting endpoints  
 async voteOnEntry(entryId: string, userId: string, action: 'like' | 'unlike'): Promise<{ success: boolean; votes: number; isLiked: boolean }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; votes: number; isLiked: boolean }>(  
 API\_CONFIG.ENDPOINTS.CONTEST\_VOTE,  
 {  
 method: 'POST',  
 body: JSON.stringify({ entryId, userId, action }),  
 }  
 );  
  
 console.log('Vote processed:', result);  
 return result;  
 } catch (error) {  
 console.error('Error processing vote:', error);  
 throw error;  
 }  
 }  
  
 async getVotes(entryId: string): Promise<{ success: boolean; votes: number }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; votes: number }>(  
 `${API\_CONFIG.ENDPOINTS.CONTEST\_VOTES}/${entryId}`  
 );  
  
 return result;  
 } catch (error) {  
 console.error('Error fetching votes:', error);  
 throw error;  
 }  
 }  
  
 // Authentication endpoints  
 async login(email: string, password: string): Promise<{ success: boolean; message: string; user?: any }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; message: string; user?: any }>(  
 API\_CONFIG.ENDPOINTS.LOGIN,  
 {  
 method: 'POST',  
 body: JSON.stringify({ email, password }),  
 }  
 );  
  
 console.log('User login successful:', result);  
 return result;  
 } catch (error) {  
 console.error('Error during login:', error);  
 throw error;  
 }  
 }  
  
 async getUserProfile(userId: string): Promise<{ success: boolean; user?: any }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; user?: any }>(  
 `${API\_CONFIG.ENDPOINTS.GET\_PROFILE}/${userId}`  
 );  
  
 console.log('User profile fetched:', result);  
 return result;  
 } catch (error) {  
 console.error('Error fetching user profile:', error);  
 throw error;  
 }  
 }  
  
 // User Profile endpoints  
 async createUserProfile(profileData: UserProfile): Promise<{ success: boolean; message: string; data?: any }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; message: string }>(  
 API\_CONFIG.ENDPOINTS.USER\_PROFILE,  
 {  
 method: 'POST',  
 body: JSON.stringify(profileData),  
 }  
 );  
   
 console.log('User profile created successfully:', result);  
 return result;  
 } catch (error) {  
 console.error('Error creating user profile:', error);  
 throw error;  
 }  
 }  
  
 // Contest endpoints  
 async submitContestEntry(entryData: {  
 language: string;  
 region: string;  
 caption: string;  
 videoFile: File;  
 }): Promise<{ success: boolean; entryId: string }> {  
 try {  
 const formData = new FormData();  
 formData.append('language', entryData.language);  
 formData.append('region', entryData.region);  
 formData.append('caption', entryData.caption);  
 formData.append('video', entryData.videoFile);  
  
 const result = await this.makeRequest<{ success: boolean; entryId: string }>(  
 API\_CONFIG.ENDPOINTS.CONTEST\_ENTRY,  
 {  
 method: 'POST',  
 body: formData,  
 }  
 );  
  
 console.log('Contest entry submitted successfully:', result);  
 return result;  
 } catch (error) {  
 console.error('Error submitting contest entry:', error);  
 throw error;  
 }  
 }  
  
 async getContestLeaderboard(filters?: {  
 sortBy?: 'votes' | 'recent' | 'rating';  
 language?: string;  
 region?: string;  
 }): Promise<LeaderboardEntry[]> {  
 try {  
 const queryParams = new URLSearchParams();  
 if (filters) {  
 Object.entries(filters).forEach(([key, value]) => {  
 if (value) queryParams.append(key, value);  
 });  
 }  
  
 const url = `${API\_CONFIG.ENDPOINTS.CONTEST\_LEADERBOARD}${queryParams.toString() ? `?${queryParams.toString()}` : ''}`;  
 const result = await this.makeRequest<LeaderboardEntry[]>(url);  
   
 console.log('Contest leaderboard fetched:', result);  
 return result;  
 } catch (error) {  
 console.error('Error fetching contest leaderboard:', error);  
 throw error;  
 }  
 }  
  
 // Raffle and referral endpoints  
 async submitRaffleEntry(referralId: string): Promise<{ success: boolean; ticketsEarned: number }> {  
 try {  
 const result = await this.makeRequest<{ success: boolean; ticketsEarned: number }>(  
 API\_CONFIG.ENDPOINTS.RAFFLE\_ENTRY,  
 {  
 method: 'POST',  
 body: JSON.stringify({ referralId }),  
 }  
 );  
  
 console.log('Raffle entry submitted successfully:', result);  
 return result;  
 } catch (error) {  
 console.error('Error submitting raffle entry:', error);  
 throw error;  
 }  
 }  
  
 async getReferralLeaderboard(): Promise<ReferralStats[]> {  
 try {  
 const result = await this.makeRequest<ReferralStats[]>(  
 API\_CONFIG.ENDPOINTS.REFERRAL\_LEADERBOARD  
 );  
   
 console.log('Referral leaderboard fetched:', result);  
 return result;  
 } catch (error) {  
 console.error('Error fetching referral leaderboard:', error);  
 throw error;  
 }  
 }  
  
 // Utility method to generate unique referral ID  
 generateReferralId(userId: string): string {  
 const timestamp = Date.now().toString(36);  
 const randomStr = Math.random().toString(36).substr(2, 5);  
 return `${userId}-${timestamp}-${randomStr}`.toUpperCase();  
 }  
}  
  
// Export a singleton instance  
export const api = new LanguageKonnectAPI();  
  
// Auto-refresh functionality for live updates  
export class LiveDataManager {  
 private refreshInterval: NodeJS.Timeout | null = null;  
 private listeners: Map<string, (() => void)[]> = new Map();  
  
 startAutoRefresh(endpoint: string, callback: () => void, interval: number = 300000) { // 5 minutes default  
 const key = endpoint;  
   
 if (!this.listeners.has(key)) {  
 this.listeners.set(key, []);  
 }  
   
 this.listeners.get(key)!.push(callback);  
  
 // Start refresh interval if not already running  
 if (!this.refreshInterval) {  
 this.refreshInterval = setInterval(() => {  
 this.listeners.forEach((callbacks) => {  
 callbacks.forEach(cb => cb());  
 });  
 }, interval);  
 }  
 }  
  
 stopAutoRefresh(endpoint: string, callback: () => void) {  
 const key = endpoint;  
 const callbacks = this.listeners.get(key);  
   
 if (callbacks) {  
 const index = callbacks.indexOf(callback);  
 if (index > -1) {  
 callbacks.splice(index, 1);  
 }  
   
 if (callbacks.length === 0) {  
 this.listeners.delete(key);  
 }  
 }  
  
 // Stop interval if no listeners  
 if (this.listeners.size === 0 && this.refreshInterval) {  
 clearInterval(this.refreshInterval);  
 this.refreshInterval = null;  
 }  
 }  
  
 cleanup() {  
 if (this.refreshInterval) {  
 clearInterval(this.refreshInterval);  
 this.refreshInterval = null;  
 }  
 this.listeners.clear();  
 }  
}  
  
export const liveDataManager = new LiveDataManager();  
  
  
  
================================================  
FILE: client/lib/utils.spec.ts  
================================================  
import { describe, it, expect } from "vitest";  
import { cn } from "./utils";  
  
describe("cn function", () => {  
 it("should merge classes correctly", () => {  
 expect(cn("text-red-500", "bg-blue-500")).toBe("text-red-500 bg-blue-500");  
 });  
  
 it("should handle conditional classes", () => {  
 const isActive = true;  
 expect(cn("base-class", isActive && "active-class")).toBe(  
 "base-class active-class",  
 );  
 });  
  
 it("should handle false and null conditions", () => {  
 const isActive = false;  
 expect(cn("base-class", isActive && "active-class", null)).toBe(  
 "base-class",  
 );  
 });  
  
 it("should merge tailwind classes properly", () => {  
 expect(cn("px-2 py-1", "px-4")).toBe("py-1 px-4");  
 });  
  
 it("should work with object notation", () => {  
 expect(cn("base", { conditional: true, "not-included": false })).toBe(  
 "base conditional",  
 );  
 });  
});  
  
  
  
================================================  
FILE: client/lib/utils.ts  
================================================  
import { clsx, type ClassValue } from "clsx";  
import { twMerge } from "tailwind-merge";  
  
export function cn(...inputs: ClassValue[]) {  
 return twMerge(clsx(inputs));  
}  
  
  
  
================================================  
FILE: client/pages/Contests.tsx  
================================================  
import { useState, useEffect } from "react";  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Input } from "@/components/ui/input";  
import { Label } from "@/components/ui/label";  
import { Textarea } from "@/components/ui/textarea";  
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";  
import { Badge } from "@/components/ui/badge";  
import { Dialog, DialogContent, DialogDescription, DialogHeader, DialogTitle, DialogTrigger } from "@/components/ui/dialog";  
import { Globe, Upload, Play, Heart, MessageCircle, Trophy, Video, Star, ThumbsUp, Eye } from "lucide-react";  
import { Link } from "react-router-dom";  
import { useToast } from "@/hooks/use-toast";  
import { useAuth } from "@/contexts/AuthContext";  
import { api } from "@/lib/api";  
import VideoPlayer from "@/components/VideoPlayer";  
  
interface ContestSubmission {  
 id: string;  
 userName: string;  
 language: string;  
 region: string;  
 caption: string;  
 videoUrl: string;  
 thumbnailUrl?: string;  
 votes: number;  
 rating: number;  
 views: number;  
 comments: number;  
 createdAt: Date;  
 isLiked?: boolean;  
}  
  
const LANGUAGES = [  
 "Spanish", "French", "German", "Italian", "Portuguese", "Russian",  
 "Chinese", "Japanese", "Korean", "Arabic", "Hindi", "Dutch"  
];  
  
const REGIONS = [  
 "North America", "South America", "Europe", "Asia", "Africa", "Oceania"  
];  
  
export default function Contests() {  
 const { toast } = useToast();  
 const { user, isAuthenticated, logout } = useAuth();  
 const [submissions, setSubmissions] = useState<ContestSubmission[]>([]);  
 const [loading, setLoading] = useState(true);  
 const [uploadDialogOpen, setUploadDialogOpen] = useState(false);  
 const [formData, setFormData] = useState({  
 language: "",  
 region: "",  
 caption: "",  
 videoFile: null as File | null  
 });  
 const [isSubmitting, setIsSubmitting] = useState(false);  
  
 const fetchSubmissions = async () => {  
 try {  
 setLoading(true);  
   
 // Mock contest submissions with video data  
 const mockSubmissions: ContestSubmission[] = [  
 {  
 id: "1",  
 userName: "Sofia Martinez",  
 language: "Spanish",  
 region: "South America",  
 caption: "Practicing Spanish pronunciation with my favorite song! 🎵",  
 videoUrl: "https://sample-videos.com/zip/10/mp4/SampleVideo\_1280x720\_1mb.mp4",  
 thumbnailUrl: "https://images.unsplash.com/photo-1494790108755-2616c9e31d76?w=400&h=300&fit=crop",  
 votes: 127,  
 rating: 4.8,  
 views: 1450,  
 comments: 23,  
 createdAt: new Date(Date.now() - 2 \* 60 \* 60 \* 1000), // 2 hours ago  
 isLiked: false  
 },  
 {  
 id: "2",   
 userName: "Liu Wei",  
 language: "Chinese",  
 region: "Asia",  
 caption: "Learning Mandarin through traditional poetry recitation 📚",  
 videoUrl: "https://sample-videos.com/zip/10/mp4/SampleVideo\_1280x720\_2mb.mp4",  
 thumbnailUrl: "https://images.unsplash.com/photo-1507003211169-0a1dd7228f2d?w=400&h=300&fit=crop",  
 votes: 95,  
 rating: 4.6,  
 views: 980,  
 comments: 18,  
 createdAt: new Date(Date.now() - 5 \* 60 \* 60 \* 1000), // 5 hours ago  
 isLiked: true  
 },  
 {  
 id: "3",  
 userName: "Emma Johnson",  
 language: "French",  
 region: "North America",  
 caption: "Bonjour! Sharing my French conversation practice from Montreal 🇫🇷",  
 videoUrl: "https://sample-videos.com/zip/10/mp4/SampleVideo\_1280x720\_5mb.mp4",  
 thumbnailUrl: "https://images.unsplash.com/photo-1438761681033-6461ffad8d80?w=400&h=300&fit=crop",  
 votes: 83,  
 rating: 4.4,  
 views: 756,  
 comments: 12,  
 createdAt: new Date(Date.now() - 8 \* 60 \* 60 \* 1000), // 8 hours ago  
 isLiked: false  
 }  
 ];  
  
 // Add user's submission if they exist (checking localStorage for recent submission)  
 if (user) {  
 const recentSubmission = localStorage.getItem(`user\_submission\_${user.id}`);  
 if (recentSubmission) {  
 try {  
 const submissionData = JSON.parse(recentSubmission);  
 const userSubmission: ContestSubmission = {  
 id: `user-${user.id}`,  
 userName: user.name,  
 language: submissionData.language || user.targetLanguages[0] || "Spanish",  
 region: submissionData.region || "Your Region",  
 caption: submissionData.caption || "My language learning progress - thanks LanguageKonnect! 🌟",  
 videoUrl: submissionData.videoUrl || "https://sample-videos.com/zip/10/mp4/SampleVideo\_1280x720\_1mb.mp4",  
 thumbnailUrl: "https://images.unsplash.com/photo-1472099645785-5658abf4ff4e?w=400&h=300&fit=crop",  
 votes: 0, // Always start with 0 votes for new submissions  
 rating: 0, // Always start with 0 rating  
 views: 0, // Start with 0 views  
 comments: 0, // Start with 0 comments  
 createdAt: new Date(submissionData.createdAt || Date.now()),  
 isLiked: false  
 };  
 mockSubmissions.push(userSubmission);  
 } catch (error) {  
 console.error('Error parsing user submission:', error);  
 }  
 }  
 }  
  
 // Sort by votes (descending)  
 mockSubmissions.sort((a, b) => b.votes - a.votes);  
 setSubmissions(mockSubmissions);  
 } catch (error) {  
 console.error('Error fetching submissions:', error);  
 toast({  
 title: "Error",  
 description: "Failed to load contest submissions",  
 variant: "destructive"  
 });  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 useEffect(() => {  
 fetchSubmissions();  
 }, [user]);  
  
 const handleSubmit = async (e: React.FormEvent) => {  
 e.preventDefault();  
 if (!isAuthenticated) {  
 toast({  
 title: "Login Required",  
 description: "Please login to submit your video",  
 variant: "destructive"  
 });  
 return;  
 }  
  
 setIsSubmitting(true);  
 try {  
 if (!formData.language || !formData.region || !formData.caption || !formData.videoFile) {  
 throw new Error("Please fill in all fields and select a video");  
 }  
  
 const response = await api.submitContestEntry({  
 language: formData.language,  
 region: formData.region,  
 caption: formData.caption,  
 videoFile: formData.videoFile  
 });  
  
