■ Frontend File List:

- src\App.css
- src\App.js
- src\index.css
- src\index.js
- src\reportWebVitals.js
- src\setupTests.js
- src\theme.css
- src\ThemeProvider.js
- src\ai_assistant\AIChatAssistant.css
- src\ai_assistant\AIChatAssistant.js
- src\ai_assistant\AIChatAssistantForDiet.css
- src\ai_assistant\AIChatAssistantForDiet.js
- src\ai_assistant\ai_service.js
- src\ai_assistant\index.js
- src\auth\forgotpassword.css
- src\auth\forgotpassword.js
- src\auth\login.css
- src\auth\login.js
- src\auth\Login.test.js
- src\components\GenericAppointments.js
- src\components\ProtectedRoute.js
- src\dashboard\counselor\counselor.css
- src\dashboard\counselor\CounselorCompletedAppointments.js
- src\dashboard\counselor\CounselorDashboard.js
- src\dashboard\counselor\CounselorsAppointments.js
- src\dashboard\counselor\components\counselor_footer.js
- src\dashboard\counselor\components\counselor_navbar.js
- src\dashboard\counselor\components\counselor_sidebar.js
- src\dashboard\counselor\pages\CounselorPasswordRequest.js
- src\dashboard\diet\DietitiansAppointments.js
- src\dashboard\diet\DietMealPlanAssign.js
- src\dashboard\diet\diet_assign.js
- src\dashboard\diet\diet_physioo.js
- src\dashboard\diet\diet_profile.js
- src\dashboard\diet\components\diet_footer.js
- src\dashboard\diet\components\diet_navbar.js
- src\dashboard\diet\components\diet_sidebar.js
- src\dashboard\diet\pages\DietPasswordRequest.js
- src\dashboard\diet__tests__\DietMealPlanAssign.test.js
- src\dashboard\doctor\DoctorAssignmentDashboard.js
- src\dashboard\doctor\DoctorDashboardHome.js
- src\dashboard\doctor\DoctorsAppointments.js
- src\dashboard\doctor\DoctorSlotBooking.css
- src\dashboard\doctor\DoctorSlotBooking.js
- src\dashboard\doctor\components\doctor_footer.js
- src\dashboard\doctor\components\doctor_navbar.js
- src\dashboard\doctor\components\doctor_sidebar.js
- src\dashboard\doctor\components\DocumentPopup.js
- src\dashboard\doctor\components\PdfCardViewer.js
- src\dashboard\doctor\components\popup.css
- src\dashboard\doctor\pages\DoctorPasswordRequest.js
- src\dashboard\doctor__tests__\DoctorDashboardHome.test.js
- src\dashboard\master_admin\ActivityLogs.js
- src\dashboard\master_admin\AdminBlogSection.js
- src\dashboard\master_admin\AdminPasswordRequests.js
- src\dashboard\master_admin\AllReports.js
- src\dashboard\master_admin\AppointmentsContainer.js
- src\dashboard\master_admin\appointments_container.css
- src\dashboard\master_admin\CounselorAppointments.js
- src\dashboard\master_admin\demo.js
- src\dashboard\master_admin\DietitianAppointments.js
- src\dashboard\master_admin\DoctorAppointments.js
- src\dashboard\master_admin\master_admin.css
- src\dashboard\master_admin\master_admin_dashboard.js
- src\dashboard\master_admin\PatientJourney.js
- src\dashboard\master_admin\PhlebotomistAppointments.js
- src\dashboard\master_admin\PhysioAppointments.js
- src\dashboard\master_admin\ReportBuilder.js
- src\dashboard\master_admin\SecurityControls.js
- src\dashboard\master_admin\Services.css
- src\dashboard\master_admin\Services.js

- src\dashboard\master_admin\SystemOverview.js
- src\dashboard\master_admin\TemplateDemo.js
- src\dashboard\master_admin\UserManagement.js
- src\dashboard\master_admin\components\master_admin_footer.js
- src\dashboard\master_admin\components\master_admin_navbar.js
- src\dashboard\master_admin\components\master_admin_sidebar.js
- src\dashboard\master_admin__tests__\AppointmentsContainer.test.js
- src\dashboard\phlebotomist\phlebotomist.css
- src\dashboard\phlebotomist\PhlebotomistAppointments.js
- src\dashboard\phlebotomist\PhlebotomistBookingForm.js
- src\dashboard\phlebotomist\PhlebotomistDashboardHome.js
- src\dashboard\phlebotomist\components\phlebotomist_footer.js
- src\dashboard\phlebotomist\components\phlebotomist_navbar.js
- src\dashboard\phlebotomist\components\phlebotomist_sidebar.js
- src\dashboard\phlebotomist\pages\PhlebotomistPasswordRequest.js
- src\dashboard\physio\assign.css
- src\dashboard\physio\clientManagement.css
- src\dashboard\physio\physio.css
- src\dashboard\physio\PhysioPlanAssign.css
- src\dashboard\physio\PhysioPlanAssign.js
- src\dashboard\physio\PhysiosAppointments.js
- src\dashboard\physio\profile.css
- src\dashboard\physio\profile.js
- src\dashboard\physio\report.css
- src\dashboard\physio\userDetails.css
- src\dashboard\physio\userDetails.js
- src\dashboard\physio\components\Footer.css
- src\dashboard\physio\components\Footer.js
- src\dashboard\physio\components\Navbar.css
- src\dashboard\physio\components\navbar.js
- src\dashboard\physio\components\Sidebar.css
- src\dashboard\physio\components\sidebar.js
- src\dashboard\physio\pages\PhysioPasswordRequest.js
- src\pages\Unauthorized.js

File: src\App.css

```
/* ===== Base Reset ===== */
html,
body {
  margin: 0;
  padding: 0;
  overflow-x: hidden;
  height: 100%;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen,
    Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;
}



* {
  box-sizing: border-box;
}

/* ===== App Root Wrapper ===== */
.app {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  background-color: var(--bg-primary);
  color: var(--text-primary);
}

/* ===== Layout ===== */
.dashboard-layout {
  display: flex;
  flex-direction: row;
  height: calc(100vh - 70px);
  overflow: hidden;
}

.dashboard-main {
  flex-grow: 1;
  overflow-y: auto;
  overflow-x: hidden;
  padding: 2rem;
  background-color: var(--bg-primary);
  width: 100%;
  max-width: 100vw;
  transition: all 0.3s ease;
}

/* ===== Headings ===== */
.dashboard-main h1 {
  font-size: 2rem;
  margin-bottom: 1.5rem;
  position: relative;
}

.dashboard-main h1::after {
  content: "";
  position: absolute;
  bottom: -8px;
  left: 0;
  width: 60px;
  height: 4px;
  background-color: var(--accent);
  border-radius: 2px;
}

/* ===== Paragraphs ===== */
.dashboard-main p {
  line-height: 1.6;
  margin-bottom: 1.5rem;
  color: var(--text-secondary);
}

/* ===== Card Component ===== */
.card {
```

```
  background-color: var(--bg-secondary);
  border-radius: 8px;
  padding: 1.5rem;
  margin-bottom: 1.5rem;
  box-shadow: 0 4px 6px var(--shadow-color);
  max-width: 100%;
  overflow-x: auto;
  word-wrap: break-word;
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}



.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 15px var(--shadow-color);
}

.card-header {
  margin-bottom: 1rem;
  color: var(--accent);
  font-weight: 600;
}

/* ===== Buttons ===== */
.btn {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 0.5rem;
  padding: 0.75rem 1.5rem;
  border-radius: 8px;
  font-weight: 500;
  cursor: pointer;
  border: none;
  transition: all 0.3s ease;
}

.btn-primary {
  background-color: var(--accent);
  color: white;
}

.btn-primary:hover {
  background-color: var(--accent-hover);
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(204, 85, 0, 0.3);
}

.btn-secondary {
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
}

.btn-secondary:hover {
  background-color: var(--bg-hover);
  color: var(--accent);
}

/* ===== Grid System ===== */
.row {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  margin: 0;
}

.col {
  flex: 1 1 calc(50% - 1rem);
  min-width: 280px;
  max-width: 100%;
}
```

```
/*footer*/


/* Force full width on small screens */
@media (max-width: 768px) {
  .col {
    flex: 1 1 100%;
    min-width: 100%;
  }
}

/* ===== Scrollbar ===== */
.dashboard-main::-webkit-scrollbar {
  width: 8px;
}

.dashboard-main::-webkit-scrollbar-thumb {
  background-color: var(--accent);
  border-radius: 4px;
}

/* ===== Responsive Breakpoints ===== */

/* Large Desktop */
@media (max-width: 1200px) {
  .dashboard-main {
    padding: 1.5rem;
  }
}

/* Tablets */
@media (max-width: 992px) {
  .dashboard-main {
    padding: 1.25rem;
  }

  .dashboard-main h1 {
    font-size: 1.75rem;
  }
}

/* Small Tablets & Mobile Landscape */
@media (max-width: 768px) {
  .dashboard-layout {
    flex-direction: column;
  }

  .dashboard-main {
    padding: 1rem;
  }

  .btn {
    width: 100%;
    justify-content: center;
  }
}

/* Phones */
@media (max-width: 480px) {
  .dashboard-main h1 {
    font-size: 1.5rem;
  }

  .card {
    padding: 1.25rem;
    width: 100%;
  }

  .btn {
    font-size: 0.95rem;
    padding: 0.65rem 1.25rem;
  }
```

```
}
```

File: src\App.js

```
import React from "react";
import {
  BrowserRouter,
  Routes,
  Route,
  Navigate,
  Outlet,
} from "react-router-dom";

// Styles
import "./index.css";
import "./App.css";
import "./theme.css";

// External Libraries
import { ToastContainer } from "react-toastify";

// Theme & Providers
import { ThemeProvider, useTheme } from "./ThemeProvider";

// Components
import ProtectedRoute from "./components/ProtectedRoute";

// ================================================
// AUTH PAGES
// ================================================
import Login from "./auth/login";
import Forgot from "./auth/forgotpassword";

// ================================================
// COMMON PAGES
// ================================================
import Unauthorized from "./pages/Unauthorized";
import { AIChatAssistant } from "./ai_assistant";
import AIChatAssistantForDiet from "./ai_assistant/AIChatAssistantForDiet";
// ================================================
// PHYSIO DASHBOARD
// ================================================
// Components
import Navbar from "./dashboard/physio/components/navbar";
import Sidebar from "./dashboard/physio/components/sidebar";
import Footer from "./dashboard/physio/components/Footer";

// Pages
import Profile from "./dashboard/physio/profile";

import UserDetails from "./dashboard/physio/userDetails";

import PhysiosAppointments from "./dashboard/physio/PhysiosAppointments";
import PhysioPasswordRequest from "./dashboard/physio/pages/PhysioPasswordRequest";
import PhysioPlanAssign from "./dashboard/physio/PhysioPlanAssign";

// ================================================
// DIET DASHBOARD
// ================================================
// Components
import DietNavbar from "./dashboard/diet/components/diet_navbar";
import DietSidebar from "./dashboard/diet/components/diet_sidebar";
import DietFooter from "./dashboard/diet/components/diet_footer";

// Pages
import DietProfile from "./dashboard/diet/diet_profile";
import DietAssign from "./dashboard/diet/diet_assign";
import DietPhysioo from "./dashboard/diet/diet_physioo";
import DietitiansAppointments from "./dashboard/diet/DietitiansAppointments";
import DietPasswordRequest from "./dashboard/diet/pages/DietPasswordRequest";
import DietMealPlanAssign from "./dashboard/diet/DietMealPlanAssign";
// ================================================
// DOCTOR DASHBOARD
// ================================================
```

```
// Components
import DoctorNavbar from "./dashboard/doctor/components/doctor_navbar";
import DoctorSidebar from "./dashboard/doctor/components/doctor_sidebar";
import DoctorFooter from "./dashboard/doctor/components/doctor_footer";

// Pages
import DoctorAssignmentDashboard from "./dashboard/doctor/DoctorAssignmentDashboard";
import DoctorsAppointments from "./dashboard/doctor/DoctorsAppointments";
import DoctorPasswordRequest from "./dashboard/doctor/pages/DoctorPasswordRequest";
import DoctorDashboardHome from "./dashboard/doctor/DoctorDashboardHome";
import DoctorSlotBooking from "./dashboard/doctor/DoctorSlotBooking";
// ================================================
// MASTER ADMIN DASHBOARD
// ================================================
// Components
import MasterAdminNavbar from "./dashboard/master_admin/components/master_admin_navbar";
import MasterAdminSidebar from "./dashboard/master_admin/components/master_admin_sidebar";
import MasterAdminFooter from "./dashboard/master_admin/components/master_admin_footer";

// Pages
import MasterAdminDashboard from "./dashboard/master_admin/master_admin_dashboard";
import UserManagement from "./dashboard/master_admin/UserManagement";
import SystemOverview from "./dashboard/master_admin/SystemOverview";
import PatientJourney from "./dashboard/master_admin/PatientJourney";
import AllReports from "./dashboard/master_admin/AllReports";
import ActivityLogs from "./dashboard/master_admin/ActivityLogs";
import SecurityControls from "./dashboard/master_admin/SecurityControls";
import Services from "./dashboard/master_admin/Services";
import AdminBlogSection from "./dashboard/master_admin/AdminBlogSection";
import ReportBuilder from "./dashboard/master_admin/ReportBuilder";
import TemplateDemo from "./dashboard/master_admin/TemplateDemo";
// Appointment Pages
import AppointmentsContainer from "./dashboard/master_admin/AppointmentsContainer";
import CounselorAppointments from "./dashboard/master_admin/CounselorAppointments";
import DoctorAppointments from "./dashboard/master_admin/DoctorAppointments";
import DietitianAppointments from "./dashboard/master_admin/DietitianAppointments";
import PhysioAppointments from "./dashboard/master_admin/PhysioAppointments";
import PhlebotomistAppointments from "./dashboard/master_admin/PhlebotomistAppointments";
import AdminPasswordRequests from "./dashboard/master_admin/AdminPasswordRequests";

// ================================================
// COUNSELOR DASHBOARD
// ================================================
// Components
import CounselorNavbar from "./dashboard/counselor/components/counselor_navbar";
import CounselorSidebar from "./dashboard/counselor/components/counselor_sidebar";
import CounselorFooter from "./dashboard/counselor/components/counselor_footer";

// Pages
import CounselorDashboard from "./dashboard/counselor/CounselorDashboard";
import CounselorsAppointments from "./dashboard/counselor/CounselorsAppointments";
import CounselorCompletedAppointments from "./dashboard/counselor/CounselorCompletedAppointments";
import CounselorPasswordRequest from "./dashboard/counselor/pages/CounselorPasswordRequest";


// ================================================
// PHLEBOTOMIST DASHBOARD
// ================================================
// Components
import PhlebotomistNavbar from "./dashboard/phlebotomist/components/phlebotomist_navbar";
import PhlebotomistSidebar from "./dashboard/phlebotomist/components/phlebotomist_sidebar";
import PhlebotomistFooter from "./dashboard/phlebotomist/components/phlebotomist_footer";

// Pages
import PhlebotomistDashboardHome from "./dashboard/phlebotomist/PhlebotomistDashboardHome";
import PhlebotomistAppointmentss from "./dashboard/phlebotomist/PhlebotomistAppointments";
import PhlebotomistBookingForm from "./dashboard/phlebotomist/PhlebotomistBookingForm";
import PhlebotomistPasswordRequest from "./dashboard/phlebotomist/pages/PhlebotomistPasswordRequest";

// ================================================
// LAYOUT COMPONENTS
// ================================================
```

```
const AuthLayout = () => {
  const { theme } = useTheme();
  return (
    <div className={`app ${theme}`}>
      <div className="auth-content">
        <Outlet />
      </div>
    </div>
  );
};

const PhysioDashboardLayout = () => {
  const { theme } = useTheme();
  return (
    <ProtectedRoute allowedRoles={["physio"]}>
      <div className={`app ${theme}`}>
        <Navbar />
        <div className="dashboard-layout">
          <Sidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <Footer />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};

const DietDashboardLayout = () => {
  const { theme } = useTheme();
  return (
    <ProtectedRoute allowedRoles={["dietitian"]}>
      <div className={`app ${theme}`}>
        <DietNavbar />
        <div className="dashboard-layout">
          <DietSidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <DietFooter />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};

const DoctorDashboardLayout = () => {
  const { theme } = useTheme();
  return (
    <ProtectedRoute allowedRoles={["doctor"]}>
      <div className={`app ${theme}`}>
        <DoctorNavbar />
        <div className="dashboard-layout">
          <DoctorSidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <DoctorFooter />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};

const MasterAdminDashboardLayout = () => {
  const { theme } = useTheme();
```

```
  return (
    <ProtectedRoute allowedRoles={["masteradmin"]}>
      <div className={`app ${theme}`}>
        <MasterAdminNavbar />
        <div className="dashboard-layout">
          <MasterAdminSidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <MasterAdminFooter />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};

const CounselorDashboardLayout = () => {
  const { theme } = useTheme();
  return (
    <ProtectedRoute allowedRoles={["counselor"]}>
      <div className={`app ${theme}`}>
        <CounselorNavbar />
        <div className="dashboard-layout">
          <CounselorSidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <CounselorFooter />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};
const PhlebotomistDashboardLayout = () => {
  const { theme } = useTheme();
  return (
    <ProtectedRoute allowedRoles={["phlebotomist"]}>
      <div className={`app ${theme}`}>
        <PhlebotomistNavbar />
        <div className="dashboard-layout">
          <PhlebotomistSidebar />
          <div className="dashboard-main">
            <div className="content-wrapper">
              <Outlet />
            </div>
            <PhlebotomistFooter />
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
};

// ================================================
// MAIN APP COMPONENT
// ================================================

const App = () => {
  return (
    <BrowserRouter>
      <ThemeProvider>
        <ToastContainer position="top-center" autoClose={2000} />
        <Routes>
          {/* ================================================
            AUTH ROUTES
            ================================================ */}
          <Route element={<AuthLayout />}>
            <Route path="/login" element={<Login />} />
            <Route path="/forgotpassword" element={<Forgot />} />
```

```
</Route>

{/* ================================================
  COMMON ROUTES
  ================================================ */}
<Route path="/unauthorized" element={<Unauthorized />} />
<Route path="/ai" element={<AIChatAssistant />} />
{/* ? AI Route Added */}
<Route
  path="diet_ai_assistant"
  element={<AIChatAssistantForDiet />}
/>
{/* ================================================
  PHYSIO DASHBOARD ROUTES
  ================================================ */}
<Route element={<PhysioDashboardLayout />}>
  <Route path="/" element={<Navigate to="/profile" replace />} />
  <Route path="/profile" element={<Profile />} />
  <Route path="/user" element={<UserDetails />} />
  <Route path="/user-details/:userId" element={<UserDetails />} />
  <Route
    path="/PhysiosAppointments"
    element={<PhysiosAppointments />}
  />
  <Route
    path="/PhysioPasswordRequest"
    element={<PhysioPasswordRequest />}
  />
  <Route path="/PhysioPlanAssign" element={<PhysioPlanAssign />} />
</Route>

{/* ================================================
  DIET DASHBOARD ROUTES
  ================================================ */}
<Route path="/diet" element={<DietDashboardLayout />}>
  <Route path="diet_profile" element={<DietProfile />} />
  <Route path="diet_assign" element={<DietAssign />} />
  <Route path="diet_physio" element={<DietPhysioo />} />
  <Route
    path="DietitiansAppointments"
    element={<DietitiansAppointments />}
  />
  <Route
    path="DietPasswordRequest"
    element={<DietPasswordRequest />}
  />
  <Route path="DietMealPlanAssign" element={<DietMealPlanAssign />} />
</Route>

{/* ================================================
  DOCTOR DASHBOARD ROUTES
  ================================================ */}
<Route path="/doctor" element={<DoctorDashboardLayout />}>
  <Route
    index
    element={<Navigate to="DoctorDashboardHome" replace />}
  />
  <Route
    path="DoctorsAppointments"
    element={<DoctorsAppointments />}
  />
  <Route
    path="DoctorAssignmentDashboard"
    element={<DoctorAssignmentDashboard />}
  />
  <Route
    path="DoctorPasswordRequest"
    element={<DoctorPasswordRequest />}
  />
  <Route
    path="DoctorDashboardHome"
    element={<DoctorDashboardHome />}
  />
```

```
    <Route path="DoctorSlotBooking" element={<DoctorSlotBooking />} />
  </Route>

  {/* ============================================
    MASTER ADMIN DASHBOARD ROUTES
    ============================================ */}
  <Route path="/masteradmin" element={<MasterAdminDashboardLayout />}>
    <Route index element={<MasterAdminDashboard />} />
    <Route path="UserManagement" element={<UserManagement />} />
    <Route path="SystemOverview" element={<SystemOverview />} />
    <Route path="PatientJourney" element={<PatientJourney />} />
    <Route path="AllReports" element={<AllReports />} />
    <Route path="ActivityLogs" element={<ActivityLogs />} />
    <Route path="SecurityControls" element={<SecurityControls />} />
    <Route path="Services" element={<Services />} />
    <Route path="AdminBlogSection" element={<AdminBlogSection />} />
    <Route path="appointments" element={<AppointmentsContainer />} />
    <Route
      path="appointments/counselor"
      element={<CounselorAppointments />}
    />
    <Route
      path="appointments/doctor"
      element={<DoctorAppointments />}
    />
    <Route
      path="appointments/dietitian"
      element={<DietitianAppointments />}
    />
    <Route
      path="appointments/physio"
      element={<PhysioAppointments />}
    />
    <Route
      path="appointments/phlebotomist"
      element={<PhlebotomistAppointments />}
    />
    <Route
      path="AdminPasswordRequests"
      element={<AdminPasswordRequests />}
    />
    <Route path="ReportBuilder" element={<ReportBuilder />} />
    <Route path="TemplateDemo" element={<TemplateDemo />} />
  </Route>

  {/* ============================================
    COUNSELOR DASHBOARD ROUTES
    ============================================ */}
  <Route path="/counselor" element={<CounselorDashboardLayout />}>
    <Route index element={<CounselorDashboard />} />
    <Route
      path="CounselorsAppointments"
      element={<CounselorsAppointments />}
    />
    <Route
      path="CounselorCompletedAppointments"
      element={<CounselorCompletedAppointments />}
    />
    <Route
      path="CounselorPasswordRequest"
      element={<CounselorPasswordRequest />}
    />
  </Route>
  <Route path="/phlebotomist" element={<PhlebotomistDashboardLayout />}>
    <Route index element={<PhlebotomistDashboardHome />} />
    <Route path="PhlebotomistAppointmentss" element={<PhlebotomistAppointmentss />} />
    <Route path="PhlebotomistBookingForm" element={<PhlebotomistBookingForm />} />
    <Route
      path="PhlebotomistPasswordRequest"
      element={<PhlebotomistPasswordRequest />}
    />
  </Route>
</Routes>
```

```
      </ThemeProvider>
    </BrowserRouter>
  );
};

export default App;
```

File: src\index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

File: src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { ThemeProvider } from './ThemeProvider'; // ? Import ThemeProvider

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <ThemeProvider>           {/* ? Wrap App here */}
      <App />
    </ThemeProvider>
  </React.StrictMode>
);

reportWebVitals();
```

File: src\reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

File: src\setupTests.js

```
// Importing the necessary extensions for jest-dom
import '@testing-library/jest-dom';

// Mocking the submit method globally to avoid errors during tests
beforeAll(() => {
  HTMLFormElement.prototype.submit = jest.fn();  // Mocking the form submit method
});
```

File: src\theme.css

```
/* theme.css */
:root {
  /* Light theme (default) */
  --bg-primary: #ffffff;
  --bg-secondary: #f8f9fa;
  --bg-hover: #e9ecef;
  --text-primary: #212529;
```

```
  --text-secondary: #6c757d;
  --accent: #cc5500; /* Client's requested copper/orange color */
  --accent-hover: #b34700; /* Darker version for hover states */
  --accent-light: #ff7733; /* Lighter version for highlights */
  --border-color: #dee2e6;
  --shadow-color: rgba(0, 0, 0, 0.1);
}

[data-theme="dark"] {
  /* Dark theme */
  --bg-primary: #121212;
  --bg-secondary: #1e1e1e;
  --bg-hover: #2d2d2d;
  --text-primary: #f8f9fa;
  --text-secondary: #adb5bd;
  --accent: #cc5500; /* Client's requested copper/orange color */
  --accent-hover: #ff6a1a; /* Brighter version for dark mode hover */
  --accent-light: #ff8c4d; /* Lighter version for dark mode highlights */
  --border-color: #343a40;
  --shadow-color: rgba(0, 0, 0, 0.3);
}

/* Global styles for the theme */
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen,
    Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;
  background-color: var(--bg-primary);
  color: var(--text-primary);
  transition: background-color 0.3s ease, color 0.3s ease;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  color: var(--text-primary);
}

a {
  color: var(--accent);
  text-decoration: none;
}
input[type="date"],
input[type="time"],
input[type="text"] {
  color-scheme: light;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
}

button {
  cursor: pointer;
}

/* Applies to all inputs, selects, and textareas */

select {
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  padding: 0.5rem;
  border-radius: 0.5rem;
  transition: background-color 0.3s, color 0.3s, border-color 0.3s;
}

input::placeholder,
textarea::placeholder {
  color: var(--text-secondary);
}
```

```
/* Focus state */
input:focus,
textarea:focus,
select:focus {
  outline: none;
  border-color: var(--accent);
  box-shadow: 0 0 0 2px var(--accent-light);
}

/* Override system color-scheme to avoid auto-dark mode styling */
input,
textarea,
select {
  color-scheme: light dark; /* lets browser use custom CSS instead of forced dark inputs */
}
```

File: src\ThemeProvider.js

```
import React, { useEffect, useState, createContext, useContext } from "react";

const ThemeContext = createContext();

export const ThemeProvider = ({ children }) => {
  // Use theme from <html data-theme> set in index.html preload script
  const [theme, setTheme] = useState(
    () => document.documentElement.getAttribute("data-theme") || "light"
  );

  // Sync theme with localStorage and <html> attribute
  useEffect(() => {
    document.documentElement.setAttribute("data-theme", theme);
    localStorage.setItem("theme", theme);
  }, [theme]);

  // Toggle theme
  const toggleTheme = () => {
    setTheme((prev) => (prev === "light" ? "dark" : "light"));
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

// Custom hook
export const useTheme = () => useContext(ThemeContext);
```

File: src\ai_assistant\AIChatAssistant.css

```
/* AIChatAssistant.css - Clean Modern UI - Optimized Responsive with Fixed Button Layouts */

:root {
  --primary-gradient: linear-gradient(135deg, #cc5500 0%, #ff7f24 100%);
  --accent-gradient: linear-gradient(135deg, #cc5500 0%, #e67300 100%);
  --success-gradient: linear-gradient(135deg, #ffa94d 0%, #ffc166 100%);

  --bg-primary: #ffffff;
  --bg-secondary: #f8f4ef;
  --bg-tertiary: #f6eee6;
  --bg-hover: #f0e0d3;
  --bg-gradient: linear-gradient(135deg, #fff7f0 0%, #ffeedd 100%);

  --text-primary: #331a00;
  --text-secondary: #804000;
  --text-muted: #b36b00;

  --border-color: #e2cbb3;
  --border-hover: #d9b385;

  --shadow-sm: 0 1px 2px 0 rgba(204, 85, 0, 0.1);
  --shadow-md: 0 4px 6px -1px rgba(204, 85, 0, 0.2),
    0 2px 4px -1px rgba(204, 85, 0, 0.1);
  --shadow-lg: 0 10px 15px -3px rgba(204, 85, 0, 0.2),
```

```
    0 4px 6px -2px rgba(204, 85, 0, 0.1);
  --shadow-xl: 0 20px 25px -5px rgba(204, 85, 0, 0.2),
    0 10px 10px -5px rgba(204, 85, 0, 0.1);

  --radius-sm: 0.5rem;
  --radius-md: 0.75rem;
  --radius-lg: 0.5rem;
  --radius-xl: 0.5rem;

  --container-padding: clamp(0.5rem, 2.5vw, 1rem);
  --header-padding: clamp(1rem, 2vw, 1.5rem);
  --chat-padding: clamp(1rem, 2vw, 1.5rem);
  --input-padding: clamp(1rem, 2vw, 1.5rem);
  --logo-size: clamp(2.5rem, 5vw, 3rem);
  --header-font: clamp(1.125rem, 3vw, 1.5rem);
  --chat-height: calc(100vh - 220px);
  --bubble-max-width: clamp(70%, 80vw, 85%);
  --button-height: 3rem;
  --button-min-width: 3rem;
}

[data-theme="dark"] {
  --bg-primary: #1a1108;
  --bg-secondary: #331a00;
  --bg-tertiary: #4d2600;
  --bg-hover: #663300;
  --bg-gradient: linear-gradient(135deg, #331a00 0%, #4d2600 100%);

  --text-primary: #ffeedd;
  --text-secondary: #ffcc99;
  --text-muted: #e67300;

  --border-color: #4d2600;
  --border-hover: #663300;

  --shadow-sm: 0 1px 2px 0 rgba(204, 85, 0, 0.4);
  --shadow-md: 0 4px 6px -1px rgba(204, 85, 0, 0.5),
    0 2px 4px -1px rgba(204, 85, 0, 0.4);
  --shadow-lg: 0 10px 15px -3px rgba(204, 85, 0, 0.5),
    0 4px 6px -2px rgba(204, 85, 0, 0.4);
  --shadow-xl: 0 20px 25px -5px rgba(204, 85, 0, 0.5),
    0 10px 10px -5px rgba(204, 85, 0, 0.4);
}

* {
  box-sizing: border-box;
}

body {
  margin: 0;
  padding: 0;
  height: 100%;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background: var(--bg-gradient);
  min-height: 100vh;
  transition: all 0.3s ease;
}

.ai-container {
  max-width: 50%;
  margin: var(--container-padding) auto;
  position: relative;
  transition: all 0.3s ease;
  display: flex;
  flex-direction: column;
  height: 96vh;
}

.ai-main-card {
```

```
  flex: 1;
  display: flex;
  flex-direction: column;
  background: var(--bg-primary);
  border-radius: var(--radius-xl);
  box-shadow: var(--shadow-xl);
  overflow: hidden;
  border: 1px solid var(--border-color);
  height: calc(90vh - 2rem);
}

.ai-header {
  background: var(--bg-secondary);
  padding: var(--header-padding);
  display: flex;
  align-items: center;
  justify-content: space-between;
  border-bottom: 1px solid var(--border-color);
  gap: 1rem;
  flex-shrink: 0;
}

.ai-header-content {
  display: flex;
  align-items: center;
  gap: clamp(0.5rem, 2vw, 1rem);
  min-width: 0;
  flex: 1;
}

@keyframes bounce {
  0%,
  100% {
    transform: translateY(0);
    animation-timing-function: ease-in-out;
  }
  50% {
    transform: translateY(-15px);
    animation-timing-function: ease-in-out;
  }
}

.ai-logo {
  width: var(--logo-size);
  height: var(--logo-size);
  background: var(--primary-gradient);
  border-radius: var(--radius-lg);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: clamp(1.25rem, 3vw, 1.5rem);
  box-shadow: var(--shadow-md);
  flex-shrink: 0;

  /* ? Animation */
  animation: bounce 5s infinite;
}

.ai-logo:hover {
  animation-play-state: paused;
}

.ai-header-text {
  min-width: 0;
  flex: 1;
}

.ai-header-text h1 {
  margin: 0;
  font-size: var(--header-font);
  font-weight: 700;
  color: var(--text-primary);
  background: var(--primary-gradient);
```

```
  background-clip: text;
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  line-height: 1.2;
}

.ai-header-text p {
  margin: 0.25rem 0 0 0;
  font-size: clamp(0.8rem, 2vw, 0.875rem);
  color: var(--text-secondary);
  line-height: 1.3;
}

.theme-toggle {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0.75rem;
  cursor: pointer;
  transition: all 0.2s ease;
  color: var(--text-primary);
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-min-width);
  height: var(--button-min-width);
}

.theme-toggle:hover {
  background: var(--bg-hover);
  transform: translateY(-2px);
  box-shadow: var(--shadow-md);
  border-color: var(--border-hover);
}

.theme-toggle:active {
  transform: translateY(0);
}
/* Chat Window Scrollable Styles - Cross-device compatible */
.ai-chat-window {
  flex: 1;
  overflow-y: auto;
  overflow-x: hidden;
  padding: var(--chat-padding);
  display: flex;
  flex-direction: column;
  gap: 1rem;
  background: var(--bg-primary);
  min-height: 0;
  height: 0; /* This is crucial - forces the flex item to respect flex: 1 */

  /* Enhanced scrolling behavior */
  -webkit-overflow-scrolling: touch; /* Smooth scrolling on iOS */
  scroll-behavior: smooth;
  overscroll-behavior: contain; /* Prevents parent scrolling */
}

/* Custom scrollbar for webkit browsers */
.ai-chat-window::-webkit-scrollbar {
  width: 6px;
}

.ai-chat-window::-webkit-scrollbar-track {
  background: transparent;
}

.ai-chat-window::-webkit-scrollbar-thumb {
  background: var(--border-color);
  border-radius: 3px;
  transition: background 0.2s ease;
}
```

```
.ai-chat-window::-webkit-scrollbar-thumb:hover {
  background: var(--border-hover);
}

/* Firefox scrollbar styling */
.ai-chat-window {
  scrollbar-width: thin;
  scrollbar-color: var(--border-color) transparent;
}

.chat-message {
  display: flex;
  align-items: flex-start;
  gap: 0.75rem;
  animation: slideIn 0.3s ease-out;
}

.chat-message.user {
  flex-direction: row-reverse;
}

.chat-avatar {
  width: 2rem;
  height: 2rem;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 0.875rem;
  flex-shrink: 0;
}

.chat-avatar.user {
  background: var(--accent-gradient);
  color: white;
}

.chat-avatar.bot {
  background: var(--success-gradient);
  color: white;
}

.chat-bubble {
  max-width: var(--bubble-max-width);
  padding: clamp(0.75rem, 2vw, 1rem) clamp(1rem, 3vw, 1.25rem);
  border-radius: var(--radius-lg);
  position: relative;
  word-wrap: break-word;
  line-height: 1.5;
  font-size: clamp(0.9rem, 2vw, 0.95rem);
  box-shadow: var(--shadow-sm);
  transition: all 0.2s ease;
}

.chat-bubble:hover {
  transform: translateY(-1px);
  box-shadow: var(--shadow-md);
}

.chat-bubble.user {
  background: linear-gradient(135deg, #ff7e5f, #feb47b);
  border-bottom-right-radius: 0.5rem;
}

.chat-bubble.bot {
  background: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  border-bottom-left-radius: 0.5rem;
}

/* File Card Styles - Fixed button alignment */
.file-card {
```

```
  cursor: pointer;
  background: rgba(255, 255, 255, 0.1);
  border: 1px solid rgba(255, 255, 255, 0.2);
  border-radius: var(--radius-md);
  padding: 0.75rem;
  margin-bottom: 0.75rem;
  display: flex;
  align-items: center;
  gap: 0.75rem;
  backdrop-filter: blur(5px);
  transition: all 0.2s ease;
}

.file-card:hover {
  background: rgba(255, 255, 255, 0.15);
  transform: translateY(-1px);
  box-shadow: var(--shadow-md);
}

.file-info {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  flex: 1;
  min-width: 0;
  overflow: hidden;
}

.file-icon {
  width: 2rem;
  height: 2rem;
  background: rgba(255, 255, 255, 0.2);
  border-radius: var(--radius-sm);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 0.875rem;
  flex-shrink: 0;
}

.file-name {
  flex: 1;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  min-width: 0;
}

.preview-btn {
  background: rgba(255, 255, 255, 0.2);
  border: none;
  border-radius: var(--radius-sm);
  padding: 0.5rem;
  cursor: pointer;
  color: inherit;
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2rem;
  height: 2rem;
}

.preview-btn:hover {
  background: rgba(255, 255, 255, 0.3);
  transform: translateY(-1px);
}

/* Loading Animation */
.loading-animation {
  display: flex;
  justify-content: center;
```

```
  align-items: center;
  gap: 0.5rem;
  padding: 1rem;
}

.loading-dot {
  width: 0.5rem;
  height: 0.5rem;
  background: var(--text-muted);
  border-radius: 50%;
  animation: bounce 1.4s infinite ease-in-out;
}

.loading-dot:nth-child(1) {
  animation-delay: -0.32s;
}
.loading-dot:nth-child(2) {
  animation-delay: -0.16s;
}
.loading-dot:nth-child(3) {
  animation-delay: 0s;
}

.ai-input-area {
  flex-shrink: 0;
  background: var(--bg-secondary);
  border-top: 1px solid var(--border-color);
  padding: var(--input-padding);
}

.ai-header,
.ai-input-area {
  flex-shrink: 0;
}

.file-upload-preview {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 1rem;
  margin-bottom: 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  animation: slideIn 0.3s ease-out;
  gap: 1rem;
}

.file-upload-info {
  display: flex;
  align-items: center;
  gap: 0.75rem;
  flex: 1;
  min-width: 0;
  overflow: hidden;
}

.file-upload-icon {
  width: 2.5rem;
  height: 2.5rem;
  background: var(--primary-gradient);
  border-radius: var(--radius-md);
  display: flex;
  align-items: center;
  justify-content: center;
  color: white;
  font-size: 1rem;
  flex-shrink: 0;
}

.file-upload-name {
  flex: 1;
  white-space: nowrap;
```

```
  overflow: hidden;
  text-overflow: ellipsis;
  min-width: 0;
  color: var(--text-primary);
  font-weight: 500;
}

.file-remove-btn {
  background: var(--bg-hover);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-sm);
  padding: 0.5rem;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2rem;
  height: 2rem;
}

.file-remove-btn:hover {
  background: #fee2e2;
  color: #dc2626;
  border-color: #fecaca;
  transform: translateY(-1px);
}

.file-remove-btn:active {
  transform: translateY(0);
}

/* Input Controls - Completely redesigned for better button alignment */
.input-controls {
  display: flex;
  align-items: stretch;
  gap: 0.75rem;
  min-height: var(--button-height);
}

.file-upload-btn {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-height);
  height: var(--button-height);
  position: relative;
  overflow: hidden;
}

.file-upload-btn:hover {
  background: var(--bg-hover);
  color: var(--text-primary);
  transform: translateY(-1px);
  border-color: var(--border-hover);
  box-shadow: var(--shadow-sm);
}

.file-upload-btn:active {
  transform: translateY(0);
}

.message-input {
```

```
  flex: 1;
  padding: 0.75rem;
  border: 1px solid var(--border-color);
  border-radius: 0.5rem;
  background: var(--card-bg);
  color: var(--text);
  font-size: 0.9rem;
  transition: all 0.3s ease;
  height: 40px;
  min-height: 48px;
}

.message-input:focus {
  outline: none;
  border-color: var(--primary);
}

.message-input::placeholder {
  color: var(--text-muted);
}

.message-input::-webkit-scrollbar {
  width: 6px;
}

.message-input::-webkit-scrollbar-track {
  background: var(--bg-tertiary);
  border-radius: 3px;
}

.message-input::-webkit-scrollbar-thumb {
  border-radius: 3px;
  transition: background 0.3s ease;
}

.message-input::-webkit-scrollbar-thumb:hover {
  background: #e67300;
}

.action-btn {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-height);
  height: var(--button-height);
  font-size: 0;
}

.action-btn:hover:not(:disabled) {
  background: var(--bg-hover);
  color: var(--text-primary);
  transform: translateY(-1px);
  border-color: var(--border-hover);
  box-shadow: var(--shadow-sm);
}

.action-btn:active:not(:disabled) {
  transform: translateY(0);
}

.action-btn:disabled {
  opacity: 0.5;
  cursor: not-allowed;
  transform: none;
}
```

```
.action-btn:disabled:hover {
  background: var(--bg-tertiary);
  color: var(--text-secondary);
  border-color: var(--border-color);
  box-shadow: none;
}

.send-btn {
  background: var(--primary-gradient);
  border: none;
  color: white;
  box-shadow: var(--shadow-md);
  border: 1px solid transparent;
}

.send-btn:hover:not(:disabled) {
  transform: translateY(-2px);
  box-shadow: var(--shadow-lg);
  background: var(--primary-gradient);
}

.send-btn:active:not(:disabled) {
  transform: translateY(-1px);
}

.send-btn:disabled {
  opacity: 0.6;
  background: var(--bg-tertiary);
  color: var(--text-muted);
  box-shadow: none;
}

/* Drag and Drop Styles */
.drag-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(102, 126, 234, 0.1);
  backdrop-filter: blur(8px);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
  padding: 1rem;
}

.drag-content {
  background: var(--bg-primary);
  border: 2px dashed #667eea;
  border-radius: var(--radius-xl);
  padding: clamp(2rem, 5vw, 3rem);
  text-align: center;
  box-shadow: var(--shadow-xl);
  animation: bounce 0.5s ease-out;
  max-width: 90vw;
}

.drag-icon {
  font-size: clamp(2rem, 5vw, 3rem);
  margin-bottom: 1rem;
  color: var(--text-primary);
}

.drag-title {
  font-size: clamp(1.25rem, 3vw, 1.5rem);
  font-weight: 700;
  color: var(--text-primary);
  margin-bottom: 0.5rem;
}
```

```
.drag-subtitle {
  color: var(--text-secondary);
  font-size: clamp(0.9rem, 2vw, 1rem);
}

/* File Preview Modal - Fixed button alignment */
.file-preview-modal {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.8);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 2000;
  padding: 1rem;
}

.file-preview-content {
  background: var(--bg-primary);
  border-radius: var(--radius-xl);
  overflow: hidden;
  width: 50%;
  height: 80%;
  position: relative;
  box-shadow: var(--shadow-xl);
  display: flex;
  flex-direction: column;
}

.file-preview-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 1.5rem;
  border-bottom: 1px solid var(--border-color);
  background: var(--bg-secondary);
  gap: 1rem;
  flex-shrink: 0;
}

.file-preview-title {
  flex: 1;
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  min-width: 0;
  font-size: clamp(0.9rem, 2vw, 1rem);
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
}

.file-preview-close {
  background: var(--bg-hover);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-md);
  padding: 0.75rem;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2.5rem;
  height: 2.5rem;
}

.file-preview-close:hover {
  background: #fee2e2;
```

```
  color: #dc2626;
  border-color: #fecaca;
  transform: translateY(-1px);
}

.file-preview-close:active {
  transform: translateY(0);
}

.file-preview-body {
  padding: 1.5rem;
  /* overflow: auto; */
  /* flex: 1; */
  min-height: 100%;
}

.file-preview-image {
  max-width: 100%;
  height: auto;
  border-radius: var(--radius-lg);
  display: block;
  margin: 0 auto;
}

.file-preview-unsupported {
  text-align: center;
  padding: 3rem;
  color: var(--text-secondary);
}

.file-preview-unsupported p {
  margin: 0;
  font-size: 1.1rem;
}

/* Bot Markdown Styles */
.bot-markdown {
  font-size: clamp(0.9rem, 2vw, 0.95rem);
  line-height: 1.6;
  color: var(--text-primary);
  margin: 0;
}

.bot-markdown * {
  margin-top: 0;
  margin-bottom: 0.5rem;
}

.bot-markdown *:last-child {
  margin-bottom: 0;
}

.bot-markdown table {
  width: 100%;
  border-collapse: collapse;
  margin: 1rem 0;
  overflow-x: auto;
  display: block;
  white-space: nowrap;
}

.bot-markdown th,
.bot-markdown td {
  border: 1px solid var(--border-color);
  padding: 0.75rem;
  text-align: left;
}

.bot-markdown th {
  background: var(--bg-tertiary);
  font-weight: 600;
}
```

```
.bot-markdown ul,
.bot-markdown ol {
  margin-left: 1.5rem;
  margin-bottom: 1rem;
}

.bot-markdown ul {
  list-style-type: disc;
}

.bot-markdown ol {
  list-style-type: decimal;
}

.bot-markdown li {
  margin-bottom: 0.25rem;
}

.bot-markdown pre {
  background: var(--bg-tertiary);
  padding: 1rem;
  border-radius: var(--radius-md);
  overflow-x: auto;
  margin: 1rem 0;
  border: 1px solid var(--border-color);
}

.bot-markdown code {
  background: var(--bg-tertiary);
  padding: 0.2rem 0.4rem;
  border-radius: var(--radius-sm);
  font-family: "SFMono-Regular", "Monaco", "Inconsolata", "Roboto Mono",
    monospace;
  font-size: 0.9em;
  border: 1px solid var(--border-color);
}

.bot-markdown pre code {
  background: transparent;
  padding: 0;
  border: none;
}

.bot-markdown blockquote {
  border-left: 4px solid var(--border-color);
  padding-left: 1rem;
  margin: 1rem 0;
  color: var(--text-secondary);
  font-style: italic;
}

.bot-markdown h1,
.bot-markdown h2,
.bot-markdown h3,
.bot-markdown h4,
.bot-markdown h5,
.bot-markdown h6 {
  color: var(--text-primary);
  font-weight: 600;
  margin-top: 1.5rem;
  margin-bottom: 0.5rem;
}

.bot-markdown h1 {
  font-size: 1.5rem;
}

.bot-markdown h2 {
  font-size: 1.3rem;
}

.bot-markdown h3 {
  font-size: 1.1rem;
```

```
}

/* Animations */
@keyframes slideIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

@keyframes bounce {
  0%,
  80%,
  100% {
    transform: translateY(0);
  }
  40% {
    transform: translateY(-10px);
  }
}

@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

@media (max-width: 768px) {
  .ai-header {
    flex-wrap: wrap;
    gap: 0.75rem;
  }

  .ai-container {
    height: 97vh;
    max-width: 100%;
  }

  .ai-header-content {
    flex: 1;
    min-width: 0;
  }

  .theme-toggle {
    order: 1;
  }
  .ai-chat-window {
    padding: 1rem;
    gap: 0.75rem;
  }

  .ai-chat-window::-webkit-scrollbar {
    width: 8px;
  }
}

@media (max-width: 640px) {
  .input-controls {
    gap: 0.5rem;
  }

  .ai-container {
    height: 97vh;
    max-width: 100%;
  }
```

```
  .message-input {
    min-height: 2.5rem;
  }

  .action-btn,
  .file-upload-btn {
    min-width: 2.5rem;
    height: 2.5rem;
  }

  .chat-bubble {
    max-width: 90%;
  }
}
@media (max-width: 480px) {
  .ai-container {
    padding: 0.5rem;
    height: 98vh;
    max-width: 100%;
  }

  .ai-header {
    padding: 1rem;
  }

  .file-preview-content {
    width: 100%;
    height: 100%;
  }

  .ai-chat-window {
    padding: 0.75rem;
    gap: 0.5rem;
  }

  .ai-chat-window::-webkit-scrollbar {
    width: 10px;
  }

  .ai-input-area {
    padding: 1rem;
  }

  .message-input {
    min-height: 2.5rem;
  }

  .input-controls {
    flex-wrap: wrap;
    gap: 0.5rem;
  }

  .action-btn,
  .file-upload-btn,
  .send-btn {
    min-width: 2.5rem;
    height: 2.5rem;
  }

  .chat-bubble {
    max-width: 90%;
  }
}
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

@keyframes fadeOut {
```

```
  from {
    opacity: 1;
  }
  to {
    opacity: 0;
  }
}

.error-message {
  position: absolute;
  bottom: 6.5rem;
  right: 1.5rem;
  background: #fee2e2;
  color: #dc2626;
  border: 1px solid #fecaca;
  padding: 0.5rem 1rem;
  border-radius: var(--radius-md);
  font-size: 0.875rem;
  box-shadow: var(--shadow-md);
  z-index: 10;
  animation: fadeIn 0.3s ease-out, fadeOut 0.3s ease-in-out 2.7s;
  pointer-events: none;
}
@keyframes pulseGlow {
  0% {
    box-shadow: 0 0 0 0 rgba(220, 38, 38, 0.6);
  }
  70% {
    box-shadow: 0 0 0 10px rgba(220, 38, 38, 0);
  }
  100% {
    box-shadow: 0 0 0 0 rgba(220, 38, 38, 0);
  }
}

.action-btn.voice-active {
  animation: pulseGlow 1.5s infinite;
  background: #fee2e2;
  border-color: #fecaca;
  color: #dc2626;
}
```

File: src\ai_assistant\AIChatAssistant.js

```
import React, { useState, useEffect, useRef } from "react";
import ReactMarkdown from "react-markdown";

import {
  Upload,
  Send,
  RefreshCcw,
  X,
  Moon,
  Sun,
  Mic,
  Sparkles,
  Brush,
  Dna,
} from "lucide-react";
import "./AIChatAssistant.css";

const AIChatAssistant = () => {
  const [theme, setTheme] = useState("light");
  const [input, setInput] = useState("");
  const [messages, setMessages] = useState([
    {
      type: "bot",
      text: "? Hello! I'm your AI assistant. How can I help you today?",
    },
  ]);
  const [file, setFile] = useState(null);
  const [loading, setLoading] = useState(false);
  const [dragActive, setDragActive] = useState(false);
  const [lastPrompt, setLastPrompt] = useState("");
```

```
const [previewFile, setPreviewFile] = useState(null);
const [error, setError] = useState("");
const recognitionRef = useRef(null);
const [isVoiceMode, setIsVoiceMode] = useState(false);
const [isListening, setIsListening] = useState(false);

const chatEndRef = useRef(null);

const MAX_FILE_SIZE = 5 * 1024 * 1024; // 5MB

useEffect(() => {
  document.documentElement.setAttribute("data-theme", theme);
  chatEndRef.current?.scrollIntoView({ behavior: "smooth" });
}, [messages, theme]);

const toggleTheme = () => {
  setTheme((prev) => (prev === "light" ? "dark" : "light"));
};
useEffect(() => {
  const SpeechRecognition =
    window.SpeechRecognition || window.webkitSpeechRecognition;

  if (SpeechRecognition) {
    recognitionRef.current = new SpeechRecognition();
    recognitionRef.current.continuous = true;
    recognitionRef.current.lang = "en-US";
    recognitionRef.current.interimResults = false;

    recognitionRef.current.onresult = (event) => {
      const transcript = event.results[0][0].transcript;
      setInput(transcript); // sets input for logging
      sendVoiceMessage(transcript); // sends it automatically
    };

    recognitionRef.current.onerror = (event) => {
      console.error("Speech recognition error:", event.error);
      setIsListening(false);
    };

    recognitionRef.current.onend = () => {
      setIsListening(false);
    };
  }
}, []);
const sendVoiceMessage = (voiceText) => {
  const newMessage = { type: "user", text: voiceText, file: null };
  setMessages((prev) => [...prev, newMessage]);
  setLastPrompt(voiceText);
  setInput("");
  setLoading(true);

  setTimeout(() => {
    const responseText = `You said: "${voiceText}". Here's a sample voice reply!`;
    const botMsg = {
      type: "bot",
      text: responseText,
    };
    setMessages((prev) => [...prev, botMsg]);
    setLoading(false);
    speakText(responseText); // use speech synthesis
  }, 1200);
};
const speakText = (text) => {
  const synth = window.speechSynthesis;
  if (!synth) return;

  const utter = new SpeechSynthesisUtterance(text);
  utter.lang = "en-US";
  utter.rate = 1;
  utter.pitch = 1;

  utter.onend = () => {
    if (isVoiceMode) {
```

```
      startListening(); // loop to listen again
    }
  };

  synth.speak(utter);
};
const toggleVoiceMode = () => {
  const newState = !isVoiceMode;
  setIsVoiceMode(newState);
  if (newState) {
    startListening();
  } else {
    stopListening();
  }
};

const startListening = () => {
  if (recognitionRef.current && !isListening) {
    recognitionRef.current.start();
    setIsListening(true);
  }
};

const stopListening = () => {
  if (recognitionRef.current && isListening) {
    recognitionRef.current.stop();
    setIsListening(false);
  }
};

const isValidFile = (file) => {
  return (
    file &&
    ["application/pdf", "image/jpeg", "image/png"].includes(file.type) &&
    file.size <= MAX_FILE_SIZE
  );
};

const handleFileChange = (file) => {
  if (!isValidFile(file)) {
    setError("Only PDF/JPG/PNG files under 5MB are allowed.");
    setTimeout(() => setError(""), 3000);
    return;
  }
  setError("");
  setFile(file);
};

const handleSend = () => {
  if (!input.trim() && !file) {
    setError("Please enter a message or upload a valid file.");
    setTimeout(() => setError(""), 3000);
    return;
  }

  setError("");
  const newMessage = { type: "user", text: input, file };
  setMessages((prev) => [...prev, newMessage]);
  setLastPrompt(input);
  setInput("");
  setFile(null);
  setLoading(true);

  setTimeout(() => {
    const botMsg = {
      type: "bot",
      text: `I understand you said: "${
        input || "[file uploaded]"
      }". This is a mock response for demonstration.`,
    };
    setMessages((prev) => [...prev, botMsg]);
    setLoading(false);
  }, 1000);
```

```
};

const handleRegenerate = () => {
  if (!lastPrompt) return;
  setInput(lastPrompt);
  handleSend();
};

const handleDrop = (e) => {
  e.preventDefault();
  e.stopPropagation();
  setDragActive(false);
  if (e.dataTransfer.files[0]) {
    handleFileChange(e.dataTransfer.files[0]);
  }
};

const handleKeyDown = (e) => {
  if (e.key === "Enter" && !e.shiftKey) {
    e.preventDefault();
    handleSend();
  }
};

const handleDrag = (e) => {
  e.preventDefault();
  e.stopPropagation();
  if (e.type === "dragenter" || e.type === "dragover") setDragActive(true);
  else if (e.type === "dragleave") setDragActive(false);
};

const renderFilePreview = (file) => {
  const fileUrl = URL.createObjectURL(file);
  const isPDF = file.type === "application/pdf";
  const isImage = file.type.startsWith("image/");

  return (
    <div className="file-preview-modal">
      <div className="file-preview-content">
        <div className="file-preview-header">
          <h3 className="file-preview-title">{file.name}</h3>
          <button
            className="file-preview-close"
            onClick={() => setPreviewFile(null)}
          >
            <X size={20} />
          </button>
        </div>
        <div className="file-preview-body">
          {isPDF ? (
            <embed
              src={fileUrl}
              type="application/pdf"
              width="100%"
              height="600px"
            />
          ) : isImage ? (
            <img src={fileUrl} alt="preview" className="file-preview-image" />
          ) : (
            <div className="file-preview-unsupported">
              <p>Cannot preview this file type.</p>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

const clearChat = () => {
  setMessages([
    {
      type: "bot",
```

```
      text: "? Hello! I'm your AI assistant. How can I help you today?",
    },
  ]);
  setError("");
  setInput("");
  setFile(null);
  setLoading(false);
  setLastPrompt("");
};

return (
  <div
    className={`ai-container`}
    onDrop={handleDrop}
    onDragOver={handleDrag}
    onDragEnter={handleDrag}
    onDragLeave={handleDrag}
  >
    <div className="ai-main-card">
      {/* Header */}
      <div className="ai-header">
        <div className="ai-header-content">
          <div className="ai-logo">
            <Dna size={35} color="#fff" />
          </div>
          <div className="ai-header-text">
            <h1>Assistant</h1>
            <p>Powered by dnalyst</p>
          </div>
        </div>
        <div style={{ display: "flex", gap: "0.5rem" }}>
          <button className="theme-toggle" onClick={toggleTheme}>
            {theme === "dark" ? <Sun /> : <Moon />}
          </button>
          <button
            className="theme-toggle"
            onClick={clearChat}
            title="Clear Chat"
          >
            <Brush size={20} />
          </button>
        </div>
      </div>

      {/* Chat Window */}
      <div className="ai-chat-window">
        {messages.map((msg, i) => (
          <div key={i} className={`chat-message ${msg.type}`}>
            <div className={`chat-avatar ${msg.type}`}>
              {msg.type === "user" ? "?" : "?"}
            </div>
            <div className={`chat-bubble ${msg.type}`}>
              {msg.file && (
                <div
                  className="file-card"
                  onClick={() => setPreviewFile(msg.file)}
                  role="button"
                  tabIndex={0}
                >
                  <div className="file-info">
                    <span className="file-name">{msg.file.name}</span>
                  </div>
                </div>
              )}
              <ReactMarkdown
                components={{
                  p: ({ node, ...props }) => (
                    <p className="bot-markdown" {...props} />
                  ),
                }}
              >
                {msg.text}
              </ReactMarkdown>
```

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

  {loading && (
    <div className="loading-animation">
      <div className="loading-dot"></div>
      <div className="loading-dot"></div>
      <div className="loading-dot"></div>
    </div>
  )}
  <div ref={chatEndRef} />
</div>

{/* Input Area */}
<div className="ai-input-area">
  {file && (
    <div className="file-upload-preview">
      <div className="file-upload-info">
        <div className="file-upload-icon">?</div>
        <span className="file-upload-name">{file.name}</span>
      </div>
      <button className="file-remove-btn" onClick={() => setFile(null)}>
        <X size={16} />
      </button>
    </div>
  )}

  {error && (
    <div key={error} className="error-message">
      {error}
    </div>
  )}

  <div className="input-controls">
    <label className="file-upload-btn">
      <input
        type="file"
        style={{ display: "none" }}
        onChange={(e) => handleFileChange(e.target.files[0])}
      />
      <Upload size={20} />
    </label>

    <textarea
      className="message-input"
      placeholder="Type your message..."
      value={input}
      onChange={(e) => setInput(e.target.value)}
      onKeyDown={handleKeyDown}
    />

    <button
      className="action-btn"
      onClick={handleRegenerate}
      disabled={!lastPrompt}
    >
      <RefreshCcw size={20} />
    </button>
    <button
      className={`action-btn ${isVoiceMode ? "voice-active" : ""}`}
      onClick={toggleVoiceMode}
      title="Use Voice Mode"
    >
      <Mic color={isVoiceMode ? "red" : "currentColor"} />
    </button>

    <button
      className="action-btn send-btn"
      onClick={handleSend}
      disabled={!input.trim() && !file}
    >
      <Send size={20} />
```

```
            </button>
          </div>
        </div>
      </div>

      {dragActive && (
        <div className="drag-overlay">
          <div className="drag-content">
            <Upload size={48} />
            <div className="drag-title">Drop your file here</div>
            <div className="drag-subtitle">Release to upload</div>
          </div>
        </div>
      )}

      {previewFile && renderFilePreview(previewFile)}
    </div>
  );
};

export default AIChatAssistant;
```

File: src\ai_assistant\AIChatAssistantForDiet.css

```
/* AIChatAssistant.css - Clean Modern UI - Optimized Responsive with Fixed Button Layouts */

:root {
  --primary-gradient: linear-gradient(135deg, #cc5500 0%, #ff7f24 100%);
  --accent-gradient: linear-gradient(135deg, #cc5500 0%, #e67300 100%);
  --success-gradient: linear-gradient(135deg, #ffa94d 0%, #ffc166 100%);

  --bg-primary: #ffffff;
  --bg-secondary: #f8f4ef;
  --bg-tertiary: #f6eee6;
  --bg-hover: #f0e0d3;
  --bg-gradient: linear-gradient(135deg, #fff7f0 0%, #ffeedd 100%);

  --text-primary: #331a00;
  --text-secondary: #804000;
  --text-muted: #b36b00;

  --border-color: #e2cbb3;
  --border-hover: #d9b385;

  --shadow-sm: 0 1px 2px 0 rgba(204, 85, 0, 0.1);
  --shadow-md: 0 4px 6px -1px rgba(204, 85, 0, 0.2),
    0 2px 4px -1px rgba(204, 85, 0, 0.1);
  --shadow-lg: 0 10px 15px -3px rgba(204, 85, 0, 0.2),
    0 4px 6px -2px rgba(204, 85, 0, 0.1);
  --shadow-xl: 0 20px 25px -5px rgba(204, 85, 0, 0.2),
    0 10px 10px -5px rgba(204, 85, 0, 0.1);

  --radius-sm: 0.5rem;
  --radius-md: 0.75rem;
  --radius-lg: 0.5rem;
  --radius-xl: 0.5rem;

  --container-padding: clamp(0.5rem, 2.5vw, 1rem);
  --header-padding: clamp(1rem, 2vw, 1.5rem);
  --chat-padding: clamp(1rem, 2vw, 1.5rem);
  --input-padding: clamp(1rem, 2vw, 1.5rem);
  --logo-size: clamp(2.5rem, 5vw, 3rem);
  --header-font: clamp(1.125rem, 3vw, 1.5rem);
  --chat-height: calc(100vh - 220px);
  --bubble-max-width: clamp(70%, 80vw, 85%);
  --button-height: 3rem;
  --button-min-width: 3rem;
}

[data-theme="dark"] {
  --bg-primary: #1a1108;
  --bg-secondary: #331a00;
  --bg-tertiary: #4d2600;
  --bg-hover: #663300;
```

```
  --bg-gradient: linear-gradient(135deg, #331a00 0%, #4d2600 100%);

  --text-primary: #ffeedd;
  --text-secondary: #ffcc99;
  --text-muted: #e67300;

  --border-color: #4d2600;
  --border-hover: #663300;

  --shadow-sm: 0 1px 2px 0 rgba(204, 85, 0, 0.4);
  --shadow-md: 0 4px 6px -1px rgba(204, 85, 0, 0.5),
    0 2px 4px -1px rgba(204, 85, 0, 0.4);
  --shadow-lg: 0 10px 15px -3px rgba(204, 85, 0, 0.5),
    0 4px 6px -2px rgba(204, 85, 0, 0.4);
  --shadow-xl: 0 20px 25px -5px rgba(204, 85, 0, 0.5),
    0 10px 10px -5px rgba(204, 85, 0, 0.4);
}

* {
  box-sizing: border-box;
}

body {
  margin: 0;
  padding: 0;
  height: 100%;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background: var(--bg-gradient);
  min-height: 100vh;
  transition: all 0.3s ease;
}

.ai-container {
  max-width: 50%;
  margin: var(--container-padding) auto;
  position: relative;
  transition: all 0.3s ease;
  display: flex;
  flex-direction: column;
  height: 96vh;
}

.ai-main-card {
  flex: 1;
  display: flex;
  flex-direction: column;
  background: var(--bg-primary);
  border-radius: var(--radius-xl);
  box-shadow: var(--shadow-xl);
  overflow: hidden;
  border: 1px solid var(--border-color);
  height: calc(90vh - 2rem);
}

.ai-header {
  background: var(--bg-secondary);
  padding: var(--header-padding);
  display: flex;
  align-items: center;
  justify-content: space-between;
  border-bottom: 1px solid var(--border-color);
  gap: 1rem;
  flex-shrink: 0;
}

.ai-header-content {
  display: flex;
  align-items: center;
  gap: clamp(0.5rem, 2vw, 1rem);
```

```
  min-width: 0;
  flex: 1;
}

@keyframes bounce {
  0%,
  100% {
    transform: translateY(0);
    animation-timing-function: ease-in-out;
  }
  50% {
    transform: translateY(-15px);
    animation-timing-function: ease-in-out;
  }
}

.ai-logo {
  width: var(--logo-size);
  height: var(--logo-size);
  background: var(--primary-gradient);
  border-radius: var(--radius-lg);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: clamp(1.25rem, 3vw, 1.5rem);
  box-shadow: var(--shadow-md);
  flex-shrink: 0;

  /* ? Animation */
  animation: bounce 5s infinite;
}

.ai-logo:hover {
  animation-play-state: paused;
}

.ai-header-text {
  min-width: 0;
  flex: 1;
}

.ai-header-text h1 {
  margin: 0;
  font-size: var(--header-font);
  font-weight: 700;
  color: var(--text-primary);
  background: var(--primary-gradient);
  background-clip: text;
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  line-height: 1.2;
}

.ai-header-text p {
  margin: 0.25rem 0 0 0;
  font-size: clamp(0.8rem, 2vw, 0.875rem);
  color: var(--text-secondary);
  line-height: 1.3;
}

.theme-toggle {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0.75rem;
  cursor: pointer;
  transition: all 0.2s ease;
  color: var(--text-primary);
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-min-width);
```

```
  height: var(--button-min-width);
}

.theme-toggle:hover {
  background: var(--bg-hover);
  transform: translateY(-2px);
  box-shadow: var(--shadow-md);
  border-color: var(--border-hover);
}

.theme-toggle:active {
  transform: translateY(0);
}
/* Chat Window Scrollable Styles - Cross-device compatible */
.ai-chat-window {
  flex: 1;
  overflow-y: auto;
  overflow-x: hidden;
  padding: var(--chat-padding);
  display: flex;
  flex-direction: column;
  gap: 1rem;
  background: var(--bg-primary);
  min-height: 0;
  height: 0; /* This is crucial - forces the flex item to respect flex: 1 */

  /* Enhanced scrolling behavior */
  -webkit-overflow-scrolling: touch; /* Smooth scrolling on iOS */
  scroll-behavior: smooth;
  overscroll-behavior: contain; /* Prevents parent scrolling */
}

/* Custom scrollbar for webkit browsers */
.ai-chat-window::-webkit-scrollbar {
  width: 6px;
}

.ai-chat-window::-webkit-scrollbar-track {
  background: transparent;
}

.ai-chat-window::-webkit-scrollbar-thumb {
  background: var(--border-color);
  border-radius: 3px;
  transition: background 0.2s ease;
}

.ai-chat-window::-webkit-scrollbar-thumb:hover {
  background: var(--border-hover);
}

/* Firefox scrollbar styling */
.ai-chat-window {
  scrollbar-width: thin;
  scrollbar-color: var(--border-color) transparent;
}

.chat-message {
  display: flex;
  align-items: flex-start;
  gap: 0.75rem;
  animation: slideIn 0.3s ease-out;
}

.chat-message.user {
  flex-direction: row-reverse;
}

.chat-avatar {
  width: 2rem;
  height: 2rem;
  border-radius: 50%;
  display: flex;
```

```
  align-items: center;
  justify-content: center;
  font-size: 0.875rem;
  flex-shrink: 0;
}

.chat-avatar.user {
  background: var(--accent-gradient);
  color: white;
}

.chat-avatar.bot {
  background: var(--success-gradient);
  color: white;
}

.chat-bubble {
  max-width: var(--bubble-max-width);
  padding: clamp(0.75rem, 2vw, 1rem) clamp(1rem, 3vw, 1.25rem);
  border-radius: var(--radius-lg);
  position: relative;
  word-wrap: break-word;
  line-height: 1.5;
  font-size: clamp(0.9rem, 2vw, 0.95rem);
  box-shadow: var(--shadow-sm);
  transition: all 0.2s ease;
}

.chat-bubble:hover {
  transform: translateY(-1px);
  box-shadow: var(--shadow-md);
}

.chat-bubble.user {
  background: linear-gradient(135deg, #ff7e5f, #feb47b);
  border-bottom-right-radius: 0.5rem;
}

.chat-bubble.bot {
  background: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  border-bottom-left-radius: 0.5rem;
}

/* File Card Styles - Fixed button alignment */
.file-card {
  cursor: pointer;
  background: rgba(255, 255, 255, 0.1);
  border: 1px solid rgba(255, 255, 255, 0.2);
  border-radius: var(--radius-md);
  padding: 0.75rem;
  margin-bottom: 0.75rem;
  display: flex;
  align-items: center;
  gap: 0.75rem;
  backdrop-filter: blur(5px);
  transition: all 0.2s ease;
}

.file-card:hover {
  background: rgba(255, 255, 255, 0.15);
  transform: translateY(-1px);
  box-shadow: var(--shadow-md);
}

.file-info {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  flex: 1;
  min-width: 0;
  overflow: hidden;
```

```
}

.file-icon {
  width: 2rem;
  height: 2rem;
  background: rgba(255, 255, 255, 0.2);
  border-radius: var(--radius-sm);
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 0.875rem;
  flex-shrink: 0;
}

.file-name {
  flex: 1;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  min-width: 0;
}

.preview-btn {
  background: rgba(255, 255, 255, 0.2);
  border: none;
  border-radius: var(--radius-sm);
  padding: 0.5rem;
  cursor: pointer;
  color: inherit;
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2rem;
  height: 2rem;
}

.preview-btn:hover {
  background: rgba(255, 255, 255, 0.3);
  transform: translateY(-1px);
}

/* Loading Animation */
.loading-animation {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 0.5rem;
  padding: 1rem;
}

.loading-dot {
  width: 0.5rem;
  height: 0.5rem;
  background: var(--text-muted);
  border-radius: 50%;
  animation: bounce 1.4s infinite ease-in-out;
}

.loading-dot:nth-child(1) {
  animation-delay: -0.32s;
}
.loading-dot:nth-child(2) {
  animation-delay: -0.16s;
}
.loading-dot:nth-child(3) {
  animation-delay: 0s;
}

.ai-input-area {
  flex-shrink: 0;
  background: var(--bg-secondary);
```

```
  border-top: 1px solid var(--border-color);
  padding: var(--input-padding);
}

.ai-header,
.ai-input-area {
  flex-shrink: 0;
}

.file-upload-preview {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 1rem;
  margin-bottom: 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  animation: slideIn 0.3s ease-out;
  gap: 1rem;
}

.file-upload-info {
  display: flex;
  align-items: center;
  gap: 0.75rem;
  flex: 1;
  min-width: 0;
  overflow: hidden;
}

.file-upload-icon {
  width: 2.5rem;
  height: 2.5rem;
  background: var(--primary-gradient);
  border-radius: var(--radius-md);
  display: flex;
  align-items: center;
  justify-content: center;
  color: white;
  font-size: 1rem;
  flex-shrink: 0;
}

.file-upload-name {
  flex: 1;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  min-width: 0;
  color: var(--text-primary);
  font-weight: 500;
}

.file-remove-btn {
  background: var(--bg-hover);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-sm);
  padding: 0.5rem;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2rem;
  height: 2rem;
}

.file-remove-btn:hover {
  background: #fee2e2;
  color: #dc2626;
```

```
  border-color: #fecaca;
  transform: translateY(-1px);
}

.file-remove-btn:active {
  transform: translateY(0);
}

/* Input Controls - Completely redesigned for better button alignment */
.input-controls {
  display: flex;
  align-items: stretch;
  gap: 0.75rem;
  min-height: var(--button-height);
}

.file-upload-btn {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-height);
  height: var(--button-height);
  position: relative;
  overflow: hidden;
}

.file-upload-btn:hover {
  background: var(--bg-hover);
  color: var(--text-primary);
  transform: translateY(-1px);
  border-color: var(--border-hover);
  box-shadow: var(--shadow-sm);
}

.file-upload-btn:active {
  transform: translateY(0);
}

.message-input {
  flex: 1;
  padding: 0.75rem;
  border: 1px solid var(--border-color);
  border-radius: 0.5rem;
  background: var(--card-bg);
  color: var(--text);
  font-size: 0.9rem;
  transition: all 0.3s ease;
  height: 40px;
  min-height: 48px;
}

.message-input:focus {
  outline: none;
  border-color: #cc5500;
  box-shadow: 0 0 0 2px rgba(204, 85, 0, 0.2); /* optional glow effect */
}

.message-input::placeholder {
  color: var(--text-muted);
}

.message-input::-webkit-scrollbar {
  width: 6px;
}
```

```
.message-input::-webkit-scrollbar-track {
  background: var(--bg-tertiary);
  border-radius: 3px;
}

.message-input::-webkit-scrollbar-thumb {
  border-radius: 3px;
  transition: background 0.3s ease;
}

.message-input::-webkit-scrollbar-thumb:hover {
  background: #e67300;
}

.action-btn {
  background: var(--bg-tertiary);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-lg);
  padding: 0;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-shrink: 0;
  min-width: var(--button-height);
  height: var(--button-height);
  font-size: 0;
}

.action-btn:hover:not(:disabled) {
  background: var(--bg-hover);
  color: var(--text-primary);
  transform: translateY(-1px);
  border-color: var(--border-hover);
  box-shadow: var(--shadow-sm);
}

.action-btn:active:not(:disabled) {
  transform: translateY(0);
}

.action-btn:disabled {
  opacity: 0.5;
  cursor: not-allowed;
  transform: none;
}

.action-btn:disabled:hover {
  background: var(--bg-tertiary);
  color: var(--text-secondary);
  border-color: var(--border-color);
  box-shadow: none;
}

.send-btn {
  background: var(--primary-gradient);
  border: none;
  color: white;
  box-shadow: var(--shadow-md);
  border: 1px solid transparent;
}

.send-btn:hover:not(:disabled) {
  transform: translateY(-2px);
  box-shadow: var(--shadow-lg);
  background: var(--primary-gradient);
}

.send-btn:active:not(:disabled) {
  transform: translateY(-1px);
}
```

```
.send-btn:disabled {
  opacity: 0.6;
  background: var(--bg-tertiary);
  color: var(--text-muted);
  box-shadow: none;
}

/* Drag and Drop Styles */
.drag-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(102, 126, 234, 0.1);
  backdrop-filter: blur(8px);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
  padding: 1rem;
}

.drag-content {
  background: var(--bg-primary);
  border: 2px dashed #667eea;
  border-radius: var(--radius-xl);
  padding: clamp(2rem, 5vw, 3rem);
  text-align: center;
  box-shadow: var(--shadow-xl);
  animation: bounce 0.5s ease-out;
  max-width: 90vw;
}

.drag-icon {
  font-size: clamp(2rem, 5vw, 3rem);
  margin-bottom: 1rem;
  color: var(--text-primary);
}

.drag-title {
  font-size: clamp(1.25rem, 3vw, 1.5rem);
  font-weight: 700;
  color: var(--text-primary);
  margin-bottom: 0.5rem;
}

.drag-subtitle {
  color: var(--text-secondary);
  font-size: clamp(0.9rem, 2vw, 1rem);
}

/* File Preview Modal - Fixed button alignment */
.file-preview-modal {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.8);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 2000;
  padding: 1rem;
}

.file-preview-content {
  background: var(--bg-primary);
  border-radius: var(--radius-xl);
  overflow: hidden;
  width: 50%;
```

```
  height: 80%;
  position: relative;
  box-shadow: var(--shadow-xl);
  display: flex;
  flex-direction: column;
}

.file-preview-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 1.5rem;
  border-bottom: 1px solid var(--border-color);
  background: var(--bg-secondary);
  gap: 1rem;
  flex-shrink: 0;
}

.file-preview-title {
  flex: 1;
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
  min-width: 0;
  font-size: clamp(0.9rem, 2vw, 1rem);
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
}

.file-preview-close {
  background: var(--bg-hover);
  border: 1px solid var(--border-color);
  border-radius: var(--radius-md);
  padding: 0.75rem;
  cursor: pointer;
  color: var(--text-secondary);
  transition: all 0.2s ease;
  flex-shrink: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  min-width: 2.5rem;
  height: 2.5rem;
}

.file-preview-close:hover {
  background: #fee2e2;
  color: #dc2626;
  border-color: #fecaca;
  transform: translateY(-1px);
}

.file-preview-close:active {
  transform: translateY(0);
}

.file-preview-body {
  padding: 1.5rem;
  /* overflow: auto; */
  /* flex: 1; */
  min-height: 100%;
}

.file-preview-image {
  max-width: 100%;
  height: auto;
  border-radius: var(--radius-lg);
  display: block;
  margin: 0 auto;
}

.file-preview-unsupported {
```

```
  text-align: center;
  padding: 3rem;
  color: var(--text-secondary);
}

.file-preview-unsupported p {
  margin: 0;
  font-size: 1.1rem;
}

/* Bot Markdown Styles */
.bot-markdown {
  font-size: clamp(0.9rem, 2vw, 0.95rem);
  line-height: 1.6;
  color: var(--text-primary);
  margin: 0;
}

.bot-markdown * {
  margin-top: 0;
  margin-bottom: 0.5rem;
}

.bot-markdown *:last-child {
  margin-bottom: 0;
}

.bot-markdown table {
  width: 100%;
  border-collapse: collapse;
  margin: 1rem 0;
  overflow-x: auto;
  display: block;
  white-space: nowrap;
}

.bot-markdown th,
.bot-markdown td {
  border: 1px solid var(--border-color);
  padding: 0.75rem;
  text-align: left;
}

.bot-markdown th {
  background: var(--bg-tertiary);
  font-weight: 600;
}

.bot-markdown ul,
.bot-markdown ol {
  margin-left: 1.5rem;
  margin-bottom: 1rem;
}

.bot-markdown ul {
  list-style-type: disc;
}

.bot-markdown ol {
  list-style-type: decimal;
}

.bot-markdown li {
  margin-bottom: 0.25rem;
}

.bot-markdown pre {
  background: var(--bg-tertiary);
  padding: 1rem;
  border-radius: var(--radius-md);
  overflow-x: auto;
  margin: 1rem 0;
  border: 1px solid var(--border-color);
```

```
}

.bot-markdown code {
  background: var(--bg-tertiary);
  padding: 0.2rem 0.4rem;
  border-radius: var(--radius-sm);
  font-family: "SFMono-Regular", "Monaco", "Inconsolata", "Roboto Mono",
    monospace;
  font-size: 0.9em;
  border: 1px solid var(--border-color);
}

.bot-markdown pre code {
  background: transparent;
  padding: 0;
  border: none;
}

.bot-markdown blockquote {
  border-left: 4px solid var(--border-color);
  padding-left: 1rem;
  margin: 1rem 0;
  color: var(--text-secondary);
  font-style: italic;
}

.bot-markdown h1,
.bot-markdown h2,
.bot-markdown h3,
.bot-markdown h4,
.bot-markdown h5,
.bot-markdown h6 {
  color: var(--text-primary);
  font-weight: 600;
  margin-top: 1.5rem;
  margin-bottom: 0.5rem;
}

.bot-markdown h1 {
  font-size: 1.5rem;
}

.bot-markdown h2 {
  font-size: 1.3rem;
}

.bot-markdown h3 {
  font-size: 1.1rem;
}

/* Animations */
@keyframes slideIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

@keyframes bounce {
  0%,
  80%,
  100% {
    transform: translateY(0);
  }
  40% {
    transform: translateY(-10px);
  }
}
```

```
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

@media (max-width: 768px) {
  .ai-header {
    flex-wrap: wrap;
    gap: 0.75rem;
  }

  .ai-container {
    height: 97vh;
    max-width: 100%;
  }

  .ai-header-content {
    flex: 1;
    min-width: 0;
  }

  .theme-toggle {
    order: 1;
  }
  .ai-chat-window {
    padding: 1rem;
    gap: 0.75rem;
  }

  .ai-chat-window::-webkit-scrollbar {
    width: 8px;
  }
}

@media (max-width: 640px) {
  .input-controls {
    gap: 0.5rem;
  }

  .ai-container {
    height: 97vh;
    max-width: 100%;
  }

  .message-input {
    min-height: 2.5rem;
  }

  .action-btn,
  .file-upload-btn {
    min-width: 2.5rem;
    height: 2.5rem;
  }

  .chat-bubble {
    max-width: 90%;
  }
}
@media (max-width: 480px) {
  .ai-container {
    padding: 0.5rem;
    height: 98vh;
    max-width: 100%;
  }

  .ai-header {
    padding: 1rem;
  }
```

```
  .file-preview-content {
    width: 100%;
    height: 100%;
  }

  .ai-chat-window {
    padding: 0.75rem;
    gap: 0.5rem;
  }

  .ai-chat-window::-webkit-scrollbar {
    width: 10px;
  }

  .ai-input-area {
    padding: 1rem;
  }

  .message-input {
    min-height: 2.5rem;
  }

  .input-controls {
    flex-wrap: wrap;
    gap: 0.5rem;
  }

  .action-btn,
  .file-upload-btn,
  .send-btn {
    min-width: 2.5rem;
    height: 2.5rem;
  }

  .chat-bubble {
    max-width: 90%;
  }
}
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

@keyframes fadeOut {
  from {
    opacity: 1;
  }
  to {
    opacity: 0;
  }
}

.error-message {
  position: absolute;
  bottom: 6.5rem;
  right: 1.5rem;
  background: #fee2e2;
  color: #dc2626;
  border: 1px solid #fecaca;
  padding: 0.5rem 1rem;
  border-radius: var(--radius-md);
  font-size: 0.875rem;
  box-shadow: var(--shadow-md);
  z-index: 10;
  animation: fadeIn 0.3s ease-out, fadeOut 0.3s ease-in-out 2.7s;
  pointer-events: none;
}
.ai-chat-window-container {
  width: 100%;
```

```
  transition: all 0.3s ease-in-out;
}

.ai-edit-window {
  animation: slideInRight 0.3s ease;
  background-color: var(--bg-primary);
  border-radius: var(--radius-md);
  padding: 1.5rem;
  min-height: 300px;
}

.edit-textarea {
  width: 100%;
  padding: 1rem;
  border-radius: 8px;
  resize: vertical;
  min-height: 200px;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
}

.chat-line-wrapper {
  display: flex;
  align-items: flex-start;
  gap: 0.5rem;
  margin-bottom: 1rem;
}

.edit-icon {
  background: transparent;
  border: none;
  cursor: pointer;
  margin-top: 4px;
  color: #888;
  transition: transform 0.2s ease;
}

.edit-icon:hover {
  transform: scale(1.2);
  color: #cc5500;
}
.ai-edit-side-panel {
  position: fixed;
  top: 80px; /* below header */
  right: 20px;
  width: 350px;
  height: calc(100vh - 120px);
  background-color: var(--bg-primary);
  border-left: 2px solid var(--border-color);
  padding: 1rem;
  border-radius: 8px;
  box-shadow: var(--shadow-lg);
  z-index: 1001;
  overflow-y: auto;
  animation: slideInPanel 0.3s ease-in-out;
}

@keyframes slideInPanel {
  from {
    transform: translateX(100%);
    opacity: 0;
  }
  to {
    transform: translateX(0%);
    opacity: 1;
  }
}

.edit-textarea {
  width: 100%;
  min-height: 150px;
  height: auto;
```

```
  resize: vertical;
  padding: 1rem;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  border-radius: 8px;
  font-family: monospace;
}


/* Optional: remove ugly scrollbar from inside textarea */
.edit-textarea::-webkit-scrollbar {
  width: 6px;
}
.edit-textarea::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 3px;
}

.ai-edit-side-panel {
  position: fixed;
  top: 80px;
  right: 20px;
  max-height: calc(90vh - 100px); /* match ai-main-card */
  overflow-y: auto;
  background-color: var(--bg-primary);
  border-left: 2px solid var(--border-color);
  padding: 1rem;
  border-radius: 8px;
  box-shadow: var(--shadow-lg);
  z-index: 1001;
  animation: slideInPanel 0.3s ease-in-out;
}

.edit-scroll-wrapper {
  /* remove overflow */
  max-height: unset;
  overflow: unset;
  padding-right: 0;
}


/* Custom Scrollbar inside edit panel */
.edit-scroll-wrapper::-webkit-scrollbar {
  width: 6px;
}
.edit-scroll-wrapper::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 4px;
}
.edit-scroll-wrapper::-webkit-scrollbar-track {
  background-color: var(--bg-tertiary);
}
.edit-panel-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1rem;
}


.edit-close-btn {
  background: transparent;
  border: none;
  color: var(--text-muted);
  cursor: pointer;
  padding: 0.4rem;
  border-radius: 50%;
  transition: all 0.2s ease;
}

.edit-close-btn:hover {
  background: #fee2e2;
```

```
  color: #dc2626;
  transform: scale(1.1);
}
/* Webkit scrollbar styling */
.custom-scroll::-webkit-scrollbar {
  width: 8px;
}

.custom-scroll::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 4px;
}

.custom-scroll::-webkit-scrollbar-track {
  background: transparent;
}
```

File: src\ai_assistant\AIChatAssistantForDiet.js

```
import { useNavigate, useLocation } from "react-router-dom"; // ? Add at top
import React, { useState, useEffect, useRef } from "react";
import ReactMarkdown from "react-markdown";

import {
  Upload,
  Send,
  RefreshCcw,
  X,
  Moon,
  Sun,
  Sparkles,
  Brush,
  Pencil,
  CookingPot,
} from "lucide-react";
import "./AIChatAssistantForDiet.css";
import { toast } from "react-toastify";
const AIChatAssistantForDiet = () => {
  const [theme, setTheme] = useState("light");
  const [input, setInput] = useState("");
  const [messages, setMessages] = useState([
    {
      type: "bot",
      text: "?? Hello! I'm your AI diet assistant. How can I help you with a meal plan today?",
    },
  ]);
  const navigate = useNavigate();
  const [file, setFile] = useState(null);
  const [loading, setLoading] = useState(false);
  const [dragActive, setDragActive] = useState(false);
  const [lastPrompt, setLastPrompt] = useState("");
  const [previewFile, setPreviewFile] = useState(null);
  const [error, setError] = useState("");
  const [isEditing, setIsEditing] = useState(false);
  const [editableContent, setEditableContent] = useState("");

  const chatEndRef = useRef(null);
  const location = useLocation();
  const aiInitialData = location.state?.aiInitialData || {};

  const MAX_FILE_SIZE = 5 * 1024 * 1024; // 5MB

  useEffect(() => {
    document.documentElement.setAttribute("data-theme", theme);
    chatEndRef.current?.scrollIntoView({ behavior: "smooth" });
  }, [messages, theme]);

  const toggleTheme = () => {
    setTheme((prev) => (prev === "light" ? "dark" : "light"));
  };

  const isValidFile = (file) => {
    return (
      file &&
```

```
    ["application/pdf", "image/jpeg", "image/png"].includes(file.type) &&
    file.size <= MAX_FILE_SIZE
  );
};

const handleFileChange = (file) => {
  if (!isValidFile(file)) {
    setError("Only PDF/JPG/PNG files under 5MB are allowed.");
    setTimeout(() => setError(""), 3000);
    return;
  }
  setError("");
  setFile(file);
};

const handleSend = () => {
  if (!input.trim() && !file) {
    setError("Please enter a message or upload a valid file.");
    setTimeout(() => setError(""), 3000);
    return;
  }

  setError("");
  const newMessage = { type: "user", text: input, file };
  setMessages((prev) => [...prev, newMessage]);
  setLastPrompt(input);
  setInput("");
  setFile(null);
  setLoading(true);

  setTimeout(() => {
    const botMsg = {
      type: "bot",
      text: `I understand you said: "${
        input || "[file uploaded]"
      }". This is a mock response for demonstration.`,
    };
    setMessages((prev) => [...prev, botMsg]);
    setLoading(false);
  }, 1000);
};

const handleRegenerate = () => {
  if (!lastPrompt) return;
  setInput(lastPrompt);
  handleSend();
};

const handleDrop = (e) => {
  e.preventDefault();
  e.stopPropagation();
  setDragActive(false);
  if (e.dataTransfer.files[0]) {
    handleFileChange(e.dataTransfer.files[0]);
  }
};

const handleKeyDown = (e) => {
  if (e.key === "Enter" && !e.shiftKey) {
    e.preventDefault();
    handleSend();
  }
};

const handleDrag = (e) => {
  e.preventDefault();
  e.stopPropagation();
  if (e.type === "dragenter" || e.type === "dragover") setDragActive(true);
  else if (e.type === "dragleave") setDragActive(false);
};

const renderFilePreview = (file) => {
  const fileUrl = URL.createObjectURL(file);
```

```
  const isPDF = file.type === "application/pdf";
  const isImage = file.type.startsWith("image/");

  return (
    <div className="file-preview-modal">
      <div className="file-preview-content">
        <div className="file-preview-header">
          <h3 className="file-preview-title">{file.name}</h3>
          <button
            className="file-preview-close"
            onClick={() => setPreviewFile(null)}
          >
            <X size={20} />
          </button>
        </div>
        <div className="file-preview-body">
          {isPDF ? (
            <embed
              src={fileUrl}
              type="application/pdf"
              width="100%"
              height="600px"
            />
          ) : isImage ? (
            <img src={fileUrl} alt="preview" className="file-preview-image" />
          ) : (
            <div className="file-preview-unsupported">
              <p>Cannot preview this file type.</p>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

const clearChat = () => {
  setMessages([
    {
      type: "bot",
      text: "? Hello! I'm your AI assistant. How can I help you today?",
    },
  ]);
  setError("");
  setInput("");
  setFile(null);
  setLoading(false);
  setLastPrompt("");
};

return (
  <div
    className={`ai-container`}
    onDrop={handleDrop}
    onDragOver={handleDrag}
    onDragEnter={handleDrag}
    onDragLeave={handleDrag}
  >
    <div className="ai-main-card">
      {/* Header */}
      <div className="ai-header">
        <div className="ai-header-content">
          <div className="ai-logo">
            <CookingPot size={20} color="#fff" />
          </div>
          <div className="ai-header-text">
            <h1>Diet Assistant</h1>
            <p>Powered by dnalyst</p>
          </div>
        </div>
        <div style={{ display: "flex", gap: "0.5rem" }}>
          <button
            className="theme-toggle"
```

```
      onClick={clearChat}
      title="Clear Chat"
    >
      <Brush size={20} />
    </button>
    <button className="theme-toggle" onClick={toggleTheme}>
      {theme === "dark" ? <Sun /> : <Moon />}
    </button>
    <button
      className="theme-toggle"
      onClick={() => navigate("/diet/DietMealPlanAssign")}
      title="Back to Diet Plan Assign"
    >
      <X size={20} />
    </button>
  </div>
</div>

{/* Chat Window */}
<div className="ai-chat-window">
  {messages.map((msg, i) => (
    <div key={i} className={`chat-message ${msg.type}`}>
      <div className={`chat-avatar ${msg.type}`}>
        {msg.type === "user" ? "?" : "?"}
      </div>
      <div className={`chat-bubble ${msg.type}`}>
        {msg.file && (
          <div
            className="file-card"
            onClick={() => setPreviewFile(msg.file)}
            role="button"
            tabIndex={0}
          >
            <div className="file-info">
              <span className="file-name">{msg.file.name}</span>
            </div>
          </div>
        )}
        <ReactMarkdown
          components={{
            p: ({ node, ...props }) => (
              <p className="bot-markdown" {...props} />
            ),
          }}
        >
          {msg.text}
        </ReactMarkdown>
      </div>

      {msg.type === "bot" && (
        <Pencil
          size={18}
          className="edit-icon"
          onClick={() => {
            setEditableContent(msg.text);
            setIsEditing(true);
          }}
        />
      )}
    </div>
  ))}

  {loading && (
    <div className="loading-animation">
      <div className="loading-dot"></div>
      <div className="loading-dot"></div>
      <div className="loading-dot"></div>
    </div>
  )}
  <div ref={chatEndRef} />
</div>

{/* Input Area */}
```

```
  <div className="ai-input-area">
    {file && (
      <div className="file-upload-preview">
        <div className="file-upload-info">
          <div className="file-upload-icon">?</div>
          <span className="file-upload-name">{file.name}</span>
        </div>
        <button className="file-remove-btn" onClick={() => setFile(null)}>
          <X size={16} />
        </button>
      </div>
    )}

    {error && (
      <div key={error} className="error-message">
        {error}
      </div>
    )}

    <div className="input-controls">
      <label className="file-upload-btn">
        <input
          type="file"
          style={{ display: "none" }}
          onChange={(e) => handleFileChange(e.target.files[0])}
        />
        <Upload size={20} />
      </label>

      <textarea
        className="message-input"
        placeholder="Type your message..."
        value={input}
        onChange={(e) => setInput(e.target.value)}
        onKeyDown={handleKeyDown}
      />

      <button
        className="action-btn"
        onClick={handleRegenerate}
        disabled={!lastPrompt}
      >
        <RefreshCcw size={20} />
      </button>

      <button
        className="action-btn send-btn"
        onClick={handleSend}
        disabled={!input.trim() && !file}
      >
        <Send size={20} />
      </button>
    </div>
  </div>
</div>

{dragActive && (
  <div className="drag-overlay">
    <div className="drag-content">
      <Upload size={48} />
      <div className="drag-title">Drop your file here</div>
      <div className="drag-subtitle">Release to upload</div>
    </div>
  </div>
)}

{previewFile && renderFilePreview(previewFile)}
{/* ? Right-side sliding Edit Panel */}
{isEditing && (
  <div className="ai-edit-side-panel">
    <div className="edit-panel-header">
      <h3>Edit Diet Plan</h3>
      <button
```

```
              onClick={() => setIsEditing(false)}
              className="edit-close-btn"
              aria-label="Close"
            >
              <X size={16} />
            </button>
          </div>

          <div className="edit-scroll-wrapper">
            <textarea
              value={editableContent}
              onChange={(e) => setEditableContent(e.target.value)}
              className="edit-textarea"
            />
            <button
              className="btn btn-success"
              style={{ marginTop: "1rem" }}
              onClick={() => {
                toast.success("Diet chart successfully saved!", {
                  autoClose: 1500,
                });
                setIsEditing(false);
                navigate("/diet/DietMealPlanAssign", {
                  state: {
                    aiGeneratedText: editableContent,
                    ...aiInitialData,
                  },
                });
              }}
            >
              Save Changes
            </button>
          </div>
        </div>
      )}
    </div>
  );
};

export default AIChatAssistantForDiet;
```

File: src\ai_assistant\ai_service.js

```
// src/ai_assistant/ai_service.js
export const askAI = async (prompt, file) => {
  const formData = new FormData();
  formData.append("prompt", prompt);
  if (file) formData.append("file", file);

  const response = await fetch("http://localhost:8080/api/ai", {
    method: "POST",
    body: formData,
  });

  return await response.json();
};
```

File: src\ai_assistant\index.js

```
// src/ai_assistant/index.js

export { default as AIChatAssistant } from "./AIChatAssistant";
export * from "./ai_service"; // Optional: only if you plan to use the API logic from other files
```

File: src\auth\forgotpassword.css

```
.register-container {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    background-color: #f9f9f9;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }
```

```
  h2 {
    text-align: center;
    color: #cc5500;
  }

  .form-group {
    margin-bottom: 15px;
  }

  label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
  }

  input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }

  .register-button {
    width: 100%;
    padding: 10px;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }

  .register-button:hover {
    background-color: #0056b3;
  }
```

File: src\auth\forgotpassword.js

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import { ArrowLeft } from "lucide-react";
import "./forgotpassword.css";

const ForgotPassword = () => {
  const [email, setEmail] = useState("");
  const [message, setMessage] = useState("");
  const [showMessage, setShowMessage] = useState(false);
  const [messageType, setMessageType] = useState(""); // 'success' | 'error'
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    if (message) {
      setShowMessage(true);
      const timer = setTimeout(() => setShowMessage(false), 3000);
      return () => clearTimeout(timer);
    }
  }, [message]);

  const handleChange = (e) => {
    setEmail(e.target.value);
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!email) {
      setMessage("Please enter your email.");
      setMessageType("error");
      return;
    }

    setLoading(true);
```

```
    // Simulated forgot password logic
    setTimeout(() => {
      setLoading(false);
      setMessage("If this email is registered, a password reset link has been sent.");
      setMessageType("success");
      setEmail("");
    }, 1500);
  };

  return (
    <div className="login-container">
      {showMessage && (
        <div className={`toast-message ${messageType}`}>
          <i
            className={`fa ${messageType === "error" ? "fa-times-circle" : "fa-check-circle"}`}
            style={{ marginRight: "8px" }}
          ></i>
          {message}
        </div>
      )}

      <div className="login-card">
        <Link to="/login" className="internal-back-arrow">

          <span>Back To Login</span>
        </Link>

        <h2 className="login-title">Forgot Password</h2>
        <form className="login-form" onSubmit={handleSubmit}>
          <div className="form-group floating-label-content">
            <input
              type="email"
              id="email"
              className="form-control floating-input"
              placeholder=" "
              value={email}
              onChange={handleChange}
              required
            />
            <label htmlFor="email" className="floating-label">
              Enter your email address
            </label>
          </div>

          <button type="submit" className="login-button" disabled={loading}>
            {loading ? (
              <>
                <span className="spinner-border spinner-border-sm me-2"></span> Sending...
              </>
            ) : (
              "Send Reset Request"
            )}
          </button>
        </form>
      </div>
    </div>
  );
};

export default ForgotPassword;
```

File: src\auth\login.css

```
/* Base reset and global styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
}
```

```
/* Login container with background */
.login-container {
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #f5f5f5;
  background-image: linear-gradient(135deg, rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.5)),
                    url('https://picsum.photos/1920/1080');
  background-size: cover;
  background-position: center;
  padding: 20px;
  position: relative;
}

/* Physiotherapist-specific background */
.login-container.physiotherapist {
  background-image: linear-gradient(135deg, rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.5)),
                    url('https://images.unsplash.com/photo-1576091160550-2173dba999ef?ixlib=rb-1.2.1&auto=formata
}

/* Dietician-specific background */
.login-container.dietician {
  background-image: linear-gradient(135deg, rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.5)),
                    url('https://images.unsplash.com/photo-1498837167922-ddd27525d352?ixlib=rb-1.2.1&auto=formata
}

/* Login card with improved styling */
.login-card {
  background: white;
  padding: 2.5rem;
  border-radius: 12px;
  width: 100%;
  max-width: 450px;
  box-shadow: 0 15px 30px rgba(0, 0, 0, 0.25);
  transition: transform 0.3s ease;
  animation: fadeIn 0.5s ease;
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(-20px); }
  to { opacity: 1; transform: translateY(0); }
}

.login-card:hover {
  transform: translateY(-5px);
}

/* Title styling with client color */
.login-title {
  text-align: center;
  margin-bottom: 2rem;
  color: #cc5500;
  font-size: 2rem;
  font-weight: 600;
  letter-spacing: 0.03em;
  position: relative;
  padding-bottom: 0.7rem;
}

.login-title::after {
  content: '';
  position: absolute;
  bottom: 0;
  left: 50%;
  width: 60px;
  height: 3px;
  background-color: #cc5500;
  transform: translateX(-50%);
}

/* Form group styling */
```

```
.form-group {
  margin-bottom: 1.5rem;
}

/* Floating label improvements */
.floating-label-content {
  position: relative;
  margin-bottom: 1.5rem;
}

.floating-input {
  width: 100%;
  padding: 12px 16px;
  border: 2px solid #ddd;
  border-radius: 8px;
  outline: none;

  font-size: 16px;
  background: #fff;
  color:--var(--text-primary, #333);
  transition: all 0.3s ease;
}

.floating-input:focus {
  border-color: #cc5500;
  box-shadow: 0 0 0 3px rgba(204, 85, 0, 0.15);
}

.floating-label {
  position: absolute;
  top: 50%;
  left: 12px;
  transform: translateY(-50%);
  background: white;
  padding: 0 8px;
  font-size: 14px;
  color: #777;
  transition: 0.2s ease all;
  pointer-events: none;
}

.floating-input:focus + .floating-label,
.floating-input:not(:placeholder-shown) + .floating-label {
  top: -1px;
  left: 12px;
  font-size: 12px;
  color: #cc5500;
  font-weight: 600;
}

/* Select dropdown styling */
select.floating-input {
  cursor: pointer;
  appearance: none;
  background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='12' height='12' view
  background-repeat: no-repeat;
  background-position: right 16px center;
  padding-right: 40px;
}

/* Button styling with client color */
.login-button {
  width: 100%;
  padding: 14px;
  background-color: #cc5500;
  border: 2px solid #cc5500;
  color: white;
  border-radius: 8px;
  font-weight: bold;
  font-size: 16px;
  letter-spacing: 0.03em;
  cursor: pointer;
  transition: 0.3s ease;
```

```
  box-shadow: 0 4px 6px rgba(204, 85, 0, 0.2);
}

.login-button:hover {
  background-color: #b54b00;
  border-color: #b54b00;
  transform: translateY(-2px);
  box-shadow: 0 6px 12px rgba(204, 85, 0, 0.25);
}

.login-button:disabled {
  background-color: #e6e6e6;
  border-color: #e6e6e6;
  color: #999;
  cursor: not-allowed;
  box-shadow: none;
}

/* Remember me and options styling */
.d-flex {
  display: flex;
}

.justify-content-between {
  justify-content: space-between;
}

.align-items-center {
  align-items: center;
}

.remember-label {
  font-size: 14px;
  color: #555;
  display: flex;
  align-items: center;
  cursor: pointer;
}

.remember-label input[type="checkbox"] {
  margin-right: 6px;
  accent-color: #cc5500;
  width: 16px;
  height: 16px;
}

.login-options {
  text-align: center;
  margin-top: 2rem;
  color: #555;
  font-size: 15px;
}

/* Link styling */
a {
  color: #cc5500;
  text-decoration: none;
  font-weight: 500;
  transition: all 0.3s ease;
}

a:hover {
  color: #993f00;
  text-decoration: underline;
}

/* Toast message styling */
.toast-message {
  position: fixed;
  top: 20px;
  left: 50%;
  transform: translateX(-50%);
  padding: 12px 24px;
```

```
  border-radius: 8px;
  font-weight: 500;
  z-index: 999;
  color: white;
  display: flex;
  align-items: center;
  box-shadow: 0px 8px 16px rgba(0, 0, 0, 0.2);
  animation: slideDown 0.3s ease forwards;
  max-width: 90%;
}

@keyframes slideDown {
  from {
    transform: translate(-50%, -20px);
    opacity: 0;
  }
  to {
    transform: translate(-50%, 0);
    opacity: 1;
  }
}

.toast-message.success {
  background-color: #2ecc71;
}

.toast-message.error {
  background-color: #e74c3c;
}

.toast-message i {
  margin-right: 10px;
  font-size: 18px;
}

/* Loading spinner */
.spinner-border {
  display: inline-block;
  width: 1rem;
  height: 1rem;
  border: 0.2em solid currentColor;
  border-right-color: transparent;
  border-radius: 50%;
  animation: spinner-border 0.75s linear infinite;
  margin-right: 8px;
}

@keyframes spinner-border {
  to { transform: rotate(360deg); }
}

.me-2 {
  margin-right: 0.5rem;
}

/* Responsive adjustments */
@media screen and (max-width: 576px) {
  .login-card {
    padding: 1.8rem;
    max-width: 95%;
  }

  .login-title {
    font-size: 1.75rem;
    margin-bottom: 1.5rem;
  }

  .floating-input {
    font-size: 15px;
    padding: 10px 14px;
  }

  .form-group.d-flex {
```

```
    flex-direction: column;
    align-items: flex-start;
    gap: 8px;
  }
}

@media screen and (max-width: 400px) {
  .login-card {
    padding: 1.5rem;
  }

  .login-title {
    font-size: 1.5rem;
  }

  .login-button {
    padding: 12px;
    font-size: 15px;
  }
}
```

File: src\auth\login.js

```
import React, { useState, useEffect } from "react";
import { useNavigate, Link } from "react-router-dom";
import "./login.css";

const Login = () => {
  const navigate = useNavigate();

  const savedEmail = localStorage.getItem("savedEmail") || "";
  const savedPassword = localStorage.getItem("savedPassword") || "";
  const savedRole = localStorage.getItem("savedRole") || "doctor"; // default role

  const [formData, setFormData] = useState({
    email: savedEmail,
    password: savedPassword,
    role: savedRole,
  });

  const [rememberMe, setRememberMe] = useState(savedEmail !== "");
  const [message, setMessage] = useState("");
  const [messageType, setMessageType] = useState("");
  const [showMessage, setShowMessage] = useState(false);
  const [loading, setLoading] = useState(false);
  const [lockoutTime, setLockoutTime] = useState(0);
  const [attempts, setAttempts] = useState(0);

  useEffect(() => {
    if (message) {
      setShowMessage(true);
      const timer = setTimeout(() => setShowMessage(false), 3000);
      return () => clearTimeout(timer);
    }
  }, [message]);

  useEffect(() => {
    if (lockoutTime > 0) {
      const interval = setInterval(() => {
        setLockoutTime((prev) => (prev > 0 ? prev - 1 : 0));
      }, 1000);
      return () => clearInterval(interval);
    }
  }, [lockoutTime]);

  const handleChange = (e) => {
    const { id, value } = e.target;
    setFormData((prev) => ({ ...prev, [id]: value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setMessage("");
```

```
try {
  const response = await fetch("http://localhost:8080/api/auth/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(formData),
  });

  const data = await response.json();

  if (data.error) {
    const nextAttempts = attempts + 1;
    setAttempts(nextAttempts);
    setLoading(false);

    if (nextAttempts >= 3) {
      setLockoutTime(30);
      setMessage(
        "Too many failed attempts. Please try again in 30 seconds."
      );
    } else {
      setMessage("Invalid credentials. Please try again.");
    }
    setMessageType("error");
    return;
  }

  // Successful login
  if (rememberMe) {
    localStorage.setItem("savedEmail", formData.email);
    localStorage.setItem("savedPassword", formData.password);
    localStorage.setItem("savedRole", formData.role);
  } else {
    localStorage.removeItem("savedEmail");
    localStorage.removeItem("savedPassword");
    localStorage.removeItem("savedRole");
  }

  localStorage.setItem("currentUser", JSON.stringify(data));
  setMessage("Login successful");
  setMessageType("success");
  setAttempts(0);

  setTimeout(() => {
    setLoading(false);
    switch (data.role.toLowerCase()) {
      case "physio":
        navigate("/profile");
        break;
      case "dietitian":
        navigate("/diet/diet_profile");
        break;
      case "doctor":
        navigate("/doctor/DoctorDashboardHome");
        break;
      case "masteradmin":
        navigate("/masteradmin");
        break;
      case "counselor":
        navigate("/counselor");
        break;
      case "phlebotomist":
        navigate("/phlebotomist");
        break; // ? THIS was missing
      default:
        navigate("/unauthorized");
    }
  });
} catch (error) {
  console.error("Login error:", error);
  setLoading(false);
  setMessage("Error connecting to server. Please try again later.");
  setMessageType("error");
```

```
  }
};

return (
  <div className="login-container">
    {showMessage && (
      <div className={`toast-message ${messageType}`}>
        <i
          className={`fa ${
            messageType === "error" ? "fa-times-circle" : "fa-check-circle"
          }`}
          style={{ marginRight: "8px" }}
        ></i>
        {message}
      </div>
    )}

    <div className="login-card">
      <h2 className="login-title">Login</h2>

      <form className="login-form" onSubmit={handleSubmit}>
        <div className="form-group floating-label-content">
          <input
            type="email"
            id="email"
            className="form-control floating-input"
            placeholder=" "
            value={formData.email}
            onChange={handleChange}
            required
          />
          <label htmlFor="email" className="floating-label">
            Email
          </label>
        </div>

        <div className="form-group floating-label-content">
          <input
            type="password"
            id="password"
            className="form-control floating-input"
            placeholder=" "
            value={formData.password}
            onChange={handleChange}
            required
          />
          <label htmlFor="password" className="floating-label">
            Password
          </label>
        </div>

        <div className="form-group floating-label-content">
          <select
            id="role"
            className="form-control floating-input"
            value={formData.role}
            onChange={handleChange}
            required
          >
            <option value="doctor">Doctor</option>
            <option value="physio">Physio</option>
            <option value="dietitian">Dietitian</option>
            <option value="counselor">Counselor</option>
            <option value="phlebotomist">Phlebotomist</option>
            <option value="masteradmin">Master Admin</option>
          </select>
          <label htmlFor="role" className="floating-label">
            Select Role
          </label>
        </div>

        <div className="form-group d-flex justify-content-between align-items-center">
          <label className="remember-label">
```

```
              <input
                type="checkbox"
                checked={rememberMe}
                onChange={() => setRememberMe(!rememberMe)}
              />
              Remember Me
            </label>
            <Link to="/forgotpassword">Forgot password?</Link>
          </div>

          <button
            type="submit"
            className="login-button"
            disabled={loading || lockoutTime > 0}
          >
            {loading ? (
              <>
                <span className="spinner-border spinner-border-sm me-2"></span>
                Signing in...
              </>
            ) : lockoutTime > 0 ? (
              `Try again in ${lockoutTime}s`
            ) : (
              "Login"
            )}
          </button>
        </form>
      </div>
    </div>
  );
};

export default Login;
```

File: src\auth\Login.test.js

```
// src/auth/Login.test.js

import React from "react";
import { render, screen, fireEvent, waitFor } from "@testing-library/react";
import { MemoryRouter } from "react-router-dom";
import Login from "./login";

// Mock useNavigate
const mockedNavigate = jest.fn();

jest.mock("react-router-dom", () => ({
  ...jest.requireActual("react-router-dom"),
  useNavigate: () => mockedNavigate,
}));

describe("Login Component", () => {
  const setup = () => {
    render(
      <MemoryRouter>
        <Login />
      </MemoryRouter>
    );
  };

  test("renders login form with fields", () => {
    setup();
    expect(screen.getByLabelText(/Email/i)).toBeInTheDocument();
    expect(screen.getByLabelText(/Password/i)).toBeInTheDocument();
    expect(screen.getByRole("button", { name: /Login/i })).toBeInTheDocument();
  });

  test("handles input changes", () => {
    setup();
    const emailInput = screen.getByLabelText(/Email/i);
    const passwordInput = screen.getByLabelText(/Password/i);

    fireEvent.change(emailInput, { target: { value: "test@example.com" } });
    fireEvent.change(passwordInput, { target: { value: "pass123" } });
```

```
    expect(emailInput.value).toBe("test@example.com");
    expect(passwordInput.value).toBe("pass123");
  });

  test("disables button during lockout", () => {
    setup();
    const loginButton = screen.getByRole("button", { name: /Login/i });
    fireEvent.click(loginButton);
    fireEvent.click(loginButton);
    fireEvent.click(loginButton);
    expect(loginButton).toBeDisabled();
  });

  test("shows error message on failed login", async () => {
    setup();
    const emailInput = screen.getByLabelText(/Email/i);
    const passwordInput = screen.getByLabelText(/Password/i);
    const loginButton = screen.getByRole("button", { name: /Login/i });

    fireEvent.change(emailInput, { target: { value: "wrong@example.com" } });
    fireEvent.change(passwordInput, { target: { value: "wrongpass" } });

    fireEvent.click(loginButton);

    await waitFor(() =>
      expect(screen.getByText(/Invalid credentials/i)).toBeInTheDocument()
    );
  });

  test("redirects user on successful login", async () => {
    render(
      <MemoryRouter>
        <Login />
      </MemoryRouter>
    );

    fireEvent.change(screen.getByLabelText(/email/i), {
      target: { value: "doctor@example.com" },
    });
    fireEvent.change(screen.getByLabelText(/password/i), {
      target: { value: "Password123" },
    });

    fireEvent.click(screen.getByRole("button", { name: /login/i }));

    await waitFor(() => {
      expect(mockedNavigate).toHaveBeenCalledWith("/doctor/doctor_profile");
    });
  });
});
```

File: src\components\GenericAppointments.js

```
// src/components/GenericAppointments.js
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const GenericAppointments = ({ role, title }) => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/${role}`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch((err) => console.error("Failed to load appointments:", err));
  }, [role]);

  const updateAppointment = (updated) => {
```

```
  setAppointments((prev) =>
    prev.map((a) => (a.id === updated.id ? updated : a))
  );
  fetch(`${API_BASE}/${updated.id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Failed to update appointment"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  let nextStatus = current.status === "Pending"
    ? "Approved"
    : current.status === "Approved"
    ? "Completed"
    : "Pending";

  updateAppointment({ ...current, status: nextStatus });
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(100000000 + Math.random() * 900000000)}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = () => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button className="btn btn-primary" onClick={() => {
          fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
            setAppointments((prev) => prev.filter((a) => a.id !== id))
          );
          toast.dismiss();
          toast.error("Appointment deleted.");
        }}>Confirm</button>
        <button className="btn btn-primary" onClick={() => {
          toast.dismiss();
          toast.info("Cancelled");
        }}>Cancel</button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>{title}</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
```

```
            <th>MRN No.</th>
            <th>Patient Name</th>
            <th>Type</th>
            <th>Date</th>
            <th>Time</th>
            <th>Notes</th>
            <th>Meeting Link</th>
            <th>Status</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {appointments.map((a) => (
            <tr key={a.id}>
              <td>{a.mrn}</td>
              <td>{a.patientName}</td>
              <td>
                <span className={`type-badge ${a.appointmentType === "Online" ? "badge-orange" : "badge-blue
                  {a.appointmentType}
                </span>
              </td>
              <td>{a.date}</td>
              <td>{a.time}</td>
              <td>
                <button className="btn btn-primary" onClick={() => setSelectedNote(a.notes)}>View</button>
              </td>
              <td>
                {a.appointmentType === "Online" ? (
                  <div style={{ display: "flex", gap: "6px", alignItems: "center" }}>
                    <input
                      value={a.meetingLink || ""}
                      onChange={(e) => handleMeetingLinkChange(a.id, e.target.value)}
                      className="search-input"
                      placeholder="Enter or Auto"
                      style={{ width: "180px" }}
                    />
                    <button className="btn btn-primary" onClick={() => handleGenerateLink(a.id)}>Auto</butto
                    <button className="btn btn-primary" style={{ backgroundColor: "#f44336" }}
                      onClick={() => handleMeetingLinkChange(a.id, "")}><XCircle size={16} /></button>
                  </div>
                ) : "-"}
              </td>
              <td>
                <span
                  className={`status-badge ${
                    a.status === "Completed" ? "badge-green"
                    : a.status === "Approved" ? "badge-blue"
                    : "badge-yellow"
                  }`}
                  style={{ cursor: "pointer" }}
                  onClick={() => handleStatusToggle(a.id)}
                >
                  {a.status}
                </span>
              </td>
              <td>
                <button className="btn btn-primary" onClick={() => handleDelete(a.id)}>Delete</button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  )}

  {selectedNote && (
    <div className="modal-overlay">
      <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
        <h2>Full Notes</h2>
        <p>{selectedNote}</p>
        <div className="center-btn">
          <center><button className="btn btn-primary" onClick={() => setSelectedNote(null)}>Close</button></
        </div>
```

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

export default GenericAppointments;
```

File: src\components\ProtectedRoute.js

```
import React from "react";
import { Navigate, useLocation } from "react-router-dom";

const ProtectedRoute = ({ children, allowedRoles = [] }) => {
  const location = useLocation();

  let user = null;

  try {
    user = JSON.parse(localStorage.getItem("currentUser"));
  } catch (err) {
    console.error("Invalid user data in localStorage:", err);
  }

  if (!user) {
    return <Navigate to="/login" state={{ from: location }} replace />;
  }

  // Normalize roles
  const userRole = (user.role || "").toLowerCase();
  const allowed = allowedRoles.map((role) => role.toLowerCase());

  if (!allowed.includes(userRole)) {
    return <Navigate to="/unauthorized" state={{ from: location }} replace />;
  }

  return children;
};

export default ProtectedRoute;
```

File: src\dashboard\counselor\counselor.css

```
.form-container {
  max-width: 500px;
  margin: 2rem auto;
  background: #fff;
  padding: 1.5rem 2rem;
  border: 1px solid #cc5500;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.counselor-form .form-group {
  margin-bottom: 1rem;
}

.input-field {
  width: 100%;
  padding: 0.6rem;
  font-size: 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
}
.search-input {
  padding: 0.6rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  width: 250px;
  max-width: 200%;
  font-size: 1rem;
}
.dark-mode-form {
  background-color: #1e1e1e;
```

```
  color: #f5f5f5;
  border-color: #cc5500;
}

.dark-mode-form .input-field {
  background-color: #2c2c2c;
  color: #f5f5f5;
  border: 1px solid #444;
}

.dark-mode-form textarea.input-field {
  background-color: #2c2c2c;
  color: #f5f5f5;
}
```

File: src\dashboard\counselor\CounselorCompletedAppointments.js

```
import React, { useEffect, useState } from "react";
import { Repeat, Edit3 } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";

const CounselorCompletedAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [search, setSearch] = useState("");

  const [reassigningAppt, setReassigningAppt] = useState(null);
  const [reassignDate, setReassignDate] = useState("");
  const [reassignTime, setReassignTime] = useState("");

  const [editingNoteAppt, setEditingNoteAppt] = useState(null);
  const [editedNote, setEditedNote] = useState("");

  const API_URL = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_URL}/role/counselor/completed`)
      .then((res) => res.json())
      .then(setAppointments)
      .catch(() => toast.error("Failed to load appointments"));
  }, []);

  const handleSearchChange = (e) => {
    setSearch(e.target.value.toLowerCase());
  };

  const filteredAppointments = appointments.filter(
    (a) =>
      a.patientName?.toLowerCase().includes(search) ||
      a.mrn?.toLowerCase().includes(search)
  );

  const handleReassign = (appt) => {
    setReassigningAppt(appt);
    setReassignDate("");
    setReassignTime("");
  };

  const confirmReassign = () => {
    if (!reassignDate || !reassignTime) {
      toast.error("Please select both date and time.");
      return;
    }

    const newAppt = {
      patientName: reassigningAppt.patientName,
      mrn: reassigningAppt.mrn,
      appointmentType: reassigningAppt.appointmentType,
      date: reassignDate,
      time: reassignTime,
      status: "Pending",
      notes: reassigningAppt.notes || "",
      role: "counselor",
```

```
    meetingLink: "",
  };

  fetch(API_URL, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(newAppt),
  })
    .then(() => {
      toast.success("Appointment reassigned");
      setReassigningAppt(null);
    })
    .catch(() => toast.error("Failed to reassign appointment"));
};

const handleEditNote = (appt) => {
  setEditingNoteAppt(appt);
  setEditedNote(appt.notes || "");
};

const confirmEditNote = () => {
  const updated = { ...editingNoteAppt, notes: editedNote };
  fetch(`${API_URL}/${editingNoteAppt.id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updated),
  })
    .then(() => {
      toast.success("Remarks updated");
      setAppointments((prev) =>
        prev.map((a) => (a.id === updated.id ? updated : a))
      );
      setEditingNoteAppt(null);
      setEditedNote("");
    })
    .catch(() => toast.error("Failed to update remarks"));
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Completed Appointments</h1>

    <div style={{ marginBottom: "1rem" }}>
      <input
        type="text"
        placeholder="Search by name or MRN..."
        className="search-input"
        value={search}
        onChange={handleSearchChange}
      />
    </div>

    {filteredAppointments.length === 0 ? (
      <p>No completed appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="user-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Date</th>
              <th>Time</th>
              <th>Remarks</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {filteredAppointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn}</td>
                <td>{a.patientName}</td>
```

```
            <td>{a.date}</td>
            <td>{a.time}</td>
            <td style={{ textAlign: "center" }}>
              <button
                className="btn btn-primary"
                onClick={() => handleEditNote(a)}
                title="Edit Remarks"
              >
                <Edit3 size={14} />
              </button>
            </td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => handleReassign(a)}
                  title="Reassign"
                >
                  <Repeat size={16} />
                </button>
              </center>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{reassigningAppt && (
  <div className="modal-overlay">
    <div
      className="modalsc-box"
      style={{
        border: "1px solid #cc5500",
        padding: "1rem",
        borderRadius: "8px",
      }}
    >
      <h2>Reassign Appointment</h2>
      <p>
        Patient: <strong>{reassigningAppt.patientName}</strong>
      </p>
      <div className="form-group">
        <label>Date:</label>
        <input
          className="search-input"
          type="date"
          value={reassignDate}
          onChange={(e) => setReassignDate(e.target.value)}
        />
      </div>
      <div className="form-group">
        <label>Time:</label>
        <input
          className="search-input"
          type="time"
          value={reassignTime}
          onChange={(e) => setReassignTime(e.target.value)}
        />
      </div>
      <div className="center-btn" style={{ marginTop: "1rem" }}>
        <center>
          <button className="btn btn-primary" onClick={confirmReassign}>
            Confirm
          </button>
          <button
            className="btn btn-primary"
            onClick={() => setReassigningAppt(null)}
          >
            Cancel
          </button>
        </center>
```

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

      {editingNoteAppt && (
        <div className="modal-overlay">
          <div
            className="modalsc-box"
            style={{
              border: "1px solid #cc5500",
              padding: "1rem",
              borderRadius: "8px",
            }}
          >
            <h2>Edit Remarks</h2>
            <p>
              Patient: <strong>{editingNoteAppt.patientName}</strong>
            </p>
            <div className="form-group">
              <label>Remarks:</label>
              <textarea
                className="search-input"
                value={editedNote}
                onChange={(e) => setEditedNote(e.target.value)}
                rows={4}
                style={{ width: "100%", fontSize: "14px" }}
              />
            </div>
            <div className="center-btn" style={{ marginTop: "1rem" }}>
              <center>
                <button className="btn btn-primary" onClick={confirmEditNote}>
                  Save
                </button>
                <button
                  className="btn btn-primary"
                  onClick={() => setEditingNoteAppt(null)}
                >
                  Cancel
                </button>
              </center>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default CounselorCompletedAppointments;
```

File: src\dashboard\counselor\CounselorDashboard.js

```
import React, { useEffect, useState } from "react";
import {
  BarChart2,
  CheckCircle,
  Clock,
  FileText,
  PlusCircle,
  AlertCircle,
  User,
  Edit3,
  Trash,
  ImagePlus,
} from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../ThemeProvider";
import { useNavigate } from "react-router-dom";
import {
  PieChart,
  Pie,
  Cell,
  Tooltip,
```

```
  ResponsiveContainer,
  Bar,
  XAxis,
  YAxis,
  CartesianGrid,
  Legend,
  BarChart,
} from "recharts";
import "react-toastify/dist/ReactToastify.css";

const CounselorDashboard = () => {
  const { theme } = useTheme();
  const navigate = useNavigate();

  const API_URL = "http://localhost:5000/counselorAppointments";
  const [appointments, setAppointments] = useState([]);
  const [motivationalQuote, setMotivationalQuote] = useState("");
  const [selectedNote, setSelectedNote] = useState(null);
  const [localNotes, setLocalNotes] = useState(() =>
    JSON.parse(localStorage.getItem("counselorNotes") || "[]")
  );
  const [newNote, setNewNote] = useState("");
  const [showNoteModal, setShowNoteModal] = useState(false);
  const [editingPic, setEditingPic] = useState(false);

  const quotes = [
    "A good counselor listens with their heart.",
    "You are the calm in someone else?s storm.",
    "Healing begins with empathy.",
    "Small steps every day lead to big change.",
  ];

  const currentUser = JSON.parse(localStorage.getItem("currentUser")) || {};
  const profilePic =
    localStorage.getItem("profilePic") || "https://i.pravatar.cc/150?img=47";
  const lastLogin = localStorage.getItem("lastLogin") || "Not Available";
  const lastLogout = localStorage.getItem("lastLogout") || "Not Available";

  useEffect(() => {
    fetch(API_URL)
      .then((res) => res.json())
      .then((data) => setAppointments(data));

    setMotivationalQuote(quotes[Math.floor(Math.random() * quotes.length)]);
  }, []);

  const today = new Date().toISOString().split("T")[0];
  const stats = {
    today: appointments.filter((a) => a.date === today).length,
    pending: appointments.filter((a) => a.status === "Pending").length,
    completed: appointments.filter((a) => a.status === "Completed").length,
    total: appointments.length,
  };

  const chartData = [
    { name: "Pending", value: stats.pending },
    { name: "Completed", value: stats.completed },
  ];

  const barChartData = [
    {
      name: "Appointments",
      Pending: stats.pending,
      Completed: stats.completed,
      Total: stats.total,
    },
  ];

  const COLORS = ["#cc5500", "#2ecc71"];

  const upcoming = appointments
    .filter((a) => new Date(a.date) >= new Date())
    .sort((a, b) => new Date(a.date) - new Date(b.date))
```

```
  .slice(0, 3);

const recentNotes = appointments
  .filter((a) => a.notes)
  .sort((a, b) => new Date(b.date) - new Date(a.date))
  .slice(0, 3);

const handleNoteAdd = () => {
  if (!newNote.trim()) return toast.error("Note cannot be empty.");
  const updated = [...localNotes, { id: Date.now(), text: newNote }];
  setLocalNotes(updated);
  localStorage.setItem("counselorNotes", JSON.stringify(updated));
  setNewNote("");
  setShowNoteModal(false);
  toast.success("Note added!");
};

const handleNoteDelete = (id) => {
  const updated = localNotes.filter((note) => note.id !== id);
  setLocalNotes(updated);
  localStorage.setItem("counselorNotes", JSON.stringify(updated));
  toast.success("Note deleted.");
};

const handleProfilePicChange = (e) => {
  const file = e.target.files[0];
  if (file) {
    const reader = new FileReader();
    reader.onloadend = () => {
      localStorage.setItem("profilePic", reader.result);
      window.location.reload();
    };
    reader.readAsDataURL(file);
  }
};

return (
  <div className={`dashboard-main ${theme}`}>
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Welcome, Counselor</h1>

    {/* Daily Quote */}
    <div
      className="card"
      style={{ textAlign: "center", marginBottom: "1.5rem" }}
    >
      <h3 className="card-header">? Daily Inspiration</h3>
      <p style={{ fontStyle: "italic", color: "var(--text-secondary)" }}>
        {motivationalQuote}
      </p>
    </div>

    {/* Profile & Chart */}
    <div className="row">
      <div className="col card" style={{ textAlign: "center" }}>
        <div style={{ position: "relative" }}>
          <img
            src={profilePic}
            alt="Counselor"
            style={{
              width: "100px",
              borderRadius: "50%",
              marginBottom: "0.5rem",
            }}
          />
          <label htmlFor="profileUpload" style={{ cursor: "pointer" }}>
            <ImagePlus size={16} />
          </label>
          <input
            id="profileUpload"
            type="file"
            accept="image/*"
            onChange={handleProfilePicChange}
```

```
        style={{ display: "none" }}
      />
    </div>
    <h3>{currentUser?.name || "Counselor Name"}</h3>
    <p style={{ color: "var(--text-secondary)" }}>
      {currentUser?.email || "email@example.com"}
    </p>
    <p>
      <User size={14} /> Role: {currentUser?.role}
    </p>
    <p>Last Login: {lastLogin}</p>
    <p>Last Logout: {lastLogout}</p>
  </div>

  <div className="col card">
    <h3 className="card-header">? Recent Remarks</h3>
    {recentNotes.length === 0 ? (
      <p>No recent notes.</p>
    ) : (
      <div className="table-responsive">
        <table className="table table-striped">
          <thead>
            <tr>
              <th>#</th>
              <th>Patient</th>
              <th>Date</th>
              <th>Action</th>
            </tr>
          </thead>
        </table>
        <div className="scrollable-remarks-table">
          <table className="table table-striped">
            <tbody>
              {recentNotes.map((a, index) => (
                <tr key={a.id}>
                  <td>{index + 1}</td>
                  <td>{a.patientName}</td>
                  <td>{a.date}</td>
                  <td>
                    <button
                      className="btn btn-primary"
                      onClick={() => setSelectedNote(a)}
                    >
                      View Note
                    </button>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    )}
  </div>
</div>

{/* Stats + Bar Chart */}

<div className="row">
  {/* Today?s Appointments */}
  <div className="col card">
    <h3 className="card-header">
      <Clock size={20} /> Today?s Appointments
    </h3>
    <div
      style={{
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
      }}
    >
      <p style={{ fontSize: "5rem" }}>{stats.today}</p>
      <ResponsiveContainer width={250} height={250}>
```

```
      <PieChart>
        <Pie
          data={[
            { name: "Today", value: stats.today },
            { name: "Others", value: stats.total - stats.today },
          ]}
          dataKey="value"
          outerRadius={100}
        >
          <Cell fill="#00bcd4" />
          <Cell fill="#444" />
        </Pie>
        <Tooltip />
        <Legend
          layout="horizontal"
          verticalAlign="bottom"
          align="center"
        />
      </PieChart>
    </ResponsiveContainer>
  </div>
</div>

{/* Pending */}
<div className="col card">
  <h3 className="card-header">
    <AlertCircle size={20} /> Pending
  </h3>
  <div
    style={{
      display: "flex",
      justifyContent: "space-between",
      alignItems: "center",
    }}
  >
    <p style={{ fontSize: "5rem" }}>{stats.pending}</p>
    <ResponsiveContainer width={250} height={250}>
      <PieChart>
        <Pie
          data={[
            { name: "Pending", value: stats.pending },
            { name: "Others", value: stats.total - stats.pending },
          ]}
          dataKey="value"
          outerRadius={100}
        >
          <Cell fill="#cc5500" />
          <Cell fill="#444" />
        </Pie>
        <Tooltip />
        <Legend
          layout="horizontal"
          verticalAlign="bottom"
          align="center"
        />
      </PieChart>
    </ResponsiveContainer>
  </div>
</div>

{/* Completed */}
<div className="col card">
  <h3 className="card-header">
    <CheckCircle size={20} /> Completed
  </h3>
  <div
    style={{
      display: "flex",
      justifyContent: "space-between",
      alignItems: "center",
    }}
  >
    <p style={{ fontSize: "5rem" }}>{stats.completed}</p>
```

```
      <ResponsiveContainer width={250} height={250}>
        <PieChart>
          <Pie
            data={[
              { name: "Completed", value: stats.completed },
              { name: "Others", value: stats.total - stats.completed },
            ]}
            dataKey="value"
            outerRadius={100}
          >
            <Cell fill="#cc5500" />
            <Cell fill="#444" />
          </Pie>
          <Tooltip />
          <Legend
            layout="horizontal"
            verticalAlign="bottom"
            align="center"
          />
        </PieChart>
      </ResponsiveContainer>
    </div>
  </div>

  {/* Total */}
  <div className="col card">
    <h3 className="card-header">
      <BarChart2 size={20} /> Total
    </h3>
    <div
      style={{
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
      }}
    >
      <p style={{ fontSize: "5rem" }}>{stats.total}</p>
      <ResponsiveContainer width={250} height={250}>
        <PieChart>
          <Pie
            data={[
              { name: "Total", value: stats.total },
              { name: "Balance", value: 10 }, // for visualization balance
            ]}
            dataKey="value"
            outerRadius={100}
          >
            <Cell fill="#cc5500" />
            <Cell fill="#444" />
          </Pie>
          <Tooltip />
          <Legend
            layout="horizontal"
            verticalAlign="bottom"
            align="center"
          />
        </PieChart>
      </ResponsiveContainer>
    </div>
  </div>
</div>

{/* Appointments and Remarks */}
<div className="row">
  <div className="col card">
    <h3 className="card-header">? Upcoming Appointments</h3>
    {upcoming.length === 0 ? (
      <p>No upcoming appointments.</p>
    ) : (
      <div className="table-responsive">
        <table className="table table-striped">
          <thead>
            <tr>
```

```
              <th>#</th>
              <th>Patient</th>
              <th>MRN</th>
              <th>Date</th>
              <th>Time</th>
              <th>Status</th>
            </tr>
          </thead>
          <tbody className="scrollable-appointments-table">
            {upcoming.map((a, index) => (
              <tr key={a.id}>
                <td>{index + 1}</td>
                <td>{a.patientName}</td>
                <td>{a.mrn}</td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <span
                    className={`badge ${
                      a.status === "Pending"
                        ? "bg-warning"
                        : a.status === "Approved"
                        ? "bg-primary"
                        : "bg-success"
                    }`}
                  >
                    {a.status}
                  </span>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}
  </div>
</div>

{/* Alerts and Personal Notes */}
<div className="row">
  <div className="col card">
    <h3 className="card-header">? Quick Actions</h3>
    <button
      className="btn btn-primary"
      onClick={() => setShowNoteModal(true)}
      style={{ marginBottom: "0.5rem" }}
    >
      <PlusCircle size={16} /> Add Personal Note
    </button>
    <button
      className="btn btn-primary"
      onClick={() => navigate("/counselor/CounselorPasswordRequest")}
    >
      <FileText size={16} /> Request Password Change
    </button>
  </div>

  <div className="col card">
    <h3 className="card-header">? My Personal Notes</h3>
    {localNotes.length === 0 ? (
      <p>No notes added yet.</p>
    ) : (
      <div className="table-responsive">
        <table className="table table-striped">
          <thead>
            <tr>
              <th>#</th>
              <th>Note</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
            {localNotes.map((note, index) => (
```

```
                    <tr key={note.id}>
                      <td>{index + 1}</td>
                      <td>{note.text}</td>
                      <td>
                        <button
                          className="btn btn-danger btn-sm"
                          onClick={() => handleNoteDelete(note.id)}
                        >
                          <Trash size={14} />
                        </button>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          )}
        </div>
      </div>

      {/* Modals */}
      {selectedNote && (
        <div className="modal-overlay">
          <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
            <h2>Note for {selectedNote.patientName}</h2>
            <p>
              <strong>Date:</strong> {selectedNote.date}
            </p>
            <p>{selectedNote.notes}</p>
            <center>
              {" "}
              <button
                className="btn btn-primary"
                onClick={() => setSelectedNote(null)}
              >
                Close
              </button>
            </center>
          </div>
        </div>
      )}

      {showNoteModal && (
        <div className="modal-overlay">
          <div className="modalsc-box">
            <h2>Add Personal Note</h2>
            <textarea

              className="search-input"
              rows="10"
              value={newNote}
              onChange={(e) => setNewNote(e.target.value)}
              placeholder="Write your task/reminder..."
            />
            <div style={{ marginTop: "10px" }}>
              <center><button className="btn btn-primary" onClick={handleNoteAdd}>
                Add
              </button>
              <button
                className="btn btn-primary"
                onClick={() => setShowNoteModal(false)}
              >
                Cancel
              </button></center>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default CounselorDashboard;
```

File: src\dashboard\counselor\CounselorsAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";

const CounselorsAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_URL = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_URL}/role/counselor`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch((err) => {
        console.error("Error fetching appointments:", err);
        toast.error("Failed to load appointments");
      });
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );
    fetch(`${API_URL}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(updated),
    }).catch(() => toast.error("Failed to update appointment"));
  };

  const handleStatusToggle = (id) => {
    const current = appointments.find((a) => a.id === id);
    let nextStatus =
      current.status === "Pending"
        ? "Approved"
        : current.status === "Approved"
        ? "Completed"
        : "Pending";

    updateAppointment({ ...current, status: nextStatus });
    toast.info(`Status updated to: ${nextStatus}`);
  };

  const handleMeetingLinkChange = (id, link) => {
    const updated = appointments.find((a) => a.id === id);
    updated.meetingLink = link;
    updateAppointment(updated);
  };

  const handleGenerateLink = (id) => {
    const dummy = `https://zoom.us/j/${Math.floor(
      100000000 + Math.random() * 900000000
    )}`;
    handleMeetingLinkChange(id, dummy);
  };

  const handleDelete = (id) => {
    const confirmToast = () => (
      <div>
        <p>Confirm delete?</p>
        <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
          <button
            className="btn btn-primary"
            onClick={() => {
              fetch(`${API_URL}/${id}`, { method: "DELETE" }).then(() =>
                setAppointments((prev) => prev.filter((a) => a.id !== id))
              );
              toast.dismiss();
              toast.error("Appointment deleted.");
```

```
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
            toast.dismiss();
            toast.info("Cancelled");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Counselor Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online"
                        ? "badge-orange"
                        : "badge-blue"
                    }`}
                  >
                    {a.appointmentType}
                  </span>
                </td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => setSelectedNote(a.notes)}
                  >
                    View
                  </button>
```

```
            </td>
            <td>
              {a.appointmentType === "Online" ? (
                <div style={{ display: "flex", gap: "5px", flexDirection: "column" }}>
                  <input
                    type="text"
                    value={a.meetingLink || ""}
                    onChange={(e) =>
                      handleMeetingLinkChange(a.id, e.target.value)
                    }
                    className="search-input"
                    placeholder="Enter or generate"
                    style={{ width: "180px" }}
                  />
                  <div style={{ display: "flex", gap: "5px" }}>
                    <button
                      className="btn btn-primary"
                      onClick={() => handleGenerateLink(a.id)}
                    >
                      Auto
                    </button>
                    <button
                      className="btn btn-primary"
                      style={{ backgroundColor: "#f44336", color: "#fff" }}
                      onClick={() => handleMeetingLinkChange(a.id, "")}
                    >
                      Clear
                    </button>
                  </div>
                </div>
              ) : (
                "-"
              )}
            </td>
            <td>
              <span
                className={`status-badge ${
                  a.status === "Completed"
                    ? "badge-green"
                    : a.status === "Approved"
                    ? "badge-blue"
                    : "badge-yellow"
                }`}
                onClick={() => handleStatusToggle(a.id)}
                style={{ cursor: "pointer" }}
              >
                {a.status}
              </span>
            </td>
            <td>
              <button
                className="btn btn-primary"
                onClick={() => handleDelete(a.id)}
              >
                Delete
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{selectedNote && (
  <div className="modal-overlay">
    <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
      <h2>Full Notes</h2>
      <p>{selectedNote}</p>
      <div className="center-btn">
        <center>
          <button
            className="btn btn-primary"
```

```
                onClick={() => setSelectedNote(null)}
              >
                Close
              </button>
            </center>
          </div>
        </div>
      </div>
    )}
  </div>
 );
};

export default CounselorsAppointments;
```

File: src\dashboard\counselor\components\counselor_footer.js

```
import React from 'react';
import '../../physio/components/Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} Admin Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\counselor\components\counselor_navbar.js

```
import React from "react";
import "../../physio/components/Navbar.css";
import { Moon, Sun } from "lucide-react";
import { useTheme } from "../../../ThemeProvider";

const DoctorNavbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="/counselor"> <div className="navbar-logo">Counselor</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default DoctorNavbar;
```

File: src\dashboard\counselor\components\counselor_sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "../../physio/components/Sidebar.css";
import {
  LogOut,
  Menu,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
  Calendar,
  ClipboardCheck,
  KeyRound,
  User2,
```

```
} from "lucide-react";

const CounselorSidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState("");
  const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
  const navigate = useNavigate();

  const toggleSidebar = () => setIsOpen(!isOpen);
  const toggleCollapse = () => setCollapsed(!collapsed);
  const handleResize = () => setIsOpen(window.innerWidth > 768);
  const closeSidebar = () => {
    if (isOpen && window.innerWidth <= 768) setIsOpen(false);
  };

  useEffect(() => {
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  useEffect(() => {
    document.body.classList.toggle(
      "sidebar-collapsed",
      collapsed && !isHovering
    );
  }, [collapsed, isHovering]);

  const navLinkClass = ({ isActive }) => (isActive ? "active" : "");

  const handleLogoutClick = (e) => {
    e.preventDefault();
    setShowLogoutConfirm(true);
  };

  const confirmLogout = () => {
    localStorage.clear();
    setShowLogoutConfirm(false);
    setToastMessage("Logged out successfully.");
    setShowToast(true);
    setTimeout(() => {
      setShowToast(false);
      navigate("/login");
    }, 2000);
  };

  const cancelLogout = () => setShowLogoutConfirm(false);

  return (
    <>
      <button className="sidebar-toggle" onClick={toggleSidebar}>
        <Menu size={24} />
      </button>

      {isOpen && window.innerWidth <= 768 && (
        <div className="sidebar-overlay show" onClick={closeSidebar}></div>
      )}

      {/* Logout Confirmation Modal */}
      {showLogoutConfirm && (
        <div className="modal-overlay">
          <div className="modal-container">
            <div className="modal-header">
              <center><h3>Logout Confirmation</h3></center>
              <button className="modal-close" onClick={cancelLogout}>
                <X size={18} />
              </button>
            </div>
            <div className="modal-body">
              <p>Are you sure you want to log out?</p>
            </div>
```

```
      <div className="modal-footer">
        <button className="btn btn-primary" onClick={confirmLogout}>
          Yes, Logout
        </button>
      </div>
    </div>
  </div>
)}

{/* Toast Message */}
{showToast && (
  <div className="toast-overlay">
    <div className="toast-container success">
      <div className="toast-content">
        <Check size={18} style={{ marginRight: "8px" }} />
        {toastMessage}
      </div>
    </div>
  </div>
)}

{/* Sidebar */}
<aside
  className={`sidebar ${isOpen ? "open" : ""} ${
    collapsed ? "collapsed" : ""
  } ${isHovering ? "hovering" : ""}`}
  onMouseEnter={() => collapsed && setIsHovering(true)}
  onMouseLeave={() => collapsed && setIsHovering(false)}
>
  <div className="sidebar-header">
    <div className="sidebar-profile">
      <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
      <a href="/counselor">
        <span className="sidebar-username">Counselor</span>
      </a>
    </div>
  </div>

  <nav className="sidebar-menu-wrapper">
    <ul className="sidebar-menu">
      <li>
        <NavLink to="/counselor/CounselorsAppointments" className={navLinkClass}>
          <Calendar size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Appointments</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/counselor/CounselorCompletedAppointments" className={navLinkClass}>
          <ClipboardCheck size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Completed</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/counselor/CounselorPasswordRequest" className={navLinkClass}>
          <KeyRound size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Password Request</span>
        </NavLink>
      </li>
      <li>
        <a
          href="#"
          onClick={handleLogoutClick}
          className={navLinkClass({ isActive: false })}
          style={{ cursor: "pointer", display: "flex", alignItems: "center" }}
        >
          <LogOut size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Log Out</span>
        </a>
      </li>
    </ul>

    <button className="collapse-toggle" onClick={toggleCollapse}>
      <div className="claude-toggle">
```

```
              {collapsed ? <ChevronRight size={16} /> : <ChevronLeft size={16} />}
            </div>
          </button>
        </nav>
      </aside>
    </>
  );
};

export default CounselorSidebar;
```

File: src\dashboard\counselor\pages\CounselorPasswordRequest.js

```
import React, { useState } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../../ThemeProvider"; // Adjust the import based on your project structure
import "react-toastify/dist/ReactToastify.css";
import "../../master_admin/master_admin.css"; // Ensure this path is correct

const CounselorPasswordRequest = () => {
  const { theme } = useTheme(); // Access the current theme
  const [formData, setFormData] = useState({
    counselorName: "",
    email: "",
    reason: "",
  });

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!formData.counselorName || !formData.email || !formData.reason) {
      toast.error("All fields are required!");
      return;
    }

    try {
      await axios.post("http://localhost:8080/api/passwordChangeRequests", {
        ...formData,
        status: "Pending", // Optional: backend defaults this anyway
        requestedAt: new Date().toISOString(),
      });

      toast.success("Request submitted to admin!");
      setFormData({ counselorName: "", email: "", reason: "" });
    } catch (err) {
      console.error("Submission error:", err);
      toast.error("Failed to submit request.");
    }
  };

  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Password Change Request</h1>
      <div className="card" style={{border:"1px solid #cc5500"}}>
        <form onSubmit={handleSubmit} className="form">
          <div className="form-group">
            <label htmlFor="counselorName">Counselor Name:</label>
            <input
              type="text"
              id="counselorName"
              name="counselorName"
              placeholder="Enter your name"
              value={formData.counselorName}
              onChange={handleChange}
              className="search-input"
              required
            />
          </div>
```

```
          <div className="form-group">
            <label htmlFor="email">Email ID:</label>
            <input
              type="email"
              id="email"
              placeholder="Enter your email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="search-input"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="reason">Reason for Change:</label>
            <input
              type="text"
              id="reason"
              placeholder="Enter reason for password change"
              name="reason"
              value={formData.reason}
              onChange={handleChange}
              className="search-input"
              required
            />
          </div>
          <div className="center-btn">
            <center>
              <button type="submit" className="btn btn-primary">
                Submit Request
              </button>
            </center>
          </div>
        </form>
      </div>
    </div>
  );
};

export default CounselorPasswordRequest;
```

File: src\dashboard\diet\DietitiansAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";

const DietitiansAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_URL = "http://localhost:5000/dietitianAppointments";

  useEffect(() => {
    fetch(API_URL)
      .then((res) => res.json())
      .then((data) => setAppointments(data));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );
    fetch(`${API_URL}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(updated),
    });
  };

  const handleStatusToggle = (id) => {
    const current = appointments.find((a) => a.id === id);
```

```
  let nextStatus;

  switch (current.status) {
    case "Pending":
      nextStatus = "Approved";
      break;
    case "Approved":
      nextStatus = "Completed";
      break;
    default:
      nextStatus = "Pending";
  }

  const updated = { ...current, status: nextStatus };
  updateAppointment(updated);
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(
    100000000 + Math.random() * 900000000
  )}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = ({ closeToast }) => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_URL}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Appointment deleted.");
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
```

```
<h1> Appointments</h1>

{appointments.length === 0 ? (
  <p>No appointments found.</p>
) : (
  <div className="table-responsive">
    <table className="appointments-table">
      <thead>
        <tr>
          <th>MRN No.</th>
          <th>Patient Name</th>
          <th>Type</th>
          <th>Date</th>
          <th>Time</th>
          <th>Notes</th>
          <th>Meeting Link</th>
          <th>Status</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {appointments.map((a) => (
          <tr key={a.id}>
            <td>{a.mrn || "N/A"}</td>
            <td>{a.patientName || "Unknown"}</td>
            <td>
              <span
                className={`type-badge ${
                  a.type === "Online" ? "badge-orange" : "badge-blue"
                }`}
              >
                {a.type}
              </span>
            </td>
            <td>{a.date}</td>
            <td>{a.time}</td>
            <td>
              <button
                className="btn btn-primary"
                onClick={() => setSelectedNote(a.notes)}
              >
                View
              </button>
            </td>
            <td>
              {a.type === "Online" ? (
                <div
                  style={{
                    display: "flex",
                    alignItems: "center",
                    gap: "8px",
                  }}
                >
                  <input
                    type="text"
                    value={a.meetingLink || ""}
                    onChange={(e) =>
                      handleMeetingLinkChange(a.id, e.target.value)
                    }
                    placeholder="Enter or generate"
                    className="search-input"
                    style={{
                      width: "180px",
                      padding: "5px 8px",
                      fontSize: "14px",
                      border: "1px solid #ccc",
                      borderRadius: "6px",
                    }}
                  />
                  <button
                    className="btn btn-primary"
                    onClick={() => handleGenerateLink(a.id)}
                  >
```

```
                        Auto
                      </button>
                      <button
                        className="btn btn-primary"
                        style={{ backgroundColor: "#f44336", color: "#fff" }}
                        onClick={() => handleMeetingLinkChange(a.id, "")}
                      >
                        <XCircle size={16} />
                      </button>
                    </div>
                  ) : (
                    "-"
                  )}
                </td>
                <td>
                  <span
                    className={`status-badge ${
                      a.status === "Completed"
                        ? "badge-green"
                        : a.status === "Approved"
                        ? "badge-blue"
                        : "badge-yellow"
                    }`}
                    onClick={() => handleStatusToggle(a.id)}
                    style={{ cursor: "pointer" }}
                  >
                    {a.status}
                  </span>
                </td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => handleDelete(a.id)}
                  >
                    Delete
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}

    {selectedNote && (
      <div className="modal-overlay">
        <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
          <h2>Full Notes</h2>
          <p>{selectedNote}</p>
          <div className="center-btn">
            <center><button
              className="btn btn-primary"
              onClick={() => setSelectedNote(null)}
            >
              Close
            </button></center>
          </div>
        </div>
      </div>
    )}
  </div>
 );
};

export default DietitiansAppointments;
```

File: src\dashboard\diet\DietMealPlanAssign.js

```
"use client";

import { useState, useRef, useEffect } from "react";
import { useLocation } from "react-router-dom";
import "../physio/assign.css";
import "../master_admin/master_admin.css";
```

```
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import jsPDF from "jspdf";
import autoTable from "jspdf-autotable";
import logo from "../../image/logo.png";
import { Link } from "react-router-dom";

const DietMealPlanAssign = () => {
  const [mrn, setMrn] = useState("");
  const [clientData, setClientData] = useState(null);
  const [showBmiPopup, setShowBmiPopup] = useState(false);
  const [showOptionPopup, setShowOptionPopup] = useState({
    index: null,
    visible: false,
  });
  const [showViewPopup, setShowViewPopup] = useState({
    visible: false,
    option: null,
  });
  const [confirmDelete, setConfirmDelete] = useState({
    visible: false,
    rowIndex: null,
    optIndex: null,
  });
  const [newOption, setNewOption] = useState({
    meal: "",
    ingredients: "",
    recipe: "",
    cookingVideo: "",
  });

  const [editIndex, setEditIndex] = useState({ row: null, opt: null });
  const [approvedDate, setApprovedDate] = useState("");
  const [pdfUrl, setPdfUrl] = useState(null);
  const [showPdfViewer, setShowPdfViewer] = useState(false);
  const [fromDateTime, setFromDateTime] = useState("");
  const [toDateTime, setToDateTime] = useState("");
  const [selectedMeals, setSelectedMeals] = useState([]);
  const [showHistoryCards, setShowHistoryCards] = useState(false);
  const [selectedPlanToView, setSelectedPlanToView] = useState(null);
  const [dietHistory, setDietHistory] = useState([]);
  const bmiCardRef = useRef(null);
  const [availableMeals, setAvailableMeals] = useState([]);
  const [dietaryGuidelines, setDietaryGuidelines] = useState([]);
  const [assignedPlans, setAssignedPlans] = useState([]);
  const [focusedCaloriesIndex, setFocusedCaloriesIndex] = useState(null);
  const [focusedProteinIndex, setFocusedProteinIndex] = useState(null);
  const [dietMode, setDietMode] = useState("");
  const [aiInjected, setAiInjected] = useState(false);

  const location = useLocation();
  const {
    aiGeneratedText = "",
    aiInitialData = {},
    mrn: stateMrn,
    clientData: stateClientData,
    fromDateTime: stateFrom,
    toDateTime: stateTo,
    bmiData: stateBmi,
    energyProteinDistribution: stateDist,
    selectedMeals: stateMeals,
  } = location.state || {};

  useEffect(() => {
    if (stateMrn) setMrn(stateMrn);
    if (stateClientData) setClientData(stateClientData);
    if (stateFrom) setFromDateTime(stateFrom);
    if (stateTo) setToDateTime(stateTo);
    if (stateBmi) setBmiData(stateBmi);
    if (stateDist) setEnergyProteinDistribution(stateDist);
    if (stateMeals) setSelectedMeals(stateMeals);
  }, []);
```

```
const [bmiData, setBmiData] = useState({
  bmiCategory: "",
  bmr: "",
  tdee: "",
  target: "",
  energy: "",
  protein: "",
  carbohydrate: "",
  fats: "",
});

const InputField = ({ label, name, value, onChange, readOnly = false }) => (
  <div className="input-field">
    <label className="input-label">{label}</label>
    <input
      type="text"
      name={name}
      value={value}
      onChange={onChange}
      readOnly={readOnly}
      className="form-control"
      style={{
        backgroundColor: readOnly ? "#f0f0f0" : "var(--bg-primary)",
        color: readOnly ? "#555" : undefined,
      }}
    />
  </div>
);

const DropdownField = ({ label, name, value, onChange, options }) => (
  <div className="input-field">
    <label className="input-label">{label}</label>
    <select
      name={name}
      value={value}
      onChange={onChange}
      className="form-control"
      style={{
        backgroundColor: "var(--bg-primary)",
        color: "var(--text-white)",
      }}
    >
      <option value="">-- Select --</option>
      {options.map((opt, i) => (
        <option key={i} value={opt.split(" ")[0]}>
          {opt}
        </option>
      ))}
    </select>
  </div>
);

useEffect(() => {
  if (
    !clientData?.height ||
    !clientData?.weight ||
    !clientData?.age ||
    !clientData?.gender
  )
    return;

  const heightInMeters = clientData.height / 100;
  const weight = clientData.weight;
  const age = clientData.age;
  const gender = clientData.gender.toLowerCase();

  // BMI
  const bmi = (weight / (heightInMeters * heightInMeters)).toFixed(2);

  // BMI Category
  let bmiCategory = "";
  if (bmi < 18.5) bmiCategory = "Underweight";
  else if (bmi < 24.9) bmiCategory = "Normal";
```

```
  else if (bmi < 29.9) bmiCategory = "Overweight";
  else bmiCategory = "Obese";

  // BMR
  let bmr;
  if (gender === "male") {
    bmr = (10 * weight + 6.25 * clientData.height - 5 * age + 5).toFixed(2);
  } else {
    bmr = (10 * weight + 6.25 * clientData.height - 5 * age - 161).toFixed(2);
  }

  // TDEE
  const activityMultiplier = Number.parseFloat(bmiData.tdee);
  let tdee = "";
  if (!isNaN(activityMultiplier)) {
    tdee = (bmr * activityMultiplier).toFixed(2);
  }

  setBmiData((prev) => ({
    ...prev,
    bmi,
    bmiCategory: prev.bmiCategory || bmiCategory,
    bmr,
    calculatedTdee: tdee || prev.calculatedTdee,
  }));
}, [clientData, bmiData.tdee]);

const [energyProteinDistribution, setEnergyProteinDistribution] = useState(
  []
);
const [showOptionGridPopup, setShowOptionGridPopup] = useState({
  visible: false,
  index: null,
});

const toggleMealEnable = (index) => {
  const updated = [...energyProteinDistribution];
  updated[index].enabled = !updated[index].enabled;
  setEnergyProteinDistribution(updated);
};

const handleSearch = async () => {
  if (!mrn.trim()) {
    toast.error("Please enter a valid MRN number");
    return;
  }

  try {
    const response = await fetch(
      `http://localhost:5000/mealclients?mrn=${mrn}`
    );
    const data = await response.json();

    if (data.length > 0) {
      const client = data[0];
      setClientData(client);

      const assignedRes = await fetch(
        `http://localhost:5000/assignedMeals?mrn=${mrn}`
      );
      const assignedData = await assignedRes.json();
      setAssignedPlans(assignedData);

      setTimeout(() => {
        if (bmiCardRef.current) {
          bmiCardRef.current.scrollIntoView({
            behavior: "smooth",
            block: "center",
          });
          bmiCardRef.current.classList.add("bmi-highlight");
          setTimeout(() => {
            bmiCardRef.current.classList.remove("bmi-highlight");
          }, 3000);
```

```
        }
      }, 400);

      setBmiData({
        bmiCategory: "",
        bmr: "",
        tdee: "",
        target: "",
        energy: "",
        protein: "",
        carbohydrate: "",
        fats: "",
      });

      setEnergyProteinDistribution((prev) =>
        prev.map((item) => ({
          ...item,
          calories: "",
          protein: "",
          options: [],
        }))
      );
    } else {
      toast.error("No client found with this MRN");
      setClientData(null);
      setAssignedPlans([]);
      setBmiData({});
    }
  } catch (error) {
    toast.error("Failed to fetch data");
    console.error("Search error:", error);
  }
};

useEffect(() => {
  if (!aiGeneratedText || aiInjected) return;

  setDietMode("ai");
  setAiInjected(true);

  const breakfastIndex = energyProteinDistribution.findIndex(
    (row) => row.mealName.toLowerCase() === "breakfast"
  );

  if (breakfastIndex !== -1) {
    const updated = [...energyProteinDistribution];
    const alreadyHasAI = updated[breakfastIndex].options.some(
      (opt) => opt.meal === "AI Suggested"
    );

    if (!alreadyHasAI) {
      updated[breakfastIndex].options.push({
        meal: "AI Suggested",
        ingredients: "-",
        recipe: aiGeneratedText,
        cookingVideo: "",
      });
      setEnergyProteinDistribution(updated);
      toast.success("AI plan added to Breakfast automatically");
    }
  } else {
    setSelectedMeals((prev) => [...prev, "Breakfast"]);
    setEnergyProteinDistribution((prev) => [
      ...prev,
      {
        mealName: "Breakfast",
        mealTime: "08:00",
        calories: "400",
        protein: "20",
        enabled: true,
        options: [
          {
            meal: "AI Suggested",
```

```
            ingredients: "-",
            recipe: aiGeneratedText,
            cookingVideo: "",
          },
        ],
      },
    ]);
    toast.success("AI plan injected as Breakfast meal");
  }

  setTimeout(() => {
    const section = document.getElementById("meal-plan-section");
    if (section) {
      section.scrollIntoView({ behavior: "smooth" });
    }
  }, 300);
}, [aiGeneratedText, energyProteinDistribution, aiInjected]);

const handleSwitchToAI = () => {
  const hasManualData = energyProteinDistribution.length > 0;

  if (hasManualData) {
    const confirmSwitch = window.confirm(
      "You've already entered a manual diet plan. Switching to AI mode will discard it. Continue?"
    );
    if (!confirmSwitch) return;

    setSelectedMeals([]);
    setEnergyProteinDistribution([]);
  }

  setDietMode("ai");
};

const ConfirmSwitchToast = ({ message, onConfirm, onCancel }) => (
  <div style={{ padding: "0.5rem" }}>
    <p style={{ marginBottom: "0.5rem" }}>{message}</p>
    <div
      style={{ display: "flex", justifyContent: "flex-end", gap: "0.5rem" }}
    >
      <button
        onClick={() => {
          toast.dismiss();
          onConfirm();
        }}
        style={{
          background: "#16a34a",
          color: "#fff",
          border: "none",
          padding: "0.3rem 0.6rem",
          borderRadius: "5px",
        }}
      >
        Yes
      </button>
      <button
        onClick={() => {
          toast.dismiss();
          if (onCancel) onCancel();
        }}
        style={{
          background: "#dc2626",
          color: "#fff",
          border: "none",
          padding: "0.3rem 0.6rem",
          borderRadius: "5px",
        }}
      >
        No
      </button>
    </div>
  </div>
);
```

```
const handleSwitchToManual = () => {
  const hasAI = aiGeneratedText || dietMode === "ai";

  if (hasAI) {
    const confirmSwitch = window.confirm(
      "You're currently using an AI-generated plan. Switching to Manual mode will discard it. Continue?"
    );
    if (!confirmSwitch) return;

    setEnergyProteinDistribution([]);
    setSelectedMeals([]);
    setAiInjected(false);
    location.state = {};
  }

  setDietMode("manual");
};

const handleBmiChange = (e) => {
  let { name, value } = e.target;

  const numericValue = value.replace(/[^0-9.]/g, "");

  if (name === "energy" || name === "target") {
    value = `${numericValue} kcal/day`;
  } else if (
    name === "protein" ||
    name === "carbohydrate" ||
    name === "fats"
  ) {
    value = `${numericValue} g`;
  }

  setBmiData((prev) => ({ ...prev, [name]: value }));
};

const handleEnergyProteinChange = (index, field, value) => {
  const updated = [...energyProteinDistribution];
  updated[index][field] = value;
  setEnergyProteinDistribution(updated);
};

const handleAddOptionClick = (index) => {
  setShowOptionPopup({ index, visible: true });
  setNewOption({ meal: "", ingredients: "", recipe: "", cookingVideo: "" });
  setEditIndex({ row: null, opt: null });
};

const handleOptionSubmit = () => {
  const updated = [...energyProteinDistribution];
  const rowIndex = showOptionPopup.index;

  if (editIndex.row !== null && editIndex.opt !== null) {
    updated[editIndex.row].options[editIndex.opt] = { ...newOption };
    toast.success("Meal option updated!");
  } else {
    updated[rowIndex].options.push({ ...newOption });
    toast.success("Meal option added!");
  }

  setEnergyProteinDistribution(updated);
  setShowOptionPopup({ index: null, visible: false });
  setEditIndex({ row: null, opt: null });
};

const handleOptionChange = (e) => {
  const { name, value } = e.target;
  setNewOption((prev) => ({ ...prev, [name]: value }));
};

const handleOptionView = (opt) => {
  setShowViewPopup({ visible: true, option: opt });
```