 // Store user's submission in localStorage for immediate display  
 if (user) {  
 const submissionData = {  
 language: formData.language,  
 region: formData.region,  
 caption: formData.caption,  
 videoUrl: response.videoUrl || "https://sample-videos.com/zip/10/mp4/SampleVideo\_1280x720\_1mb.mp4",  
 createdAt: new Date().toISOString()  
 };  
 localStorage.setItem(`user\_submission\_${user.id}`, JSON.stringify(submissionData));  
 }  
  
 toast({  
 title: "✅ Video submitted!",  
 description: "Your contest entry has been uploaded successfully!",  
 });  
  
 // Reset form and close dialog  
 setFormData({  
 language: "",  
 region: "",  
 caption: "",  
 videoFile: null  
 });  
 setUploadDialogOpen(false);  
  
 // Refresh submissions  
 fetchSubmissions();  
  
 } catch (error) {  
 toast({  
 title: "Upload Failed",  
 description: error instanceof Error ? error.message : "Failed to upload video",  
 variant: "destructive"  
 });  
 } finally {  
 setIsSubmitting(false);  
 }  
 };  
  
 const handleVote = async (submissionId: string) => {  
 if (!isAuthenticated || !user) {  
 toast({  
 title: "Login Required",  
 description: "Please login to vote",  
 variant: "destructive"  
 });  
 return;  
 }  
  
 try {  
 const currentSubmission = submissions.find(s => s.id === submissionId);  
 if (!currentSubmission) return;  
  
 const action = currentSubmission.isLiked ? 'unlike' : 'like';  
  
 // Call voting API  
 const response = await api.voteOnEntry(submissionId, user.id, action);  
  
 // Update votes with real response  
 setSubmissions(prev => prev.map(submission =>  
 submission.id === submissionId  
 ? {  
 ...submission,  
 votes: response.votes,  
 isLiked: response.isLiked  
 }  
 : submission  
 ));  
  
 toast({  
 title: "Vote Recorded!",  
 description: "Thanks for voting! Your support helps the community grow.",  
 });  
 } catch (error) {  
 console.error('Error voting:', error);  
 toast({  
 title: "Error",  
 description: "Failed to record vote. Please try again.",  
 variant: "destructive"  
 });  
 }  
 };  
  
 const getTimeAgo = (date: Date) => {  
 const now = new Date();  
 const diffInHours = Math.floor((now.getTime() - date.getTime()) / (1000 \* 60 \* 60));  
   
 if (diffInHours < 1) return "Just now";  
 if (diffInHours < 24) return `${diffInHours}h ago`;  
 return `${Math.floor(diffInHours / 24)}d ago`;  
 };  
  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/leaderboard">  
 <Button variant="ghost" aria-label="Go to leaderboard">Leaderboard</Button>  
 </Link>  
 <Link to="/referrals">  
 <Button variant="ghost" aria-label="Go to referrals">Referrals</Button>  
 </Link>  
 {isAuthenticated ? (  
 <div className="flex items-center space-x-3">  
 <div className="text-sm">  
 <span className="text-muted-foreground">Welcome, </span>  
 <span className="font-medium">{user?.name}</span>  
 </div>  
 <Button variant="outline" onClick={logout} aria-label="Logout">  
 Logout  
 </Button>  
 </div>  
 ) : (  
 <Link to="/login">  
 <Button variant="outline" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 )}  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="py-8 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto">  
 {/\* Header \*/}  
 <div className="text-center mb-8">  
 <div className="flex items-center justify-center mb-4">  
 <Video className="w-8 h-8 text-primary mr-3" />  
 <h1 className="text-3xl sm:text-4xl font-bold text-foreground">  
 Language Contests  
 </h1>  
 </div>  
 <p className="text-xl text-muted-foreground max-w-2xl mx-auto mb-6">  
 Upload 60-second videos showcasing your language skills and vote for your favorites!  
 </p>  
   
 {/\* Upload Button \*/}  
 <Dialog open={uploadDialogOpen} onOpenChange={setUploadDialogOpen}>  
 <DialogTrigger asChild>  
 <Button size="lg" className="text-lg px-8 py-3" aria-label="Upload contest video">  
 <Upload className="mr-2 h-5 w-5" />  
 Submit Your Video  
 </Button>  
 </DialogTrigger>  
 <DialogContent className="max-w-2xl">  
 <DialogHeader>  
 <DialogTitle>Submit Contest Entry</DialogTitle>  
 <DialogDescription>  
 Upload your 60-second language video and join the competition!  
 </DialogDescription>  
 </DialogHeader>  
 <form onSubmit={handleSubmit} className="space-y-4">  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-4">  
 <div className="space-y-2">  
 <Label htmlFor="language">Language \*</Label>  
 <Select value={formData.language} onValueChange={(value) => setFormData(prev => ({ ...prev, language: value }))}>  
 <SelectTrigger aria-label="Select language">  
 <SelectValue placeholder="Select language" />  
 </SelectTrigger>  
 <SelectContent>  
 {LANGUAGES.map((lang) => (  
 <SelectItem key={lang} value={lang}>{lang}</SelectItem>  
 ))}  
 </SelectContent>  
 </Select>  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="region">Region \*</Label>  
 <Select value={formData.region} onValueChange={(value) => setFormData(prev => ({ ...prev, region: value }))}>  
 <SelectTrigger aria-label="Select region">  
 <SelectValue placeholder="Select region" />  
 </SelectTrigger>  
 <SelectContent>  
 {REGIONS.map((region) => (  
 <SelectItem key={region} value={region}>{region}</SelectItem>  
 ))}  
 </SelectContent>  
 </Select>  
 </div>  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="caption">Caption \*</Label>  
 <Textarea  
 id="caption"  
 value={formData.caption}  
 onChange={(e) => setFormData(prev => ({ ...prev, caption: e.target.value }))}  
 placeholder="Describe your video and language learning journey..."  
 maxLength={200}  
 aria-label="Enter video caption"  
 />  
 <div className="text-sm text-muted-foreground text-right">  
 {formData.caption.length}/200  
 </div>  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="video">Video File (MP4 only, max 60 seconds) \*</Label>  
 <div className="border-2 border-dashed border-muted-foreground/25 rounded-lg p-6 text-center">  
 <Video className="w-12 h-12 text-muted-foreground mx-auto mb-4" />  
 <Input  
 id="video"  
 type="file"  
 accept="video/mp4"  
 onChange={(e) => setFormData(prev => ({ ...prev, videoFile: e.target.files?.[0] || null }))}  
 className="hidden"  
 aria-label="Upload video file"  
 />  
 <Label htmlFor="video" className="cursor-pointer">  
 {formData.videoFile ? (  
 <div>  
 <p className="font-medium text-primary">{formData.videoFile.name}</p>  
 <p className="text-sm text-muted-foreground">Click to change file</p>  
 </div>  
 ) : (  
 <div>  
 <p className="font-medium">Click to upload video</p>  
 <p className="text-sm text-muted-foreground">MP4 format, max 60 seconds</p>  
 </div>  
 )}  
 </Label>  
 </div>  
 </div>  
 <div className="flex justify-end space-x-3">  
 <Button type="button" variant="outline" onClick={() => setUploadDialogOpen(false)}>  
 Cancel  
 </Button>  
 <Button type="submit" disabled={isSubmitting}>  
 {isSubmitting ? "Uploading..." : "Submit Entry"}  
 </Button>  
 </div>  
 </form>  
 </DialogContent>  
 </Dialog>  
 </div>  
  
 {/\* Contest Submissions \*/}  
 <div className="space-y-6">  
 <div className="flex items-center justify-between">  
 <h2 className="text-2xl font-bold text-foreground">Latest Submissions</h2>  
 <Button variant="outline" onClick={fetchSubmissions} disabled={loading}>  
 {loading ? "Loading..." : "Refresh"}  
 </Button>  
 </div>  
  
 {loading ? (  
 <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">  
 {[1, 2, 3].map((i) => (  
 <Card key={i} className="animate-pulse">  
 <div className="aspect-video bg-muted rounded-t-lg"></div>  
 <CardContent className="p-4">  
 <div className="h-4 bg-muted rounded mb-2"></div>  
 <div className="h-3 bg-muted rounded w-3/4"></div>  
 </CardContent>  
 </Card>  
 ))}  
 </div>  
 ) : submissions.length === 0 ? (  
 <Card className="text-center py-12">  
 <CardContent>  
 <Video className="w-16 h-16 text-muted-foreground mx-auto mb-4" />  
 <h3 className="text-lg font-semibold mb-2">No submissions yet</h3>  
 <p className="text-muted-foreground mb-4">Be the first to upload a contest video!</p>  
 <Button onClick={() => setUploadDialogOpen(true)}>  
 Submit First Video  
 </Button>  
 </CardContent>  
 </Card>  
 ) : (  
 <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">  
 {submissions.map((submission) => (  
 <Card key={submission.id} className="overflow-hidden hover:shadow-lg transition-shadow">  
 <div className="relative">  
 <VideoPlayer  
 src={submission.videoUrl}  
 thumbnail={submission.thumbnailUrl}  
 title={`${submission.userName} - ${submission.caption}`}  
 className="rounded-t-lg"  
 />  
 <div className="absolute top-2 right-2">  
 <Badge variant="secondary" className="bg-black/70 text-white">  
 {getTimeAgo(submission.createdAt)}  
 </Badge>  
 </div>  
 </div>  
   
 <CardContent className="p-4">  
 <div className="flex items-start justify-between mb-2">  
 <div>  
 <h3 className="font-semibold text-lg">{submission.userName}</h3>  
 <div className="flex items-center gap-2 text-sm text-muted-foreground">  
 <Badge variant="outline" className="text-xs">{submission.language}</Badge>  
 <span>•</span>  
 <span>{submission.region}</span>  
 </div>  
 </div>  
 <div className="flex items-center gap-1">  
 <Star className="w-4 h-4 fill-yellow-400 text-yellow-400" />  
 <span className="text-sm font-medium">{submission.rating.toFixed(1)}</span>  
 </div>  
 </div>  
   
 <p className="text-sm text-muted-foreground mb-4 line-clamp-2">  
 {submission.caption}  
 </p>  
   
 <div className="flex items-center justify-between">  
 <div className="flex items-center gap-4 text-sm text-muted-foreground">  
 <div className="flex items-center gap-1">  
 <Eye className="w-4 h-4" />  
 <span>{submission.views.toLocaleString()}</span>  
 </div>  
 <div className="flex items-center gap-1">  
 <MessageCircle className="w-4 h-4" />  
 <span>{submission.comments}</span>  
 </div>  
 </div>  
   
 <Button  
 variant={submission.isLiked ? "default" : "outline"}  
 size="sm"  
 onClick={() => handleVote(submission.id)}  
 className="flex items-center gap-1"  
 aria-label={submission.isLiked ? "Unlike video" : "Like video"}  
 >  
 <ThumbsUp className={`w-4 h-4 ${submission.isLiked ? 'fill-current' : ''}`} />  
 <span>{submission.votes}</span>  
 </Button>  
 </div>  
 </CardContent>  
 </Card>  
 ))}  
 </div>  
 )}  
 </div>  
  
 {/\* Call to Action \*/}  
 {!isAuthenticated && (  
 <Card className="mt-12 text-center bg-gradient-to-r from-primary/10 to-accent/10">  
 <CardContent className="py-12">  
 <Trophy className="w-16 h-16 text-primary mx-auto mb-4" />  
 <h3 className="text-2xl font-bold mb-4">Join the Competition!</h3>  
 <p className="text-muted-foreground mb-6 max-w-md mx-auto">  
 Create your account to upload videos, vote for favorites, and climb the leaderboard!  
 </p>  
 <div className="flex flex-col sm:flex-row gap-3 justify-center">  
 <Link to="/signup">  
 <Button size="lg">Sign Up Now</Button>  
 </Link>  
 <Link to="/login">  
 <Button variant="outline" size="lg">Login</Button>  
 </Link>  
 </div>  
 </CardContent>  
 </Card>  
 )}  
 </div>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Index.tsx  
================================================  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Badge } from "@/components/ui/badge";  
import { Globe, Users, Trophy, Gift, MessageCircle, Star, ArrowRight } from "lucide-react";  
import { Link } from "react-router-dom";  
  
export default function Index() {  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <div className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </div>  
 <div className="flex items-center space-x-4">  
 <Link to="/login">  
 <Button variant="ghost" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 <Link to="/signup">  
 <Button aria-label="Sign up for LanguageKonnect">Sign Up</Button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Hero Section \*/}  
 <section className="pt-16 pb-24 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto text-center">  
 <div className="max-w-3xl mx-auto">  
 <Badge variant="secondary" className="mb-6 px-4 py-2">  
 🌍 Connect • Learn • Compete  
 </Badge>  
 <h1 className="text-4xl sm:text-5xl lg:text-6xl font-bold text-foreground mb-6">  
 Connect with{" "}  
 <span className="bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 Language Learners  
 </span>{" "}  
 Worldwide  
 </h1>  
 <p className="text-xl text-muted-foreground mb-8 leading-relaxed">  
 Join the global community where language enthusiasts connect, learn together,   
 compete in exciting challenges, and win amazing prizes through our referral program.  
 </p>  
 <div className="flex flex-col sm:flex-row gap-4 justify-center">  
 <Link to="/signup">  
 <Button size="lg" className="w-full sm:w-auto" aria-label="Start your language journey">  
 Start Your Journey  
 <ArrowRight className="ml-2 h-4 w-4" />  
 </Button>  
 </Link>  
 <Link to="/contests">  
 <Button size="lg" variant="outline" className="w-full sm:w-auto" aria-label="Explore language contests">  
 Explore Contests  
 </Button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 </section>  
  
 {/\* Features Section \*/}  
 <section className="py-24 px-4 sm:px-6 lg:px-8 bg-muted/30">  
 <div className="max-w-7xl mx-auto">  
 <div className="text-center mb-16">  
 <h2 className="text-3xl sm:text-4xl font-bold text-foreground mb-4">  
 Everything You Need to Master Languages  
 </h2>  
 <p className="text-xl text-muted-foreground max-w-2xl mx-auto">  
 From personalized profiles to global competitions, we provide all the tools   
 you need for your language learning journey.  
 </p>  
 </div>  
   
 <div className="grid md:grid-cols-3 gap-8">  
 <Card className="border-0 shadow-lg bg-gradient-to-br from-card to-blue-50/20 hover:shadow-xl transition-shadow">  
 <CardHeader className="text-center pb-4">  
 <div className="w-12 h-12 bg-primary/10 rounded-lg mx-auto mb-4 flex items-center justify-center">  
 <Users className="h-6 w-6 text-primary" />  
 </div>  
 <CardTitle className="text-xl">Connect & Profile</CardTitle>  
 </CardHeader>  
 <CardContent className="text-center">  
 <CardDescription className="text-base leading-relaxed">  
 Create your personalized profile with native and target languages,   
 experience levels, and connect with learners worldwide.  
 </CardDescription>  
 <div className="mt-4 space-y-2">  
 <Badge variant="secondary" className="mr-2">Profile Creation</Badge>  
 <Badge variant="secondary" className="mr-2">Language Matching</Badge>  
 <Badge variant="secondary">Video Intros</Badge>  
 </div>  
 </CardContent>  
 </Card>  
  
 <Card className="border-0 shadow-lg bg-gradient-to-br from-card to-purple-50/20 hover:shadow-xl transition-shadow">  
 <CardHeader className="text-center pb-4">  
 <div className="w-12 h-12 bg-accent/10 rounded-lg mx-auto mb-4 flex items-center justify-center">  
 <Trophy className="h-6 w-6 text-accent" />  
 </div>  
 <CardTitle className="text-xl">Contests & Challenges</CardTitle>  
 </CardHeader>  
 <CardContent className="text-center">  
 <CardDescription className="text-base leading-relaxed">  
 Participate in 60-second video challenges, showcase your language skills,   
 and climb the live leaderboards with real-time updates.  
 </CardDescription>  
 <div className="mt-4 space-y-2">  
 <Badge variant="secondary" className="mr-2">Video Contests</Badge>  
 <Badge variant="secondary" className="mr-2">Live Leaderboards</Badge>  
 <Badge variant="secondary">Instant Results</Badge>  
 </div>  
 </CardContent>  
 </Card>  
  
 <Card className="border-0 shadow-lg bg-gradient-to-br from-card to-green-50/20 hover:shadow-xl transition-shadow">  
 <CardHeader className="text-center pb-4">  
 <div className="w-12 h-12 bg-green-500/10 rounded-lg mx-auto mb-4 flex items-center justify-center">  
 <Gift className="h-6 w-6 text-green-600" />  
 </div>  
 <CardTitle className="text-xl">Referrals & Rewards</CardTitle>  
 </CardHeader>  
 <CardContent className="text-center">  
 <CardDescription className="text-base leading-relaxed">  
 Earn raffle tickets by referring friends, track your progress,   
 and compete for amazing prizes in our daily referral challenges.  
 </CardDescription>  
 <div className="mt-4 space-y-2">  
 <Badge variant="secondary" className="mr-2">Referral System</Badge>  
 <Badge variant="secondary" className="mr-2">Raffle Tickets</Badge>  
 <Badge variant="secondary">Daily Prizes</Badge>  
 </div>  
 </CardContent>  
 </Card>  
 </div>  
 </div>  
 </section>  
  
 {/\* Stats Section \*/}  
 <section className="py-16 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto">  
 <div className="grid grid-cols-2 md:grid-cols-4 gap-8 text-center">  
 <div>  
 <div className="text-3xl font-bold text-primary mb-2">150+</div>  
 <div className="text-muted-foreground">Languages Supported</div>  
 </div>  
 <div>  
 <div className="text-3xl font-bold text-primary mb-2">50K+</div>  
 <div className="text-muted-foreground">Active Learners</div>  
 </div>  
 <div>  
 <div className="text-3xl font-bold text-primary mb-2">10K+</div>  
 <div className="text-muted-foreground">Video Submissions</div>  
 </div>  
 <div>  
 <div className="text-3xl font-bold text-primary mb-2">$25K+</div>  
 <div className="text-muted-foreground">Prizes Awarded</div>  
 </div>  
 </div>  
 </div>  
 </section>  
  
 {/\* Testimonials \*/}  
 <section className="py-24 px-4 sm:px-6 lg:px-8 bg-muted/30">  
 <div className="max-w-7xl mx-auto">  
 <div className="text-center mb-16">  
 <h2 className="text-3xl sm:text-4xl font-bold text-foreground mb-4">  
 What Our Community Says  
 </h2>  
 </div>  
   
 <div className="grid md:grid-cols-3 gap-8">  
 <Card className="border-0 shadow-lg">  
 <CardContent className="pt-6">  
 <div className="flex items-center mb-4">  
 {[1, 2, 3, 4, 5].map((i) => (  
 <Star key={i} className="w-4 h-4 fill-yellow-400 text-yellow-400" />  
 ))}  
 </div>  
 <p className="text-muted-foreground mb-4">  
 "LanguageKonnect helped me find amazing conversation partners. The video contests   
 are so much fun and really motivate me to practice!"  
 </p>  
 <div className="flex items-center">  
 <div className="w-8 h-8 bg-gradient-to-r from-pink-400 to-purple-400 rounded-full mr-3"></div>  
 <div>  
 <p className="font-medium">Sarah Chen</p>  
 <p className="text-sm text-muted-foreground">Learning Spanish</p>  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
  
 <Card className="border-0 shadow-lg">  
 <CardContent className="pt-6">  
 <div className="flex items-center mb-4">  
 {[1, 2, 3, 4, 5].map((i) => (  
 <Star key={i} className="w-4 h-4 fill-yellow-400 text-yellow-400" />  
 ))}  
 </div>  
 <p className="text-muted-foreground mb-4">  
 "The referral system is genius! I've earned so many raffle tickets by inviting friends.   
 The community here is incredibly supportive."  
 </p>  
 <div className="flex items-center">  
 <div className="w-8 h-8 bg-gradient-to-r from-blue-400 to-cyan-400 rounded-full mr-3"></div>  
 <div>  
 <p className="font-medium">Miguel Rodriguez</p>  
 <p className="text-sm text-muted-foreground">Learning Japanese</p>  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
  
 <Card className="border-0 shadow-lg">  
 <CardContent className="pt-6">  
 <div className="flex items-center mb-4">  
 {[1, 2, 3, 4, 5].map((i) => (  
 <Star key={i} className="w-4 h-4 fill-yellow-400 text-yellow-400" />  
 ))}  
 </div>  
 <p className="text-muted-foreground mb-4">  
 "I love the competitive aspect! The leaderboards push me to create better content   
 and really improve my pronunciation."  
 </p>  
 <div className="flex items-center">  
 <div className="w-8 h-8 bg-gradient-to-r from-green-400 to-emerald-400 rounded-full mr-3"></div>  
 <div>  
 <p className="font-medium">Emma Thompson</p>  
 <p className="text-sm text-muted-foreground">Learning French</p>  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
 </div>  
 </div>  
 </section>  
  
 {/\* CTA Section \*/}  
 <section className="py-24 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-4xl mx-auto text-center">  
 <h2 className="text-3xl sm:text-4xl font-bold text-foreground mb-6">  
 Ready to Start Your Language Journey?  
 </h2>  
 <p className="text-xl text-muted-foreground mb-8">  
 Join thousands of language learners from around the world. Create your profile,   
 participate in contests, and start winning today!  
 </p>  
 <Link to="/signup">  
 <Button size="lg" className="text-lg px-8 py-3" aria-label="Join LanguageKonnect now">  
 Join LanguageKonnect Now  
 <ArrowRight className="ml-2 h-5 w-5" />  
 </Button>  
 </Link>  
 </div>  
 </section>  
  
 {/\* Footer \*/}  
 <footer className="border-t bg-muted/30 py-12 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto">  
 <div className="grid md:grid-cols-4 gap-8">  
 <div>  
 <div className="flex items-center space-x-3 mb-4">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </div>  
 <p className="text-muted-foreground">  
 Connecting language learners worldwide through community, competition, and rewards.  
 </p>  
 </div>  
   
 <div>  
 <h4 className="font-semibold mb-4">Platform</h4>  
 <ul className="space-y-2 text-muted-foreground">  
 <li><Link to="/signup" className="hover:text-foreground transition-colors">Create Profile</Link></li>  
 <li><Link to="/contests" className="hover:text-foreground transition-colors">Contests</Link></li>  
 <li><Link to="/leaderboard" className="hover:text-foreground transition-colors">Leaderboard</Link></li>  
 <li><Link to="/referrals" className="hover:text-foreground transition-colors">Referrals</Link></li>  
 </ul>  
 </div>  
   
 <div>  
 <h4 className="font-semibold mb-4">Support</h4>  
 <ul className="space-y-2 text-muted-foreground">  
 <li><a href="mailto:support@languagekonnect.com" className="hover:text-foreground transition-colors">Help Center</a></li>  
 <li><a href="mailto:support@languagekonnect.com" className="hover:text-foreground transition-colors">Contact Us</a></li>  
 <li><Link to="/terms" className="hover:text-foreground transition-colors">Terms of Service</Link></li>  
 <li><Link to="/privacy" className="hover:text-foreground transition-colors">Privacy Policy</Link></li>  
 </ul>  
 </div>  
   
 <div>  
 <h4 className="font-semibold mb-4">Community</h4>  
 <ul className="space-y-2 text-muted-foreground">  
 <li><a href="#" className="hover:text-foreground transition-colors">Discord</a></li>  
 <li><a href="#" className="hover:text-foreground transition-colors">Twitter</a></li>  
 <li><a href="#" className="hover:text-foreground transition-colors">Instagram</a></li>  
 <li><a href="#" className="hover:text-foreground transition-colors">YouTube</a></li>  
 </ul>  
 </div>  
 </div>  
   
 <div className="border-t pt-8 mt-8 text-center text-muted-foreground">  
 <p>&copy; 2024 LanguageKonnect. All rights reserved.</p>  
 </div>  
 </div>  
 </footer>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Leaderboard.tsx  
================================================  
import { useState, useEffect } from "react";  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Badge } from "@/components/ui/badge";  
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";  
import { Globe, Trophy, Filter, Star, Users, TrendingUp, ArrowUp, ArrowDown } from "lucide-react";  
import { Link } from "react-router-dom";  
import { api } from "@/lib/api";  
import { useToast } from "@/hooks/use-toast";  
import { useAuth } from "@/contexts/AuthContext";  
  
interface LeaderboardEntry {  
 id: string;  
 userName: string;  
 score: number;  
 rank: number;  
 language?: string;  
 region?: string;  
 caption?: string;  
 rating?: number;  
 createdAt?: Date;  
}  
  
export default function Leaderboard() {  
 const { toast } = useToast();  
 const { user, isAuthenticated, logout } = useAuth();  
 const [leaderboardData, setLeaderboardData] = useState<LeaderboardEntry[]>([]);  
 const [loading, setLoading] = useState(true);  
 const [sortBy, setSortBy] = useState<'votes' | 'recent' | 'rating'>('votes');  
 const [filterLanguage, setFilterLanguage] = useState<string>('all');  
 const [filterRegion, setFilterRegion] = useState<string>('all');  
  
 const fetchLeaderboard = async () => {  
 try {  
 setLoading(true);  
 const filters: any = { sortBy };  
 if (filterLanguage && filterLanguage !== 'all') filters.language = filterLanguage;  
 if (filterRegion && filterRegion !== 'all') filters.region = filterRegion;  
   
 const data = await api.getContestLeaderboard(filters);  
  
 // Add authenticated user to leaderboard if they have submitted a contest entry  
 let updatedData = [...data];  
 if (isAuthenticated && user) {  
 const userExists = data.some(entry => entry.userName === user.name);  
 if (!userExists) {  
 // Check if user has a contest submission  
 const recentSubmission = localStorage.getItem(`user\_submission\_${user.id}`);  
 if (recentSubmission) {  
 try {  
 const submissionData = JSON.parse(recentSubmission);  
 const userEntry: LeaderboardEntry = {  
 id: user.id,  
 userName: user.name,  
 score: 0, // Start with 0 votes for new submissions  
 rank: 0, // Will be calculated below  
 language: submissionData.language || user.targetLanguages[0] || "Spanish",  
 region: submissionData.region || "Your Region",  
 caption: submissionData.caption || "My language learning journey!",  
 rating: 0, // Start with 0 rating  
 createdAt: new Date(submissionData.createdAt || user.joinedAt)  
 };  
 updatedData.push(userEntry);  
 } catch (error) {  
 console.error('Error parsing user submission for leaderboard:', error);  
 }  
 }  
 }  
 }  
  
 // Sort by score and assign ranks  
 updatedData.sort((a, b) => b.score - a.score);  
 updatedData = updatedData.map((entry, index) => ({  
 ...entry,  
 rank: index + 1  
 }));  
  
 setLeaderboardData(updatedData);  
 } catch (error) {  
 console.error('Error fetching leaderboard:', error);  
 toast({  
 title: "Error",  
 description: "Failed to load leaderboard data",  
 variant: "destructive"  
 });  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 useEffect(() => {  
 fetchLeaderboard();  
 }, [sortBy, filterLanguage, filterRegion]);  
  
 // Auto-refresh every 5 minutes  
 useEffect(() => {  
 const interval = setInterval(fetchLeaderboard, 300000);  
 return () => clearInterval(interval);  
 }, [sortBy, filterLanguage, filterRegion]);  
  
 const getRankIcon = (rank: number) => {  
 if (rank === 1) return <Trophy className="w-5 h-5 text-yellow-500" />;  
 if (rank === 2) return <Star className="w-5 h-5 text-gray-400" />;  
 if (rank === 3) return <Star className="w-5 h-5 text-orange-500" />;  
 return <span className="w-5 h-5 flex items-center justify-center text-muted-foreground font-semibold">{rank}</span>;  
 };  
  
 const getRankBadgeColor = (rank: number) => {  
 if (rank === 1) return "bg-gradient-to-r from-yellow-400 to-yellow-600";  
 if (rank === 2) return "bg-gradient-to-r from-gray-300 to-gray-500";  
 if (rank === 3) return "bg-gradient-to-r from-orange-400 to-orange-600";  
 return "bg-muted";  
 };  
  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/contests">  
 <Button variant="ghost" aria-label="Go to contests">Contests</Button>  
 </Link>  
 <Link to="/referrals">  
 <Button variant="ghost" aria-label="Go to referrals">Referrals</Button>  
 </Link>  
 {isAuthenticated ? (  
 <div className="flex items-center space-x-3">  
 <div className="text-sm">  
 <span className="text-muted-foreground">Welcome, </span>  
 <span className="font-medium">{user?.name}</span>  
 </div>  
 <Button variant="outline" onClick={logout} aria-label="Logout">  
 Logout  
 </Button>  
 </div>  
 ) : (  
 <Link to="/login">  
 <Button variant="outline" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 )}  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="py-8 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto">  
 {/\* Header \*/}  
 <div className="text-center mb-8">  
 <div className="flex items-center justify-center mb-4">  
 <Trophy className="w-8 h-8 text-primary mr-3" />  
 <h1 className="text-3xl sm:text-4xl font-bold text-foreground">  
 Contest Leaderboard  
 </h1>  
 </div>  
 <p className="text-xl text-muted-foreground max-w-2xl mx-auto">  
 See who's leading in our language contests. Compete, vote, and climb the ranks!  
 </p>  
 </div>  
  
 {/\* Stats Cards \*/}  
 <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-8">  
 <Card className="text-center">  
 <CardContent className="pt-6">  
 <Users className="w-8 h-8 text-primary mx-auto mb-2" />  
 <div className="text-2xl font-bold text-foreground">{leaderboardData.length}</div>  
 <div className="text-muted-foreground">Active Contestants</div>  
 </CardContent>  
 </Card>  
 <Card className="text-center">  
 <CardContent className="pt-6">  
 <TrendingUp className="w-8 h-8 text-green-600 mx-auto mb-2" />  
 <div className="text-2xl font-bold text-foreground">  
 {leaderboardData.reduce((sum, entry) => sum + entry.score, 0)}  
 </div>  
 <div className="text-muted-foreground">Total Votes</div>  
 </CardContent>  
 </Card>  
 <Card className="text-center">  
 <CardContent className="pt-6">  
 <Globe className="w-8 h-8 text-blue-600 mx-auto mb-2" />  
 <div className="text-2xl font-bold text-foreground">  
 {new Set(leaderboardData.map(entry => entry.language)).size}  
 </div>  
 <div className="text-muted-foreground">Languages</div>  
 </CardContent>  
 </Card>  
 </div>  
  