```
};

const handleOptionEdit = (row, optIndex) => {
  setNewOption({ ...energyProteinDistribution[row].options[optIndex] });
  setShowOptionPopup({ index: row, visible: true });
  setEditIndex({ row, opt: optIndex });
};

const handleDeleteConfirm = (rowIndex, optIndex) => {
  setConfirmDelete({ visible: true, rowIndex, optIndex });
};

const confirmDeleteOption = () => {
  const updated = [...energyProteinDistribution];
  updated[confirmDelete.rowIndex].options.splice(confirmDelete.optIndex, 1);
  setEnergyProteinDistribution(updated);
  setConfirmDelete({ visible: false, rowIndex: null, optIndex: null });
  toast.info("Option deleted");
};

const isBmiFilled = () => {
  return Object.values(bmiData).every((val) => val.trim() !== "");
};

const handleCompleteAssignment = async () => {
  const isBmiFilled = Object.values(bmiData).every(
    (val) => val.trim() !== ""
  );
  if (!isBmiFilled) {
    toast.error("Please complete the Modified BMI section.");

    if (bmiCardRef?.current) {
      bmiCardRef.current.scrollIntoView({
        behavior: "smooth",
        block: "center",
      });
      bmiCardRef.current.classList.add("bmi-highlight");
      setTimeout(() => {
        bmiCardRef.current.classList.remove("bmi-highlight");
      }, 3000);
    }

    const btn = document.querySelector("#complete-btn");
    if (btn) {
      btn.classList.add("shake");
      setTimeout(() => btn.classList.remove("shake"), 500);
    }

    return;
  }

  if (!fromDateTime || !toDateTime || selectedMeals.length === 0) {
    toast.error("Please fill date range and select at least one meal");
    return;
  }

  const today = new Date().toISOString().split("T")[0];
  if (fromDateTime < today) {
    toast.error("From date cannot be in the past.");
    return;
  }

  if (toDateTime < fromDateTime) {
    toast.error("To date cannot be earlier than From date.");
    return;
  }

  const selectedMealData = energyProteinDistribution.filter((row) =>
    selectedMeals.includes(row.mealName)
  );

  const isIncomplete = selectedMealData.some((meal) => {
    return (
```

```
      !meal.calories.trim() ||
      !meal.protein.trim() ||
      meal.options.length === 0
    );
  });

  if (isIncomplete) {
    toast.error(
      "Please fill calories, protein, and add at least one option for each selected meal."
    );
    return;
  }

  const payload = {
    mrn,
    fromDateTime,
    toDateTime,
    bmiData,
    meals: selectedMealData,
  };

  try {
    await fetch("http://localhost:5000/assignedMeals", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(payload),
    });
    toast.success("Diet Plan Assigned and Saved!");
  } catch (error) {
    console.error(error);
    toast.error("Failed to save data.");
  }

  setFromDateTime("");
  setToDateTime("");
  setSelectedMeals([]);
  setEnergyProteinDistribution([]);
  setClientData(null);
};

const handleBmiSubmit = () => {
  const isAnyEmpty = Object.values(bmiData).some((val) => !val.trim());

  if (isAnyEmpty) {
    toast.error("All BMI fields are required.");
    return;
  }

  console.log("Submitted BMI Data:", bmiData);
  toast.success("MODIFIED BMI data saved successfully!");
  setShowBmiPopup(false);
};

const generateDietPDF = () => {
  const pdf = new jsPDF("p", "mm", "a4");
  pdf.addImage(logo, "PNG", 14, 10, 50, 20);

  pdf.setFontSize(18);
  pdf.setFont("helvetica", "bold");
  pdf.text("Personalized - Meal Plan", 105, 20, { align: "center" });

  pdf.setFontSize(14);
  pdf.text("Client Profile", 14, 35);

  const col1X = 14;
  const col2X = 105;
  let y = 45;

  pdf.setFontSize(12);
  pdf.setFont("helvetica", "normal");
  pdf.text(`Name: ${clientData?.name || "-"}`, col1X, y);
  pdf.text(`Age: ${clientData?.age || "-"}`, col2X, y);
  y += 8;
```

```
pdf.text(`Height: ${clientData?.height || "-"} cm`, col1X, y);
pdf.text(`Weight: ${clientData?.weight || "-"} kg`, col2X, y);
y += 8;
pdf.text(`BMI: ${clientData?.bmi || "-"}`, col1X, y);
pdf.text(`Ideal Body Weight: ${bmiData?.ibw || "-"}`, col2X, y);
y += 14;

pdf.setFontSize(14);
pdf.setFont("helvetica", "bold");
pdf.text("Personalized Nutritional Blueprint", 14, y);
y += 4;

autoTable(pdf, {
  startY: y + 2,
  head: [
    [
      "BMI Category",
      "BMR",
      "TDEE",
      "Target",
      "Energy",
      "Protein",
      "Carbohydrate",
      "Fats",
    ],
  ],
  body: [
    [
      bmiData?.bmiCategory || "-",
      bmiData?.bmr || "-",
      bmiData?.tdee || "-",
      bmiData?.target || "-",
      bmiData?.energy || "-",
      bmiData?.protein || "-",
      bmiData?.carbohydrate || "-",
      bmiData?.fats || "-",
    ],
  ],
  theme: "striped",
  styles: { fontSize: 10, halign: "center" },
  headStyles: { fillColor: [204, 85, 0], textColor: "#fff" },
});

let distY = pdf.lastAutoTable.finalY + 10;
pdf.setFontSize(14);
pdf.setFont("helvetica", "bold");
pdf.text("Daily Energy & Protein Distribution", 14, distY);
distY += 4;

const distRows = energyProteinDistribution
  .filter((row) => selectedMeals.includes(row.mealName))
  .map((row) => [
    `${row.mealName} (${row.mealTime || "-"})`,
    row.calories || "-",
    row.protein || "-",
  ]);

autoTable(pdf, {
  startY: distY + 2,
  head: [["Meal Time", "Target Calories", "Target Protein"]],
  body: distRows,
  theme: "grid",
  styles: { fontSize: 10 },
  headStyles: { fillColor: [204, 85, 0], textColor: "#fff" },
});

let optionY = pdf.lastAutoTable.finalY + 10;
pdf.setFontSize(14);
pdf.setFont("helvetica", "bold");
pdf.text("Detailed Meal Plan", 14, optionY);
optionY += 4;

const optionTableRows = [];
```

```
  energyProteinDistribution
    .filter((row) => selectedMeals.includes(row.mealName))
    .forEach((meal) => {
      const mealTime = `${meal.mealName} (${meal.mealTime || "-"})`;

      meal.options.forEach((opt, index) => {
        optionTableRows.push([
          mealTime,
          `Option ${index + 1}`,
          opt.meal || "-",
          opt.ingredients || "-",
          opt.recipe || "-",
          opt.cookingVideo || "-",
        ]);
      });
    });

  autoTable(pdf, {
    startY: optionY + 2,
    head: [["Meal Time", "Option", "Meal", "Ingredients", "Recipe", "Video"]],
    body: optionTableRows,
    theme: "striped",
    styles: { fontSize: 9, cellPadding: 2, valign: "top" },
    headStyles: { fillColor: [204, 85, 0], textColor: "#fff" },
    columnStyles: {
      0: { cellWidth: 35 },
      1: { cellWidth: 20 },
      2: { cellWidth: 35 },
      3: { cellWidth: 50 },
      4: { cellWidth: 50 },
      5: { cellWidth: 50 },
    },
  });

  let guideY = pdf.lastAutoTable.finalY + 10;

  pdf.setFontSize(14);
  pdf.setFont("helvetica", "bold");
  pdf.text("Dietary Guidelines", 14, guideY);
  guideY += 4;

  autoTable(pdf, {
    startY: guideY + 2,
    head: [["Guideline", "Description"]],
    body: dietaryGuidelines.map((item) => [item.type, item.description]),
    theme: "striped",
    styles: { fontSize: 10, valign: "top" },
    headStyles: { fillColor: [204, 85, 0], textColor: "#fff" },
    columnStyles: {
      0: { cellWidth: 50 },
      1: { cellWidth: 130 },
    },
  });

  pdf.save(`DietPlan_${mrn}.pdf`);
  toast.success("PDF successfully generated and downloaded!");
};

useEffect(() => {
  const fetchInitialData = async () => {
    try {
      const [mealsRes, guidelinesRes] = await Promise.all([
        fetch("http://localhost:3001/mealTimes"),
        fetch("http://localhost:3001/dietGuidelines"),
      ]);
      const meals = await mealsRes.json();
      const guidelines = await guidelinesRes.json();
      setAvailableMeals(meals);
      setDietaryGuidelines(guidelines);
    } catch (err) {
      toast.error("Failed to load initial data");
    }
```

```
  };

  fetchInitialData();
}, []);

return (
  <div className="dashboard-main">
    <style jsx>{`
      .dashboard-main {
        padding: 1rem;
        width: 100%;
        overflow-x: hidden;
      }

      .card {
        background: var(--bg-primary);
        border: 1px solid #cc5500;
        border-radius: 8px;
        padding: 1rem;
        margin-bottom: 1rem;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
      }

      .card-header {
        font-size: 1.25rem;
        font-weight: bold;
        color: var(--text-primary);
        margin-bottom: 1rem;
        padding-bottom: 0.5rem;
        border-bottom: 2px solid #cc5500;
      }

      .search-containers {
        display: flex;
        align-items: center;
        gap: 0.5rem;
        padding: 0.5rem 0;
      }

      .search-input {
        flex: 1;
        min-width: 200px;
        padding: 0.75rem;
        font-size: 1rem;
        border: 1px solid #cc5500;
        border-radius: 4px;
        background: var(
          --bg-primary,
          #1e1e1e
        ); /* fallback if var not defined */
        color: var(--text-white, #ffffff);
      }

      .search-button {
        padding: 0.75rem 1rem;
        font-size: 1rem;
        border: none;
        border-radius: 4px;
        background-color: #cc5500;
        color: white;
        cursor: pointer;
      }

      .btn.btn-primary {
        padding: 0.75rem 1rem;
        background-color: #cc5500;
        color: #fff;
        border: none;
        border-radius: 4px;
        cursor: pointer;
      }

      .btn {
```

```
  padding: 0.75rem 1.5rem;
  font-size: 1rem;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  transition: all 0.2s ease;
  white-space: nowrap;
  min-width: fit-content;
}

.btn-primary {
  background: linear-gradient(135deg, #cc5500, #e06600);
  color: white;
  font-weight: bold;
}

.btn-primary:hover {
  background: linear-gradient(135deg, #b84d00, #cc5500);
  transform: translateY(-1px);
}

.btn-secondary {
  background: #6c757d;
  color: white;
}

.btn-secondary:hover {
  background: #5a6268;
}

.bmi-card {
  text-align: center;
  padding: 2rem 1rem;
}

.bmi-highlight {
  animation: highlight 3s ease-in-out;
}

@keyframes highlight {
  0%,
  100% {
    background-color: var(--bg-primary);
  }
  50% {
    background-color: rgba(204, 85, 0, 0.1);
  }
}

.date-range-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 1rem;
  margin-bottom: 1rem;
}

.input-field {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
}

.input-label {
  font-weight: 600;
  color: var(--text-white);
}

.form-control {
  padding: 0.75rem;
  border: 1px solid #cc5500;
  border-radius: 4px;
  font-size: 1rem;
  background: var(--bg-primary);
```

```
  color: var(--text-white);
  width: 100%;
}

.form-control:focus {
  outline: none;
  border-color: #cc5500;
  box-shadow: 0 0 0 3px rgba(204, 85, 0, 0.1);
}

.diet-mode-selector {
  display: flex;
  gap: 1rem;
  justify-content: center;
  flex-wrap: wrap;
  margin: 1rem 0;
}

.meal-selector {
  margin-bottom: 1rem;
}

.selected-meals {
  display: flex;
  flex-wrap: wrap;
  gap: 0.5rem;
  margin-top: 0.5rem;
}

.selected-meal-btn {
  position: relative;
  padding: 0.5rem 1rem;
  background: transparent;
  border: 1px solid #cc5500;
  border-radius: 20px;
  color: var(--text-white);
  font-weight: bold;
}

.remove-meal {
  position: absolute;
  top: -6px;
  right: -6px;
  background: #cc5500;
  color: white;
  border-radius: 50%;
  width: 20px;
  height: 20px;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  font-size: 12px;
  font-weight: bold;
}

.meal-table-container {
  overflow-x: auto;
  margin: 1rem 0;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.meal-table {
  width: 100%;
  min-width: 800px;
  border-collapse: collapse;
  background: var(--bg-primary);
}

.meal-table th {
  background: linear-gradient(135deg, #cc5500, #e06600);
  color: white;
```

```
  padding: 1rem 0.5rem;
  text-align: center;
  font-weight: 600;
  position: sticky;
  top: 0;
  z-index: 10;
}

.meal-table td {
  padding: 1rem 0.5rem;
  text-align: center;
  border-bottom: 1px solid #cc5500;
  vertical-align: middle;
}

.meal-table tr:hover {
  background-color: rgba(204, 85, 0, 0.05);
}

.meal-input {
  width: 100%;
  max-width: 120px;
  padding: 0.5rem;
  border: 2px solid #cc5500;
  border-radius: 4px;
  font-size: 0.9rem;
  background: white;
  color: #333;
}

.meal-input:focus {
  outline: none;
  border-color: #cc5500;
  box-shadow: 0 0 0 3px rgba(204, 85, 0, 0.1);
}

.meal-cards {
  display: none;
}

.popup-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;
  background: rgba(0, 0, 0, 0.5);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 9999;
  padding: 1rem;
}

.popup-content {
  background: var(--bg-primary);
  border-radius: 8px;
  padding: 2rem;
  width: 100%;
  max-width: 600px;
  max-height: 90vh;
  overflow-y: auto;
  position: relative;
}

.close-btn {
  position: absolute;
  top: 10px;
  right: 10px;
  background: #cc5500;
  color: white;
  border: none;
  padding: 0.5rem 1rem;
```

```
  border-radius: 4px;
  cursor: pointer;
  font-weight: bold;
}

.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 600;
}

.form-group textarea {
  min-height: 100px;
  resize: vertical;
}

.action-buttons {
  display: flex;
  flex-direction: column;
  gap: 1rem;
  align-items: center;
  margin: 2rem 0;
}

.action-buttons .btn {
  width: 100%;
  max-width: 300px;
}

.history-cards {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
  gap: 1rem;
  margin: 1rem 0;
}

.history-card {
  border: 1px solid #cc5500;
  border-radius: 8px;
  padding: 1rem;
  background: var(--bg-primary);
}

.shake {
  animation: shake 0.5s ease-in-out;
}

@keyframes shake {
  0%,
  100% {
    transform: translateX(0);
  }
  25% {
    transform: translateX(-5px);
  }
  75% {
    transform: translateX(5px);
  }
}

/* Mobile Responsive Styles */
@media (max-width: 768px) {
  .dashboard-main {
    padding: 0.5rem;
  }

  .card {
    padding: 0.75rem;
    margin-bottom: 0.75rem;
```

```
}

.card-header {
  font-size: 1.1rem;
}

.search-containers {
  gap: 0.75rem;
}

.search-input {
  min-width: 150px;
  padding: 0.6rem;
  font-size: 0.9rem;
}

.btn {
  padding: 0.6rem 1rem;
  font-size: 0.9rem;
}

.date-range-container {
  grid-template-columns: 1fr;
  gap: 0.75rem;
}

.diet-mode-selector {
  flex-direction: column;
  align-items: center;
}

.diet-mode-selector .btn {
  width: 100%;
  max-width: 250px;
}

.meal-table-container {
  display: none;
}

.meal-cards {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

.meal-card {
  border: 2px solid #cc5500;
  border-radius: 8px;
  padding: 1rem;
  background: var(--bg-primary);
}

.meal-card-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1rem;
  padding-bottom: 0.5rem;
  border-bottom: 1px solid #cc5500;
}

.meal-card-title {
  font-weight: bold;
  color: var(--text-primary);
  font-size: 1.1rem;
}

.meal-card-grid {
  display: grid;
  grid-template-columns: 1fr;
  gap: 0.75rem;
  margin-bottom: 1rem;
```

```
  }

  .meal-card-actions {
    display: flex;
    flex-direction: column;
    gap: 0.5rem;
  }

  .popup-content {
    padding: 1rem;
    margin: 0.5rem;
  }

  .bmi-card {
    padding: 1.5rem 0.5rem;
  }

  .action-buttons {
    margin: 1rem 0;
  }

  .history-cards {
    grid-template-columns: 1fr;
  }

  .selected-meals {
    justify-content: center;
  }
}

@media (max-width: 480px) {
  .dashboard-main {
    padding: 0.25rem;
  }

  .card {
    padding: 0.5rem;
  }

  .card-header {
    font-size: 1rem;
  }

  .search-containers {
    flex-direction: column;
    align-items: stretch;
  }

  .search-button {
    width: 50%;
  }

  .search-input {
    padding: 0.5rem;
    font-size: 0.85rem;
  }

  .btn {
    padding: 0.5rem 0.75rem;
    font-size: 0.85rem;
  }

  .popup-content {
    padding: 0.75rem;
    margin: 0.25rem;
  }

  .meal-card {
    padding: 0.75rem;
  }

  .meal-card-title {
    font-size: 1rem;
```

```
    }

    .form-control {
      padding: 0.5rem;
      font-size: 0.9rem;
    }

    .bmi-card {
      padding: 1rem 0.25rem;
    }
  }

  /* Large screen optimizations */
  @media (min-width: 1200px) {
    .dashboard-main {
      max-width: 100%;
      margin: 0 auto;
    }

    .date-range-container {
      grid-template-columns: repeat(2, 1fr);
    }

    .diet-mode-selector {
      justify-content: center;
    }

    .meal-table {
      min-width: 900px;
    }

    .action-buttons {
      flex-direction: row;
      justify-content: center;
    }

    .action-buttons .btn {
      width: auto;
      min-width: 200px;
    }
  }
`}</style>

<ToastContainer position="top-center" autoClose={2000} />
<h1>Diet Plan Assignment</h1>

{/* Search Section */}
<div className="card">
  <div className="search-containers">
    <input
      type="text"
      placeholder="Enter MRN Number"
      className="search-input"
      value={mrn}
      onChange={(e) => setMrn(e.target.value.toUpperCase())}
      onKeyDown={(e) => {
        if (e.key === "Enter") {
          e.preventDefault();
          handleSearch();
        }
      }}
    />
    <button className="search-button" onClick={handleSearch}>
      Search
    </button>
  </div>
</div>

{clientData && (
  <>
    {/* BMI Card */}
    <div className="card" ref={bmiCardRef}>
      <div className="bmi-card">
```

```
    <h2 className="text-primary" style={{ margin: 0 }}>
      MODIFIED BMI
    </h2>
    <p
      className="text-muted"
      style={{ margin: "0.5rem 0", fontSize: "0.9rem" }}
    >
      Click the button below to enter MODIFIED BMI details.
    </p>
    <button
      className="btn btn-primary"
      onClick={() => setShowBmiPopup(true)}
    >
      Enter MODIFIED BMI
    </button>
  </div>
</div>

{/* Main Content Card */}
<div className="card">
  <h2 className="card-header">Daily Energy & Protein Distribution</h2>

  {(!fromDateTime || !toDateTime) && (
    <div
      style={{
        textAlign: "center",
        padding: "1rem",
        marginBottom: "1rem",
        backgroundColor: "rgba(255, 191, 0, 0.1)",
        color: "#FF0000",
        border: "1px solid #FFBF00",
        borderRadius: "5px",
      }}
    >
      Please select a <strong>From</strong> and <strong>To</strong>{" "}
      date range to activate the distribution section.
    </div>
  )}

  {/* Date Range Pickers */}
  <div className="date-range-container">
    <div className="input-field">
      <label className="input-label">From Date</label>
      <input
        type="date"
        className="form-control"
        value={fromDateTime}
        onChange={(e) => setFromDateTime(e.target.value)}
        min={new Date().toISOString().split("T")[0]}
      />
    </div>
    <div className="input-field">
      <label className="input-label">To Date</label>
      <input
        type="date"
        className="form-control"
        value={toDateTime}
        onChange={(e) => setToDateTime(e.target.value)}
        min={fromDateTime || new Date().toISOString().split("T")[0]}
      />
    </div>
  </div>

  {/* Diet Mode Selector */}
  {clientData && fromDateTime && toDateTime && (
    <div className="card">
      <h2 className="card-header">Choose Diet Plan Mode</h2>
      <div className="diet-mode-selector">
        <button
          className={`btn ${
            dietMode === "manual" ? "btn-primary" : "btn-secondary"
          }`}
          onClick={() => {
```

```
            const hasAI = aiGeneratedText || dietMode === "ai";

            if (hasAI) {
              toast.info(
                <ConfirmSwitchToast
                  message="You're currently using an AI-generated plan. Switch to Manual Mode and disca
                  onConfirm={() => {
                    setEnergyProteinDistribution([]);
                    setSelectedMeals([]);
                    setAiInjected(false);
                    if (location.state)
                      location.state.aiGeneratedText = "";
                    setDietMode("manual");
                    toast.success("Switched to Manual Mode");
                  }}
                />,
                { autoClose: false }
              );
            } else {
              setDietMode("manual");
            }
          }}
        >
          Manual Mode
        </button>

        <button
          className={`btn ${
            dietMode === "ai" ? "btn-primary" : "btn-secondary"
          }`}
          onClick={() => {
            const hasManualData =
              energyProteinDistribution.length > 0;

            if (hasManualData) {
              toast.info(
                <ConfirmSwitchToast
                  message="You've already entered a manual diet plan. Switch to AI Mode and discard it
                  onConfirm={() => {
                    setEnergyProteinDistribution([]);
                    setSelectedMeals([]);
                    setDietMode("ai");
                    toast.success("Switched to AI Mode");
                  }}
                />,
                { autoClose: false }
              );
            } else {
              setDietMode("ai");
            }
          }}
        >
          AI Generated Plan ?
        </button>
      </div>
    </div>
  )}

  {/* Content based on diet mode */}
  {dietMode === "manual" && (
    <div className="card">
      <h2>Manual Meal Plan</h2>
      <p>Select Meal Time, Calories, and Protein</p>

      {/* Meal Selector */}
      <div className="meal-selector">
        <label className="input-label">Select Meal Time</label>
        <select
          className="form-control"
          style={{ maxWidth: "250px" }}
          onChange={(e) => {
            const selected = e.target.value;
            if (selected && !selectedMeals.includes(selected)) {
```

```
        setSelectedMeals((prev) => [...prev, selected]);
        setEnergyProteinDistribution((prev) => [
          ...prev,
          {
            mealName: selected,
            mealTime: "",
            calories: "",
            protein: "",
            options: [],
            enabled: true,
          },
        ]);
      }
    }}
  >
    <option value="">-- Choose Meal --</option>
    {availableMeals.map((meal) => (
      <option key={meal.id} value={meal.name}>
        {meal.name}
      </option>
    ))}
  </select>

  {/* Selected Meals */}
  <div className="selected-meals">
    {selectedMeals.map((meal, index) => (
      <div key={index} className="selected-meal-btn">
        {meal}
        <span
          className="remove-meal"
          onClick={() => {
            setSelectedMeals((prev) =>
              prev.filter((m) => m !== meal)
            );
            setEnergyProteinDistribution((prev) =>
              prev.filter((row) => row.mealName !== meal)
            );
          }}
          title="Remove"
        >
          ×
        </span>
      </div>
    ))}
  </div>
</div>

{/* Meal Table/Cards */}
{fromDateTime && toDateTime && selectedMeals.length > 0 && (
  <>
    {/* Desktop Table */}
    <div className="meal-table-container">
      <table className="meal-table">
        <thead>
          <tr>
            <th>Meal Name</th>
            <th>Time</th>
            <th>Calories</th>
            <th>Protein</th>
            <th>Actions</th>
            <th>Options</th>
          </tr>
        </thead>
        <tbody>
          {energyProteinDistribution.map((row, index) => (
            <tr key={index}>
              <td>
                <div style={{ fontWeight: "bold" }}>
                  {row.mealName}
                </div>
              </td>
              <td>
                <input
```

```
    type="time"
    className="meal-input"
    value={row.mealTime}
    onChange={(e) =>
      handleEnergyProteinChange(
        index,
        "mealTime",
        e.target.value
      )
    }
  />
</td>
<td>
  <input
    className="meal-input"
    type={
      focusedCaloriesIndex === index ||
      row.calories
        ? "number"
        : "text"
    }
    placeholder="kcal"
    value={row.calories}
    onChange={(e) =>
      handleEnergyProteinChange(
        index,
        "calories",
        e.target.value
      )
    }
    onFocus={() => setFocusedCaloriesIndex(index)}
    onBlur={() => setFocusedCaloriesIndex(null)}
  />
</td>
<td>
  <input
    className="meal-input"
    type={
      focusedProteinIndex === index || row.protein
        ? "number"
        : "text"
    }
    placeholder="g"
    value={row.protein}
    onChange={(e) =>
      handleEnergyProteinChange(
        index,
        "protein",
        e.target.value
      )
    }
    onFocus={() => setFocusedProteinIndex(index)}
    onBlur={() => setFocusedProteinIndex(null)}
  />
</td>
<td>
  <button
    className="btn btn-primary"
    style={{
      padding: "0.5rem 1rem",
      fontSize: "0.85rem",
    }}
    onClick={() => handleAddOptionClick(index)}
  >
    Add
  </button>
</td>
<td>
  <button
    className="btn btn-primary"
    style={{
      padding: "0.5rem 1rem",
      fontSize: "0.85rem",
```

```
              }}
              onClick={() =>
                setShowOptionGridPopup({
                  visible: true,
                  index,
                })
              }
              disabled={row.options.length === 0}
            >
              View ({row.options.length})
            </button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
</div>

{/* Mobile Cards */}
<div className="meal-cards">
  {energyProteinDistribution.map((row, index) => (
    <div key={index} className="meal-card">
      <div className="meal-card-header">
        <div className="meal-card-title">
          {row.mealName}
        </div>
      </div>

      <div className="meal-card-grid">
        <div className="input-field">
          <label className="input-label">Time</label>
          <input
            type="time"
            className="form-control"
            value={row.mealTime}
            onChange={(e) =>
              handleEnergyProteinChange(
                index,
                "mealTime",
                e.target.value
              )
            }
          />
        </div>

        <div className="input-field">
          <label className="input-label">Calories</label>
          <input
            className="form-control"
            type={
              focusedCaloriesIndex === index || row.calories
                ? "number"
                : "text"
            }
            placeholder="kcal"
            value={row.calories}
            onChange={(e) =>
              handleEnergyProteinChange(
                index,
                "calories",
                e.target.value
              )
            }
            onFocus={() => setFocusedCaloriesIndex(index)}
            onBlur={() => setFocusedCaloriesIndex(null)}
          />
        </div>

        <div className="input-field">
          <label className="input-label">Protein</label>
          <input
            className="form-control"
            type={
```

```
                    focusedProteinIndex === index || row.protein
                      ? "number"
                      : "text"
                  }
                  placeholder="g"
                  value={row.protein}
                  onChange={(e) =>
                    handleEnergyProteinChange(
                      index,
                      "protein",
                      e.target.value
                    )
                  }
                  onFocus={() => setFocusedProteinIndex(index)}
                  onBlur={() => setFocusedProteinIndex(null)}
                />
              </div>
            </div>

            <div className="meal-card-actions">
              <button
                className="btn btn-primary"
                onClick={() => handleAddOptionClick(index)}
              >
                Add Option
              </button>
              <button
                className="btn btn-primary"
                onClick={() =>
                  setShowOptionGridPopup({
                    visible: true,
                    index,
                  })
                }
                disabled={row.options.length === 0}
              >
                View Options ({row.options.length})
              </button>
            </div>
          </div>
        ))}
      </div>
    </>
  )}
</div>
)}

{dietMode === "ai" && (
  <div id="meal-plan-section" className="card">
    <h2 className="card-header">AI-Generated Meal Plan</h2>
    <p>
      This section will automatically generate a personalized plan
      based on the client's profile and goals.
    </p>

    <div style={{ textAlign: "center", margin: "1rem 0" }}>
      <Link
        to="/diet_ai_assistant"
        state={{
          aiInitialData: {
            mrn,
            clientData,
            fromDateTime,
            toDateTime,
            bmiData,
            energyProteinDistribution,
            selectedMeals,
          },
        }}
        className="btn btn-primary"
      >
        Launch AI Diet Assistant ?
      </Link>
```

```
          </div>

          {aiGeneratedText && (
            <div>
              <h2 className="card-header">AI Generated Diet Plan ?</h2>
              <div
                style={{
                  whiteSpace: "pre-wrap",
                  padding: "1rem",
                  color: "#331a00",
                  maxHeight: "300px",
                  overflowY: "auto",
                  border: "1px solid #cc5500",
                  borderRadius: "6px",
                  backgroundColor: "rgba(204, 85, 0, 0.05)",
                }}
              >
                {aiGeneratedText}
              </div>
            </div>
          )}
        </div>
      )}

      {/* Action Buttons */}
      <div className="action-buttons">
        <button
          className="btn btn-primary"
          onClick={generateDietPDF}
          disabled={
            !fromDateTime ||
            !toDateTime ||
            selectedMeals.length === 0 ||
            energyProteinDistribution.length === 0
          }
        >
          Generate PDF
        </button>

        <button
          id="complete-btn"
          className="btn btn-primary"
          onClick={handleCompleteAssignment}
        >
          Complete Assignment
        </button>

        {pdfUrl && (
          <button
            className="btn btn-secondary"
            onClick={() => setShowPdfViewer(true)}
          >
            View Generated PDF
          </button>
        )}
      </div>
    </div>

    {/* Previous Charts Button */}
    {clientData && (
      <div style={{ textAlign: "center", margin: "1rem 0" }}>
        <button
          className="btn btn-primary"
          onClick={() => setShowHistoryCards(true)}
        >
          Previous Assigned Charts
        </button>
      </div>
    )}
  </>
)}

{/* History Cards */}
```

```
{showHistoryCards && (
  <div className="card">
    <h4 style={{ textAlign: "center", marginBottom: "1rem" }}>
      Previous Plans
    </h4>
    <div className="history-cards">
      {assignedPlans.map((plan, idx) => (
        <div className="history-card" key={idx}>
          <p>
            <strong>From:</strong> {plan.fromDateTime}
          </p>
          <p>
            <strong>To:</strong> {plan.toDateTime}
          </p>
          <div style={{ textAlign: "right", marginTop: "1rem" }}>
            <button
              className="btn btn-primary"
              onClick={() => setSelectedPlanToView(plan)}
            >
              View
            </button>
          </div>
        </div>
      ))}
    </div>

    <div style={{ textAlign: "center", marginTop: "1rem" }}>
      <button
        className="btn btn-primary"
        onClick={() => setShowHistoryCards(false)}
      >
        Close
      </button>
    </div>
  </div>
)}

{/* All Popups */}
{selectedPlanToView && (
  <div
    className="popup-overlay"
    onClick={() => setSelectedPlanToView(null)}
  >
    <div className="popup-content" onClick={(e) => e.stopPropagation()}>
      <button
        className="close-btn"
        onClick={() => setSelectedPlanToView(null)}
      >
        Close
      </button>

      <h3 style={{ textAlign: "center", marginBottom: "1rem" }}>
        Assigned Diet Plan
      </h3>
      <p>
        <strong>From:</strong> {selectedPlanToView.fromDateTime}
      </p>
      <p>
        <strong>To:</strong> {selectedPlanToView.toDateTime}
      </p>
      <p>
        <strong>BMI Category:</strong>{" "}
        {selectedPlanToView.bmiData?.bmiCategory || "-"}
      </p>

      {selectedPlanToView.meals.map((meal, idx) => (
        <div key={idx} className="card" style={{ marginBottom: "1rem" }}>
          <h5 style={{ color: "#cc5500" }}>
            {meal.meal || meal.mealName} (
            {meal.mealTime || "Not specified"})
          </h5>
          <p>
            <strong>Calories:</strong> {meal.calories} kcal
```

```
          </p>
          <p>
            <strong>Protein:</strong> {meal.protein} g
          </p>
          <ul>
            {meal.options.map((opt, i) => (
              <li key={i}>
                <strong>{opt.meal}</strong>
                <br />
                <em>Ingredients:</em> {opt.ingredients}
                <br />
                <em>Recipe:</em> {opt.recipe}
              </li>
            ))}
          </ul>
        </div>
      ))}
    </div>
  </div>
)}

{showViewPopup.visible && (
  <div
    className="popup-overlay"
    onClick={() => setShowViewPopup({ visible: false, option: null })}
  >
    <div className="popup-content" onClick={(e) => e.stopPropagation()}>
      <button
        className="close-btn"
        onClick={() => setShowViewPopup({ visible: false, option: null })}
      >
        Close
      </button>
      <h2>Meal Option Details</h2>
      <p>
        <strong>Meal:</strong> {showViewPopup.option.meal}
      </p>
      <p>
        <strong>Ingredients:</strong> {showViewPopup.option.ingredients}
      </p>
      <p>
        <strong>Recipe:</strong> {showViewPopup.option.recipe}
      </p>
      <p>
        <strong>Video:</strong>{" "}
        <a
          href={showViewPopup.option.cookingVideo}
          target="_blank"
          rel="noreferrer"
        >
          Watch Video
        </a>
      </p>
    </div>
  </div>
)}

{confirmDelete.visible && (
  <div
    className="popup-overlay"
    onClick={() =>
      setConfirmDelete({ visible: false, rowIndex: null, optIndex: null })
    }
  >
    <div className="popup-content" onClick={(e) => e.stopPropagation()}>
      <h3>Confirm Deletion</h3>
      <p>Are you sure you want to delete this meal option?</p>
      <div
        style={{
          display: "flex",
          justifyContent: "flex-end",
          gap: "1rem",
          marginTop: "1rem",
```

```
        }}
      >
        <button className="btn btn-primary" onClick={confirmDeleteOption}>
          Yes, Delete
        </button>
        <button
          className="btn btn-secondary"
          onClick={() =>
            setConfirmDelete({
              visible: false,
              rowIndex: null,
              optIndex: null,
            })
          }
        >
          Cancel
        </button>
      </div>
    </div>
  </div>
)}

{showBmiPopup && (
  <div className="popup-overlay" onClick={() => setShowBmiPopup(false)}>
    <div
      className="popup-content"
      onClick={(e) => e.stopPropagation()}
      style={{ maxWidth: "750px" }}
    >
      <button
        className="close-btn"
        onClick={() => setShowBmiPopup(false)}
      >
        ?
      </button>

      <h2 style={{ textAlign: "center", marginBottom: "1rem" }}>
        Modified BMI
      </h2>

      <form
        onSubmit={(e) => {
          e.preventDefault();
          handleBmiSubmit();
        }}
        style={{
          display: "grid",
          gridTemplateColumns: "repeat(auto-fit, minmax(250px, 1fr))",
          gap: "1rem 2rem",
        }}
      >
        <div
          style={{
            gridColumn: "1 / -1",
            fontWeight: "600",
            fontSize: "1.1rem",
          }}
        >
          ? Client Information
        </div>

        <InputField
          label="Name"
          value={clientData?.name || ""}
          readOnly
        />
        <InputField label="Age" value={clientData?.age || ""} readOnly />
        <InputField
          label="Gender"
          value={clientData?.gender || ""}
          readOnly
        />
        <InputField
```

```
  label="Height (cm)"
  value={clientData?.height || ""}
  readOnly
/>
<InputField
  label="Weight (kg)"
  value={clientData?.weight || ""}
  readOnly
/>

<div
  style={{
    gridColumn: "1 / -1",
    fontWeight: "600",
    fontSize: "1.1rem",
    marginTop: "1rem",
  }}
>
  ?? Auto Calculated & Dropdowns
</div>

<div>
  <label style={{ fontWeight: "600" }}>
    BMI{" "}
    <span
      style={{
        fontWeight: "400",
        fontSize: "0.85rem",
        color: "gray",
      }}
    >
      (Auto-calculated)
    </span>
  </label>
  <input
    type="text"
    value={bmiData.bmi ? `${bmiData.bmi} kg/m²` : ""}
    readOnly
    className="form-control"
    style={{ backgroundColor: "#f0f0f0", color: "#333" }}
  />
</div>

<DropdownField
  label="BMI Category"
  name="bmiCategory"
  value={bmiData.bmiCategory}
  onChange={handleBmiChange}
  options={["Underweight", "Normal", "Overweight", "Obese"]}
/>

<div>
  <label style={{ fontWeight: "600" }}>
    BMR{" "}
    <span
      style={{
        fontWeight: "400",
        fontSize: "0.85rem",
        color: "gray",
      }}
    >
      (Auto-calculated)
    </span>
  </label>
  <input
    type="text"
    value={bmiData.bmr ? `${bmiData.bmr} kcal/day` : ""}
    readOnly
    className="form-control"
    style={{ backgroundColor: "#f0f0f0", color: "#333" }}
  />
</div>
```

```
<DropdownField
  label="Activity Level"
  name="tdee"
  value={bmiData.tdee}
  onChange={handleBmiChange}
  options={[
    "1.2 - Sedentary (little or no exercise)",
    "1.37 - Lightly active (1?3 days/week)",
    "1.55 - Moderately active (3?5 days/week)",
    "1.7 - Very active (6?7 days/week)",
    "1.9 - Extra active (physical job or 2x/day training)",
  ]}
/>

<div>
  <label style={{ fontWeight: "600" }}>
    TDEE{" "}
    <span
      style={{
        fontWeight: "400",
        fontSize: "0.85rem",
        color: "gray",
      }}
    >
      (Auto-calculated after activity selection)
    </span>
  </label>
  <input
    type="text"
    value={
      bmiData.calculatedTdee
        ? `${bmiData.calculatedTdee} kcal/day`
        : ""
    }
    readOnly
    className="form-control"
    style={{ backgroundColor: "#f0f0f0", color: "#333" }}
  />
</div>

<div
  style={{
    gridColumn: "1 / -1",
    fontWeight: "600",
    fontSize: "1.1rem",
    marginTop: "1rem",
  }}
>
  ?? Manual Inputs
</div>

<InputField
  label="Target (kcal/day)"
  name="target"
  value={bmiData.target}
  onChange={handleBmiChange}
/>
<InputField
  label="Energy (kcal/day)"
  name="energy"
  value={bmiData.energy}
  onChange={handleBmiChange}
/>
<InputField
  label="Protein (g)"
  name="protein"
  value={bmiData.protein}
  onChange={handleBmiChange}
/>
<InputField
  label="Carbohydrate (g)"
  name="carbohydrate"
  value={bmiData.carbohydrate}
```

```
          onChange={handleBmiChange}
        />
        <InputField
          label="Fats (g)"
          name="fats"
          value={bmiData.fats}
          onChange={handleBmiChange}
        />

        <div
          style={{
            gridColumn: "1 / -1",
            textAlign: "center",
            marginTop: "1rem",
          }}
        >
          <button type="submit" className="btn btn-primary">
            Save BMI Data
          </button>
        </div>
      </form>
    </div>
  </div>
)}

{showOptionPopup.visible && (
  <div
    className="popup-overlay"
    onClick={() => setShowOptionPopup({ index: null, visible: false })}
  >
    <div className="popup-content" onClick={(e) => e.stopPropagation()}>
      <button
        className="close-btn"
        onClick={() =>
          setShowOptionPopup({ index: null, visible: false })
        }
      >
        Close
      </button>
      <h2>{editIndex.row !== null ? "Edit" : "Add"} Meal Option</h2>

      <div className="form-group">
        <label>Meal</label>
        <input
          type="text"
          name="meal"
          value={newOption.meal}
          onChange={handleOptionChange}
          className="form-control"
          required
        />
      </div>

      <div className="form-group">
        <label>Ingredients</label>
        <textarea
          name="ingredients"
          value={newOption.ingredients}
          onChange={handleOptionChange}
          className="form-control"
          style={{ height: "160px" }}
          required
        />
      </div>

      <div className="form-group">
        <label>Recipe</label>
        <textarea
          name="recipe"
          value={newOption.recipe}
          onChange={handleOptionChange}
          className="form-control"
          style={{ height: "160px" }}
```

```
            required
          />
        </div>

        <div className="form-group">
          <label>Cooking Video Link</label>
          <input
            type="text"
            name="cookingVideo"
            value={newOption.cookingVideo}
            onChange={(e) =>
              setNewOption({ ...newOption, cookingVideo: e.target.value })
            }
            className="form-control"
            placeholder="https://youtube.com/..."
          />
        </div>

        <div style={{ textAlign: "center", marginTop: "1rem" }}>
          <button
            className="btn btn-primary"
            type="button"
            onClick={handleOptionSubmit}
          >
            Submit
          </button>
        </div>
      </div>
    </div>
  )}

  {showOptionGridPopup.visible && (
    <div
      className="popup-overlay"
      onClick={() =>
        setShowOptionGridPopup({ visible: false, index: null })
      }
    >
      <div
        className="popup-content"
        onClick={(e) => e.stopPropagation()}
        style={{ maxWidth: "980px" }}
      >
        <h3 style={{ textAlign: "center", marginBottom: "1rem" }}>
          Assigned Meal Options
        </h3>

        <div className="history-cards">
          {energyProteinDistribution[showOptionGridPopup.index].options.map(
            (opt, i) => (
              <div className="history-card" key={i}>
                <h5 style={{ color: "#cc5500", marginBottom: "0.5rem" }}>
                  Option {i + 1}
                </h5>
                <p
                  style={{
                    wordBreak: "break-word",
                    whiteSpace: "normal",
                    overflowWrap: "break-word",
                    lineHeight: "1.4",
                  }}
                >
                  <strong>Meal:</strong> {opt.meal || "N/A"}
                </p>
                <div
                  style={{
                    display: "flex",
                    justifyContent: "space-between",
                    gap: "0.5rem",
                    marginTop: "1rem",
                    flexWrap: "wrap",
                  }}
                >
```

```
                <button
                  className="btn btn-primary"
                  style={{ flex: "1", minWidth: "60px" }}
                  onClick={() => handleOptionView(opt)}
                >
                  View
                </button>
                <button
                  className="btn btn-primary"
                  style={{ flex: "1", minWidth: "60px" }}
                  onClick={() =>
                    handleOptionEdit(showOptionGridPopup.index, i)
                  }
                >
                  Edit
                </button>
                <button
                  className="btn btn-primary"
                  style={{ flex: "1", minWidth: "60px" }}
                  onClick={() =>
                    handleDeleteConfirm(showOptionGridPopup.index, i)
                  }
                >
                  Delete
                </button>
              </div>
            </div>
          )
        )}
      </div>

      <div style={{ textAlign: "center", marginTop: "1rem" }}>
        <button
          className="btn btn-primary"
          onClick={() =>
            setShowOptionGridPopup({ visible: false, index: null })
          }
        >
          Close
        </button>
      </div>
    </div>
  </div>
)}

{showPdfViewer && pdfUrl && (
  <div className="popup-overlay" onClick={() => setShowPdfViewer(false)}>
    <div
      className="popup-content"
      onClick={(e) => e.stopPropagation()}
      style={{ maxWidth: "800px" }}
    >
      <button
        className="close-btn"
        onClick={() => setShowPdfViewer(false)}
      >
        Close
      </button>
      <iframe
        src={pdfUrl}
        title="Diet PDF"
        width="100%"
        height="600px"
        style={{ border: "none" }}
      ></iframe>
    </div>
  </div>
)}

<div
  id="pdf-preview"
  style={{
    visibility: "hidden",
```

```
    position: "absolute",
    left: "-9999px",
    top: 0,
    backgroundColor: "var(--bg-primary)",
    color: "black",
    padding: "2rem",
    width: "794px",
    fontSize: "14px",
    lineHeight: "1.6",
    zIndex: -1,
  }}
>
  <h2 style={{ textAlign: "center" }}>Personalized Meal Plan</h2>
  <p>
    <strong>Name:</strong> {clientData?.name}
  </p>
  <p>
    <strong>Age:</strong> {clientData?.age}
  </p>
  <p>
    <strong>Height:</strong> {clientData?.height} cm
  </p>
  <p>
    <strong>Weight:</strong> {clientData?.weight} kg
  </p>
  <p>
    <strong>BMI:</strong> {clientData?.bmi}
  </p>
  <hr />
  <h3>Nutrition Blueprint</h3>
  <table style={{ width: "100%", borderCollapse: "collapse" }} border="1">
    <thead>
      <tr>
        <th>BMI Category</th>
        <th>BMR</th>
        <th>TDEE</th>
        <th>Target</th>
        <th>Energy</th>
        <th>Protein</th>
        <th>Carbs</th>
        <th>Fats</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>{bmiData.bmiCategory}</td>
        <td>{bmiData.bmr}</td>
        <td>{bmiData.tdee}</td>
        <td>{bmiData.target}</td>
        <td>{bmiData.energy}</td>
        <td>{bmiData.protein}</td>
        <td>{bmiData.carbohydrate}</td>
        <td>{bmiData.fats}</td>
      </tr>
    </tbody>
  </table>

  <h3 style={{ marginTop: "2rem" }}>
    Daily Energy & Protein Distribution
  </h3>
  <table style={{ width: "100%", borderCollapse: "collapse" }} border="1">
    <thead>
      <tr>
        <th>Meal Time</th>
        <th>Calories</th>
        <th>Protein</th>
        <th>Options</th>
      </tr>
    </thead>
    <tbody>
      {energyProteinDistribution.map((meal, idx) => (
        <tr key={idx}>
          <td>{meal.meal || meal.mealName}</td>
```

```
                <td>{meal.calories}</td>
                <td>{meal.protein}</td>
                <td>
                  <table style={{ width: "100%", borderCollapse: "collapse" }}>
                    <tbody>
                      {meal.options.map((opt, i) => (
                        <tr key={i}>
                          <td style={{ paddingBottom: "10px" }}>
                            <strong>Option {i + 1}:</strong> {opt.meal}
                            <br />
                            <em>Ingredients:</em> {opt.ingredients}
                            <br />
                            <em>Recipe:</em> {opt.recipe}
                            {opt.cookingVideo && (
                              <>
                                <br />
                                <em>Video:</em>{" "}
                                <a
                                  href={opt.cookingVideo}
                                  target="_blank"
                                  rel="noreferrer"
                                >
                                  Watch
                                </a>
                              </>
                            )}
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
};

export default DietMealPlanAssign;
```

File: src\dashboard\diet\diet_assign.js

```
import React, { useState } from "react";
import { useTheme } from "../../ThemeProvider";
import {
  Search,
  Upload,
  Eye,
  Trash2,
  Calendar,
  FileCheck,
  X,
  Download,
  FileType,
} from "lucide-react";
import "../physio/assign.css";

const Assign = () => {
  const { theme } = useTheme();
  const [searchMRN, setSearchMRN] = useState("");
  const [userDetails, setUserDetails] = useState(null);
  const [patientReport, setPatientReport] = useState(null);
  const [uploadedFiles, setUploadedFiles] = useState(Array(15).fill(null));
  const [feedback, setFeedback] = useState(Array(15).fill(""));
  const [pdfs, setPdfs] = useState(Array(15).fill(null));
  const [viewedPdf, setViewedPdf] = useState(null);
  const [viewedFileType, setViewedFileType] = useState(null);

  const fetchUserDetails = async () => {
    setUserDetails({
      name: "John Doe",
```

```
    age: 35,
    weight: "70kg",
    product: "Knee Brace",
  });
};

const fetchPatientReport = async () => {
  setPatientReport({
    details:
      "Patient is responding well to therapy with minor stiffness in left knee.",
  });
};

const handleFileUpload = (event, index) => {
  const file = event.target.files[0];
  if (file) {
    setUploadedFiles((prev) => {
      const newFiles = [...prev];
      newFiles[index] = file;
      return newFiles;
    });
    setPdfs((prev) => {
      const newPdfs = [...prev];
      newPdfs[index] = URL.createObjectURL(file);
      return newPdfs;
    });
  }
};

const handleFeedbackChange = (e, index) => {
  const value = e.target.value;
  setFeedback((prev) => {
    const newFeedback = [...prev];
    newFeedback[index] = value;
    return newFeedback;
  });
};

const removeChartData = (index) => {
  setUploadedFiles((prev) => {
    const newFiles = [...prev];
    newFiles[index] = null;
    return newFiles;
  });
  setPdfs((prev) => {
    const newPdfs = [...prev];
    newPdfs[index] = null;
    return newPdfs;
  });
  setViewedPdf(null);
};

const viewChartData = (index) => {
  setViewedPdf(pdfs[index]);
  setViewedFileType(
    uploadedFiles[index]?.type?.includes("pdf") ? "pdf" : "excel"
  );
};

// Function to truncate filename for display
const getDisplayFileName = (file) => {
  if (!file) return "Upload";
  const maxLength = 15; // Adjust this value to control the length
  const fileName = file.name;
  return fileName.length > maxLength
    ? fileName.substring(0, maxLength - 3) + "..."
    : fileName;
};

return (
  <div className={`assign-page ${theme}`}>
    <div className="assign-container">
      <div className="assign-flex">
```

```
    <div className={`assign-card ${theme === "dark" ? "dark" : ""}`}>
      <h2 className="section-title">Client Details</h2>
      <div className="input-group">
        <input
          type="text"
          className="search-inputs"
          placeholder="Enter MRN"
          value={searchMRN}
          onChange={(e) => setSearchMRN(e.target.value)}
        />
        <button className="search-button" onClick={fetchUserDetails}>
          <Search size={16} />
          <span className="button-text">Search</span>
        </button>
      </div>

      {userDetails && (
        <div className="user-info">
          <p>
            <strong>Username:</strong> {userDetails.name}
          </p>
          <p>
            <strong>Age:</strong> {userDetails.age}
          </p>
          <p>
            <strong>Weight:</strong> {userDetails.weight}
          </p>
          <p>
            <strong>Product:</strong> {userDetails.product}
          </p>
        </div>
      )}
    </div>
    {/* Patient Report */}
    <div className={`assign-card ${theme === "dark" ? "dark" : ""}`}>
      <h2 className="section-title">Client Report</h2>
      <center>
        <button className="btn btn-primary" onClick={fetchPatientReport}>
          <FileCheck size={16} />
          <span className="button-text">View Report</span>
        </button>
      </center>
      {patientReport && (
        <div className="report-text">
          <p>{patientReport.details}</p>
        </div>
      )}
    </div>
  </div>
</div>

{/* Charts and Documentation Table */}
<div className="table-section">
  <h3 className="section-title">Charts and Documentation</h3>
  <div className="table-responsive">
    <table className="table">
      <thead>
        <tr>
          <th className="chart-column">Charts</th>
          <th className="dates-column">Dates (From-To)</th>
          <th className="upload-column">Upload</th>
          <th className="feedback-column">Feedback</th>
          <th className="actions-column">Actions</th>
        </tr>
      </thead>
      <tbody>
        {Array.from({ length: 15 }, (_, i) => (
          <tr key={i}>
            <td>Chart {i + 1}</td>
            <td>
              <div className="date-inputs">
                <div className="date-field">
                  <Calendar size={14} />
```

```
                  <input type="date" className="date-input" />
                </div>
                <span className="date-separator">to</span>
                <div className="date-field">
                  <Calendar size={14} />
                  <input type="date" className="date-input" />
                </div>
              </div>
            </td>
            <td>
              <div className="upload-container">
                <label
                  htmlFor={`upload-${i}`}
                  className="upload-label"
                  title={uploadedFiles[i] ? uploadedFiles[i].name : ""}
                >
                  <Upload size={16} />
                  <span className="file-name">
                    {getDisplayFileName(uploadedFiles[i])}
                  </span>
                </label>
                <input
                  id={`upload-${i}`}
                  type="file"
                  className="file-input"
                  onChange={(e) => handleFileUpload(e, i)}
                  accept=".pdf,.xls,.xlsx"
                />
              </div>
            </td>
            <td>
              <input
                type="text"
                className="feedback-input"
                placeholder="Enter feedback"
                value={feedback[i]}
                onChange={(e) => handleFeedbackChange(e, i)}
              />
            </td>
            <td className="actions-cell">
              <div className="action-buttons">
                <button
                  className="btn btn-view"
                  onClick={() => viewChartData(i)}
                  disabled={!uploadedFiles[i]}
                >
                  <Eye size={16} />
                  <span className="action-text">View</span>
                </button>
                <button
                  className="btn btn-remove"
                  onClick={() => removeChartData(i)}
                  disabled={!uploadedFiles[i]}
                >
                  <Trash2 size={16} />
                  <span className="action-text">Remove</span>
                </button>
              </div>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

{/* Modal for viewing PDF/Excel */}
{viewedPdf && (
  <div className="modal">
    <div className="modal-content">
      <div className="modal-header">
        <h3>Document Viewer</h3>
        <button
```

```
                  className="close-button"
                  onClick={() => setViewedPdf(null)}
                >
                  <X size={20} />
                </button>
              </div>
              <div className="modal-body">
                {viewedFileType === "pdf" ? (
                  <embed
                    src={viewedPdf}
                    type="application/pdf"
                    width="100%"
                    height="600px"
                  />
                ) : (
                  <div className="excel-container">
                    <FileType size={48} />
                    <p>Excel file preview not supported. Please download.</p>
                    <a href={viewedPdf} download className="btn download-button">
                      <Download size={16} /> Download Excel
                    </a>
                  </div>
                )}
              </div>
            </div>
          </div>
        )}
      </div>
    );
};

export default Assign;
```

File: src\dashboard\diet\diet_physioo.js

```
import React, { useState, useEffect, useCallback } from "react";
import { Chart } from "react-google-charts";
import { useNavigate } from "react-router-dom";
import axios from "axios";
import "../physio/physio.css";

const Physioo = () => {
  const [searchMRN, setSearchMRN] = useState("");
  const [users, setUsers] = useState([]);
  const [filteredUsers, setFilteredUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [chartOptions, setChartOptions] = useState({});
  const navigate = useNavigate();

  // Function to get current theme-based chart options
  const getChartOptions = useCallback(() => {
    const computed = getComputedStyle(document.body);

    const textColor = computed.getPropertyValue("--text-primary").trim() || "#1a1a1a";
    const bgColor = computed.getPropertyValue("--bg-primary").trim() || "#ffffff";

    return {
      title: "User Distribution",
      is3D: true,
      slices: { 0: { offset: 0.05 } },
      colors: ["#36A2EB", "#FF6384"],
      titleTextStyle: {
        fontSize: 18,
        bold: true,
        color: textColor,
      },
      legend: {
        position: "bottom",
        textStyle: {
          color: textColor,
          fontSize: 13,
        },
      },
```

```
    pieSliceText: "value",
    pieSliceTextStyle: {
      color: "#ffffff",
      fontSize: 14,
      bold: true,
    },
    backgroundColor: "transparent",
    chartArea: { width: "90%", height: "80%" },
    tooltip: {
      textStyle: { color: textColor },
      backgroundColor: bgColor,
    },
  };
}, []);

// Update chart options whenever theme changes
useEffect(() => {
  const updateChartOptions = () => {
    setChartOptions(getChartOptions());
  };

  // Initialize chart options
  updateChartOptions();

  // Observe data-theme attribute changes on <body>
  const observer = new MutationObserver((mutations) => {
    for (const mutation of mutations) {
      if (
        mutation.type === "attributes" &&
        mutation.attributeName === "data-theme"
      ) {
        updateChartOptions();
      }
    }
  });

  observer.observe(document.body, {
    attributes: true,
    attributeFilter: ["data-theme"],
  });

  // Listen for system theme changes
  const mediaQuery = window.matchMedia("(prefers-color-scheme: dark)");
  mediaQuery.addEventListener("change", updateChartOptions);

  return () => {
    observer.disconnect();
    mediaQuery.removeEventListener("change", updateChartOptions);
  };
}, [getChartOptions]);

useEffect(() => {
  fetchUsers();
}, []);

const fetchUsers = async () => {
  try {
    setLoading(true);
    const response = await axios.get("http://localhost:3001/users");
    setUsers(response.data);
    setFilteredUsers(response.data);
  } catch (error) {
    console.error("Error fetching users:", error);
    setError("Failed to load users data");
  } finally {
    setLoading(false);
  }
};

const handleSearch = () => {
  if (!searchMRN.trim()) {
    setFilteredUsers(users);
    return;
```

```
  }

  const result = users.filter((user) =>
    user.mrn.toLowerCase().includes(searchMRN.toLowerCase())
  );
  setFilteredUsers(result);
};

const handleViewUser = (userId) => {
  navigate(`/user-details/${userId}`);
};

if (loading) {
  return (
    <div className="content-container">
      <div className="loading-spinner">Loading...</div>
    </div>
  );
}

if (error) {
  return (
    <div className="content-container">
      <div className="error-message">{error}</div>
    </div>
  );
}

return (
  <div className="physio-container">
    <h2>User and Subscribers Statistics</h2>

    <div className="physio-header">
      <div className="chart-container">
        <div className="card piechart">
          <Chart
            chartType="PieChart"
            data={[
              ["Metric", "Count"],
              ["Total Users", users.length],
              ["Total Subscribers", users.length],
            ]}
            options={chartOptions}
            width="100%"
            height="300px"
          />
        </div>
      </div>

      <div className="search-container">
        <input
          type="text"
          placeholder="Enter MRN ID"
          value={searchMRN}
          onChange={(e) => setSearchMRN(e.target.value.toUpperCase())}
          className="search-input"
        />
        <button
          onClick={handleSearch}
          className="btn btn-primary search-button"
        >
          Search
        </button>
      </div>
    </div>

    <div className="card user-details">
      <h3 className="section-title">Client Details</h3>
      <div className="table-responsive">
        <table className="user-table">
          <thead>
            <tr>
              <th>MRN ID</th>
```

```
                <th>Name</th>
                <th>Gender</th>
                <th>Product</th>
                <th>Details</th>
              </tr>
            </thead>
            <tbody>
              {filteredUsers.length > 0 ? (
                filteredUsers.map((user) => (
                  <tr key={user.mrn}>
                    <td>{user.mrn}</td>
                    <td>{user.name}</td>
                    <td>{user.gender}</td>
                    <td>{user.product}</td>
                    <td>
                      <button
                        className="btn btn-primary"
                        onClick={() => handleViewUser(user.id)}
                      >
                        View
                      </button>
                    </td>
                  </tr>
                ))
              ) : (
                <tr>
                  <td colSpan="5">No users found.</td>
                </tr>
              )}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  );
};

export default Physioo;
```

File: src\dashboard\diet\diet_profile.js

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { Chart } from "react-google-charts";
import { useTheme } from "../../ThemeProvider";
import TimePicker from "react-time-picker";
import "react-time-picker/dist/TimePicker.css";
import { Eye, Pencil, Trash2 } from "lucide-react";

const useIsMobile = () => {
  const [isMobile, setIsMobile] = useState(window.innerWidth <= 768);

  useEffect(() => {
    const handleResize = () => setIsMobile(window.innerWidth <= 768);
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  return isMobile;
};
// Toast Message Component
const ToastMessage = ({ message, type = "success", duration = 2000 }) => {
  const [visible, setVisible] = useState(!!message);

  useEffect(() => {
    if (message) {
      setVisible(true);
      const timer = setTimeout(() => setVisible(false), duration);
      return () => clearTimeout(timer);
    }
  }, [message, duration]);

  if (!visible || !message) return null;
```

```
  return (
    <div
      className={`toast-message ${type}`}
      style={{
        position: "fixed",
        top: "20px",
        right: "20px",
        padding: "12px 20px",
        borderRadius: "8px",
        color: "white",
        fontWeight: "bold",
        transition: "opacity 0.5s ease-in-out",
        opacity: visible ? 1 : 0,
        zIndex: 1000,
        backgroundColor: type === "error" ? "#ff4d4d" : "#28a745",
        boxShadow: "0px 4px 10px rgba(0,0,0,0.2)",
        display: "flex",
        alignItems: "center",
        textAlign:"center",
        gap: "8px",
      }}
    >
      <i
        className={`fa ${
          type === "error" ? "fa-times-circle" : "fa-check-circle"
        }`}
        style={{ fontSize: "18px" }}
      ></i>
      {message}
    </div>
  );
};

const Profile = () => {
  const navigate = useNavigate();
  const { theme, toggleTheme } = useTheme();
  const [isPopupOpen, setIsPopupOpen] = useState(false);
  const isMobile = useIsMobile();
  const togglePopup = () => setIsPopupOpen(!isPopupOpen);
  const [allclientPopupOpen, setallclientPopupOpen] = useState(false);
  const toggleallclientPopup = () => setallclientPopupOpen(!allclientPopupOpen);

  const [selectedTime, setSelectedTime] = useState("10:00");

  // Toast state
  const [toast, setToast] = useState({
    visible: false,
    message: "",
    type: "success",
  });

  // Show toast message function
  const showToast = (message, type = "success") => {
    setToast({
      visible: true,
      message,
      type,
    });

    // Auto hide after 3 seconds
    setTimeout(() => {
      setToast({
        visible: false,
        message: "",
        type: "success",
      });
    }, 3000);
  };

  const [profileData, setProfileData] = useState(null);
  const fetchProfileData = async () => {
    try {
      const res = await fetch("http://localhost:3001/profile");
```

```
    const data = await res.json();
    setProfileData(data);
  } catch (error) {
    console.error("Error fetching profile:", error);
    showToast("Failed to load profile", "error");
  }
};
useEffect(() => {
  fetchProfileData();
}, []);
if (!profileData)
  return (
    <div className="p-8 text-center text-lg font-medium">
      Loading profile...
    </div>
  );

const handleUpdateProfile = () => {
  showToast("Profile updated successfully!");
};

const handleLogout = () => {
  localStorage.removeItem("token");
  sessionStorage.clear();
  showToast("You have been logged out.");
  navigate("/login");
};

// Added custom button styles
const buttonStyle = {
  display: "block",
  margin: "1rem auto",
  padding: "0.5rem 1.5rem",
};

// Added card style for consistent spacing
const cardStyle = {
  padding: "1.5rem",
  margin: "1rem",
  borderRadius: "8px",
  boxShadow: "0 2px 10px rgba(0,0,0,0.1)",
};

// Added section style for consistent section spacing
const sectionStyle = {
  marginBottom: "2rem",
};

// Responsive styles
const responsiveRowStyle = {
  display: "flex",
  flexDirection: window.innerWidth <= 768 ? "column" : "row",
  gap: "1.5rem",
  marginBottom: "2rem",
};

return (
  <div
    className="profile"
    style={{
      marginRight: isMobile ? "1.2rem" : "0rem",
    }}
  >
    {/* Toast Message Component */}
    {toast.visible && (
      <ToastMessage message={toast.message} type={toast.type} />
    )}
    {/* Personal Info + Client Info */}
    <div className="row" style={responsiveRowStyle}>
      <div
        className="col card"
        style={{
          ...cardStyle,
```

```
      flex: 1,
      border: "0.1px solid #cc5500",
    }}
  >
    <center>
      <h3>Personal Information</h3>
    </center>
    <p>
      <strong>Name:</strong> {profileData.personalInfo.name}
    </p>
    <p>
      <strong>Email:</strong> {profileData.personalInfo.email}
    </p>
    <p>
      <strong>Phone:</strong> {profileData.personalInfo.phone}
    </p>
    <p>
      <strong>DOB:</strong> {profileData.personalInfo.dob}
    </p>
    <div style={{ textAlign: "center" }}>
      <button
        className="btn btn-primary"
        onClick={togglePopup}
        style={buttonStyle}
      >
        View More
      </button>
    </div>
  </div>

  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "1px solid #cc5500",
    }}
  >
    <center>
      <h3>Client Info</h3>
    </center>
    <p>
      <strong>Total Clients:</strong>{" "}
      {profileData.clientData.totalClients}
    </p>
    <p style={{ fontSize: "1.5em", color: "var(--accent)" }}>
      <strong>Random Number:</strong>{" "}
      {profileData.clientData.randomNumber}
    </p>
  </div>
</div>
{/* Performance & Analytics + Weekly Patient Flow side-by-side */}
<div className="row" style={responsiveRowStyle}>
  {/* Performance Chart */}
  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "0.1px solid #cc5500",
    }}
  >
    <center>
      <h3>Performance & Analytics</h3>
    </center>
    <Chart
      chartType="PieChart"
      data={[
        ["Metric", "Value"],
        ["Patients Treated", profileData.performanceData.patientsTreated],
        [
          "Remaining",
          profileData.performanceData.totalPatients -
```

```
            profileData.performanceData.patientsTreated,
          ],
        ]}
        options={{
          title: "Treatment Overview",
          titleTextStyle: {
            fontSize: 18,
            bold: true,
            textAlign: "center",
            color: theme === "dark" ? "#f8f9fa" : "#212529",
          },
          is3D: true,
          pieHole: 0.3,
          colors: ["#cc5500", "#ffb380"],
          legend: {
            position: "bottom",
            textStyle: {
              color: theme === "dark" ? "#f8f9fa" : "#212529",
              fontSize: 13,
            },
          },
          pieSliceText: "value",
          pieSliceTextStyle: {
            color: "#fff",
            fontSize: 14,
            bold: true,
          },
          slices: {
            0: { offset: 0.05 },
          },
          backgroundColor: "transparent",
          chartArea: { width: "80%", height: "80%" }, // Better responsive chart
        }}
        width={"100%"}
        height={"300px"}
      />
      <center>
        {" "}
        <p>
          <strong>Success Rate:</strong>{" "}
          {profileData.performanceData.successRate}
        </p>
      </center>
    </div>

    {/* Weekly Bar Chart */}
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "1px solid #cc5500",
      }}
    >
      <center>
        <h3>Weekly Patient Flow</h3>
      </center>
      <Chart
        chartType="BarChart"
        data={[
          ["Week", "Appointments", "Patients Treated"],
          ["Week 1", 10, profileData.weeklyMetrics.patientsPerWeek[0]],
          ["Week 2", 12, profileData.weeklyMetrics.patientsPerWeek[1]],
          ["Week 3", 14, profileData.weeklyMetrics.patientsPerWeek[2]],
          ["Week 4", 16, profileData.weeklyMetrics.patientsPerWeek[3]],
        ]}
        options={{
          title: "",
          backgroundColor: "transparent",
          titleTextStyle: {
            fontSize: 18,
            bold: true,
            color: theme === "dark" ? "#f8f9fa" : "#212529",
```

```
        },
        legend: {
          position: "bottom",
          textStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
            fontSize: 13,
          },
        },
        chartArea: { width: "60%" },
        hAxis: {
          title: "Count",
          minValue: 0,
          textStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
          },
          titleTextStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
            bold: true,
          },
          gridlines: {
            color: theme === "dark" ? "#444" : "#ccc",
          },
        },
        vAxis: {
          title: "Week",
          textStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
          },
          titleTextStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
            bold: true,
          },
        },
        colors: ["#cc5500", "#ff884d"],
        bar: { groupWidth: "60%" },
      }}
      width={"100%"}
      height={"300px"}
    />
  </div>
</div>
<div className="section" style={sectionStyle}>
  <div className="section-header mb-4 text-[#cc5500]">
    <h2>Client Management</h2>
  </div>
  <div className="row" style={responsiveRowStyle}>
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "0.1px solid #cc5500",
      }}
    >
      <center>
        <h3>Active Clients</h3>
      </center>
      <div
        className="metric-display"
        style={{ textAlign: "center", margin: "1.5rem 0" }}
      >
        <span
          className="metric-number"
          style={{
            fontSize: "2.5rem",
            fontWeight: "bold",
            display: "block",
          }}
        >
          {profileData.clientManagement.activeClients}
        </span>
        <span
          className="metric-label"
```

```
      style={{ display: "block", color: "#666" }}
    >
      Active
    </span>
  </div>
  <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
    Currently enrolled in treatment plans
  </p>
  <div style={{ textAlign: "center" }}>
    <button
      className="btn btn-primary"
      style={buttonStyle}
      onClick={() => {
        toggleallclientPopup();
      }}
    >
      View All Active Clients
    </button>
  </div>
</div>

<div
  className="col card"
  style={{
    ...cardStyle,
    flex: 1,
    border: "1px solid #cc5500",
  }}
>
  <center>
    <h3>Past Clients</h3>
  </center>
  <div
    className="metric-display"
    style={{ textAlign: "center", margin: "1.5rem 0" }}
  >
    <span
      className="metric-number"
      style={{
        fontSize: "2.5rem",
        fontWeight: "bold",
        display: "block",
      }}
    >
      {profileData.clientManagement.pastClients}
    </span>
    <span
      className="metric-label"
      style={{ display: "block", color: "#666" }}
    >
      Past
    </span>
  </div>
  <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
    Completed treatment programs
  </p>
  <div style={{ textAlign: "center" }}>
    <button
      className="btn btn-primary"
      style={buttonStyle}
      onClick={() => {
        toggleallclientPopup();
      }}
    >
      View Treatment History
    </button>
  </div>
</div>

<div
  className="col card"
  style={{
    ...cardStyle,
```

```
          flex: 1,
          border: "1px solid #cc5500",
        }}
      >
        <center>
          <h3>Pending Consultations</h3>
        </center>
        <div
          className="metric-display"
          style={{ textAlign: "center", margin: "1.5rem 0" }}
        >
          <span
            className="metric-number"
            style={{
              fontSize: "2.5rem",
              fontWeight: "bold",
              display: "block",
            }}
          >
            {profileData.clientManagement.pendingConsultations}
          </span>
          <span
            className="metric-label"
            style={{ display: "block", color: "#666" }}
          >
            Pending
          </span>
        </div>
        <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
          Awaiting initial assessment
        </p>
        <div style={{ textAlign: "center" }}>
          <button
            className="btn btn-primary"
            style={buttonStyle}
            onClick={() => {
              togglePopup();
            }}
          >
            Schedule Consultation
          </button>
        </div>
      </div>
    </div>
  </div>
  {/* NEW SECTION: Exercise Plans */}
  <div className="section" style={sectionStyle}>
    <div className="section-header" style={{ marginBottom: "1rem" }}>
      <h2>Exercise Plans</h2>
    </div>
    <div className="row" style={responsiveRowStyle}>
      <div
        className="col card"
        style={{
          ...cardStyle,
          flex: 1,
          border: "0.1px solid #cc5500",
        }}
      >
        <center>
          <h3>Create New Plan</h3>
        </center>
        <div className="plan-form" style={{ marginTop: "1.5rem" }}>
          <select
            className="form-control"
            style={{
              marginBottom: "1rem",
              padding: "0.5rem",
              width: "100%",
            }}
          >
            <option>Select Client...</option>
            <option>Alice Smith</option>
```

```
      <option>Bob Johnson</option>
      <option>Carol Davis</option>
    </select>
    <select
      className="form-control"
      style={{
        marginBottom: "1.5rem",
        padding: "0.5rem",
        width: "100%",
      }}
    >
      <option>Select Template...</option>
      <option>Lower Back Rehabilitation</option>
      <option>Knee Strengthening</option>
      <option>Post-Surgery Recovery</option>
    </select>
    <div style={{ textAlign: "center" }}>
      <button
        className="btn btn-primary"
        style={buttonStyle}
        onClick={() => {
          togglePopup();
        }}
      >
        Create Custom Plan
      </button>
    </div>
  </div>
</div>

<div
  className="col card"
  style={{
    ...cardStyle,
    flex: 1,
    border: "1px solid #cc5500",
  }}
>
  <center>
    <h3>Manage Existing Plans</h3>
  </center>
  <p style={{ margin: "1rem 0" }}>
    <strong>Total Active Plans:</strong>{" "}
    {profileData.exercisePlans.totalPlans}
  </p>
  <ul
    className="plan-list"
    style={{ listStyle: "none", padding: 0, margin: "1.5rem 0" }}
  >
    <li
      style={{
        margin: "0.5rem 0",
        display: "flex",
        justifyContent: "space-between",
      }}
    >
      Lower Back Rehabilitation{" "}
      <span
        className="badge"
        style={{
          background: "#cc5500",
          color: "white",
          padding: "0.25rem 0.5rem",
          borderRadius: "4px",
        }}
      >
        12 Clients
      </span>
    </li>
    <li
      style={{
        margin: "0.5rem 0",
        display: "flex",
```

```
          justifyContent: "space-between",
        }}
      >
        Knee Strengthening{" "}
        <span
          className="badge"
          style={{
            background: "#cc5500",
            color: "white",
            padding: "0.25rem 0.5rem",
            borderRadius: "4px",
          }}
        >
          8 Clients
        </span>
      </li>
      <li
        style={{
          margin: "0.5rem 0",
          display: "flex",
          justifyContent: "space-between",
        }}
      >
        Shoulder Mobility{" "}
        <span
          className="badge"
          style={{
            background: "#cc5500",
            color: "white",
            padding: "0.25rem 0.5rem",
            borderRadius: "4px",
          }}
        >
          14 Clients
        </span>
      </li>
    </ul>
    <div style={{ textAlign: "center" }}>
      <button
        className="btn btn-primary"
        style={buttonStyle}
        onClick={() => {
          togglePopup();
        }}
      >
        View All Plans
      </button>
    </div>
  </div>

  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "1px solid #cc5500",
    }}
  >
    <center>
      <h3>Plan Templates</h3>
    </center>
    <p style={{ margin: "1rem 0" }}>
      <strong>Available Templates:</strong>{" "}
      {profileData.exercisePlans.templates}
    </p>

    <table
      style={{
        width: "100%",
        marginTop: "1rem",
        borderSpacing: "0 0.5rem",
      }}
    >
```

```
        <tbody>
          {[
            "Lower Back Rehabilitation",
            "Knee Strengthening",
            "Post-Surgery Recovery",
          ].map((template, index) => (
            <tr key={index} style={{ backgroundColor: "transparent" }}>
              <td style={{ padding: "0.5rem 0" }}>{template}</td>
              <td style={{ textAlign: "right" }}>
                <button
                  className="btn btn-primary"
                  style={{ padding: "0.25rem 0.75rem" }}
                  onClick={() =>
                    showToast(`Editing template: ${template}`)
                  }
                >
                  Edit
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
      <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
        <button
          className="btn btn-primary"
          style={buttonStyle}
          onClick={() => {
            togglePopup();
          }}
        >
          Create New Template
        </button>
      </div>
    </div>
  </div>
</div>
{/* NEW SECTION: Appointments */}
<div className="section" style={sectionStyle}>
  <div className="section-header" style={{ marginBottom: "1rem" }}>
    <h2>Appointments</h2>
  </div>

  {/* Row 1: Schedule Form and Calendar */}
  <div className="row" style={responsiveRowStyle}>
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "0.1px solid #cc5500",
      }}
    >
      <center>
        <h3>Schedule Consultation</h3>
      </center>
      <form className="consultation-form" style={{ width: "100%" }}>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="clientName">Client Name:</label>
          <input
            type="text"
            id="clientName"
            className="form-control"
            style={{
              width: "100%",
              padding: "0.5rem",
              marginTop: "0.5rem",
            }}
          />
        </div>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="date">Date:</label>
          <input
```

```
          type="date"
          id="date"
          className="form-control"
          style={{
            width: "100%",
            padding: "0.5rem",
            marginTop: "0.5rem",
          }}
        />
      </div>
      <div style={{ marginBottom: "1rem" }}>
        <label htmlFor="time">Time:</label>
        <input
          type="time"
          id="time"
          className="form-control"
          style={{
            width: "100%",
            padding: "0.5rem",
            marginTop: "0.5rem",
          }}
        />
      </div>
      <div style={{ marginBottom: "1rem" }}>
        <label htmlFor="type">Consultation Type:</label>
        <select
          id="type"
          className="form-control"
          style={{
            width: "100%",
            padding: "0.5rem",
            marginTop: "0.5rem",
          }}
        >
          <option>Initial Assessment</option>
          <option>Follow-up</option>
          <option>Treatment Session</option>
        </select>
      </div>
      <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
        <button
          type="button"
          className="btn btn-primary"
          style={buttonStyle}
          onClick={() =>
            showToast("Appointment scheduled successfully!")
          }
        >
          Schedule
        </button>
      </div>
    </form>
  </div>
</div>

{/* Row 2: Today's Appointments */}
<div className="row" style={responsiveRowStyle}>
  <div
    className="appointments-card card"
    style={{
      ...cardStyle,
      flex: 1,
      width: "100%",
      border: "0.1px solid #cc5500",
    }}
  >
    <center>
      <h3>Today's Appointments</h3>
    </center>
    <div style={{ overflowX: "auto" }}>
      {" "}
      {/* Makes table scrollable on small screens */}
      <table
```

```
  className="appointments-table"
  style={{
    width: "100%",
    borderCollapse: "separate",
    borderSpacing: "0 0.5rem",
    marginTop: "1rem",
  }}
>
  <thead>
    <tr>
      <th
        style={{
          textAlign: "left",
          padding: "0.5rem",
          borderBottom: "1px solid #ddd",
        }}
      >
        Patient
      </th>
      <th
        style={{
          textAlign: "left",
          padding: "0.5rem",
          borderBottom: "1px solid #ddd",
        }}
      >
        Time
      </th>
      <th
        style={{
          textAlign: "left",
          padding: "0.5rem",
          borderBottom: "1px solid #ddd",
        }}
      >
        Status
      </th>
      <th
        style={{
          textAlign: "center",
          padding: "0.5rem",
          borderBottom: "1px solid #ddd",
        }}
      >
        Actions
      </th>
    </tr>
  </thead>
  <tbody>
    {profileData.todaysAppointments.map((appointment, index) => (
      <tr key={index}>
        <td style={{ padding: "0.75rem 0.5rem" }}>
          {appointment.patient}
        </td>
        <td style={{ padding: "0.75rem 0.5rem" }}>
          {appointment.time}
        </td>
        <td style={{ padding: "0.75rem 0.5rem" }}>
          <span
            className={`status-tag status-${appointment.status.toLowerCase()}`}
            style={{
              background:
                appointment.status === "Confirmed"
                  ? "#28a745"
                  : "#ffc107",
              color: "white",
              padding: "0.25rem 0.5rem",
              borderRadius: "4px",
              fontSize: "0.75rem",
            }}
          >
            {appointment.status}
          </span>
```

```
            </td>
            <td
              style={{
                textAlign: "center",
                padding: "0.75rem 0.5rem",
              }}
            >
              <div
                className="action-buttons"
                style={{
                  display: "flex",
                  flexDirection:
                    window.innerWidth <= 500 ? "column" : "row",
                  gap: "0.5rem",
                  justifyContent: "center",
                }}
              >
                <button
                  className="btn btn-primary"
                  style={{
                    padding: "0.25rem 0.75rem",
                  }}
                  onClick={() =>
                    showToast(
                      `Session started with ${appointment.patient}`
                    )
                  }
                >
                  Start
                </button>
                <button
                  className="btn btn-success"
                  style={{
                    padding: "0.25rem 0.75rem",
                  }}
                  onClick={() =>
                    showToast(
                      `Rescheduling ${appointment.patient}'s appointment`
                    )
                  }
                >
                  Reschedule
                </button>
              </div>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>

  {/* Time Picker Component */}
  <div style={{ marginTop: "2rem" }}>
    <h4>Quick Time Selection</h4>
    <div
      style={{
        display: "flex",
        alignItems: "center",
        marginTop: "1rem",
        flexDirection: window.innerWidth <= 768 ? "column" : "row",
      }}
    >
      <label
        htmlFor="time"
        style={{
          marginRight: "1rem",
          marginBottom: window.innerWidth <= 768 ? "0.5rem" : 0,
        }}
      >
        Select Time:
      </label>
      <TimePicker
        id="time"
```

```
            onChange={(time) => {
              setSelectedTime(time);
              showToast(`Time selected: ${time}`);
            }}
            value={selectedTime}
            disableClock={true}
            clearIcon={null}
            className="form-control"
            format="hh:mm a"
          />
        </div>
      </div>
    </div>
  </div>
  <div
    className="row"
    style={{ display: "flex", gap: "1.5rem", marginBottom: "1.5rem" }}
  >
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "1px solid #cc5500",
        overflow: "hidden",

        display: "flex",
        flexDirection: "column",
        maxHeight: "360px", // adjust as needed
      }}
    >
      <center>
        <h3>Upcoming Appointments</h3>
      </center>

      <div
        style={{
          overflowY: "auto",
          flex: 2,
          marginTop: "1rem",
          scrollbarColor: "#cc5500 transparent",
          scrollbarWidth: "thin",
        }}
      >
        <style>
          {`
  /* WebKit-based browsers */
  .col.card::-webkit-scrollbar {
    width: 8px;
  }
  .col.card::-webkit-scrollbar-track {
    background: transparent;
  }
  .col.card::-webkit-scrollbar-thumb {
    background-color: #cc5500;
    border-radius: 4px;
  }
`}
        </style>
        <table
          style={{
            width: "100%",
            borderSpacing: "0 0.75rem", // horizontal 0, vertical 0.75rem space between rows
            fontSize: "0.9rem",
            borderCollapse: "separate", // IMPORTANT: allows borderSpacing to work
          }}
        >
          <thead>
            <tr>
              <th
                style={{
                  borderBottom: "1px solid #ccc",
                  textAlign: "left",
```

```
                    padding: "8px",
                    backgroundColor: "var(--bg-primary)",
                    position: "sticky",
                    top: 0,
                    zIndex: 1,
                  }}
                >
                  Date
                </th>
                <th
                  style={{
                    borderBottom: "1px solid #ccc",
                    textAlign: "left",
                    padding: "8px",
                    backgroundColor: "var(--bg-primary)",
                    position: "sticky",
                    top: 0,
                    zIndex: 1,
                  }}
                >
                  Patient
                </th>
                <th
                  style={{
                    borderBottom: "1px solid #ccc",
                    textAlign: "left",
                    padding: "8px",
                    backgroundColor: "var(--bg-primary)",
                    position: "sticky",
                    top: 0,
                    zIndex: 1,
                  }}
                >
                  Time
                </th>
              </tr>
            </thead>
            <tbody>
              {profileData.appointments.map((appointment, index) => (
                <tr key={index}>
                  <td
                    style={{
                      padding: "8px",
                      borderBottom: "1px solid #eee",
                    }}
                  >
                    {appointment.date}
                  </td>
                  <td
                    style={{
                      padding: "8px",
                      borderBottom: "1px solid #eee",
                    }}
                  >
                    {appointment.patient}
                  </td>
                  <td
                    style={{
                      padding: "8px",
                      borderBottom: "1px solid #eee",
                    }}
                  >
                    {appointment.time}
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</div>{" "}
{isPopupOpen && (
```

```
<div
  className="popup"
  style={{
    position: "fixed",
    top: 0,
    left: 0,
    width: "100vw",
    height: "100vh",
    backgroundColor: "rgba(0, 0, 0, 0.5)",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    zIndex: 1000,
  }}
>
  <div
    className="popup-content card"
    style={{
      backgroundColor: "var(--bg-primary)",
      padding: "2rem",
      borderRadius: "8px",
      maxWidth: "600px",
      width: "90%",
      maxHeight: "90vh",
      overflow: "auto",
    }}
  >
    <h3>Physiotherapist Details</h3>
    <p>
      <strong>Specialization:</strong>{" "}
      {profileData.professionalInfo.specialization}
    </p>
    <p>
      <strong>Experience:</strong>{" "}
      {profileData.professionalInfo.experience}
    </p>
    <p>
      <strong>Certifications:</strong>{" "}
      {profileData.professionalInfo.certifications.join(", ")}
    </p>
    <Chart
      chartType="PieChart"
      data={[
        ["Metric", "Value"],
        [
          "Patients Treated",
          profileData.performanceData.patientsTreated,
        ],
        [
          "Remaining",
          profileData.performanceData.totalPatients -
            profileData.performanceData.patientsTreated,
        ],
      ]}
      options={{
        title: "Performance Insights",
        backgroundColor: "transparent",
        titleTextStyle: {
          color: theme === "dark" ? "white" : "#212529",
        },
        legend: {
          textStyle: {
            color: theme === "dark" ? "white" : "#212529",
          },
        },
        pieSliceTextStyle: {
          color: theme === "dark" ? "white" : "#212529",
        },
        tooltip: {
          textStyle: {
            color: theme === "dark" ? "white" : "#212529",
          },
        },
```

```
        }}
        width={"100%"}
        height={"300px"}
      />

      <div style={{ textAlign: "center" }}>
        <button
          className="btn btn-primary"
          onClick={togglePopup}
          style={buttonStyle}
        >
          Close
        </button>
      </div>
    </div>
  </div>
)}
{allclientPopupOpen && (
  <div
    className="popup"
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      width: "100%",
      height: "100%",
      backgroundColor: "rgba(0, 0, 0, 0.5)",
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      zIndex: 1000,
    }}
  >
    <div
      className="popup-content card"
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        top: "50%",
        left: "50%",
        borderRadius: "8px",
        border: "1px solid #cc5500",
        maxWidth: "900px",
        width: "95%",
        maxHeight: "90vh",
        overflow: "auto",
        color: "var(--text-primary)",
      }}
    >
      <h3 style={{ marginBottom: "1rem" }}>Upcoming Appointments</h3>

      <table style={{ width: "100%", borderCollapse: "collapse" }}>
        <thead>
          <tr style={{ backgroundColor: "#cc5500" }}>
            <th style={{ padding: "0.75rem", textAlign: "left" }}>
              Date
            </th>
            <th style={{ padding: "0.75rem", textAlign: "left" }}>
              Patient
            </th>
            <th style={{ padding: "0.75rem", textAlign: "left" }}>
              Time
            </th>
            <th style={{ padding: "0.75rem", textAlign: "left" }}>
              Status
            </th>
            <th style={{ padding: "0.75rem", textAlign: "center" }}>
              Actions
            </th>
          </tr>
        </thead>
        <tbody>
          {profileData.appointments.map((appt, index) => (
```

```
                <tr
                  key={index}
                  style={{ borderBottom: "1px solid var(--border-color)" }}
                >
                  <td style={{ padding: "0.75rem" }}>{appt.date}</td>
                  <td style={{ padding: "0.75rem" }}>{appt.patient}</td>
                  <td style={{ padding: "0.75rem" }}>{appt.time}</td>
                  <td style={{ padding: "0.75rem" }}>Scheduled</td>
                  <td style={{ padding: "0.75rem", textAlign: "center" }}>
                    <button
                      className="btn btn-secondary"
                      style={{ marginRight: "0.5rem" }}
                      onClick={() => showToast(`Viewing ${appt.patient}`)}
                    >
                      <Eye size={16} />
                    </button>
                    <button
                      className="btn btn-secondary"
                      style={{ marginRight: "0.5rem" }}
                      onClick={() => showToast(`Editing ${appt.patient}`)}
                    >
                      <Pencil size={16} />
                    </button>
                    <button
                      className="btn btn-secondary"
                      onClick={() => showToast(`Deleting ${appt.patient}`)}
                    >
                      <Trash2 size={16} />
                    </button>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>

          <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
            <button
              className="btn btn-primary"
              onClick={toggleallclientPopup}
              style={{ padding: "0.5rem 1.5rem", fontSize: "1rem" }}
            >
              Close
            </button>
          </div>
        </div>
      </div>
    )}
  </div>
);
};

export default Profile;
```