 {/\* Filters \*/}  
 <Card className="mb-8">  
 <CardHeader>  
 <div className="flex items-center gap-2">  
 <Filter className="w-5 h-5" />  
 <CardTitle>Filters & Sorting</CardTitle>  
 </div>  
 <CardDescription>  
 Filter and sort the leaderboard to find specific competitions  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 <div className="grid grid-cols-1 md:grid-cols-3 gap-4">  
 <div className="space-y-2">  
 <label className="text-sm font-medium">Sort By</label>  
 <Select value={sortBy} onValueChange={(value: any) => setSortBy(value)}>  
 <SelectTrigger aria-label="Sort leaderboard by">  
 <SelectValue />  
 </SelectTrigger>  
 <SelectContent>  
 <SelectItem value="votes">Most Votes</SelectItem>  
 <SelectItem value="rating">Highest Rating</SelectItem>  
 <SelectItem value="recent">Most Recent</SelectItem>  
 </SelectContent>  
 </Select>  
 </div>  
 <div className="space-y-2">  
 <label className="text-sm font-medium">Language</label>  
 <Select value={filterLanguage} onValueChange={setFilterLanguage}>  
 <SelectTrigger aria-label="Filter by language">  
 <SelectValue placeholder="All Languages" />  
 </SelectTrigger>  
 <SelectContent>  
 <SelectItem value="all">All Languages</SelectItem>  
 <SelectItem value="Spanish">Spanish</SelectItem>  
 <SelectItem value="French">French</SelectItem>  
 <SelectItem value="Japanese">Japanese</SelectItem>  
 <SelectItem value="German">German</SelectItem>  
 <SelectItem value="Chinese">Chinese</SelectItem>  
 </SelectContent>  
 </Select>  
 </div>  
 <div className="space-y-2">  
 <label className="text-sm font-medium">Region</label>  
 <Select value={filterRegion} onValueChange={setFilterRegion}>  
 <SelectTrigger aria-label="Filter by region">  
 <SelectValue placeholder="All Regions" />  
 </SelectTrigger>  
 <SelectContent>  
 <SelectItem value="all">All Regions</SelectItem>  
 <SelectItem value="North America">North America</SelectItem>  
 <SelectItem value="South America">South America</SelectItem>  
 <SelectItem value="Europe">Europe</SelectItem>  
 <SelectItem value="Asia">Asia</SelectItem>  
 <SelectItem value="Africa">Africa</SelectItem>  
 <SelectItem value="Oceania">Oceania</SelectItem>  
 </SelectContent>  
 </Select>  
 </div>  
 </div>  
 <div className="mt-4 flex justify-between items-center">  
 <Button   
 variant="outline"   
 onClick={fetchLeaderboard}  
 disabled={loading}  
 aria-label="Refresh leaderboard data"  
 >  
 {loading ? "Loading..." : "Refresh"}  
 </Button>  
 <div className="text-sm text-muted-foreground">  
 Auto-refreshes every 5 minutes  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
  
 {/\* Leaderboard \*/}  
 <Card>  
 <CardHeader>  
 <CardTitle className="flex items-center gap-2">  
 <Trophy className="w-5 h-5" />  
 Live Rankings  
 </CardTitle>  
 <CardDescription>  
 Real-time contest results updated instantly  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 {loading ? (  
 <div className="text-center py-8">  
 <div className="animate-spin h-8 w-8 border-4 border-primary border-t-transparent rounded-full mx-auto mb-4"></div>  
 <p className="text-muted-foreground">Loading leaderboard...</p>  
 </div>  
 ) : leaderboardData.length === 0 ? (  
 <div className="text-center py-8">  
 <Trophy className="w-12 h-12 text-muted-foreground mx-auto mb-4" />  
 <p className="text-muted-foreground">No contest entries found</p>  
 <Link to="/contests">  
 <Button className="mt-4" aria-label="Submit your first contest entry">  
 Submit Your First Entry  
 </Button>  
 </Link>  
 </div>  
 ) : (  
 <div className="space-y-4">  
 {leaderboardData.map((entry, index) => (  
 <div  
 key={entry.id}  
 className={`flex items-center justify-between p-4 rounded-lg border transition-colors hover:bg-muted/50 ${  
 entry.rank <= 3 ? 'bg-gradient-to-r from-primary/5 to-accent/5' : ''  
 }`}  
 >  
 <div className="flex items-center space-x-4 flex-1">  
 <div className={`w-12 h-12 rounded-lg flex items-center justify-center ${getRankBadgeColor(entry.rank)}`}>  
 {getRankIcon(entry.rank)}  
 </div>  
 <div className="flex-1">  
 <div className="flex items-center gap-2 mb-1">  
 <h3 className="font-semibold text-foreground">{entry.userName}</h3>  
 {entry.rank <= 3 && (  
 <Badge variant="secondary" className="text-xs">  
 Top {entry.rank}  
 </Badge>  
 )}  
 </div>  
 {entry.caption && (  
 <p className="text-sm text-muted-foreground mb-2">"{entry.caption}"</p>  
 )}  
 <div className="flex items-center gap-4 text-sm text-muted-foreground">  
 {entry.language && (  
 <Badge variant="outline" className="text-xs">  
 {entry.language}  
 </Badge>  
 )}  
 {entry.region && (  
 <Badge variant="outline" className="text-xs">  
 {entry.region}  
 </Badge>  
 )}  
 {entry.rating && (  
 <div className="flex items-center gap-1">  
 <Star className="w-3 h-3 fill-yellow-400 text-yellow-400" />  
 <span>{entry.rating.toFixed(1)}</span>  
 </div>  
 )}  
 </div>  
 </div>  
 </div>  
 <div className="text-right">  
 <div className="text-2xl font-bold text-primary">{entry.score}</div>  
 <div className="text-sm text-muted-foreground">votes</div>  
 {index < leaderboardData.length - 1 && entry.score > leaderboardData[index + 1]?.score && (  
 <ArrowUp className="w-4 h-4 text-green-600 mx-auto mt-1" />  
 )}  
 {index > 0 && entry.score < leaderboardData[index - 1]?.score && (  
 <ArrowDown className="w-4 h-4 text-red-600 mx-auto mt-1" />  
 )}  
 </div>  
 </div>  
 ))}  
 </div>  
 )}  
 </CardContent>  
 </Card>  
  
 {/\* Call to Action \*/}  
 <div className="text-center mt-8">  
 <Card className="max-w-md mx-auto">  
 <CardContent className="pt-6">  
 <Trophy className="w-12 h-12 text-primary mx-auto mb-4" />  
 <h3 className="text-lg font-semibold mb-2">Ready to Compete?</h3>  
 <p className="text-muted-foreground mb-4">  
 Upload your 60-second language video and join the competition!  
 </p>  
 <Link to="/contests">  
 <Button className="w-full" aria-label="Submit contest entry">  
 Submit Your Entry  
 </Button>  
 </Link>  
 </CardContent>  
 </Card>  
 </div>  
 </div>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Login.tsx  
================================================  
import { useState } from "react";  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Input } from "@/components/ui/input";  
import { Label } from "@/components/ui/label";  
import { Checkbox } from "@/components/ui/checkbox";  
import { Globe, ArrowRight } from "lucide-react";  
import { Link, useNavigate } from "react-router-dom";  
import { useToast } from "@/hooks/use-toast";  
import { useAuth } from "@/contexts/AuthContext";  
import { api } from "@/lib/api";  
  
interface LoginFormData {  
 email: string;  
 password: string;  
 rememberMe: boolean;  
}  
  
export default function Login() {  
 const { toast } = useToast();  
 const navigate = useNavigate();  
 const { login } = useAuth();  
 const [formData, setFormData] = useState<LoginFormData>({  
 email: "",  
 password: "",  
 rememberMe: false  
 });  
 const [isSubmitting, setIsSubmitting] = useState(false);  
  
 const handleSubmit = async (e: React.FormEvent) => {  
 e.preventDefault();  
 setIsSubmitting(true);  
  
 try {  
 // Validation  
 if (!formData.email || !formData.password) {  
 throw new Error("Please fill in all fields");  
 }  
  
 // Call login API  
 const response = await api.login(formData.email, formData.password);  
  
 if (response.success && response.user) {  
 login(response.user);  
 } else {  
 throw new Error(response.message || "Login failed");  
 }  
  
 // Show success toast  
 toast({  
 title: "✅ Login successful!",  
 description: "Welcome back to LanguageKonnect!",  
 });  
  
 // Reset form  
 setFormData({  
 email: "",  
 password: "",  
 rememberMe: false  
 });  
  
 // Redirect to leaderboard/dashboard after short delay  
 setTimeout(() => {  
 navigate("/leaderboard");  
 }, 1000);  
  
 } catch (error) {  
 toast({  
 title: "Login Failed",  
 description: error instanceof Error ? error.message : "Invalid credentials",  
 variant: "destructive"  
 });  
 } finally {  
 setIsSubmitting(false);  
 }  
 };  
  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/signup">  
 <Button aria-label="Sign up for LanguageKonnect">Sign Up</Button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="flex-1 flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-md w-full">  
 <div className="text-center mb-8">  
 <h1 className="text-3xl font-bold text-foreground mb-2">  
 Welcome Back  
 </h1>  
 <p className="text-muted-foreground">  
 Sign in to continue your language learning journey  
 </p>  
 </div>  
  
 <Card className="shadow-lg">  
 <CardHeader>  
 <CardTitle>Sign In</CardTitle>  
 <CardDescription>  
 Enter your credentials to access your LanguageKonnect account  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 <form onSubmit={handleSubmit} className="space-y-4">  
 <div className="space-y-2">  
 <Label htmlFor="email">Email Address</Label>  
 <Input  
 id="email"  
 type="email"  
 value={formData.email}  
 onChange={(e) => setFormData(prev => ({ ...prev, email: e.target.value }))}  
 placeholder="Enter your email"  
 aria-label="Enter your email address"  
 required  
 />  
 </div>  
  
 <div className="space-y-2">  
 <Label htmlFor="password">Password</Label>  
 <Input  
 id="password"  
 type="password"  
 value={formData.password}  
 onChange={(e) => setFormData(prev => ({ ...prev, password: e.target.value }))}  
 placeholder="Enter your password"  
 aria-label="Enter your password"  
 required  
 />  
 </div>  
  
 <div className="flex items-center justify-between">  
 <div className="flex items-center space-x-2">  
 <Checkbox  
 id="rememberMe"  
 checked={formData.rememberMe}  
 onCheckedChange={(checked) => setFormData(prev => ({ ...prev, rememberMe: checked as boolean }))}  
 aria-label="Remember me"  
 />  
 <Label htmlFor="rememberMe" className="text-sm">  
 Remember me  
 </Label>  
 </div>  
 <Button variant="link" className="px-0 text-sm" type="button">  
 Forgot password?  
 </Button>  
 </div>  
  
 <Button  
 type="submit"  
 className="w-full"  
 disabled={isSubmitting}  
 aria-label="Sign in to LanguageKonnect"  
 >  
 {isSubmitting ? "Signing In..." : "Sign In"}  
 {!isSubmitting && <ArrowRight className="ml-2 h-4 w-4" />}  
 </Button>  
 </form>  
  
 <div className="mt-6">  
 <div className="relative">  
 <div className="absolute inset-0 flex items-center">  
 <div className="w-full border-t border-muted-foreground/20" />  
 </div>  
 <div className="relative flex justify-center text-xs uppercase">  
 <span className="bg-card px-2 text-muted-foreground">Or continue with</span>  
 </div>  
 </div>  
  
 <div className="mt-4 grid grid-cols-2 gap-3">  
 <Button variant="outline" type="button" aria-label="Sign in with Google">  
 <svg className="w-4 h-4 mr-2" viewBox="0 0 24 24">  
 <path fill="currentColor" 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"/>  
 <path fill="currentColor" 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"/>  
 <path fill="currentColor" 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"/>  
 <path fill="currentColor" 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"/>  
 </svg>  
 Google  
 </Button>  
 <Button variant="outline" type="button" aria-label="Sign in with Facebook">  
 <svg className="w-4 h-4 mr-2" fill="currentColor" viewBox="0 0 24 24">  
 <path d="M24 12.073c0-6.627-5.373-12-12-12s-12 5.373-12 12c0 5.99 4.388 10.954 10.125 11.854v-8.385H7.078v-3.47h3.047V9.43c0-3.007 1.792-4.669 4.533-4.669 1.312 0 2.686.235 2.686.235v2.953H15.83c-1.491 0-1.956.925-1.956 1.874v2.25h3.328l-.532 3.47h-2.796v8.385C19.612 23.027 24 18.062 24 12.073z"/>  
 </svg>  
 Facebook  
 </Button>  
 </div>  
 </div>  
  
 <div className="mt-6 text-center">  
 <p className="text-sm text-muted-foreground">  
 Don't have an account?{" "}  
 <Link to="/signup" className="text-primary hover:underline">  
 Sign up  
 </Link>  
 </p>  
 </div>  
 </CardContent>  
 </Card>  
 </div>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/NotFound.tsx  
================================================  
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="min-h-screen flex items-center justify-center bg-gray-100">  
 <div className="text-center">  
 <h1 className="text-4xl font-bold mb-4">404</h1>  
 <p className="text-xl text-gray-600 mb-4">Oops! Page not found</p>  
 <a href="/" className="text-blue-500 hover:text-blue-700 underline">  
 Return to Home  
 </a>  
 </div>  
 </div>  
 );  
};  
  
export default NotFound;  
  
  
  
================================================  
FILE: client/pages/PlaceholderPage.tsx  
================================================  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Globe, ArrowRight, Construction } from "lucide-react";  
import { Link } from "react-router-dom";  
  
interface PlaceholderPageProps {  
 title: string;  
 description: string;  
 icon?: React.ReactNode;  
}  
  
export default function PlaceholderPage({   
 title,   
 description,   
 icon = <Construction className="w-8 h-8 text-primary" />  
}: PlaceholderPageProps) {  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/login">  
 <Button variant="ghost" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 <Link to="/signup">  
 <Button aria-label="Sign up for LanguageKonnect">Sign Up</Button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="flex-1 flex items-center justify-center p-8">  
 <Card className="max-w-md w-full text-center shadow-lg">  
 <CardHeader className="pb-4">  
 <div className="w-16 h-16 bg-primary/10 rounded-lg mx-auto mb-4 flex items-center justify-center">  
 {icon}  
 </div>  
 <CardTitle className="text-2xl">{title}</CardTitle>  
 <CardDescription className="text-base">  
 {description}  
 </CardDescription>  
 </CardHeader>  
 <CardContent className="space-y-4">  
 <p className="text-muted-foreground text-sm">  
 This page is currently under development. Check back soon for exciting updates!  
 </p>  
 <div className="flex flex-col sm:flex-row gap-3">  
 <Link to="/" className="flex-1">  
 <Button variant="outline" className="w-full">  
 Return Home  
 </Button>  
 </Link>  
 <Link to="/signup" className="flex-1">  
 <Button className="w-full" aria-label="Get started with LanguageKonnect">  
 Get Started  
 <ArrowRight className="ml-2 h-4 w-4" />  
 </Button>  
 </Link>  
 </div>  
 </CardContent>  
 </Card>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Privacy.tsx  
================================================  
import PlaceholderPage from "./PlaceholderPage";  
import { Shield } from "lucide-react";  
  
export default function Privacy() {  
 return (  
 <PlaceholderPage  
 title="Privacy Policy"  
 description="Learn how we protect your data and privacy while using LanguageKonnect's language learning and contest platform."  
 icon={<Shield className="w-8 h-8 text-primary" />}  
 />  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Referrals.tsx  
================================================  
import { useState, useEffect } from "react";  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Input } from "@/components/ui/input";  
import { Label } from "@/components/ui/label";  
import { Badge } from "@/components/ui/badge";  
import { Progress } from "@/components/ui/progress";  
import { Globe, Gift, Copy, Share2, Trophy, Users, Ticket, Crown, Medal, Award } from "lucide-react";  
import { Link } from "react-router-dom";  
import { useToast } from "@/hooks/use-toast";  
import { useAuth } from "@/contexts/AuthContext";  
import { api } from "@/lib/api";  
  
interface ReferralStats {  
 userId: string;  
 userName: string;  
 referralId: string;  
 totalReferrals: number;  
 ticketsEarned: number;  
 rank: number;  
}  
  
interface Prize {  
 id: string;  
 name: string;  
 description: string;  
 ticketsRequired: number;  
 imageUrl: string;  
 available: number;  
}  
  
export default function Referrals() {  
 const { toast } = useToast();  
 const { user, isAuthenticated, logout } = useAuth();  
 const [referralStats, setReferralStats] = useState<ReferralStats[]>([]);  
 const [userStats, setUserStats] = useState<ReferralStats | null>(null);  
 const [loading, setLoading] = useState(true);  
 const [prizes] = useState<Prize[]>([  
 {  
 id: "1",  
 name: "LanguageKonnect Premium (1 Year)",  
 description: "Access to premium features, advanced analytics, and exclusive content",  
 ticketsRequired: 50,  
 imageUrl: "https://images.unsplash.com/photo-1516321318423-f06f85e504b3?w=300&h=200&fit=crop",  
 available: 10  
 },  
 {  
 id: "2",   
 name: "Language Learning Course Bundle",  
 description: "Complete online courses for Spanish, French, and German",  
 ticketsRequired: 30,  
 imageUrl: "https://images.unsplash.com/photo-1456513080510-7bf3a84b82f8?w=300&h=200&fit=crop",  
 available: 25  
 },  
 {  
 id: "3",  
 name: "Wireless Headphones",  
 description: "High-quality noise-cancelling headphones perfect for language practice",  
 ticketsRequired: 25,  
 imageUrl: "https://images.unsplash.com/photo-1505740420928-5e560c06d30e?w=300&h=200&fit=crop",  
 available: 15  
 },  
 {  
 id: "4",  
 name: "Language Exchange Trip",  
 description: "3-day language immersion experience in Barcelona, Spain",  
 ticketsRequired: 100,  
 imageUrl: "https://images.unsplash.com/photo-1539037116277-4db20889f2d4?w=300&h=200&fit=crop",  
 available: 2  
 },  
 {  
 id: "5",  
 name: "LanguageKonnect Merchandise Kit",  
 description: "T-shirt, notebook, and branded language learning accessories",  
 ticketsRequired: 10,  
 imageUrl: "https://images.unsplash.com/photo-1521572163474-6864f9cf17ab?w=300&h=200&fit=crop",  
 available: 50  
 }  
 ]);  
  
 const fetchReferralData = async () => {  
 try {  
 setLoading(true);  
   
 // Mock referral leaderboard data  
 const mockReferralStats: ReferralStats[] = [  
 {  
 userId: "1",  
 userName: "Alex Rivera",  
 referralId: "LK-ALEX2024",  
 totalReferrals: 28,  
 ticketsEarned: 28,  
 rank: 1  
 },  
 {  
 userId: "2",  
 userName: "Maria Santos",  
 referralId: "LK-MARIA789",  
 totalReferrals: 22,  
 ticketsEarned: 22,  
 rank: 2  
 },  
 {  
 userId: "3",  
 userName: "David Kim",  
 referralId: "LK-DAVID456",  
 totalReferrals: 19,  
 ticketsEarned: 19,  
 rank: 3  
 },  
 {  
 userId: "4",  
 userName: "Sophie Chen",  
 referralId: "LK-SOPHIE123",  
 totalReferrals: 15,  
 ticketsEarned: 15,  
 rank: 4  
 },  
 {  
 userId: "5",  
 userName: "James Wilson",  
 referralId: "LK-JAMES999",  
 totalReferrals: 12,  
 ticketsEarned: 12,  
 rank: 5  
 }  
 ];  
  
 // Add user to leaderboard if authenticated  
 if (isAuthenticated && user) {  
 const existingUserIndex = mockReferralStats.findIndex(stat => stat.userName === user.name);  
   
 if (existingUserIndex === -1) {  
 // Add new user with some random stats  
 const userReferrals = Math.floor(Math.random() \* 10) + 1; // 1-10 referrals  
 const newUserStats: ReferralStats = {  
 userId: user.id,  
 userName: user.name,  
 referralId: user.referralId,  
 totalReferrals: userReferrals,  
 ticketsEarned: userReferrals,  
 rank: 0 // Will be calculated after sorting  
 };  
 mockReferralStats.push(newUserStats);  
 }  
 }  
  
 // Sort by total referrals and assign ranks  
 mockReferralStats.sort((a, b) => b.totalReferrals - a.totalReferrals);  
 const rankedStats = mockReferralStats.map((stat, index) => ({  
 ...stat,  
 rank: index + 1  
 }));  
  
 setReferralStats(rankedStats);  
  
 // Set user stats if authenticated  
 if (isAuthenticated && user) {  
 const currentUserStats = rankedStats.find(stat => stat.userName === user.name);  
 setUserStats(currentUserStats || null);  
 }  
  
 } catch (error) {  
 console.error('Error fetching referral data:', error);  
 toast({  
 title: "Error",  
 description: "Failed to load referral data",  
 variant: "destructive"  
 });  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 useEffect(() => {  
 fetchReferralData();  
 }, [user, isAuthenticated]);  
  
 const copyReferralLink = async () => {  
 if (!user) return;  
   
 const referralLink = `${window.location.origin}/signup?ref=${user.referralId}`;  
   
 try {  
 await navigator.clipboard.writeText(referralLink);  
 toast({  
 title: "Link Copied!",  
 description: "Your referral link has been copied to clipboard",  
 });  
 } catch (error) {  
 // Fallback for older browsers  
 const textArea = document.createElement('textarea');  
 textArea.value = referralLink;  
 document.body.appendChild(textArea);  
 textArea.select();  
 document.execCommand('copy');  
 document.body.removeChild(textArea);  
   
 toast({  
 title: "Link Copied!",  
 description: "Your referral link has been copied to clipboard",  
 });  
 }  
 };  
  
 const shareReferralLink = async () => {  
 if (!user) return;  
   
 const referralLink = `${window.location.origin}/signup?ref=${user.referralId}`;  
 const shareData = {  
 title: 'Join LanguageKonnect!',  
 text: 'Connect with language learners worldwide and join amazing contests!',  
 url: referralLink,  
 };  
  
 try {  
 if (navigator.share) {  
 await navigator.share(shareData);  
 } else {  
 // Fallback to copy  
 copyReferralLink();  
 }  
 } catch (error) {  
 console.error('Error sharing:', error);  
 }  
 };  
  
 const getRankIcon = (rank: number) => {  
 if (rank === 1) return <Crown className="w-5 h-5 text-yellow-500" />;  
 if (rank === 2) return <Medal className="w-5 h-5 text-gray-400" />;  
 if (rank === 3) return <Award className="w-5 h-5 text-orange-500" />;  
 return <span className="w-5 h-5 flex items-center justify-center text-muted-foreground font-semibold">#{rank}</span>;  
 };  
  
 const getPrizeProgressColor = (ticketsEarned: number, ticketsRequired: number) => {  
 const percentage = (ticketsEarned / ticketsRequired) \* 100;  
 if (percentage >= 100) return "bg-green-500";  
 if (percentage >= 75) return "bg-yellow-500";  
 if (percentage >= 50) return "bg-blue-500";  
 return "bg-gray-300";  
 };  
  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-green-50/30 to-blue-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/contests">  
 <Button variant="ghost" aria-label="Go to contests">Contests</Button>  
 </Link>  
 <Link to="/leaderboard">  
 <Button variant="ghost" aria-label="Go to leaderboard">Leaderboard</Button>  
 </Link>  
 {isAuthenticated ? (  
 <div className="flex items-center space-x-3">  
 <div className="text-sm">  
 <span className="text-muted-foreground">Welcome, </span>  
 <span className="font-medium">{user?.name}</span>  
 </div>  
 <Button variant="outline" onClick={logout} aria-label="Logout">  
 Logout  
 </Button>  
 </div>  
 ) : (  
 <Link to="/login">  
 <Button variant="outline" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 )}  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="py-8 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-7xl mx-auto">  
 {/\* Header \*/}  
 <div className="text-center mb-8">  
 <div className="flex items-center justify-center mb-4">  
 <Gift className="w-8 h-8 text-green-600 mr-3" />  
 <h1 className="text-3xl sm:text-4xl font-bold text-foreground">  
 Referrals & Rewards  
 </h1>  
 </div>  
 <p className="text-xl text-muted-foreground max-w-2xl mx-auto">  
 Invite friends to LanguageKonnect and earn raffle tickets for amazing prizes!  
 </p>  
 </div>  
  
 {isAuthenticated && user ? (  
 <>  
 {/\* User Stats \*/}  
 <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-8">  
 <Card className="bg-gradient-to-br from-green-50 to-green-100 border-green-200">  
 <CardContent className="pt-6 text-center">  
 <Users className="w-8 h-8 text-green-600 mx-auto mb-2" />  
 <div className="text-2xl font-bold text-green-700">  
 {userStats?.totalReferrals || 0}  
 </div>  
 <div className="text-green-600">Referrals Made</div>  
 </CardContent>  
 </Card>  
 <Card className="bg-gradient-to-br from-blue-50 to-blue-100 border-blue-200">  
 <CardContent className="pt-6 text-center">  
 <Ticket className="w-8 h-8 text-blue-600 mx-auto mb-2" />  
 <div className="text-2xl font-bold text-blue-700">  
 {userStats?.ticketsEarned || 0}  
 </div>  
 <div className="text-blue-600">Raffle Tickets</div>  
 </CardContent>  
 </Card>  
 <Card className="bg-gradient-to-br from-purple-50 to-purple-100 border-purple-200">  
 <CardContent className="pt-6 text-center">  
 <Trophy className="w-8 h-8 text-purple-600 mx-auto mb-2" />  
 <div className="text-2xl font-bold text-purple-700">  
 #{userStats?.rank || 'N/A'}  
 </div>  
 <div className="text-purple-600">Your Rank</div>  
 </CardContent>  
 </Card>  
 </div>  
  
 {/\* Referral Link Section \*/}  
 <Card className="mb-8">  
 <CardHeader>  
 <CardTitle className="flex items-center gap-2">  
 <Share2 className="w-5 h-5" />  
 Your Referral Link  
 </CardTitle>  
 <CardDescription>  
 Share this link with friends to earn raffle tickets when they sign up!  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 <div className="space-y-4">  
 <div className="flex items-center gap-2">  
 <Label htmlFor="referralId">Your Referral ID:</Label>  
 <Badge variant="secondary" className="font-mono text-lg px-3 py-1">  
 {user.referralId}  
 </Badge>  
 </div>  
 <div className="flex gap-2">  
 <Input  
 readOnly  
 value={`${window.location.origin}/signup?ref=${user.referralId}`}  
 className="font-mono text-sm"  
 aria-label="Your referral link"  
 />  
 <Button onClick={copyReferralLink} variant="outline" aria-label="Copy referral link">  
 <Copy className="w-4 h-4" />  
 </Button>  
 <Button onClick={shareReferralLink} aria-label="Share referral link">  
 <Share2 className="w-4 h-4" />  
 </Button>  
 </div>  
 <div className="bg-muted/50 rounded-lg p-4">  
 <h4 className="font-semibold mb-2">🎉 How it works:</h4>  
 <ul className="text-sm text-muted-foreground space-y-1">  
 <li>• Share your unique referral link with friends</li>  
 <li>• When they sign up using your link, you both earn rewards</li>  
 <li>• Collect raffle tickets to enter prize drawings</li>  
 <li>• Climb the leaderboard and compete for daily prizes</li>  
 </ul>  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
  
 {/\* Prizes Section \*/}  
 <Card className="mb-8">  
 <CardHeader>  
 <CardTitle className="flex items-center gap-2">  
 <Gift className="w-5 h-5" />  
 Available Prizes  
 </CardTitle>  
 <CardDescription>  
 Use your raffle tickets to win amazing prizes!  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">  
 {prizes.map((prize) => {  
 const userTickets = userStats?.ticketsEarned || 0;  
 const canAfford = userTickets >= prize.ticketsRequired;  
 const progress = Math.min((userTickets / prize.ticketsRequired) \* 100, 100);  
   
 return (  
 <Card key={prize.id} className={`overflow-hidden ${canAfford ? 'ring-2 ring-green-500' : ''}`}>  
 <div className="aspect-video relative">  
 <img   
 src={prize.imageUrl}   
 alt={prize.name}  
 className="w-full h-full object-cover"  
 />  
 {canAfford && (  
 <div className="absolute top-2 right-2">  
 <Badge className="bg-green-500 text-white">Available!</Badge>  
 </div>  
 )}  
 </div>  
 <CardContent className="p-4">  
 <h3 className="font-semibold mb-2">{prize.name}</h3>  
 <p className="text-sm text-muted-foreground mb-4 line-clamp-2">  
 {prize.description}  
 </p>  
 <div className="space-y-3">  
 <div className="flex items-center justify-between text-sm">  
 <span>Progress:</span>  
 <span className="font-medium">  
 {userTickets}/{prize.ticketsRequired} tickets  
 </span>  
 </div>  
 <Progress value={progress} className="h-2" />  
 <div className="flex items-center justify-between">  
 <Badge variant="outline" className="text-xs">  
 {prize.available} left  
 </Badge>  
 <Button   
 size="sm"   
 disabled={!canAfford}  
 aria-label={`Enter raffle for ${prize.name}`}  
 >  
 {canAfford ? "Enter Raffle" : `Need ${prize.ticketsRequired - userTickets} more`}  
 </Button>  
 </div>  
 </div>  
 </CardContent>  
 </Card>  
 );  
 })}  
 </div>  
 </CardContent>  
 </Card>  
 </>  
 ) : (  
 <Card className="text-center py-12 mb-8">  
 <CardContent>  
 <Users className="w-16 h-16 text-muted-foreground mx-auto mb-4" />  
 <h3 className="text-xl font-semibold mb-2">Login to Access Referrals</h3>  
 <p className="text-muted-foreground mb-6">  
 Create an account to get your unique referral link and start earning rewards!  
 </p>  
 <div className="flex flex-col sm:flex-row gap-3 justify-center">  
 <Link to="/signup">  
 <Button size="lg">Sign Up Now</Button>  
 </Link>  
 <Link to="/login">  
 <Button variant="outline" size="lg">Login</Button>  
 </Link>  
 </div>  
 </CardContent>  
 </Card>  
 )}  
  