File: src\dashboard\diet\components\diet_footer.js

```
import React from 'react';
import '../../physio/components/Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} BalanceBite Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\diet\components\diet_navbar.js

```
import React from "react";
import "../../physio/components/Navbar.css";
import { Moon, Sun } from "lucide-react";
```

```
import { useTheme } from "../../../ThemeProvider";

const Navbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="../diet/diet_profile"> <div className="navbar-logo">BalanceBite</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default Navbar;
```

File: src\dashboard\diet\components\diet_sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "../../physio/components/Sidebar.css";
import {
  Home,
  Users,
  Dumbbell,
  BarChart2,
  LogOut,
  Hamburger,
  Menu,
  Workflow,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
} from "lucide-react";

const DietSidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState("");
  const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
  const navigate = useNavigate();

  const toggleSidebar = () => {
    setIsOpen(!isOpen);
  };

  const toggleCollapse = () => {
    setCollapsed(!collapsed);
  };

  const handleResize = () => {
    setIsOpen(window.innerWidth > 768);
  };

  const closeSidebar = () => {
    if (isOpen && window.innerWidth <= 768) {
      setIsOpen(false);
    }
  };

  const handleMouseEnter = () => {
    if (collapsed && window.innerWidth > 768) {
      setIsHovering(true);
    }
  };
```

```
const handleMouseLeave = () => {
  if (collapsed && window.innerWidth > 768) {
    setIsHovering(false);
  }
};

useEffect(() => {
  window.addEventListener("resize", handleResize);
  return () => window.removeEventListener("resize", handleResize);
}, []);

useEffect(() => {
  document.body.classList.toggle(
    "sidebar-collapsed",
    collapsed && !isHovering
  );
}, [collapsed, isHovering]);

const navLinkClass = ({ isActive }) => (isActive ? "active" : "");

const handleLogoutClick = (e) => {
  e.preventDefault();
  setShowLogoutConfirm(true);
};

const confirmLogout = () => {
  localStorage.clear();
  setShowLogoutConfirm(false);
  setToastMessage("Logged out successfully.");
  setShowToast(true);
  setTimeout(() => {
    setShowToast(false);
    navigate("/login");
  }, 3000);
};

const cancelLogout = () => {
  setShowLogoutConfirm(false);
};

return (
  <>
    <button
      className="sidebar-toggle"
      onClick={toggleSidebar}
      aria-label="Toggle sidebar"
    >
      <Menu size={24} />
    </button>

    {isOpen && window.innerWidth <= 768 && (
      <div className="sidebar-overlay show" onClick={closeSidebar}></div>
    )}

    {showLogoutConfirm && (
      <div className="modal-overlay">
        <div className="modal-container">
          <div className="modal-header">
            <center>
              <h3>Logout Confirmation</h3>
            </center>
            <button className="modal-close" onClick={cancelLogout}>
              <X size={18} />
            </button>
          </div>
          <div className="modal-body">
            <p>Are you sure you want to log out?</p>
          </div>
          <div className="modal-footer">
            <button className="btn btn-primary" onClick={confirmLogout}>
              Yes, Logout
            </button>
```

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

{showToast && (
  <div className="toast-overlay">
    <div className="toast-container success">
      <div className="toast-content">
        <Check size={18} style={{ marginRight: "8px" }} />
        {toastMessage}
      </div>
    </div>
  </div>
)}

<aside
  className={`sidebar ${isOpen ? "open" : ""} ${
    collapsed ? "collapsed" : ""
  } ${isHovering ? "hovering" : ""}`}
  onMouseEnter={handleMouseEnter}
  onMouseLeave={handleMouseLeave}
>
  <div className="sidebar-header">
    <div className="sidebar-profile">
      <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
      <span className="sidebar-username">Jane Dietitian</span>
    </div>
  </div>

  <nav className="sidebar-menu-wrapper">
    <ul className="sidebar-menu">
      <li>
        <NavLink to="/diet/diet_profile" className={navLinkClass}>
          <Home size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Dashboard</span>
        </NavLink>
      </li>

      <li>
        <NavLink to="/diet/diet_assign" className={navLinkClass}>
          <Hamburger size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Assign Diet Plan</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/diet/DietitiansAppointments" className={navLinkClass}>
          <Hamburger size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Appointments</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/diet/DietMealPlanAssign" className={navLinkClass}>
          <Hamburger size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>DietMealPlanAssign</span>
        </NavLink>
      </li>

      <li>
        <NavLink to="/diet/DietPasswordRequest" className={navLinkClass}>
          <Workflow size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Password Request</span>
        </NavLink>
      </li>
      <li>
        <a
          href="#"
          onClick={handleLogoutClick}
          className={navLinkClass({ isActive: false })}
          style={{ cursor: "pointer", display: "flex", alignItems: "center" }}
        >
          <LogOut size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
          <span>Log Out</span>
```

```
              </a>
            </li>
          </ul>

          <button className="collapse-toggle" onClick={toggleCollapse}>
            <div className="claude-toggle">
              {collapsed ? <ChevronRight size={16} /> : <ChevronLeft size={16} />}
            </div>
          </button>
        </nav>
      </aside>
    </>
  );
};

export default DietSidebar;
```

File: src\dashboard\diet\pages\DietPasswordRequest.js

```
import React, { useState } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../../ThemeProvider"; // Adjust the import based on your project structure
import "react-toastify/dist/ReactToastify.css";
import "../../master_admin/master_admin.css"; // Ensure this path is correct

const DietPasswordRequest = () => {
  const { theme } = useTheme(); // Access the current theme
  const [formData, setFormData] = useState({
    counselorName: "",
    email: "",
    reason: "",
  });

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!formData.counselorName || !formData.email || !formData.reason) {
      toast.error("All fields are required!");
      return;
    }

    try {
      await axios.post("http://localhost:5000/passwordChangeRequests", {
        ...formData,
        status: "Pending",
        requestedAt: new Date().toISOString(),
      });

      toast.success("Request submitted to admin!");
      setFormData({ counselorName: "", email: "", reason: "" });
    } catch (err) {
      toast.error("Failed to submit request.");
    }
  };

  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Password Change Request</h1>
      <div className="card" style={{border:"1px solid #cc5500"}}>
        <form onSubmit={handleSubmit} className="form">
          <div className="form-group">
            <label htmlFor="counselorName">Counselor Name:</label>
            <input
              type="text"
              id="counselorName"
              name="counselorName"
              placeholder="Enter your name"
              value={formData.counselorName}
```

```
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="email">Email ID:</label>
            <input
              type="email"
              id="email"
              placeholder="Enter your email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="reason">Reason for Change:</label>
            <input
              type="text"
              id="reason"
              placeholder="Enter reason for password change"
              name="reason"
              value={formData.reason}
              onChange={handleChange}
              className="input-field"

              required
            />
          </div>
          <div className="center-btn">
           <center> <button type="submit" className="btn btn-primary">
              Submit Request
            </button></center>
          </div>
        </form>
      </div>
    </div>
  );
};

export default DietPasswordRequest;
```

File: src\dashboard\diet__tests__\DietMealPlanAssign.test.js

```
import React from "react";
import {
  act,
  render,
  screen,
  fireEvent,
  waitFor,
} from "@testing-library/react";
import DietMealPlanAssign from "../DietMealPlanAssign";
import { ToastContainer } from "react-toastify";

// Mock scrollIntoView for hidden elements like PDF preview
beforeAll(() => {
  window.HTMLElement.prototype.scrollIntoView = function () {};
});

// Helper function for act-wrapped render
const customRender = async () => {
  await act(async () => {
    render(
      <>
        <ToastContainer />
        <DietMealPlanAssign />
      </>
    );
  });
};
```

```
describe("DietMealPlanAssign Component", () => {
  beforeEach(() => {
    global.fetch = jest.fn((url) => {
      if (url.includes("mealclients?mrn=MRN123")) {
        return Promise.resolve({
          ok: true,
          json: () =>
            Promise.resolve([
              {
                mrn: "MRN123",
                name: "Test User",
                age: 25,
                height: 170,
                weight: 65,
                bmi: 22,
                goal: "Muscle Gain",
              },
            ]),
        });
      }

      if (url.includes("assignedMeals?mrn=MRN123")) {
        return Promise.resolve({
          ok: true,
          json: () => Promise.resolve([]),
        });
      }

      if (url.includes("availableMeals")) {
        return Promise.resolve({
          ok: true,
          json: () =>
            Promise.resolve([
              {
                mealTime: "Breakfast",
                options: ["Oats", "Eggs"],
              },
              {
                mealTime: "Lunch",
                options: ["Rice", "Dal"],
              },
            ]),
        });
      }

      if (url.includes("dietaryGuidelines")) {
        return Promise.resolve({
          ok: true,
          json: () => Promise.resolve(["No sugar", "High protein"]),
        });
      }

      if (url.includes("assignedMeals")) {
        return Promise.resolve({
          ok: true,
          json: () => Promise.resolve({}),
        });
      }

      return Promise.resolve({
        ok: true,
        json: () => Promise.resolve([]),
      });
    });
  });

  test("renders page title", async () => {
    await customRender();
    expect(screen.getByText(/Diet Plan Assignment/i)).toBeInTheDocument();
  });

  test("shows toast on empty MRN", async () => {
```

```
    await customRender();
    fireEvent.click(screen.getByText("Search"));
    expect(await screen.findByText(/please enter a valid mrn/i)).toBeInTheDocument();
  });

  test("fetches client by MRN and shows client info", async () => {
    await customRender();
    fireEvent.change(screen.getByPlaceholderText(/Enter MRN Number/i), {
      target: { value: "MRN123" },
    });
    fireEvent.click(screen.getByText("Search"));

    await waitFor(() => {
      expect(screen.getByText(/Muscle Gain/)).toBeInTheDocument();
    });

    const nameMatches = await screen.findAllByText(/Test User/);
    expect(nameMatches.length).toBeGreaterThanOrEqual(1);
  });

test("shows and validates BMI popup", async () => {
  await customRender();

  fireEvent.change(screen.getByPlaceholderText("Enter MRN Number"), {
    target: { value: "MRN123" },
  });
  fireEvent.click(screen.getByText("Search"));

  await waitFor(() => {
    expect(screen.getByText(/Muscle Gain/)).toBeInTheDocument();
  });

  const bmiButton = screen.getByRole("button", {
    name: /Enter MODIFIED BMI/i,
  });
  fireEvent.click(bmiButton);

  // Fix: Use role or more specific query
  const modalHeadings = await screen.findAllByText(/MODIFIED BMI/i);
  expect(modalHeadings.length).toBeGreaterThan(0);

  fireEvent.click(screen.getByText("Submit"));
  expect(await screen.findByText(/All BMI fields are required/i)).toBeInTheDocument();
});

test("selects date range and meal time", async () => {
  render(
    <>
      <ToastContainer />
      <DietMealPlanAssign />
    </>
  );

  fireEvent.change(screen.getByPlaceholderText("Enter MRN Number"), {
    target: { value: "MRN123" },
  });
  fireEvent.click(screen.getByText("Search"));

  // Wait until client loads
  await waitFor(() => {
    expect(screen.getByText(/Muscle Gain/i)).toBeInTheDocument();
  });

  // Wait until the meal options render
  await waitFor(() =>
    expect(screen.getByRole("option", { name: "Breakfast" })).toBeInTheDocument()
  );

  // Set dates
  const dateInputs = screen.getAllByPlaceholderText(/DD-MM-YYYY/i);
  fireEvent.change(dateInputs[0], { target: { value: "2025-07-01" } });
  fireEvent.change(dateInputs[1], { target: { value: "2025-07-15" } });
```

```
  // Select meal
  const select = screen.getByRole("combobox");
  fireEvent.change(select, { target: { value: "Breakfast" } });

  expect(select.value).toBe("Breakfast");
});

  test("shows buttons for Generate PDF and Complete Assignment", async () => {
    await customRender();
    fireEvent.change(screen.getByPlaceholderText("Enter MRN Number"), {
      target: { value: "MRN123" },
    });
    fireEvent.click(screen.getByText("Search"));

    await waitFor(() => {
      expect(screen.getByText(/Complete Assignment/i)).toBeInTheDocument();
    });

    expect(screen.getByText(/Generate PDF/i)).toBeInTheDocument();
    expect(screen.getByText(/Complete Assignment/i)).toBeInTheDocument();
  });
});
// Note: The test for PDF generation is not included as it requires additional setup for the PDF library.
```

File: src\dashboard\doctor\DoctorAssignmentDashboard.js

```
import React, { useEffect, useState } from "react";
import { useTheme } from "../../ThemeProvider";
import "../physio/assign.css";
import { FileText, Loader } from "lucide-react";

const DoctorAssignmentDashboard = () => {
  const { theme } = useTheme();
  const [assignedPatients, setAssignedPatients] = useState([]);
  const [loading, setLoading] = useState(true);
  const [searchMRN, setSearchMRN] = useState("");

  // Document Viewer States
  const [showDocumentModal, setShowDocumentModal] = useState(false);
  const [activeTab, setActiveTab] = useState("medical");
  const [pdfUrl, setPdfUrl] = useState("");
  const [medicalData, setMedicalData] = useState([]);
  const [testRecords, setTestRecords] = useState([]);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const res = await fetch("http://localhost:3001/patients_detailed");
        const patients = await res.json();

        const filtered = patients
          .filter((p) => p.approvals?.physio || p.approvals?.diet)
          .sort((a, b) => {
            const dateA = new Date(a.assignedTests?.[0]?.assignedDate || 0);
            const dateB = new Date(b.assignedTests?.[0]?.assignedDate || 0);
            return dateB - dateA;
          });

        setAssignedPatients(filtered);
      } catch (error) {
        console.error("Error loading data:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  const filteredPatients = assignedPatients.filter((p) =>
    p.id.toLowerCase().includes(searchMRN.toLowerCase())
  );

  const handleViewHistory = (patientId) => {
```

```
  // Replace with actual API data later
  const dummyMedical = [
    {
      date: "2024-05-10",
      time: "10:00 AM",
      clinicalNote: "/dummy-clinical.pdf",
      prescription: "/dummy-prescription.pdf",
    },
  ];
  const dummyTests = [
    { date: "2024-05-09", name: "Blood Test", url: "/dummy-test.pdf" },
  ];

  setMedicalData(dummyMedical);
  setTestRecords(dummyTests);
  setShowDocumentModal(true);
};

if (loading) {
  return (
    <div className={`assign-page ${theme}`}>
      <div className="assign-container">
        <div className="loading-container">
          <Loader className="loading-spinner" size={32} />
          <p className="loading-text">Loading assigned patients...</p>
        </div>
      </div>
    </div>
  );
}

return (
  <div className={`assign-page ${theme}`}>
    <div className="assign-container">
      <h2 className="section-title">Assigned Patients Dashboard</h2>

      <div style={{ margin: "1rem 0" }}>
        <input
          type="text"
          className="search-input"
          placeholder="Search by MRN"
          value={searchMRN}
          onChange={(e) => setSearchMRN(e.target.value.toUpperCase())}
          style={{
            padding: "0.6rem 1rem",
            fontSize: "1rem",
            width: "100%",
            maxWidth: "400px",
            borderRadius: "4px",
            border: "1px solid #ccc",
          }}
        />
      </div>

      {filteredPatients.length === 0 ? (
        <div className="empty-state">
          <FileText size={48} className="empty-icon" />
          <p className="empty-text">No patients assigned yet</p>
        </div>
      ) : (
        <div className="table-wrapper">
          <table className="styled-table">
            <thead>
              <tr>
                <th>Name</th>
                <th>MRN</th>
                <th>Assigned To</th>
                <th>Prescription File</th>
                <th>Assign Tests</th>
                <th>Counselor Type</th>
                <th>History</th> {/* NEW COLUMN */}
              </tr>
            </thead>
```

```
<tbody>
  {filteredPatients.map((patient) => {
    const testAssigned = patient.assignedTests?.length > 0;
    const counselorType = patient.counselorAssignment?.type;

    return (
      <tr key={patient.id}>
        <td data-label="Name">{patient.clientName}</td>
        <td data-label="MRN">{patient.id}</td>
        <td data-label="Assigned To">
          <div className="assignment-tags">
            {patient.approvals?.physio && (
              <span className="assignment-tag physio">
                Physio
              </span>
            )}
            {patient.approvals?.diet && (
              <span className="assignment-tag dietitian">
                Dietitian
              </span>
            )}
            {patient.approvals?.nutrition && (
              <span className="assignment-tag nutrition">
                Nutrition
              </span>
            )}
          </div>
        </td>
        <td data-label="Prescription File">
          {patient.lastVisit?.prescription ? (
            <a
              href={patient.lastVisit.prescription}
              target="_blank"
              rel="noopener noreferrer"
              className="pdf-link"
            >
              <FileText size={18} /> View File
            </a>
          ) : (
            <span className="no-file">No File</span>
          )}
        </td>
        <td data-label="Assign Tests">
          {testAssigned ? (
            <span className="assignment-tag">? Assigned</span>
          ) : (
            <span
              className="assignment-tag"
              style={{ backgroundColor: "#eee", color: "#999" }}
            >
              ? Not Assigned
            </span>
          )}
        </td>
        <td data-label="Counselor Type">
          {counselorType ? (
            <span className="assignment-tag">
              {counselorType}
            </span>
          ) : (
            <span
              className="assignment-tag"
              style={{ backgroundColor: "#eee", color: "#999" }}
            >
              ? Not Assigned
            </span>
          )}
        </td>
        <td data-label="History" style={{ textAlign: "center" }}>
          <center>
            <button
              className="btn btn-primary"
              onClick={() => handleViewHistory(patient.id)}
```

```
                      title="View History"
                      style={{ padding: "0.3rem 0.6rem" }}
                    >
                      <FileText size={18} />
                    </button>
                  </center>
                </td>
              </tr>
            );
          })}
        </tbody>
      </table>
    </div>
  )}
</div>

{/* Document Viewer Modal */}
{showDocumentModal && (
  <div
    onClick={() => setShowDocumentModal(false)}
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
      backgroundColor: "rgba(0,0,0,0.6)",
      zIndex: 9999,
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      padding: "1rem",
      boxSizing: "border-box",
    }}
  >
    <div
      onClick={(e) => e.stopPropagation()}
      className="modal-responsive"
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        borderRadius: "8px",
        maxWidth: "70%",
        width: "90%",
        maxHeight: "90vh",
        overflow: "auto",
        border: "1px solid #cc5500",
        position: "relative",
        boxSizing: "border-box",
      }}
    >
      {/* Tabs */}
      <div
        className="tab-buttons"
        style={{
          display: "flex",
          gap: "1rem",
          marginBottom: "1rem",
          flexWrap: "wrap",
        }}
      >
        <button
          onClick={() => setActiveTab("medical")}
          className={`btn btn-responsive ${
            activeTab === "medical" ? "btn-active" : "btn-primary"
          }`}
          style={{ flex: "1" }}
        >
          Medical
        </button>
        <button
          onClick={() => setActiveTab("test")}
          className={`btn btn-responsive ${
```

```
      activeTab === "test" ? "btn-active" : "btn-primary"
    }`}
    style={{ flex: "1" }}
  >
    Test Record
  </button>
</div>

{/* Medical Tab */}
{activeTab === "medical" && (
  <div className="responsive-table">
    <table className="user-table">
      <thead>
        <tr>
          <th>Date</th>
          <th>Time</th>
          <th style={{ textAlign: "center" }}>Clinical Note</th>
          <th style={{ textAlign: "center" }}>Prescription</th>
        </tr>
      </thead>
      <tbody>
        {medicalData.map((item, idx) => (
          <tr key={idx}>
            <td>{item.date}</td>
            <td>{item.time}</td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setPdfUrl(item.clinicalNote)}
                >
                  View
                </button>
              </center>
            </td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setPdfUrl(item.prescription)}
                >
                  View
                </button>
              </center>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{/* Test Tab */}
{activeTab === "test" && (
  <div className="responsive-table">
    <table className="user-table">
      <thead>
        <tr>
          <th>Date</th>
          <th>Test Name</th>
          <th style={{ textAlign: "center" }}>View</th>
        </tr>
      </thead>
      <tbody>
        {testRecords.map((item, idx) => (
          <tr key={idx}>
            <td>{item.date}</td>
            <td>{item.name}</td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setPdfUrl(item.url)}
```

```
                    >
                      View
                    </button>
                  </center>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}

    {/* PDF Viewer */}
    {pdfUrl && (
      <div
        style={{
          marginTop: "1rem",
          padding: "1rem",
          background: "#f9f9f9",
          borderRadius: "8px",
        }}
      >
        <div
          style={{
            display: "flex",
            justifyContent: "space-between",
            alignItems: "center",
            marginBottom: "1rem",
            flexWrap: "wrap",
          }}
        >
          <button
            onClick={() => setPdfUrl("")}
            className="btn btn-primary"
          >
            ×
          </button>
          <a
            href={pdfUrl}
            target="_blank"
            rel="noopener noreferrer"
            className="btn btn-primary"
          >
            ? Open in New Tab
          </a>
        </div>
        <iframe
          src={pdfUrl}
          title="PDF Viewer"
          width="100%"
          height="500px"
          style={{
            border: "1px solid #ddd",
            borderRadius: "4px",
            minHeight: "300px",
          }}
        />
      </div>
    )}

    <center>
      <button
        className="btn btn-primary"
        onClick={() => setShowDocumentModal(false)}
        style={{ marginTop: "2rem" }}
      >
        Close
      </button>
    </center>
  </div>
</div>
)}
<style jsx>{`
```

```
/* Loading Styles */
.loading-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  padding: 3rem 1rem;
  gap: 1rem;
}

.loading-spinner {
  animation: spin 1s linear infinite;
  color: ${theme === "dark"
    ? "var(--primary-dark, #64b5f6)"
    : "var(--primary-light, #1976d2)"};
}

.loading-text {
  color: ${theme === "dark"
    ? "var(--text-secondary-dark, #cccccc)"
    : "var(--text-secondary-light, #666666)"};
  font-size: 1rem;
  margin: 0;
}

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

/* Empty State */
.empty-state {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  padding: 3rem 1rem;
  gap: 1rem;
}

.empty-icon {
  color: ${theme === "dark"
    ? "var(--text-tertiary-dark, #888888)"
    : "var(--text-tertiary-light, #999999)"};
  opacity: 0.6;
}

.empty-text {
  color: ${theme === "dark"
    ? "var(--text-secondary-dark, #cccccc)"
    : "var(--text-secondary-light, #666666)"};
  font-size: 1.1rem;
  margin: 0;
}

/* Table Wrapper */
.table-wrapper {
  width: 100%;
  overflow-x: auto;
  margin-top: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

/* Table Styles */
.styled-table {
  width: 100%;
  border-collapse: collapse;
  min-width: 700px;
```

```
  background-color: ${theme === "dark"
    ? "var(--background-secondary, #1a1a1a)"
    : "var(--background-primary, #ffffff)"};
}

.styled-table th,
.styled-table td {
  padding: 1rem 0.75rem;
  border-bottom: 1px solid
    ${theme === "dark"
      ? "var(--border-dark, #333)"
      : "var(--border-light, #e0e0e0)"};
  text-align: left;
  vertical-align: middle;
}

.styled-table th {
  background-color: ${theme === "dark"
    ? "var(--accent-dark, #2d2d2d)"
    : "var(--accent-light, #f8f9fa)"};
  color: ${theme === "dark"
    ? "var(--text-primary-dark, #ffffff)"
    : "var(--text-primary-light, #333333)"};
  font-weight: 600;
  font-size: 0.875rem;
  text-transform: uppercase;
  letter-spacing: 0.5px;
  position: sticky;
  top: 0;
  z-index: 10;
}

.styled-table td {
  color: ${theme === "dark"
    ? "var(--text-secondary-dark, #cccccc)"
    : "var(--text-secondary-light, #666666)"};
  font-size: 0.875rem;
}

.styled-table tr:hover {
  background-color: ${theme === "dark"
    ? "var(--hover-dark, #2a2a2a)"
    : "var(--hover-light, #f5f5f5)"};
  transition: background-color 0.2s ease;
}

/* Prescription Text */
.prescription-text {
  display: block;
  max-width: 200px;
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}

/* Assignment Tags */
.assignment-tags {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
}

.assignment-tag {
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  font-size: 0.75rem;
  font-weight: 500;
  text-transform: uppercase;
  letter-spacing: 0.5px;
}

.assignment-tag.physio {
  background-color: ${theme === "dark"
```

```
    ? "rgba(76, 175, 80, 0.2)"
    : "rgba(76, 175, 80, 0.1)"};
  color: ${theme === "dark" ? "#81c784" : "#4caf50"};
  border: 1px solid
    ${theme === "dark" ? "#4caf50" : "rgba(76, 175, 80, 0.3)"};
}

.assignment-tag.dietitian {
  background-color: ${theme === "dark"
    ? "rgba(255, 152, 0, 0.2)"
    : "rgba(255, 152, 0, 0.1)"};
  color: ${theme === "dark" ? "#ffb74d" : "#ff9800"};
  border: 1px solid
    ${theme === "dark" ? "#ff9800" : "rgba(255, 152, 0, 0.3)"};
}
.assignment-tag.nutrition {
  background-color: ${theme === "dark"
    ? "rgba(33, 150, 243, 0.2)"
    : "rgba(33, 150, 243, 0.1)"};
  color: ${theme === "dark" ? "#64b5f6" : "#2196f3"};
  border: 1px solid
    ${theme === "dark" ? "#2196f3" : "rgba(33, 150, 243, 0.3)"};
}

/* PDF Link */
.pdf-link {
  display: inline-flex;
  align-items: center;
  gap: 0.5rem;
  color: ${theme === "dark"
    ? "var(--primary-dark, #64b5f6)"
    : "var(--primary-light, #1976d2)"};
  font-weight: 500;
  text-decoration: none;
  padding: 0.25rem 0.5rem;
  border-radius: 4px;
  transition: all 0.2s ease;
}

.pdf-link:hover {
  background-color: ${theme === "dark"
    ? "rgba(100, 181, 246, 0.1)"
    : "rgba(25, 118, 210, 0.1)"};
  text-decoration: none;
}

.no-file {
  color: ${theme === "dark"
    ? "var(--text-tertiary-dark, #888888)"
    : "var(--text-tertiary-light, #999999)"};
  font-style: italic;
}

/* Tablet Styles */
@media (max-width: 1024px) {
  .table-wrapper {
    margin-top: 1rem;
  }

  .styled-table {
    min-width: 600px;
  }

  .styled-table th,
  .styled-table td {
    padding: 0.75rem 0.5rem;
    font-size: 0.8rem;
  }

  .prescription-text {
    max-width: 150px;
  }
}
```

```
/* Mobile Styles */
@media (max-width: 768px) {
  .table-wrapper {
    overflow-x: visible;
    box-shadow: none;
  }

  .styled-table,
  .styled-table thead,
  .styled-table tbody,
  .styled-table th,
  .styled-table td,
  .styled-table tr {
    display: block;
  }

  .styled-table {
    min-width: auto;
    background-color: transparent;
  }

  .styled-table thead {
    display: none;
  }

  .styled-table tr {
    margin-bottom: 1rem;
    border: 1px solid
      ${theme === "dark"
        ? "var(--border-dark, #333)"
        : "var(--border-light, #e0e0e0)"};
    border-radius: 8px;
    padding: 1rem;
    background-color: ${theme === "dark"
      ? "var(--background-secondary, #1a1a1a)"
      : "var(--background-primary, #ffffff)"};
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  .styled-table tr:hover {
    background-color: ${theme === "dark"
      ? "var(--hover-dark, #2a2a2a)"
      : "var(--hover-light, #f5f5f5)"};
  }

  .styled-table td {
    display: flex;
    justify-content: space-between;
    align-items: flex-start;
    padding: 0.75rem 0;
    border-bottom: 1px solid
      ${theme === "dark"
        ? "var(--border-dark, #333)"
        : "var(--border-light, #e0e0e0)"};
    font-size: 0.875rem;
  }

  .styled-table td:last-child {
    border-bottom: none;
  }

  .styled-table td::before {
    content: attr(data-label);
    font-weight: 600;
    color: ${theme === "dark"
      ? "var(--text-primary-dark, #ffffff)"
      : "var(--text-primary-light, #333333)"};
    width: 40%;
    flex-shrink: 0;
    font-size: 0.8rem;
    text-transform: uppercase;
    letter-spacing: 0.5px;
```

```
  }

  .prescription-text {
    max-width: none;
    text-align: right;
    width: 60%;
  }

  .assignment-tags {
    justify-content: flex-end;
    width: 60%;
  }

  .pdf-link {
    justify-content: flex-end;
    padding: 0.5rem 0.75rem;
    border-radius: 6px;
    font-size: 0.8rem;
  }

  .no-file {
    text-align: right;
    width: 60%;
  }
}

/* Small Mobile Styles */
@media (max-width: 480px) {
  .assign-container {
    padding: 0.5rem;
  }

  .section-title {
    font-size: 1.25rem;
    margin-bottom: 1rem;
  }

  .styled-table tr {
    padding: 0.75rem;
  }

  .styled-table td {
    flex-direction: column;
    align-items: flex-start;
    gap: 0.5rem;
    padding: 0.5rem 0;
  }

  .styled-table td::before {
    width: 100%;
    margin-bottom: 0.25rem;
  }

  .prescription-text,
  .assignment-tags,
  .pdf-link,
  .no-file {
    width: 100%;
    text-align: left;
  }

  .assignment-tags {
    justify-content: flex-start;
  }

  .pdf-link {
    justify-content: flex-start;
    align-self: flex-start;
    margin-top: 0.25rem;
  }
}

/* Large Desktop Styles */
```

```
        @media (min-width: 1200px) {
          .styled-table th,
          .styled-table td {
            padding: 1.25rem 1rem;
          }

          .styled-table {
            font-size: 0.9rem;
          }

          .prescription-text {
            max-width: 250px;
          }
        }

        /* Print Styles */
        @media print {
          .table-wrapper {
            box-shadow: none;
          }

          .styled-table {
            border: 1px solid #000;
          }

          .styled-table th,
          .styled-table td {
            border: 1px solid #000;
            color: #000;
            background-color: #fff;
          }

          .pdf-link {
            color: #000;
          }

          .assignment-tag {
            border: 1px solid #000;
            background-color: #fff;
            color: #000;
          }
        }
      `}</style>
    </div>
  );
};

export default DoctorAssignmentDashboard;
```

File: src\dashboard\doctor\DoctorDashboardHome.js

```
import React, { useState, useEffect } from "react";
import { useTheme } from "../../ThemeProvider";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../physio/clientManagement.css";
import { CheckCircle } from "lucide-react";

const DoctorDashboardHome = () => {
  const { theme } = useTheme();

  const [quote, setQuote] = useState("");
  const [appointments, setAppointments] = useState([]);
  const [testCategories, setTestCategories] = useState([]);
  const [searchMRN, setSearchMRN] = useState("");
  const [patientData, setPatientData] = useState(null);
  const [showDocumentModal, setShowDocumentModal] = useState(false);
  const [activeTab, setActiveTab] = useState("medical");
  const [pdfUrl, setPdfUrl] = useState(null);
  const [showCounselorModal, setShowCounselorModal] = useState(false);

  const [counselorTypes, setCounselorTypes] = useState([]);
  const [selectedCounselor, setSelectedCounselor] = useState("");
  const [counselorNote, setCounselorNote] = useState("");
```

```
const [showLastVisitModal, setShowLastVisitModal] = useState(false);
const scrollRef = React.useRef(null);
const [showAssignTestModal, setShowAssignTestModal] = useState(false);
const [testForm, setTestForm] = useState({
  testType: "",
  testName: "",
  priority: "normal",
  notes: "",
  scheduledDate: "",
  scheduledTime: "",
});

const quotes = [
  "Every patient is a story waiting to be heard.",
  "The best doctors give the best of themselves.",
  "Wherever the art of medicine is loved, there is also a love of humanity.",
  "A kind doctor makes healing happen faster.",
];

useEffect(() => {
  // Set motivational quote
  setQuote(quotes[Math.floor(Math.random() * quotes.length)]);

  // Fetch today's appointments
  fetch("http://localhost:3001/appointments")
    .then((res) => {
      if (!res.ok) throw new Error("Failed to fetch appointments");
      return res.json();
    })
    .then(setAppointments)
    .catch(() => {
      toast.error("Could not fetch appointments");
    });

  // Fetch test categories for dynamic test name dropdown
  fetch("http://localhost:3001/testCategories")
    .then((res) => {
      if (!res.ok) throw new Error("Failed to fetch test categories");
      return res.json();
    })
    .then(setTestCategories)
    .catch(() => {
      toast.error("Could not load test types");
    });

  // Fetch counselor types
  fetch("http://localhost:3001/counselorTypes")
    .then((res) => {
      if (!res.ok) throw new Error("Failed to fetch counselor types");
      return res.json();
    })
    .then(setCounselorTypes)
    .catch(() => {
      toast.error("Could not load counselor types");
    });
}, []);

const handleSearch = () => {
  if (!searchMRN.trim()) {
    toast.warning("Please enter a valid MRN");
    return;
  }

  fetch(`http://localhost:3001/patients_detailed/${searchMRN}`)
    .then((res) => {
      if (!res.ok) throw new Error("Patient not found");
      return res.json();
    })
    .then((data) => {
      toast.success("Patient found!");
      setPatientData(data);
    })
```

```
    .catch(() => {
      toast.error("No patient found with this MRN");
      setPatientData(null);
    });
};

const handleApproval = (type) => {
  if (!patientData || !type) return;

  const updatedApproval = {
    ...patientData.approvals,
    [type]: true,
  };

  fetch(`http://localhost:3001/patients_detailed/${patientData.id}`, {
    method: "PATCH",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ approvals: updatedApproval }),
  })
    .then((res) => {
      if (!res.ok) throw new Error("Failed to update approval");
      return res.json();
    })
    .then((data) => {
      toast.success(
        `${type.charAt(0).toUpperCase() + type.slice(1)} Approved!`
      );
      setPatientData(data); // Update state with backend response
    })
    .catch(() => {
      toast.error("Could not update approval in database");
    });
};
const showUndoApprovalToast = (typeLabelText, onConfirm, theme) => {
  const toastId = "deassign-confirmation"; // prevent duplicate confirmation toasts

  if (toast.isActive(toastId)) return; // skip if it's already showing

  toast(
    ({ closeToast }) => (
      <div style={{ textAlign: "center" }}>
        <p style={{ marginBottom: "0.5rem" }}>
          Are you sure you want to deassign this patient from{" "}
          <strong>{typeLabelText}</strong>?
        </p>
        <div
          style={{
            display: "flex",
            justifyContent: "center",
            gap: "1rem",
          }}
        >
          <button
            onClick={() => {
              onConfirm();
              toast.dismiss(toastId);
            }}
            style={{
              padding: "0.4rem 1rem",
              backgroundColor: "#cc5500",
              color: "white",
              border: "none",
              borderRadius: "4px",
              cursor: "pointer",
            }}
          >
            Confirm
          </button>
          <button
            onClick={() => toast.dismiss(toastId)}
            style={{
```

```
              padding: "0.4rem 1rem",
              backgroundColor: "#e0e0e0",
              color: "#333",
              border: "none",
              borderRadius: "4px",
              cursor: "pointer",
            }}
          >
            Cancel
          </button>
        </div>
      </div>
    ),
    {
      toastId, // prevent stacking
      position: "top-center",
      autoClose: false,
      closeOnClick: false,
      closeButton: false,
      draggable: false,
    }
  );
};

const handleAssignTest = () => {
  if (
    !testForm.testType ||
    !testForm.testName ||
    !testForm.scheduledDate ||
    !testForm.scheduledTime
  ) {
    toast.warning("Please fill all required fields");
    return;
  }

  const newTest = {
    id: Date.now(), // Simple ID generation
    ...testForm,
    status: "scheduled",
    assignedDate: new Date().toISOString().split("T")[0],
    patientId: patientData.id,
  };

  // Update patient data with new test
  const updatedPatient = {
    ...patientData,
    assignedTests: [...(patientData.assignedTests || []), newTest],
  };

  fetch(`http://localhost:3001/patients_detailed/${patientData.id}`, {
    method: "PATCH",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ assignedTests: updatedPatient.assignedTests }),
  })
    .then((res) => {
      if (!res.ok) throw new Error("Failed to assign test");
      return res.json();
    })
    .then((data) => {
      toast.success("Test assigned successfully!");
      setPatientData(data);
      setShowAssignTestModal(false);
      // Reset form
      setTestForm({
        testType: "",
        testName: "",
        priority: "normal",
        notes: "",
        scheduledDate: "",
        scheduledTime: "",
      });
```

```
    })
    .catch(() => {
      toast.error("Could not assign test");
    });
};
const medicalData = [
  {
    date: "2025-06-01",
    time: "10:30 AM",
    clinicalNote: "/pdfs/MRN01.pdf",
    prescription: "/prescriptions/MRN123456_labreport.pdf",
  },
];

const testRecords =
  patientData?.documents?.map((doc) => ({
    name: doc,
    url: `/pdfs/${doc}`,
  })) || [];

return (
  <div
    className={`dashboard-main ${theme}`}
    style={{
      padding: "1rem",
      minHeight: "100vh",
      boxSizing: "border-box",
    }}
  >
    <ToastContainer position="top-center" autoClose={2000} />

    {/* Responsive Styles */}
    <style jsx>{`
      /* Theme-aware Scrollbar Styles */
      .dashboard-main ::-webkit-scrollbar {
        width: 8px;
        height: 8px;
      }

      .dashboard-main ::-webkit-scrollbar-track {
        background: var(--bg-secondary, #f1f1f1);
        border-radius: 4px;
      }

      .dashboard-main ::-webkit-scrollbar-thumb {
        background: var(--primary-color, #cc5500);
        border-radius: 4px;
        transition: background 0.3s ease;
      }

      .dashboard-main ::-webkit-scrollbar-thumb:hover {
        background: var(--primary-hover, #b84a00);
      }

      /* Dark theme scrollbar */
      .dashboard-main.dark ::-webkit-scrollbar-track {
        background: var(--bg-secondary, #2a2a2a);
      }

      .dashboard-main.dark ::-webkit-scrollbar-thumb {
        background: var(--primary-color, #ff6b35);
      }

      .dashboard-main.dark ::-webkit-scrollbar-thumb:hover {
        background: var(--primary-hover, #ff5722);
      }

      /* Modal scrollbars */
      .modal-responsive ::-webkit-scrollbar {
        width: 6px;
        height: 6px;
      }
```

```
.modal-responsive ::-webkit-scrollbar-track {
  background: var(--bg-secondary, #f9f9f9);
  border-radius: 3px;
}

.modal-responsive ::-webkit-scrollbar-thumb {
  background: var(--primary-color, #cc5500);
  border-radius: 3px;
}

.modal-responsive ::-webkit-scrollbar-thumb:hover {
  background: var(--primary-hover, #b84a00);
}

/* Table scrollbars */
.responsive-table ::-webkit-scrollbar {
  height: 6px;
}

.responsive-table ::-webkit-scrollbar-track {
  background: var(--bg-secondary, #f1f1f1);
  border-radius: 3px;
}

.responsive-table ::-webkit-scrollbar-thumb {
  background: var(--primary-color, #cc5500);
  border-radius: 3px;
}

.responsive-table ::-webkit-scrollbar-thumb:hover {
  background: var(--primary-hover, #b84a00);
}

/* Firefox scrollbar styling */
.dashboard-main {
  scrollbar-width: thin;
  scrollbar-color: var(--primary-color, #cc5500)
    var(--bg-secondary, #f1f1f1);
}

.dashboard-main.dark {
  scrollbar-color: var(--primary-color, #ff6b35)
    var(--bg-secondary, #2a2a2a);
}
.date-time-grid {
  grid-template-columns: 1fr 1fr;
}

@media (max-width: 480px) {
  .date-time-grid {
    grid-template-columns: 1fr !important;
  }

  .modal-responsive form > div > input,
  .modal-responsive form > div > select,
  .modal-responsive form > div > textarea {
    font-size: 16px !important; /* Prevents zoom on iOS */
  }
}
@media (max-width: 768px) {
  .responsive-table {
    display: block;
    overflow-x: auto;
    white-space: nowrap;
  }

  .responsive-table table {
    min-width: 100%;
    font-size: 0.8rem;
  }

  .responsive-table th,
  .responsive-table td {
```

```
    padding: 0.5rem 0.3rem;
    white-space: nowrap;
  }

  .mobile-search {
    flex-direction: column;
    gap: 0.5rem;
  }

  .mobile-search input {
    width: 100%;
    margin-bottom: 0.5rem;
  }

  .mobile-search button {
    width: 100%;
  }

  .modal-responsive {
    width: 95% !important;
    max-width: 95% !important;
    margin: 1rem !important;
    padding: 1rem !important;
  }

  .tab-buttons {
    flex-direction: column;
    gap: 0.5rem;
  }

  .tab-buttons button {
    width: 100%;
  }

  .patient-details-responsive {
    display: block;
    overflow-x: auto;
  }

  .patient-details-responsive table {
    min-width: 600px;
  }

  .btn-responsive {
    padding: 0.4rem 0.8rem;
    font-size: 0.8rem;
    min-width: 70px;
  }
}

@media (max-width: 480px) {
  .dashboard-main {
    padding: 0.5rem;
  }

  .card {
    margin-bottom: 1rem;
    padding: 1rem;
  }

  .card-header {
    font-size: 1.2rem;
    margin-bottom: 0.8rem;
  }

  .responsive-table th,
  .responsive-table td {
    padding: 0.4rem 0.2rem;
    font-size: 0.7rem;
  }

  .btn-responsive {
    padding: 0.3rem 0.6rem;
```

```
      font-size: 0.7rem;
      min-width: 60px;
    }

    .modal-responsive {
      width: 98% !important;
      max-width: 98% !important;
      margin: 0.5rem !important;
      padding: 0.8rem !important;
    }

    .pdf-viewer-responsive {
      height: 300px !important;
    }
  }

  @media (min-width: 769px) and (max-width: 1024px) {
    .dashboard-main {
      padding: 1.5rem;
    }

    .modal-responsive {
      width: 85% !important;
      max-width: 85% !important;
    }
  }

  @media (min-width: 1025px) {
    .dashboard-main {
      padding: 2rem;
    }
  }
`}</style>

{/* Welcome Section */}
<div
  className="card"
  style={{ textAlign: "center", border: "1px solid #cc5500" }}
>
  <h2 className="card-header">Welcome, Doctor</h2>
  <p
    style={{
      fontStyle: "italic",
      color: "var(--text-secondary)",
      fontSize: "clamp(0.9rem, 2vw, 1.1rem)",
      lineHeight: "1.4",
    }}
  >
    {quote}
  </p>
</div>

{/* Today's Appointments */}
<div className="card" style={{ border: "1px solid #cc5500" }}>
  <h3 className="card-header">Today's Appointments</h3>
  {appointments.length === 0 ? (
    <p>No appointments for today.</p>
  ) : (
    <div className="table-responsive responsive-table">
      <table className="user-table">
        <thead>
          <tr>
            <th>MRN</th>
            <th>Patient Name</th>
            <th>Time</th>
            <th>Status</th>
          </tr>
        </thead>
        <tbody>
          {appointments.map((appt, index) => (
            <tr key={index}>
              <td>{appt.mrn}</td>
              <td>{appt.patientName}</td>
```

```
              <td>{appt.time}</td>
              <td>{appt.status}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  )}
</div>

{/* MRN Search */}
<div
  className="card"
  ref={scrollRef}
  style={{ border: "1px solid #cc5500" }}
>
  <h3 className="card-header">Search Patient by MRN</h3>
  <div
    className="mobile-search"
    style={{
      display: "flex",
      gap: "1rem",
      flexWrap: "wrap",
      alignItems: "center",
    }}
  >
    <input
      type="text"
      value={searchMRN}
      onChange={(e) => setSearchMRN(e.target.value.toUpperCase())}
      onKeyDown={(e) => {
        if (e.key === "Enter") {
          handleSearch();
        }
      }}
      placeholder="Enter MRN"
      className="search-input"
      style={{
        flex: 1,
        minWidth: "200px",
        padding: "0.8rem",
        fontSize: "1rem",
        height: "auto",
        boxSizing: "border-box",
      }}
    />

    <button
      className="btn btn-primary btn-responsive"
      onClick={handleSearch}
      style={{
        minWidth: "100px",
        padding: "0.8rem 1.5rem",
        height: "auto",
        boxSizing: "border-box",
        fontSize: "1rem",
      }}
    >
      Search
    </button>
  </div>
</div>

{/* Patient Details Table */}
{patientData && (
  <div className="card" style={{ border: "1px solid #cc5500" }}>
    <h3 className="card-header">Patient Details</h3>
    <div className="table-responsive patient-details-responsive">
      <table className="user-table">
        <thead>
          <tr>
            <th>Client Name</th>
            <th>Reference Number</th>
```

```
        <th>Next Visit</th>
        <th style={{ textAlign: "center" }}>Last Visit</th>
        <th style={{ textAlign: "center" }}>Records / Documents</th>
        <th style={{ textAlign: "center" }}>Assign Tests</th>
        <th style={{ textAlign: "center" }}>Assign Counselor</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>{patientData.clientName}</td>
        <td>{patientData.referenceNumber}</td>

        <td>
          {patientData?.newVisit?.date
            ? `${patientData.newVisit.date} at ${patientData.newVisit.time}`
            : "N/A"}
        </td>

        <td>
          <center>
            <button
              className="btn btn-primary btn-responsive"
              onClick={() => setShowLastVisitModal(true)}
            >
              View Last Visit
            </button>
          </center>
        </td>
        <td>
          <center>
            <button
              className="btn btn-primary btn-responsive"
              onClick={() => setShowDocumentModal(true)}
            >
              View Documents
            </button>
          </center>
        </td>
        <td>
          <center>
            <button
              className="btn btn-primary btn-responsive"
              onClick={() => setShowAssignTestModal(true)}
            >
              Assign Test
            </button>
          </center>
        </td>
        <td>
          <center>
            {" "}
            <button
              className="btn btn-primary btn-responsive"
              onClick={() => setShowCounselorModal(true)}
              style={{ marginTop: "0.5rem" }}
            >
              Refer to Counselor
            </button>
          </center>
        </td>
      </tr>
    </tbody>
  </table>
</div>
{/* Approval Buttons */}
{/* Approval Buttons */}
{/* Approval Buttons with Undo Confirmation */}
<div style={{ marginTop: "1.5rem", textAlign: "center" }}>
  {["nutrition", "diet", "physio"].map((type) => {
    const isApproved = patientData.approvals?.[type];
    const typeLabel = {
      nutrition: "Neutrogenomic",
      diet: "Dietician",
```

```
      physio: "Physiotherapist",
    };

    return (
      <button
        key={type}
        className="btn btn-primary btn-responsive"
        onClick={() => {
          if (isApproved) {
            showUndoApprovalToast(
              typeLabel[type],
              () => {
                const updatedApproval = {
                  ...patientData.approvals,
                  [type]: false,
                };

                fetch(
                  `http://localhost:3001/patients_detailed/${patientData.id}`,
                  {
                    method: "PATCH",
                    headers: {
                      "Content-Type": "application/json",
                    },
                    body: JSON.stringify({
                      approvals: updatedApproval,
                    }),
                  }
                )
                  .then((res) => {
                    if (!res.ok)
                      throw new Error("Failed to update approval");
                    return res.json();
                  })
                  .then((data) => {
                    toast.success(
                      `${typeLabel[type]} deassigned successfully.`
                    );
                    setPatientData(data);
                  })
                  .catch(() => {
                    toast.error("Failed to update approval status.");
                  });
              },
              theme
            );
          } else {
            handleApproval(type);
          }
        }}
        style={{ margin: "0.5rem" }}
      >
        {isApproved
          ? `${typeLabel[type]} Approved ? (Click to Undo)`
          : `Approve for ${typeLabel[type]}'s`}
      </button>
    );
  })}
</div>

{(patientData?.approvals?.physio ||
  patientData?.approvals?.diet ||
  patientData?.approvals?.nutrition) && (
  <div
    style={{
      background: "--var(--bg-primary)",
      color: "--var(--text-primary)",
      border: "1px solid #cc5500",
      borderRadius: "8px",
      padding: "1rem",
      marginTop: "1.5rem",
      textAlign: "left",
    }}
```

```
  >
    <h4 style={{ color: "#ff6b35", marginBottom: "0.5rem" }}>
      <strong>Patient Assigned</strong>
    </h4>
    <p style={{ marginBottom: "0.5rem" }}>
      <strong>{patientData.clientName}</strong> is currently assigned
      to:
    </p>
    <ul style={{ listStyle: "none", paddingLeft: 0 }}>
      {patientData.approvals.physio && (
        <li
          style={{
            display: "flex",
            alignItems: "center",
            gap: "0.5rem",
          }}
        >
          <CheckCircle size={18} color="green" />
          <strong>Physio</strong>
        </li>
      )}
      {patientData.approvals.diet && (
        <li
          style={{
            display: "flex",
            alignItems: "center",
            gap: "0.5rem",
          }}
        >
          <CheckCircle size={18} color="green" />
          <strong>Dietitian</strong>
        </li>
      )}
      {patientData.approvals.nutrition && (
        <li
          style={{
            display: "flex",
            alignItems: "center",
            gap: "0.5rem",
          }}
        >
          <CheckCircle size={18} color="green" />
          <strong>Neutrogenomic</strong>
        </li>
      )}
    </ul>
  </div>
)}
{(patientData?.approvals?.physio ||
  patientData?.approvals?.diet ||
  patientData?.approvals?.nutrition) && (
  <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
    <button
      className="btn btn-primary"
      style={{
        backgroundColor: "#ff6b35",
        border: "none",
        padding: "0.8rem 2rem",
        fontWeight: "bold",
        borderRadius: "6px",
        fontSize: "1rem",
      }}
      onClick={() => {
        toast.success("Patient Assignment Completed!");
        setPatientData(null); // Clear search result
        setSearchMRN(""); // Clear input
        scrollRef.current?.scrollIntoView({ behavior: "smooth" }); // Scroll back
      }}
    >
      Click To Proceed
    </button>
  </div>
)}
```

```
  </div>
)}
{/* Assign Test Modal */}
{showAssignTestModal && (
  <div
    onClick={() => setShowAssignTestModal(false)}
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
      backgroundColor: "rgba(0,0,0,0.6)",
      zIndex: 9999,
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      padding: "1rem",
      boxSizing: "border-box",
    }}
  >
    <div
      onClick={(e) => e.stopPropagation()}
      className="modal-responsive"
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        borderRadius: "8px",
        maxWidth: "500px",
        width: "90%",
        border: "1px solid #cc5500",
        boxSizing: "border-box",
      }}
    >
      <h2
        style={{
          marginBottom: "1.5rem",
          fontSize: "clamp(1.2rem, 3vw, 1.5rem)",
          color: "var(--text-primary)",
        }}
      >
        Assign Test to {patientData?.clientName}
      </h2>

      <form
        onSubmit={(e) => {
          e.preventDefault();
          handleAssignTest(); // Uses only testType and testName now
        }}
        style={{ display: "flex", flexDirection: "column", gap: "1rem" }}
      >
        {/* Test Type */}
        <div>
          <label
            style={{
              display: "block",
              marginBottom: "0.5rem",
              fontWeight: "bold",
              color: "var(--text-primary)",
            }}
          >
            Test Type *
          </label>
          <select
            value={testForm.testType}
            onChange={(e) =>
              setTestForm({
                ...testForm,
                testType: e.target.value,
                testName: "", // Reset test name when type changes
              })
            }
            required
```

```
    style={{
      width: "100%",
      padding: "0.8rem",
      border: "1px solid #ddd",
      borderRadius: "4px",
      fontSize: "1rem",
      backgroundColor: "var(--bg-secondary)",
      color: "var(--text-primary)",
    }}
  >
    <option value="">Select Test Type</option>
    {testCategories.map((cat) => (
      <option key={cat.type} value={cat.type}>
        {cat.type}
      </option>
    ))}
  </select>
</div>

{/* Test Name */}
<div>
  <label
    style={{
      display: "block",
      marginBottom: "0.5rem",
      fontWeight: "bold",
      color: "var(--text-primary)",
    }}
  >
    Test Name *
  </label>
  <select
    value={testForm.testName}
    onChange={(e) =>
      setTestForm({ ...testForm, testName: e.target.value })
    }
    required
    style={{
      width: "100%",
      padding: "0.8rem",
      border: "1px solid #ddd",
      borderRadius: "4px",
      fontSize: "1rem",
      backgroundColor: "var(--bg-secondary)",
      color: "var(--text-primary)",
    }}
  >
    <option value="">Select Test Name</option>
    {testCategories
      .find((cat) => cat.type === testForm.testType)
      ?.subTests.map((name, index) => (
        <option key={index} value={name}>
          {name}
        </option>
      ))}
  </select>
</div>

{/* Submit and Cancel Buttons */}
<div
  style={{
    display: "flex",
    justifyContent: "flex-end",
    gap: "1rem",
    marginTop: "1rem",
  }}
>
  <button
    type="button"
    onClick={() => setShowAssignTestModal(false)}
    style={{
      padding: "0.6rem 1.2rem",
      backgroundColor: "var(--bg-secondary)",
```

```
              color: "var(--text-primary)",
              border: "1px solid #ccc",
              borderRadius: "4px",
              cursor: "pointer",
            }}
          >
            Cancel
          </button>
          <button
            type="submit"
            style={{
              padding: "0.6rem 1.2rem",
              backgroundColor: "#cc5500",
              color: "#fff",
              border: "none",
              borderRadius: "4px",
              fontWeight: "bold",
              cursor: "pointer",
            }}
          >
            Assign
          </button>
        </div>
      </form>
    </div>
  </div>
)}

{showCounselorModal && (
  <div
    onClick={() => setShowCounselorModal(false)}
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
      backgroundColor: "rgba(0,0,0,0.6)",
      zIndex: 9999,
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      padding: "1rem",
      boxSizing: "border-box",
    }}
  >
    <div
      onClick={(e) => e.stopPropagation()}
      className="modal-responsive"
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        borderRadius: "8px",
        maxWidth: "500px",
        width: "100%",
        border: "1px solid #cc5500",
        boxSizing: "border-box",
      }}
    >
      <h2 style={{ marginBottom: "1.5rem" }}>Refer to Counselor</h2>

      <form
        onSubmit={(e) => {
          e.preventDefault();
          if (!selectedCounselor) {
            toast.error("Please select a counselor type.");
            return;
          }
          toast.success(`Referred to ${selectedCounselor}`);
          setShowCounselorModal(false);
          setSelectedCounselor("");
          setCounselorNote("");
        }}
```

```
        style={{ display: "flex", flexDirection: "column", gap: "1rem" }}
      >
        <select
          value={selectedCounselor}
          onChange={(e) => setSelectedCounselor(e.target.value)}
          required
          style={{
            padding: "0.8rem",
            border: "1px solid #ccc",
            borderRadius: "4px",
            backgroundColor: "var(--bg-secondary)",
            color: "var(--text-primary)",
          }}
        >
          <option value="">Select Counselor Type</option>
          {counselorTypes.map((item) => (
            <option key={item.id} value={item.type}>
              {item.type}
            </option>
          ))}
        </select>

        <textarea
          placeholder="Reason or notes (optional)"
          value={counselorNote}
          onChange={(e) => setCounselorNote(e.target.value)}
          rows={3}
          style={{
            padding: "0.8rem",
            border: "1px solid #ccc",
            borderRadius: "4px",
            backgroundColor: "var(--bg-secondary)",
            color: "var(--text-primary)",
            resize: "vertical",
          }}
        />

        <div
          style={{
            display: "flex",
            justifyContent: "flex-end",
            gap: "1rem",
          }}
        >
          <button
            type="button"
            onClick={() => setShowCounselorModal(false)}
            className="btn btn-responsive"
            style={{
              border: "1px solid #ccc",
              backgroundColor: "var(--bg-secondary)",
              color: "var(--text-primary)",
            }}
          >
            Cancel
          </button>
          <button
            type="submit"
            className="btn btn-primary btn-responsive"
            style={{ backgroundColor: "#cc5500", color: "#fff" }}
          >
            Assign
          </button>
        </div>
      </form>
    </div>
  </div>
)}

{/* Document Modal */}
{showDocumentModal && (
  <div
    onClick={() => setShowDocumentModal(false)}
```

```
  style={{
    position: "fixed",
    top: 0,
    left: 0,
    right: 0,
    bottom: 0,
    backgroundColor: "rgba(0,0,0,0.6)",
    zIndex: 9999,
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    padding: "1rem",
    boxSizing: "border-box",
  }}
>
  <div
    onClick={(e) => e.stopPropagation()}
    className="modal-responsive"
    style={{
      backgroundColor: "var(--bg-primary)",
      padding: "2rem",
      borderRadius: "8px",
      maxWidth: "70%",
      width: "90%",
      maxHeight: "90vh",
      overflow: "auto",
      border: "1px solid #cc5500",
      position: "relative",
      boxSizing: "border-box",
    }}
  >
    {/* Tabs */}
    <div
      className="tab-buttons"
      style={{
        display: "flex",
        gap: "1rem",
        marginBottom: "1rem",
        flexWrap: "wrap",
      }}
    >
      <button
        onClick={() => setActiveTab("medical")}
        className={`btn btn-responsive ${
          activeTab === "medical" ? "btn-active" : "btn-primary"
        }`}
        style={{ flex: "1" }}
      >
        Medical
      </button>
      <button
        onClick={() => setActiveTab("test")}
        className={`btn btn-responsive ${
          activeTab === "test" ? "btn-active" : "btn-primary"
        }`}
        style={{ flex: "1" }}
      >
        Test Record
      </button>
    </div>

    {/* Medical Tab */}
    {activeTab === "medical" && (
      <div className="responsive-table">
        <table className="user-table">
          <thead>
            <tr>
              <th>Date</th>
              <th>Time</th>
              <th style={{ textAlign: "center" }}>Clinical Note</th>
              <th style={{ textAlign: "center" }}>Prescription</th>
            </tr>
          </thead>
```

```
      <tbody>
        {medicalData.map((item, idx) => (
          <tr key={idx}>
            <td>{item.date}</td>
            <td>{item.time}</td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setPdfUrl(item.clinicalNote)}
                >
                  View
                </button>
              </center>
            </td>
            <td>
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setPdfUrl(item.prescription)}
                >
                  View
                </button>
              </center>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{/* Test Record Tab */}
{activeTab === "test" && (
  <div className="responsive-table">
    <table className="user-table">
      <thead>
        <tr>
          <th>Date</th> {/* New column added here */}
          <th>Test Name</th>
          <th style={{ textAlign: "center" }}>View</th>
        </tr>
      </thead>
      <tbody>
        {testRecords.map((item, idx) => (
          <tr key={idx}>
            <td>{item.date}</td> {/* New date data rendered here */}
            <td>{item.name}</td>
            <td>
              <center>
                <button
                  className="btn btn-primary btn-responsive"
                  onClick={() => setPdfUrl(item.url)}
                >
                  View
                </button>
              </center>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{/* PDF Viewer */}
{pdfUrl && (
  <div
    style={{
      marginTop: "1rem",
      padding: "1rem",
      background: "#f9f9f9",
      borderRadius: "8px",
```

```
            position: "relative",
          }}
        >
          {/* Top Action Bar */}
          <div
            style={{
              display: "flex",
              justifyContent: "space-between",
              alignItems: "center",
              marginBottom: "1rem",
              flexWrap: "wrap",
              gap: "0.5rem",
            }}
          >
            {/* Close Button */}
            <button
              onClick={() => setPdfUrl("")}
              className="btn btn-primary"
              style={{
                fontWeight: "bold",
                fontSize: "1.2rem",
                lineHeight: "1",
                padding: "0.4rem 0.8rem",
                cursor: "pointer",
              }}
              title="Close PDF Viewer"
            >
              ×
            </button>

            {/* Open in New Tab */}
            <a
              href={pdfUrl}
              target="_blank"
              rel="noopener noreferrer"
              className="btn btn-primary btn-responsive"
              style={{
                textDecoration: "none",
                display: "inline-block",
              }}
            >
              ? Open in New Tab
            </a>
          </div>

          {/* PDF Iframe Viewer */}
          <iframe
            src={pdfUrl}
            title="PDF Viewer"
            width="100%"
            height="500px"
            className="pdf-viewer-responsive"
            style={{
              border: "1px solid #ddd",
              borderRadius: "4px",
              minHeight: "300px",
            }}
          />
        </div>
      )}

      <center>
        <button
          style={{ marginTop: "2rem", marginBottom: "0rem" }}
          onClick={() => setShowDocumentModal(false)}
          className="btn btn-primary btn-responsive"
        >
          Close
        </button>
      </center>
    </div>
  </div>
)}
```

```
{/* Last Visit Modal */}
{showLastVisitModal && (
  <div
    onClick={() => setShowLastVisitModal(false)}
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      right: 0,
      bottom: 0,
      backgroundColor: "rgba(0,0,0,0.6)",
      zIndex: 9999,
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      padding: "1rem",
      boxSizing: "border-box",
    }}
  >
    <div
      onClick={(e) => e.stopPropagation()}
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        borderRadius: "8px",
        maxWidth: "600px",
        width: "90%",
        maxHeight: "90vh",
        overflow: "auto",
        position: "relative",
        border: "1px solid #CC5500",
        boxSizing: "border-box",
      }}
    >
      <button
        onClick={() => setShowLastVisitModal(false)}
        style={{
          position: "absolute",
          top: "10px",
          right: "10px",
          background: "#cc5500",
          border: "none",
          padding: "0.5rem 1rem",
          fontWeight: "bold",
          borderRadius: "4px",
          color: "white",
          cursor: "pointer",
        }}
      >
        Close
      </button>

      <h2
        style={{
          marginBottom: "1rem",
          fontSize: "clamp(1.2rem, 3vw, 1.5rem)",
          paddingRight: "3rem",
        }}
      >
        Last Visit Summary
      </h2>

      <div style={{ lineHeight: "1.6" }}>
        <p>
          <strong>Date:</strong> {patientData.lastVisit?.date}
        </p>
        <p>
          <strong>Time:</strong> {patientData.lastVisit?.time}
        </p>
        <p>
          <strong>Summary:</strong> {patientData.lastVisit?.summary}
        </p>
```

```
              <h3
                style={{
                  marginTop: "1rem",
                  fontSize: "clamp(1rem, 2.5vw, 1.2rem)",
                }}
              >
                Vitals
              </h3>
              <ul style={{ paddingLeft: "1.2rem" }}>
                <li>
                  <strong>BP:</strong> {patientData.lastVisit?.vitals?.BP}
                </li>
                <li>
                  <strong>Pulse:</strong> {patientData.lastVisit?.vitals?.Pulse}
                </li>
                <li>
                  <strong>Temp:</strong> {patientData.lastVisit?.vitals?.Temp}
                </li>
              </ul>

              <p>
                <strong>Doctor Notes:</strong>
                <br />
                {patientData.lastVisit?.doctorNotes}
              </p>
              <p>
                <strong>Recommendation:</strong>
                <br />
                {patientData.lastVisit?.recommendation}
              </p>

              {patientData.lastVisit?.prescription && (
                <div style={{ marginTop: "1rem" }}>
                  <strong>Prescription:</strong>
                  <br />
                  <a
                    href={patientData.lastVisit.prescription}
                    target="_blank"
                    rel="noopener noreferrer"
                    className="btn btn-primary btn-responsive"
                    style={{ marginTop: "0.5rem" }}
                  >
                    View PDF
                  </a>
                </div>
              )}
            </div>
          </div>
        </div>
      )}
    </div>
  );
};
export default DoctorDashboardHome;
```

File: src\dashboard\doctor\DoctorsAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";

const DoctorsAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_URL = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_URL}/role/doctor`)
      .then((res) => {
        if (!res.ok) throw new Error("Failed to fetch appointments");
```

```
      return res.json();
    })
    .then((data) => setAppointments(data))
    .catch((err) => {
      console.error("Error loading doctor appointments:", err);
      toast.error("Unable to load appointments.");
    });
}, []);

const updateAppointment = (updated) => {
  setAppointments((prev) =>
    prev.map((a) => (a.id === updated.id ? updated : a))
  );
  fetch(`${API_URL}/${updated.id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Failed to update appointment"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  let nextStatus;

  switch (current.status) {
    case "Pending":
      nextStatus = "Approved";
      break;
    case "Approved":
      nextStatus = "Completed";
      break;
    default:
      nextStatus = "Pending";
  }

  const updated = { ...current, status: nextStatus };
  updateAppointment(updated);
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(
    100000000 + Math.random() * 900000000
  )}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = () => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_URL}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Appointment deleted.");
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
```

```
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Doctor Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online"
                        ? "badge-orange"
                        : "badge-blue"
                    }`}
                  >
                    {a.appointmentType}
                  </span>
                </td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => setSelectedNote(a.notes)}
                  >
                    View
                  </button>
                </td>
                <td>
                  {a.appointmentType === "Online" ? (
                    <div
                      style={{
                        display: "flex",
```

```
                    alignItems: "center",
                    gap: "8px",
                    flexDirection: "column",
                  }}
                >
                  <input
                    type="text"
                    value={a.meetingLink || ""}
                    onChange={(e) =>
                      handleMeetingLinkChange(a.id, e.target.value)
                    }
                    placeholder="Enter or generate"
                    className="search-input"
                    style={{
                      width: "180px",
                      padding: "5px 8px",
                      fontSize: "14px",
                      border: "1px solid #ccc",
                      borderRadius: "6px",
                    }}
                  />
                  <div style={{ display: "flex", gap: "6px" }}>
                    <button
                      className="btn btn-primary"
                      onClick={() => handleGenerateLink(a.id)}
                    >
                      Auto
                    </button>
                    <button
                      className="btn btn-primary"
                      style={{ backgroundColor: "#f44336", color: "#fff" }}
                      onClick={() => handleMeetingLinkChange(a.id, "")}
                    >
                      Clear
                    </button>
                  </div>
                </div>
              ) : (
                "-"
              )}
            </td>
            <td>
              <span
                className={`status-badge ${
                  a.status === "Completed"
                    ? "badge-green"
                    : a.status === "Approved"
                    ? "badge-blue"
                    : "badge-yellow"
                }`}
                onClick={() => handleStatusToggle(a.id)}
                style={{ cursor: "pointer" }}
              >
                {a.status}
              </span>
            </td>
            <td>
              <button
                className="btn btn-primary"
                onClick={() => handleDelete(a.id)}
              >
                Delete
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{selectedNote && (
  <div className="modal-overlay">
```

```
          <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
            <h2>Full Notes</h2>
            <p>{selectedNote}</p>
            <div className="center-btn">
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setSelectedNote(null)}
                >
                  Close
                </button>
              </center>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default DoctorsAppointments;
```

File: src\dashboard\doctor\DoctorSlotBooking.css

```
/* CSS Custom Properties for Light/Dark Mode */
:root {
  --primary-color: #cc5500;
  --primary-hover: #b84d00;
  --secondary-color: #6c757d;
  --success-color: #28a745;
  --danger-color: #dc3545;
  --warning-color: #ffc107;
  --info-color: #17a2b8;

  --bg-primary: #ffffff;
  --bg-secondary: #f8f9fa;
  --bg-tertiary: #e9ecef;
  --bg-card: #ffffff;
  --bg-modal: #ffffff;

  --text-primary: #212529;
  --text-secondary: #6c757d;
  --text-muted: #868e96;
  --text-inverse: #ffffff;

  --border-color: #dee2e6;
  --border-primary: var(--primary-color);
  --border-light: #e9ecef;

  --shadow-sm: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
  --shadow-md: 0 0.25rem 0.5rem rgba(0, 0, 0, 0.1);
  --shadow-lg: 0 0.5rem 1rem rgba(0, 0, 0, 0.15);
  --shadow-xl: 0 1rem 2rem rgba(0, 0, 0, 0.2);

  --border-radius: 0.25rem;
  --border-radius-lg: 0.375rem;
  --border-radius-xl: 0.5rem;

  --transition: all 0.15s ease-in-out;
  --transition-fast: all 0.1s ease-in-out;
}

/* Dark Mode Variables */
.dark,
[data-theme="dark"] {
  --bg-primary: #1a1a1a;
  --bg-secondary: #2d2d2d;
  --bg-tertiary: #404040;
  --bg-card: #2d2d2d;
  --bg-modal: #2d2d2d;

  --text-primary: #ffffff;
  --text-secondary: #b3b3b3;
  --text-muted: #999999;
```

```
  --border-color: #404040;
  --border-light: #555555;

  --shadow-sm: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.3);
  --shadow-md: 0 0.25rem 0.5rem rgba(0, 0, 0, 0.4);
  --shadow-lg: 0 0.5rem 1rem rgba(0, 0, 0, 0.5);
  --shadow-xl: 0 1rem 2rem rgba(0, 0, 0, 0.6);
}

/* Base Styles */
* {
  box-sizing: border-box;
}

.doctor-slot-container {
  min-height: 100vh;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  padding: 0.75rem;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", sans-serif;
  line-height: 1.5;
}

.main-content {
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  gap: 1.25rem;
}

/* Page Header */
.page-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
  gap: 0.75rem;
  margin-bottom: 0.5rem;
}

.page-title {
  font-size: clamp(1.5rem, 3.5vw, 2rem);
  font-weight: 700;
  color: var(--text-primary);
  margin: 0;
  display: flex;
  align-items: center;
  gap: 0.5rem;
}

.title-icon {
  font-size: 1em;
}

/* Card Styles */
.card {
  background-color: var(--bg-card);
  border: 1px solid var(--border-color);
  border-radius: var(--border-radius-lg);
  box-shadow: var(--shadow-sm);
  overflow: hidden;
  transition: var(--transition);
}

.card:hover {
  box-shadow: var(--shadow-md);
}

.card-header {
  background-color: var(--bg-secondary);
```

```
  border-bottom: 1px solid var(--border-color);
  padding: 1rem;
}

.card-title {
  font-size: clamp(1.1rem, 2.5vw, 1.4rem);
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
  display: flex;
  align-items: center;
  gap: 0.5rem;
}

/* Form Styles */
.slot-form {
  padding: 1.25rem;
}

.date-inputs-section {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1rem;
  margin-bottom: 1.5rem;
}

.input-group {
  display: flex;
  flex-direction: column;
  gap: 0.4rem;
}

.input-label {
  font-weight: 600;
  color: var(--text-secondary);
  font-size: 0.8rem;
  display: flex;
  align-items: center;
  gap: 0.4rem;
}

.label-icon {
  font-size: 0.9em;
}

.form-input {
  padding: 0.6rem 0.8rem;
  border: 1px solid var(--border-color);
  border-radius: var(--border-radius);
  background-color: var(--bg-primary);
  color: var(--text-primary);
  font-size: 0.9rem;
  transition: var(--transition);
}

.form-input:focus {
  outline: none;
  border-color: var(--primary-color);
  box-shadow: 0 0 0 0.15rem rgba(204, 85, 0, 0.2);
}

/* Button Styles */
.btn {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 0.4rem;
  padding: 0.6rem 1.2rem;
  font-size: 0.8rem;
  font-weight: 500;
  border: 1px solid transparent;
  border-radius: var(--border-radius);
  cursor: pointer;
```

```
  transition: var(--transition);
  text-decoration: none;
  white-space: nowrap;
  min-height: 36px;
}

.btn:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

.btn-large {
  padding: 0.75rem 1.5rem;
  font-size: 0.9rem;
  min-height: 40px;
}

.btn-small {
  padding: 0.4rem 0.8rem;
  font-size: 0.75rem;
  min-height: 32px;
}

.btn-primary {
  background-color: var(--primary-color);
  color: var(--text-inverse);
  border-color: var(--primary-color);
}

.btn-primary:hover:not(:disabled) {
  background-color: var(--primary-hover);
  border-color: var(--primary-hover);
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
}

.btn-secondary {
  background-color: var(--secondary-color);
  color: var(--text-inverse);
  border-color: var(--secondary-color);
}

.btn-secondary:hover:not(:disabled) {
  background-color: #5a6268;
  border-color: #545b62;
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
}

.btn-danger {
  background-color: var(--danger-color);
  color: var(--text-inverse);
  border-color: var(--danger-color);
}

.btn-danger:hover:not(:disabled) {
  background-color: #c82333;
  border-color: #bd2130;
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
}

.btn-warning {
  background-color: var(--warning-color);
  color: #212529;
  border-color: var(--warning-color);
}

.btn-warning:hover:not(:disabled) {
  background-color: #e0a800;
  border-color: #d39e00;
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
```

```
}

.btn-info {
  background-color: var(--info-color);
  color: var(--text-inverse);
  border-color: var(--info-color);
}

.btn-info:hover:not(:disabled) {
  background-color: #138496;
  border-color: #117a8b;
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
}

.btn-outline {
  background-color: transparent;
  color: var(--primary-color);
  border-color: var(--primary-color);
}

.btn-outline:hover:not(:disabled) {
  background-color: var(--primary-color);
  color: var(--text-inverse);
  transform: translateY(-1px);
  box-shadow: var(--shadow-sm);
}

.btn-icon {
  font-size: 0.9em;
}

/* Availability Section */
.availability-section {
  margin-bottom: 1.5rem;
}

.section-title {
  font-size: 1.1rem;
  font-weight: 600;
  color: var(--text-primary);
  margin-bottom: 0.75rem;
  padding-bottom: 0.4rem;
  border-bottom: 1px solid var(--border-light);
}

/* Table Styles */
.table-container {
  overflow-x: auto;
  border-radius: var(--border-radius);
  box-shadow: var(--shadow-sm);
}

.availability-table,
.summary-table,
.modal-table {
  width: 100%;
  border-collapse: collapse;
  background-color: var(--bg-card);
  font-size: 0.85rem;
}

.availability-table th,
.summary-table th,
.modal-table th {
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  font-weight: 600;
  padding: 0.75rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
  position: sticky;
  top: 0;
```

```
  z-index: 10;
}

.availability-table td,
.summary-table td,
.modal-table td {
  padding: 0.75rem;
  border-bottom: 1px solid var(--border-light);
  vertical-align: top;
}

.availability-table tr:hover,
.summary-table tr:hover,
.modal-table tr:hover {
  background-color: var(--bg-tertiary);
}

.available-row {
  background-color: rgba(40, 167, 69, 0.08);
}

.unavailable-row {
  background-color: rgba(220, 53, 69, 0.08);
}

/* Day Cell */
.day-cell {
  font-weight: 600;
  color: var(--text-primary);
}

.day-name {
  display: flex;
  align-items: center;
  gap: 0.4rem;
}

/* Checkbox Styles */
.checkbox-cell {
  text-align: center;
}

.checkbox-wrapper {
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 0.4rem;
  cursor: pointer;
}

.availability-checkbox {
  display: none;
}

.checkbox-custom {
  width: 20px;
  height: 20px;
  border: 1px solid var(--border-color);
  border-radius: var(--border-radius);
  background-color: var(--bg-primary);
  position: relative;
  transition: var(--transition);
}

.availability-checkbox:checked + .checkbox-custom {
  background-color: var(--primary-color);
  border-color: var(--primary-color);
}

.availability-checkbox:checked + .checkbox-custom::after {
  content: "?";
  position: absolute;
  top: 50%;
```

```
  left: 50%;
  transform: translate(-50%, -50%);
  color: white;
  font-weight: bold;
  font-size: 12px;
}

.checkbox-label {
  font-weight: 500;
  color: var(--text-secondary);
  font-size: 0.8rem;
}

/* Time Blocks */
.time-blocks-container {
  display: flex;
  flex-direction: column;
  gap: 0.75rem;
}

.time-block {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
  padding: 0.75rem;
  background-color: var(--bg-secondary);
  border-radius: var(--border-radius);
  border: 1px solid var(--border-light);
}

.time-inputs {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  flex-wrap: wrap;
}

.time-input {
  flex: 1;
  min-width: 100px;
  padding: 0.4rem 0.6rem;
  border: 1px solid var(--border-color);
  border-radius: var(--border-radius);
  background-color: var(--bg-primary);
  color: var(--text-primary);
  font-size: 0.8rem;
  transition: var(--transition);
}

.time-input:focus {
  outline: none;
  border-color: var(--primary-color);
  box-shadow: 0 0 0 0.1rem rgba(204, 85, 0, 0.2);
}

.time-separator {
  font-weight: 500;
  color: var(--text-secondary);
  white-space: nowrap;
  font-size: 0.8rem;
}

.remove-btn {
  align-self: flex-start;
}

.add-time-btn {
  align-self: flex-start;
  margin-top: 0.25rem;
}

.not-available-message {
  display: flex;
```

```
  align-items: center;
  gap: 0.4rem;
  color: var(--text-muted);
  font-style: italic;
  padding: 0.75rem;
  background-color: var(--bg-tertiary);
  border-radius: var(--border-radius);
  font-size: 0.8rem;
}

/* Form Actions */
.form-actions {
  display: flex;
  gap: 0.75rem;
  justify-content: center;
  margin-top: 1.5rem;
  flex-wrap: wrap;
}

/* Summary Table Specific */
.serial-cell {
  text-align: center;
  font-weight: 600;
  width: 80px;
}

.serial-number {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  width: 28px;
  height: 28px;
  background-color: var(--primary-color);
  color: var(--text-inverse);
  border-radius: 50%;
  font-size: 0.8rem;
  font-weight: 600;
}

.date-range-cell {
  font-weight: 500;
  min-width: 200px;
}

.date-range {
  display: flex;
  align-items: center;
  gap: 0.5rem;
  flex-wrap: nowrap;
  white-space: nowrap;
}

.date-from,
.date-to {
  padding: 0.3rem 0.6rem;
  background-color: var(--bg-secondary);
  border-radius: var(--border-radius);
  font-size: 0.8rem;
  font-weight: 500;
  border: 1px solid var(--border-light);
}

.date-separator {
  color: var(--text-muted);
  font-weight: 500;
  font-size: 0.8rem;
  margin: 0 0.2rem;
}

/* Action Buttons */
.actions-cell {
  text-align: center;
  width: 250px;
```

```
}

.action-buttons {
  display: flex;
  gap: 0.5rem;
  justify-content: center;
  flex-wrap: nowrap;
}

/* Summary Table Layout */
.summary-table {
  table-layout: fixed;
  width: 100%;
}

.summary-table th:first-child {
  width: 80px;
}

.summary-table th:nth-child(2) {
  width: auto;
  min-width: 200px;
}

.summary-table th:last-child {
  width: 250px;
}

/* Modal Styles */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
  padding: 0.75rem;
  backdrop-filter: blur(2px);
}

.modal-content {
  background-color: var(--bg-modal);
  border-radius: var(--border-radius-xl);
  box-shadow: var(--shadow-lg);
  width: 100%;
  max-width: 700px;
  max-height: 85vh;
  overflow-y: auto;
  position: relative;
  animation: modalSlideIn 0.2s ease-out;
}

@keyframes modalSlideIn {
  from {
    opacity: 0;
    transform: translateY(-30px) scale(0.98);
  }
  to {
    opacity: 1;
    transform: translateY(0) scale(1);
  }
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem;
  border-bottom: 1px solid var(--border-color);
```

```
  background-color: var(--bg-secondary);
  border-radius: var(--border-radius-xl) var(--border-radius-xl) 0 0;
}

.modal-title {
  font-size: 1.1rem;
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
  display: flex;
  align-items: center;
  gap: 0.5rem;
}

.modal-close-btn {
  background: none;
  border: none;
  font-size: 1.1rem;
  cursor: pointer;
  padding: 0.4rem;
  border-radius: var(--border-radius);
  transition: var(--transition);
  color: var(--text-secondary);
}

.modal-close-btn:hover {
  background-color: var(--bg-tertiary);
  color: var(--text-primary);
}

.modal-body {
  padding: 1rem;
}

.date-range-info {
  display: flex;
  align-items: center;
  gap: 0.75rem;
  margin-bottom: 1rem;
  padding: 0.75rem;
  background-color: var(--bg-secondary);
  border-radius: var(--border-radius);
  flex-wrap: wrap;
}

.date-range-label {
  font-weight: 600;
  color: var(--text-secondary);
  font-size: 0.85rem;
}

.date-range-value {
  font-weight: 500;
  color: var(--primary-color);
  font-size: 0.95rem;
}

.modal-table-container {
  border-radius: var(--border-radius);
  overflow: hidden;
  box-shadow: var(--shadow-sm);
}

.availability-cell {
  padding: 0.6rem;
}

.time-slots {
  display: flex;
  flex-direction: column;
  gap: 0.4rem;
}
```

```
.time-slot-badge {
  display: inline-flex;
  align-items: center;
  gap: 0.4rem;
  padding: 0.4rem 0.6rem;
  background-color: var(--success-color);
  color: var(--text-inverse);
  border-radius: var(--border-radius);
  font-size: 0.8rem;
  font-weight: 500;
}

.time-icon {
  font-size: 0.9em;
}

.not-available-badge {
  display: inline-flex;
  align-items: center;
  gap: 0.4rem;
  padding: 0.4rem 0.6rem;
  background-color: var(--danger-color);
  color: var(--text-inverse);
  border-radius: var(--border-radius);
  font-size: 0.8rem;
  font-weight: 500;
}

.modal-footer {
  padding: 1rem;
  border-top: 1px solid var(--border-color);
  background-color: var(--bg-secondary);
  text-align: center;
  border-radius: 0 0 var(--border-radius-xl) var(--border-radius-xl);
}

/* Delete Confirmation */
.delete-confirmation {
  text-align: center;
  padding: 0.75rem;
}

.delete-confirmation p {
  margin-bottom: 0.75rem;
  color: var(--text-primary);
  font-weight: 500;
  font-size: 0.9rem;
}

.delete-confirmation-buttons {
  display: flex;
  justify-content: center;
  gap: 0.75rem;
  flex-wrap: wrap;
}

/* Responsive Design */
@media (max-width: 1024px) {
  .main-content {
    max-width: 100%;
  }

  .page-header {
    flex-direction: column;
    align-items: stretch;
    text-align: center;
  }

  .date-inputs-section {
    grid-template-columns: 1fr;
  }

  .time-inputs {
```

```
    flex-direction: column;
    align-items: stretch;
  }

  .time-input {
    min-width: unset;
  }
}

@media (max-width: 768px) {
  .doctor-slot-container {
    padding: 0.5rem;
  }

  .main-content {
    gap: 1rem;
  }

  .slot-form {
    padding: 1rem;
  }

  .card-header {
    padding: 0.75rem;
  }

  /* Mobile Table Styles */
  .availability-table,
  .summary-table {
    font-size: 0.8rem;
  }

  .availability-table thead,
  .summary-table thead {
    display: none;
  }

  .availability-table tr,
  .summary-table tr {
    display: block;
    margin-bottom: 0.75rem;
    border: 1px solid var(--border-color);
    border-radius: var(--border-radius);
    padding: 0.75rem;
    background-color: var(--bg-card);
    box-shadow: var(--shadow-sm);
  }

  .availability-table td,
  .summary-table td {
    display: block;
    padding: 0.4rem 0;
    border: none;
    text-align: left !important;
  }

  .availability-table td:before,
  .summary-table td:before {
    content: attr(data-label) ": ";
    font-weight: 600;
    color: var(--text-secondary);
    display: inline-block;
    min-width: 80px;
    margin-bottom: 0.2rem;
    font-size: 0.75rem;
  }

  .checkbox-wrapper {
    justify-content: flex-start;
  }

  .time-block {
    margin-top: 0.4rem;
```

```
  }

  .action-buttons {
    flex-direction: column;
    gap: 0.4rem;
  }

  .action-buttons .btn {
    width: 100%;
  }

  .form-actions {
    flex-direction: column;
  }

  .form-actions .btn {
    width: 100%;
  }

  .modal-content {
    margin: 0.5rem;
    max-width: calc(100vw - 1rem);
  }

  .modal-header {
    padding: 0.75rem;
  }

  .modal-body {
    padding: 0.75rem;
  }

  .modal-footer {
    padding: 0.75rem;
  }

  .date-range-info {
    flex-direction: column;
    align-items: flex-start;
    gap: 0.4rem;
  }
}

@media (max-width: 480px) {
  .page-title {
    font-size: 1.4rem;
  }

  .card-title {
    font-size: 1.1rem;
  }

  .btn {
    padding: 0.6rem 1rem;
    font-size: 0.75rem;
  }

  .btn-large {
    padding: 0.75rem 1.2rem;
    font-size: 0.85rem;
  }

  .time-slots {
    gap: 0.2rem;
  }

  .time-slot-badge {
    font-size: 0.75rem;
    padding: 0.3rem 0.5rem;
  }

  .delete-confirmation-buttons {
    flex-direction: column;
```

```
  }

  .delete-confirmation-buttons .btn {
    width: 100%;
  }
}

/* Print Styles */
@media print {
  .modal-overlay,
  .btn,
  .modal-close-btn,
  .action-buttons {
    display: none !important;
  }

  .doctor-slot-container {
    background: white !important;
    color: black !important;
    padding: 0;
  }

  .card {
    border: 1px solid #000 !important;
    box-shadow: none !important;
    page-break-inside: avoid;
  }

  .page-title {
    color: black !important;
  }
}

/* High Contrast Mode */
@media (prefers-contrast: high) {
  :root {
    --border-color: #000000;
    --text-secondary: #000000;
  }

  .btn {
    border-width: 2px;
  }

  .form-input {
    border-width: 2px;
  }
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  * {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
  }

  .modal-content {
    animation: none;
  }
}

/* Focus Styles for Accessibility */
.btn:focus,
.form-input:focus,
.time-input:focus,
.availability-checkbox:focus + .checkbox-custom {
  outline: 2px solid var(--primary-color);
  outline-offset: 1px;
}

/* Loading States */
.btn:disabled {
```

```
  position: relative;
  color: transparent;
}

.btn:disabled::after {
  content: "";
  position: absolute;
  width: 14px;
  height: 14px;
  top: 50%;
  left: 50%;
  margin-left: -7px;
  margin-top: -7px;
  border: 2px solid transparent;
  border-top-color: currentColor;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

/* Hover Effects */
.card:hover {
  transform: translateY(-1px);
}

.btn:hover:not(:disabled) {
  transform: translateY(-1px);
}

.time-slot-badge:hover {
  transform: scale(1.02);
}

/* Selection Styles */
::selection {
  background-color: var(--primary-color);
  color: var(--text-inverse);
}

/* Scrollbar Styles */
.table-container::-webkit-scrollbar,
.modal-content::-webkit-scrollbar {
  width: 6px;
  height: 6px;
}

.table-container::-webkit-scrollbar-track,
.modal-content::-webkit-scrollbar-track {
  background: var(--bg-secondary);
  border-radius: var(--border-radius);
}

.table-container::-webkit-scrollbar-thumb,
.modal-content::-webkit-scrollbar-thumb {
  background: var(--border-color);
  border-radius: var(--border-radius);
}

.table-container::-webkit-scrollbar-thumb:hover,
.modal-content::-webkit-scrollbar-thumb:hover {
  background: var(--text-muted);
}

/* Mobile responsive for summary table */
@media (max-width: 768px) {
```

```
  .summary-table tr {
    display: block;
    margin-bottom: 0.75rem;
    border: 1px solid var(--border-color);
    border-radius: var(--border-radius);
    padding: 0.75rem;
    background-color: var(--bg-card);
    box-shadow: var(--shadow-sm);
  }

  .summary-table td {
    display: block;
    padding: 0.4rem 0;
    border: none;
    text-align: left !important;
  }

  .summary-table td:before {
    content: attr(data-label) ": ";
    font-weight: 600;
    color: var(--text-secondary);
    display: inline-block;
    min-width: 80px;
    margin-bottom: 0.2rem;
    font-size: 0.75rem;
  }

  .date-range {
    flex-wrap: wrap;
    margin-top: 0.25rem;
  }

  .action-buttons {
    flex-direction: row;
    gap: 0.4rem;
    margin-top: 0.5rem;
  }

  .action-buttons .btn {
    flex: 1;
    min-width: 0;
  }
}

@media (max-width: 480px) {
  .action-buttons {
    flex-direction: column;
    gap: 0.4rem;
  }

  .action-buttons .btn {
    width: 100%;
  }

  .date-range {
    flex-direction: column;
    align-items: flex-start;
    gap: 0.25rem;
  }

  .date-separator {
    align-self: center;
  }
}

/* Summary Card Improvements */
.summary-card {
  margin-top: 1rem;
}

.slot-count {
  font-size: 0.8rem;
  font-weight: 400;
```

```
  color: var(--text-muted);
  margin-left: 0.5rem;
}

/* Desktop Table View */
.desktop-table-view {
  display: block;
}

.mobile-card-view {
  display: none;
}

/* Enhanced Summary Table */
.summary-table {
  table-layout: fixed;
  width: 100%;
  border-collapse: separate;
  border-spacing: 0;
  border-radius: var(--border-radius);
  overflow: hidden;
}

.summary-table th {
  background: linear-gradient(
    135deg,
    var(--bg-secondary) 0%,
    var(--bg-tertiary) 100%
  );
  color: var(--text-primary);
  font-weight: 600;
  padding: 1rem 0.75rem;
  text-align: center;
  border-bottom: 2px solid var(--border-color);
  font-size: 0.85rem;
  text-transform: uppercase;
  letter-spacing: 0.5px;
}

.summary-table th:first-child {
  width: 80px;
  text-align: center;
}

.summary-table th:nth-child(2) {
  width: auto;
  min-width: 250px;
  text-align: left;
}

.summary-table th:last-child {
  width: 280px;
  text-align: center;
}

.summary-row {
  transition: all 0.2s ease;
  border-bottom: 1px solid var(--border-light);
}

.summary-row:hover {
  background-color: var(--bg-tertiary);
  transform: translateY(-1px);
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

.summary-row:last-child {
  border-bottom: none;
}

/* Serial Cell Enhanced */
.serial-cell {
  text-align: center;
```

```
  padding: 1rem 0.75rem;
}

.serial-number {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  width: 32px;
  height: 32px;
  background: linear-gradient(
    135deg,
    var(--primary-color) 0%,
    var(--primary-hover) 100%
  );
  color: var(--text-inverse);
  border-radius: 50%;
  font-size: 0.85rem;
  font-weight: 700;
  box-shadow: 0 2px 4px rgba(204, 85, 0, 0.3);
}

/* Date Range Enhanced */
.date-range-cell {
  padding: 1rem 0.75rem;
}

.date-range-container {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
}

.date-range {
  display: flex;
  align-items: center;
  gap: 0.75rem;
  flex-wrap: nowrap;
}

.date-badge {
  padding: 0.4rem 0.8rem;
  background-color: var(--bg-secondary);
  border: 1px solid var(--border-light);
  border-radius: var(--border-radius);
  font-size: 0.8rem;
  font-weight: 500;
  font-family: "Courier New", monospace;
  white-space: nowrap;
  transition: var(--transition);
}

.date-from {
  background-color: rgba(40, 167, 69, 0.1);
  border-color: rgba(40, 167, 69, 0.3);
  color: var(--success-color);
}

.date-to {
  background-color: rgba(220, 53, 69, 0.1);
  border-color: rgba(220, 53, 69, 0.3);
  color: var(--danger-color);
}

.date-separator {
  color: var(--text-muted);
  font-weight: 600;
  font-size: 1rem;
}

.date-duration {
  font-size: 0.75rem;
  color: var(--text-muted);
  font-weight: 500;
```

```
  padding: 0.2rem 0.5rem;
  background-color: var(--bg-tertiary);
  border-radius: var(--border-radius);
  display: inline-block;
  width: fit-content;
}

/* Actions Enhanced */
.actions-cell {
  text-align: center;
  padding: 1rem 0.75rem;
}

.action-buttons {
  display: flex;
  gap: 0.5rem;
  justify-content: center;
  flex-wrap: nowrap;
}

.btn-small {
  padding: 0.5rem 0.75rem;
  font-size: 0.75rem;
  min-height: 34px;
  border-radius: var(--border-radius);
  font-weight: 500;
  transition: all 0.2s ease;
}

.btn-small:hover:not(:disabled) {
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);
}

.btn-text {
  margin-left: 0.25rem;
}

/* Mobile Card View */
@media (max-width: 768px) {
  .desktop-table-view {
    display: none;
  }

  .mobile-card-view {
    display: block;
    padding: 1rem;
  }

  .slot-cards-container {
    display: flex;
    flex-direction: column;
    gap: 1rem;
  }

  .slot-card {
    background-color: var(--bg-primary);
    border: 1px solid var(--border-color);
    border-radius: var(--border-radius-lg);
    box-shadow: var(--shadow-sm);
    overflow: hidden;
    transition: var(--transition);
  }

  .slot-card:hover {
    box-shadow: var(--shadow-md);
    transform: translateY(-2px);
  }

  .slot-card-header {
    background: linear-gradient(
      135deg,
      var(--bg-secondary) 0%,
```

```
    var(--bg-tertiary) 100%
  );
  padding: 0.75rem 1rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  border-bottom: 1px solid var(--border-color);
}

.slot-number-badge {
  background: linear-gradient(
    135deg,
    var(--primary-color) 0%,
    var(--primary-hover) 100%
  );
  color: var(--text-inverse);
  padding: 0.3rem 0.6rem;
  border-radius: var(--border-radius);
  font-size: 0.8rem;
  font-weight: 600;
}

.slot-duration {
  font-size: 0.8rem;
  color: var(--text-muted);
  font-weight: 500;
  background-color: var(--bg-primary);
  padding: 0.3rem 0.6rem;
  border-radius: var(--border-radius);
  border: 1px solid var(--border-light);
}

.slot-card-content {
  padding: 1rem;
}

.date-label {
  font-size: 0.8rem;
  font-weight: 600;
  color: var(--text-secondary);
  margin-bottom: 0.5rem;
  text-transform: uppercase;
  letter-spacing: 0.5px;
}

.mobile-date-range {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
}

.mobile-date-item {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0.5rem;
  background-color: var(--bg-secondary);
  border-radius: var(--border-radius);
  border: 1px solid var(--border-light);
}

.mobile-date-label {
  font-size: 0.8rem;
  font-weight: 500;
  color: var(--text-secondary);
}

.mobile-date-value {
  font-size: 0.85rem;
  font-weight: 600;
  color: var(--text-primary);
  font-family: "Courier New", monospace;
}
```

```
  .slot-card-actions {
    padding: 0.75rem 1rem;
    background-color: var(--bg-secondary);
    border-top: 1px solid var(--border-color);
    display: flex;
    gap: 0.5rem;
  }

  .btn-mobile {
    flex: 1;
    padding: 0.6rem 0.5rem;
    font-size: 0.75rem;
    min-height: 38px;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 0.3rem;
    border-radius: var(--border-radius);
    font-weight: 500;
    transition: all 0.2s ease;
  }

  .btn-mobile:hover:not(:disabled) {
    transform: translateY(-1px);
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  .btn-mobile .btn-icon {
    font-size: 0.9em;
  }
}

/* Tablet View Adjustments */
@media (max-width: 1024px) and (min-width: 769px) {
  .action-buttons {
    flex-direction: column;
    gap: 0.3rem;
  }

  .btn-small {
    width: 100%;
    justify-content: center;
  }

  .summary-table th:last-child {
    width: 120px;
  }
}

/* Small Mobile Adjustments */
@media (max-width: 480px) {
  .mobile-card-view {
    padding: 0.5rem;
  }

  .slot-card-actions {
    flex-direction: column;
    gap: 0.4rem;
  }

  .btn-mobile {
    width: 100%;
    min-height: 42px;
  }

  .mobile-date-item {
    flex-direction: column;
    align-items: flex-start;
    gap: 0.25rem;
  }

  .mobile-date-value {
```

```
    align-self: flex-end;
  }
}

/* Enhanced Hover Effects */
.date-badge:hover {
  transform: scale(1.02);
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.slot-card:active {
  transform: translateY(0);
}

/* Loading State for Summary Table */
.summary-table.loading {
  opacity: 0.6;
  pointer-events: none;
}

.summary-table.loading::after {
  content: "";
  position: absolute;
  top: 50%;
  left: 50%;
  width: 20px;
  height: 20px;
  margin: -10px 0 0 -10px;
  border: 2px solid var(--border-color);
  border-top-color: var(--primary-color);
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

/* Empty State */
.empty-state {
  text-align: center;
  padding: 2rem;
  color: var(--text-muted);
}

.empty-state-icon {
  font-size: 3rem;
  margin-bottom: 1rem;
}

.empty-state-text {
  font-size: 1.1rem;
  font-weight: 500;
}

/* Toast Notifications - Theme Compatible */
.Toastify__toast-container {
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", sans-serif;
}

/* Light Theme Toast Styles */
.Toastify__toast {
  background-color: var(--bg-primary) !important;
  color: var(--text-primary) !important;
  border: 1px solid var(--border-color) !important;
  border-radius: var(--border-radius-lg) !important;
  box-shadow: var(--shadow-lg) !important;
  font-size: 0.9rem !important;
  min-height: 64px !important;
}

.Toastify__toast--success {
  background-color: var(--bg-primary) !important;
  border-left: 4px solid var(--success-color) !important;
}
```

```
.Toastify__toast--error {
  background-color: var(--bg-primary) !important;
  border-left: 4px solid var(--danger-color) !important;
}

.Toastify__toast--info {
  background-color: var(--bg-primary) !important;
  border-left: 4px solid var(--info-color) !important;
}

.Toastify__toast--warning {
  background-color: var(--bg-primary) !important;
  border-left: 4px solid var(--warning-color) !important;
}

/* Toast Body */
.Toastify__toast-body {
  color: var(--text-primary) !important;
  padding: 0.75rem !important;
  font-weight: 500 !important;
  line-height: 1.4 !important;
}

/* Toast Icons */
.Toastify__toast-icon {
  width: 20px !important;
  height: 20px !important;
  margin-right: 0.75rem !important;
}

.Toastify__toast-icon svg {
  fill: var(--text-primary) !important;
}

.Toastify__toast--success .Toastify__toast-icon svg {
  fill: var(--success-color) !important;
}

.Toastify__toast--error .Toastify__toast-icon svg {
  fill: var(--danger-color) !important;
}

.Toastify__toast--info .Toastify__toast-icon svg {
  fill: var(--info-color) !important;
}

.Toastify__toast--warning .Toastify__toast-icon svg {
  fill: var(--warning-color) !important;
}

/* Close Button */
.Toastify__close-button {
  color: var(--text-secondary) !important;
  opacity: 0.7 !important;
  align-self: flex-start !important;
  margin-top: 0.25rem !important;
}

.Toastify__close-button:hover {
  color: var(--text-primary) !important;
  opacity: 1 !important;
}

/* Progress Bar */
.Toastify__progress-bar {
  background: linear-gradient(
    90deg,
    var(--primary-color),
    var(--primary-hover)
  ) !important;
  height: 3px !important;
}
```

```
.Toastify__progress-bar--success {
  background: linear-gradient(90deg, var(--success-color), #1e7e34) !important;
}

.Toastify__progress-bar--error {
  background: linear-gradient(90deg, var(--danger-color), #c82333) !important;
}

.Toastify__progress-bar--info {
  background: linear-gradient(90deg, var(--info-color), #138496) !important;
}

.Toastify__progress-bar--warning {
  background: linear-gradient(90deg, var(--warning-color), #e0a800) !important;
}

/* Dark Theme Specific Overrides */
.dark .Toastify__toast,
[data-theme="dark"] .Toastify__toast {
  background-color: var(--bg-card) !important;
  color: var(--text-primary) !important;
  border-color: var(--border-color) !important;
}

.dark .Toastify__toast-body,
[data-theme="dark"] .Toastify__toast-body {
  color: var(--text-primary) !important;
}

.dark .Toastify__close-button,
[data-theme="dark"] .Toastify__close-button {
  color: var(--text-secondary) !important;
}

.dark .Toastify__close-button:hover,
[data-theme="dark"] .Toastify__close-button:hover {
  color: var(--text-primary) !important;
}

/* Custom Delete Confirmation Toast Styling */
.delete-confirmation {
  text-align: center;
  padding: 1rem;
  color: var(--text-primary) !important;
}

.delete-confirmation p {
  margin-bottom: 1rem;
  color: var(--text-primary) !important;
  font-weight: 500;
  font-size: 0.9rem;
  line-height: 1.4;
}

.delete-confirmation strong {
  color: var(--danger-color);
  font-weight: 600;
}

.delete-confirmation-buttons {
  display: flex;
  justify-content: center;
  gap: 0.75rem;
  flex-wrap: wrap;
}

.delete-confirmation-buttons .btn {
  min-width: 80px;
  font-size: 0.8rem;
  padding: 0.5rem 1rem;
}

/* Toast Animation Improvements */
```

```
.Toastify__toast {
  animation-duration: 0.3s !important;
}

.Toastify__bounce-enter--top-center {
  animation-name: toastSlideInDown !important;
}

.Toastify__bounce-exit--top-center {
  animation-name: toastSlideOutUp !important;
}

@keyframes toastSlideInDown {
  from {
    opacity: 0;
    transform: translate3d(-50%, -100%, 0) scale(0.95);
  }
  to {
    opacity: 1;
    transform: translate3d(-50%, 0, 0) scale(1);
  }
}

@keyframes toastSlideOutUp {
  from {
    opacity: 1;
    transform: translate3d(-50%, 0, 0) scale(1);
  }
  to {
    opacity: 0;
    transform: translate3d(-50%, -100%, 0) scale(0.95);
  }
}

/* Mobile Toast Adjustments */
@media (max-width: 768px) {
  .Toastify__toast-container {
    width: calc(100vw - 2rem) !important;
    left: 1rem !important;
    right: 1rem !important;
    margin: 0 !important;
  }

  .Toastify__toast {
    margin-bottom: 0.5rem !important;
    border-radius: var(--border-radius) !important;
    min-height: 56px !important;
  }

  .Toastify__toast-body {
    padding: 0.6rem !important;
    font-size: 0.85rem !important;
  }

  .delete-confirmation {
    padding: 0.75rem;
  }

  .delete-confirmation p {
    font-size: 0.85rem;
    margin-bottom: 0.75rem;
  }

  .delete-confirmation-buttons {
    flex-direction: column;
    gap: 0.5rem;
  }

  .delete-confirmation-buttons .btn {
    width: 100%;
    min-width: unset;
  }
}
```

```
/* High Contrast Mode for Toasts */
@media (prefers-contrast: high) {
  .Toastify__toast {
    border-width: 2px !important;
  }

  .Toastify__toast--success {
    border-left-width: 6px !important;
  }

  .Toastify__toast--error {
    border-left-width: 6px !important;
  }

  .Toastify__toast--info {
    border-left-width: 6px !important;
  }

  .Toastify__toast--warning {
    border-left-width: 6px !important;
  }
}

/* Reduced Motion for Toasts */
@media (prefers-reduced-motion: reduce) {
  .Toastify__toast {
    animation-duration: 0.01ms !important;
  }

  @keyframes toastSlideInDown {
    from,
    to {
      opacity: 1;
      transform: translate3d(-50%, 0, 0) scale(1);
    }
  }

  @keyframes toastSlideOutUp {
    from,
    to {
      opacity: 0;
      transform: translate3d(-50%, 0, 0) scale(1);
    }
  }
}

/* Toast Container Positioning */
.Toastify__toast-container--top-center {
  top: 2rem !important;
  left: 50% !important;
  transform: translateX(-50%) !important;
  width: auto !important;
  max-width: 500px !important;
}

/* Success Toast Enhancements */
.Toastify__toast--success .Toastify__toast-body::before {
  content: "? ";
  margin-right: 0.5rem;
}

/* Error Toast Enhancements */
.Toastify__toast--error .Toastify__toast-body::before {
  content: "? ";
  margin-right: 0.5rem;
}

/* Info Toast Enhancements */
.Toastify__toast--info .Toastify__toast-body::before {
  content: "?? ";
  margin-right: 0.5rem;
}
```

```
/* Warning Toast Enhancements */
.Toastify__toast--warning .Toastify__toast-body::before {
  content: "?? ";
  margin-right: 0.5rem;
}
```

File: src\dashboard\doctor\DoctorSlotBooking.js

```
"use client"

import { useState, useEffect } from "react"
import { toast, ToastContainer } from "react-toastify"
import { useTheme } from "../../ThemeProvider"
import "react-toastify/dist/ReactToastify.css"
import "./DoctorSlotBooking.css"

const DoctorSlotBooking = () => {
  const { theme } = useTheme()

  const daysOfWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

  const initialSlots = daysOfWeek.map((day) => ({
    day,
    available: false,
    timeBlocks: [],
  }))

  const [slots, setSlots] = useState(initialSlots)
  const [startDate, setStartDate] = useState("")
  const [endDate, setEndDate] = useState("")
  const [allSlotData, setAllSlotData] = useState([])
  const [showForm, setShowForm] = useState(false)
  const [editingIndex, setEditingIndex] = useState(null)
  const [editingId, setEditingId] = useState(null)
  const [viewSlotIndex, setViewSlotIndex] = useState(null)

  const toggleAvailability = (day) => {
    setSlots((prev) =>
      prev.map((slot) =>
        slot.day === day
          ? {
              ...slot,
              available: !slot.available,
              timeBlocks: slot.available ? [] : [{ start: "", end: "" }],
            }
          : slot,
      ),
    )
  }

  const updateTimeBlock = (day, idx, field, value) => {
    setSlots((prev) =>
      prev.map((slot) => {
        if (slot.day === day) {
          const newBlocks = [...slot.timeBlocks]
          newBlocks[idx][field] = value
          return { ...slot, timeBlocks: newBlocks }
        }
        return slot
      }),
    )
  }

  const addTimeBlock = (day) => {
    setSlots((prev) =>
      prev.map((slot) =>
        slot.day === day
          ? {
              ...slot,
              timeBlocks: [...slot.timeBlocks, { start: "", end: "" }],
            }
          : slot,
      ),
```

```
  )
}

const removeTimeBlock = (day, indexToRemove) => {
  setSlots((prev) =>
    prev.map((slot) =>
      slot.day === day
        ? {
            ...slot,
            timeBlocks: slot.timeBlocks.filter((_, i) => i !== indexToRemove),
          }
        : slot,
    ),
  )
}

const resetForm = () => {
  setSlots(initialSlots)
  setStartDate("")
  setEndDate("")
  setEditingIndex(null)
  setEditingId(null)
  setShowForm(false)
}

const validateSlots = () => {
  for (const slot of slots) {
    if (slot.available) {
      for (const block of slot.timeBlocks) {
        if (!block.start || !block.end) {
          toast.error(`Incomplete time block in ${slot.day}`)
          return false
        }
        if (block.start >= block.end) {
          toast.error(`Start time must be before end time in ${slot.day}`)
          return false
        }
      }
    }
  }
  return true
}

const handleSubmit = (e) => {
  e.preventDefault()
  if (!startDate || !endDate) {
    toast.error("Please enter both start and end dates.")
    return
  }
  if (!validateSlots()) return

  const entry = {
    startDate,
    endDate,
    slots,
  }

  const request = editingId
    ? fetch(`http://localhost:3001/doctorSlots/${editingId}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ ...entry, id: editingId }),
      })
    : fetch("http://localhost:3001/doctorSlots", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(entry),
      })

  request
    .then((res) => {
      if (!res.ok) throw new Error("Failed to save")
      return res.json()
```

```
    })
    .then(() => {
      fetch("http://localhost:3001/doctorSlots")
        .then((res) => res.json())
        .then((data) => {
          setAllSlotData(data)
          toast.success(editingId ? "Slot updated!" : "Slot booked!")
          resetForm()
        })
    })
    .catch(() => toast.error("Server error"))
}

const handleEdit = (index) => {
  const entry = allSlotData[index]
  setSlots(entry.slots)
  setStartDate(entry.startDate)
  setEndDate(entry.endDate)
  setEditingIndex(index)
  setEditingId(entry.id)
  setShowForm(true)
  window.scrollTo({ top: 0, behavior: "smooth" })
}

const handleDelete = (id) => {
  fetch(`http://localhost:3001/doctorSlots/${id}`, {
    method: "DELETE",
  })
    .then(() => {
      setAllSlotData((prev) => prev.filter((item) => item.id !== id))
      toast.info("Slot deleted")
    })
    .catch(() => toast.error("Failed to delete"))
}

const confirmDelete = (id) => {
  toast(
    ({ closeToast }) => (
      <div className="delete-confirmation">
        <p>Are you sure you want to delete this slot?</p>
        <div className="delete-confirmation-buttons">
          <button
            className="btn btn-danger"
            onClick={() => {
              handleDelete(id)
              closeToast()
            }}
          >
            Yes
          </button>
          <button className="btn btn-secondary" onClick={closeToast}>
            Cancel
          </button>
        </div>
      </div>
    ),
    {
      position: "top-center",
      autoClose: false,
      closeOnClick: false,
      draggable: false,
      closeButton: false,
    },
  )
}

useEffect(() => {
  fetch("http://localhost:3001/doctorSlots")
    .then((res) => res.json())
    .then((data) => setAllSlotData(data))
    .catch(() => toast.error("Failed to fetch slots"))
}, [])
```

```
return (
  <div className={`doctor-slot-container ${theme}`}>
    <ToastContainer position="top-center" autoClose={2000} />

    <div className="main-content">
      <div className="page-header">
        <h1 className="page-title">
          <span className="title-icon">?</span>
          Doctor Slot Booking
        </h1>

        {!showForm && (
          <button className="btn btn-primary btn-large" onClick={() => setShowForm(true)}>
            <span className="btn-icon">?</span>
            {editingIndex !== null ? "Edit Slot" : "Book New Slot"}
          </button>
        )}
      </div>

      {showForm && (
        <div className="card form-card">
          <div className="card-header">
            <h2 className="card-title">
              <span className="title-icon">?</span>
              {editingIndex !== null ? "Update Slots" : "Select Weekly Availability"}
            </h2>
          </div>

          <form className="slot-form" onSubmit={handleSubmit}>
            <div className="date-inputs-section">
              <div className="input-group">
                <label htmlFor="start-date" className="input-label">
                  <span className="label-icon">?</span>
                  Start Date
                </label>
                <input
                  id="start-date"
                  type="date"
                  value={startDate}
                  onChange={(e) => setStartDate(e.target.value)}
                  className="form-input"
                  required
                />
              </div>

              <div className="input-group">
                <label htmlFor="end-date" className="input-label">
                  <span className="label-icon">?</span>
                  End Date
                </label>
                <input
                  id="end-date"
                  type="date"
                  value={endDate}
                  onChange={(e) => setEndDate(e.target.value)}
                  className="form-input"
                  required
                />
              </div>
            </div>

            <div className="availability-section">
              <h3 className="section-title">Weekly Availability</h3>

              <div className="table-container">
                <table className="availability-table">
                  <thead>
                    <tr>
                      <th>Day</th>
                      <th>Available</th>
                      <th>Time Blocks</th>
                    </tr>
                  </thead>
```

```
<tbody>
  {slots.map(({ day, available, timeBlocks }) => (
    <tr key={day} className={available ? "available-row" : ""}>
      <td data-label="Day" className="day-cell">
        <span className="day-name">{day}</span>
      </td>

      <td data-label="Available" className="checkbox-cell">
        <label className="checkbox-wrapper">
          <input
            type="checkbox"
            checked={available}
            onChange={() => toggleAvailability(day)}
            className="availability-checkbox"
          />
          <span className="checkbox-custom"></span>
          <span className="checkbox-label">{available ? "Available" : "Not Available"}</span
        </label>
      </td>

      <td data-label="Time Blocks" className="time-blocks-cell">
        {available && (
          <div className="time-blocks-container">
            {timeBlocks.map((block, idx) => (
              <div key={idx} className="time-block">
                <div className="time-inputs">
                  <input
                    type="time"
                    value={block.start}
                    onChange={(e) => updateTimeBlock(day, idx, "start", e.target.value)}
                    className="time-input"
                    placeholder="Start time"
                  />
                  <span className="time-separator">to</span>
                  <input
                    type="time"
                    value={block.end}
                    onChange={(e) => updateTimeBlock(day, idx, "end", e.target.value)}
                    className="time-input"
                    placeholder="End time"
                  />
                </div>
                <button
                  type="button"
                  className="btn btn-danger btn-small remove-btn"
                  onClick={() => removeTimeBlock(day, idx)}
                  title="Remove time block"
                >
                  <span className="btn-icon">??</span>
                  Remove
                </button>
              </div>
            ))}

            <button
              type="button"
              className="btn btn-outline add-time-btn"
              onClick={() => addTimeBlock(day)}
            >
              <span className="btn-icon">?</span>
              Add Time Slot
            </button>
          </div>
        )}

        {!available && (
          <div className="not-available-message">
            <span className="icon">?</span>
            Not available on this day
          </div>
        )}
      </td>
    </tr>
```

```
              ))}
            </tbody>
          </table>
        </div>
      </div>

      <div className="form-actions">
        <button type="submit" className="btn btn-primary btn-large">
          <span className="btn-icon">?</span>
          {editingIndex !== null ? "Update Slot" : "Submit Availability"}
        </button>
        <button type="button" className="btn btn-secondary btn-large" onClick={resetForm}>
          <span className="btn-icon">?</span>
          Cancel
        </button>
      </div>
    </form>
  </div>
)}

{/* Summary View */}
{allSlotData.length > 0 && (
  <div className="card summary-card">
    <div className="card-header">
      <h2 className="card-title">
        <span className="title-icon">?</span>
        Saved Slot Periods
        <span className="slot-count">({allSlotData.length})</span>
      </h2>
    </div>

    {/* Desktop Table View */}
    <div className="desktop-table-view">
      <div className="table-container">
        <table className="summary-table">
          <thead>
            <tr>
              <th>SI_NO</th>
              <th>Date Range</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {allSlotData.map((entry, index) => (
              <tr key={index} className="summary-row">
                <td className="serial-cell">
                  <span className="serial-number">{index + 1}</span>
                </td>
                <td className="date-range-cell">
                  <div className="date-range-container">
                    <div className="date-range">
                      <span className="date-badge date-from">{entry.startDate}</span>
                      <span className="date-separator">?</span>
                      <span className="date-badge date-to">{entry.endDate}</span>
                    </div>
                    <div className="date-duration">
                      {(() => {
                        const start = new Date(entry.startDate)
                        const end = new Date(entry.endDate)
                        const diffTime = Math.abs(end - start)
                        const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24)) + 1
                        return `${diffDays} day${diffDays > 1 ? "s" : ""}`
                      })()}
                    </div>
                  </div>
                </td>
                <td className="actions-cell">
                  <div className="action-buttons">
                    <button
                      className="btn btn-info btn-small"
                      onClick={() => setViewSlotIndex(index)}
                      title="View details"
                    >
```

```
                  <span className="btn-icon">??</span>
                  <span className="btn-text">View</span>
                </button>
                <button
                  className="btn btn-warning btn-small"
                  onClick={() => handleEdit(index)}
                  title="Edit slot"
                >
                  <span className="btn-icon">??</span>
                  <span className="btn-text">Edit</span>
                </button>
                <button
                  className="btn btn-danger btn-small"
                  onClick={() => confirmDelete(entry.id)}
                  title="Delete slot"
                >
                  <span className="btn-icon">??</span>
                  <span className="btn-text">Delete</span>
                </button>
              </div>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

{/* Mobile Card View */}
<div className="mobile-card-view">
  <div className="slot-cards-container">
    {allSlotData.map((entry, index) => (
      <div key={index} className="slot-card">
        <div className="slot-card-header">
          <div className="slot-number">
            <span className="slot-number-badge">#{index + 1}</span>
          </div>
          <div className="slot-duration">
            {(() => {
              const start = new Date(entry.startDate)
              const end = new Date(entry.endDate)
              const diffTime = Math.abs(end - start)
              const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24)) + 1
              return `${diffDays} day${diffDays > 1 ? "s" : ""}`
            })()}
          </div>
        </div>

        <div className="slot-card-content">
          <div className="date-info">
            <div className="date-label">Period</div>
            <div className="mobile-date-range">
              <div className="mobile-date-item">
                <span className="mobile-date-label">From:</span>
                <span className="mobile-date-value">{entry.startDate}</span>
              </div>
              <div className="mobile-date-item">
                <span className="mobile-date-label">To:</span>
                <span className="mobile-date-value">{entry.endDate}</span>
              </div>
            </div>
          </div>
        </div>

        <div className="slot-card-actions">
          <button
            className="btn btn-info btn-mobile"
            onClick={() => setViewSlotIndex(index)}
            title="View details"
          >
            <span className="btn-icon">??</span>
            View
          </button>
```

```
                <button
                  className="btn btn-warning btn-mobile"
                  onClick={() => handleEdit(index)}
                  title="Edit slot"
                >
                  <span className="btn-icon">??</span>
                  Edit
                </button>
                <button
                  className="btn btn-danger btn-mobile"
                  onClick={() => confirmDelete(entry.id)}
                  title="Delete slot"
                >
                  <span className="btn-icon">??</span>
                  Delete
                </button>
              </div>
            </div>
          ))}
        </div>
      </div>
    </div>
  )}

  {/* Modal View */}
  {viewSlotIndex !== null && (
    <div className="modal-overlay" onClick={() => setViewSlotIndex(null)}>
      <div className="modal-content" onClick={(e) => e.stopPropagation()}>
        <div className="modal-header">
          <h3 className="modal-title">
            <span className="title-icon">?</span>
            Slot Availability Details
          </h3>
          <button className="modal-close-btn" onClick={() => setViewSlotIndex(null)} title="Close modal">
            ?
          </button>
        </div>

        <div className="modal-body">
          <div className="date-range-info">
            <span className="date-range-label">Period:</span>
            <span className="date-range-value">
              {allSlotData[viewSlotIndex].startDate} to {allSlotData[viewSlotIndex].endDate}
            </span>
          </div>

          <div className="modal-table-container">
            <table className="modal-table">
              <thead>
                <tr>
                  <th>Day</th>
                  <th>Availability</th>
                </tr>
              </thead>
              <tbody>
                {allSlotData[viewSlotIndex].slots.map((slot) => (
                  <tr key={slot.day} className={slot.available ? "available-row" : "unavailable-row"}>
                    <td data-label="Day" className="day-cell">
                      <span className="day-name">{slot.day}</span>
                    </td>
                    <td data-label="Availability" className="availability-cell">
                      {slot.available ? (
                        <div className="time-slots">
                          {slot.timeBlocks.map((block, i) => (
                            <div key={i} className="time-slot-badge">
                              <span className="time-icon">?</span>
                              {block.start} - {block.end}
                            </div>
                          ))}
                        </div>
                      ) : (
                        <div className="not-available-badge">
                          <span className="icon">?</span>
```

```
                            Not Available
                          </div>
                        )}
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>

          <div className="modal-footer">
            <button className="btn btn-primary btn-large" onClick={() => setViewSlotIndex(null)}>
              <span className="btn-icon">?</span>
              Close
            </button>
          </div>
        </div>
      </div>
    )}
  </div>
 )
}

export default DoctorSlotBooking
```

File: src\dashboard\doctor\components\doctor_footer.js

```
import React from 'react';
import '../../physio/components/Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} Doctor Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\doctor\components\doctor_navbar.js

```
import React from "react";
import "../../physio/components/Navbar.css";
import { Moon, Sun } from "lucide-react";
import { useTheme } from "../../../ThemeProvider";

const DoctorNavbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="../doctor/DoctorDashboardHome"> <div className="navbar-logo">Doctor</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default DoctorNavbar;
```

File: src\dashboard\doctor\components\doctor_sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "../../physio/components/Sidebar.css";
```

```
import {
  Home,
  Users,
  Dumbbell,
  BarChart2,
  LogOut,
  Hamburger,
  Menu,
  Workflow,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
  Hospital,
  LayoutDashboard,
  Calendar,
  KeyRoundIcon,
} from "lucide-react";

const DoctorSidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState("");
  const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
  const navigate = useNavigate();

  const toggleSidebar = () => {
    setIsOpen(!isOpen);
  };

  const toggleCollapse = () => {
    setCollapsed(!collapsed);
  };

  const handleResize = () => {
    setIsOpen(window.innerWidth > 768);
  };

  const closeSidebar = () => {
    if (isOpen && window.innerWidth <= 768) {
      setIsOpen(false);
    }
  };

  const handleMouseEnter = () => {
    if (collapsed && window.innerWidth > 768) {
      setIsHovering(true);
    }
  };

  const handleMouseLeave = () => {
    if (collapsed && window.innerWidth > 768) {
      setIsHovering(false);
    }
  };

  useEffect(() => {
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  useEffect(() => {
    document.body.classList.toggle(
      "sidebar-collapsed",
      collapsed && !isHovering
    );
  }, [collapsed, isHovering]);

  const navLinkClass = ({ isActive }) => (isActive ? "active" : "");

  const handleLogoutClick = (e) => {
```

```
  e.preventDefault();
  setShowLogoutConfirm(true);
};

const confirmLogout = () => {
  localStorage.clear();
  setShowLogoutConfirm(false);
  setToastMessage("Logged out successfully.");
  setShowToast(true);
  setTimeout(() => {
    setShowToast(false);
    navigate("/login");
  }, 3000);
};

const cancelLogout = () => {
  setShowLogoutConfirm(false);
};

return (
  <>
    <button
      className="sidebar-toggle"
      onClick={toggleSidebar}
      aria-label="Toggle sidebar"
    >
      <Menu size={24} />
    </button>

    {isOpen && window.innerWidth <= 768 && (
      <div className="sidebar-overlay show" onClick={closeSidebar}></div>
    )}

    {showLogoutConfirm && (
      <div className="modal-overlay">
        <div className="modal-container">
          <div className="modal-header">
            <center>
              <h3>Logout Confirmation</h3>
            </center>
            <button className="modal-close" onClick={cancelLogout}>
              <X size={18} />
            </button>
          </div>
          <div className="modal-body">
            <p>Are you sure you want to log out?</p>
          </div>
          <div className="modal-footer">
            <button className="btn btn-primary" onClick={confirmLogout}>
              Yes, Logout
            </button>
          </div>
        </div>
      </div>
    )}

    {showToast && (
      <div className="toast-overlay">
        <div className="toast-container success">
          <div className="toast-content">
            <Check size={18} style={{ marginRight: "8px" }} />
            {toastMessage}
          </div>
        </div>
      </div>
    )}

    <aside
      className={`sidebar ${isOpen ? "open" : ""} ${
        collapsed ? "collapsed" : ""
      } ${isHovering ? "hovering" : ""}`}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
```

```
>
  <div className="sidebar-header">
    <div className="sidebar-profile">
      <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
      <a href="/doctor/DoctorDashboardHome">
        {" "}
        <span className="sidebar-username">Doctor</span>
      </a>
    </div>
  </div>

  <nav className="sidebar-menu-wrapper">
    <ul className="sidebar-menu">
      {/* <li>
        <NavLink to="/doctor/doctor_profile" className={navLinkClass}>
          <Home
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Dashboard</span>
        </NavLink>
      </li> */}
      <li>
        <NavLink
          to="/doctor/DoctorDashboardHome"
          className={navLinkClass}
        >
          <Home
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Dashboard Home</span>
        </NavLink>
      </li>

      <li>
        <NavLink
          to="/doctor/DoctorsAppointments"
          className={navLinkClass}
        >
          <Calendar
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span> Appointments</span>
        </NavLink>
      </li>

      <li>
        <NavLink
          to="/doctor/DoctorAssignmentDashboard"
          className={navLinkClass}
        >
          <LayoutDashboard
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Assign Dashboard</span>
        </NavLink>
      </li>
      <li>
        <NavLink
          to="/doctor/DoctorSlotBooking"
          className={navLinkClass}
        >
```

```
              <LayoutDashboard
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Slot Booking</span>
            </NavLink>
          </li>

          <li>
            <NavLink
              to="/doctor/DoctorPasswordRequest"
              className={navLinkClass}
            >
              <KeyRoundIcon
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Reset Password</span>
            </NavLink>
          </li>

          <li>
            <a
              href="#"
              onClick={handleLogoutClick}
              className={navLinkClass({ isActive: false })}
              style={{
                cursor: "pointer",
                display: "flex",
                alignItems: "center",
              }}
            >
              <LogOut
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Log Out</span>
            </a>
          </li>
        </ul>

        <button className="collapse-toggle" onClick={toggleCollapse}>
          <div className="claude-toggle">
            {collapsed ? (
              <ChevronRight size={16} />
            ) : (
              <ChevronLeft size={16} />
            )}
          </div>
        </button>
      </nav>
    </aside>
  </>
 );
};

export default DoctorSidebar;
```

File: src\dashboard\doctor\components\DocumentPopup.js

```
import React, { useState } from "react";
import PdfCardViewer from "./PdfCardViewer";
import "./popup.css";

const DocumentPopup = ({ onClose, documents }) => {
  const [activeTab, setActiveTab] = useState("medical");
  const [pdfUrl, setPdfUrl] = useState(null);
```

```
const medicalData = [
  {
    date: "2025-06-01",
    time: "10:30 AM",
    clinicalNote: "/pdfs/MRN01.pdf",
    prescription: "/prescriptions/MRN123456_labreport.pdf"
  }
];

const testRecords = documents.map((doc) => ({
  name: doc,
  url: `/pdfs/${doc}`
}));

return (
  <div className="popup-overlay">
    <div className="popup-card">
      <button className="close-btn" onClick={onClose}>X</button>
      <div className="tab-buttons">
        <button onClick={() => setActiveTab("medical")} className={activeTab === "medical" ? "active" : ""}>Me
        <button onClick={() => setActiveTab("test")} className={activeTab === "test" ? "active" : ""}>Test Rec
      </div>

      {activeTab === "medical" && (
        <table className="popup-table">
          <thead>
            <tr>
              <th>Date</th>
              <th>Time</th>
              <th>Clinical Note</th>
              <th>Prescription</th>
            </tr>
          </thead>
          <tbody>
            {medicalData.map((entry, i) => (
              <tr key={i}>
                <td>{entry.date}</td>
                <td>{entry.time}</td>
                <td>
                  <button onClick={() => setPdfUrl(entry.clinicalNote)}>View</button>
                </td>
                <td>
                  <button onClick={() => setPdfUrl(entry.prescription)}>View</button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      )}

      {activeTab === "test" && (
        <table className="popup-table">
          <thead>
            <tr>
              <th>Test Name</th>
              <th>View Result</th>
            </tr>
          </thead>
          <tbody>
            {testRecords.map((test, i) => (
              <tr key={i}>
                <td>{test.name}</td>
                <td>
                  <button onClick={() => setPdfUrl(test.url)}>View</button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      )}

      {pdfUrl && (
        <PdfCardViewer url={pdfUrl} onClose={() => setPdfUrl(null)} />
```

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

export default DocumentPopup;
```

File: src\dashboard\doctor\components\PdfCardViewer.js

```
import React from "react";

const PdfCardViewer = ({ url, onClose }) => {
  return (
    <div className="pdf-card">
      <button className="close-btn" onClick={onClose}>X</button>
      <iframe
        src={url}
        title="PDF Viewer"
        width="100%"
        height="500px"
        style={{ border: "none" }}
      />
    </div>
  );
};

export default PdfCardViewer;
```

File: src\dashboard\doctor\components\popup.css

```
/* .popup-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.6);
  z-index: 999;
  display: flex;
  justify-content: center;
  align-items: center;
}

.popup-card {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  width: 80%;
  max-width: 900px;
  position: relative;
}

.close-btn {
  position: absolute;
  right: 1rem;
  top: 1rem;
  border: none;
  background: #eee;
  padding: 0.5rem 1rem;
  cursor: pointer;
  font-weight: bold;
}

.tab-buttons {
  display: flex;
  justify-content: space-around;
  margin-bottom: 1rem;
}

.tab-buttons button {
  padding: 0.5rem 1rem;
  border: none;
  background: #ddd;
  cursor: pointer;
```

```
}

.tab-buttons button.active {
  background: #4caf50;
  color: white;
}

.popup-table {
  width: 100%;
  border-collapse: collapse;
}

.popup-table th, .popup-table td {
  border: 1px solid #ccc;
  padding: 0.5rem;
  text-align: center;
}

.pdf-card {
  background: #fff;
  padding: 1rem;
  border-radius: 8px;
  margin-top: 1rem;
  border: 1px solid #ccc;
} */
```

File: src\dashboard\doctor\pages\DoctorPasswordRequest.js

```
import React, { useState } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../../ThemeProvider"; // Adjust the import based on your project structure
import "react-toastify/dist/ReactToastify.css";
import "../../master_admin/master_admin.css"; // Ensure this path is correct

const DoctorPasswordRequest = () => {
  const { theme } = useTheme(); // Access the current theme
  const [formData, setFormData] = useState({
    counselorName: "",
    email: "",
    reason: "",
  });

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!formData.counselorName || !formData.email || !formData.reason) {
      toast.error("All fields are required!");
      return;
    }

    try {
      await axios.post("http://localhost:5000/passwordChangeRequests", {
        ...formData,
        status: "Pending",
        requestedAt: new Date().toISOString(),
      });

      toast.success("Request submitted to admin!");
      setFormData({ counselorName: "", email: "", reason: "" });
    } catch (err) {
      toast.error("Failed to submit request.");
    }
  };

  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Password Change Request</h1>
      <div className="card" style={{border:"1px solid #cc5500"}}>
```

```
        <form onSubmit={handleSubmit} className="form">
          <div className="form-group">
            <label htmlFor="counselorName">Counselor Name:</label>
            <input
              type="text"
              id="counselorName"
              name="counselorName"
              placeholder="Enter your name"
              value={formData.counselorName}
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="email">Email ID:</label>
            <input
              type="email"
              id="email"
              placeholder="Enter your email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="reason">Reason for Change:</label>
            <input
              type="text"
              id="reason"
              placeholder="Enter reason for password change"
              name="reason"
              value={formData.reason}
              onChange={handleChange}
              className="input-field"

              required
            />
          </div>
          <div className="center-btn">
           <center> <button type="submit" className="btn btn-primary">
              Submit Request
            </button></center>
          </div>
        </form>
      </div>
    </div>
  );
};

export default DoctorPasswordRequest;
```

File: src\dashboard\doctor__tests__\DoctorDashboardHome.test.js

```
import {
  render,
  screen,
  fireEvent,
  waitFor,
  act,
} from "@testing-library/react";
import DoctorDashboardHome from "../DoctorDashboardHome";
import { useTheme } from "../../../ThemeProvider";
import { toast } from "react-toastify";

jest.mock("../../../ThemeProvider", () => ({
  useTheme: jest.fn(),
}));

jest.mock("react-toastify", () => ({
  toast: {
    error: jest.fn(),
```

```
    warning: jest.fn(),
    info: jest.fn(),
  },
  ToastContainer: jest.fn(),
}));

beforeEach(() => {
  global.fetch = jest.fn((url) => {
    if (url === "http://localhost:3001/appointments") {
      return Promise.resolve({
        ok: true,
        json: () =>
          Promise.resolve([
            {
              mrn: "MRN001",
              patientName: "John Doe",
              time: "10:00 AM",
              status: "Pending",
            },
          ]),
      });
    }
    return Promise.reject(new Error("API Not Found"));
  });
});

afterEach(() => {
  jest.clearAllMocks();
});

describe("DoctorDashboardHome", () => {
  beforeEach(() => {
    useTheme.mockReturnValue({ theme: "light" });
  });

  test("renders the DoctorDashboardHome component", async () => {
    await act(async () => {
      render(<DoctorDashboardHome />);

      // Check if quote and appointment data are rendered
      const quoteElement = await screen.findByText(/Every patient is a story/i);
      expect(quoteElement).toBeInTheDocument();

      const appointmentTable = await screen.findByRole("table");
      expect(appointmentTable).toBeInTheDocument();
    });
  });

  test("fetches and displays appointments", async () => {
    await act(async () => {
      render(<DoctorDashboardHome />);

      // Wait for the appointments to appear
      await waitFor(() => screen.getByText(/MRN001/i));
      expect(screen.getByText(/John Doe/i)).toBeInTheDocument();
    });
  });

  test("shows an error if fetch fails", async () => {
    global.fetch = jest.fn(() => Promise.reject(new Error("Failed to fetch")));

    await act(async () => {
      render(<DoctorDashboardHome />);

      await waitFor(() => screen.findByText(/Could not fetch appointments/i));
      expect(toast.error).toHaveBeenCalledWith("Could not fetch appointments");
    });
  });

  test("shows a warning if MRN is empty when searching", async () => {
    await act(async () => {
      render(<DoctorDashboardHome />);
```

```
      const searchButton = screen.getByText(/Search/i);

      // Wait for the button to be rendered before firing an event
      await waitFor(() => expect(searchButton).toBeInTheDocument());

      fireEvent.click(searchButton);

      expect(toast.warning).toHaveBeenCalledWith("Please enter a valid MRN");
    });
  });
});
```

File: src\dashboard\master_admin\ActivityLogs.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../ThemeProvider";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const ActivityLogs = () => {
  const { theme } = useTheme();
  const [logs, setLogs] = useState([]);
  const [search, setSearch] = useState("");
  const [roleFilter, setRoleFilter] = useState("");
  const [startDate, setStartDate] = useState("");
  const [endDate, setEndDate] = useState("");
  const [page, setPage] = useState(1);
  const [loading, setLoading] = useState(true);
  const itemsPerPage = 10;

  useEffect(() => {
    const fetchLogs = async () => {
      try {
        const res = await axios.get("http://localhost:8080/api/activityLogs");
        const sorted = res.data.sort(
          (a, b) => new Date(b.timestamp) - new Date(a.timestamp)
        );
        setLogs(sorted);
      } catch (err) {
        console.error("Error fetching logs:", err);
        toast.error("Failed to fetch activity logs");
      } finally {
        setLoading(false);
      }
    };
    fetchLogs();
  }, []);

  const isWithinDateRange = (timestamp) => {
    const ts = new Date(timestamp);
    const start = startDate ? new Date(startDate) : null;
    const end = endDate ? new Date(endDate) : null;
    if (start && ts < start) return false;
    if (end && ts > end) return false;
    return true;
  };

  const filtered = logs.filter((log) => {
    const matchSearch =
      log.mrn?.toLowerCase().includes(search.toLowerCase()) ||
      log.actor?.toLowerCase().includes(search.toLowerCase()) ||
      log.action?.toLowerCase().includes(search.toLowerCase());
    const matchRole = roleFilter
      ? log.actor?.toLowerCase().includes(roleFilter.toLowerCase())
      : true;
    return matchSearch && matchRole && isWithinDateRange(log.timestamp);
  });

  const paginated = filtered.slice(
    (page - 1) * itemsPerPage,
    page * itemsPerPage
  );
```

```
const exportCSV = () => {
  const headers = ["Timestamp", "Actor", "MRN", "Action"];
  const rows = filtered.map((log) => [
    new Date(log.timestamp).toLocaleString(),
    log.actor,
    log.mrn,
    log.action,
  ]);
  const csvContent =
    "data:text/csv;charset=utf-8," +
    [headers, ...rows].map((e) => e.join(",")).join("\n");

  const encodedUri = encodeURI(csvContent);
  const link = document.createElement("a");
  link.setAttribute("href", encodedUri);
  link.setAttribute("download", "activity_logs.csv");
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
  toast.success("CSV exported successfully!");
};

const handleReset = () => {
  setSearch("");
  setRoleFilter("");
  setStartDate("");
  setEndDate("");
};

const actionClass = (action) => {
  const act = action?.toLowerCase() || "";
  if (act.includes("create")) return "badge-green";
  if (act.includes("update")) return "badge-yellow";
  if (act.includes("delete")) return "badge-red";
  return "badge-blue";
};

const highlightText = (text, keyword) => {
  if (!keyword || !text) return text;
  const parts = text.split(new RegExp(`(${keyword})`, "gi"));
  return parts.map((part, i) =>
    part.toLowerCase() === keyword.toLowerCase() ? <mark key={i}>{part}</mark> : part
  );
};

return (
  <div className={`dashboard-main ${theme}`}>
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Activity Logs</h1>

    <div className="log-controls">
      <input
        type="text"
        placeholder="Search by MRN, role, or action"
        value={search}
        onChange={(e) => setSearch(e.target.value)}
      />
      <select value={roleFilter} onChange={(e) => setRoleFilter(e.target.value)}>
        <option value="">All Roles</option>
        <option value="doctor">Doctor</option>
        <option value="physio">Physio</option>
        <option value="dietitian">Dietitian</option>
        <option value="phlebotomist">Phlebotomist</option>
        <option value="lab">Lab</option>
      </select>
      <input type="date" value={startDate} onChange={(e) => setStartDate(e.target.value)} />
      <input type="date" value={endDate} onChange={(e) => setEndDate(e.target.value)} />
      <button className="btn btn-primary" onClick={handleReset}>Reset Filters</button>
      <button className="btn btn-primary" onClick={exportCSV}>Export to CSV</button>
    </div>

    {loading ? (
```

```
        <p className="loading-text">Loading activity logs...</p>
      ) : (
        <>
          <div className="table-responsive">
            <table className="appointments-table activity-log-table">
              <thead>
                <tr>
                  <th>Timestamp</th>
                  <th>Actor</th>
                  <th>MRN</th>
                  <th>Action</th>
                </tr>
              </thead>
              <tbody>
                {paginated.map((log) => (
                  <tr key={log.id}>
                    <td>{new Date(log.timestamp).toLocaleString()}</td>
                    <td>
                      <span className={`role-badge badge-${log.actor?.toLowerCase()}`}>
                        {log.actor?.toUpperCase()}
                      </span>
                    </td>
                    <td>{highlightText(log.mrn, search)}</td>
                    <td>
                      <span className={`status-badge ${actionClass(log.action)}`}>
                        {highlightText(log.action, search)}
                      </span>
                    </td>
                  </tr>
                ))}
              </tbody>
            </table>
            {filtered.length === 0 && <p className="error-text">No logs found.</p>}
          </div>

          <p className="pagination-info">
            Showing {(page - 1) * itemsPerPage + 1}?{Math.min(page * itemsPerPage, filtered.length)} of {filtere
          </p>

          {filtered.length > itemsPerPage && (
            <div className="pagination">
              <button
                className="btn btn-secondary"
                disabled={page === 1}
                onClick={() => setPage((p) => p - 1)}
              >
                Prev
              </button>
              <span>Page {page}</span>
              <button
                className="btn btn-secondary"
                disabled={page * itemsPerPage >= filtered.length}
                onClick={() => setPage((p) => p + 1)}
              >
                Next
              </button>
            </div>
          )}
        </>
      )}
    </div>
  );
};

export default ActivityLogs;
```

File: src\dashboard\master_admin\AdminBlogSection.js

```
import React, { useState, useEffect, useRef } from "react";
import { useTheme } from "../../ThemeProvider";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";
```

```
const AdminBlogSection = () => {
  const { theme } = useTheme();
  const [blogs, setBlogs] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [formData, setFormData] = useState({
    title: "",
    smallDesc: "",
    longDesc: "",
    banner: ""
  });
  const [editId, setEditId] = useState(null);
  const [selectedBlog, setSelectedBlog] = useState(null);
  const formRef = useRef(null);
  const API_URL = "http://localhost:8080/api/blogs";

  useEffect(() => {
    fetch(API_URL)
      .then((res) => res.json())
      .then((data) => setBlogs(data));
  }, []);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prev) => ({ ...prev, [name]: value }));
  };

  const handleBannerUpload = (e) => {
    const file = e.target.files[0];
    if (file) {
      const url = URL.createObjectURL(file);
      setFormData((prev) => ({ ...prev, banner: url }));
    }
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    const blogData = { ...formData };

    if (editId) {
      fetch(`${API_URL}/${editId}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(blogData),
      })
        .then((res) => res.json())
        .then((updated) => {
          const updatedList = blogs.map((b) => (b.id === editId ? updated : b));
          setBlogs(updatedList);
          setEditId(null);
          toast.info("Blog updated!");
          resetForm();
          setShowForm(false);
        });
    } else {
      fetch(API_URL, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(blogData),
      })
        .then((res) => res.json())
        .then((newBlog) => {
          setBlogs([...blogs, newBlog]);
          toast.success("Blog published successfully!");
          resetForm();
          setShowForm(false);
        });
    }
  };

  const resetForm = () => {
    setFormData({
      title: "",
      smallDesc: "",
```

```
    longDesc: "",
    banner: ""
  });
};

const handleEdit = (blog) => {
  setFormData({
    title: blog.title,
    smallDesc: blog.smallDesc,
    longDesc: blog.longDesc,
    banner: blog.banner
  });
  setEditId(blog.id);
  setShowForm(true);
  setTimeout(() => {
    formRef.current?.scrollIntoView({ behavior: "smooth" });
  }, 100);
};

const handleDelete = (id) => {
  const ConfirmToast = () => (
    <div>
      <p>Are you sure you want to delete this blog?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          onClick={() => {
            fetch(`${API_URL}/${id}`, { method: "DELETE" }).then(() => {
              setBlogs((prev) => prev.filter((b) => b.id !== id));
              toast.dismiss();
              toast.error("Blog deleted.");
            });
          }}
          className="btn btn-primary"
        >
          Confirm
        </button>
        <button
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
          className="btn btn-primary"
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(<ConfirmToast />, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className={`dashboard-main ${theme}`}>
    <ToastContainer position="top-center" autoClose={3000} />
    <h1>Admin Blog Section</h1>

    <button className="btn btn-primary" onClick={() => setShowForm(!showForm)}>
      {showForm ? "Close Blog Form" : "Add Blog"}
    </button>

    <div
      ref={formRef}
      style={{
        maxHeight: showForm ? "2000px" : "0",
        overflow: "hidden",
        transition: "max-height 0.5s ease-in-out",
```

```
  }}
>
  {showForm && (
    <form onSubmit={handleSubmit} className="card" style={{ marginTop: "20px" }}>
      <label>Blog Title</label>
      <input
        name="title"
        placeholder="Enter blog title"
        value={formData.title}
        onChange={handleChange}
        required
      />
      <label>Short Description</label>
      <input
        name="smallDesc"
        placeholder="Enter short description"
        value={formData.smallDesc}
        onChange={handleChange}
        required
      />
      <label>Long Description</label>
      <textarea
        name="longDesc"
        placeholder="Enter full blog content"
        value={formData.longDesc}
        onChange={handleChange}
        rows={6}
        required
      />
      <label>Banner Image</label>
      <input
        type="file"
        accept="image/*"
        onChange={handleBannerUpload}
        required={!formData.banner}
      />
      {formData.banner && (
        <img
          src={formData.banner}
          alt="banner-preview"
          style={{ maxHeight: "150px", marginTop: "10px" }}
        />
      )}
      <div className="center-btn">
        <button type="submit" className="btn btn-primary">
          {editId ? "Update Blog" : "Publish Blog"}
        </button>
      </div>
    </form>
  )}
</div>

<h2 style={{ marginTop: "30px" }}>Published Blogs</h2>
{blogs.length === 0 ? (
  <p>No blog posts yet.</p>
) : (
  <div className="blog-grid">
    {blogs.map((b) => (
      <div key={b.id} className="card blog-card">
        <img
          src={b.banner}
          alt="banner"
          style={{ height: "100px", objectFit: "cover", width: "100%" }}
        />
        <h3>{b.title}</h3>
        <p>
          <strong>{new Date(b.date).toLocaleString()}</strong>
        </p>
        <p>
          <em>{b.smallDesc}</em>
        </p>
        <div
          className="center-btn"
```

```
              style={{ display: "flex", justifyContent: "space-between", gap: "10px" }}
            >
              <button className="btn btn-primary" onClick={() => setSelectedBlog(b)}>
                View
              </button>
              <button className="btn btn-primary" onClick={() => handleEdit(b)}>
                Edit
              </button>
              <button className="btn btn-primary" onClick={() => handleDelete(b.id)}>
                Delete
              </button>
            </div>
          </div>
        ))}
      </div>
    )}

    {selectedBlog && (
      <div className="modal-overlay" style={{ border: "1px solid #cc5500" }}>
        <div className="modals-box" style={{ border: "1px solid #cc5500" }}>
          <h2>{selectedBlog.title}</h2>

          <img
            src={selectedBlog.banner}
            alt="blog"
            style={{ width: "100%", maxHeight: "200px", objectFit: "cover" }}
          />
          <p>
            <strong>{new Date(selectedBlog.date).toLocaleString()}</strong>
          </p>
          <p>
            <em>{selectedBlog.smallDesc}</em>
          </p>
          <p>{selectedBlog.longDesc}</p>
          <div className="center-btn" style={{ marginTop: "20px" }}>
            <center>
              <button className="btn btn-primary" onClick={() => setSelectedBlog(null)}>
                Close
              </button>
            </center>
          </div>
        </div>
      </div>
    )}
  </div>
);
};

export default AdminBlogSection;
```

File: src\dashboard\master_admin\AdminPasswordRequests.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const AdminPasswordRequests = () => {
  const [requests, setRequests] = useState([]);

  const fetchRequests = async () => {
    try {
      const res = await axios.get("http://localhost:8080/api/passwordChangeRequests");
      setRequests(res.data);
    } catch (err) {
      toast.error("Failed to fetch password requests.");
    }
  };

  useEffect(() => {
    fetchRequests();
  }, []);
```

```
const updateStatus = async (id, newStatus) => {
  try {
    await axios.patch(`http://localhost:8080/api/passwordChangeRequests/${id}`, {
      status: newStatus,
    });
    toast.success(`Request ${newStatus}`);
    fetchRequests();
  } catch (err) {
    toast.error("Failed to update request.");
  }
};

return (
  <div className="table-container">
    <ToastContainer position="top-center" autoClose={2000} />
    <h2>Password Change Requests</h2>
    <div className="table-responsive">
      <table className="user-table">
        <thead>
          <tr>
            <th>Counselor Name</th>
            <th>Email</th>
            <th>Reason</th>
            <th>Requested At</th>
            <th>Status</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          {requests.map((req) => (
            <tr key={req.id}>
              <td>{req.counselorName}</td>
              <td>{req.email}</td>
              <td>{req.reason}</td>
              <td>{new Date(req.requestedAt).toLocaleString()}</td>
              <td>
                <span
                  className={`status-tag ${
                    req.status === "Pending"
                      ? "pending"
                      : req.status === "Approved"
                      ? "approved"
                      : "rejected"
                  }`}
                >
                  {req.status}
                </span>
              </td>
              <td>
                {req.status === "Pending" ? (
                  <>
                    <button
                      className="btn btn-success"
                      onClick={() => updateStatus(req.id, "Approved")}
                    >
                      Approve
                    </button>
                    <button
                      className="btn btn-danger"
                      onClick={() => updateStatus(req.id, "Rejected")}
                    >
                      Reject
                    </button>
                  </>
                ) : (
                  <em>Processed</em>
                )}
              </td>
            </tr>
          ))}
          {requests.length === 0 && (
            <tr>
              <td colSpan="6" style={{ textAlign: "center" }}>
```

```
                No requests found.
              </td>
            </tr>
          )}
        </tbody>
      </table>
    </div>
  </div>
 );
};

export default AdminPasswordRequests;
```

File: src\dashboard\master_admin\AllReports.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useTheme } from "../../ThemeProvider";
import "./master_admin.css";

const AllReports = () => {
  const { theme } = useTheme();
  const [patients, setPatients] = useState([]);
  const [search, setSearch] = useState("");

  useEffect(() => {
    const fetchReports = async () => {
      try {
        const res = await axios.get("http://localhost:8080/api/patients/withReports");
        setPatients(res.data);
      } catch (err) {
        console.error("Error fetching reports:", err);
      }
    };

    fetchReports();
  }, []);

  const filtered = patients.filter(
    (p) =>
      p.name.toLowerCase().includes(search.toLowerCase()) ||
      p.mrn.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <div className={`dashboard-main ${theme}`}>
      <h1>All Reports</h1>

      <div className="report-search">
        <input
          type="text"
          placeholder="Search by name or MRN"
          value={search}
          onChange={(e) => setSearch(e.target.value)}
        />
      </div>

      <div className="table-responsive">
        <table className="user-table">
          <thead>
            <tr>
              <th>MRN</th>
              <th>Name</th>
              <th>Condition</th>
              <th>View Report</th>
            </tr>
          </thead>
          <tbody>
            {filtered.map((p) => (
              <tr key={p.id}>
                <td>{p.mrn}</td>
                <td>{p.name}</td>
                <td>{p.condition}</td>
                <td>
```

```
                  <a
                    href={p.reportPdfUrl}
                    target="_blank"
                    rel="noopener noreferrer"
                    className="btn btn-primary"
                  >
                    View PDF
                  </a>
                </td>
              </tr>
            ))}
            {filtered.length === 0 && (
              <tr>
                <td colSpan="4" className="error-text" style={{ textAlign: "center" }}>
                  No reports found.
                </td>
              </tr>
            )}
          </tbody>
        </table>
      </div>
    </div>
  );
};

export default AllReports;
```

File: src\dashboard\master_admin\AppointmentsContainer.js

```
import React from "react";
import { useNavigate } from "react-router-dom";
import "./master_admin.css";

const appointmentTypes = [
  { label: "Doctor", icon: "?", path: "/masteradmin/appointments/doctor" },
  { label: "Dietitian", icon: "?", path: "/masteradmin/appointments/dietitian" },
  { label: "Physio", icon: "?", path: "/masteradmin/appointments/physio" },
  { label: "Counselor", icon: "?", path: "/masteradmin/appointments/counselor" },
  { label: "Phlebotomist", icon: "?", path: "/masteradmin/appointments/phlebotomist" },
];


const AppointmentsContainer = () => {
  const navigate = useNavigate();

  return (
    <div className="dashboard-main">
      <h1>Appointments</h1>
      <div className="appointments-grid">
        {appointmentTypes.map((type) => (
          <div
            key={type.label}
            className="card appointment-type-card"
            onClick={() => navigate(type.path)}
          >
            <div className="appointment-icon">{type.icon}</div>
            <h3>{type.label} Appointments</h3>
            <p>Manage all {type.label.toLowerCase()} bookings</p>
            <div className="center-btn">
              <button className="btn btn-primary">View</button>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default AppointmentsContainer;
```

File: src\dashboard\master_admin\appointments_container.css

```
.appointments-container {
  padding: 30px;
  text-align: center;
```

```
}

.appointment-card-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(240px, 1fr));
  gap: 30px;
  margin-top: 30px;
}

.appointment-card {
  background-color: #fff;
  border: 1px solid #ddd;
  padding: 25px;
  border-radius: 12px;
  cursor: pointer;
  transition: transform 0.2s, box-shadow 0.2s;
  box-shadow: 0 2px 6px rgba(0,0,0,0.05);
  text-align: center;
}

.appointment-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
}

.card-icon {
  font-size: 40px;
  margin-bottom: 10px;
}

.view-btn {
  margin-top: 15px;
  padding: 8px 16px;
  background-color: #4a90e2;
  color: white;
  border: none;
  border-radius: 6px;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.2s;
}

.view-btn:hover {
  background-color: #357abd;
}
```

File: src\dashboard\master_admin\CounselorAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const CounselorAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/counselor`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch((err) => console.error("Failed to load appointments:", err));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );
    fetch(`${API_BASE}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
```

```
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Update failed"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  let nextStatus;

  switch (current.status) {
    case "Pending":
      nextStatus = "Approved";
      break;
    case "Approved":
      nextStatus = "Completed";
      break;
    default:
      nextStatus = "Pending";
  }

  const updated = { ...current, status: nextStatus };
  updateAppointment(updated);
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(
    100000000 + Math.random() * 900000000
  )}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = () => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-secondary"
          onClick={() => {
            fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Appointment deleted.");
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-secondary"
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
```

```
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Counselor Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online" ? "badge-orange" : "badge-blue"
                    }`}
                  >
                    {a.appointmentType}
                  </span>
                </td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => setSelectedNote(a.notes)}
                  >
                    View
                  </button>
                </td>
                <td>
                  {a.appointmentType === "Online" ? (
                    <div
                      style={{
                        display: "flex",
                        flexDirection: "column",
                        alignItems: "center",
                      }}
                    >
                      <input
                        type="text"
                        value={a.meetingLink || ""}
                        onChange={(e) =>
                          handleMeetingLinkChange(a.id, e.target.value)
                        }
                        placeholder="Enter or generate"
                        style={{ width: "180px", marginBottom: "5px" }}
                        className="search-input"
                      />
                      <div style={{ display: "flex", gap: "5px" }}>
                        <button
                          className="btn btn-primary"
                          onClick={() => handleGenerateLink(a.id)}
```

```
                        >
                          Auto
                        </button>
                        <button
                          className="btn btn-primary"
                          style={{
                            backgroundColor: "#f44336",
                            color: "#fff",
                          }}
                          onClick={() => handleMeetingLinkChange(a.id, "")}
                        >
                          <XCircle size={16} />
                        </button>
                      </div>
                    </div>
                  ) : (
                    "-"
                  )}
                </td>
                <td>
                  <span
                    className={`status-badge ${
                      a.status === "Completed"
                        ? "badge-green"
                        : a.status === "Approved"
                        ? "badge-blue"
                        : "badge-yellow"
                    }`}
                    onClick={() => handleStatusToggle(a.id)}
                    style={{ cursor: "pointer" }}
                  >
                    {a.status}
                  </span>
                </td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => handleDelete(a.id)}
                  >
                    Delete
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}

    {selectedNote && (
      <div className="modal-overlay">
        <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
          <h2>Full Notes</h2>
          <p>{selectedNote}</p>
          <div className="center-btn">
            <center>
              <button
                className="btn btn-primary"
                onClick={() => setSelectedNote(null)}
              >
                Close
              </button>
            </center>
          </div>
        </div>
      </div>
    )}
  </div>
 );
};

export default CounselorAppointments;
```

File: src\dashboard\master_admin\demo.js

```
import React, { useState } from 'react';

const ResponsiveCategoryForm = () => {
  // Mock state for demonstration - replace with your actual state
  const [categories, setCategories] = useState([
    {
      name: 'Sample Category',
      subCategories: [
        {
          name: 'Sample Sub',
          groups: [
            {
              name: 'Sample Group',
              hasTraits: false,
              traits: [],
              fields: []
            }
          ]
        }
      ]
    }
  ]);

  const [selectedCategoryIndex, setSelectedCategoryIndex] = useState(null);
  const [selectedSubCategoryIndex, setSelectedSubCategoryIndex] = useState(null);
  const [selectedGroupIndex, setSelectedGroupIndex] = useState(null);
  const [showSubCategoryCard, setShowSubCategoryCard] = useState(false);
  const [showGroupCard, setShowGroupCard] = useState(false);
  const [showGroupDetailsCard, setShowGroupDetailsCard] = useState(false);
  const [showAddTraitForm, setShowAddTraitForm] = useState(false);
  const [groupFinalizedPopup, setGroupFinalizedPopup] = useState(false);
  const [isSubmitted, setIsSubmitted] = useState(false);
  const [showContentCard, setShowContentCard] = useState(true);

  const [subCategoryInput, setSubCategoryInput] = useState({ name: '', heading: '' });
  const [groupInput, setGroupInput] = useState('');
  const [newTrait, setNewTrait] = useState({ name: '', fields: [] });
  const [newFeature, setNewFeature] = useState({ key: '', value: '', type: 'text' });