 {/\* Referral Leaderboard \*/}  
 <Card>  
 <CardHeader>  
 <CardTitle className="flex items-center gap-2">  
 <Trophy className="w-5 h-5" />  
 Daily Referral Leaderboard  
 </CardTitle>  
 <CardDescription>  
 Top referrers earn bonus prizes and recognition!  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 {loading ? (  
 <div className="space-y-4">  
 {[1, 2, 3, 4, 5].map((i) => (  
 <div key={i} className="flex items-center space-x-4 p-4 rounded-lg bg-muted/50 animate-pulse">  
 <div className="w-10 h-10 bg-muted rounded-full"></div>  
 <div className="flex-1">  
 <div className="h-4 bg-muted rounded w-1/3 mb-2"></div>  
 <div className="h-3 bg-muted rounded w-1/4"></div>  
 </div>  
 <div className="h-8 bg-muted rounded w-16"></div>  
 </div>  
 ))}  
 </div>  
 ) : (  
 <div className="space-y-4">  
 {referralStats.slice(0, 10).map((stat) => (  
 <div  
 key={stat.userId}  
 className={`flex items-center justify-between p-4 rounded-lg border transition-colors ${  
 stat.rank <= 3 ? 'bg-gradient-to-r from-primary/5 to-accent/5' : 'hover:bg-muted/50'  
 } ${user && stat.userName === user.name ? 'ring-2 ring-primary' : ''}`}  
 >  
 <div className="flex items-center space-x-4">  
 <div className="w-10 h-10 bg-gradient-to-br from-green-400 to-blue-500 rounded-full flex items-center justify-center">  
 {getRankIcon(stat.rank)}  
 </div>  
 <div>  
 <h3 className="font-semibold text-foreground flex items-center gap-2">  
 {stat.userName}  
 {user && stat.userName === user.name && (  
 <Badge variant="default" className="text-xs">You</Badge>  
 )}  
 </h3>  
 <p className="text-sm text-muted-foreground">  
 ID: {stat.referralId}  
 </p>  
 </div>  
 </div>  
 <div className="text-right">  
 <div className="text-xl font-bold text-primary">{stat.totalReferrals}</div>  
 <div className="text-sm text-muted-foreground">referrals</div>  
 <div className="flex items-center gap-1 text-sm text-muted-foreground mt-1">  
 <Ticket className="w-3 h-3" />  
 <span>{stat.ticketsEarned}</span>  
 </div>  
 </div>  
 </div>  
 ))}  
 </div>  
 )}  
 </CardContent>  
 </Card>  
 </div>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Signup.tsx  
================================================  
import { useState } from "react";  
import { Button } from "@/components/ui/button";  
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";  
import { Input } from "@/components/ui/input";  
import { Label } from "@/components/ui/label";  
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";  
import { Textarea } from "@/components/ui/textarea";  
import { Checkbox } from "@/components/ui/checkbox";  
import { Badge } from "@/components/ui/badge";  
import { Globe, Upload, X, ArrowRight } from "lucide-react";  
import { Link, useNavigate } from "react-router-dom";  
import { useToast } from "@/hooks/use-toast";  
import { api } from "@/lib/api";  
import { useAuth } from "@/contexts/AuthContext";  
  
interface FormData {  
 name: string;  
 email: string;  
 password: string;  
 confirmPassword: string;  
 nativeLanguages: string[];  
 targetLanguages: string[];  
 experienceLevel: string;  
 profilePicture: File | null;  
 videoIntro: File | null;  
 agreeToTerms: boolean;  
}  
  
const LANGUAGE\_OPTIONS = [  
 "English", "Spanish", "French", "German", "Italian", "Portuguese", "Russian",  
 "Chinese (Mandarin)", "Japanese", "Korean", "Arabic", "Hindi", "Dutch", "Swedish",  
 "Norwegian", "Danish", "Finnish", "Polish", "Czech", "Hungarian", "Turkish",  
 "Greek", "Hebrew", "Thai", "Vietnamese", "Indonesian", "Tagalog", "Swahili"  
];  
  
const EXPERIENCE\_LEVELS = [  
 { value: "beginner", label: "Beginner (A1-A2)" },  
 { value: "intermediate", label: "Intermediate (B1-B2)" },  
 { value: "advanced", label: "Advanced (C1-C2)" },  
 { value: "native", label: "Native Speaker" }  
];  
  
export default function Signup() {  
 const { toast } = useToast();  
 const navigate = useNavigate();  
 const { login } = useAuth();  
 const [formData, setFormData] = useState<FormData>({  
 name: "",  
 email: "",  
 password: "",  
 confirmPassword: "",  
 nativeLanguages: [],  
 targetLanguages: [],  
 experienceLevel: "",  
 profilePicture: null,  
 videoIntro: null,  
 agreeToTerms: false  
 });  
  
 const [isSubmitting, setIsSubmitting] = useState(false);  
  
 const handleLanguageToggle = (language: string, type: 'native' | 'target') => {  
 const key = type === 'native' ? 'nativeLanguages' : 'targetLanguages';  
 const currentLanguages = formData[key];  
   
 if (currentLanguages.includes(language)) {  
 setFormData(prev => ({  
 ...prev,  
 [key]: currentLanguages.filter(lang => lang !== language)  
 }));  
 } else {  
 setFormData(prev => ({  
 ...prev,  
 [key]: [...currentLanguages, language]  
 }));  
 }  
 };  
  
 const handleFileUpload = (file: File, type: 'profilePicture' | 'videoIntro') => {  
 setFormData(prev => ({  
 ...prev,  
 [type]: file  
 }));  
 };  
  
 const handleSubmit = async (e: React.FormEvent) => {  
 e.preventDefault();  
 setIsSubmitting(true);  
  
 try {  
 // Validation  
 if (!formData.name || !formData.email || !formData.password) {  
 throw new Error("Please fill in all required fields");  
 }  
  
 if (formData.password !== formData.confirmPassword) {  
 throw new Error("Passwords do not match");  
 }  
  
 if (formData.nativeLanguages.length === 0) {  
 throw new Error("Please select at least one native language");  
 }  
  
 if (formData.targetLanguages.length === 0) {  
 throw new Error("Please select at least one target language");  
 }  
  
 if (!formData.experienceLevel) {  
 throw new Error("Please select your experience level");  
 }  
  
 if (!formData.agreeToTerms) {  
 throw new Error("Please agree to the terms of service");  
 }  
  
 // Submit to API  
 const response = await api.createUserProfile({  
 name: formData.name,  
 email: formData.email,  
 nativeLanguages: formData.nativeLanguages,  
 targetLanguages: formData.targetLanguages,  
 experienceLevel: formData.experienceLevel,  
 profilePicture: formData.profilePicture?.name,  
 videoIntro: formData.videoIntro?.name,  
 });  
  
 // Create user object and login  
 const userData = {  
 id: response.data?.id || Date.now().toString(),  
 name: formData.name,  
 email: formData.email,  
 nativeLanguages: formData.nativeLanguages,  
 targetLanguages: formData.targetLanguages,  
 experienceLevel: formData.experienceLevel,  
 profilePicture: formData.profilePicture?.name,  
 videoIntro: formData.videoIntro?.name,  
 referralId: response.data?.referralId || `LK-${Date.now().toString(36).toUpperCase()}`,  
 joinedAt: new Date(),  
 };  
  
 login(userData);  
  
 // Show success toast  
 toast({  
 title: "✅ Submission successful!",  
 description: "Welcome to LanguageKonnect! Your profile has been created.",  
 });  
  
 // Reset form  
 setFormData({  
 name: "",  
 email: "",  
 password: "",  
 confirmPassword: "",  
 nativeLanguages: [],  
 targetLanguages: [],  
 experienceLevel: "",  
 profilePicture: null,  
 videoIntro: null,  
 agreeToTerms: false  
 });  
  
 // Redirect to leaderboard after short delay  
 setTimeout(() => {  
 navigate("/leaderboard");  
 }, 1500);  
  
 } catch (error) {  
 toast({  
 title: "Error",  
 description: error instanceof Error ? error.message : "Something went wrong",  
 variant: "destructive"  
 });  
 } finally {  
 setIsSubmitting(false);  
 }  
 };  
  
 return (  
 <div className="min-h-screen bg-gradient-to-br from-background via-blue-50/30 to-purple-50/30">  
 {/\* Navigation \*/}  
 <nav className="border-b bg-background/80 backdrop-blur-sm sticky top-0 z-50">  
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">  
 <div className="flex justify-between items-center h-16">  
 <Link to="/" className="flex items-center space-x-3">  
 <div className="w-8 h-8 bg-gradient-to-br from-primary to-accent rounded-lg flex items-center justify-center">  
 <Globe className="w-5 h-5 text-white" />  
 </div>  
 <span className="text-xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transparent">  
 LanguageKonnect  
 </span>  
 </Link>  
 <div className="flex items-center space-x-4">  
 <Link to="/login">  
 <Button variant="ghost" aria-label="Login to LanguageKonnect">Login</Button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 </nav>  
  
 {/\* Main Content \*/}  
 <main className="py-12 px-4 sm:px-6 lg:px-8">  
 <div className="max-w-2xl mx-auto">  
 <div className="text-center mb-8">  
 <h1 className="text-3xl font-bold text-foreground mb-2">  
 Join LanguageKonnect  
 </h1>  
 <p className="text-muted-foreground">  
 Create your profile to start connecting with language learners worldwide  
 </p>  
 </div>  
  
 <Card className="shadow-lg">  
 <CardHeader>  
 <CardTitle>Create Your Profile</CardTitle>  
 <CardDescription>  
 Fill out your information to get started on your language learning journey  
 </CardDescription>  
 </CardHeader>  
 <CardContent>  
 <form onSubmit={handleSubmit} className="space-y-6">  
 {/\* Basic Information \*/}  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-4">  
 <div className="space-y-2">  
 <Label htmlFor="name">Full Name \*</Label>  
 <Input  
 id="name"  
 type="text"  
 value={formData.name}  
 onChange={(e) => setFormData(prev => ({ ...prev, name: e.target.value }))}  
 aria-label="Enter your full name"  
 required  
 />  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="email">Email Address \*</Label>  
 <Input  
 id="email"  
 type="email"  
 value={formData.email}  
 onChange={(e) => setFormData(prev => ({ ...prev, email: e.target.value }))}  
 aria-label="Enter your email address"  
 required  
 />  
 </div>  
 </div>  
  
 {/\* Password \*/}  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-4">  
 <div className="space-y-2">  
 <Label htmlFor="password">Password \*</Label>  
 <Input  
 id="password"  
 type="password"  
 value={formData.password}  
 onChange={(e) => setFormData(prev => ({ ...prev, password: e.target.value }))}  
 aria-label="Enter your password"  
 required  
 />  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="confirmPassword">Confirm Password \*</Label>  
 <Input  
 id="confirmPassword"  
 type="password"  
 value={formData.confirmPassword}  
 onChange={(e) => setFormData(prev => ({ ...prev, confirmPassword: e.target.value }))}  
 aria-label="Confirm your password"  
 required  
 />  
 </div>  
 </div>  
  
 {/\* Native Languages \*/}  
 <div className="space-y-3">  
 <Label>Native Language(s) \*</Label>  
 <div className="flex flex-wrap gap-2 max-h-32 overflow-y-auto p-3 border rounded-md">  
 {LANGUAGE\_OPTIONS.map((language) => (  
 <Badge  
 key={language}  
 variant={formData.nativeLanguages.includes(language) ? "default" : "outline"}  
 className="cursor-pointer"  
 onClick={() => handleLanguageToggle(language, 'native')}  
 >  
 {language}  
 {formData.nativeLanguages.includes(language) && (  
 <X className="w-3 h-3 ml-1" />  
 )}  
 </Badge>  
 ))}  
 </div>  
 {formData.nativeLanguages.length > 0 && (  
 <p className="text-sm text-muted-foreground">  
 Selected: {formData.nativeLanguages.join(", ")}  
 </p>  
 )}  
 </div>  
  
 {/\* Target Languages \*/}  
 <div className="space-y-3">  
 <Label>Target Language(s) \*</Label>  
 <div className="flex flex-wrap gap-2 max-h-32 overflow-y-auto p-3 border rounded-md">  
 {LANGUAGE\_OPTIONS.map((language) => (  
 <Badge  
 key={language}  
 variant={formData.targetLanguages.includes(language) ? "default" : "outline"}  
 className="cursor-pointer"  
 onClick={() => handleLanguageToggle(language, 'target')}  
 >  
 {language}  
 {formData.targetLanguages.includes(language) && (  
 <X className="w-3 h-3 ml-1" />  
 )}  
 </Badge>  
 ))}  
 </div>  
 {formData.targetLanguages.length > 0 && (  
 <p className="text-sm text-muted-foreground">  
 Selected: {formData.targetLanguages.join(", ")}  
 </p>  
 )}  
 </div>  
  
 {/\* Experience Level \*/}  
 <div className="space-y-2">  
 <Label htmlFor="experience">Experience Level \*</Label>  
 <Select   
 value={formData.experienceLevel}   
 onValueChange={(value) => setFormData(prev => ({ ...prev, experienceLevel: value }))}  
 >  
 <SelectTrigger aria-label="Select your experience level">  
 <SelectValue placeholder="Select your experience level" />  
 </SelectTrigger>  
 <SelectContent>  
 {EXPERIENCE\_LEVELS.map((level) => (  
 <SelectItem key={level.value} value={level.value}>  
 {level.label}  
 </SelectItem>  
 ))}  
 </SelectContent>  
 </Select>  
 </div>  
  
 {/\* File Uploads \*/}  
 <div className="grid grid-cols-1 md:grid-cols-2 gap-4">  
 <div className="space-y-2">  
 <Label htmlFor="profilePicture">Profile Picture</Label>  
 <div className="border-2 border-dashed border-muted-foreground/25 rounded-lg p-4 text-center">  
 <Upload className="w-8 h-8 text-muted-foreground mx-auto mb-2" />  
 <Input  
 id="profilePicture"  
 type="file"  
 accept="image/\*"  
 onChange={(e) => e.target.files?.[0] && handleFileUpload(e.target.files[0], 'profilePicture')}  
 className="hidden"  
 aria-label="Upload profile picture"  
 />  
 <Label htmlFor="profilePicture" className="cursor-pointer text-sm text-muted-foreground">  
 {formData.profilePicture ? formData.profilePicture.name : "Click to upload image"}  
 </Label>  
 </div>  
 </div>  
 <div className="space-y-2">  
 <Label htmlFor="videoIntro">Video Introduction</Label>  
 <div className="border-2 border-dashed border-muted-foreground/25 rounded-lg p-4 text-center">  
 <Upload className="w-8 h-8 text-muted-foreground mx-auto mb-2" />  
 <Input  
 id="videoIntro"  
 type="file"  
 accept="video/\*"  
 onChange={(e) => e.target.files?.[0] && handleFileUpload(e.target.files[0], 'videoIntro')}  
 className="hidden"  
 aria-label="Upload video introduction"  
 />  
 <Label htmlFor="videoIntro" className="cursor-pointer text-sm text-muted-foreground">  
 {formData.videoIntro ? formData.videoIntro.name : "Click to upload video"}  
 </Label>  
 </div>  
 </div>  
 </div>  
  