  // Mock functions - replace with your actual functions
  const addSubCategory = () => console.log('Add sub category');
  const addGroup = () => console.log('Add group');
  const addTraitField = () => {
    setNewTrait({
      ...newTrait,
      fields: [...newTrait.fields, { key: '', value: '', type: 'text' }]
    });
  };
  const removeTraitField = (index) => {
    setNewTrait({
      ...newTrait,
      fields: newTrait.fields.filter((_, i) => i !== index)
    });
  };
  const updateTraitField = (index, field, value) => {
    const updatedFields = [...newTrait.fields];
    updatedFields[index][field] = value;
    setNewTrait({ ...newTrait, fields: updatedFields });
  };
  const handleSubmitTrait = () => {
    setShowAddTraitForm(false);
    setNewTrait({ name: '', fields: [] });
  };
  const addDirectFeature = () => console.log('Add direct feature');
  const finalizeCategoryFlow = () => console.log('Finalize category');

  return (
    <div className="min-h-screen bg-gray-50 p-2 sm:p-4 lg:p-6">
      <style jsx>{`
        .card {
          background: white;
          border-radius: 12px;
```

```
  box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
  margin-bottom: 1.5rem;
  overflow: hidden;
  transition: all 0.3s ease;
}

.card:hover {
  box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1);
  transform: translateY(-2px);
}

.card-header {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 1rem 1.5rem;
  margin: 0;
  font-size: 1.25rem;
  font-weight: 600;
  border-bottom: 1px solid rgba(255, 255, 255, 0.1);
}

.card-content {
  padding: 1.5rem;
}

.form-group {
  margin-bottom: 1.5rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 600;
  color: #374151;
  font-size: 0.875rem;
  text-transform: uppercase;
  letter-spacing: 0.05em;
}

.form-group input,
.form-group select {
  width: 100%;
  padding: 0.75rem;
  border: 2px solid #e5e7eb;
  border-radius: 8px;
  font-size: 1rem;
  transition: all 0.3s ease;
  background: white;
}

.form-group input:focus,
.form-group select:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.btn {
  padding: 0.75rem 1.5rem;
  border: none;
  border-radius: 8px;
  font-weight: 600;
  font-size: 0.875rem;
  cursor: pointer;
  transition: all 0.3s ease;
  display: inline-flex;
  align-items: center;
  gap: 0.5rem;
  text-decoration: none;
  margin: 0.25rem;
}

.btn:hover {
```

```
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
}

.btn-primary {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
}

.btn-outline-primary {
  background: transparent;
  border: 2px solid #667eea;
  color: #667eea;
}

.btn-outline-primary:hover {
  background: #667eea;
  color: white;
}

.btn-outline-success {
  background: transparent;
  border: 2px solid #10b981;
  color: #10b981;
}

.btn-outline-success:hover {
  background: #10b981;
  color: white;
}

.btn-success {
  background: linear-gradient(135deg, #10b981 0%, #059669 100%);
  color: white;
}

.btn-dark {
  background: linear-gradient(135deg, #374151 0%, #1f2937 100%);
  color: white;
}

.btn-secondary {
  background: #ef4444;
  color: white;
}

.btn-outline-secondary {
  background: transparent;
  border: 2px solid #ef4444;
  color: #ef4444;
}

.btn-outline-secondary:hover {
  background: #ef4444;
  color: white;
}

.grid-fields {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr auto;
  gap: 0.75rem;
  align-items: center;
  padding: 0.75rem;
  border-bottom: 1px solid #e5e7eb;
}

.grid-header {
  background: #f9fafb;
  font-weight: 600;
  color: #374151;
}

.trait-card {
```

```
  background: #f8fafc;
  border: 2px solid #e2e8f0;
  border-radius: 8px;
  padding: 1.5rem;
  margin-top: 1rem;
}

.trait-card h4 {
  margin: 0 0 1rem 0;
  color: #1e293b;
  font-size: 1.25rem;
}

.trait-card ul {
  margin: 0;
  padding-left: 1.5rem;
}

.trait-card li {
  margin-bottom: 0.5rem;
  color: #475569;
}

.center-btn {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 0.5rem;
}

.nested-form {
  background: #f8fafc;
  border: 2px solid #e2e8f0;
  border-radius: 8px;
  padding: 1.5rem;
  margin-top: 1rem;
}

.checkbox-label {
  display: flex !important;
  align-items: center;
  gap: 0.5rem;
  cursor: pointer;
  font-weight: 500 !important;
  text-transform: none !important;
  letter-spacing: normal !important;
}

.checkbox-label input[type="checkbox"] {
  width: auto !important;
  margin: 0 !important;
}

@media (max-width: 768px) {
  .grid-fields {
    grid-template-columns: 1fr;
    gap: 0.5rem;
  }

  .grid-fields > span:last-child {
    justify-self: stretch;
  }

  .btn {
    width: 100%;
    justify-content: center;
    margin: 0.25rem 0;
  }

  .center-btn {
    flex-direction: column;
  }
```

```
    .card-content {
      padding: 1rem;
    }

    .nested-form {
      padding: 1rem;
    }
  }

  @media (max-width: 480px) {
    .card-header {
      padding: 0.75rem 1rem;
      font-size: 1.125rem;
    }

    .form-group input,
    .form-group select {
      padding: 0.625rem;
      font-size: 0.875rem;
    }

    .btn {
      padding: 0.625rem 1rem;
      font-size: 0.8125rem;
    }
  }
`}</style>

{showContentCard && !isSubmitted && (
  <>
    {/* CATEGORY SELECTION & SUB-CATEGORY MODULE */}
    <div className="card">
      <h2 className="card-header">? Sub-Category</h2>
      <div className="card-content">
        <div className="form-group">
          <label>Select Category</label>
          <select
            value={selectedCategoryIndex ?? ""}
            onChange={(e) => {
              const index = Number(e.target.value);
              setSelectedCategoryIndex(index);
              setSelectedSubCategoryIndex(null);
              setSelectedGroupIndex(null);
              setShowSubCategoryCard(false);
              setShowGroupCard(false);
            }}
          >
            <option value="">-- Select --</option>
            {categories.map((cat, index) => (
              <option key={index} value={index}>
                {cat.name}
              </option>
            ))}
          </select>
        </div>

        {selectedCategoryIndex !== null && (
          <div className="center-btn">
            <button
              className="btn btn-outline-primary"
              onClick={() =>
                setShowSubCategoryCard(!showSubCategoryCard)
              }
            >
              {showSubCategoryCard
                ? "? Close Sub-Category Form"
                : "? Add Sub-Category"}
            </button>
          </div>
        )}

        {showSubCategoryCard && (
          <div className="nested-form">
```

```
        <div className="form-group">
          <label>Sub-Category Name</label>
          <input
            type="text"
            value={subCategoryInput.name}
            onChange={(e) =>
              setSubCategoryInput({
                ...subCategoryInput,
                name: e.target.value,
              })
            }
            placeholder="Enter sub-category name"
          />
        </div>
        <div className="form-group">
          <label>Heading/Description</label>
          <input
            type="text"
            value={subCategoryInput.heading}
            onChange={(e) =>
              setSubCategoryInput({
                ...subCategoryInput,
                heading: e.target.value,
              })
            }
            placeholder="Enter heading or description"
          />
        </div>
        <div className="center-btn">
          <button
            className="btn btn-primary"
            onClick={() => {
              addSubCategory();
              setShowSubCategoryCard(false);
            }}
          >
            ? Submit Sub-Category
          </button>
        </div>
      </div>
    )}

    {selectedCategoryIndex !== null &&
      categories[selectedCategoryIndex]?.subCategories.length > 0 && (
        <div className="form-group">
          <label>Select Sub-Category</label>
          <select
            value={selectedSubCategoryIndex ?? ""}
            onChange={(e) => {
              setSelectedSubCategoryIndex(Number(e.target.value));
              setSelectedGroupIndex(null);
              setShowGroupCard(false);
            }}
          >
            <option value="">-- Select --</option>
            {categories[selectedCategoryIndex].subCategories.map(
              (sub, index) => (
                <option key={index} value={index}>
                  {sub.name}
                </option>
              )
            )}
          </select>
        </div>
      )}
  </div>
</div>

{/* GROUP MODULE */}
{selectedSubCategoryIndex !== null && (
  <div className="card">
    <h2 className="card-header">? Group & Traits</h2>
    <div className="card-content">
```

```
<div className="center-btn">
  <button
    className="btn btn-outline-primary"
    onClick={() => setShowGroupCard(!showGroupCard)}
  >
    {showGroupCard ? "? Close Group Form" : "? Add Group"}
  </button>
</div>

{showGroupCard && (
  <div className="nested-form">
    <div className="form-group">
      <label>Group Name</label>
      <input
        type="text"
        value={groupInput}
        onChange={(e) => setGroupInput(e.target.value)}
        placeholder="e.g., FATTY ACIDS"
      />
    </div>
    <div className="center-btn">
      <button
        className="btn btn-primary"
        onClick={() => {
          addGroup();
          setShowGroupCard(false);
        }}
      >
        ? Submit Group
      </button>
    </div>
  </div>
)}

{categories[selectedCategoryIndex].subCategories[
  selectedSubCategoryIndex
]?.groups.length > 0 && (
  <div className="form-group">
    <label>Select Group</label>
    <select
      value={selectedGroupIndex ?? ""}
      onChange={(e) => {
        setSelectedGroupIndex(Number(e.target.value));
        setShowGroupDetailsCard(false);
      }}
    >
      <option value="">-- Select --</option>
      {categories[selectedCategoryIndex].subCategories[
        selectedSubCategoryIndex
      ].groups.map((grp, index) => (
        <option key={index} value={index}>
          {grp.name}
        </option>
      ))}
    </select>
  </div>
)}

{/* GROUP DETAILS MODULE */}
{selectedGroupIndex !== null && (
  <>
    <div className="center-btn">
      <button
        className="btn btn-outline-success"
        onClick={() =>
          setShowGroupDetailsCard(!showGroupDetailsCard)
        }
      >
        {showGroupDetailsCard
          ? "? Close Group Details"
          : "? Add Details (Traits / Features)"}
      </button>
    </div>
```

```
{showGroupDetailsCard && (
  <div className="nested-form">
    <div className="form-group">
      <label className="checkbox-label">
        <input
          type="checkbox"
          checked={
            categories[selectedCategoryIndex]
              .subCategories[selectedSubCategoryIndex]
              .groups[selectedGroupIndex].hasTraits ||
            false
          }
          onChange={(e) => {
            const updated = [...categories];
            updated[selectedCategoryIndex].subCategories[
              selectedSubCategoryIndex
            ].groups[selectedGroupIndex].hasTraits =
              e.target.checked;
            setCategories(updated);
          }}
        />
        This group has traits
      </label>
    </div>

    {/* ? GROUP HAS TRAITS */}
    {categories[selectedCategoryIndex].subCategories[
      selectedSubCategoryIndex
    ].groups[selectedGroupIndex].hasTraits ? (
      <div className="form-group">
        {/* Add Trait Toggle Button */}
        <div className="center-btn">
          <button
            className="btn btn-outline-primary"
            onClick={() =>
              setShowAddTraitForm(!showAddTraitForm)
            }
          >
            {showAddTraitForm
              ? "? Cancel Add Trait"
              : "? Add Trait"}
          </button>
        </div>

        {/* Conditional Trait Form */}
        {showAddTraitForm && (
          <div className="nested-form">
            <div className="form-group">
              <label>Trait Name</label>
              <input
                type="text"
                value={newTrait.name || ""}
                onChange={(e) =>
                  setNewTrait({
                    ...newTrait,
                    name: e.target.value,
                  })
                }
                placeholder="Trait name"
              />
            </div>

            <h5 style={{ marginBottom: "1rem", color: "#374151" }}>
              Trait Fields:
            </h5>

            {newTrait.fields.length > 0 && (
              <div className="grid-fields grid-header">
                <span>Key</span>
                <span>Value</span>
                <span>Type</span>
                <span>Action</span>
```

```
      </div>
    )}

    {newTrait.fields.map((field, index) => (
      <div key={index} className="grid-fields">
        <input
          type="text"
          placeholder="Key"
          value={field.key}
          onChange={(e) =>
            updateTraitField(
              index,
              "key",
              e.target.value
            )
          }
        />
        <input
          type="text"
          placeholder="Value"
          value={field.value}
          onChange={(e) =>
            updateTraitField(
              index,
              "value",
              e.target.value
            )
          }
        />
        <select
          value={field.type}
          onChange={(e) =>
            updateTraitField(
              index,
              "type",
              e.target.value
            )
          }
        >
          <option value="text">Text</option>
          <option value="number">Number</option>
          <option value="date">Date</option>
        </select>
        <button
          className="btn btn-secondary"
          type="button"
          onClick={() => removeTraitField(index)}
        >
          ? Remove
        </button>
      </div>
    ))}

    <div className="center-btn" style={{ marginTop: "1rem" }}>
      <button
        className="btn btn-outline-secondary"
        onClick={addTraitField}
      >
        ? Add Field
      </button>

      <button
        className="btn btn-primary"
        onClick={handleSubmitTrait}
      >
        ? Submit Trait
      </button>
    </div>
  </div>
)}

{/* Display Existing Traits Below */}
<hr style={{ margin: "2rem 0", border: "1px solid #e5e7eb" }} />
```

```
    {categories[selectedCategoryIndex].subCategories[
      selectedSubCategoryIndex
    ].groups[selectedGroupIndex].traits.map(
      (trait, i) => (
        <div key={i} className="trait-card">
          <h4>{trait.name}</h4>
          <ul>
            {trait.fields?.map((f, idx) => (
              <li key={idx}>
                <strong>{f.key}</strong>: {f.value}{" "}
                <em style={{ color: "#64748b" }}>
                  ({f.type})
                </em>
              </li>
            ))}
          </ul>
        </div>
      )
    )}
  </div>
) : (
  /* ? GROUP HAS NO TRAITS ? direct fields */
  <div className="form-group">
    <div className="form-group">
      <label>Field Key</label>
      <input
        type="text"
        value={newFeature.key || ""}
        onChange={(e) =>
          setNewFeature({
            ...newFeature,
            key: e.target.value,
          })
        }
        placeholder="e.g., addiction_chance"
      />
    </div>
    <div className="form-group">
      <label>Field Value</label>
      <input
        type="text"
        value={newFeature.value || ""}
        onChange={(e) =>
          setNewFeature({
            ...newFeature,
            value: e.target.value,
          })
        }
        placeholder="e.g., high"
      />
    </div>
    <div className="form-group">
      <label>Type</label>
      <select
        value={newFeature.type || "text"}
        onChange={(e) =>
          setNewFeature({
            ...newFeature,
            type: e.target.value,
          })
        }
      >
        <option value="text">Text</option>
        <option value="number">Number</option>
        <option value="boolean">Boolean</option>
      </select>
    </div>
    <div className="center-btn">
      <button
        className="btn btn-primary"
        onClick={addDirectFeature}
      >
        ? Add Field
```

```
                            </button>
                          </div>
                          <ul style={{ marginTop: "1rem", paddingLeft: "1.5rem" }}>
                            {categories[
                              selectedCategoryIndex
                            ].subCategories[
                              selectedSubCategoryIndex
                            ].groups[selectedGroupIndex].fields?.map(
                              (f, i) => (
                                <li key={i} style={{ marginBottom: "0.5rem", color: "#475569" }}>
                                  <strong>{f.key}</strong>: {f.value}{" "}
                                  <em style={{ color: "#64748b" }}>
                                    ({f.type})
                                  </em>
                                </li>
                              )
                            )}
                          </ul>
                        </div>
                      )}

                      {/* ? Final Confirm Group Button */}
                      <div className="center-btn" style={{ marginTop: "2rem" }}>
                        <button
                          className="btn btn-success"
                          onClick={() => {
                            setGroupFinalizedPopup(true);
                            setShowGroupDetailsCard(false);
                          }}
                        >
                          ? Confirm This Group
                        </button>
                      </div>
                    </div>
                  )}
                </>
              )}
              <div className="center-btn" style={{ marginTop: "2rem" }}>
                <button
                  className="btn btn-dark"
                  onClick={finalizeCategoryFlow}
                >
                  ? Final Submission for This Category
                </button>
              </div>
            </div>
          </div>
        )}
      </>
    )}
  </div>
);
};

export default ResponsiveCategoryForm;
```

File: src\dashboard\master_admin\DietitianAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const DietitianAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/dietitian`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
```

```
    .catch((err) => console.error("Failed to load appointments:", err));
}, []);

const updateAppointment = (updated) => {
  setAppointments((prev) =>
    prev.map((a) => (a.id === updated.id ? updated : a))
  );
  fetch(`${API_BASE}/${updated.id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Update failed"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  let nextStatus;

  switch (current.status) {
    case "Pending":
      nextStatus = "Approved";
      break;
    case "Approved":
      nextStatus = "Completed";
      break;
    default:
      nextStatus = "Pending";
  }

  const updated = { ...current, status: nextStatus };
  updateAppointment(updated);
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(
    100000000 + Math.random() * 900000000
  )}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = () => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Appointment deleted.");
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
```

```
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Dietitian Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online" ? "badge-orange" : "badge-blue"
                    }`}
                  >
                    {a.appointmentType}
                  </span>
                </td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => setSelectedNote(a.notes)}
                  >
                    View
                  </button>
                </td>
                <td>
                  {a.appointmentType === "Online" ? (
                    <div style={{ display: "flex", alignItems: "center", gap: "8px" }}>
                      <input
                        type="text"
                        className="search-input"
                        value={a.meetingLink || ""}
                        onChange={(e) =>
                          handleMeetingLinkChange(a.id, e.target.value)
                        }
                        placeholder="Enter or generate"
                        style={{
                          width: "180px",
```

```
                        padding: "5px 8px",
                        fontSize: "14px",
                        border: "1px solid #ccc",
                        borderRadius: "6px",
                      }}
                    />
                    <button
                      className="btn btn-primary"
                      onClick={() => handleGenerateLink(a.id)}
                    >
                      Auto
                    </button>
                    <button
                      className="btn btn-primary"
                      style={{ backgroundColor: "#f44336", color: "#fff" }}
                      onClick={() => handleMeetingLinkChange(a.id, "")}
                    >
                      <XCircle size={16} />
                    </button>
                  </div>
                ) : (
                  "-"
                )}
              </td>
              <td>
                <span
                  className={`status-badge ${
                    a.status === "Completed"
                      ? "badge-green"
                      : a.status === "Approved"
                      ? "badge-blue"
                      : "badge-yellow"
                  }`}
                  onClick={() => handleStatusToggle(a.id)}
                  style={{ cursor: "pointer" }}
                >
                  {a.status}
                </span>
              </td>
              <td>
                <button
                  className="btn btn-primary"
                  onClick={() => handleDelete(a.id)}
                >
                  Delete
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  )}

  {selectedNote && (
    <div className="modal-overlay">
      <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
        <h2>Full Notes</h2>
        <p>{selectedNote}</p>
        <div className="center-btn">
          <center>
            <button
              className="btn btn-primary"
              onClick={() => setSelectedNote(null)}
            >
              Close
            </button>
          </center>
        </div>
      </div>
    </div>
  )}
</div>
```

```
  );
};

export default DietitianAppointments;
```

File: src\dashboard\master_admin\DoctorAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const DoctorAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/doctor`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch((err) => console.error("Failed to load appointments:", err));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );

    fetch(`${API_BASE}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(updated),
    }).catch(() => toast.error("Failed to update appointment"));
  };

  const handleStatusToggle = (id) => {
    const current = appointments.find((a) => a.id === id);
    let nextStatus;

    switch (current.status) {
      case "Pending":
        nextStatus = "Approved";
        break;
      case "Approved":
        nextStatus = "Completed";
        break;
      default:
        nextStatus = "Pending";
    }

    const updated = { ...current, status: nextStatus };
    updateAppointment(updated);
    toast.info(`Status updated to: ${nextStatus}`);
  };

  const handleMeetingLinkChange = (id, link) => {
    const updated = appointments.find((a) => a.id === id);
    updated.meetingLink = link;
    updateAppointment(updated);
  };

  const handleGenerateLink = (id) => {
    const dummy = `https://zoom.us/j/${Math.floor(
      100000000 + Math.random() * 900000000
    )}`;
    handleMeetingLinkChange(id, dummy);
  };

  const handleDelete = (id) => {
    const confirmToast = ({ closeToast }) => (
      <div>
```

```
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Appointment deleted.");
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Doctor Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online"
                        ? "badge-orange"
                        : "badge-blue"
                    }`}
                  >
                    {a.appointmentType}
```

```
    </span>
  </td>
  <td>{a.date}</td>
  <td>{a.time}</td>
  <td>
    <button
      className="btn btn-primary"
      onClick={() => setSelectedNote(a.notes)}
    >
      View
    </button>
  </td>
  <td>
    {a.appointmentType === "Online" ? (
      <div style={{ display: "flex", alignItems: "center", gap: "8px" }}>
        <input
          type="text"
          className="search-input"
          value={a.meetingLink || ""}
          onChange={(e) =>
            handleMeetingLinkChange(a.id, e.target.value)
          }
          placeholder="Enter or generate"
          style={{
            width: "180px",
            padding: "5px 8px",
            fontSize: "14px",
            border: "1px solid #ccc",
            borderRadius: "6px",
          }}
        />
        <button
          className="btn btn-primary"
          onClick={() => handleGenerateLink(a.id)}
        >
          Auto
        </button>
        <button
          className="btn btn-primary"
          style={{ backgroundColor: "#f44336", color: "#fff" }}
          onClick={() => handleMeetingLinkChange(a.id, "")}
        >
          <XCircle size={16} />
        </button>
      </div>
    ) : (
      "-"
    )}
  </td>
  <td>
    <span
      className={`status-badge ${
        a.status === "Completed"
          ? "badge-green"
          : a.status === "Approved"
          ? "badge-blue"
          : "badge-yellow"
      }`}
      onClick={() => handleStatusToggle(a.id)}
      style={{ cursor: "pointer" }}
    >
      {a.status}
    </span>
  </td>
  <td>
    <button
      className="btn btn-primary"
      onClick={() => handleDelete(a.id)}
    >
      Delete
    </button>
  </td>
</tr>
```

```
              ))}
            </tbody>
          </table>
        </div>
      )}

      {selectedNote && (
        <div className="modal-overlay">
          <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
            <h2>Full Notes</h2>
            <p>{selectedNote}</p>
            <div className="center-btn">
              <center>
                <button
                  className="btn btn-primary"
                  onClick={() => setSelectedNote(null)}
                >
                  Close
                </button>
              </center>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default DoctorAppointments;
```

File: src\dashboard\master_admin\master_admin.css

```
.user-actions {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1.5rem;
  flex-wrap: wrap;
  gap: 1rem;
}
.toast-message {
  position: fixed;
  top: 20px;
  right: 20px;
  background: #28a745;
  color: white;
  padding: 12px 16px;
  border-radius: 8px;
  z-index: 9999;
  display: flex;
  align-items: center;
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.25);
  gap: 10px;
}x

.toast-message.error {
  background: #dc3545;
}

.toast-message.confirm {
  background: #ff9800;
}

.toast-message.info {
  background: #17a2b8;
}

.toast-actions {
  display: flex;
  gap: 8px;
  margin-left: auto;
}

.btn-sm {
```

```
  padding: 4px 10px;
  font-size: 0.85rem;
}
.card-table {
  background: var(--bg-secondary);
  margin: 2rem 0;
  padding: 1rem;
  border-radius: 12px;
  box-shadow: 0 2px 6px var(--shadow-color);
  overflow-x: auto;
}

.card-table h3 {
  margin-bottom: 1rem;
  color: var(--accent);
}

.card-table table {
  width: 100%;
  border-collapse: collapse;
}

.card-table th,
.card-table td {
  padding: 0.75rem 1rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
  color: var(--text-primary);
}

.close-toast {
  background: transparent;
  border: none;
  color: white;
  font-size: 1.2rem;
  margin-left: auto;
  cursor: pointer;
}

.search-input {
  padding: 0.6rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  width: 250px;
  max-width: 100%;
  font-size: 1rem;
}

.user-table {
  width: 100%;
  border-collapse: collapse;
  background: var(--bg-secondary);
  /* border-radius: 8px; */
  overflow: hidden;
  box-shadow: 0 2px 8px var(--shadow-color);
}

.user-table th,
.user-table td {
  padding: 1rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
}

.user-table th {
  background-color: var(--bg-hover);
  color: var(--text-primary);
}

.btn-remove {
  background-color: #e74c3c;
  color: #fff;
  margin-left: 0.5rem;
```

```
}

.btn-remove:hover {
  background-color: #c0392b;
}
.role-filters {
  display: flex;
  flex-wrap: wrap;
  gap: 0.5rem;
}
.mini-table {
  margin-top: 1rem;
  max-height: 200px;
  overflow-y: auto;
  overflow-x: auto;
}

.mini-table table {
  width: 100%;
  border-collapse: collapse;
  font-size: 0.85rem;
}

.mini-table th,
.mini-table td {
  padding: 0.5rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
  color: var(--text-primary);
}

.filter-btn {
  padding: 0.5rem 1rem;
  border: 1px solid var(--border-color);
  background: var(--bg-secondary);
  color: var(--text-primary);
  border-radius: 6px;
  cursor: pointer;
  transition: all 0.2s ease;
}

.filter-btn.active {
  background-color: var(--accent);
  color: white;
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.6);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 99999;
}
.modal-box {
  background: var(--bg-primary);
  color: var(--text-primary);
  padding: 2rem;
  border-radius: 12px;
  width: 90%;
  max-width: 450px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
  z-index: 10000;
}
.modals-box {
  background: var(--bg-primary);
  color: var(--text-primary);
  padding: 2rem;
  border-radius: 12px;
```

```
  width: 90%;
  max-width: 800px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
  z-index: 10000;

  /* Add these for scroll */
  max-height: 90vh;
  overflow-y: auto;
}
.modalsc-box {
  background: var(--bg-primary);
  color: var(--text-primary);
  padding: 2rem;
  border-radius: 12px;
  border-color: 1px solid #cc5500;
  width: 90%;
  max-width: 500px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
  z-index: 10000;

  /* Add these for scroll */
  max-height: 90vh;
  overflow-y: auto;
}
.modalslot-box {
  background: var(--bg-primary);
  color: var(--text-primary);
  padding: 2rem;
  border-radius: 12px;
  border-color: 1px solid #cc5500;
  width: 90%;
  max-width: 1000px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
  z-index: 10000;

  /* Add these for scroll */
  max-height: 90vh;
  overflow-y: auto;
}
/* Optional: Custom scrollbar styles */
.modals-box::-webkit-scrollbar {
  width: 10px;
}

.modals-box::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 6px;
}

.modals-box::-webkit-scrollbar-track {
  background: #f0f0f0;
}

/* .modal-box {
  background: var(--bg-primary);
  color: var(--text-primary);
  padding: 2rem;
  border-radius: 12px;
  width: 100%;
  max-width: 500px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
  z-index: 10000;
} */

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.modal-body {
  margin-top: 1rem;
}
```

```
.modal-body .form-group {
  margin-bottom: 1rem;
}

.modal-body input,
.modal-body select {
  width: 100%;
  padding: 0.6rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}

.modal-actions {
  display: flex;
  justify-content: flex-end;
  gap: 1rem;
  margin-top: 1rem;
}

.close-modal {
  font-size: 1.5rem;
  border: none;
  background: transparent;
  cursor: pointer;
  color: var(--text-primary);
}
.overview-grid {
  display: flex;
  flex-direction: column;
  gap: 1.5rem;
}

.card {
  flex: 1 1 calc(50% - 1rem);
  background: var(--bg-secondary);
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 2px 6px var(--shadow-color);
}

.card h3 {
  margin-bottom: 1rem;
  color: var(--accent);
}

.charts-section {
  display: flex;
  flex-wrap: wrap;
  gap: 1.5rem;
}

.chart {
  flex: 1 1 100%;
  min-width: 300px;
}

@media (max-width: 768px) {
  .overview-grid .card,
  .charts-section .chart {
    flex: 1 1 100%;
  }
}
/* PatientJourney  */
.journey-search {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  margin-bottom: 1.5rem;
}
```

```
.journey-search input {
  padding: 0.6rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  flex: 1;
  min-width: 200px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}

.journey-card {
  background: var(--bg-secondary);
  border-radius: 8px;
  padding: 1.5rem;
  box-shadow: 0 2px 6px var(--shadow-color);
}

.journey-steps {
  list-style: none;
  padding: 0;
  margin-top: 1rem;
}

.journey-steps li {
  padding: 0.6rem 0;
  border-bottom: 1px solid var(--border-color);
}

.error-text {
  color: red;
  margin-top: 0.5rem;
}
.status {
  display: inline-flex;
  align-items: center;
  gap: 0.5rem;
  font-weight: 500;
}

.status.complete {
  color: #2ecc71; /* green */
}

.status.pending {
  color: #e67e22; /* orange */
}

.icon {
  margin-top: 1px;
}
/*all reports */
.report-search {
  margin-bottom: 1.5rem;
  display: flex;
  justify-content: flex-start;
}

.report-search input {
  padding: 0.6rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  width: 250px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}
.table-responsive {
  width: 100%;
  overflow-x: auto;
}

.user-table {
  width: 100%;
  min-width: 600px; /* Prevent table from squashing too much */
```

```
  border-collapse: collapse;
}

.user-table th,
.user-table td {
  padding: 12px 16px;
  border: 1px solid #ddd;
  text-align: left;
}

/* Optional: make scroll bar stylish */
.table-responsive::-webkit-scrollbar {
  height: 6px;
}
.table-responsive::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 3px;
}

/* activity log */
.log-search {
  margin-bottom: 1.5rem;
}

.log-search input {
  padding: 0.6rem 1rem;
  width: 100%;
  max-width: 400px;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}

.activity-log-list {
  list-style: none;
  padding: 0;
}

.activity-log-item {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  padding: 1rem;
  margin-bottom: 0.75rem;
  background: var(--bg-secondary);
  border-left: 4px solid var(--accent);
  border-radius: 6px;
  box-shadow: 0 2px 4px var(--shadow-color);
}

.timestamp {
  font-size: 0.9rem;
  color: var(--text-secondary);
  flex: 1 1 100%;
}

.actor,
.mrn,
.action {
  flex: 1 1 auto;
  font-weight: 500;
}
.log-controls {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  margin-bottom: 1.5rem;
  align-items: center;
}

.log-controls input[type="text"],
.log-controls select,
```

```
.log-controls textarea,
.log-controls input[type="date"] {
  padding: 0.6rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}

/* Sticky header */
.activity-log-table thead th {
  position: sticky;
  top: 0;
  background-color: var(--bg-secondary);
  z-index: 1;
}

/* Role badges */
.role-badge {
  padding: 4px 8px;
  border-radius: 6px;
  font-size: 12px;
  font-weight: 600;
  color: white;
}

.badge-lab {
  background-color: #6c757d;
}
.badge-Phlebotomist {
  background-color: red;
}
.badge-doctor {
  background-color: #007bff;
}
.badge-physio {
  background-color: #28a745;
}
.badge-dietitian {
  background-color: #cc5500;
}

/* Optional: highlight search match */
mark {
  background-color: yellow;
  padding: 0 2px;
  border-radius: 2px;
}

/* Pagination info */
.pagination-info {
  margin-top: 1rem;
  font-size: 14px;
  text-align: center;
}

/* Loading */
.loading-text {
  font-style: italic;
  color: gray;
  padding: 1rem;
  text-align: center;
}

/* SecurityControls */
.success-text {
  color: var(--accent);
  margin-bottom: 1rem;
}
.modal-overlay {
  position: fixed;
  inset: 0;
  background: rgba(0, 0, 0, 0.5);
```

```
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 99;
}

/* .modal-box {
  background: var(--bg-primary);
  padding: 2rem;
  border-radius: 12px;
  width: 90%;
  max-width: 400px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.25);
} */

.modal-actions {
  display: flex;
  justify-content: flex-end;
  gap: 1rem;
  margin-top: 1.5rem;
}

.cancel-btn,
.save-btn {
  padding: 0.6rem 1.2rem;
  border-radius: 6px;
  cursor: pointer;
}

.cancel-btn {
  background: #ccc;
  color: #333;
}

.save-btn {
  background: var(--accent);
  color: #fff;
}
.modal-box input {
  width: 100%;
  margin-top: 0.5rem;
  padding: 0.6rem;
  border-radius: 6px;
  border: 1px solid var(--border-color);
  background: var(--bg-secondary);
  color: var(--text-primary);
}

.form-group input + input {
  margin-top: 0.75rem;
}
.role-select {
  padding: 0.5rem 1rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  font-size: 14px;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  cursor: pointer;
  width: 100%;
}

/* Optional focus style */
.role-select:focus {
  outline: none;
  border-color: var(--accent);
  box-shadow: 0 0 0 2px rgba(204, 85, 0, 0.2);
}

/*services */
/* --- Service Form Custom Layout Fixes --- */

form.card input,
```

```
form.card textarea {
  width: 100%;
  padding: 10px;
  font-size: 1rem;
  margin-bottom: 12px;
  border: 1px solid #ccc;
  border-radius: 6px;
  background: var(--bg-secondary);
  color: var(--text-primary);
}

/* Fixing textarea height + font */
form.card textarea {
  resize: vertical;
  height: 100px;
}

/* 3-column layout for pricing options */
form.card .pricing-row {
  display: flex;
  gap: 12px;
  flex-wrap: wrap;
}

form.card .pricing-row input {
  flex: 1 1 calc(33.33% - 8px);
  min-width: 150px;
}

/* Align Add Service button to center */
form.card .center-btn {
  display: flex;
  justify-content: center;
  margin-top: 20px;
}

form.card .center-btn .btn {
  padding: 10px 24px;
  font-size: 1rem;
  background-color: var(--accent, #cc5500);
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
}

/* Adjust Add Feature button spacing */
form.card .btn-secondary {
  margin-top: 10px;
}

/* Program features input alignment */
form.card .d-flex input {
  flex: 1;
}

/* --- Flex row for features with mobile responsiveness --- */
form.card .d-flex {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  gap: 6px;
  margin-bottom: 16px;
}

/* Input should take most space */
form.card .d-flex input {
  width: 100%;
  height: 38px;
  padding: 8px 12px;
  font-size: 1rem;
  border-radius: 6px;
  border: 1px solid #ccc;
```

```
}

/* Button styling */
form.card .d-flex .btn-secondary {
  height: 32px;
  padding: 0 12px;
  font-size: 0.85rem;
  border-radius: 5px;
  background-color: #ff6600;
  color: white;
  border: none;
  cursor: pointer;
  align-self: flex-start;
}
form.card .d-flex {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  gap: 4px;
  margin-bottom: 12px;
}

form.card .d-flex input {
  width: 100%;
  height: 38px;
  padding: 8px 12px;
  font-size: 1rem;
  border-radius: 6px;
  border: 1px solid #ccc;
}

/* Smaller, left-aligned remove button */
form.card .d-flex .btn-secondary {
  height: 32px;
  padding: 4px 12px; /* reduced vertical padding */
  font-size: 0.85rem;
  border-radius: 5px;
  background-color: #ff6600;
  color: white;
  border: none;
  cursor: pointer;
  align-self: flex-start;
  line-height: 1.2; /* compact line height */
}

form.card .d-flex .btn-secondary:hover {
  background-color: #e65c00;
}

/* Responsive fallback */
@media screen and (max-width: 600px) {
  form.card .d-flex {
    flex-direction: column;
    align-items: stretch;
  }

  form.card .d-flex .btn-secondary {
    width: 100%;
    text-align: center;
  }
}
.service-list {
  list-style: none;
  padding: 0;
  margin-top: 20px;
}

.service-list-item {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  gap: 10px;
  justify-content: space-between;
```

```
  padding: 12px 16px;
  border: 1px solid #ccc;
  border-radius: 8px;
  margin-bottom: 12px;
  background-color: var(--card-bg, #fff);
}

.service-title {
  font-weight: 600;
  flex: 1 1 150px;
}

.service-sub {
  flex: 1 1 120px;
  color: gray;
}

.service-price {
  flex: 1 1 180px;
  color: #555;
}

.action-buttons {
  display: flex;
  gap: 10px;
}
.blog-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
  margin-top: 20px;
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.6);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
}

.modal-content {
  background: white;
  padding: 30px;
  border-radius: 12px;
  width: 90%;
  max-width: 600px;
  max-height: 90vh;
  overflow-y: auto;
}
/* appoitment*/
.appointments-container {
  padding: 30px;
  text-align: center;
}

.appointment-card-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(240px, 1fr));
  gap: 30px;
  margin-top: 30px;
}

.appointment-card {
  background-color: #fff;
  border: 1px solid #ddd;
  padding: 25px;
  border-radius: 12px;
```

```
  cursor: pointer;
  transition: transform 0.2s, box-shadow 0.2s;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.05);
  text-align: center;
}

.appointment-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

.card-icon {
  font-size: 40px;
  margin-bottom: 10px;
}

.view-btn {
  margin-top: 15px;
  padding: 8px 16px;
  background-color: #4a90e2;
  color: white;
  border: none;
  border-radius: 6px;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.2s;
}

.view-btn:hover {
  background-color: #357abd;
}
.appointments-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(260px, 1fr));
  gap: 20px;
  margin-top: 30px;
}

.appointment-type-card {
  cursor: pointer;
  text-align: center;
  padding: 20px;
  transition: transform 0.2s ease, box-shadow 0.2s ease;
}

.appointment-type-card:hover {
  transform: translateY(-4px);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

.appointment-icon {
  font-size: 36px;
  margin-bottom: 10px;
}
/* CounselorAppointments*/
.appointments-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

.appointments-table th,
.appointments-table td {
  border: 1px solid #ddd;
  padding: 10px;
  text-align: center;
}

.type-badge,
.status-badge {
  padding: 5px 10px;
  border-radius: 8px;
  font-weight: bold;
```

```
  color: white;
}

.badge-orange {
  background-color: #e67e22;
}

.badge-blue {
  background-color: #2980b9;
}

.badge-green {
  background-color: #27ae60;
}

.badge-yellow {
  background-color: #f1c40f;
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  height: 100vh;
  width: 100vw;
  background-color: rgba(0, 0, 0, 0.6);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
}

.modal-content {
  background: white;
  padding: 30px;
  border-radius: 10px;
  width: 90%;
  max-width: 600px;
  max-height: 80vh;
  overflow-y: auto;
}
.table-responsive {
  overflow-x: auto;
  width: 100%;
}
/*passsword request*/
.status-tag {
  padding: 4px 8px;
  border-radius: 4px;
  font-size: 13px;
  font-weight: 500;
  text-transform: uppercase;
}

.status-tag.pending {
  background-color: #fff3cd;
  color: #856404;
}

.status-tag.approved {
  background-color: #d4edda;
  color: #155724;
}

.status-tag.rejected {
  background-color: #f8d7da;
  color: #721c24;
}

.btn-success {
  background-color: #28a745;
  color: white;
  border: none;
```

```
  margin-right: 0.5rem;
  padding: 5px 12px;
  border-radius: 4px;
}

.btn-danger {
  background-color: #dc3545;
  color: white;
  border: none;
  padding: 5px 12px;
  border-radius: 4px;
}
.scrollable-notes-table {
  max-height: 180px; /* Adjust to fit ~3 entries */
  overflow-y: auto;
}

.scrollable-notes-table::-webkit-scrollbar {
  width: 8px;
}

.scrollable-notes-table::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 4px;
}

.scrollable-notes-table::-webkit-scrollbar-track {
  background-color: transparent;
}
.category-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));
  gap: 1rem;
}
.modalsc-box {
  scrollbar-width: thin;
  scrollbar-color: #cc5500 #f5f5f5; /* Firefox */
}

/* Chrome, Edge, Safari */
.modalsc-box::-webkit-scrollbar {
  width: 8px;
}

.modalsc-box::-webkit-scrollbar-track {
  background: #f5f5f5;
  border-radius: 10px;
}

.modalsc-box::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 10px;
  border: 2px solid transparent;
  background-clip: content-box;
}
.dashboard-cards {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  justify-content: flex-start;
  margin-bottom: 20px;
}

.dashboard-card {
  padding: 12px;
  min-width: 240px;
  max-width: 100%;
  border: 1px solid #cc5500;
  background-color: transparent;
  text-align: center;
  font-size: 20px;
  border-radius: 8px;
  flex: 0 0 auto;
```

```
  flex-basis: calc(33.33% - 20px); /* 3 cards per row by default */
}

/* Medium screens (tablets) - 2 cards per row */
@media (max-width: 768px) {
  .dashboard-card {
    flex-basis: calc(50% - 20px);
  }
}

/* Small screens (mobile) - 1 card per row */
@media (max-width: 480px) {
  .dashboard-card {
    flex-basis: 100%;
  }
}
```

File: src\dashboard\master_admin\master_admin_dashboard.js

```
import React, { useEffect, useState } from "react";
import { useTheme } from "../../ThemeProvider";
import axios from "axios";
import {
  BarChart,
  Bar,
  PieChart,
  Pie,
  Cell,
  Tooltip,
  ResponsiveContainer,
  LineChart,
  Line,
  XAxis,
  YAxis,
  Legend,
  CartesianGrid,
} from "recharts";
import { FiChevronDown, FiChevronUp, FiDownload } from "react-icons/fi";

import "./master_admin.css";

const COLORS = ["#0088FE", "#00C49F", "#FFBB28", "#FF8042", "#AA00FF"];

const ChartBox = ({ title, children, exportData }) => {
  const [collapsed, setCollapsed] = useState(false);

  const handleExport = () => {
    if (!exportData || exportData.length === 0) return;
    const keys = Object.keys(exportData[0]);
    const csv = [
      keys.join(","),
      ...exportData.map((row) => keys.map((k) => row[k]).join(",")),
    ].join("\n");
    const blob = new Blob([csv], { type: "text/csv;charset=utf-8;" });
    const link = document.createElement("a");
    link.href = URL.createObjectURL(blob);
    link.download = `${title.replace(/\s+/g, "_")}.csv`;
    link.click();
  };

  return (
    <div className="chart-box">
      <div className="chart-header">
        <h3>{title}</h3>
        <div className="chart-actions">
          {exportData && (
            <FiDownload
              className="icon-btn"
              onClick={handleExport}
              title="Export to CSV"
            />
          )}
```

```
          {collapsed ? (
            <FiChevronDown
              className="icon-btn"
              onClick={() => setCollapsed(false)}
              title="Expand"
            />
          ) : (
            <FiChevronUp
              className="icon-btn"
              onClick={() => setCollapsed(true)}
              title="Collapse"
            />
          )}
        </div>
      </div>
      {!collapsed && children}
    </div>
  );
};

const BasicPage = () => {
  const { theme } = useTheme();
  const [metrics, setMetrics] = useState(null);

  useEffect(() => {
    axios
      .get("http://localhost:3001/dashboardMetrics")
      .then((res) => setMetrics(res.data));
  }, []);

  if (!metrics)
    return (
      <div className={`app ${theme}`}>
        <p>Loading dashboard metrics...</p>
      </div>
    );

  return (
    <div className={`app ${theme}`}>
      <div className="dashboard-main">
        <h1>Master Admin Dashboard</h1>

        {/* Small, responsive, wrapping role cards */}
        <div className="dashboard-cards">
          {metrics.roleCounts.map((r, i) => (
            <div className="dashboard-card" key={i}>
              <h4 style={{ margin: 0, color: "#cc5500" }}>{r.role}</h4>
              <p style={{ fontSize: "20px", margin: "6px 0" }}>
                {r.count} Registered
              </p>
            </div>
          ))}
        </div>

        <div className="dashboard-charts">
          {/* Appointments Status */}
          <div className="col card">
            <ChartBox
              title="Appointments Status"
              exportData={metrics.appointmentsStatus}
            >
              <ResponsiveContainer width="100%" height={300}>
                <BarChart data={metrics.appointmentsStatus}>
                  <CartesianGrid strokeDasharray="3 3" />
                  <XAxis dataKey="role" />
                  <YAxis />
                  <Tooltip />
                  <Legend />
                  <Bar dataKey="completed" stackId="a" fill="#82ca9d" />
                  <Bar dataKey="pending" stackId="a" fill="#ffbb28" />
                </BarChart>
              </ResponsiveContainer>
            </ChartBox>
```

```
</div>

{/* Daily Appointment Trends */}
<div className="col card">
  <ChartBox
    title="Daily Appointment Trends"
    exportData={metrics.dailyAppointments}
  >
    <ResponsiveContainer width="100%" height={300}>
      <LineChart data={metrics.dailyAppointments}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="date" />
        <YAxis />
        <Tooltip />
        <Legend />
        {Object.keys(metrics.dailyAppointments[0])
          .filter((k) => k !== "date")
          .map((key, i) => (
            <Line
              key={i}
              type="monotone"
              dataKey={key}
              stroke={COLORS[i % COLORS.length]}
            />
          ))}
      </LineChart>
    </ResponsiveContainer>
  </ChartBox>
</div>

{/* Top Doctors */}
<div className="col card">
  <ChartBox
    title="Top 5 Busiest Doctors"
    exportData={metrics.topDoctors}
  >
    <ResponsiveContainer width="100%" height={300}>
      <BarChart data={metrics.topDoctors} layout="vertical">
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis type="number" />
        <YAxis dataKey="name" type="category" />
        <Tooltip />
        <Bar dataKey="count" fill="#cc5500" />
      </BarChart>
    </ResponsiveContainer>
  </ChartBox>
</div>

{/* Report Contributions */}
<div className="col card">
  <ChartBox
    title="Report Contributions"
    exportData={metrics.reportContributions}
  >
    <ResponsiveContainer width="100%" height={300}>
      <PieChart>
        <Pie
          data={metrics.reportContributions}
          dataKey="value"
          nameKey="role"
          cx="50%"
          cy="50%"
          outerRadius={100}
          label
        >
          {metrics.reportContributions.map((entry, index) => (
            <Cell
              key={`cell-${index}`}
              fill={COLORS[index % COLORS.length]}
            />
          ))}
        </Pie>
        <Tooltip />
```

```
              </PieChart>
            </ResponsiveContainer>
          </ChartBox>
        </div>

        {/* Patient-to-Role Ratio */}
        <div className="col card">
          <ChartBox
            title="Patient-to-Role Ratio"
            exportData={metrics.roleAssignment}
          >
            <ResponsiveContainer width="100%" height={300}>
              <PieChart>
                <Pie
                  data={metrics.roleAssignment}
                  dataKey="patients"
                  nameKey="role"
                  cx="50%"
                  cy="50%"
                  outerRadius={100}
                  label
                >
                  {metrics.roleAssignment.map((entry, index) => (
                    <Cell
                      key={`cell-${index}`}
                      fill={COLORS[index % COLORS.length]}
                    />
                  ))}
                </Pie>
                <Tooltip />
              </PieChart>
            </ResponsiveContainer>
          </ChartBox>
        </div>
      </div>
    </div>
  );
};

export default BasicPage;
```

File: src\dashboard\master_admin\PatientJourney.js

```
import React, { useState } from "react";
import axios from "axios";
import { useTheme } from "../../ThemeProvider";
import { toast, ToastContainer } from "react-toastify";
import { CheckCircle, Hourglass } from "lucide-react";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const PatientJourney = () => {
  const { theme } = useTheme();
  const [mrn, setMrn] = useState("");
  const [patient, setPatient] = useState(null);

  const fetchJourney = async () => {
    if (!mrn.trim()) {
      toast.error("Please enter an MRN.");
      setPatient(null);
      return;
    }

    try {
      const res = await axios.get(`http://localhost:3001/patients?mrn=${mrn}`);
      if (res.data.length === 0) {
        toast.error("No patient found.");
        setPatient(null);
      } else {
        setPatient(res.data[0]);
      }
    } catch (err) {
      console.error(err);
```

```
    toast.error("Failed to fetch patient data.");
  }
};

const getStatusBadge = (condition) =>
  condition ? (
    <span className="status-badge badge-green">
      <CheckCircle size={16} style={{ marginRight: "5px" }} />
      Completed
    </span>
  ) : (
    <span className="status-badge badge-yellow">
      <Hourglass size={16} style={{ marginRight: "5px" }} />
      Pending
    </span>
  );

const isResolved = (p) =>
  p.reportPdfUrl &&
  (p.assignedTo.physio || p.assignedTo.dietitian) &&
  p.prescription.fileUrl;

const handleKeyDown = (e) => {
  if (e.key === "Enter") {
    fetchJourney();
  }
};

return (
  <div className={`dashboard-main ${theme}`}>
    <ToastContainer position="top-center" autoClose={2500} />
    <h1>Patient Journey Viewer</h1>

    <div className="journey-search">
      <input
        type="text"
        placeholder="Enter MRN"
        value={mrn}
        onChange={(e) => setMrn(e.target.value.toUpperCase())}
        onKeyDown={handleKeyDown}
      />
      <button className="btn btn-primary" onClick={fetchJourney}>
        Search
      </button>
    </div>

    {patient && (
      <div className="journey-card">
        <h2 style={{ color: "orangered" }}>
          {patient.name} ({patient.mrn})
        </h2>

        <div className="table-responsive">
          <table className="appointments-table">
            <thead>
              <tr>
                <th>Step</th>
                <th>Status</th>
                <th>Reports</th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <td>Sample Processed</td>
                <td>{getStatusBadge(!!patient.reportPdfUrl)}</td>
                <td>
                  {patient.reportPdfUrl ? (
                    <a
                      href={patient.reportPdfUrl}
                      target="_blank"
                      rel="noopener noreferrer"
                      className="btn btn-primary"
                    >
```

```
                      View Report
                    </a>
                  ) : (
                    "-"
                  )}
                </td>
              </tr>
              <tr>
                <td>Assigned to Physio</td>
                <td>{getStatusBadge(patient.assignedTo.physio)}</td>
                <td>-</td>
              </tr>
              <tr>
                <td>Assigned to Dietitian</td>
                <td>{getStatusBadge(patient.assignedTo.dietitian)}</td>
                <td>-</td>
              </tr>
              <tr>
                <td>Prescription Uploaded</td>
                <td>{getStatusBadge(!!patient.prescription.fileUrl)}</td>
                <td>
                  {patient.prescription.fileUrl ? (
                    <a
                      href={patient.prescription.fileUrl}
                      target="_blank"
                      rel="noopener noreferrer"
                      className="btn btn-primary"
                    >
                      View Prescription
                    </a>
                  ) : (
                    "-"
                  )}
                </td>
              </tr>
              <tr>
                <td>Final Resolution</td>
                <td>{getStatusBadge(isResolved(patient))}</td>
                <td>-</td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    )}
  </div>
 );
};

export default PatientJourney;
```

File: src\dashboard\master_admin\PhlebotomistAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const PhlebotomistAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);
  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/phlebotomist`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch(() => toast.error("Failed to fetch appointments"));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
```

```
  );
  fetch(`${API_BASE}/${updated.id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Update failed"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  const next =
    current.status === "Pending"
      ? "Approved"
      : current.status === "Approved"
      ? "Completed"
      : "Pending";
  updateAppointment({ ...current, status: next });
  toast.info(`Status updated to ${next}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  updated.meetingLink = link;
  updateAppointment(updated);
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(100000000 + Math.random() * 900000000)}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  const confirmToast = () => (
    <div>
      <p>Confirm delete?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
              setAppointments((prev) => prev.filter((a) => a.id !== id))
            );
            toast.dismiss();
            toast.error("Deleted");
          }}
        >
          Confirm
        </button>
        <button className="btn btn-primary" onClick={() => toast.dismiss()}>
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeButton: false,
    draggable: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Phlebotomist Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
```

```
      <thead>
        <tr>
          <th>MRN</th>
          <th>Patient</th>
          <th>Type</th>
          <th>Date</th>
          <th>Time</th>
          <th>Notes</th>
          <th>Meeting Link</th>
          <th>Status</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {appointments.map((a) => (
          <tr key={a.id}>
            <td>{a.mrn}</td>
            <td>{a.patientName}</td>
            <td>
              <span className={`type-badge ${a.appointmentType === "Online" ? "badge-orange" : "badge-blue
                {a.appointmentType}
              </span>
            </td>
            <td>{a.date}</td>
            <td>{a.time}</td>
            <td>
              <button className="btn btn-primary" onClick={() => setSelectedNote(a.notes)}>View</button>
            </td>
            <td>
              {a.appointmentType === "Online" ? (
                <div style={{ display: "flex", gap: "6px", alignItems: "center" }}>
                  <input
                    className="search-input"
                    value={a.meetingLink || ""}
                    onChange={(e) => handleMeetingLinkChange(a.id, e.target.value)}
                    placeholder="Enter/Auto"
                  />
                  <button className="btn btn-primary" onClick={() => handleGenerateLink(a.id)}>Auto</butto
                  <button className="btn btn-primary" onClick={() => handleMeetingLinkChange(a.id, "")}>
                    <XCircle size={16} />
                  </button>
                </div>
              ) : "-"}
            </td>
            <td>
              <span
                className={`status-badge ${
                  a.status === "Completed" ? "badge-green" : a.status === "Approved" ? "badge-blue" : "bad
                }`}
                onClick={() => handleStatusToggle(a.id)}
                style={{ cursor: "pointer" }}
              >
                {a.status}
              </span>
            </td>
            <td>
              <button className="btn btn-primary" onClick={() => handleDelete(a.id)}>Delete</button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
)}

{selectedNote && (
  <div className="modal-overlay">
    <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
      <h2>Full Notes</h2>
      <p>{selectedNote}</p>
      <div className="center-btn">
        <button className="btn btn-primary" onClick={() => setSelectedNote(null)}>Close</button>
      </div>
```

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

export default PhlebotomistAppointments;
```

File: src\dashboard\master_admin\PhysioAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const PhysioAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);
  const API_BASE = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_BASE}/role/physio`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch(() => toast.error("Failed to fetch appointments"));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );
    fetch(`${API_BASE}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(updated),
    }).catch(() => toast.error("Update failed"));
  };

  const handleStatusToggle = (id) => {
    const current = appointments.find((a) => a.id === id);
    const next =
      current.status === "Pending"
        ? "Approved"
        : current.status === "Approved"
        ? "Completed"
        : "Pending";
    updateAppointment({ ...current, status: next });
    toast.info(`Status updated to ${next}`);
  };

  const handleMeetingLinkChange = (id, link) => {
    const updated = appointments.find((a) => a.id === id);
    updated.meetingLink = link;
    updateAppointment(updated);
  };

  const handleGenerateLink = (id) => {
    const dummy = `https://zoom.us/j/${Math.floor(100000000 + Math.random() * 900000000)}`;
    handleMeetingLinkChange(id, dummy);
  };

  const handleDelete = (id) => {
    const confirmToast = () => (
      <div>
        <p>Confirm delete?</p>
        <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
          <button
            className="btn btn-primary"
            onClick={() => {
              fetch(`${API_BASE}/${id}`, { method: "DELETE" }).then(() =>
                setAppointments((prev) => prev.filter((a) => a.id !== id))
              );
```

```
            toast.dismiss();
            toast.error("Deleted");
          }}
        >
          Confirm
        </button>
        <button className="btn btn-primary" onClick={() => toast.dismiss()}>
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeButton: false,
    draggable: false,
  });
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Physio Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN</th>
              <th>Patient</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn}</td>
                <td>{a.patientName}</td>
                <td>
                  <span className={`type-badge ${a.appointmentType === "Online" ? "badge-orange" : "badge-blue
                    {a.appointmentType}
                  </span>
                </td>
                <td>{a.date}</td>
                <td>{a.time}</td>
                <td>
                  <button className="btn btn-primary" onClick={() => setSelectedNote(a.notes)}>View</button>
                </td>
                <td>
                  {a.appointmentType === "Online" ? (
                    <div style={{ display: "flex", gap: "6px", alignItems: "center" }}>
                      <input
                        className="search-input"
                        value={a.meetingLink || ""}
                        onChange={(e) => handleMeetingLinkChange(a.id, e.target.value)}
                        placeholder="Enter/Auto"
                      />
                      <button className="btn btn-primary" onClick={() => handleGenerateLink(a.id)}>Auto</butto
                      <button className="btn btn-primary" onClick={() => handleMeetingLinkChange(a.id, "")}>
                        <XCircle size={16} />
                      </button>
                    </div>
```

```
                  ) : "-"}
                </td>
                <td>
                  <span
                    className={`status-badge ${
                      a.status === "Completed" ? "badge-green" : a.status === "Approved" ? "badge-blue" : "bad
                    }`}
                    onClick={() => handleStatusToggle(a.id)}
                    style={{ cursor: "pointer" }}
                  >
                    {a.status}
                  </span>
                </td>
                <td>
                  <button className="btn btn-primary" onClick={() => handleDelete(a.id)}>Delete</button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}

    {selectedNote && (
      <div className="modal-overlay">
        <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
          <h2>Full Notes</h2>
          <p>{selectedNote}</p>
          <div className="center-btn">
            <button className="btn btn-primary" onClick={() => setSelectedNote(null)}>Close</button>
          </div>
        </div>
      </div>
    )}
  </div>
 );
};

export default PhysioAppointments;
```

File: src\dashboard\master_admin\ReportBuilder.js

```
// ReportBuilder.js

import React, { useState, useEffect } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";
import "../physio/assign.css";

const ReportBuilder = () => {
  const API_URL = "http://localhost:3001/categories";
  const [dnlId, setDnlId] = useState("");

  const [categories, setCategories] = useState([]);
  const [newCategory, setNewCategory] = useState("");

  const [subCategoryInput, setSubCategoryInput] = useState({
    name: "",
    heading: "",
  });
  const [groupInput, setGroupInput] = useState("");

  const [selectedCategoryIndex, setSelectedCategoryIndex] = useState(null);
  const [selectedSubCategoryIndex, setSelectedSubCategoryIndex] =
    useState(null);
  const [selectedGroupIndex, setSelectedGroupIndex] = useState(null);

  const [newTrait, setNewTrait] = useState({ name: "", fields: [] });
  const [showAddTraitForm, setShowAddTraitForm] = useState(false);

  const [newTraitFeature, setNewTraitFeature] = useState({ label: "" });
  const [newFeature, setNewFeature] = useState({ label: "" });
```

```
const [isSubmitted, setIsSubmitted] = useState(false);
const [showPopup, setShowPopup] = useState(false);
const [popupData, setPopupData] = useState(null);

const [showAddCategoryCard, setShowAddCategoryCard] = useState(false);
const [showContentCard, setShowContentCard] = useState(false);
const [showSubCategoryCard, setShowSubCategoryCard] = useState(false);
const [showGroupCard, setShowGroupCard] = useState(false);
const [showGroupDetailsCard, setShowGroupDetailsCard] = useState(false);
const [newTraitFeatures, setNewTraitFeatures] = useState({});
const [mrn, setMrn] = useState("");
const [selectedPatient, setSelectedPatient] = useState(null);
const [isAssigned, setIsAssigned] = useState(false);
const [deleteLock, setDeleteLock] = useState(false);

const [subCategoryAddedPopup, setSubCategoryAddedPopup] = useState(false);
const [groupAddedPopup, setGroupAddedPopup] = useState(false);
const [groupFinalizedPopup, setGroupFinalizedPopup] = useState(false);
const finalizeCategoryFlow = () => {
  if (selectedCategoryIndex !== null) {
    const categoryName =
      categories[selectedCategoryIndex]?.name || "selected category";
    toast.success(`All data is saved for the ${categoryName}`, {
      position: "top-center",
    });
  }

  // Collapse all open forms/cards
  setShowSubCategoryCard(false); // ? Close sub-category form
  setShowGroupCard(false); // ? Close group form
  setShowGroupDetailsCard(false); // ? Close trait/feature form
  setShowContentCard(false); // ? CLOSE THIS TOO (Fix for your issue)

  // Reset selections
  setSelectedCategoryIndex(null);
  setSelectedSubCategoryIndex(null);
  setSelectedGroupIndex(null);
};

useEffect(() => {
  axios
    .get(API_URL)
    .then((res) => setCategories(res.data))
    .catch(() =>
      toast.error("Failed to load categories", { position: "top-center" })
    );
}, []);
const handleSearch = () => {
  if (!mrn.trim()) return;

  axios
    .get(`http://localhost:3001/patients?mrn=${mrn}`)
    .then((res) => {
      if (res.data.length > 0) {
        const patient = res.data[0];

        setSelectedPatient(patient);
        setDnlId(patient.dnlId || ""); // NEW: Store dnlId from backend

        toast.success(`Patient ${patient.name} found!`, {
          position: "top-center",
        });
      } else {
        setSelectedPatient(null);
        setDnlId(""); // Clear dnlId if no patient found
        toast.error("No patient found with this MRN", {
          position: "top-center",
        });
      }
    })
    .catch(() => {
      setSelectedPatient(null);
```

```
      setDnlId(""); // Ensure clean state on error
      toast.error("Error fetching patient", {
        position: "top-center",
      });
    });
};

const addCategory = () => {
  if (!newCategory.trim()) return;
  const newEntry = { name: newCategory, subCategories: [] };
  axios
    .post(API_URL, newEntry)
    .then((res) => {
      setCategories([...categories, res.data]);
      toast.success("Category added successfully!", {
        position: "top-center",
      });
      setNewCategory("");
      setShowAddCategoryCard(false);
    })
    .catch(() =>
      toast.error("Failed to add category", { position: "top-center" })
    );
};
const addTraitField = () => {
  setNewTrait({
    ...newTrait,
    fields: [...newTrait.fields, { key: "", value: "" }],
  });
};

const addSubCategory = () => {
  if (selectedCategoryIndex === null || !subCategoryInput.name.trim()) return;

  const updated = [...categories];
  updated[selectedCategoryIndex].subCategories.push({
    name: subCategoryInput.name,
    heading: subCategoryInput.heading,
    groups: [],
  });

  const id = updated[selectedCategoryIndex].id;
  axios
    .put(`${API_URL}/${id}`, updated[selectedCategoryIndex])
    .then(() => {
      setCategories(updated);
      setSubCategoryInput({ name: "", heading: "" });
      toast.success("Sub-Category added and saved", {
        position: "top-center",
      });
    })
    .catch(() =>
      toast.error("Failed to save Sub-Category", { position: "top-center" })
    );
};

const addGroup = () => {
  if (
    selectedCategoryIndex === null ||
    selectedSubCategoryIndex === null ||
    !groupInput.trim()
  )
    return;

  const updated = [...categories];
  updated[selectedCategoryIndex].subCategories[
    selectedSubCategoryIndex
  ].groups.push({
    name: groupInput,
    hasTraits: null,
    traits: [],
    directFeatures: [],
  });
```

```
  const id = updated[selectedCategoryIndex].id;
  axios
    .put(`${API_URL}/${id}`, updated[selectedCategoryIndex])
    .then(() => {
      setCategories(updated);
      setGroupInput("");
      toast.success("Group added and saved", { position: "top-center" });
    })
    .catch(() =>
      toast.error("Failed to save group", { position: "top-center" })
    );
};
const removeTraitField = (index) => {
  const updatedFields = newTrait.fields.filter((_, i) => i !== index);
  setNewTrait({ ...newTrait, fields: updatedFields });
};

const handleSubmitTrait = () => {
  if (!newTrait.name.trim() || newTrait.fields.length === 0) {
    toast.error("Trait name and at least one field are required", {
      position: "top-center",
    });
    return;
  }

  const updated = [...categories];

  updated[selectedCategoryIndex].subCategories[
    selectedSubCategoryIndex
  ].groups[selectedGroupIndex].traits.push({
    name: newTrait.name,
    fields: newTrait.fields,
  });

  const id = updated[selectedCategoryIndex].id;

  axios
    .put(`${API_URL}/${id}`, updated[selectedCategoryIndex])
    .then(() => {
      setCategories(updated);
      setNewTrait({ name: "", fields: [] });
      setShowAddTraitForm(false); // ? hide form after submit
      toast.success("Trait added and saved", { position: "top-center" });
    })
    .catch(() =>
      toast.error("Failed to save trait", { position: "top-center" })
    );
};
const exportJSON = () => {
  const primaryCategory = categories[0]?.name || "Untitled";

  // Transform all categories to backend format
  const transformedCategories = categories.map(({ id, ...cat }) => ({
    ...cat,
    subCategories: cat.subCategories.map((sub) => ({
      name: sub.name,
      groups: sub.groups.map((grp) => ({
        name: grp.name,
        hasTraits: grp.hasTraits,
        ...(grp.hasTraits
          ? {
              traits: grp.traits.map((trait) => ({
                name: trait.name,
                fields: trait.fields, // Traits still include type
              })),
            }
          : {
              fields: (grp.fields || []).map(({ key, value }) => ({
                key,
                value,
              })), // ? Remove type from direct fields
            }),
```

```
      })),
    })),
  }));

  const payload = {
    dnlId: mrn,
    reportName: `${primaryCategory} Comprehensive Report`,
    categories: transformedCategories,
  };

  const blob = new Blob([JSON.stringify(payload, null, 2)], {
    type: "application/json",
  });
  const link = document.createElement("a");
  link.href = URL.createObjectURL(blob);
  link.download = `${dnlId || "report"}_report.json`;
  link.click();
};

const addTraitFeature = (traitIndex) => {
  const label = newTraitFeatures[traitIndex];
  if (!label?.trim()) return;

  const updated = [...categories];
  updated[selectedCategoryIndex].subCategories[
    selectedSubCategoryIndex
  ].groups[selectedGroupIndex].traits[traitIndex].features.push({ label });

  const id = updated[selectedCategoryIndex].id;
  axios
    .put(`${API_URL}/${id}`, updated[selectedCategoryIndex])
    .then(() => {
      setCategories(updated);
      setNewTraitFeatures({
        ...newTraitFeatures,
        [traitIndex]: "", // Clear only the one you added
      });
      toast.success("Feature added and saved", { position: "top-center" });
    })
    .catch(() =>
      toast.error("Failed to save feature", { position: "top-center" })
    );
};
const updateTraitField = (index, key, value) => {
  const updatedFields = [...(newTrait.fields || [])];
  updatedFields[index][key] = value;
  setNewTrait({ ...newTrait, fields: updatedFields });
};

const addDirectFeature = () => {
  const updated = [...categories];
  const group =
    updated[selectedCategoryIndex].subCategories[selectedSubCategoryIndex]
      .groups[selectedGroupIndex];

  if (!group.fields) group.fields = [];

  group.fields.push({
    key: newFeature.key,
    value: newFeature.value,
  });

  const id = updated[selectedCategoryIndex].id;
  axios
    .put(`${API_URL}/${id}`, updated[selectedCategoryIndex])
    .then(() => {
      setCategories(updated);
      setNewFeature({ key: "", value: "" });
      toast.success("Field added and saved", { position: "top-center" });
    })
    .catch(() =>
      toast.error("Failed to save field", { position: "top-center" })
    );
```

```
};

const updateCategory = (updatedCategory, index) => {
  const id = categories[index]?.id;
  if (!id) return;
  return axios.put(`${API_URL}/${id}`, updatedCategory);
};

const handleSubmit = () => {
  if (!dnlId || categories.length === 0) {
    toast.error("Missing DNL ID or categories", { position: "top-center" });
    return;
  }

  const primaryCategory = categories[0]?.name || "Untitled";

  const transformedCategories = categories.map(({ id, ...cat }) => ({
    ...cat,
    subCategories: cat.subCategories.map((sub) => ({
      name: sub.name,
      groups: sub.groups.map((grp) => ({
        name: grp.name,
        hasTraits: grp.hasTraits,
        ...(grp.hasTraits
          ? {
              traits: grp.traits.map((trait) => ({
                name: trait.name,
                fields: trait.fields,
              })),
            }
          : {
              fields: grp.fields || [],
            }),
      })),
    })),
  }));

  const payload = {
    dnlId: mrn,
    reportName: `${primaryCategory} Comprehensive Report`,
    categories: transformedCategories,
  };

  axios
    .post("http://localhost:3001/reports", payload)
    .then(() => {
      toast.success("Report Template Created Successfully!", {
        position: "top-center",
      });
      setIsSubmitted(true);
    })
    .catch(() =>
      toast.error("Submission failed", { position: "top-center" })
    );
};

const handleView = (category) => {
  setPopupData(category);
  setShowPopup(true);
};

return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={3000} />
    <h1 className="card-header">? Report Builder</h1>
    <div className="card">
      <h2 className="card-header">? Search Patient by MRN</h2>
      <div
        className="form-group"
        style={{
          display: "flex",
          alignItems: "center",
          gap: "1rem",
```

```
      flexWrap: "wrap",
    }}
  >
    <input
      type="text"
      placeholder="Enter MRN Number"
      className="search-input"
      value={mrn}
      onChange={(e) => setMrn(e.target.value.toUpperCase())}
      onKeyDown={(e) => {
        if (e.key === "Enter") {
          e.preventDefault();
          handleSearch();
        }
      }}
      style={{
        flex: "1",
        minWidth: "200px",
        padding: "0.75rem",
        fontSize: "1rem",
        border: "1px solid #cc5500",
        borderRadius: "4px",
      }}
    />

    <button
      className="btn btn-primary"
      onClick={handleSearch}
      style={{
        padding: "0.75rem 1.5rem",
        fontSize: "1rem",
        whiteSpace: "nowrap",
      }}
    >
      Search
    </button>
  </div>
</div>

{selectedPatient && !isAssigned && (
  <>
    {/* CategoryBuilder Form JSX goes here */}
    {/* === ACTION BUTTONS === */}
    <div className="row">
      <div className="col">
        <button
          className="btn btn-primary"
          onClick={() => setShowAddCategoryCard(!showAddCategoryCard)}
        >
          {showAddCategoryCard ? "Close Add Category" : "? Add Category"}
        </button>
      </div>
      {categories.length > 0 && (
        <div className="col">
          <button
            className="btn btn-secondary"
            onClick={() => setShowContentCard(!showContentCard)}
          >
            {showContentCard
              ? "Close Content Panel"
              : "? Add Content to the Category"}
          </button>
        </div>
      )}
    </div>

    {/* === ADD CATEGORY CARD === */}
    {showAddCategoryCard && (
      <div className="card" style={{ marginTop: "1rem" }}>
        <div className="form-group">
          <label>New Category Name</label>
          <input
            type="text"
```

```
        value={newCategory}
        onChange={(e) => setNewCategory(e.target.value)}
        placeholder="e.g., Nutrition"
      />
      <button
        className="btn btn-primary"
        style={{ marginTop: "0.75rem" }}
        onClick={addCategory}
      >
        Submit
      </button>
    </div>
  </div>
)}

{/* === CONTENT PANEL === */}
{showContentCard && !isSubmitted && (
  <>
    {/* CATEGORY SELECTION & SUB-CATEGORY MODULE */}
    <div className="card">
      <h2 className="card-header">? Sub-Category</h2>
      <div className="form-group">
        <label>Select Category</label>
        <select
          value={selectedCategoryIndex ?? ""}
          onChange={(e) => {
            const index = Number(e.target.value);
            setSelectedCategoryIndex(index);
            setSelectedSubCategoryIndex(null);
            setSelectedGroupIndex(null);
            setShowSubCategoryCard(false);
            setShowGroupCard(false);
          }}
        >
          <option value="">-- Select --</option>
          {categories.map((cat, index) => (
            <option key={index} value={index}>
              {cat.name}
            </option>
          ))}
        </select>
      </div>

      {selectedCategoryIndex !== null && (
        <>
          <button
            className="btn btn-outline-primary"
            onClick={() =>
              setShowSubCategoryCard(!showSubCategoryCard)
            }
          >
            {showSubCategoryCard
              ? "? Close Sub-Category Form"
              : "? Add Sub-Category"}
          </button>

          {showSubCategoryCard && (
            <div className="card" style={{ marginTop: "1rem" }}>
              <div className="form-group">
                <label>Sub-Category Name</label>
                <input
                  type="text"
                  value={subCategoryInput.name}
                  onChange={(e) =>
                    setSubCategoryInput({
                      ...subCategoryInput,
                      name: e.target.value,
                    })
                  }
                />
                <label>Heading/Description</label>
                <input
                  type="text"
```

```
              value={subCategoryInput.heading}
              onChange={(e) =>
                setSubCategoryInput({
                  ...subCategoryInput,
                  heading: e.target.value,
                })
              }
            />
            <button
              className="btn btn-primary"
              onClick={() => {
                addSubCategory();
                setShowSubCategoryCard(false);
              }}
            >
              ? Submit Sub-Category
            </button>
          </div>
        </div>
      )}
    </>
  )}

  {selectedCategoryIndex !== null &&
    categories[selectedCategoryIndex]?.subCategories.length >
      0 && (
      <div className="form-group">
        <label>Select Sub-Category</label>
        <select
          value={selectedSubCategoryIndex ?? ""}
          onChange={(e) => {
            setSelectedSubCategoryIndex(Number(e.target.value));
            setSelectedGroupIndex(null);
            setShowGroupCard(false);
          }}
        >
          <option value="">-- Select --</option>
          {categories[selectedCategoryIndex].subCategories.map(
            (sub, index) => (
              <option key={index} value={index}>
                {sub.name}
              </option>
            )
          )}
        </select>
      </div>
    )}
</div>

{/* GROUP MODULE */}
{selectedSubCategoryIndex !== null && (
  <div className="card">
    <h2 className="card-header">? Group & Traits</h2>
    <button
      className="btn btn-outline-primary"
      onClick={() => setShowGroupCard(!showGroupCard)}
    >
      {showGroupCard ? "? Close Group Form" : "? Add Group"}
    </button>

    {showGroupCard && (
      <div className="card" style={{ marginTop: "1rem" }}>
        <div className="form-group">
          <label>Group Name</label>
          <input
            type="text"
            value={groupInput}
            onChange={(e) => setGroupInput(e.target.value)}
            placeholder="e.g., FATTY ACIDS"
          />
          <button
            className="btn btn-primary"
            onClick={() => {
```

```
          addGroup();
          setShowGroupCard(false);
        }}
      >
        ? Submit Group
      </button>
    </div>
  </div>
)}

{categories[selectedCategoryIndex].subCategories[
  selectedSubCategoryIndex
]?.groups.length > 0 && (
  <div className="form-group">
    <label>Select Group</label>
    <select
      value={selectedGroupIndex ?? ""}
      onChange={(e) => {
        setSelectedGroupIndex(Number(e.target.value));
        setShowGroupDetailsCard(false);
      }}
    >
      <option value="">-- Select --</option>
      {categories[selectedCategoryIndex].subCategories[
        selectedSubCategoryIndex
      ].groups.map((grp, index) => (
        <option key={index} value={index}>
          {grp.name}
        </option>
      ))}
    </select>
  </div>
)}

{/* GROUP DETAILS MODULE */}
{selectedGroupIndex !== null && (
  <>
    <button
      className="btn btn-outline-success"
      onClick={() =>
        setShowGroupDetailsCard(!showGroupDetailsCard)
      }
    >
      {showGroupDetailsCard
        ? "? Close Group Details"
        : "? Add Details (Traits / Features)"}
    </button>

    {showGroupDetailsCard && (
      <div className="card" style={{ marginTop: "1rem" }}>
        <div className="form-group">
          <label>
            <input
              type="checkbox"
              checked={
                categories[selectedCategoryIndex]
                  .subCategories[selectedSubCategoryIndex]
                  .groups[selectedGroupIndex].hasTraits ||
                false
              }
              onChange={(e) => {
                const updated = [...categories];
                updated[selectedCategoryIndex].subCategories[
                  selectedSubCategoryIndex
                ].groups[selectedGroupIndex].hasTraits =
                  e.target.checked;
                setCategories(updated);
              }}
            />{" "}
            This group has traits
          </label>
        </div>
```

```
{/* ? GROUP HAS TRAITS */}
{categories[selectedCategoryIndex].subCategories[
  selectedSubCategoryIndex
].groups[selectedGroupIndex].hasTraits ? (
  <div className="form-group">
    {/* Add Trait Toggle Button */}
    <button
      className="btn btn-outline-primary"
      onClick={() =>
        setShowAddTraitForm(!showAddTraitForm)
      }
    >
      {showAddTraitForm
        ? "? Cancel Add Trait"
        : "? Add Trait"}
    </button>

    {/* Conditional Trait Form */}
    {showAddTraitForm && (
      <div
        style={{
          marginTop: "1rem",
          border: "1px solid #ccc",
          padding: "1rem",
          borderRadius: "6px",
        }}
      >
        <label>Trait Name</label>
        <input
          type="text"
          value={newTrait.name || ""}
          onChange={(e) =>
            setNewTrait({
              ...newTrait,
              name: e.target.value,
            })
          }
          placeholder="Trait name"
          style={{
            width: "100%",
            padding: "0.5rem",
            marginTop: "0.5rem",
            borderRadius: "4px",
            border: "1px solid #ccc",
          }}
        />

        <h5
          style={{
            marginTop: "1rem",
            marginBottom: "0.5rem",
          }}
        >
          Trait Fields:
        </h5>

        {newTrait.fields.length > 0 && (
          <div
            style={{
              display: "grid",
              gridTemplateColumns: "1fr 1fr 1fr auto",
              backgroundColor: "#f8f9fa",
              padding: "0.5rem",
              fontWeight: "600",
              color: "#cc5500",
              borderBottom: "1px solid #ccc",
            }}
          >
            <span>Key</span>
            <span>Value</span>
            <span>Type</span>
            <span>Action</span>
          </div>
```

```
)}

{newTrait.fields.map((field, index) => (
  <div
    key={index}
    style={{
      display: "grid",
      gridTemplateColumns: "1fr 1fr 1fr auto",
      gap: "0.5rem",
      alignItems: "center",
      borderBottom: "1px solid #eee",
      padding: "0.5rem 0",
    }}
  >
    <input
      type="text"
      placeholder="Key"
      value={field.key}
      onChange={(e) =>
        updateTraitField(
          index,
          "key",
          e.target.value
        )
      }
      style={{
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
      }}
    />
    <input
      type="text"
      placeholder="Value"
      value={field.value}
      onChange={(e) =>
        updateTraitField(
          index,
          "value",
          e.target.value
        )
      }
      style={{
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
      }}
    />
    <button
      className="btn btn-secondary"
      type="button"
      onClick={() => removeTraitField(index)}
      style={{
        padding: "0.5rem 1rem",
        backgroundColor: "#cc5500",
        color: "white",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
      }}
    >
      ? Remove
    </button>
  </div>
))}

<div
  style={{
    marginTop: "1rem",
    display: "flex",
    gap: "1rem",
    flexWrap: "wrap",
  }}
```

```
          >
            <button
              className="btn btn-outline-secondary"
              onClick={addTraitField}
              style={{
                padding: "0.5rem 1rem",
                border: "1px solid #cc5500",
                color: "#cc5500",
                borderRadius: "4px",
                backgroundColor: "transparent",
                cursor: "pointer",
              }}
            >
              ? Add Field
            </button>

            <button
              className="btn btn-primary"
              onClick={handleSubmitTrait}
              style={{
                padding: "0.5rem 1rem",
                backgroundColor: "#cc5500",
                color: "white",
                border: "none",
                borderRadius: "4px",
                cursor: "pointer",
              }}
            >
              ? Submit Trait
            </button>
          </div>
        </div>
      )}

      {/* Display Existing Traits Below */}
      <hr style={{ marginTop: "2rem" }} />
      {categories[selectedCategoryIndex].subCategories[
        selectedSubCategoryIndex
      ].groups[selectedGroupIndex].traits.map(
        (trait, i) => (
          <div
            key={i}
            className="card"
            style={{
              marginTop: "1rem",
              padding: "1rem",
              border: "1px solid #ddd",
              borderRadius: "6px",
            }}
          >
            <h4>{trait.name}</h4>
            <ul>
              {trait.fields?.map((f, idx) => (
                <li key={idx}>
                  <strong>{f.key}</strong>: {f.value}{" "}
                  <em style={{ color: "gray" }}>
                    ({f.type})
                  </em>
                </li>
              ))}
            </ul>
          </div>
        )
      )}
    </div>
  ) : (
    /* ? GROUP HAS NO TRAITS ? direct fields */
    <div className="form-group">
      <label>Field Key</label>
      <input
        type="text"
        value={newFeature.key || ""}
        onChange={(e) =>
```

```
                setNewFeature({
                  ...newFeature,
                  key: e.target.value,
                })
              }
              placeholder="e.g., addiction_chance"
            />
            <label>Field Value</label>
            <input
              type="text"
              value={newFeature.value || ""}
              onChange={(e) =>
                setNewFeature({
                  ...newFeature,
                  value: e.target.value,
                })
              }
              placeholder="e.g., high"
            />

            <button
              className="btn btn-primary"
              style={{ marginTop: "0.75rem" }}
              onClick={addDirectFeature}
            >
              Add Field
            </button>
            <ul style={{ marginTop: "0.5rem" }}>
              {categories[
                selectedCategoryIndex
              ].subCategories[
                selectedSubCategoryIndex
              ].groups[selectedGroupIndex].fields?.map(
                (f, i) => (
                  <li key={i}>
                    <strong>{f.key}</strong>: {f.value}{" "}
                    <em style={{ color: "gray" }}>
                      ({f.type})
                    </em>
                  </li>
                )
              )}
            </ul>
          </div>
        )}

        {/* ? Final Confirm Group Button */}
        <div
          className="center-btn"
          style={{ marginTop: "1rem" }}
        >
          <button
            className="btn btn-success"
            onClick={() => {
              setGroupFinalizedPopup(true);
              setShowGroupDetailsCard(false);
            }}
          >
            ? Confirm This Group
          </button>
        </div>
      </div>
    )}
  </>
)}
<div className="center-btn" style={{ marginTop: "1rem" }}>
  <button
    className="btn btn-dark"
    onClick={finalizeCategoryFlow}
  >
    ? Final Submission for This Category
  </button>
</div>
```

```
      </div>
    )}
  </>
)}
{/* === DYNAMIC RENDERING OF ALL SUBCATEGORIES === */}

{/* === SUBMISSION VIEW AFTER SUBMIT === */}
{isSubmitted && (
  <div className="card">
    <h2 className="card-header">? Submitted Categories</h2>
    <div
      style={{
        display: "grid",
        gridTemplateColumns: "repeat(auto-fill, minmax(300px, 1fr))",
        gap: "1rem",
      }}
    >
      {categories.map((cat, index) => (
        <div key={index} className="card">
          <h3>{cat.name}</h3>
          <p>{cat.subCategories.length} subcategories</p>
          <button
            className="btn btn-secondary"
            onClick={() => handleView(cat)}
          >
            View
          </button>
        </div>
      ))}
    </div>
  </div>
)}
{/* === PREVIOUSLY ADDED CATEGORIES SECTION === */}
<div className="card">
  <h2 className="card-header">? Previously Added Categories</h2>

  {categories.length === 0 ? (
    <center>
      <p
        style={{
          padding: "1rem",
          color: "gray",
          fontStyle: "italic",
        }}
      >
        No categories added yet.
      </p>
    </center>
  ) : (
    <div
      style={{
        display: "grid",
        gridTemplateColumns: "repeat(auto-fill, minmax(280px, 1fr))",
        gap: "1rem",
      }}
    >
      {categories.map((cat, index) => (
        <div key={index} className="card" style={{ padding: "1rem" }}>
          <h3 style={{ marginBottom: "1rem", color: "#cc5500" }}>
            {cat.name}
          </h3>
          <div
            style={{
              display: "flex",
              justifyContent: "space-between",
            }}
          >
            <button
              className="btn btn-secondary"
              onClick={() => handleView(cat)}
            >
              ?? View
            </button>
```

```
<button
  className="btn btn-primary"
  onClick={() => {
    setSelectedCategoryIndex(index);
    setShowContentCard(true);
    toast.info(`Editing category: ${cat.name}`, {
      position: "top-center",
    });
  }}
>
  ?? Edit
</button>
<button
  className="btn btn-danger"
  onClick={() => {
    if (deleteLock) return; // ? prevent duplicate dialogs
    setDeleteLock(true); // ? lock delete popup

    toast(
      ({ closeToast }) => (
        <div>
          <p>
            Are you sure you want to delete{" "}
            <b>{cat.name}</b>?
          </p>
          <div
            style={{
              display: "flex",
              gap: "10px",
              marginTop: "10px",
            }}
          >
            <button
              className="btn btn-danger"
              onClick={() => {
                const id = cat.id;
                axios
                  .delete(`${API_URL}/${id}`)
                  .then(() => {
                    setCategories(
                      categories.filter(
                        (_, i) => i !== index
                      )
                    );
                    setDeleteLock(false); // ? unlock on confirm
                    toast.dismiss();
                    toast.error("Category deleted", {
                      position: "top-center",
                    });
                  });
              }}
            >
              Confirm
            </button>
            <button
              className="btn btn-secondary"
              onClick={() => {
                setDeleteLock(false); // ? unlock on cancel
                toast.dismiss();
                toast.info("Deletion cancelled", {
                  position: "top-center",
                });
              }}
            >
              Cancel
            </button>
          </div>
        </div>
      ),
      {
        position: "top-center",
        autoClose: false,
        draggable: false,
```

```
                    closeButton: false,
                  }
                );
              }}
            >
              ?? Delete
            </button>
          </div>
        </div>
      ))}
    </div>
  )}
</div>

{/* === POPUPS === */}

{groupFinalizedPopup && (
  <div className="modal-overlay">
    <div className="modalsc-box">
      <h2>Group Finalized ?</h2>
      <p>All traits and features saved successfully.</p>
      <div className="center-btn">
        <button
          className="btn btn-primary"
          onClick={() => setGroupFinalizedPopup(false)}
        >
          Close
        </button>
      </div>
    </div>
  </div>
)}

{showPopup && popupData && (
  <div className="modal-overlay">
    <div
      className="modalsc-box"
      style={{
        maxWidth: "60vw",
        maxHeight: "90vh",
        overflowY: "auto",
      }}
    >
      <h2>?? Category Details: {popupData.name}</h2>

      {popupData.subCategories?.map((sub, i) => (
        <div key={i} className="card" style={{ marginTop: "1rem" }}>
          <h3>? Sub-Category: {sub.name}</h3>
          <p>
            <b>Description:</b> {sub.heading || "N/A"}
          </p>

          {sub.groups?.map((grp, j) => (
            <div
              key={j}
              style={{ marginLeft: "1rem", marginTop: "0.5rem" }}
            >
              <h4>? Group: {grp?.name || "Unnamed Group"}</h4>
              <p>
                <b>Has Traits:</b> {grp?.hasTraits ? "Yes" : "No"}
              </p>

              {grp?.hasTraits && Array.isArray(grp.traits) ? (
                grp.traits.map((trait, tIdx) => (
                  <div
                    key={tIdx}
                    style={{
                      marginLeft: "1rem",
                      marginTop: "0.5rem",
                    }}
                  >
                    <p>
                      <b>Trait:</b> {trait?.name || "Unnamed Trait"}
```

```
                    </p>
                    {Array.isArray(trait.fields) &&
                    trait.fields.length > 0 ? (
                      <ul>
                        {trait.fields.map((field, fIdx) => (
                          <li key={fIdx}>
                            <strong>{field.key || "key"}:</strong>{" "}
                            {field.value || "N/A"}{" "}
                            <em style={{ color: "gray" }}>
                              ({field.type || "text"})
                            </em>
                          </li>
                        ))}
                      </ul>
                    ) : (
                      <p
                        style={{ marginLeft: "1rem", color: "#999" }}
                      >
                        No fields
                      </p>
                    )}
                  </div>
                ))
              ) : grp?.fields?.length > 0 ? (
                <>
                  <p>
                    <b>Fields:</b>
                  </p>
                  <ul>
                    {grp.fields.map((field, fIdx) => (
                      <li key={fIdx}>
                        <strong>{field.key || "key"}:</strong>{" "}
                        {field.value || "N/A"}
                        {/* ? Removed (type) display here */}
                      </li>
                    ))}
                  </ul>
                </>
              ) : (
                <p style={{ marginLeft: "1rem", color: "#999" }}>
                  No traits or fields available.
                </p>
              )}
            </div>
          ))}
        </div>
      ))}

      <div className="center-btn" style={{ marginTop: "1rem" }}>
        <button
          className="btn btn-primary"
          onClick={() => setShowPopup(false)}
        >
          Close
        </button>
      </div>
    </div>
  </div>
)}
{/* ? "Assign Report" button at the end */}
<div className="center-btn" style={{ marginTop: "1rem" }}>
  <button className="btn btn-secondary" onClick={exportJSON}>
    ? Export JSON
  </button>
</div>

<div className="center-btn" style={{ marginTop: "2rem" }}>
  <center>
    <button
      className="btn btn-primary"
      onClick={() => {
        setIsAssigned(true);
        toast.success("Report assigned and saved!", {
```

```
                  position: "top-center",
                });
              }}
            >
              ? Assign Report
            </button>
          </center>
        </div>
      </>
    )}
  </div>
 );
};
export default ReportBuilder;
```