 {/\* Terms Checkbox \*/}  
 <div className="flex items-center space-x-2">  
 <Checkbox  
 id="terms"  
 checked={formData.agreeToTerms}  
 onCheckedChange={(checked) => setFormData(prev => ({ ...prev, agreeToTerms: checked as boolean }))}  
 aria-label="Agree to terms of service"  
 />  
 <Label htmlFor="terms" className="text-sm">  
 I agree to the{" "}  
 <Link to="/terms" className="text-primary hover:underline">  
 Terms of Service  
 </Link>{" "}  
 and{" "}  
 <Link to="/privacy" className="text-primary hover:underline">  
 Privacy Policy  
 </Link>  
 </Label>  
 </div>  
  
 {/\* Submit Button \*/}  
 <Button  
 type="submit"  
 className="w-full"  
 disabled={isSubmitting}  
 aria-label="Create your LanguageKonnect account"  
 >  
 {isSubmitting ? "Creating Account..." : "Create Account"}  
 {!isSubmitting && <ArrowRight className="ml-2 h-4 w-4" />}  
 </Button>  
 </form>  
  
 <div className="mt-6 text-center">  
 <p className="text-sm text-muted-foreground">  
 Already have an account?{" "}  
 <Link to="/login" className="text-primary hover:underline">  
 Sign in  
 </Link>  
 </p>  
 </div>  
 </CardContent>  
 </Card>  
 </div>  
 </main>  
 </div>  
 );  
}  
  
  
  
================================================  
FILE: client/pages/Terms.tsx  
================================================  
import PlaceholderPage from "./PlaceholderPage";  
import { FileText } from "lucide-react";  
  
export default function Terms() {  
 return (  
 <PlaceholderPage  
 title="Terms of Service"  
 description="Read our terms of service and user agreement for using the LanguageKonnect platform and participating in contests."  
 icon={<FileText className="w-8 h-8 text-primary" />}  
 />  
 );  
}  
  
  
  
================================================  
FILE: netlify/functions/api.ts  
================================================  
import serverless from "serverless-http";  
  
import { createServer } from "../../server";  
  
export const handler = serverless(createServer());  
  
  
  
================================================  
FILE: public/robots.txt  
================================================  
User-agent: Googlebot  
Allow: /  
  
User-agent: Bingbot  
Allow: /  
  
User-agent: Twitterbot  
Allow: /  
  
User-agent: facebookexternalhit  
Allow: /  
  
User-agent: \*  
Allow: /  
  
  
  
================================================  
FILE: server/index.ts  
================================================  
import "dotenv/config";  
import express from "express";  
import cors from "cors";  
import { handleDemo } from "./routes/demo";  
import { connectDB } from "./db/connection";  
import { handleUserProfile } from "./routes/user-profile";  
import { handleContestEntry, handleContestLeaderboard, uploadMiddleware } from "./routes/contest";  
import { handleRaffleEntry, handleReferralLeaderboard } from "./routes/raffle";  
import { handleLogin, handleGetProfile } from "./routes/auth";  
import { handleVote, handleGetVotes } from "./routes/voting";  
  
export function createServer() {  
 const app = express();  
  
 // Connect to MongoDB (non-blocking)  
 connectDB().catch(err => {  
 console.log('MongoDB connection failed, continuing with mock data');  
 });  
  
 // Middleware  
 app.use(cors());  
 app.use(express.json());  
 app.use(express.urlencoded({ extended: true }));  
  
 // Example API routes  
 app.get("/api/ping", (\_req, res) => {  
 const ping = process.env.PING\_MESSAGE ?? "ping";  
 res.json({ message: ping });  
 });  
  
 app.get("/api/demo", handleDemo);  
  
 // LanguageKonnect API routes  
 app.post("/api/auth/login", handleLogin);  
 app.get("/api/auth/profile/:userId", handleGetProfile);  
 app.post("/api/user/profile", handleUserProfile);  
 app.post("/api/contest/entry", uploadMiddleware, handleContestEntry);  
 app.get("/api/contest/leaderboard", handleContestLeaderboard);  
 app.post("/api/contest/vote", handleVote);  
 app.get("/api/contest/votes/:entryId", handleGetVotes);  
 app.post("/api/raffle-entry", handleRaffleEntry);  
 app.get("/api/referral/leaderboard", handleReferralLeaderboard);  
  
 return app;  
}  
  
  
  
================================================  
FILE: server/node-build.ts  
================================================  
import path from "path";  
import { createServer } from "./index";  
import \* as express from "express";  
  
const app = createServer();  
const port = process.env.PORT || 3000;  
  
// In production, serve the built SPA files  
const \_\_dirname = import.meta.dirname;  
const distPath = path.join(\_\_dirname, "../spa");  
  
// Serve static files  
app.use(express.static(distPath));  
  
// Handle React Router - serve index.html for all non-API routes  
app.get("\*", (req, res) => {  
 // Don't serve index.html for API routes  
 if (req.path.startsWith("/api/") || req.path.startsWith("/health")) {  
 return res.status(404).json({ error: "API endpoint not found" });  
 }  
  
 res.sendFile(path.join(distPath, "index.html"));  
});  
  
app.listen(port, () => {  
 console.log(`🚀 Fusion Starter server running on port ${port}`);  
 console.log(`📱 Frontend: http://localhost:${port}`);  
 console.log(`🔧 API: http://localhost:${port}/api`);  
});  
  
// Graceful shutdown  
process.on("SIGTERM", () => {  
 console.log("🛑 Received SIGTERM, shutting down gracefully");  
 process.exit(0);  
});  
  
process.on("SIGINT", () => {  
 console.log("🛑 Received SIGINT, shutting down gracefully");  
 process.exit(0);  
});  
  
  
  
================================================  
FILE: server/db/connection.ts  
================================================  
import mongoose from 'mongoose';  
  
const MONGODB\_URI = 'mongodb+srv://akashmaji200680:4z2w5e5Hc4QTE1Wi@cluster01.xgihl1k.mongodb.net/languagekonnect';  
  
let isConnected = false;  
  
export const connectDB = async () => {  
 try {  
 const conn = await mongoose.connect(MONGODB\_URI);  
 console.log(`MongoDB Connected: ${conn.connection.host}`);  
 isConnected = true;  
 return conn;  
 } catch (error) {  
 console.error('MongoDB connection error:', error);  
 console.log('Server will continue without MongoDB. API endpoints will return mock data.');  
 isConnected = false;  
 return null;  
 }  
};  
  
export const isDBConnected = () => isConnected;  
  
export default mongoose;  
  
  
  
================================================  
FILE: server/db/models.ts  
================================================  
import mongoose, { Document, Schema } from 'mongoose';  
  
// User Profile Schema  
export interface IUserProfile extends Document {  
 name: string;  
 email: string;  
 password?: string;  
 nativeLanguages: string[];  
 targetLanguages: string[];  
 experienceLevel: string;  
 profilePicture?: string;  
 videoIntro?: string;  
 referralId: string;  
 createdAt: Date;  
 updatedAt: Date;  
}  
  
const UserProfileSchema = new Schema<IUserProfile>({  
 name: { type: String, required: true },  
 email: { type: String, required: true, unique: true },  
 password: { type: String },  
 nativeLanguages: [{ type: String, required: true }],  
 targetLanguages: [{ type: String, required: true }],  
 experienceLevel: {   
 type: String,   
 required: true,  
 enum: ['beginner', 'intermediate', 'advanced', 'native']  
 },  
 profilePicture: { type: String },  
 videoIntro: { type: String },  
 referralId: { type: String, unique: true },  
 createdAt: { type: Date, default: Date.now },  
 updatedAt: { type: Date, default: Date.now }  
});  
  
// Contest Entry Schema  
export interface IContestEntry extends Document {  
 userId: mongoose.Types.ObjectId;  
 language: string;  
 region: string;  
 caption: string;  
 videoUrl: string;  
 votes: number;  
 rating: number;  
 createdAt: Date;  
}  
  
const ContestEntrySchema = new Schema<IContestEntry>({  
 userId: { type: Schema.Types.ObjectId, ref: 'UserProfile', required: true },  
 language: { type: String, required: true },  
 region: { type: String, required: true },  
 caption: { type: String, required: true },  
 videoUrl: { type: String, required: true },  
 votes: { type: Number, default: 0 },  
 rating: { type: Number, default: 0 },  
 createdAt: { type: Date, default: Date.now }  
});  
  
// Raffle Entry Schema  
export interface IRaffleEntry extends Document {  
 userId: mongoose.Types.ObjectId;  
 referralId: string;  
 referredUserId: mongoose.Types.ObjectId;  
 ticketsEarned: number;  
 createdAt: Date;  
}  
  
const RaffleEntrySchema = new Schema<IRaffleEntry>({  
 userId: { type: Schema.Types.ObjectId, ref: 'UserProfile', required: true },  
 referralId: { type: String, required: true },  
 referredUserId: { type: Schema.Types.ObjectId, ref: 'UserProfile', required: true },  
 ticketsEarned: { type: Number, default: 1 },  
 createdAt: { type: Date, default: Date.now }  
});  
  
// Export models  
export const UserProfile = mongoose.model<IUserProfile>('UserProfile', UserProfileSchema);  
export const ContestEntry = mongoose.model<IContestEntry>('ContestEntry', ContestEntrySchema);  
export const RaffleEntry = mongoose.model<IRaffleEntry>('RaffleEntry', RaffleEntrySchema);  
  
  
  
================================================  
FILE: server/routes/auth.ts  
================================================  
import { RequestHandler } from "express";  
import { UserProfile } from "../db/models";  
import { isDBConnected } from "../db/connection";  
  
export const handleLogin: RequestHandler = async (req, res) => {  
 try {  
 const { email, password } = req.body;  
  
 // Validation  
 if (!email || !password) {  
 return res.status(400).json({   
 success: false,   
 message: "Missing required fields: email, password"   
 });  
 }  
  
 if (!isDBConnected()) {  
 // Mock login when database is not available  
 const mockUser = {  
 id: "mock-user-" + Date.now(),  
 name: "Demo User",  
 email: email,  
 nativeLanguages: ["English"],  
 targetLanguages: ["Spanish", "French"],  
 experienceLevel: "intermediate",  
 referralId: `LK-${Date.now().toString(36).toUpperCase()}`,  
 joinedAt: new Date(),  
 };  
  
 console.log('Using mock data - user login:', {   
 email: mockUser.email,  
 name: mockUser.name   
 });  
  
 return res.status(200).json({   
 success: true,   
 message: "Login successful (demo mode)",  
 user: mockUser  
 });  
 }  
  
 // Find user by email  
 const user = await UserProfile.findOne({ email });  
 if (!user) {  
 return res.status(401).json({   
 success: false,   
 message: "Invalid email or password"   
 });  
 }  
  
 // In a real app, you would verify the password hash here  
 // For now, we'll accept any password for demo purposes  
  
 const userData = {  
 id: user.\_id.toString(),  
 name: user.name,  
 email: user.email,  
 nativeLanguages: user.nativeLanguages,  
 targetLanguages: user.targetLanguages,  
 experienceLevel: user.experienceLevel,  
 profilePicture: user.profilePicture,  
 videoIntro: user.videoIntro,  
 referralId: user.referralId,  
 joinedAt: user.createdAt,  
 };  
  
 console.log('User login successful:', {   
 id: userData.id,   
 email: userData.email,  
 name: userData.name   
 });  
  
 res.status(200).json({   
 success: true,   
 message: "Login successful",  
 user: userData  
 });  
  
 } catch (error) {  
 console.error('Error during login:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
export const handleGetProfile: RequestHandler = async (req, res) => {  
 try {  
 const { userId } = req.params;  
  
 if (!isDBConnected()) {  
 return res.status(200).json({  
 success: true,  
 user: null  
 });  
 }  
  
 const user = await UserProfile.findById(userId);  
 if (!user) {  
 return res.status(404).json({  
 success: false,  
 message: "User not found"  
 });  
 }  
  
 const userData = {  
 id: user.\_id.toString(),  
 name: user.name,  
 email: user.email,  
 nativeLanguages: user.nativeLanguages,  
 targetLanguages: user.targetLanguages,  
 experienceLevel: user.experienceLevel,  
 profilePicture: user.profilePicture,  
 videoIntro: user.videoIntro,  
 referralId: user.referralId,  
 joinedAt: user.createdAt,  
 };  
  
 res.status(200).json({  
 success: true,  
 user: userData  
 });  
  
 } catch (error) {  
 console.error('Error fetching user profile:', error);  
 res.status(500).json({  
 success: false,  
 message: "Internal server error"  
 });  
 }  
};  
  
  
  
================================================  
FILE: server/routes/contest.ts  
================================================  
import { RequestHandler } from "express";  
import { ContestEntry, UserProfile } from "../db/models";  
import { isDBConnected } from "../db/connection";  
import multer from "multer";  
  
// Configure multer for file uploads  
const upload = multer({  
 limits: {  
 fileSize: 50 \* 1024 \* 1024, // 50MB limit  
 },  
 fileFilter: (req, file, cb) => {  
 if (file.mimetype === 'video/mp4') {  
 cb(null, true);  
 } else {  
 cb(new Error('Only MP4 files are allowed'));  
 }  
 }  
});  
  
export const uploadMiddleware = upload.single('video');  
  
export const handleContestEntry: RequestHandler = async (req, res) => {  
 try {  
 const { language, region, caption, userId } = req.body;  
 const videoFile = req.file;  
  
 // Validation  
 if (!language || !region || !caption) {  
 return res.status(400).json({  
 success: false,  
 message: "Missing required fields: language, region, caption"  
 });  
 }  
  
 // Check if video file was uploaded  
 if (!videoFile) {  
 return res.status(400).json({  
 success: false,  
 message: "Video file is required"  
 });  
 }  
  
 // For now, we'll simulate video storage and use a placeholder URL  
 const videoUrl = `https://placeholder.example.com/videos/${Date.now()}.mp4`;  
  
 console.log('Video file received:', {  
 filename: videoFile.originalname,  
 size: videoFile.size,  
 mimetype: videoFile.mimetype  
 });  
  
 if (!isDBConnected()) {  
 // Mock response when database is not available  
 const mockEntryId = "mock-entry-" + Date.now();  
 console.log('Using mock data - contest entry created:', {  
 id: mockEntryId,  
 language,  
 region,  
 videoFile: videoFile ? videoFile.originalname : 'No file'  
 });  
  
 return res.status(201).json({  
 success: true,  
 entryId: mockEntryId,  
 message: "Video submitted! (demo mode)",  
 videoUrl: videoUrl // Return the video URL for client use  
 });  
 }  
  
 // Create contest entry  
 const contestEntry = new ContestEntry({  
 userId: userId || req.body.userId || '507f1f77bcf86cd799439011', // Use provided user ID  
 language,  
 region,  
 caption,  
 videoUrl,  
 votes: 0, // Always start with 0 votes  
 rating: 0, // Always start with 0 rating  
 createdAt: new Date()  
 });  
  
 await contestEntry.save();  
  
 console.log('Contest entry created successfully:', {  
 id: contestEntry.\_id,  
 language: contestEntry.language,  
 region: contestEntry.region  
 });  
  
 res.status(201).json({  
 success: true,  
 entryId: contestEntry.\_id,  
 message: "Video submitted!"  
 });  
  