File: src\dashboard\master_admin\SecurityControls.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../ThemeProvider";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const SecurityControls = () => {
  const { theme } = useTheme();
  const [users, setUsers] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [search, setSearch] = useState("");
  const [confirmModal, setConfirmModal] = useState({ visible: false, user: null, action: null });
  const [newPassword, setNewPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");

  useEffect(() => {
    fetchUsers();
  }, []);

  useEffect(() => {
    const lower = search.toLowerCase();
    setFiltered(
      users.filter(
        (u) =>
          u.name.toLowerCase().includes(lower) ||
          u.email.toLowerCase().includes(lower)
      )
    );
  }, [search, users]);

  const fetchUsers = async () => {
    try {
      const res = await axios.get("http://localhost:8080/api/users");
      setUsers(res.data);
    } catch {
      toast.error("Failed to fetch users.");
    }
  };

  const logSecurityAction = async (userId, action, actor = "Admin") => {
    try {
      await axios.post("http://localhost:8080/api/securityLogs", {
        userId,
        action,
        actor
      });
    } catch {
      console.error("Logging failed");
    }
  };

  const handleRoleChange = async (userId, newRole) => {
    try {
      await axios.patch(`http://localhost:8080/api/users/${userId}`, {
        role: newRole,
      });
```

```
    await logSecurityAction(userId, `Role changed to ${newRole}`);
    toast.success(`Role updated for user ID ${userId} ? ${newRole}`);
    fetchUsers();
  } catch {
    toast.error("Role update failed.");
  }
};

const confirmAction = (user, action) => {
  setConfirmModal({ visible: true, user, action });
  setNewPassword("");
  setConfirmPassword("");
};

const executeConfirmedAction = async () => {
  const { user, action } = confirmModal;

  if (action === "logout") {
    toast.warn(`User ID ${user.id} has been logged out`);
    await logSecurityAction(user.id, "Forced Logout");
    setConfirmModal({ visible: false });
  }

  if (action === "reset") {
    if (!newPassword || !confirmPassword) return toast.warning("Fill all password fields.");
    if (newPassword !== confirmPassword) return toast.error("Passwords do not match.");

    try {
      await axios.patch(`http://localhost:8080/api/users/${user.id}`, {
        password: newPassword,
      });
      await logSecurityAction(user.id, "Password Reset");
      toast.success(`Password reset for ${user.name}`);
      setConfirmModal({ visible: false });
    } catch {
      toast.error("Password update failed.");
    }
  }
};

const getPasswordStrength = () => {
  if (newPassword.length < 6) return "Weak";
  if (newPassword.match(/[A-Z]/) && newPassword.match(/\d/)) return "Strong";
  return "Moderate";
};

const exportSecurityLog = async () => {
  try {
    const res = await axios.get("http://localhost:8080/api/securityLogs");
    const csv = [
      ["Timestamp", "User ID", "Action", "Actor"],
      ...res.data.map((l) => [
        new Date(l.timestamp).toLocaleString(),
        l.userId,
        l.action,
        l.actor,
      ]),
    ];
    const csvContent =
      "data:text/csv;charset=utf-8," + csv.map((e) => e.join(",")).join("\n");
    const link = document.createElement("a");
    link.setAttribute("href", encodeURI(csvContent));
    link.setAttribute("download", "security_logs.csv");
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
    toast.success("Security log exported.");
  } catch {
    toast.error("Failed to export logs.");
  }
};

return (
```

```
<div className={`dashboard-main ${theme}`}>
  <ToastContainer position="top-center" autoClose={2000} />
  <h1>Security Controls</h1>

  <div className="log-controls">
    <input
      type="text"
      placeholder="Search by name or email"
      value={search}
      onChange={(e) => setSearch(e.target.value)}
    />
    <button className="btn btn-primary" onClick={exportSecurityLog}>
      Export Logs
    </button>
  </div>

  <div className="table-responsive">
    <table className="user-table">
      <thead>
        <tr>
          <th>Name</th>
          <th>Email</th>
          <th>Current Role</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {filtered.map((u) => (
          <tr key={u.id}>
            <td>{u.name}</td>
            <td>{u.email}</td>
            <td>
              <select
                value={u.role}
                onChange={(e) => handleRoleChange(u.id, e.target.value)}
                className={`role-select ${u.role}`}
              >
                <option value="doctor">Doctor</option>
                <option value="physio">Physio</option>
                <option value="dietitian">Dietitian</option>
                <option value="lab">Lab</option>
                <option value="phlebotomist">Phlebotomist</option>
              </select>
            </td>
            <td>
              <button className="btn btn-primary" onClick={() => confirmAction(u, "reset")}>
                Reset Password
              </button>
              <button className="btn btn-primary" onClick={() => confirmAction(u, "logout")}>
                Force Logout
              </button>
            </td>
          </tr>
        ))}
        {filtered.length === 0 && (
          <tr>
            <td colSpan="4" className="error-text">
              No users found.
            </td>
          </tr>
        )}
      </tbody>
    </table>
  </div>

  {/* Confirmation Modal */}
  {confirmModal.visible && (
    <div className="modal-overlay" onClick={() => setConfirmModal({ visible: false })}>
      <div className="modal-box" onClick={(e) => e.stopPropagation()}>
        <h3>
          {confirmModal.action === "reset" ? "Password Reset" : "Force Logout"}
        </h3>
        <p>
```

```
              {confirmModal.action === "reset" ? (
                <>
                  Changing password for <strong>{confirmModal.user.name}</strong> (
                  {confirmModal.user.email})
                </>
              ) : (
                <>
                  Are you sure you want to <strong>logout</strong> user ID{" "}
                  {confirmModal.user.id}?
                </>
              )}
            </p>

            {confirmModal.action === "reset" && (
              <>
                <input
                  type="password"
                  placeholder="New Password"
                  value={newPassword}
                  onChange={(e) => setNewPassword(e.target.value)}
                />
                <input
                  type="password"
                  placeholder="Confirm Password"
                  value={confirmPassword}
                  onChange={(e) => setConfirmPassword(e.target.value)}
                />
                {newPassword && (
                  <p className="strength-msg">Strength: {getPasswordStrength()}</p>
                )}
              </>
            )}

            <div className="modal-actions">
              <button className="btn btn-primary" onClick={() => setConfirmModal({ visible: false })}>
                Cancel
              </button>
              <button className="btn btn-primary" onClick={executeConfirmedAction}>
                Confirm
              </button>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default SecurityControls;
```

File: src\dashboard\master_admin\Services.css

```
/* Services Component Styles */
.dashboard-main {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  transition: all 0.3s ease;
}

/* Theme styles */
.dashboard-main.light {
  background-color: #f8f9fa;
  color: #333;
}

.dashboard-main.dark {
  background-color: #1a1a1a;
  color: #e0e0e0;
}

/* Main heading */
.dashboard-main h1 {
```

```
  text-align: center;
  margin-bottom: 30px;
  font-size: 2.5rem;
  font-weight: 700;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  background-clip: text;
}

.dashboard-main h2 {
  margin: 40px 0 20px 0;
  font-size: 2rem;
  font-weight: 600;
  color: #4a5568;
}

.dashboard-main.dark h2 {
  color: #cbd5e0;
}

/* Card styles */
.card {
  background: #ffffff;
  border-radius: 12px;
  padding: 30px;
  margin-bottom: 30px;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.08);
  border: 1px solid #e2e8f0;
  transition: all 0.3s ease;
}

.card:hover {
  box-shadow: 0 8px 30px rgba(0, 0, 0, 0.12);
  transform: translateY(-2px);
}

.dashboard-main.dark .card {
  background: #2d3748;
  border-color: #4a5568;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
}

.dashboard-main.dark .card:hover {
  box-shadow: 0 8px 30px rgba(0, 0, 0, 0.4);
}

/* Form styles */
form.card {
  max-width: 800px;
  margin: 0 auto 40px auto;
}

/* Label styles */
label {
  display: block;
  margin: 20px 0 8px 0;
  font-weight: 600;
  font-size: 1.1rem;
  color: #2d3748;
}

.dashboard-main.dark label {
  color: #e2e8f0;
}

/* Input styles */
input[type="text"],
input[type="number"],
input[type="file"],
textarea {
  width: 100%;
  padding: 12px 16px;
```

```
  border: 2px solid #e2e8f0;
  border-radius: 8px;
  font-size: 1rem;
  transition: all 0.3s ease;
  background-color: #ffffff;
  box-sizing: border-box;
}

input[type="text"]:focus,
input[type="number"]:focus,
input[type="file"]:focus,
textarea:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
  transform: translateY(-1px);
}

.dashboard-main.dark input[type="text"],
.dashboard-main.dark input[type="number"],
.dashboard-main.dark input[type="file"],
.dashboard-main.dark textarea {
  background-color: #4a5568;
  border-color: #718096;
  color: #e2e8f0;
}

.dashboard-main.dark input[type="text"]:focus,
.dashboard-main.dark input[type="number"]:focus,
.dashboard-main.dark input[type="file"]:focus,
.dashboard-main.dark textarea:focus {
  border-color: #667eea;
  background-color: #2d3748;
}

/* File input styling */
input[type="file"] {
  padding: 10px;
  background: linear-gradient(135deg, #f7fafc 0%, #edf2f7 100%);
  cursor: pointer;
  position: relative;
}

.dashboard-main.dark input[type="file"] {
  background: linear-gradient(135deg, #4a5568 0%, #2d3748 100%);
}

input[type="file"]:hover {
  background: linear-gradient(135deg, #edf2f7 0%, #e2e8f0 100%);
}

.dashboard-main.dark input[type="file"]:hover {
  background: linear-gradient(135deg, #2d3748 0%, #1a202c 100%);
}

/* Preview image styling */
img[alt="preview"] {
  border-radius: 8px;
  border: 2px solid #e2e8f0;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  transition: all 0.3s ease;
}

img[alt="preview"]:hover {
  transform: scale(1.02);
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.15);
}

/* Pricing inputs container */
.pricing-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 15px;
```

```
  margin: 10px 0;
}

/* Features section */
.features-section {
  margin: 20px 0;
}

.feature-input-group {
  display: flex;
  gap: 12px;
  margin-bottom: 12px;
  align-items: center;
}

.feature-input-group input {
  flex: 1;
  margin: 0;
}

/* Button styles */
.btn {
  padding: 12px 24px;
  border: none;
  border-radius: 8px;
  font-size: 1rem;
  font-weight: 600;
  cursor: pointer;
  transition: all 0.3s ease;
  text-transform: uppercase;
  letter-spacing: 0.5px;
  min-width: 120px;
}

.btn-primary {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  box-shadow: 0 4px 15px rgba(102, 126, 234, 0.3);
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 20px rgba(102, 126, 234, 0.4);
  background: linear-gradient(135deg, #5a6fd8 0%, #6a4190 100%);
}

.btn-secondary {
  background: linear-gradient(135deg, #718096 0%, #4a5568 100%);
  color: white;
  box-shadow: 0 4px 15px rgba(113, 128, 150, 0.3);
}

.btn-secondary:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 20px rgba(113, 128, 150, 0.4);
  background: linear-gradient(135deg, #4a5568 0%, #2d3748 100%);
}

.btn-danger {
  background: linear-gradient(135deg, #fc8181 0%, #e53e3e 100%);
  color: white;
  box-shadow: 0 4px 15px rgba(229, 62, 62, 0.3);
}

.btn-danger:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 20px rgba(229, 62, 62, 0.4);
  background: linear-gradient(135deg, #e53e3e 0%, #c53030 100%);
}

/* Add feature button special styling */
.add-feature-btn {
  background: linear-gradient(135deg, #48bb78 0%, #38a169 100%);
```

```
  color: white;
  margin: 10px 0;
  box-shadow: 0 4px 15px rgba(72, 187, 120, 0.3);
}

.add-feature-btn:hover {
  background: linear-gradient(135deg, #38a169 0%, #2f855a 100%);
  box-shadow: 0 6px 20px rgba(72, 187, 120, 0.4);
}

/* Service display cards */
.service-card {
  background: linear-gradient(135deg, #ffffff 0%, #f7fafc 100%);
  border-radius: 16px;
  padding: 25px;
  margin-bottom: 25px;
  box-shadow: 0 6px 25px rgba(0, 0, 0, 0.1);
  border: 1px solid #e2e8f0;
  transition: all 0.3s ease;
}

.dashboard-main.dark .service-card {
  background: linear-gradient(135deg, #2d3748 0%, #1a202c 100%);
  border-color: #4a5568;
}

.service-card:hover {
  transform: translateY(-3px);
  box-shadow: 0 10px 35px rgba(0, 0, 0, 0.15);
}

.service-card h3 {
  color: #2d3748;
  font-size: 1.8rem;
  font-weight: 700;
  margin-bottom: 20px;
  text-align: center;
}

.dashboard-main.dark .service-card h3 {
  color: #e2e8f0;
}

.service-card img[alt="banner"] {
  width: 100%;
  height: 200px;
  object-fit: cover;
  border-radius: 12px;
  margin-bottom: 20px;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
}

/* Service info styling */
.service-info {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 15px;
  margin: 20px 0;
}

.service-info p {
  background: #f7fafc;
  padding: 12px 16px;
  border-radius: 8px;
  margin: 0;
  font-weight: 600;
  border-left: 4px solid #667eea;
}

.dashboard-main.dark .service-info p {
  background: #4a5568;
  color: #e2e8f0;
}
```

```
/* Features list */
.service-card ul {
  list-style: none;
  padding: 0;
  margin: 20px 0;
}

.service-card li {
  background: linear-gradient(135deg, #edf2f7 0%, #e2e8f0 100%);
  padding: 12px 16px;
  margin: 8px 0;
  border-radius: 8px;
  border-left: 4px solid #48bb78;
  font-weight: 500;
  transition: all 0.3s ease;
}

.dashboard-main.dark .service-card li {
  background: linear-gradient(135deg, #4a5568 0%, #2d3748 100%);
  color: #e2e8f0;
}

.service-card li:hover {
  transform: translateX(5px);
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

/* Video iframe styling */
iframe {
  border-radius: 12px;
  margin: 20px 0;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
}

/* Responsive design */
@media (max-width: 768px) {
  .dashboard-main {
    padding: 15px;
  }

  .dashboard-main h1 {
    font-size: 2rem;
  }

  .card {
    padding: 20px;
  }

  .feature-input-group {
    flex-direction: column;
    gap: 8px;
  }

  .feature-input-group input {
    margin-bottom: 0;
  }

  .btn {
    width: 100%;
    margin: 5px 0;
  }

  .pricing-container {
    grid-template-columns: 1fr;
  }

  .service-info {
    grid-template-columns: 1fr;
  }
}

@media (max-width: 480px) {
```

```
  .dashboard-main h1 {
    font-size: 1.5rem;
  }

  .card {
    padding: 15px;
  }

  .service-card img[alt="banner"] {
    height: 150px;
  }
}

/* Loading and transition effects */
.card {
  animation: fadeInUp 0.6s ease-out;
}

@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(30px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

/* Focus indicators for accessibility */
.btn:focus,
input:focus,
textarea:focus {
  outline: 2px solid #667eea;
  outline-offset: 2px;
}

/* Smooth scrolling */
html {
  scroll-behavior: smooth;
}
```

File: src\dashboard\master_admin\Services.js

```
"use client";

import { useState, useEffect, useRef } from "react";
import { useTheme } from "../../ThemeProvider";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

// API functions - easier to mock for testing
const createApiService = (apiUrl = "http://localhost:8080/api/services") => ({
  fetchServices: async () => {
    const response = await fetch(apiUrl);
    return response.json();
  },

  createService: async (serviceData) => {
    const response = await fetch(apiUrl, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(serviceData),
    });
    return response.json();
  },

  updateService: async (id, serviceData) => {
    const response = await fetch(`${apiUrl}/${id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(serviceData),
    });
```

```
    return response.json();
  },

  deleteService: async (id) => {
    return fetch(`${apiUrl}/${id}`, { method: "DELETE" });
  },
});

// Toast utility functions - easier to test
const toastUtils = {
  showSuccess: (message) => toast.success(message),
  showError: (message) => toast.error(message),
  showInfo: (message) => toast.info(message),
  showConfirmation: (message, onConfirm, onCancel) => {
    toast.info(
      <div>
        <p style={{ color: "black" }}>{message}</p>
        <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
          <button
            className="btn btn-primary"
            onClick={() => {
              onConfirm();
              toast.dismiss();
            }}
          >
            Confirm
          </button>
          <button
            className="btn btn-primary"
            onClick={() => {
              toast.dismiss();
              if (onCancel) onCancel();
              else toastUtils.showInfo("Deletion cancelled.");
            }}
          >
            Cancel
          </button>
        </div>
      </div>,
      {
        position: "top-center",
        autoClose: false,
        closeOnClick: false,
        draggable: false,
        closeButton: false,
      }
    );
  },
};

// Form utilities - pure functions for easier testing
const formUtils = {
  getInitialFormData: () => ({
    banner: "",
    title: "",
    subtitle: "",
    priceMonthly: "",
    priceQuarterly: "",
    priceYearly: "",
    rating: "",
    description: "",
    previewVideo: "",
    features: [""],
  }),

  handleInputChange: (formData, name, value) => ({
    ...formData,
    [name]: value,
  }),

  handleFeatureChange: (formData, index, value) => {
    const updated = [...formData.features];
    updated[index] = value;
```

```
    return { ...formData, features: updated };
  },

  addFeature: (formData) => ({
    ...formData,
    features: [...formData.features, ""],
  }),

  removeFeature: (formData, index) => ({
    ...formData,
    features: formData.features.filter((_, i) => i !== index),
  }),

  handleBannerUpload: (formData, file) => ({
    ...formData,
    banner: URL.createObjectURL(file),
  }),
};

// Custom hook for services data - easier to test
const useServices = (apiService) => {
  const [services, setServices] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  const fetchServices = async () => {
    try {
      setLoading(true);
      const data = await apiService.fetchServices();
      setServices(data);
      setError(null);
    } catch (err) {
      setError("Error fetching services");
      console.error("Error fetching services:", err);
    } finally {
      setLoading(false);
    }
  };

  const addService = async (serviceData) => {
    const newService = await apiService.createService(serviceData);
    setServices((prev) => [...prev, newService]);
    return newService;
  };

  const editService = async (id, serviceData) => {
    const updatedService = await apiService.updateService(id, serviceData);
    setServices((prev) => prev.map((s) => (s.id === id ? updatedService : s)));
    return updatedService;
  };

  const removeService = async (id) => {
    await apiService.deleteService(id);
    setServices((prev) => prev.filter((s) => s.id !== id));
  };

  useEffect(() => {
    fetchServices();
  }, []);

  return {
    services,
    loading,
    error,
    addService,
    editService,
    removeService,
    refetch: fetchServices,
  };
};

// Custom hook for form state - easier to test
const useServiceForm = () => {
```

```
  const [formData, setFormData] = useState(formUtils.getInitialFormData());
  const [editingServiceId, setEditingServiceId] = useState(null);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prev) => formUtils.handleInputChange(prev, name, value));
  };

  const handleFeatureChange = (index, value) => {
    setFormData((prev) => formUtils.handleFeatureChange(prev, index, value));
  };

  const addFeature = () => {
    setFormData((prev) => formUtils.addFeature(prev));
  };

  const removeFeature = (index) => {
    setFormData((prev) => formUtils.removeFeature(prev, index));
  };

  const handleBannerUpload = (file) => {
    setFormData((prev) => formUtils.handleBannerUpload(prev, file));
  };

  const resetForm = () => {
    setFormData(formUtils.getInitialFormData());
    setEditingServiceId(null);
  };

  const setEditingService = (service) => {
    setFormData(service);
    setEditingServiceId(service.id);
  };

  return {
    formData,
    editingServiceId,
    handleChange,
    handleFeatureChange,
    addFeature,
    removeFeature,
    handleBannerUpload,
    resetForm,
    setEditingService,
  };
};

// ServiceForm component - easier to test in isolation
const ServiceForm = ({
  formData,
  editingServiceId,
  onSubmit,
  onChange,
  onFeatureChange,
  onAddFeature,
  onRemoveFeature,
  onBannerUpload,
}) => (
  <form onSubmit={onSubmit} className="card" style={{ marginTop: "20px" }}>
    <label htmlFor="banner">Service Banner</label>
    <input
      id="banner"
      type="file"
      accept="image/*"
      onChange={onBannerUpload}
      required
    />
    {formData.banner && (
      <img
        src={formData.banner || "/placeholder.svg"}
        alt="preview"
        style={{ maxHeight: "150px", marginTop: "10px" }}
      />
```

```
)}

<label htmlFor="title">Title</label>
<input
  id="title"
  name="title"
  value={formData.title}
  onChange={onChange}
  required
/>

<label htmlFor="subtitle">Sub-Title</label>
<input
  id="subtitle"
  name="subtitle"
  value={formData.subtitle}
  onChange={onChange}
  required
/>

<label htmlFor="priceMonthly">Pricing Options</label>
<div className="pricing-row">
  <input
    id="priceMonthly"
    name="priceMonthly"
    type="number"
    placeholder="1 Month"
    value={formData.priceMonthly}
    onChange={onChange}
    required
  />
  <input
    id="priceQuarterly"
    name="priceQuarterly"
    type="number"
    placeholder="3 Month"
    value={formData.priceQuarterly}
    onChange={onChange}
    required
  />
  <input
    id="priceYearly"
    name="priceYearly"
    type="number"
    placeholder="6 Month"
    value={formData.priceYearly}
    onChange={onChange}
    required
  />
</div>

<label htmlFor="rating">Rating</label>
<input
  id="rating"
  name="rating"
  type="number"
  step="0.1"
  max="5"
  value={formData.rating}
  onChange={onChange}
  required
/>

<label htmlFor="description">Description</label>
<textarea
  id="description"
  name="description"
  value={formData.description}
  onChange={onChange}
  rows={4}
  required
/>
```

```
    <label htmlFor="previewVideo">Preview Video (YouTube)</label>
    <input
      id="previewVideo"
      name="previewVideo"
      value={formData.previewVideo}
      onChange={onChange}
      required
    />

    <label>Program Features</label>
    {formData.features.map((f, i) => (
      <div key={i} className="d-flex">
        <input
          id={`feature-${i}`}
          value={f}
          onChange={(e) => onFeatureChange(i, e.target.value)}
          placeholder={`Feature ${i + 1}`}
          required
        />
        {formData.features.length > 1 && (
          <button
            type="button"
            className="btn btn-primary"
            onClick={() => onRemoveFeature(i)}
          >
            Remove
          </button>
        )}
      </div>
    ))}
    <button type="button" className="btn btn-primary" onClick={onAddFeature}>
      Add Feature
    </button>

    <div className="center-btn">
      <button type="submit" className="btn btn-primary">
        {editingServiceId ? "Update Service" : "Submit Service"}
      </button>
    </div>
  </form>
);

// ServiceCard component - easier to test in isolation
const ServiceCard = ({ service, onView, onEdit, onDelete }) => (
  <div className="card blog-card">
    <img
      src={service.banner || "/placeholder.svg"}
      alt="banner"
      style={{ height: "100px", objectFit: "cover", width: "100%" }}
    />
    <h3>{service.title}</h3>
    <p>
      <strong>{service.description}</strong>
    </p>
    <p>
      <em>{service.subtitle}</em>
    </p>
    <div className="center-btn">
      <button className="btn btn-primary" onClick={() => onView(service)}>
        View
      </button>
      <button className="btn btn-primary" onClick={() => onEdit(service)}>
        Edit
      </button>
      <button className="btn btn-primary" onClick={() => onDelete(service.id)}>
        Delete
      </button>
    </div>
  </div>
);

// ServiceModal component - easier to test in isolation
const ServiceModal = ({ service, onClose }) => {
```

```
if (!service) return null;

return (
  <div
    className="modal-overlay"
    style={{ border: "1px solid #cc5500" }}
    onClick={onClose}
  >
    <div
      className="modals-box"
      style={{ border: "1px solid #cc5500" }}
      onClick={(e) => e.stopPropagation()}
    >
      <h2>{service.title}</h2>
      <img
        src={service.banner || "/placeholder.svg"}
        alt="service-banner"
        style={{
          width: "100%",
          maxHeight: "200px",
          objectFit: "cover",
          borderRadius: "10px",
          marginBottom: "20px",
        }}
      />
      <table className="popup-table">
        <tbody>
          <tr>
            <td>
              <strong>Sub-Title:</strong>
            </td>
            <td>{service.subtitle}</td>
          </tr>
          <tr>
            <td>
              <strong>1 Month:</strong>
            </td>
            <td>?{service.priceMonthly}</td>
          </tr>
          <tr>
            <td>
              <strong>3 Month:</strong>
            </td>
            <td>?{service.priceQuarterly}</td>
          </tr>
          <tr>
            <td>
              <strong>6 Month:</strong>
            </td>
            <td>?{service.priceYearly}</td>
          </tr>
          <tr>
            <td>
              <strong>Rating:</strong>
            </td>
            <td>{service.rating}</td>
          </tr>
          <tr>
            <td>
              <strong>Description:</strong>
            </td>
            <td>{service.description}</td>
          </tr>
        </tbody>
      </table>
      <iframe
        width="100%"
        height="200"
        src={service.previewVideo.replace("watch?v=", "embed/")}
        title="Preview Video"
        frameBorder="0"
        allowFullScreen
        style={{ margin: "20px 0" }}
```

```
        />
        <strong>Program Features:</strong>
        <ul style={{ paddingLeft: "10px" }}>
          {service.features.map((f, i) => (
            <li key={i}>{f}</li>
          ))}
        </ul>
        <div className="center-btn" style={{ marginTop: "20px" }}>
          <center>
            <button className="btn btn-primary" onClick={onClose}>
              Close
            </button>
          </center>
        </div>
      </div>
    </div>
  );
};

// Main Services component
const Services = ({ apiUrl } = {}) => {
  const { theme } = useTheme();
  const [showForm, setShowForm] = useState(false);
  const [selectedService, setSelectedService] = useState(null);
  const formRef = useRef(null);

  // Initialize API service
  const apiService = createApiService(apiUrl);

  // Use custom hooks
  const { services, addService, editService, removeService } =
    useServices(apiService);
  const {
    formData,
    editingServiceId,
    handleChange,
    handleFeatureChange,
    addFeature,
    removeFeature,
    handleBannerUpload,
    resetForm,
    setEditingService,
  } = useServiceForm();

  // Event handlers - now pure functions that are easier to test
  const handleBannerUploadWrapper = (e) => {
    const file = e.target.files[0];
    if (file) {
      handleBannerUpload(file);
    }
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      if (editingServiceId) {
        await editService(editingServiceId, formData);
        toastUtils.showSuccess("Service updated successfully!");
      } else {
        await addService(formData);
        toastUtils.showSuccess("Service added successfully!");
      }
      resetForm();
      setShowForm(false);
    } catch (error) {
      toastUtils.showError("Error saving service.");
    }
  };

  const handleDelete = (id) => {
    toastUtils.showConfirmation(
      "Are you sure you want to delete this service?",
      async () => {
```

```
      try {
        await removeService(id);
        toastUtils.showError("Service deleted.");
      } catch (error) {
        toastUtils.showError("Error deleting service.");
      }
    }
  );
};

const handleEdit = (service) => {
  setEditingService(service);
  setShowForm(true);
  setTimeout(() => {
    formRef.current?.scrollIntoView({ behavior: "smooth" });
  }, 100);
};

const handleToggleForm = () => {
  setShowForm(!showForm);
  if (!showForm) resetForm();
};

return (
  <div className={`dashboard-main ${theme}`}>
    <ToastContainer position="top-center" autoClose={3000} />
    <h1>Manage Services</h1>

    <button className="btn btn-primary" onClick={handleToggleForm}>
      {showForm ? "Close Form" : "Add Service"}
    </button>

    <div
      ref={formRef}
      style={{
        maxHeight: showForm ? "2000px" : "0",
        overflow: "hidden",
        transition: "max-height 0.5s ease-in-out",
      }}
    >
      {showForm && (
        <ServiceForm
          formData={formData}
          editingServiceId={editingServiceId}
          onSubmit={handleSubmit}
          onChange={handleChange}
          onFeatureChange={handleFeatureChange}
          onAddFeature={addFeature}
          onRemoveFeature={removeFeature}
          onBannerUpload={handleBannerUploadWrapper}
        />
      )}
    </div>

    <h2 style={{ marginTop: "30px" }}>Added Services</h2>
    {services.length === 0 ? (
      <p>No services added</p>
    ) : (
      <div className="blog-grid">
        {services.map((service) => (
          <ServiceCard
            key={service.id}
            service={service}
            onView={setSelectedService}
            onEdit={handleEdit}
            onDelete={handleDelete}
          />
        ))}
      </div>
    )}

    <ServiceModal
      service={selectedService}
```

```
        onClose={() => setSelectedService(null)}
      />
    </div>
  );
};

export default Services;

// Export utilities for testing
export {
  createApiService,
  toastUtils,
  formUtils,
  useServices,
  useServiceForm,
  ServiceForm,
  ServiceCard,
  ServiceModal,
};
```

File: src\dashboard\master_admin\SystemOverview.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { Chart } from "react-google-charts";
import { useTheme } from "../../ThemeProvider";
import "./master_admin.css";

const SystemOverview = () => {
  const { theme } = useTheme();
  const [stats, setStats] = useState({
    collected: 0,
    processed: 0,
    pendingAssignments: 0,
    completedAssignments: 0,
    scheduledSessions: 0,
    completedSessions: 0,
    inProgress: 0,
    resolved: 0,
  });

  const [samples, setSamples] = useState([]);
  const [assignments, setAssignments] = useState([]);
  const [sessions, setSessions] = useState([]);
  const [resolutions, setResolutions] = useState([]);
  const [physioCharts, setPhysioCharts] = useState([]);
  const [dietCharts, setDietCharts] = useState([]);
  const [labTracking, setLabTracking] = useState([]);

  const fetchStats = async () => {
    try {
      const [
        samplesRes,
        assignmentsRes,
        sessionsRes,
        resolutionsRes,
        physioRes,
        dietRes,
        labRes,
      ] = await Promise.all([
        axios.get("http://localhost:3001/samples"),
        axios.get("http://localhost:3001/assignments"),
        axios.get("http://localhost:3001/sessions"),
        axios.get("http://localhost:3001/resolutions"),
        axios.get("http://localhost:3001/physioCharts"),
        axios.get("http://localhost:3001/dietCharts"),
        axios.get("http://localhost:3001/labTracking"),
      ]);

      const collected = samplesRes.data.filter(
        (s) => s.status === "collected"
      ).length;
      const processed = samplesRes.data.filter(
        (s) => s.status === "processed"
```

```
    ).length;

    const pendingAssignments = assignmentsRes.data.filter(
      (a) => a.status === "pending"
    ).length;
    const completedAssignments = assignmentsRes.data.filter(
      (a) => a.status === "completed"
    ).length;

    const scheduledSessions = sessionsRes.data.filter(
      (s) => s.status === "scheduled"
    ).length;
    const completedSessions = sessionsRes.data.filter(
      (s) => s.status === "completed"
    ).length;

    const inProgress = resolutionsRes.data.filter(
      (r) => r.status === "in progress"
    ).length;
    const resolved = resolutionsRes.data.filter(
      (r) => r.status === "resolved"
    ).length;

    setStats({
      collected,
      processed,
      pendingAssignments,
      completedAssignments,
      scheduledSessions,
      completedSessions,
      inProgress,
      resolved,
    });

    setSamples(samplesRes.data);
    setAssignments(assignmentsRes.data);
    setSessions(sessionsRes.data);
    setResolutions(resolutionsRes.data);
    setPhysioCharts(physioRes.data);
    setDietCharts(dietRes.data);
    setLabTracking(labRes.data);
  } catch (error) {
    console.error("Error fetching stats:", error);
  }
};

useEffect(() => {
  fetchStats();
}, []);

return (
  <div className={`dashboard-main ${theme}`}>
    <h1>System Overview</h1>

    {/* Summary Cards */}
    <div className="overview-grid">
      <div className="card">
        <h3>Samples</h3>
        <p>Collected: {stats.collected}</p>
        <p>Processed: {stats.processed}</p>
        <div className="mini-table">
          <table>
            <thead>
              <tr>
                <th>MRN</th>
                <th>Status</th>
                <th>Phlebotomist</th>
                <th>Date</th>
              </tr>
            </thead>
            <tbody>
              {samples.map((s) => (
                <tr key={s.id}>
```

```
            <td>{s.mrn}</td>
            <td>{s.status}</td>
            <td>{s.phlebotomist || "-"}</td>
            <td>{s.date || "-"}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

<div className="card">
  <h3>Doctor Assignments</h3>
  <p>Pending: {stats.pendingAssignments}</p>
  <p>Completed: {stats.completedAssignments}</p>
  <div className="mini-table">
    <table>
      <thead>
        <tr>
          <th>MRN</th>
          <th>Status</th>
          <th>Doctor</th>
          <th>Date</th>
        </tr>
      </thead>
      <tbody>
        {assignments.map((a) => (
          <tr key={a.id}>
            <td>{a.mrn}</td>
            <td>{a.status}</td>
            <td>{a.doctor || "-"}</td>
            <td>{a.date || "-"}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

<div className="card">
  <h3>Sessions</h3>
  <p>Scheduled: {stats.scheduledSessions}</p>
  <p>Completed: {stats.completedSessions}</p>
  <div className="mini-table">
    <table>
      <thead>
        <tr>
          <th>MRN</th>
          <th>Type</th>
          <th>Status</th>
          <th>Date</th>
        </tr>
      </thead>
      <tbody>
        {sessions.map((s) => (
          <tr key={s.id}>
            <td>{s.mrn}</td>
            <td>{s.type}</td>
            <td>{s.status}</td>
            <td>{s.date || "-"}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

<div className="card">
  <h3>Resolutions</h3>
  <p>In Progress: {stats.inProgress}</p>
  <p>Resolved: {stats.resolved}</p>
  <div className="mini-table">
    <table>
```

```
      <thead>
        <tr>
          <th>MRN</th>
          <th>Status</th>
          <th>Resolved By</th>
          <th>Date</th>
        </tr>
      </thead>
      <tbody>
        {resolutions.map((r) => (
          <tr key={r.id}>
            <td>{r.mrn}</td>
            <td>{r.status}</td>
            <td>{r.by || "-"}</td>
            <td>{r.date || "-"}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>

<div className="card">
  <h3>Physio Charts</h3>
  <div className="mini-table">
    <table>
      <thead>
        <tr>
          <th>MRN</th>
          <th>Current Chart</th>
          <th>Past Charts</th>
          <th>Handled By</th>
          <th>Submitted By</th>
        </tr>
      </thead>
      <tbody>
        {physioCharts.map((entry) => {
          const latest = entry.charts[entry.charts.length - 1];
          return (
            <tr key={entry.id}>
              <td>{entry.mrn}</td>
              <td>
                {latest.chartNumber} - {latest.chartName}
              </td>
              <td>
                <details>
                  <summary
                    style={{
                      cursor: "pointer",
                      color: "var(--text-primary)",
                    }}
                  >
                    View Chart History
                  </summary>
                  <ul style={{ paddingLeft: "1rem" }}>
                    {entry.charts
                      .slice(0, -1)
                      .reverse()
                      .map((c, idx) => (
                        <li key={idx}>
                          {c.chartNumber} - {c.chartName}
                        </li>
                      ))}
                  </ul>
                </details>
              </td>
              <td>{entry.handlerName || "-"}</td>
              <td>{entry.submittedByDoctor || "-"}</td>
            </tr>
          );
        })}
      </tbody>
    </table>
```

```
  </div>
</div>

<div className="card">
  <h3>Dietitian Charts</h3>
  <div className="mini-table">
    <table>
      <thead>
        <tr>
          <th>MRN</th>
          <th>Current Chart</th>
          <th>Past Charts</th>
          <th>Handled By</th>
          <th>Submitted By</th>
        </tr>
      </thead>
      <tbody>
        {dietCharts.map((entry) => {
          const chartList = entry.charts || [];
          const latest = chartList[chartList.length - 1];

          return (
            <tr key={entry.id}>
              <td>{entry.mrn}</td>
              <td>
                {latest
                  ? `${latest.chartNumber} - ${latest.chartName}`
                  : "-"}
              </td>
              <td>
                <details>
                  <summary
                    style={{
                      cursor: "pointer",
                      color: "var(--text-primary)",
                    }}
                  >
                    View Chart History
                  </summary>
                  <ul style={{ paddingLeft: "1rem" }}>
                    {entry.charts
                      .slice(0, -1) // exclude the current one
                      .reverse()
                      .map((c, idx) => (
                        <li key={idx}>
                          {c.chartNumber} - {c.chartName} (
                          {c.dateAssigned})
                        </li>
                      ))}
                  </ul>
                </details>
              </td>

              <td>{entry.handlerName || "-"}</td>
              <td>{entry.submittedByDoctor || "-"}</td>
            </tr>
          );
        })}
      </tbody>
    </table>
  </div>
</div>

<div className="card">
  <h3>Lab Tracking</h3>
  <div className="mini-table">
    <table>
      <thead>
        <tr>
          <th>MRN</th>
          <th>Submitted By</th>
          <th>Submitted Date</th>
          <th>Handled By</th>
```

```
            <th>Report Date</th>
            <th>Report</th>
          </tr>
        </thead>
        <tbody>
          {labTracking.map((l) => (
            <tr key={l.id}>
              <td>{l.mrn}</td>
              <td>{l.submittedBy}</td>
              <td>{l.submittedDate}</td>
              <td>{l.handledBy}</td>
              <td>{l.reportGivenDate}</td>
              <td>
                {l.reportFile ? (
                  <a
                    href={l.reportFile}
                    target="_blank"
                    rel="noopener noreferrer"
                  >
                    View Report
                  </a>
                ) : (
                  "-"
                )}
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  </div>
</div>

{/* Charts */}
<div className="charts-section">
  <div className="card">
    <h3>Resolution Status</h3>
    <Chart
      chartType="ColumnChart"
      data={[
        ["Status", "Count", { role: "style" }],
        ["In Progress", stats.inProgress, "#f1c40f"],
        ["Resolved", stats.resolved, "#3498db"],
      ]}
      options={{
        backgroundColor: "transparent",
        legend: "none",
        titleTextStyle: { color: "var(--text-primary)", fontSize: 18 },
        vAxis: { textStyle: { color: "var(--text-primary)" } },
        hAxis: { textStyle: { color: "var(--text-primary)" } },
      }}
      width="100%"
      height="300px"
    />
  </div>

  <div className="card">
    <h3>Doctor Assignments</h3>
    <Chart
      chartType="BarChart"
      data={[
        ["Status", "Count", { role: "style" }],
        ["Pending", stats.pendingAssignments, "#e67e22"],
        ["Completed", stats.completedAssignments, "#2ecc71"],
      ]}
      options={{
        backgroundColor: "transparent",
        legend: "none",
        titleTextStyle: { color: "var(--text-primary)", fontSize: 18 },
        hAxis: { textStyle: { color: "var(--text-primary)" } },
        vAxis: { textStyle: { color: "var(--text-primary)" } },
      }}
      width="100%"
```

```
            height="300px"
          />
        </div>
      </div>
    </div>
  );
};

export default SystemOverview;
```

File: src\dashboard\master_admin\TemplateDemo.js

```
import React, { useState } from "react";

const fieldTypes = ["text", "textarea", "dropdown", "date"];

const TemplateDemo = () => {
  const [templateFields, setTemplateFields] = useState([]);
  const [currentField, setCurrentField] = useState({ label: "", type: "text", options: "" });
  const [savedTemplate, setSavedTemplate] = useState(null);
  const [formData, setFormData] = useState({});

  // ? Handle field change for admin
  const handleFieldChange = (e) => {
    setCurrentField({ ...currentField, [e.target.name]: e.target.value });
  };

  // ? Add field to template
  const addField = () => {
    if (!currentField.label) return;
    setTemplateFields([...templateFields, currentField]);
    setCurrentField({ label: "", type: "text", options: "" });
  };

  // ? Clear the template
  const clearTemplate = () => {
    setTemplateFields([]);
    setSavedTemplate(null);
  };

  // ? Save the template
  const saveTemplate = () => {
    setSavedTemplate(templateFields);
  };

  // ??? Handle form input in data entry form
  const handleInputChange = (label, value) => {
    setFormData({ ...formData, [label]: value });
  };

  return (
    <div style={{ padding: "2rem", fontFamily: "sans-serif" }}>
      <h2>? Admin Template Builder</h2>
      <div style={{ display: "flex", gap: "1rem", marginBottom: "1rem" }}>
        <input
          type="text"
          name="label"
          placeholder="Field Label"
          value={currentField.label}
          onChange={handleFieldChange}
        />
        <select name="type" value={currentField.type} onChange={handleFieldChange}>
          {fieldTypes.map((type) => (
            <option key={type} value={type}>{type}</option>
          ))}
        </select>
        {currentField.type === "dropdown" && (
          <input
            type="text"
            name="options"
            placeholder="Comma separated options"
            value={currentField.options}
            onChange={handleFieldChange}
          />
```

```
  )}
  <button onClick={addField}>Add Field</button>
</div>

<div>
  <h4>? Current Fields:</h4>
  <ul>
    {templateFields.map((f, i) => (
      <li key={i}>
        <strong>{f.label}</strong> ({f.type}) {f.type === "dropdown" && `? [${f.options}]`}
      </li>
    ))}
  </ul>
  <button onClick={saveTemplate}>? Save Template</button>
  <button onClick={clearTemplate} style={{ marginLeft: "1rem" }}>? Clear</button>
</div>

<hr style={{ margin: "2rem 0" }} />

<h2>??? Data Entry Form (Auto-Generated)</h2>
{savedTemplate ? (
  <form style={{ display: "flex", flexDirection: "column", gap: "1rem" }}>
    {savedTemplate.map((field, i) => {
      const value = formData[field.label] || "";
      switch (field.type) {
        case "text":
          return (
            <input
              key={i}
              type="text"
              placeholder={field.label}
              value={value}
              onChange={(e) => handleInputChange(field.label, e.target.value)}
            />
          );
        case "textarea":
          return (
            <textarea
              key={i}
              placeholder={field.label}
              value={value}
              onChange={(e) => handleInputChange(field.label, e.target.value)}
            />
          );
        case "date":
          return (
            <input
              key={i}
              type="date"
              value={value}
              onChange={(e) => handleInputChange(field.label, e.target.value)}
            />
          );
        case "dropdown":
          return (
            <select
              key={i}
              value={value}
              onChange={(e) => handleInputChange(field.label, e.target.value)}
            >
              <option value="">Select {field.label}</option>
              {field.options.split(",").map((opt, j) => (
                <option key={j} value={opt.trim()}>{opt.trim()}</option>
              ))}
            </select>
          );
        default:
          return null;
      }
    })}
    <button type="button" onClick={() => alert(JSON.stringify(formData, null, 2))}>
      ? Submit Form
    </button>
```

```
        </form>
      ) : (
        <p><i>No template saved yet. Create and save a template above.</i></p>
      )}
    </div>
  );
};

export default TemplateDemo;
```

File: src\dashboard\master_admin\UserManagement.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useTheme } from "../../ThemeProvider";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./master_admin.css";

const roles = ["All", "doctor", "physio", "dietitian", "lab", "phlebotomist"];

const UserManagement = () => {
  const { theme } = useTheme();
  const [users, setUsers] = useState([]);
  const [search, setSearch] = useState("");
  const [activeRole, setActiveRole] = useState("All");
  const [isMobile, setIsMobile] = useState(window.innerWidth <= 768);

  const [showModal, setShowModal] = useState(false);
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
    role: "doctor",
  });
  const [editingUser, setEditingUser] = useState(null);
  const API_URL = "http://localhost:8080/api/users";

  useEffect(() => {
    fetchUsers();

    const handleResize = () => {
      setIsMobile(window.innerWidth <= 768);
    };

    window.addEventListener('resize', handleResize);
    return () => window.removeEventListener('resize', handleResize);
  }, []);

  const fetchUsers = async () => {
    try {
      const response = await axios.get(API_URL);
      setUsers(response.data);
    } catch (error) {
      console.error("Error fetching users:", error);
    }
  };

  const openModal = (user = null) => {
    setEditingUser(user);
    setFormData(
      user
        ? { ...user, password: "" }
        : { name: "", email: "", password: "", role: "doctor" }
    );
    setShowModal(true);
  };

  const closeModal = () => {
    setShowModal(false);
    setEditingUser(null);
  };

  const handleInputChange = (e) => {
```

```
  setFormData({ ...formData, [e.target.name]: e.target.value });
};

const handleSave = async () => {
  if (!formData.name || !formData.email || !formData.role) {
    alert("Please fill in all fields");
    return;
  }

  try {
    if (editingUser) {
      await axios.put(`${API_URL}/${editingUser.id}`, formData);
      toast.success("User updated successfully");
    } else {
      await axios.post(API_URL, formData);
      toast.success("User added successfully");
    }
    closeModal();
    fetchUsers();
  } catch (error) {
    console.error("Error saving user:", error);
    toast.error("Error saving user");
  }
};

const handleDelete = (id) => {
  const confirmToast = ({ closeToast }) => (
    <div>
      <p style={{ color: "black" }}>
        Are you sure you want to delete this user?
      </p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={async () => {
            try {
              await axios.delete(`${API_URL}/${id}`);
              setUsers((prev) => prev.filter((u) => u.id !== id));
              toast.dismiss();
              toast.error("User deleted.");
            } catch (error) {
              console.error("Error deleting user:", error);
              toast.dismiss();
              toast.error("Failed to delete user.");
            }
          }}
        >
          Confirm
        </button>
        <button
          className="btn btn-primary"
          onClick={() => {
            toast.dismiss();
            toast.info("Deletion cancelled.");
          }}
        >
          Cancel
        </button>
      </div>
    </div>
  );

  toast(confirmToast, {
    position: "top-center",
    autoClose: false,
    closeOnClick: false,
    draggable: false,
    closeButton: false,
  });
};

const filteredUsers = users.filter((user) => {
  const matchSearch =
```

```
    user.name?.toLowerCase().includes(search.toLowerCase()) ||
    user.email?.toLowerCase().includes(search.toLowerCase());

  const matchRole =
    activeRole === "All" ||
    (typeof user.role === "string" &&
      user.role.toLowerCase() === activeRole.toLowerCase());

  return matchSearch && matchRole;
});

return (
  <div className={`dashboard-main ${theme}`}>
    <style jsx>{`
      .dashboard-main {
        padding: 1rem;
        min-height: 100vh;
      }

      .dashboard-main h1 {
        margin-bottom: 1.5rem;
        font-size: clamp(1.5rem, 4vw, 2rem);
      }

      .user-actions {
        display: flex;
        flex-direction: column;
        gap: 1rem;
        margin-bottom: 1.5rem;
      }

      @media (min-width: 768px) {
        .user-actions {
          flex-direction: row;
          align-items: center;
          justify-content: space-between;
          flex-wrap: wrap;
        }
      }

      .search-input {
        flex: 1;
        min-width: 200px;
        padding: 0.5rem;
        border-radius: 4px;
        border: 1px solid #ddd;
        font-size: 1rem;
      }

      .role-filters {
        display: flex;
        flex-wrap: wrap;
        gap: 0.5rem;
        justify-content: center;
      }

      @media (min-width: 768px) {
        .role-filters {
          justify-content: flex-start;
        }
      }

      .filter-btn {
        padding: 0.5rem 1rem;
        border: 1px solid #ddd;
        background: transparent;
        border-radius: 4px;
        cursor: pointer;
        transition: all 0.2s;
        font-size: 0.9rem;
      }

      .filter-btn.active {
```

```
  background: #007bff;
  color: white;
  border-color: #007bff;
}

.btn {
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 0.9rem;
  transition: all 0.2s;
  white-space: nowrap;
}

.btn-primary {
  background: #007bff;
  color: white;
}

.btn-primary:hover {
  background: #0056b3;
}

.table-responsive {
  overflow-x: auto;
  -webkit-overflow-scrolling: touch;
  margin-bottom: 1rem;
}

.user-table {
  width: 100%;
  border-collapse: collapse;
  min-width: 600px;
}

.user-table th,
.user-table td {
  padding: 0.75rem;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

.user-table th {
  background: #f8f9fa;
  font-weight: 600;
  position: sticky;
  top: 0;
  z-index: 1;
}

.user-table td {
  vertical-align: middle;
}

.user-table td:last-child {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
}

@media (max-width: 767px) {
  .user-table td:last-child {
    flex-direction: column;
  }
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
```

```
  background: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
  padding: 1rem;
}

.modal-box {
  background: white;
  border-radius: 8px;
  width: 100%;
  max-width: 500px;
  max-height: 90vh;
  overflow-y: auto;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem;
  border-bottom: 1px solid #ddd;
}

.modal-header h3 {
  margin: 0;
  font-size: 1.25rem;
}

.close-modal {
  background: none;
  border: none;
  font-size: 1.5rem;
  cursor: pointer;
  padding: 0;
  width: 30px;
  height: 30px;
  display: flex;
  align-items: center;
  justify-content: center;
}

.modal-body {
  padding: 1rem;
}

.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
}

.form-group input,
.form-group select {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 1rem;
  box-sizing: border-box;
}

.modal-actions {
  display: flex;
  justify-content: flex-end;
  gap: 0.5rem;
  padding: 1rem;
```

```
    border-top: 1px solid #ddd;
  }

  @media (max-width: 480px) {
    .modal-actions {
      flex-direction: column-reverse;
    }

    .modal-actions button {
      width: 100%;
    }
  }

  /* Dark theme support */
  .dark .search-input,
  .dark .filter-btn,
  .dark .form-group input,
  .dark .form-group select {
    background: #333;
    color: white;
    border-color: #555;
  }

  .dark .filter-btn.active {
    background: #0056b3;
    border-color: #0056b3;
  }

  .dark .user-table th {
    background: #333;
    color: white;
  }

  .dark .modal-box {
    background: #222;
    color: white;
  }

  .dark .modal-header,
  .dark .modal-actions {
    border-color: #444;
  }
`}</style>

<ToastContainer position="top-center" autoClose={3000} />
<h1>User Management</h1>

<div className="user-actions">
  <input
    type="text"
    className="search-input"
    placeholder="Search by name or email"
    value={search}
    onChange={(e) => setSearch(e.target.value)}
  />
  <div className="role-filters">
    {roles.map((role) => (
      <button
        key={role}
        className={`filter-btn ${activeRole === role ? "active" : ""}`}
        onClick={() => setActiveRole(role)}
      >
        {role.charAt(0).toUpperCase() + role.slice(1)}
      </button>
    ))}
  </div>
  <button className="btn btn-primary" onClick={() => openModal()}>
    Add New User
  </button>
</div>

<div className="table-responsive">
  <table className="user-table">
```

```
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Role</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {filteredUsers.map((user) => (
        <tr key={user.id}>
          <td>{user.name}</td>
          <td>{user.email}</td>
          <td>{user.role}</td>
          <td>
            <button
              className="btn btn-primary"
              onClick={() => openModal(user)}
            >
              Edit
            </button>
            <button
              className="btn btn-primary"
              onClick={() => handleDelete(user.id)}
            >
              Delete
            </button>
          </td>
        </tr>
      ))}
      {filteredUsers.length === 0 && (
        <tr>
          <td colSpan="4" style={{ textAlign: "center", padding: "1rem" }}>
            No users found.
          </td>
        </tr>
      )}
    </tbody>
  </table>
</div>

{showModal && (
  <div className="modal-overlay" onClick={closeModal}>
    <div className="modal-box" onClick={(e) => e.stopPropagation()}>
      <div className="modal-header">
        <h3>{editingUser ? "Edit User" : "Add New User"}</h3>
        <button className="close-modal" onClick={closeModal}>
          ×
        </button>
      </div>
      <div className="modal-body">
        <div className="form-group">
          <label>Name</label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleInputChange}
            placeholder="Enter name"
          />
        </div>
        <div className="form-group">
          <label>Email</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleInputChange}
            placeholder="Enter email"
          />
        </div>
        <div className="form-group">
          <label>Password</label>
```

```
                <input
                  type="password"
                  name="password"
                  value={formData.password}
                  onChange={handleInputChange}
                  placeholder="Enter password"
                />
              </div>
              <div className="form-group">
                <label>Role</label>
                <select
                  name="role"
                  value={formData.role}
                  onChange={handleInputChange}
                >
                  <option value="counselor">Counselor</option>
                  <option value="physio">Physio</option>
                  <option value="dietitian">Dietitian</option>
                  <option value="lab">Lab</option>
                  <option value="doctor">Doctor</option>
                  <option value="phlebotomist">Phlebotomist</option>
                </select>
              </div>
            </div>
            <div className="modal-actions">
              <button className="btn btn-primary" onClick={closeModal}>
                Cancel
              </button>
              <button className="btn btn-primary" onClick={handleSave}>
                Save
              </button>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default UserManagement;
```

File: src\dashboard\master_admin\components\master_admin_footer.js

```
import React from 'react';
import '../../physio/components/Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} Admin Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\master_admin\components\master_admin_navbar.js

```
import React from "react";
import "../../physio/components/Navbar.css";
import { Moon, Sun } from "lucide-react";
import { useTheme } from "../../../ThemeProvider";

const DoctorNavbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="/masteradmin"> <div className="navbar-logo">Admin</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
```

```
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default DoctorNavbar;
```

File: src\dashboard\master_admin\components\master_admin_sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "../../physio/components/Sidebar.css";
import {
  Home,
  Users,
  Dumbbell,
  BarChart2,
  LogOut,
  Menu,
  Workflow,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
  Hospital,
  LayoutDashboard,
  BookText,
  Lock,
  Calendar,
  Newspaper,
  Key,
  KeyRound,
} from "lucide-react";

const MasterAdminSidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState("");
  const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
  const navigate = useNavigate();

  const toggleSidebar = () => setIsOpen(!isOpen);
  const toggleCollapse = () => setCollapsed(!collapsed);
  const handleResize = () => setIsOpen(window.innerWidth > 768);
  const closeSidebar = () => {
    if (isOpen && window.innerWidth <= 768) setIsOpen(false);
  };

  const handleMouseEnter = () => {
    if (collapsed && window.innerWidth > 768) setIsHovering(true);
  };
  const handleMouseLeave = () => {
    if (collapsed && window.innerWidth > 768) setIsHovering(false);
  };

  useEffect(() => {
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  useEffect(() => {
    document.body.classList.toggle(
      "sidebar-collapsed",
      collapsed && !isHovering
    );
  }, [collapsed, isHovering]);

  const navLinkClass = ({ isActive }) => (isActive ? "active" : "");
```

```
const handleLogoutClick = (e) => {
  e.preventDefault();
  setShowLogoutConfirm(true);
};

const confirmLogout = () => {
  localStorage.clear();
  setShowLogoutConfirm(false);
  setToastMessage("Logged out successfully.");
  setShowToast(true);
  setTimeout(() => {
    setShowToast(false);
    navigate("/login");
  }, 3000);
};

const cancelLogout = () => setShowLogoutConfirm(false);

return (
  <>
    <button className="sidebar-toggle" onClick={toggleSidebar}>
      <Menu size={24} />
    </button>

    {isOpen && window.innerWidth <= 768 && (
      <div className="sidebar-overlay show" onClick={closeSidebar}></div>
    )}

    {showLogoutConfirm && (
      <div className="modal-overlay">
        <div className="modal-container">
          <div className="modal-header">
            <center><h3>Logout Confirmation</h3></center>
            <button className="modal-close" onClick={cancelLogout}>
              <X size={18} />
            </button>
          </div>
          <div className="modal-body">
            <p>Are you sure you want to log out?</p>
          </div>
          <div className="modal-footer">
            <button className="btn btn-primary" onClick={confirmLogout}>
              Yes, Logout
            </button>
          </div>
        </div>
      </div>
    )}

    {showToast && (
      <div className="toast-overlay">
        <div className="toast-container success">
          <div className="toast-content">
            <Check size={18} style={{ marginRight: "8px" }} />
            {toastMessage}
          </div>
        </div>
      </div>
    )}

    <aside
      className={`sidebar ${isOpen ? "open" : ""} ${
        collapsed ? "collapsed" : ""
      } ${isHovering ? "hovering" : ""}`}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
    >
      <div className="sidebar-header">
        <div className="sidebar-profile">
          <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
         <a href="/masteradmin"> <span className="sidebar-username">Admin</span></a>
        </div>
```

```
</div>

<nav className="sidebar-menu-wrapper">
  <ul className="sidebar-menu">
    <li>
      <NavLink to="/masteradmin/UserManagement" className={navLinkClass}>
        <Users size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>User Management</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/SystemOverview" className={navLinkClass}>
        <LayoutDashboard size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>System Overview</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/Services" className={navLinkClass}>
        <Hospital size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Services</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/PatientJourney" className={navLinkClass}>
        <Workflow size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Patient Journey</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/appointments" className={navLinkClass}>
        <Calendar size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Appointments</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/AdminBlogSection" className={navLinkClass}>
        <Newspaper size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Blog Section</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/AllReports" className={navLinkClass}>
        <BarChart2 size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>All Reports</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/ReportBuilder" className={navLinkClass}>
        <BarChart2 size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Report Builder</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/TemplateDemo" className={navLinkClass}>
        <BarChart2 size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>TemplateDemo</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/ActivityLogs" className={navLinkClass}>
        <BookText size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Activity Logs</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/SecurityControls" className={navLinkClass}>
        <Lock size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
        <span>Security Controls</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/masteradmin/AdminPasswordRequests" className={navLinkClass}>
        <KeyRound size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
```

```
              <span>Password Requests</span>
            </NavLink>
          </li>
          <li>
            <a
              href="#"
              onClick={handleLogoutClick}
              className={navLinkClass({ isActive: false })}
              style={{ cursor: "pointer", display: "flex", alignItems: "center" }}
            >
              <LogOut size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
              <span>Log Out</span>
            </a>
          </li>
        </ul>

        <button className="collapse-toggle" onClick={toggleCollapse}>
          <div className="claude-toggle">
            {collapsed ? <ChevronRight size={16} /> : <ChevronLeft size={16} />}
          </div>
        </button>
      </nav>
    </aside>
  </>
 );
};

export default MasterAdminSidebar;
```

File: src\dashboard\master_admin__tests__\AppointmentsContainer.test.js

```
import React from "react";
import { render, screen } from "@testing-library/react";
import AppointmentsContainer from "../AppointmentsContainer";
import { MemoryRouter } from "react-router-dom";

describe("AppointmentsContainer - Admin Cards", () => {
  beforeEach(() => {
    render(
      <MemoryRouter>
        <AppointmentsContainer />
      </MemoryRouter>
    );
  });

  it("renders doctor appointment card", () => {
    expect(screen.getByText(/Doctor Appointments/i)).toBeInTheDocument();
    expect(screen.getByText(/Manage all doctor bookings/i)).toBeInTheDocument();
  });

  it("renders dietitian appointment card", () => {
    expect(screen.getByText(/Dietitian Appointments/i)).toBeInTheDocument();
    expect(
      screen.getByText(/Manage all dietitian bookings/i)
    ).toBeInTheDocument();
  });

  it("renders physio appointment card", () => {
    expect(screen.getByText(/Physio Appointments/i)).toBeInTheDocument();
    expect(screen.getByText(/Manage all physio bookings/i)).toBeInTheDocument();
  });

  it("renders counselor appointment card", () => {
    expect(screen.getByText(/Counselor Appointments/i)).toBeInTheDocument();
    expect(
      screen.getByText(/Manage all counselor bookings/i)
    ).toBeInTheDocument();
  });

  it("renders phlebotomist appointment card", () => {
    expect(screen.getByText(/Phlebotomist Appointments/i)).toBeInTheDocument();
    expect(
      screen.getByText(/Manage all phlebotomist bookings/i)
    ).toBeInTheDocument();
```

```
  });

  it("renders correct number of view buttons", () => {
    const buttons = screen.getAllByRole("button", { name: /View/i });
    expect(buttons.length).toBe(5); // 1 for each appointment type
  });
});
```

File: src\dashboard\phlebotomist\phlebotomist.css

```
/* Import Google Fonts */
@import url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap');

/* Reset and Base Styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Inter', sans-serif;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  min-height: 100vh;
  padding: 20px;
  color: #333;
}

/* Container */
.phlebotomist-container {
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
}

/* Card Styles */
.card {
  background: rgba(255, 255, 255, 0.95);
  backdrop-filter: blur(10px);
  border-radius: 20px;
  padding: 40px;
  box-shadow: 0 20px 40px rgba(0, 0, 0, 0.1);
  border: 1px solid rgba(255, 255, 255, 0.2);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 25px 50px rgba(0, 0, 0, 0.15);
}

/* Header */
.card h2 {
  font-size: 2.5rem;
  font-weight: 700;
  text-align: center;
  margin-bottom: 30px;
  background: linear-gradient(135deg, #667eea, #764ba2);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  background-clip: text;
}

/* Form Styles */
.form {
  display: flex;
  flex-direction: column;
  gap: 25px;
}

/* Label Styles */
label {
  display: flex;
  flex-direction: column;
```

```
  font-weight: 500;
  font-size: 1rem;
  color: #4a5568;
  gap: 8px;
}

/* Input Styles */
input[type="text"],
input[type="email"],
input[type="tel"],
input[type="datetime-local"],
select,
textarea {
  padding: 15px 20px;
  border: 2px solid #e2e8f0;
  border-radius: 12px;
  font-size: 1rem;
  font-family: 'Inter', sans-serif;
  transition: all 0.3s ease;
  background: #ffffff;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
}

input[type="text"]:focus,
input[type="email"]:focus,
input[type="tel"]:focus,
input[type="datetime-local"]:focus,
select:focus,
textarea:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
  transform: translateY(-2px);
}

/* Textarea Specific */
textarea {
  resize: vertical;
  min-height: 100px;
  line-height: 1.5;
}

/* Select Dropdown */
select {
  cursor: pointer;
  appearance: none;
  background-image: url("data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/svg' fill='none' viewBox='0 0 2
  background-position: right 15px center;
  background-repeat: no-repeat;
  background-size: 16px;
  padding-right: 50px;
}

/* Disabled Select Options */
select option:disabled {
  color: #9ca3af;
  background-color: #f9fafb;
}

/* Small Helper Text */
small {
  font-size: 0.875rem;
  color: #6b7280;
  margin-top: 5px;
  font-style: italic;
}

/* Button Styles */
.btn {
  padding: 15px 30px;
  border: none;
  border-radius: 12px;
  font-size: 1rem;
```

```
  font-weight: 600;
  cursor: pointer;
  transition: all 0.3s ease;
  font-family: 'Inter', sans-serif;
  text-transform: uppercase;
  letter-spacing: 0.5px;
  position: relative;
  overflow: hidden;
}

.btn-primary {
  background: linear-gradient(135deg, #667eea, #764ba2);
  color: white;
  box-shadow: 0 4px 15px rgba(102, 126, 234, 0.3);
}

.btn-primary:hover {
  transform: translateY(-3px);
  box-shadow: 0 8px 25px rgba(102, 126, 234, 0.4);
}

.btn-primary:active {
  transform: translateY(-1px);
}

/* Button Ripple Effect */
.btn::before {
  content: '';
  position: absolute;
  top: 50%;
  left: 50%;
  width: 0;
  height: 0;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.3);
  transform: translate(-50%, -50%);
  transition: width 0.3s, height 0.3s;
}

.btn:active::before {
  width: 300px;
  height: 300px;
}

/* Phone Input Step */
.phone-step {
  text-align: center;
}

.phone-step label {
  font-size: 1.1rem;
  margin-bottom: 20px;
  color: #4a5568;
}

.phone-step input {
  margin-bottom: 25px;
  text-align: center;
  font-size: 1.1rem;
}

/* Returning User Badge */
.returning-user-badge {
  display: inline-block;
  background: linear-gradient(135deg, #10b981, #059669);
  color: white;
  padding: 8px 16px;
  border-radius: 20px;
  font-size: 0.875rem;
  font-weight: 500;
  margin-bottom: 20px;
  box-shadow: 0 2px 8px rgba(16, 185, 129, 0.3);
}
```

```
/* Slot Status Indicators */
.slot-status {
  display: flex;
  align-items: center;
  gap: 10px;
  margin-bottom: 20px;
  padding: 15px;
  background: #f8fafc;
  border-radius: 10px;
  border-left: 4px solid #667eea;
}

.slot-indicator {
  display: flex;
  align-items: center;
  gap: 5px;
  font-size: 0.875rem;
}

.slot-dot {
  width: 8px;
  height: 8px;
  border-radius: 50%;
}

.slot-dot.available {
  background: #10b981;
}

.slot-dot.full {
  background: #ef4444;
}

.slot-dot.limited {
  background: #f59e0b;
}

/* Form Animations */
.form-enter {
  animation: slideInUp 0.6s ease-out;
}

@keyframes slideInUp {
  from {
    opacity: 0;
    transform: translateY(30px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

/* Loading State */
.loading {
  opacity: 0.7;
  pointer-events: none;
}

.loading .btn {
  position: relative;
}

.loading .btn::after {
  content: '';
  position: absolute;
  top: 50%;
  left: 50%;
  width: 20px;
  height: 20px;
  margin: -10px 0 0 -10px;
  border: 2px solid transparent;
```

```
  border-top: 2px solid #ffffff;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}

/* Success Message */
.success-message {
  background: linear-gradient(135deg, #10b981, #059669);
  color: white;
  padding: 20px;
  border-radius: 12px;
  margin-top: 20px;
  text-align: center;
  font-weight: 500;
  box-shadow: 0 4px 15px rgba(16, 185, 129, 0.3);
}

/* Error Message */
.error-message {
  background: linear-gradient(135deg, #ef4444, #dc2626);
  color: white;
  padding: 20px;
  border-radius: 12px;
  margin-top: 20px;
  text-align: center;
  font-weight: 500;
  box-shadow: 0 4px 15px rgba(239, 68, 68, 0.3);
}

/* Medical Icons */
.medical-icon {
  font-size: 3rem;
  text-align: center;
  margin-bottom: 20px;
  color: #667eea;
}

/* Responsive Design */
@media (max-width: 768px) {
  .phlebotomist-container {
    padding: 10px;
  }

  .card {
    padding: 30px 20px;
    border-radius: 15px;
  }

  .card h2 {
    font-size: 2rem;
  }

  .form {
    gap: 20px;
  }

  .btn {
    padding: 12px 25px;
    font-size: 0.9rem;
  }
}

@media (max-width: 480px) {
  .card {
    padding: 20px 15px;
  }

  .card h2 {
```

```
    font-size: 1.75rem;
  }

  input[type="text"],
  input[type="email"],
  input[type="tel"],
  input[type="datetime-local"],
  select,
  textarea {
    padding: 12px 15px;
    font-size: 0.9rem;
  }
}

/* Print Styles */
@media print {
  body {
    background: white;
  }

  .card {
    box-shadow: none;
    border: 1px solid #ddd;
  }

  .btn {
    display: none;
  }
}
```

File: src\dashboard\phlebotomist\PhlebotomistAppointments.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { toast } from "react-toastify";

const API = "http://localhost:3001/phleboBookingsData";

const PhlebotomistAppointments = () => {
  const [bookings, setBookings] = useState([]);

  const fetchBookings = async () => {
    try {
      const res = await axios.get(API);
      const today = new Date().toISOString().split("T")[0];
      const filtered = res.data.filter((b) =>
        b.fastingDate.startsWith(today)
      );
      setBookings(filtered);
    } catch (err) {
      console.error("Error fetching bookings:", err);
      toast.error("Failed to load appointments");
    }
  };

  const handleDelete = async (id) => {
    try {
      await axios.delete(`${API}/${id}`);
      toast.success("Booking cancelled");
      fetchBookings(); // reload list
    } catch (err) {
      console.error("Delete error:", err);
      toast.error("Could not cancel booking");
    }
  };

  const handleReschedule = (phone, slot) => {
    const msg = `Hi, please reschedule your phlebotomy slot (${slot}) at your convenience.`;
    const waLink = `https://wa.me/${phone}?text=${encodeURIComponent(msg)}`;
    window.open(waLink, "_blank");
    toast.info("Opening WhatsApp to reschedule...");
  };

  useEffect(() => {
```

```
    fetchBookings();
  }, []);

  return (
    <div className="card">
      <h2>Today's Bookings</h2>
      {bookings.length === 0 ? (
        <p>No bookings for today.</p>
      ) : (
        <div style={{ overflowX: "auto" }}>
          <table className="table">
            <thead>
              <tr>
                <th>Name</th>
                <th>Phone</th>
                <th>Test</th>
                <th>Slot</th>
                <th>Reschedule</th>
                <th>Cancel</th>
              </tr>
            </thead>
            <tbody>
              {bookings.map((b) => (
                <tr key={b.id}>
                  <td>{b.name}</td>
                  <td>{b.phone}</td>
                  <td>{b.testType}</td>
                  <td>{b.slot}</td>
                  <td>
                    <button className="btn btn-secondary" onClick={() => handleReschedule(b.phone, b.slot)}>
                      WhatsApp
                    </button>
                  </td>
                  <td>
                    <button className="btn btn-danger" onClick={() => handleDelete(b.id)}>
                      Delete ??
                    </button>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      )}
    </div>
  );
};

export default PhlebotomistAppointments;
```

File: src\dashboard\phlebotomist\PhlebotomistBookingForm.js

```
import React, { useEffect, useState } from "react";
import { toast } from "react-toastify";

const PhlebotomistBookingForm = () => {
  const initialForm = {
    name: "",
    phone: "",
    email: "",
    address: "",
    testType: "",
    isFirstTime: "yes",
    fastingDate: "",
    slot: "",
  };

  const [formData, setFormData] = useState(initialForm);
  const [slotCounts, setSlotCounts] = useState({});
  const [isReturningUser, setIsReturningUser] = useState(null);
  const [step, setStep] = useState("phone"); // 'phone' or 'details'

  const slots = ["7:00 AM", "7:15 AM", "7:30 AM", "7:45 AM", "8:00 AM"];
```

```
useEffect(() => {
  const counts = JSON.parse(localStorage.getItem("phleboBookings")) || {};
  setSlotCounts(counts);
}, []);

const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData((prev) => ({ ...prev, [name]: value }));
};

const handlePhoneCheck = () => {
  const stored = JSON.parse(localStorage.getItem("phleboBookingsData")) || [];
  const bookingsForPhone = stored.filter((entry) => entry.phone === formData.phone);

  if (bookingsForPhone.length >= 2) {
    setIsReturningUser(true);
    setFormData((prev) => ({
      ...prev,
      name: bookingsForPhone[0].name,
      email: bookingsForPhone[0].email,
      address: bookingsForPhone[0].address,
      testType: bookingsForPhone[0].testType,
      isFirstTime: "no",
    }));
  } else {
    setIsReturningUser(false);
  }

  setStep("details");
};

const handleSubmit = (e) => {
  e.preventDefault();

  const selectedTime = new Date(formData.fastingDate);
  const hour = selectedTime.getHours();
  const minute = selectedTime.getMinutes();
  if (hour !== 7 && !(hour === 8 && minute === 0)) {
    toast.error("Fasting must be between 7:00 AM and 8:00 AM");
    return;
  }

  const currentSlot = formData.slot;
  const updatedCounts = { ...slotCounts };
  updatedCounts[currentSlot] = (updatedCounts[currentSlot] || 0) + 1;

  if (updatedCounts[currentSlot] > 5) {
    toast.error("Slot full. Please choose another time.");
    return;
  }

  // Save full data
  const existingData = JSON.parse(localStorage.getItem("phleboBookingsData")) || [];
  const updatedData = [...existingData, formData];
  localStorage.setItem("phleboBookingsData", JSON.stringify(updatedData));

  // Save updated slot count
  localStorage.setItem("phleboBookings", JSON.stringify(updatedCounts));
  setSlotCounts(updatedCounts);

  toast.success("Booking confirmed & notified via WhatsApp/Email!");
  console.log("Booking Info:", formData);

  // Reset
  setFormData(initialForm);
  setIsReturningUser(null);
  setStep("phone");
};

return (
  <div className="card">
    <h2>Phlebotomist Booking</h2>
```

```
      {step === "phone" && (
        <>
          <label>Enter Phone to Continue:</label>
          <input
            name="phone"
            value={formData.phone}
            onChange={handleChange}
            required
          />
          <button className="btn btn-primary" onClick={handlePhoneCheck}>
            Proceed
          </button>
        </>
      )}

      {step === "details" && (
        <form onSubmit={handleSubmit} className="form" style={{ marginTop: "1rem" }}>
          {!isReturningUser && (
            <>
              <label>Name:
                <input name="name" value={formData.name} onChange={handleChange} required />
              </label>
              <label>Email:
                <input name="email" type="email" value={formData.email} onChange={handleChange} required />
              </label>
              <label>Address:
                <textarea name="address" value={formData.address} onChange={handleChange} required />
              </label>
              <label>Test Type:
                <select name="testType" value={formData.testType} onChange={handleChange} required>
                  <option value="">Select</option>
                  <option value="Blood Test">Blood Test</option>
                  <option value="Diabetes Test">Diabetes Test</option>
                </select>
              </label>
            </>
          )}

          <label>Fasting Time:
            <input type="datetime-local" name="fastingDate" value={formData.fastingDate} onChange={handleChange}
            <small>Allowed between 7:00 ? 8:00 AM</small>
          </label>

          <label>Slot:
            <select name="slot" value={formData.slot} onChange={handleChange} required>
              <option value="">Choose Slot</option>
              {slots.map((s) => (
                <option key={s} value={s} disabled={slotCounts[s] >= 5}>
                  {s} {slotCounts[s] >= 5 ? "(Full)" : `(Booked: ${slotCounts[s] || 0})`}
                </option>
              ))}
            </select>
          </label>

          <button type="submit" className="btn btn-primary">Confirm Booking</button>
        </form>
      )}
    </div>
  );
};

export default PhlebotomistBookingForm;
```