 } catch (error) {  
 console.error('Error creating contest entry:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
export const handleContestLeaderboard: RequestHandler = async (req, res) => {  
 try {  
 const { sortBy = 'votes', language, region } = req.query;  
  
 if (!isDBConnected()) {  
 // Mock leaderboard data when database is not available  
 const mockLeaderboard = [  
 {  
 id: "mock-1",  
 userName: "Sarah Chen",  
 score: 125,  
 rank: 1,  
 language: "Spanish",  
 region: "North America",  
 caption: "Practicing my Spanish pronunciation!",  
 rating: 4.8,  
 createdAt: new Date()  
 },  
 {  
 id: "mock-2",  
 userName: "Miguel Rodriguez",  
 score: 98,  
 rank: 2,  
 language: "Japanese",  
 region: "South America",  
 caption: "Learning Japanese through anime!",  
 rating: 4.6,  
 createdAt: new Date()  
 },  
 {  
 id: "mock-3",  
 userName: "Emma Thompson",  
 score: 87,  
 rank: 3,  
 language: "French",  
 region: "Europe",  
 caption: "Bonjour from Paris!",  
 rating: 4.5,  
 createdAt: new Date()  
 }  
 ];  
  
 console.log('Using mock leaderboard data');  
 return res.json(mockLeaderboard);  
 }  
  
 // Build filter query  
 const filter: any = {};  
 if (language) filter.language = language;  
 if (region) filter.region = region;  
  
 // Build sort query  
 let sortQuery: any = {};  
 switch (sortBy) {  
 case 'recent':  
 sortQuery = { createdAt: -1 };  
 break;  
 case 'rating':  
 sortQuery = { rating: -1, votes: -1 };  
 break;  
 case 'votes':  
 default:  
 sortQuery = { votes: -1, createdAt: -1 };  
 break;  
 }  
  
 // Get contest entries with user data  
 const entries = await ContestEntry.find(filter)  
 .populate('userId', 'name email')  
 .sort(sortQuery)  
 .limit(50)  
 .lean();  
  
 // Transform data for leaderboard format  
 const leaderboard = entries.map((entry, index) => ({  
 id: entry.\_id,  
 userName: (entry.userId as any)?.name || 'Anonymous',  
 score: entry.votes,  
 rank: index + 1,  
 language: entry.language,  
 region: entry.region,  
 caption: entry.caption,  
 rating: entry.rating,  
 createdAt: entry.createdAt  
 }));  
  
 console.log('Contest leaderboard fetched:', {  
 count: leaderboard.length,  
 sortBy,  
 filters: filter  
 });  
  
 res.json(leaderboard);  
  
 } catch (error) {  
 console.error('Error fetching contest leaderboard:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
  
  
================================================  
FILE: server/routes/demo.ts  
================================================  
import { RequestHandler } from "express";  
import { DemoResponse } from "@shared/api";  
  
export const handleDemo: RequestHandler = (req, res) => {  
 const response: DemoResponse = {  
 message: "Hello from Express server",  
 };  
 res.status(200).json(response);  
};  
  
  
  
================================================  
FILE: server/routes/raffle.ts  
================================================  
import { RequestHandler } from "express";  
import { RaffleEntry, UserProfile } from "../db/models";  
import { isDBConnected } from "../db/connection";  
  
export const handleRaffleEntry: RequestHandler = async (req, res) => {  
 try {  
 const { referralId, userId } = req.body;  
  
 // Validation  
 if (!referralId) {  
 return res.status(400).json({  
 success: false,  
 message: "Missing required field: referralId"  
 });  
 }  
  
 if (!isDBConnected()) {  
 // Mock response when database is not available  
 console.log('Using mock data - raffle entry created:', {  
 referralId,  
 userId  
 });  
  
 return res.status(201).json({  
 success: true,  
 ticketsEarned: 1,  
 message: "🎉 You earned a raffle ticket! (demo mode)"  
 });  
 }  
  
 // Find the user who owns this referral ID  
 const referringUser = await UserProfile.findOne({ referralId });  
 if (!referringUser) {  
 return res.status(404).json({  
 success: false,  
 message: "Invalid referral ID"  
 });  
 }  
  
 // For this example, we'll assume the referred user is created separately  
 // In a real implementation, this would be called when a new user signs up with a referral  
 const referredUserId = userId || '507f1f77bcf86cd799439012'; // Placeholder  
  
 // Check if this referral already exists  
 const existingEntry = await RaffleEntry.findOne({  
 userId: referringUser.\_id,  
 referredUserId: referredUserId  
 });  
  
 if (existingEntry) {  
 return res.status(409).json({  
 success: false,  
 message: "Referral already recorded"  
 });  
 }  
  
 // Create raffle entry  
 const raffleEntry = new RaffleEntry({  
 userId: referringUser.\_id,  
 referralId,  
 referredUserId: referredUserId,  
 ticketsEarned: 1,  
 createdAt: new Date()  
 });  
  
 await raffleEntry.save();  
  
 console.log('Raffle entry created successfully:', {  
 id: raffleEntry.\_id,  
 referringUserId: referringUser.\_id,  
 referredUserId: referredUserId  
 });  
  
 res.status(201).json({  
 success: true,  
 ticketsEarned: 1,  
 message: "🎉 You earned a raffle ticket!"  
 });  
  
 } catch (error) {  
 console.error('Error creating raffle entry:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
export const handleReferralLeaderboard: RequestHandler = async (req, res) => {  
 try {  
 if (!isDBConnected()) {  
 // Mock referral leaderboard when database is not available  
 const mockReferralLeaderboard = [  
 {  
 userId: "mock-user-1",  
 userName: "Alex Johnson",  
 referralId: "LK-ABC123",  
 totalReferrals: 15,  
 ticketsEarned: 15,  
 rank: 1  
 },  
 {  
 userId: "mock-user-2",  
 userName: "Maria Garcia",  
 referralId: "LK-DEF456",  
 totalReferrals: 12,  
 ticketsEarned: 12,  
 rank: 2  
 },  
 {  
 userId: "mock-user-3",  
 userName: "James Wilson",  
 referralId: "LK-GHI789",  
 totalReferrals: 8,  
 ticketsEarned: 8,  
 rank: 3  
 }  
 ];  
  
 console.log('Using mock referral leaderboard data');  
 return res.json(mockReferralLeaderboard);  
 }  
  
 // Aggregate referral stats  
 const leaderboard = await RaffleEntry.aggregate([  
 {  
 $group: {  
 \_id: '$userId',  
 totalReferrals: { $sum: 1 },  
 ticketsEarned: { $sum: '$ticketsEarned' }  
 }  
 },  
 {  
 $lookup: {  
 from: 'userprofiles',  
 localField: '\_id',  
 foreignField: '\_id',  
 as: 'user'  
 }  
 },  
 {  
 $unwind: '$user'  
 },  
 {  
 $project: {  
 userId: '$\_id',  
 userName: '$user.name',  
 referralId: '$user.referralId',  
 totalReferrals: 1,  
 ticketsEarned: 1  
 }  
 },  
 {  
 $sort: { totalReferrals: -1, ticketsEarned: -1 }  
 },  
 {  
 $limit: 50  
 }  
 ]);  
  
 // Add rank to each entry  
 const rankedLeaderboard = leaderboard.map((entry, index) => ({  
 ...entry,  
 rank: index + 1  
 }));  
  
 console.log('Referral leaderboard fetched:', {  
 count: rankedLeaderboard.length  
 });  
  
 res.json(rankedLeaderboard);  
  
 } catch (error) {  
 console.error('Error fetching referral leaderboard:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
  
  
================================================  
FILE: server/routes/user-profile.ts  
================================================  
import { RequestHandler } from "express";  
import { UserProfile } from "../db/models";  
import { generateReferralId } from "../utils/helpers";  
import { isDBConnected } from "../db/connection";  
  
export const handleUserProfile: RequestHandler = async (req, res) => {  
 try {  
 const { name, email, nativeLanguages, targetLanguages, experienceLevel, profilePicture, videoIntro } = req.body;  
  
 // Validation  
 if (!name || !email || !nativeLanguages || !targetLanguages || !experienceLevel) {  
 return res.status(400).json({  
 success: false,  
 message: "Missing required fields: name, email, nativeLanguages, targetLanguages, experienceLevel"  
 });  
 }  
  
 // Generate unique referral ID  
 const referralId = generateReferralId();  
  
 if (!isDBConnected()) {  
 // Mock response when database is not available  
 console.log('Using mock data - user profile created:', {  
 name,  
 email,  
 referralId  
 });  
  
 return res.status(201).json({  
 success: true,  
 message: "User profile created successfully (demo mode)",  
 data: {  
 id: "mock-" + Date.now(),  
 referralId: referralId  
 }  
 });  
 }  
  
 // Check if user already exists  
 const existingUser = await UserProfile.findOne({ email });  
 if (existingUser) {  
 return res.status(409).json({  
 success: false,  
 message: "User with this email already exists"  
 });  
 }  
  
 // Create new user profile  
 const userProfile = new UserProfile({  
 name,  
 email,  
 nativeLanguages,  
 targetLanguages,  
 experienceLevel,  
 profilePicture,  
 videoIntro,  
 referralId,  
 createdAt: new Date(),  
 updatedAt: new Date()  
 });  
  
 await userProfile.save();  
  
 console.log('User profile created successfully:', {  
 id: userProfile.\_id,  
 email: userProfile.email,  
 referralId: userProfile.referralId  
 });  
  
 res.status(201).json({  
 success: true,  
 message: "User profile created successfully",  
 data: {  
 id: userProfile.\_id,  
 referralId: userProfile.referralId  
 }  
 });  
  
 } catch (error) {  
 console.error('Error creating user profile:', error);  
 res.status(500).json({   
 success: false,   
 message: "Internal server error"   
 });  
 }  
};  
  
  
  
================================================  
FILE: server/routes/voting.ts  
================================================  
import { RequestHandler } from "express";  
import { ContestEntry } from "../db/models";  
import { isDBConnected } from "../db/connection";  
  
// In-memory vote storage for demo (in production, use database)  
const voteStorage = new Map<string, Set<string>>(); // entryId -> Set of userIds who voted  
  
export const handleVote: RequestHandler = async (req, res) => {  
 try {  
 const { entryId, userId, action } = req.body; // action: 'like' or 'unlike'  
  
 if (!entryId || !userId || !action) {  
 return res.status(400).json({  
 success: false,  
 message: "Missing required fields: entryId, userId, action"  
 });  
 }  
  
 if (!isDBConnected()) {  
 // Mock voting system when database is not available  
 if (!voteStorage.has(entryId)) {  
 voteStorage.set(entryId, new Set());  
 }  
  
 const voters = voteStorage.get(entryId)!;  
 let newVoteCount = voters.size;  
  
 if (action === 'like' && !voters.has(userId)) {  
 voters.add(userId);  
 newVoteCount = voters.size;  
 } else if (action === 'unlike' && voters.has(userId)) {  
 voters.delete(userId);  
 newVoteCount = voters.size;  
 }  
  
 console.log('Vote processed (mock):', {  
 entryId,  
 userId,  
 action,  
 newVoteCount  
 });  
  
 return res.status(200).json({  
 success: true,  
 message: "Vote recorded!",  
 votes: newVoteCount,  
 isLiked: voters.has(userId)  
 });  
 }  
  
 // Database implementation  
 const entry = await ContestEntry.findById(entryId);  
 if (!entry) {  
 return res.status(404).json({  
 success: false,  
 message: "Contest entry not found"  
 });  
 }  
  
 // In a real implementation, you'd have a separate votes collection  
 // For now, we'll just update the vote count directly  
 if (action === 'like') {  
 entry.votes += 1;  
 } else if (action === 'unlike') {  
 entry.votes = Math.max(0, entry.votes - 1);  
 }  
  
 await entry.save();  
  
 console.log('Vote processed:', {  
 entryId,  
 userId,  
 action,  
 newVoteCount: entry.votes  
 });  
  
 res.status(200).json({  
 success: true,  
 message: "Vote recorded!",  
 votes: entry.votes,  
 isLiked: action === 'like'  
 });  
  
 } catch (error) {  
 console.error('Error processing vote:', error);  
 res.status(500).json({  
 success: false,  
 message: "Internal server error"  
 });  
 }  
};  
  
export const handleGetVotes: RequestHandler = async (req, res) => {  
 try {  
 const { entryId } = req.params;  
  
 if (!isDBConnected()) {  
 // Return mock vote count  
 const voters = voteStorage.get(entryId) || new Set();  
 return res.status(200).json({  
 success: true,  
 votes: voters.size  
 });  
 }  
  
 const entry = await ContestEntry.findById(entryId);  
 if (!entry) {  
 return res.status(404).json({  
 success: false,  
 message: "Contest entry not found"  
 });  
 }  
  
 res.status(200).json({  
 success: true,  
 votes: entry.votes  
 });  
  
 } catch (error) {  
 console.error('Error fetching votes:', error);  
 res.status(500).json({  
 success: false,  
 message: "Internal server error"  
 });  
 }  
};  
  
  
  
================================================  
FILE: server/utils/helpers.ts  
================================================  
// Generate unique referral ID  
export function generateReferralId(): string {  
 const timestamp = Date.now().toString(36);  
 const randomStr = Math.random().toString(36).substr(2, 5);  
 return `LK-${timestamp}-${randomStr}`.toUpperCase();  
}  
  
// Validate email format  
export function isValidEmail(email: string): boolean {  
 const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;  
 return emailRegex.test(email);  
}  
  
// Sanitize user input  
export function sanitizeString(input: string): string {  
 return input.trim().replace(/[<>]/g, '');  
}  
  
  
  
================================================  
FILE: shared/api.ts  
================================================  
/\*\*  
 \* Shared code between client and server  
 \* Useful to share types between client and server  
 \* and/or small pure JS functions that can be used on both client and server  
 \*/  
  
/\*\*  
 \* Example response type for /api/demo  
 \*/  
export interface DemoResponse {  
 message: string;  
}  
  
  
  
================================================  
FILE: .builder/rules/deploy-app.mdc  
================================================  
---  
description: Deploy the app to production using Netlify or Vercel to make the site live  
globs:  
alwaysApply: true  
---  
  
Deploying the app to production is powered by either Netlify or Vercel through MCP (Model Context Protocol) integrations:  
  
\*\*For Netlify:\*\*  
  
- If the Netlify tools are not available, tell the user to [connect the Netlify MCP](#open-mcp-popover).  
- If the user asks to deploy the app, use the appropriate Netlify MCP tools to deploy the app.  
- Note: Netlify builds the source code on their servers, so while it's good to check if `npm run build` works locally, it's not required.  
  
\*\*For Vercel:\*\*  
  
- If the Vercel tools are not available, tell the user to [connect the Vercel MCP](#open-mcp-popover).  
- Vercel deployment should work automatically once connected - simply use the appropriate Vercel MCP tools.  
  
You can also suggest sharing the [Preview link](#open-preview), but explain that it won't be a production-ready link.  
  
  
  
================================================  
FILE: .builder/rules/organize-ui.mdc  
================================================  
---  
description: Keep component and files small  
globs:  
alwaysApply: true  
---  
  
As development goes, break down complex UI into smaller components, avoiding very nested JSX trees.  
Also, even within a single page (route), feel free to create auxiliary extra smaller files to make the page more maintenable.  
Breaking big components into multiple files will help down the road when the user asks for changes or refactors.