File: src\dashboard\phlebotomist\PhlebotomistDashboardHome.js

```
// Example: PhlebotomistDashboardHome.js
import React from 'react';

const PhlebotomistDashboardHome = () => {
  return <h1>Phlebotomist Dashboard Home</h1>;
};

export default PhlebotomistDashboardHome;
```

File: src\dashboard\phlebotomist\components\phlebotomist_footer.js

```
import React from 'react';
import '../../physio/components/Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} Doctor Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\phlebotomist\components\phlebotomist_navbar.js

```
import React from "react";
import "../../physio/components/Navbar.css";
import { Moon, Sun } from "lucide-react";
import { useTheme } from "../../../ThemeProvider";

const PhlebotomistNavbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="../phlebotomist/PhlebotomistDashboardHome"> <div className="navbar-logo">Doctor</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default PhlebotomistNavbar;
```

File: src\dashboard\phlebotomist\components\phlebotomist_sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "../../physio/components/Sidebar.css";
import {
  Home,
  Users,
  Dumbbell,
  BarChart2,
  LogOut,
  Hamburger,
  Menu,
  KeyRoundIcon,
  Calendar,
  Workflow,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
} from "lucide-react";

const DietSidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [toastMessage, setToastMessage] = useState("");
  const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
  const navigate = useNavigate();

  const toggleSidebar = () => {
```

```
  setIsOpen(!isOpen);
};

const toggleCollapse = () => {
  setCollapsed(!collapsed);
};

const handleResize = () => {
  setIsOpen(window.innerWidth > 768);
};

const closeSidebar = () => {
  if (isOpen && window.innerWidth <= 768) {
    setIsOpen(false);
  }
};

const handleMouseEnter = () => {
  if (collapsed && window.innerWidth > 768) {
    setIsHovering(true);
  }
};

const handleMouseLeave = () => {
  if (collapsed && window.innerWidth > 768) {
    setIsHovering(false);
  }
};

useEffect(() => {
  window.addEventListener("resize", handleResize);
  return () => window.removeEventListener("resize", handleResize);
}, []);

useEffect(() => {
  document.body.classList.toggle(
    "sidebar-collapsed",
    collapsed && !isHovering
  );
}, [collapsed, isHovering]);

const navLinkClass = ({ isActive }) => (isActive ? "active" : "");

const handleLogoutClick = (e) => {
  e.preventDefault();
  setShowLogoutConfirm(true);
};

const confirmLogout = () => {
  localStorage.clear();
  setShowLogoutConfirm(false);
  setToastMessage("Logged out successfully.");
  setShowToast(true);
  setTimeout(() => {
    setShowToast(false);
    navigate("/login");
  }, 3000);
};

const cancelLogout = () => {
  setShowLogoutConfirm(false);
};

return (
  <>
    <button
      className="sidebar-toggle"
      onClick={toggleSidebar}
      aria-label="Toggle sidebar"
    >
      <Menu size={24} />
    </button>
```

```
{isOpen && window.innerWidth <= 768 && (
  <div className="sidebar-overlay show" onClick={closeSidebar}></div>
)}

{showLogoutConfirm && (
  <div className="modal-overlay">
    <div className="modal-container">
      <div className="modal-header">
        <center>
          <h3>Logout Confirmation</h3>
        </center>
        <button className="modal-close" onClick={cancelLogout}>
          <X size={18} />
        </button>
      </div>
      <div className="modal-body">
        <p>Are you sure you want to log out?</p>
      </div>
      <div className="modal-footer">
        <button className="btn btn-primary" onClick={confirmLogout}>
          Yes, Logout
        </button>
      </div>
    </div>
  </div>
)}

{showToast && (
  <div className="toast-overlay">
    <div className="toast-container success">
      <div className="toast-content">
        <Check size={18} style={{ marginRight: "8px" }} />
        {toastMessage}
      </div>
    </div>
  </div>
)}

<aside
  className={`sidebar ${isOpen ? "open" : ""} ${
    collapsed ? "collapsed" : ""
  } ${isHovering ? "hovering" : ""}`}
  onMouseEnter={handleMouseEnter}
  onMouseLeave={handleMouseLeave}
>
  <div className="sidebar-header">
    <div className="sidebar-profile">
      <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
      <a href="/phlebotomist">
        {" "}
        <span className="sidebar-username">Phlebotomist</span>
      </a>
    </div>
  </div>

  <nav className="sidebar-menu-wrapper">
    <ul className="sidebar-menu">
      <li>
        <NavLink to="/phlebotomist" className={navLinkClass}>

          <Home
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Dashboard Home</span>
        </NavLink>
      </li>

      <li>
       <NavLink to="/phlebotomist/PhlebotomistAppointmentss" className={navLinkClass}>
          <Calendar
```

```
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Appointments</span>
            </NavLink>
          </li>
          <li>
           <NavLink to="/phlebotomist/PhlebotomistBookingForm" className={navLinkClass}>
              <Calendar
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span> Booking</span>
            </NavLink>
          </li>
          <li>
           <NavLink to="/phlebotomist/PhlebotomistPasswordRequest" className={navLinkClass}>
              <KeyRoundIcon
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Reset Password</span>
            </NavLink>
          </li>

          <li>
            <a
              href="#"
              onClick={handleLogoutClick}
              className={navLinkClass({ isActive: false })}
              style={{
                cursor: "pointer",
                display: "flex",
                alignItems: "center",
              }}
            >
              <LogOut
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Log Out</span>
            </a>
          </li>
        </ul>

        <button className="collapse-toggle" onClick={toggleCollapse}>
          <div className="claude-toggle">
            {collapsed ? (
              <ChevronRight size={16} />
            ) : (
              <ChevronLeft size={16} />
            )}
          </div>
        </button>
      </nav>
    </aside>
  </>
 );
};

export default DietSidebar;
```

File: src\dashboard\phlebotomist\pages\PhlebotomistPasswordRequest.js

```
import React, { useState } from "react";
import axios from "axios";
```

```
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../../ThemeProvider"; // Adjust the import based on your project structure
import "react-toastify/dist/ReactToastify.css";
import "../../master_admin/master_admin.css"; // Ensure this path is correct

const PhlebotomistPasswordRequest = () => {
  const { theme } = useTheme(); // Access the current theme
  const [formData, setFormData] = useState({
    counselorName: "",
    email: "",
    reason: "",
  });

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!formData.counselorName || !formData.email || !formData.reason) {
      toast.error("All fields are required!");
      return;
    }

    try {
      await axios.post("http://localhost:5000/passwordChangeRequests", {
        ...formData,
        status: "Pending",
        requestedAt: new Date().toISOString(),
      });

      toast.success("Request submitted to admin!");
      setFormData({ counselorName: "", email: "", reason: "" });
    } catch (err) {
      toast.error("Failed to submit request.");
    }
  };

  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Password Change Request</h1>
      <div className="card" style={{border:"1px solid #cc5500"}}>
        <form onSubmit={handleSubmit} className="form">
          <div className="form-group">
            <label htmlFor="counselorName">Counselor Name:</label>
            <input
              type="text"
              id="counselorName"
              name="counselorName"
              placeholder="Enter your name"
              value={formData.counselorName}
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="email">Email ID:</label>
            <input
              type="email"
              id="email"
              placeholder="Enter your email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="input-field"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="reason">Reason for Change:</label>
```

```
            <input
              type="text"
              id="reason"
              placeholder="Enter reason for password change"
              name="reason"
              value={formData.reason}
              onChange={handleChange}
              className="input-field"

              required
            />
          </div>
          <div className="center-btn">
           <center> <button type="submit" className="btn btn-primary">
              Submit Request
            </button></center>
          </div>
        </form>
      </div>
    </div>
  );
};

export default PhlebotomistPasswordRequest;
```

File: src\dashboard\physio\assign.css

```
/* Base reset */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* Light Theme (Default) */
:root {
  --bg-primary: #ffffff;
  --bg-secondary: #f8f9fa;
  --text-primary: #212529;
  --text-secondary: #6c757d;
  --accent: #d35400;
  --accent-light: #e67e22;
  --accent-hover: #a04000;
  --border-color: #cc5500;
  --shadow-color: rgba(0, 0, 0, 0.1);
  --bg-hover: #f1f1f1;
}

/* Dark Theme */
.dark {
  --bg-primary: #121212;
  --bg-secondary: #1e1e1e;
  --text-primary: #f1f1f1;
  --text-secondary: #bbbbbb;
  --accent: #ff7f50;
  --accent-light: #ffa07a;
  --accent-hover: #ff5722;
  --border-color: #444;
  --shadow-color: rgba(255, 255, 255, 0.05);
  --bg-hover: #2a2a2a;
}

/* Layout */
.dashboard-container,
.assign-container {
  display: flex;
  flex-direction: column;
  padding: 1rem;
  background-color: var(--bg-primary);
  color: var(--text-primary);
  width: 100%;
  max-width: 100%;
}
```

```
/* Responsive container */
.assign-page {
  width: 100%;
  min-height: 100vh;
  background-color: var(--bg-primary);
}

/* Horizontal flex for top section */
.assign-flex {
  display: flex;
  gap: 20px;
  margin-bottom: 2rem;
  flex-wrap: wrap;
  width: 100%;
}

/* Box-style section */
.assign-card {
  flex: 1;
  min-width: 280px;
  border: 1px solid var(--border-color);
  border-radius: 8px;
  padding: 1.25rem;
  background-color: var(--bg-secondary);
  margin-bottom: 1rem;
}

/* Section Title */
.section-title {
  font-size: 1.2rem;
  margin-bottom: 1rem;
  color: var(--accent);
  border-bottom: 2px solid var(--accent-light);
  padding-bottom: 0.5rem;
}

.input-group {
  display: flex;
  width: 100%;
  align-items: center;
  gap: 0.5rem;
  flex-wrap: wrap;
  padding-bottom: 20px;
}

.search-inputs {
  flex: 1; /* Input stretches fully */
  min-width: 150px;
  padding: 0.75rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background-color: var(--bg-primary);
  color: var(--text-primary);
}

.search-button {
  padding: 0.6rem 1rem;
  background-color: var(--accent);
  color: white;
  border: none;
  border-radius: 6px;
  white-space: nowrap;
  cursor: pointer;
  transition: background 0.3s ease;
  display: flex;
  align-items: center;
  gap: 5px;
}

.search-button:hover {
  background-color: var(--accent-hover);
}
```

```
/* User Info & Report */
.user-info p,
.report-text p {
  margin-bottom: 0.5rem;
}

/* Button Styles */
.btn {
  display: inline-flex;
  align-items: center;
  gap: 0.4rem;
  padding: 0.6rem 1rem;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  transition: all 0.2s ease;
  font-size: 0.9rem;
}

.btn-primary {
  background-color: var(--accent);
  color: white;
}

.btn-primary:hover {
  background-color: var(--accent-hover);
}

.btn:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

.btn-view {
  background-color: var(--accent);
  color: white;
}

.btn-remove {
  background-color: #dc3545;
  color: white;
}

.btn-view:hover {
  background-color: var(--accent-hover);
}

.btn-remove:hover {
  background-color: #b52a37;
}

/* Table Section */
.table-section {
  background-color: var(--bg-secondary);
  border-radius: 8px;
  padding: 1rem;
  border: 1px solid var(--border-color);
  box-shadow: 0 4px 8px var(--shadow-color);
  overflow: hidden;
  margin-bottom: 30px;
  width: 100%;
}

.table-responsive {
  width: 100%;
  overflow-x: auto;
  -webkit-overflow-scrolling: touch;
}

.table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 1rem;
```

```
  table-layout: fixed;
}

.table th,
.table td {
  padding: 0.75rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
  vertical-align: middle;
}

.table th {
  background-color: var(--bg-hover);
  color: var(--text-primary);
  font-weight: bold;
  white-space: nowrap;
}

/* Table column widths */
.chart-column {
  width: 10%;
}

.dates-column {
  width: 27%;
}

.upload-column {
  width: 20%;
}

.feedback-column {
  width: 20%;
}

.actions-column {
  width: 23%;
}

/* Date Inputs */
.date-inputs {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
  align-items: center;
  width: 100%;
}

.date-field {
  display: flex;
  align-items: center;
  gap: 0.3rem;
  background: var(--bg-primary);
  padding: 0.4rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  min-width: 110px;
  flex: 1;
}

.date-input {
  border: none;
  background: transparent;
  color: var(--text-primary);
  font-size: 0.85rem;
  outline: none;
  width: 100%;
}

.date-separator {
  color: var(--text-secondary);
  white-space: nowrap;
  padding: 0 0.3rem;
```

```
}

/* Upload */
.upload-container {
  width: 100%;
  overflow: hidden;
}

.upload-label {
  display: inline-flex;
  align-items: center;
  gap: 0.4rem;
  padding: 0.5rem 0.75rem;
  border: 1px solid var(--accent);
  border-radius: 6px;
  background: transparent;
  color: var(--accent);
  cursor: pointer;
  transition: background 0.3s ease;
  max-width: 100%;
  overflow: hidden;
  width: 100%;
}

.file-name {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  max-width: 100%;
}

.upload-label:hover {
  background-color: var(--accent);
  color: white;
}

.file-input {
  display: none;
}

/* Feedback Input */
.feedback-input {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background-color: var(--bg-primary);
  color: var(--text-primary);
}

/* Actions */
.actions-cell {
  width: 100%;
}

.button-text {
  display: inline;
  padding: 0.3rem 0.5rem;
}

.action-buttons {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
  justify-content: flex-start;
}

/* Modal */
.modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
```

```
  height: 100%;
  background-color: rgba(0, 0, 0, 0.6);
  z-index: 1000;
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 1rem;
}

.modal-content {
  background: var(--bg-primary);
  border-radius: 10px;
  width: 95%;
  max-width: 900px;
  max-height: 90vh;
  display: flex;
  flex-direction: column;
  overflow: hidden;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
}

.modal-header {
  padding: 1rem;
  border-bottom: 1px solid var(--border-color);
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.modal-header h3 {
  color: var(--accent);
  font-size: 1.2rem;
}

.close-button {
  background: none;
  border: none;
  color: var(--text-secondary);
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
}

.modal-body {
  padding: 1rem;
  overflow-y: auto;
}

/* Excel fallback */
.excel-container {
  text-align: center;
  padding: 2rem 1rem;
}

.download-button {
  display: inline-flex;
  align-items: center;
  gap: 0.5rem;
  background: var(--accent);
  color: white;
  padding: 0.6rem 1rem;
  border-radius: 6px;
  text-decoration: none;
  font-weight: 500;
  transition: background 0.3s ease;
  margin-top: 1rem;
}

.download-button:hover {
  background: var(--accent-hover);
}
```

```
@media (max-width: 768px) {
  .meal-input {
    max-width: 100%;
  }
}

/* Enhanced Responsive Adjustments */
@media (max-width: 1200px) {
  .assign-flex {
    flex-direction: column;
  }

  .assign-card {
    width: 100%;
    min-width: 100%;
  }

  .table th,
  .table td {
    padding: 0.75rem 0.5rem;
  }
}

@media (max-width: 992px) {
  .action-text {
    display: none;
  }

  .btn {
    padding: 0.5rem;
    min-width: 36px;
    justify-content: center;
  }

  .chart-column {
    width: 10%;
  }

  .dates-column {
    width: 30%;
  }

  .upload-column {
    width: 20%;
  }

  .feedback-column {
    width: 20%;
  }

  .actions-column {
    width: 20%;
  }
}

@media (max-width: 768px) {
  .input-group {
    flex-direction: column;
    align-items: flex-start;
  }

  .search-inputs,
  .search-button {
    width: 100%;
    height: 100%;
  }

  .button-text {
    display: inline;
  }

  .date-inputs {
    flex-direction: column;
```

```
  width: 100%;
  gap: 0.75rem;
}

.date-separator {
  align-self: center;
  padding: 0.3rem 0;
}

.table th,
.table td {
  padding: 0.5rem 0.3rem;
}

/* Stack the table */
.table thead {
  display: none;
}

.table,
.table tbody,
.table tr,
.table td {
  display: block;
  width: 100%;
}

.table tr {
  margin-bottom: 1.5rem;
  border: 1px solid var(--border-color);
  border-radius: 8px;
  overflow: hidden;
  background-color: var(--bg-primary);
  padding: 0.5rem;
}

.table td {
  position: relative;
  padding-left: 120px;
  text-align: left;
  min-height: 50px;
}

.table td:before {
  content: attr(data-label);
  position: absolute;
  left: 0;
  top: 0;
  padding: 0.5rem;
  width: 110px;
  font-weight: bold;
  color: var(--accent);
}

.table td:nth-child(1):before {
  content: "Chart";
}

.table td:nth-child(2):before {
  content: "Dates";
}

.table td:nth-child(3):before {
  content: "Upload";
}

.table td:nth-child(4):before {
  content: "Feedback";
}

.table td:nth-child(5):before {
  content: "Actions";
}
```

```
  /* Responsive actions */
  .action-buttons {
    justify-content: flex-start;
  }

  .btn {
    padding: 0.5rem 0.7rem;
  }

  .action-text {
    display: inline;
  }
}

@media (max-width: 576px) {
  .assign-container,
  .table-section {
    padding: 0.75rem;
  }

  .section-title {
    font-size: 1.1rem;
  }

  .modal-content {
    width: 100%;
    height: 100%;
    max-height: 100%;
    border-radius: 0;
  }

  .table td {
    padding-left: 100px;
  }

  .table td:before {
    width: 90px;
  }

  .date-field {
    min-width: 100%;
  }
}

/* Print styles */
@media print {
  .modal,
  .action-buttons button:not(.btn-view) {
    display: none;
  }

  .assign-page {
    background: white;
  }

  .table {
    border: 1px solid #ddd;
  }
}
/* In assign.css or master_admin.css */
.responsive-table-wrapper {
  width: 100%;
  overflow-x: auto;
  -webkit-overflow-scrolling: touch;
}

.responsive-table-wrapper table {
  min-width: 800px;
  border-collapse: collapse;
}
/* Custom Scrollbar for MODIFIED BMI popup */
.popup-content::-webkit-scrollbar {
```

```
  width: 8px;
}

.popup-content::-webkit-scrollbar-track {
  background: transparent;
}

.popup-content::-webkit-scrollbar-thumb {
  background-color: #cc5500;
  border-radius: 4px;
}

.popup-content::-webkit-scrollbar-thumb:hover {
  background-color: #b34700; /* slightly darker on hover */
}

/* Firefox Support */
.popup-content {
  scrollbar-color: #cc5500 transparent;
  scrollbar-width: thin;
}
/* Compact the checkbox column */
.table th:first-child,
.table td:first-child {
  width: 50px;
  text-align: center;
  vertical-align: middle;
  padding: 0.25rem;
  color: #cc5500;
}
.bmi-highlight {
  animation: pulse-glow 1s ease-in-out infinite alternate;
  border: 2px solid #cc5500 !important;
  box-shadow: 0 0 15px rgba(204, 85, 0, 0.6);
}

@keyframes pulse-glow {
  0% {
    box-shadow: 0 0 10px rgba(204, 85, 0, 0.2);
  }
  100% {
    box-shadow: 0 0 20px rgba(204, 85, 0, 0.9);
  }
}
/* .shake {
  animation: shake 0.4s;
}

@keyframes shake {
  0% { transform: translateX(0); }
  25% { transform: translateX(-5px); }
  50% { transform: translateX(5px); }
  75% { transform: translateX(-5px); }
  100% { transform: translateX(0); }
}
 */
```

File: src\dashboard\physio\clientManagement.css

```
.client-page {
  min-height: 100vh;
  font-family: "Inter", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
    Oxygen, Ubuntu, Cantarell, sans-serif;
  transition: all 0.3s ease;
}

.client-page.light {
  background-color: #f9f9f9;
  color: #1f2937;
  --bg-card: #ffffff;
  --bg-secondary: #ffffff;
  --border-color: #e5e7eb;
  --text-secondary: #6b7280;
  --highlight-hover: #f3f4f6;
```

```
}

.client-page.dark {
  background-color: transparent;
  color: #f3f4f6;
  --bg-card: #1e1e1e;
  --bg-secondary: #1e1e1e;
  --border-color: #374151;
  --text-secondary: #9ca3af;
  --highlight-hover: #2d3748;
}

/* Primary brand color applied to both themes */
.client-page {
  --brand-color: #cc5500;
  --brand-color-light: #e07733;
  --brand-color-dark: #a34400;
}

.client-container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 1.5rem;
}

.client-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 2rem;
  flex-wrap: wrap;
  gap: 1rem;
}

.client-header h2 {
  font-size: 1.75rem;
  font-weight: 600;
  color: var(--brand-color);
  margin: 0;
}

.client-controls {
  display: flex;
  gap: 1rem;
  align-items: center;
  flex-wrap: wrap;
  margin-bottom: 20px;
}

/* Search Bar Styles */


.search-input {
  /* padding: 0.75rem 1rem; */
  border-radius: 8px;
  border: 1px solid var(--border-color);
  background-color: var(--bg-card);
  color: inherit;
  min-width: 240px;
  transition: all 0.2s ease;
}

.search-input:focus {
  outline: none;
  box-shadow: 0 0 0 2px var(--brand-color-light);
}

/* Filter Button Styles */
.filter-buttons {
  display: flex;
  gap: 8px;
}
```

```
.filter-btn {
  padding: 0.75rem 1rem;
  border-radius: 8px;
  background-color: green;
  border: 1px solid var(--border-color);
  color: inherit;
  font-weight: 500;
  cursor: pointer;
  transition: all 0.2s ease;
}

.filter-btn:hover {
  background-color: var(--highlight-hover);
}

.filter-btn.in-progress {
  background-color: #ffc107; /* Bootstrap yellow */
  color: #212529; /* dark text for contrast */
  border: 1px solid #ffca2c;
}


.filter-btn.active {
  background-color: var(--brand-color);
  color: white;
  border-color: var(--brand-color);
}
.btn-active {
  background-color: #cc5500 !important; /* Highlight color */
  color: white !important;
  cursor: default;
  font-weight: bold;
  border: 2px solid #cc5500;}
/* Action Button Styles */
.action-buttons-right {
  display: flex;
  gap: 10px;
  margin-left: auto;
}

.add-btn {
  background: var(--brand-color);
  color: white;
  padding: 0.75rem 1.25rem;
  border: none;
  border-radius: 8px;
  font-weight: 500;
  cursor: pointer;
  display: flex;
  align-items: center;
  gap: 0.5rem;
  transition: background-color 0.2s ease;
}

.add-btn:hover {
  background: var(--brand-color-light);
}

.add-consultant-btn {
  background-color: #4a6da7;
  color: white;
  padding: 0.75rem 1.25rem;
  border: none;
  border-radius: 8px;
  font-weight: 500;
  cursor: pointer;
  display: flex;
  align-items: center;
  transition: background-color 0.2s ease;
}

.add-consultant-btn:hover {
  background-color: #3c5a8a;
```

```
}

.btn-icon {
  font-size: 1.1rem;
  font-weight: bold;
}

/* Badge Styles */
.type-badge,
.resolution-badge {
  display: inline-block;
  padding: 0.4rem 0.75rem;
  border-radius: 9999px;
  font-size: 0.85rem;
  font-weight: 500;
  text-align: center;
}

.type-badge.client {
  background-color: rgba(16, 185, 129, 0.1);
  color: #10b981;
}

.type-badge.consultant {
  background-color: rgba(245, 158, 11, 0.1);
  color: #f59e0b;
}

.resolution-badge.in-progress {
  background-color: rgba(59, 130, 246, 0.1);
  color: #3b82f6;
}

.resolution-badge.resolved {
  background-color: rgba(124, 58, 237, 0.1);
  color: #7c3aed;
}

/* Table Styling */
.table-responsive {
  overflow-x: auto;
  background-color: var(--bg-card);
  /* border-radius: 10px; */
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
}

.client-table {
  width: 100%;
  border-collapse: separate;
  border-spacing: 0;
}

.client-table th,
.client-table td {
  padding: 1rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
}

.client-table th {
  font-weight: 600;
  color: var(--text-secondary);
  text-transform: uppercase;
  font-size: 0.8rem;
  letter-spacing: 0.05em;
}

.client-table tbody tr {
  transition: background-color 0.2s ease;
}

.client-table tbody tr:hover {
  background-color: var(--highlight-hover);
```

```
}

.client-info {
  display: flex;
  align-items: center;
  gap: 0.75rem;
}

.avatar {
  width: 40px;
  height: 40px;
  border-radius: 50%;
  object-fit: cover;
  border: 2px solid var(--brand-color);
}

.client-details {
  display: flex;
  flex-direction: column;
}

.client-name {
  font-weight: 500;
}

.client-email-mobile {
  display: none;
  font-size: 0.85rem;
  color: var(--text-secondary);
}

.status-badge {
  display: inline-block;
  padding: 0.4rem 0.75rem;
  border-radius: 9999px;
  font-size: 0.85rem;
  font-weight: 500;
  text-align: center;
}

.status-badge.active {
  background-color: rgba(16, 185, 129, 0.1);
  color: #10b981;
}

.status-badge.inactive {
  background-color: rgba(239, 68, 68, 0.1);
  color: #ef4444;
}

.action-buttons {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
}

.edit-btn,
.delete-btn,
.resolve-btn {
  padding: 0.5rem 1rem;
  border-radius: 6px;
  border: none;
  font-size: 0.85rem;
  cursor: pointer;
  transition: all 0.2s ease;
}

.edit-btn {
  background-color: rgba(37, 99, 235, 0.1);
  color: #2563eb;
}

.edit-btn:hover {
```

```
  background-color: rgba(37, 99, 235, 0.2);
}

.delete-btn {
  background-color: rgba(239, 68, 68, 0.1);
  color: #ef4444;
}

.delete-btn:hover {
  background-color: rgba(239, 68, 68, 0.2);
}

.resolve-btn {
  background-color: rgba(16, 185, 129, 0.1);
  color: #10b981;
  white-space: nowrap;
}

.resolve-btn:hover {
  background-color: rgba(16, 185, 129, 0.2);
}

/* Modal Styling */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  background: rgba(0, 0, 0, 0.6);
  width: 100vw;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 2000;
  backdrop-filter: blur(4px);
}

.modal-box {
  background-color: var(--bg-secondary);
  color: inherit;
  border-radius: 12px;
  width: 450px;
  max-width: 90vw;
  box-shadow: 0 20px 25px -5px rgba(0, 0, 0, 0.1),
    0 10px 10px -5px rgba(0, 0, 0, 0.04);
  overflow: hidden;
  animation: modalFadeIn 0.3s ease-out;
}

@keyframes modalFadeIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1.25rem 1.5rem;
  border-bottom: 1px solid var(--border-color);
}

.modal-header h3 {
  font-size: 1.25rem;
  font-weight: 600;
  margin: 0;
  color: var(--brand-color);
```

```
}

.close-modal {
  background: none;
  border: none;
  font-size: 1.5rem;
  cursor: pointer;
  color: var(--text-secondary);
}

.modal-body {
  padding: 1.5rem;
}

.form-group {
  margin-bottom: 1.25rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
  font-size: 0.9rem;
  text-align: left;
}

.form-group input,
.form-group textarea,
.form-group select {
  width: 100%;
  padding: 0.75rem;
  border-radius: 8px;
  border: 1px solid var(--border-color);
  background-color: inherit;
  color: --var(--text-primary);
  font-size: 1rem;
  transition: all 0.2s ease;
}

.form-group input:focus,
.form-group select:focus {
  outline: none;
  box-shadow: 0 0 0 2px var(--brand-color-light);
  border-color: var(--brand-color);
}

.modal-actions {
  display: flex;
  justify-content: flex-end;
  gap: 1rem;
  padding: 1.25rem 1.5rem;
  border-top: 1px solid var(--border-color);
}

.cancel-btn {
  padding: 0.75rem 1.25rem;
  border-radius: 8px;
  border: 1px solid var(--border-color);
  background-color: transparent;
  color: inherit;
  font-weight: 500;
  cursor: pointer;
  transition: all 0.2s ease;
}

.cancel-btn:hover {
  background-color: var(--highlight-hover);
}

.save-btn {
  padding: 0.75rem 1.25rem;
  border-radius: 8px;
  border: none;
```

```
  background-color: var(--brand-color);
  color: white;
  font-weight: 500;
  cursor: pointer;
  transition: all 0.2s ease;
}

.save-btn:hover {
  background-color: var(--brand-color-light);
}

.no-results {
  padding: 3rem;
  text-align: center;
  color: var(--text-secondary);
}

/* Responsive styling */
@media (max-width: 768px) {
  .client-header {
    flex-direction: column;
    align-items: flex-start;
    gap: 1rem;
  }

  .client-controls {
    width: 100%;
    flex-direction: column;
    align-items: stretch;
  }

  .search-container {
    width: 100%;
    margin-bottom: 0.75rem;
  }

  .search-input {
    min-width: 0;
    width: 100%;
  }

  .filter-buttons {
    width: 100%;
    justify-content: space-between;
    margin-bottom: 0.75rem;
  }

  .action-buttons-right {
    width: 100%;
    justify-content: space-between;
    margin-left: 0;
  }

  .hide-sm {
    display: none;
  }

  .client-email-mobile {
    display: block;
  }

  .btn-text {
    display: none;
  }

  .edit-btn::before {
    content: "??";
  }

  .delete-btn::before {
    content: "??";
  }
```

```
  .resolve-btn::before {
    content: "?";
  }

  .edit-btn,
  .delete-btn,
  .resolve-btn {
    padding: 0.5rem;
    display: flex;
    align-items: center;
    justify-content: center;
  }
}
```

File: src\dashboard\physio\physio.css

```
:root {
  --text-primary: #212529;
  --bg-primary: #ffffff;
  --bg-secondary: #f8f9fa;
  --bg-hover: #eeeeee;
  --accent: #ff6600;
  --accent-hover: #cc5500;
  --border-color: #cc5500;
  --shadow-color: rgba(0, 0, 0, 0.1);
}

[data-theme="dark"] {
  --text-primary: #f0f0f0;
  --bg-primary: #121212;
  --bg-secondary: #1e1e1e;
  --bg-hover: #2c2c2c;
  --accent: #ff6600;
  --accent-hover: #cc5500;
  --border-color: #cc5500;
  --shadow-color: rgba(0, 0, 0, 0.4);
}

body {
  font-family: "Segoe UI", sans-serif;
  margin: 0;
  padding: 0;
  background-color: var(--bg-primary);
  color: var(--text-primary);
  transition: background-color 0.3s ease, color 0.3s ease;
}

.physio-container {
  background-color: var(--bg-primary);
  min-height: 100vh;
  transition: background-color 0.3s ease;
}

h2 {
  margin-bottom: 1.5rem;
  color: var(--accent);
  text-align: center;
}

.physio-header {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 1.5rem;
  margin-bottom: 2rem;
}

.chart-container {
  display: flex;
  justify-content: center;
  width: 100%;
  color: var(--text-primary);
}
```

```
.search-container {
  display: flex;
  gap: 1rem;
  margin-bottom: 1rem;
}

.search-input {
  flex: 1;
  height: 44px;
  padding: 0 0.75rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background-color: var(--bg-primary);
  color: var(--text-primary);
  font-size: 1rem;
  text-transform: uppercase;
  transition: background-color 0.3s ease, color 0.3s ease;
}

.search-button {
  height: 44px;
  padding: 0 1.25rem;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  background-color: var(--accent);
  color: white;
  font-size: 1rem;
  font-weight: 500;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.search-button:hover {
  background-color: var(--accent-hover);
}

.card {
  background-color: var(--bg-secondary);
  border-radius: 10px;
  padding: 1.5rem;
  box-shadow: 0 4px 6px var(--shadow-color);
  border: 1px solid var(--border-color);
  width: 100%;
  max-width: 100%;
  transition: background-color 0.3s ease, box-shadow 0.3s ease;
}

.piechart {
  max-width: 300px;
}

.user-details {
  margin-top: 2rem;
  width: 100%;
}

.section-title {
  color: var(--accent);
  margin-bottom: 1rem;
}

.table-responsive {
  width: 100%;
  overflow-x: auto;
}

.user-table {
  width: 100%;
  min-width: 600px;
  border-collapse: collapse;
}

.user-table th,
```

```
.user-table td {
  padding: 1rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
  color: var(--text-primary);
  transition: color 0.3s ease, background-color 0.3s ease;
}

.user-table th {
  background-color: var(--bg-hover);
  font-weight: bold;
}

.btn {
  padding: 0.5rem 1rem;
  background-color: var(--accent);
  margin: 1%;
  color: white;
  border: 1px solid var(--border-color);
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.3s ease;

  white-space: nowrap;
  min-width: 80px;
  text-align: center;
}

.content-container {
  padding: 2rem;
  text-align: center;
}

.loading-spinner,
.error-message {
  font-size: 1.2rem;
  color: var(--accent);
  font-weight: bold;
}

@media (max-width: 768px) {
  .search-section {
    flex-direction: column;
  }
  .search-input,
  .search-button {
    width: 100%;
  }
  .user-table {
    font-size: 14px;
    min-width: 500px;
  }
}

@media (max-width: 600px) {
  .search-container {
    flex-direction: column;
  }
  .search-input,
  .search-button {
    width: 100%;
  }
  .search-button {
    margin-top: 0.5rem;
  }
}

@media (max-width: 480px) {
  .user-table th,
  .user-table td {
    padding: 0.75rem;
  }
  .btn {
```

```
    width: 100%;
    text-align: center;
  }
}
```

File: src\dashboard\physio\PhysioPlanAssign.css

```
/* Base Variables for Light/Dark Mode */
:root {
  --primary-color: #cc5500;
  --primary-hover: #b84d00;
  --secondary-color: #6c757d;
  --success-color: #28a745;
  --danger-color: #dc3545;
  --warning-color: #ffc107;
  --info-color: #17a2b8;

  --bg-primary: #ffffff;
  --bg-secondary: #f8f9fa;
  --bg-tertiary: #e9ecef;
  --text-primary: #212529;
  --text-secondary: #6c757d;
  --text-muted: #868e96;

  --border-color: #dee2e6;
  --border-primary: var(--primary-color);
  --shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
  --shadow-lg: 0 0.5rem 1rem rgba(0, 0, 0, 0.15);

  --border-radius: 0.375rem;
  --border-radius-lg: 0.5rem;
  --transition: all 0.15s ease-in-out;
}

/* Dark Mode Variables */
@media (prefers-color-scheme: dark) {
  :root {
    --bg-primary: #1a1a1a;
    --bg-secondary: #2d2d2d;
    --bg-tertiary: #404040;
    --text-primary: #ffffff;
    --text-secondary: #b3b3b3;
    --text-muted: #999999;
    --border-color: #404040;
    --shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.3);
    --shadow-lg: 0 0.5rem 1rem rgba(0, 0, 0, 0.4);
  }
}

/* Dark mode class for manual toggle */
.dark-mode {
  --bg-primary: #1a1a1a;
  --bg-secondary: #2d2d2d;
  --bg-tertiary: #404040;
  --text-primary: #ffffff;
  --text-secondary: #b3b3b3;
  --text-muted: #999999;
  --border-color: #404040;
  --shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.3);
  --shadow-lg: 0 0.5rem 1rem rgba(0, 0, 0, 0.4);
}

/* Base Styles */
* {
  box-sizing: border-box;
}

.physio-plan-container {
  min-height: 100vh;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  padding: 1rem;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", sans-serif;
}
```

```
.main-content {
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  gap: 1.5rem;
}

.page-title {
  font-size: 2rem;
  font-weight: 700;
  color: var(--text-primary);
  margin: 0 0 2rem 0;
  text-align: left;
}

/* Card Styles */
.card {
  background-color: var(--bg-primary);
  border: 1px solid var(--border-primary);
  border-radius: var(--border-radius-lg);
  box-shadow: var(--shadow);
  overflow: hidden;
  transition: var(--transition);
}

.card:hover {
  box-shadow: var(--shadow-lg);
}

.card-header {
  background-color: var(--bg-secondary);
  border-bottom: 1px solid var(--border-color);
  padding: 1rem 1.5rem;
}

.card-title {
  font-size: 1.25rem;
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
  display: flex;
  align-items: center;
  gap: 0.5rem;
}

.card-content {
  padding: 1.5rem;
}

.icon {
  font-size: 1.1em;
}

/* Form Styles */
.form-input,
.form-select {
  width: 100%;
  padding: 0.75rem;
  font-size: 1rem;
  border: 1px solid var(--border-primary);
  border-radius: var(--border-radius);
  background-color: var(--bg-primary);
  color: var(--text-primary);
  transition: var(--transition);
}

.form-input:focus,
.form-select:focus {
  outline: none;
  border-color: var(--primary-hover);
  box-shadow: 0 0 0 0.2rem rgba(204, 85, 0, 0.25);
```

```
}

.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
  color: var(--text-secondary);
}

/* Button Styles */
.btn {
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 0.5rem;
  padding: 0.5rem 1rem;
  font-size: 0.875rem;
  font-weight: 500;
  border: 1px solid transparent;
  border-radius: var(--border-radius);
  cursor: pointer;
  transition: var(--transition);
  text-decoration: none;
  white-space: nowrap;
}

.btn:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

.btn-primary {
  background-color: var(--primary-color);
  color: white;
  border-color: var(--primary-color);
}

.btn-primary:hover:not(:disabled) {
  background-color: var(--primary-hover);
  border-color: var(--primary-hover);
}

.btn-outline {
  background-color: transparent;
  color: var(--primary-color);
  border-color: var(--primary-color);
}

.btn-outline:hover:not(:disabled) {
  background-color: var(--primary-color);
  color: white;
}

.btn-danger {
  background-color: var(--danger-color);
  color: white;
  border-color: var(--danger-color);
}

.btn-danger:hover:not(:disabled) {
  background-color: #c82333;
  border-color: #bd2130;
}

.btn-warning {
  background-color: var(--warning-color);
  color: #212529;
  border-color: var(--warning-color);
}
```

```
.btn-warning:hover:not(:disabled) {
  background-color: #e0a800;
  border-color: #d39e00;
}

.btn-secondary {
  background-color: var(--secondary-color);
  color: white;
  border-color: var(--secondary-color);
}

.btn-secondary:hover:not(:disabled) {
  background-color: #5a6268;
  border-color: #545b62;
}

.btn-large {
  padding: 0.75rem 1.5rem;
  font-size: 1rem;
}

/* Search Form */
.search-form {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

.input-group {
  display: flex;
  gap: 1rem;
  align-items: end;
}

.input-group .form-input {
  flex: 1;
}

/* Client Details */
.client-details {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 1.5rem;
}

.detail-item label {
  font-size: 0.875rem;
  font-weight: 500;
  color: var(--text-muted);
  margin-bottom: 0.25rem;
}

.detail-item p {
  font-weight: 600;
  color: var(--text-primary);
  margin: 0;
}

/* Date Range Inputs */
.date-range-inputs {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 1rem;
  margin-bottom: 1.5rem;
}

.date-input-group {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
}
```

```
/* Warning Message */
.warning-message {
  background-color: #fff3cd;
  border: 1px solid #ffeaa7;
  color: #856404;
  padding: 1rem;
  border-radius: var(--border-radius);
  margin-bottom: 1.5rem;
  text-align: center;
}

@media (prefers-color-scheme: dark) {
  .warning-message {
    background-color: #332701;
    border-color: #664d03;
    color: #ffda6a;
  }
}

.dark-mode .warning-message {
  background-color: #332701;
  border-color: #664d03;
  color: #ffda6a;
}

/* Date Selection */
.date-selection {
  margin-top: 1.5rem;
}

.select-group {
  margin-bottom: 1rem;
}

.select-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
  color: var(--text-secondary);
}

.selected-dates {
  margin-top: 1rem;
}

.selected-dates label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
  color: var(--text-secondary);
}

.selected-dates-list {
  display: flex;
  flex-wrap: wrap;
  gap: 0.5rem;
}

.selected-date-badge {
  position: relative;
  display: inline-flex;
  align-items: center;
  padding: 0.5rem 1rem;
  background-color: transparent;
  border: 1px solid var(--primary-color);
  border-radius: var(--border-radius-lg);
  color: var(--primary-color);
  font-weight: 500;
  font-size: 0.875rem;
}

.remove-date-btn {
  position: absolute;
```

```
  top: -6px;
  right: -6px;
  width: 20px;
  height: 20px;
  background-color: var(--primary-color);
  color: white;
  border: none;
  border-radius: 50%;
  font-size: 14px;
  font-weight: bold;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: var(--transition);
}

.remove-date-btn:hover {
  background-color: var(--primary-hover);
}

/* Table Styles */
.table-container {
  overflow-x: auto;
  margin: 1rem 0;
}

.responsive-table {
  width: 100%;
  border-collapse: collapse;
  background-color: var(--bg-primary);
  border-radius: var(--border-radius);
  overflow: hidden;
  box-shadow: var(--shadow);
}

.responsive-table th,
.responsive-table td {
  padding: 1rem;
  text-align: left;
  border-bottom: 1px solid var(--border-color);
}

.responsive-table th {
  background-color: var(--bg-secondary);
  font-weight: 600;
  color: var(--text-primary);
}

.responsive-table tr:hover {
  background-color: var(--bg-tertiary);
}

.action-buttons {
  display: flex;
  gap: 0.5rem;
  flex-wrap: wrap;
}

/* Assigned Dates Section */
.assigned-dates-section {
  margin-top: 2rem;
}

.assigned-dates-section h4 {
  font-size: 1.25rem;
  font-weight: 600;
  color: var(--text-primary);
  margin-bottom: 1rem;
}

.confirm-section {
  display: flex;
```

```
  justify-content: center;
  margin-top: 2rem;
}

.previous-plans-button {
  display: flex;
  justify-content: center;
  margin-top: 1rem;
}

/* Modal Styles */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
  padding: 1rem;
}

.modal-content {
  background-color: var(--bg-primary);
  border-radius: var(--border-radius-lg);
  box-shadow: var(--shadow-lg);
  width: 100%;
  max-width: 600px;
  max-height: 90vh;
  overflow-y: auto;
  position: relative;
}

.modal-large {
  max-width: 900px;
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1.5rem;
  border-bottom: 1px solid var(--border-color);
  background-color: var(--bg-secondary);
}

.modal-header h3 {
  margin: 0;
  font-size: 1.25rem;
  font-weight: 600;
  color: var(--text-primary);
}

.modal-close-btn {
  background: none;
  border: none;
  font-size: 1.5rem;
  color: var(--text-secondary);
  cursor: pointer;
  padding: 0;
  width: 30px;
  height: 30px;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: var(--border-radius);
  transition: var(--transition);
}

.modal-close-btn:hover {
```

```
  background-color: var(--bg-tertiary);
  color: var(--text-primary);
}

.modal-body {
  padding: 1.5rem;
}

.modal-footer {
  display: flex;
  justify-content: center;
  padding-top: 1rem;
  border-top: 1px solid var(--border-color);
  margin-top: 1.5rem;
}

/* Exercise Inputs */
.exercise-inputs {
  margin-top: 1rem;
}

.input-row {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  gap: 1rem;
  margin-bottom: 1rem;
}

/* Assigned Exercises */
.assigned-exercises {
  margin-top: 1.5rem;
}

.assigned-exercises h5 {
  font-size: 1.125rem;
  font-weight: 600;
  color: var(--text-primary);
  margin-bottom: 1rem;
}

/* Exercise Cards */
.exercises-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  gap: 1rem;
}

.exercise-card {
  background-color: var(--bg-secondary);
  border: 1px solid var(--border-color);
  border-radius: var(--border-radius);
  padding: 1rem;
  transition: var(--transition);
}

.exercise-card:hover {
  box-shadow: var(--shadow);
}

.exercise-card h6 {
  font-size: 1rem;
  font-weight: 600;
  color: var(--text-primary);
  margin: 0 0 0.5rem 0;
}

.exercise-card p {
  font-size: 0.875rem;
  color: var(--text-secondary);
  margin: 0;
}

/* Batch View */
```

```
.batch-view {
  display: flex;
  flex-direction: column;
  gap: 1.5rem;
}

.day-card {
  border: 1px solid var(--border-primary);
  border-radius: var(--border-radius);
  padding: 1rem;
  background-color: var(--bg-secondary);
}

.day-title {
  color: var(--primary-color);
  font-size: 1.125rem;
  font-weight: 600;
  margin: 0 0 1rem 0;
}

/* Previous Plans */
.plans-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
  gap: 1.5rem;
  margin-bottom: 2rem;
}

.plan-card {
  border: 1px solid var(--border-primary);
  border-radius: var(--border-radius);
  padding: 1rem;
  background-color: var(--bg-primary);
  box-shadow: var(--shadow);
  transition: var(--transition);
}

.plan-card:hover {
  box-shadow: var(--shadow-lg);
}

.plan-header h5 {
  color: var(--primary-color);
  font-size: 1rem;
  font-weight: 600;
  margin: 0 0 1rem 0;
}

.plan-details {
  margin-bottom: 1rem;
}

.detail-row {
  display: flex;
  justify-content: space-between;
  margin-bottom: 0.5rem;
  font-size: 0.875rem;
}

.detail-row .label {
  font-weight: 500;
  color: var(--text-secondary);
}

.plan-actions {
  display: flex;
  gap: 0.5rem;
}

.close-section {
  display: flex;
  justify-content: center;
}
```

```
/* Delete Confirmation */
.delete-confirmation {
  text-align: center;
}

.delete-confirmation p {
  margin-bottom: 1rem;
  color: var(--text-primary);
}

.delete-confirmation-buttons {
  display: flex;
  justify-content: center;
  gap: 1rem;
}

/* Responsive Design */
@media (max-width: 768px) {
  .physio-plan-container {
    padding: 0.5rem;
  }

  .page-title {
    font-size: 1.5rem;
    margin-bottom: 1rem;
  }

  .card-content {
    padding: 1rem;
  }

  .input-group {
    flex-direction: column;
    align-items: stretch;
  }

  .date-range-inputs {
    grid-template-columns: 1fr;
  }

  .client-details {
    grid-template-columns: 1fr;
    gap: 1rem;
  }

  .action-buttons {
    flex-direction: column;
  }

  .action-buttons .btn {
    width: 100%;
  }

  .input-row {
    grid-template-columns: 1fr;
  }

  .exercises-grid {
    grid-template-columns: 1fr;
  }

  .plans-grid {
    grid-template-columns: 1fr;
  }

  .plan-actions {
    flex-direction: column;
  }

  .plan-actions .btn {
    width: 100%;
  }
```

```
  /* Mobile Table */
  .responsive-table {
    border: 0;
  }

  .responsive-table thead {
    display: none;
  }

  .responsive-table tr {
    border-bottom: 3px solid var(--border-color);
    display: block;
    margin-bottom: 1rem;
    padding: 1rem;
    background-color: var(--bg-primary);
    border-radius: var(--border-radius);
    box-shadow: var(--shadow);
  }

  .responsive-table td {
    border: none;
    display: block;
    padding: 0.5rem 0;
    text-align: right;
    border-bottom: 1px solid var(--border-color);
  }

  .responsive-table td:last-child {
    border-bottom: none;
  }

  .responsive-table td:before {
    content: attr(data-label) ": ";
    float: left;
    font-weight: 600;
    color: var(--text-secondary);
  }

  .modal-content {
    margin: 0.5rem;
    max-width: calc(100vw - 1rem);
  }

  .modal-header {
    padding: 1rem;
  }

  .modal-body {
    padding: 1rem;
  }
}

@media (max-width: 480px) {
  .selected-dates-list {
    flex-direction: column;
  }

  .selected-date-badge {
    width: 100%;
    justify-content: center;
  }

  .delete-confirmation-buttons {
    flex-direction: column;
  }

  .delete-confirmation-buttons .btn {
    width: 100%;
  }
}

/* Print Styles */
```

```
@media print {
  .modal-overlay,
  .btn,
  .modal-close-btn {
    display: none !important;
  }

  .physio-plan-container {
    background: white !important;
    color: black !important;
  }

  .card {
    border: 1px solid #000 !important;
    box-shadow: none !important;
  }
}

/* High Contrast Mode */
@media (prefers-contrast: high) {
  :root {
    --border-color: #000000;
    --text-secondary: #000000;
  }

  .btn-outline {
    border-width: 2px;
  }

  .form-input,
  .form-select {
    border-width: 2px;
  }
}

/* Reduced Motion */
@media (prefers-reduced-motion: reduce) {
  * {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
  }
}
```

File: src\dashboard\physio\PhysioPlanAssign.js

```
"use client"

import { useState, useRef } from "react"
import { toast, ToastContainer } from "react-toastify"
import "react-toastify/dist/ReactToastify.css"
import axios from "axios"
import "./PhysioPlanAssign.css" // We'll create this CSS file

const PhysioPlanAssign = () => {
  const [mrn, setMrn] = useState("")
  const [clientData, setClientData] = useState(null)
  const [fromDate, setFromDate] = useState("")
  const [toDate, setToDate] = useState("")
  const [availableDates, setAvailableDates] = useState([])
  const [selectedDate, setSelectedDate] = useState("")
  const [fromDateTime, setFromDateTime] = useState("")
  const [toDateTime, setToDateTime] = useState("")
  const [assignedDates, setAssignedDates] = useState([])
  const [physioAssignedPlans, setPhysioAssignedPlans] = useState({})
  const [assignmentConfirmed, setAssignmentConfirmed] = useState(false)
  const [showPreviousCard, setShowPreviousCard] = useState(false)
  const scrollToRef = useRef(null)

  const deleteBatch = async (batchId) => {
    try {
      const res = await axios.get(`http://localhost:3001/physioAssignedPlans?mrn=${mrn}`)
      const batchToDelete = res.data.find((b) => b.batchId === batchId)
      if (batchToDelete?.id) {
```

```
      await axios.delete(`http://localhost:3001/physioAssignedPlans/${batchToDelete.id}`)
      setPhysioAssignedPlans((prev) => ({
        ...prev,
        [mrn]: prev[mrn].filter((b) => b.batchId !== batchId),
      }))
      toast.success("Batch deleted successfully!")
    }
  } catch (err) {
    console.error("? Delete batch failed", err)
    toast.error("Failed to delete batch.")
  }
}

const [showViewPopup, setShowViewPopup] = useState({
  visible: false,
  date: "",
  data: [],
})
const [selectedDates, setSelectedDates] = useState([])
const [dateError, setDateError] = useState("")
const [showAssignModal, setShowAssignModal] = useState({
  visible: false,
  date: null,
})
const [selectedType, setSelectedType] = useState("")
const [selectedExercise, setSelectedExercise] = useState("")
const [exerciseInputs, setExerciseInputs] = useState({})
const [assignedExercises, setAssignedExercises] = useState([])

const exerciseList = {
  Yoga: ["Diaphragmatic breathing", "Ujjay breathing", "Bhramari", "Nadi shodhana"],
  Resistance: ["Kettle bell swing", "Goblet squat", "Push ups", "Dumbbell row"],
}

const jsonForThisUser = {
  mrn,
  batchId: Date.now(),
  from: fromDateTime,
  to: toDateTime,
  assignedDates,
}

const confirmAssignment = async () => {
  if (!fromDateTime || !toDateTime || !mrn) {
    toast.error("Missing From/To date or MRN")
    return
  }
  if (!assignedDates?.length || !assignedDates.some((d) => d.exercises?.length)) {
    toast.error("Please assign at least one exercise")
    return
  }
  try {
    const createRes = await axios.post(`http://localhost:3001/physioAssignedPlans`, jsonForThisUser)
    jsonForThisUser.id = createRes.data.id

    setPhysioAssignedPlans((prev) => {
      const existingPlans = Array.isArray(prev[mrn]) ? prev[mrn] : []
      return {
        ...prev,
        [mrn]: [...existingPlans, jsonForThisUser],
      }
    })
    setAssignmentConfirmed(true)
    toast.success("? Assignment confirmed and saved!")
    setAssignedDates([])
    setSelectedDates([])
    setSelectedDate("")
    setFromDateTime("")
    setToDateTime("")
  } catch (error) {
    console.error("? Assignment Save Error:", error)
    toast.error("? Failed to confirm assignment")
  }
```

```
}

const deleteAssignedDate = (dateToDelete) => {
  showDeleteConfirmation(dateToDelete, async () => {
    const current = physioAssignedPlans[mrn]
    const updated = {
      ...current,
      assignedDates: current.assignedDates.filter((d) => d.date !== dateToDelete),
    }
    try {
      const res = await axios.get(`http://localhost:3001/physioAssignedPlans?mrn=${mrn}`)
      if (res.data.length > 0) {
        await axios.put(`http://localhost:3001/physioAssignedPlans/${res.data[0].id}`, updated)
      }
      setPhysioAssignedPlans((prev) => ({
        ...prev,
        [mrn]: updated,
      }))
      toast.success(" Plan deleted successfully!")
    } catch (err) {
      console.error(err)
      toast.error("? Failed to delete plan")
    }
  })
}

const showDeleteConfirmation = (dateToDelete, onConfirm) => {
  const toastId = toast(
    ({ closeToast }) => (
      <div className="delete-confirmation">
        <p>
          Are you sure you want to delete the plan for <strong>{dateToDelete}</strong>?
        </p>
        <div className="delete-confirmation-buttons">
          <button
            className="btn btn-danger"
            onClick={() => {
              toast.dismiss(toastId)
              onConfirm()
            }}
          >
            Yes, Delete
          </button>
          <button className="btn btn-secondary" onClick={() => toast.dismiss(toastId)}>
            Cancel
          </button>
        </div>
      </div>
    ),
    {
      autoClose: false,
      closeOnClick: false,
      closeButton: false,
    },
  )
}

const generateDatesBetween = (start, end) => {
  const dateArray = []
  const current = new Date(start)
  const endDate = new Date(end)
  while (current <= endDate) {
    dateArray.push(current.toISOString().split("T")[0])
    current.setDate(current.getDate() + 1)
  }
  return dateArray
}

const handleDateRangeChange = () => {
  if (!fromDateTime || !toDateTime) {
    setDateError("Please select both From and To dates.")
    return
  }
```

```
  if (new Date(toDateTime) < new Date(fromDateTime)) {
    setDateError("'To Date' must be after 'From Date'")
    return
  }
  const allDates = generateDatesBetween(fromDateTime, toDateTime)
  setAvailableDates(allDates)
  setSelectedDate("")
  setDateError("")
}

const handleSearch = async () => {
  try {
    const clientRes = await axios.get(`http://localhost:3001/physioClients?mrn=${mrn}`)
    const planRes = await axios.get(`http://localhost:3001/physioAssignedPlans?mrn=${mrn}`)
    if (planRes.data.length > 0) {
      setPhysioAssignedPlans((prev) => ({
        ...prev,
        [mrn]: planRes.data,
      }))
    }
    if (clientRes.data.length > 0) {
      setClientData(clientRes.data[0])
    }
    if (planRes.data.length > 0) {
      setPhysioAssignedPlans((prev) => ({
        ...prev,
        [mrn]: planRes.data[0],
      }))
    }
  } catch (error) {
    toast.error("Error fetching client or assignment data")
    console.error(error)
  }
}

return (
  <div className="physio-plan-container">
    <ToastContainer position="top-center" autoClose={2000} />

    <div className="main-content">
      <h1 className="page-title">Physio Plan Assignment</h1>

      {/* MRN Search Card */}
      <div className="card search-card">
        <div className="card-header">
          <h3 className="card-title">
            <span className="icon">?</span>
            Search Patient by MRN
          </h3>
        </div>
        <div className="card-content">
          <div className="search-form">
            <div className="input-group">
              <input
                type="text"
                placeholder="Enter MRN Number"
                className="form-input"
                value={mrn}
                onChange={(e) => setMrn(e.target.value.toUpperCase())}
                onKeyDown={(e) => {
                  if (e.key === "Enter") {
                    e.preventDefault()
                    handleSearch()
                  }
                }}
              />
              <button className="btn btn-primary" onClick={handleSearch}>
                <span className="icon">?</span>
                Search
              </button>
            </div>
          </div>
        </div>
```

```
</div>

{!assignmentConfirmed && (
  <>
    {/* Client Data Display */}
    {clientData && (
      <div className="card client-card">
        <div className="card-header">
          <h3 className="card-title">
            <span className="icon">?</span>
            Client Details
          </h3>
        </div>
        <div className="card-content">
          <div className="client-details">
            <div className="detail-item">
              <label>Name</label>
              <p>{clientData.name}</p>
            </div>
            <div className="detail-item">
              <label>Age</label>
              <p>{clientData.age}</p>
            </div>
            <div className="detail-item">
              <label>Gender</label>
              <p>{clientData.gender}</p>
            </div>
            <div className="detail-item">
              <label>Weight</label>
              <p>{clientData.weight}</p>
            </div>
            <div className="detail-item">
              <label>Issue</label>
              <p>{clientData.condition || "N/A"}</p>
            </div>
          </div>
        </div>
      </div>
    )}

    {/* Date Range Picker */}
    {clientData && (
      <div className="card date-range-card">
        <div className="card-header">
          <h3 className="card-title">
            <span className="icon">?</span>
            Select Date Range
          </h3>
        </div>
        <div className="card-content">
          {/* Warning */}
          {(!fromDateTime || !toDateTime || dateError) && (
            <div className="warning-message">
              <p>
                {dateError ? (
                  dateError
                ) : (
                  <>
                    Please select a <strong>From</strong> and <strong>To</strong> date range to activate
                    plan assignment section.
                  </>
                )}
              </p>
            </div>
          )}

          {/* Date Range Pickers */}
          <div className="date-range-inputs">
            <div className="date-input-group">
              <label htmlFor="from-date">From Date</label>
              <input
                id="from-date"
                type="date"
```

```
      className="form-input"
      value={fromDateTime}
      onChange={(e) => setFromDateTime(e.target.value)}
      min={new Date().toISOString().split("T")[0]}
    />
  </div>
  <div className="date-input-group">
    <label htmlFor="to-date">To Date</label>
    <input
      id="to-date"
      type="date"
      className="form-input"
      value={toDateTime}
      onChange={(e) => setToDateTime(e.target.value)}
      min={fromDateTime || new Date().toISOString().split("T")[0]}
    />
  </div>
</div>

<button className="btn btn-primary" onClick={handleDateRangeChange}>
  Generate Dates
</button>

{/* Date Selection */}
{availableDates.length > 0 && (
  <div className="date-selection">
    <div className="select-group">
      <label>Select Date to Assign Plan:</label>
      <select
        className="form-select"
        value={selectedDate}
        onChange={(e) => {
          const newDate = e.target.value
          setSelectedDate(newDate)
          if (!selectedDates.includes(newDate)) {
            setSelectedDates((prev) => [...prev, newDate])
          }
          const alreadyExists = assignedDates.some((entry) => entry.date === newDate)
          if (!alreadyExists && newDate) {
            setAssignedDates((prev) => [...prev, { date: newDate, exercises: [] }])
          }
          setTimeout(() => {
            scrollToRef.current?.scrollIntoView({
              behavior: "smooth",
            })
          }, 100)
        }}
      >
        <option value="">-- Select Date --</option>
        {availableDates.map((date, idx) => (
          <option key={idx} value={date}>
            {date}
          </option>
        ))}
      </select>
    </div>

    {selectedDates.length > 0 && (
      <div className="selected-dates">
        <label>Selected Dates:</label>
        <div className="selected-dates-list">
          {selectedDates.map((date, index) => (
            <div key={index} className="selected-date-badge">
              <span>? {date}</span>
              <button
                className="remove-date-btn"
                onClick={() => {
                  setSelectedDates((prev) => prev.filter((d) => d !== date))
                  setAssignedDates((prev) => prev.filter((row) => row.date !== date))
                  if (selectedDate === date) setSelectedDate("")
                }}
                title="Remove"
              >
```

```
                ×
              </button>
            </div>
          ))}
        </div>
      </div>
    )}
  </div>
)}

{/* Assigned Dates Table */}
{(assignedDates.length > 0 || selectedDate) && (
  <div ref={scrollToRef} className="assigned-dates-section">
    <h4>Assigned Dates</h4>
    <div className="table-container">
      <table className="responsive-table">
        <thead>
          <tr>
            <th>Sr No.</th>
            <th>Date</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {assignedDates.map((item, idx) => (
            <tr key={idx}>
              <td data-label="Sr No.">{idx + 1}</td>
              <td data-label="Date">{item.date}</td>
              <td data-label="Actions">
                <div className="action-buttons">
                  {item.exercises && item.exercises.length > 0 ? (
                    <>
                      <button
                        className="btn btn-outline"
                        onClick={() =>
                          setShowViewPopup({
                            visible: true,
                            data: item.exercises,
                            date: item.date,
                          })
                        }
                      >
                        <span className="icon">??</span>
                        View
                      </button>
                      <button
                        className="btn btn-outline"
                        onClick={() => {
                          const dateEntry = assignedDates.find((entry) => entry.date === item.
                          setAssignedExercises(dateEntry?.exercises || [])
                          if (dateEntry?.exercises?.length) {
                            setSelectedType(dateEntry.exercises[0].type)
                          } else {
                            setSelectedType("")
                          }
                          setShowAssignModal({
                            visible: true,
                            date: item.date,
                          })
                        }}
                      >
                        <span className="icon">??</span>
                        Edit
                      </button>
                    </>
                  ) : (
                    <button
                      className="btn btn-warning"
                      onClick={() =>
                        setShowAssignModal({
                          visible: true,
                          date: item.date,
                        })
```

```
                              }
                            >
                              <span className="icon">?</span>
                              Assign Plan
                            </button>
                          )}
                        </div>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        )}

        {assignedDates.length > 0 && (
          <div className="confirm-section">
            <button className="btn btn-primary btn-large" onClick={confirmAssignment}>
              Confirm Assignment
            </button>
          </div>
        )}
      </div>
    </div>
  )}

  {physioAssignedPlans[mrn] && (
    <div className="previous-plans-button">
      <button className="btn btn-outline" onClick={() => setShowPreviousCard(true)}>
        <span className="icon">?</span>
        View Previous Assigned Plan
      </button>
    </div>
  )}
 </>
)}

{/* Assignment Modal */}
{showAssignModal.visible && (
  <div className="modal-overlay" onClick={() => setShowAssignModal({ visible: false, date: null })}>
    <div className="modal-content" onClick={(e) => e.stopPropagation()}>
      <div className="modal-header">
        <h3>Assign Plan for {showAssignModal.date}</h3>
        <button className="modal-close-btn" onClick={() => setShowAssignModal({ visible: false, date: nu
          ×
        </button>
      </div>

      <div className="modal-body">
        {/* Plan Type */}
        <div className="form-group">
          <label>Plan Type</label>
          <select
            className="form-select"
            value={selectedType}
            onChange={(e) => {
              setSelectedType(e.target.value)
              setSelectedExercise("")
              setExerciseInputs({})
            }}
          >
            <option value="">-- Select --</option>
            <option value="Yoga">Yoga</option>
            <option value="Resistance">Resistance Training</option>
          </select>
        </div>

        {/* Exercise List */}
        {selectedType && (
          <div className="form-group">
            <label>Exercise</label>
            <select
```

```
      className="form-select"
      value={selectedExercise}
      onChange={(e) => setSelectedExercise(e.target.value)}
    >
      <option value="">-- Select Exercise --</option>
      {exerciseList[selectedType].map((ex, idx) => (
        <option key={idx} value={ex}>
          {ex}
        </option>
      ))}
    </select>
  </div>
)}

{/* Dynamic Inputs */}
{selectedExercise && (
  <div className="exercise-inputs">
    {selectedType === "Yoga" ? (
      <div className="input-row">
        <div className="form-group">
          <label>Breaths per round</label>
          <input
            className="form-input"
            placeholder="Breaths per round"
            value={exerciseInputs.breaths || ""}
            onChange={(e) =>
              setExerciseInputs({
                ...exerciseInputs,
                breaths: e.target.value,
              })
            }
          />
        </div>
        <div className="form-group">
          <label>Rounds</label>
          <input
            className="form-input"
            placeholder="Rounds"
            value={exerciseInputs.rounds || ""}
            onChange={(e) =>
              setExerciseInputs({
                ...exerciseInputs,
                rounds: e.target.value,
              })
            }
          />
        </div>
      </div>
    ) : (
      <div className="input-row">
        <div className="form-group">
          <label>Sets</label>
          <input
            className="form-input"
            placeholder="Sets"
            value={exerciseInputs.sets || ""}
            onChange={(e) =>
              setExerciseInputs({
                ...exerciseInputs,
                sets: e.target.value,
              })
            }
          />
        </div>
        <div className="form-group">
          <label>Reps</label>
          <input
            className="form-input"
            placeholder="Reps"
            value={exerciseInputs.reps || ""}
            onChange={(e) =>
              setExerciseInputs({
                ...exerciseInputs,
```

```
                reps: e.target.value,
              })
            }
          />
        </div>
        <div className="form-group">
          <label>Weight (optional)</label>
          <input
            className="form-input"
            placeholder="Weight (optional)"
            value={exerciseInputs.weight || ""}
            onChange={(e) =>
              setExerciseInputs({
                ...exerciseInputs,
                weight: e.target.value,
              })
            }
          />
        </div>
      </div>
    )}
    <button
      className="btn btn-primary"
      onClick={() => {
        if (!selectedExercise) return
        setAssignedExercises((prev) => [
          ...prev,
          {
            type: selectedType,
            name: selectedExercise,
            ...exerciseInputs,
          },
        ])
        setSelectedExercise("")
        setExerciseInputs({})
      }}
    >
      <span className="icon">?</span>
      Add Exercise
    </button>
  </div>
)}

{/* Assigned Exercises Preview */}
{assignedExercises.length > 0 && (
  <div className="assigned-exercises">
    <h5>Assigned Exercises</h5>
    <div className="table-container">
      <table className="responsive-table">
        <thead>
          <tr>
            <th>Type</th>
            <th>Exercise</th>
            <th>Details</th>
          </tr>
        </thead>
        <tbody>
          {assignedExercises.map((item, idx) => (
            <tr key={idx}>
              <td data-label="Type">{item.type}</td>
              <td data-label="Exercise">{item.name}</td>
              <td data-label="Details">
                {item.type === "Yoga"
                  ? `${item.rounds || "-"} rounds × ${item.breaths || "-"} breaths`
                  : `${item.sets || "-"} sets × ${item.reps || "-"} reps${
                      item.weight ? ` (${item.weight} kg)` : ""
                    }`}
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
```

```
          </div>
        )}

        {/* Submit Button */}
        <div className="modal-footer">
          <button
            className="btn btn-primary btn-large"
            onClick={async () => {
              if (!assignedExercises.length) {
                toast.error("Please add at least one exercise")
                return
              }
              const updatedAssignedDates = assignedDates.filter((entry) => entry.date !== showAssignModa
              updatedAssignedDates.push({
                date: showAssignModal.date,
                exercises: assignedExercises,
              })

              const updatedPlan = {
                ...physioAssignedPlans[mrn],
                assignedDates: updatedAssignedDates,
              }

              try {
                await axios.put(`http://localhost:3001/physioAssignedPlans/${updatedPlan.id}`, updatedPl
                setAssignedDates(updatedAssignedDates)
                setPhysioAssignedPlans((prev) => ({
                  ...prev,
                  [mrn]: updatedPlan,
                }))
                toast.success("? Plan updated successfully!")
              } catch (err) {
                console.error("? Edit save failed:", err)
                toast.error("? Failed to update plan")
              }

              setShowAssignModal({ visible: false, date: null })
              setAssignedExercises([])
              setSelectedType("")
              setSelectedExercise("")
              setExerciseInputs({})
            }}
          >
            Submit Plan
          </button>
        </div>
      </div>
    </div>
  </div>
)}

{/* View Popup Modal */}
{showViewPopup.visible && (
  <div
    className="modal-overlay"
    onClick={() =>
      setShowViewPopup({
        visible: false,
        data: [],
        date: "",
        isBatch: false,
      })
    }
  >
    <div className="modal-content modal-large" onClick={(e) => e.stopPropagation()}>
      <div className="modal-header">
        <h3>Assigned Plan for {showViewPopup.date}</h3>
        <button
          className="modal-close-btn"
          onClick={() =>
            setShowViewPopup({
              visible: false,
              data: [],
```

```
              date: "",
              isBatch: false,
            })
          }
        >
          ×
        </button>
      </div>

      <div className="modal-body">
        {showViewPopup.isBatch ? (
          <div className="batch-view">
            {showViewPopup.data.map((day, idx) => (
              <div key={idx} className="day-card">
                <h5 className="day-title">? {day.date}</h5>
                <div className="exercises-grid">
                  {day.exercises.map((exercise, exIdx) => (
                    <div key={exIdx} className="exercise-card">
                      <h6>{exercise.name}</h6>
                      <p>
                        {exercise.type === "Yoga"
                          ? `${exercise.rounds || "-"} rounds × ${exercise.breaths || "-"} breaths`
                          : `${exercise.sets || "-"} sets × ${exercise.reps || "-"} reps${
                              exercise.weight ? ` (${exercise.weight}kg)` : ""
                            }`}
                      </p>
                    </div>
                  ))}
                </div>
              </div>
            ))}
          </div>
        ) : (
          <div className="exercises-grid">
            {showViewPopup.data.map((exercise, index) => (
              <div key={index} className="exercise-card">
                <h6>{exercise.name}</h6>
                <p>
                  {exercise.type === "Yoga"
                    ? `${exercise.rounds || "-"} rounds × ${exercise.breaths || "-"} breaths`
                    : `${exercise.sets || "-"} sets × ${exercise.reps || "-"} reps${
                        exercise.weight ? ` (${exercise.weight}kg)` : ""
                      }`}
                </p>
              </div>
            ))}
          </div>
        )}
      </div>
    </div>
  </div>
)}

{/* Previous Plans Card */}
{showPreviousCard && (
  <div className="card previous-plans-card">
    <div className="card-header">
      <h3 className="card-title">Previous Assigned Plans for {mrn}</h3>
    </div>
    <div className="card-content">
      <div className="plans-grid">
        {Array.isArray(physioAssignedPlans[mrn]) &&
          physioAssignedPlans[mrn].map((batch, batchIdx) => (
            <div key={batch.batchId || batchIdx} className="plan-card">
              <div className="plan-header">
                <h5>?? Plan Group {batchIdx + 1}</h5>
              </div>
              <div className="plan-details">
                <div className="detail-row">
                  <span className="label">From:</span>
                  <span>{batch.from}</span>
                </div>
                <div className="detail-row">
```

```
                        <span className="label">To:</span>
                        <span>{batch.to}</span>
                      </div>
                      <div className="detail-row">
                        <span className="label">Total Dates:</span>
                        <span>{batch.assignedDates.length}</span>
                      </div>
                    </div>
                    <div className="plan-actions">
                      <button
                        className="btn btn-outline"
                        onClick={() =>
                          setShowViewPopup({
                            visible: true,
                            date: `Plan Group ${batchIdx + 1}`,
                            data: batch.assignedDates,
                            isBatch: true,
                          })
                        }
                      >
                        <span className="icon">??</span>
                        View
                      </button>
                      <button className="btn btn-danger" onClick={() => deleteBatch(batch.batchId)}>
                        <span className="icon">??</span>
                        Delete
                      </button>
                    </div>
                  </div>
                ))}
            </div>
            <div className="close-section">
              <button className="btn btn-outline" onClick={() => setShowPreviousCard(false)}>
                Close
              </button>
            </div>
          </div>
        </div>
      )}
    </div>
  </div>
 )
}

export default PhysioPlanAssign
```

File: src\dashboard\physio\PhysiosAppointments.js

```
import React, { useEffect, useState } from "react";
import { XCircle } from "lucide-react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../master_admin/master_admin.css";

const PhysiosAppointments = () => {
  const [appointments, setAppointments] = useState([]);
  const [selectedNote, setSelectedNote] = useState(null);

  const API_URL = "http://localhost:8080/api/appointments";

  useEffect(() => {
    fetch(`${API_URL}/role/physio`)
      .then((res) => res.json())
      .then((data) => setAppointments(data))
      .catch(() => toast.error("Failed to fetch appointments"));
  }, []);

  const updateAppointment = (updated) => {
    setAppointments((prev) =>
      prev.map((a) => (a.id === updated.id ? updated : a))
    );
    fetch(`${API_URL}/${updated.id}`, {
      method: "PUT",
      headers: { "Content-Type": "application/json" },
```

```
    body: JSON.stringify(updated),
  }).catch(() => toast.error("Failed to update appointment"));
};

const handleStatusToggle = (id) => {
  const current = appointments.find((a) => a.id === id);
  if (!current) return;

  const nextStatus =
    current.status === "Pending"
      ? "Approved"
      : current.status === "Approved"
      ? "Completed"
      : "Pending";

  updateAppointment({ ...current, status: nextStatus });
  toast.info(`Status updated to: ${nextStatus}`);
};

const handleMeetingLinkChange = (id, link) => {
  const updated = appointments.find((a) => a.id === id);
  if (!updated) return;
  updateAppointment({ ...updated, meetingLink: link });
};

const handleGenerateLink = (id) => {
  const dummy = `https://zoom.us/j/${Math.floor(
    100000000 + Math.random() * 900000000
  )}`;
  handleMeetingLinkChange(id, dummy);
};

const handleDelete = (id) => {
  toast(
    () => (
      <div>
        <p>Confirm delete?</p>
        <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
          <button
            className="btn btn-primary"
            onClick={() => {
              fetch(`${API_URL}/${id}`, { method: "DELETE" })
                .then(() =>
                  setAppointments((prev) => prev.filter((a) => a.id !== id))
                )
                .then(() => toast.error("Appointment deleted."))
                .catch(() => toast.error("Failed to delete appointment"));
              toast.dismiss();
            }}
          >
            Confirm
          </button>
          <button
            className="btn btn-primary"
            onClick={() => {
              toast.dismiss();
              toast.info("Deletion cancelled.");
            }}
          >
            Cancel
          </button>
        </div>
      </div>
    ),
    {
      position: "top-center",
      autoClose: false,
      closeOnClick: false,
      draggable: false,
      closeButton: false,
    }
  );
};
```

```
return (
  <div className="dashboard-main">
    <ToastContainer position="top-center" autoClose={2000} />
    <h1>Physio Appointments</h1>

    {appointments.length === 0 ? (
      <p>No appointments found.</p>
    ) : (
      <div className="table-responsive">
        <table className="appointments-table">
          <thead>
            <tr>
              <th>MRN No.</th>
              <th>Patient Name</th>
              <th>Type</th>
              <th>Date</th>
              <th>Time</th>
              <th>Notes</th>
              <th>Meeting Link</th>
              <th>Status</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {appointments.map((a) => (
              <tr key={a.id}>
                <td>{a.mrn || "N/A"}</td>
                <td>{a.patientName || "Unknown"}</td>
                <td>
                  <span
                    className={`type-badge ${
                      a.appointmentType === "Online"
                        ? "badge-orange"
                        : "badge-blue"
                    }`}
                  >
                    {a.appointmentType || "N/A"}
                  </span>
                </td>
                <td>{a.date || "-"}</td>
                <td>{a.time || "-"}</td>
                <td>
                  <button
                    className="btn btn-primary"
                    onClick={() => setSelectedNote(a.notes || "No notes")}
                  >
                    View
                  </button>
                </td>
                <td>
                  {a.appointmentType === "Online" ? (
                    <div
                      style={{
                        display: "flex",
                        alignItems: "center",
                        gap: "8px",
                      }}
                    >
                      <input
                        type="text"
                        value={a.meetingLink || ""}
                        onChange={(e) =>
                          handleMeetingLinkChange(a.id, e.target.value)
                        }
                        placeholder="Enter or generate"
                        className="search-input"
                        style={{
                          width: "180px",
                          padding: "5px 8px",
                          fontSize: "14px",
                          border: "1px solid #ccc",
                          borderRadius: "6px",
```

```
                      }}
                    />
                    <button
                      className="btn btn-primary"
                      onClick={() => handleGenerateLink(a.id)}
                    >
                      Auto
                    </button>
                    <button
                      className="btn btn-primary"
                      style={{ backgroundColor: "#f44336", color: "#fff" }}
                      onClick={() => handleMeetingLinkChange(a.id, "")}
                    >
                      <XCircle size={16} />
                    </button>
                  </div>
                ) : (
                  "-"
                )}
              </td>
              <td>
                <span
                  className={`status-badge ${
                    a.status === "Completed"
                      ? "badge-green"
                      : a.status === "Approved"
                      ? "badge-blue"
                      : "badge-yellow"
                  }`}
                  onClick={() => handleStatusToggle(a.id)}
                  style={{ cursor: "pointer" }}
                >
                  {a.status || "Pending"}
                </span>
              </td>
              <td>
                <button
                  className="btn btn-primary"
                  onClick={() => handleDelete(a.id)}
                >
                  Delete
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  )}

  {selectedNote && (
    <div className="modal-overlay">
      <div className="modalsc-box" style={{ border: "1px solid #cc5500" }}>
        <h2>Full Notes</h2>
        <p>{selectedNote}</p>
        <div className="center-btn">
          <center>
            <button
              className="btn btn-primary"
              onClick={() => setSelectedNote(null)}
            >
              Close
            </button>
          </center>
        </div>
      </div>
    </div>
  )}
 </div>
 );
};

export default PhysiosAppointments;
```

File: src\dashboard\physio\profile.css

```
.mobile-margin {
  margin-right: 0;
}

@media (max-width: 768px) {
  .mobile-margin {
    margin-right: 1rem;
  }
}
```

File: src\dashboard\physio\profile.js

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { Chart } from "react-google-charts";
import { useTheme } from "../../ThemeProvider";
import TimePicker from "react-time-picker";
import "react-time-picker/dist/TimePicker.css";
import { Eye, Pencil, Trash2 } from "lucide-react";

const useIsMobile = () => {
  const [isMobile, setIsMobile] = useState(window.innerWidth <= 768);

  useEffect(() => {
    const handleResize = () => setIsMobile(window.innerWidth <= 768);
    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize", handleResize);
  }, []);

  return isMobile;
};
// Toast Message Component
const ToastMessage = ({ message, type = "success", duration = 3000 }) => {
  const [visible, setVisible] = useState(!!message);

  useEffect(() => {
    if (message) {
      setVisible(true);
      const timer = setTimeout(() => setVisible(false), duration);
      return () => clearTimeout(timer);
    }
  }, [message, duration]);

  if (!visible || !message) return null;

  return (
    <div
      className={`toast-message ${type}`}
      style={{
        position: "fixed",
        top: "20px",
        right: "20px",
        padding: "12px 20px",
        borderRadius: "8px",
        color: "white",
        fontWeight: "bold",
        transition: "opacity 0.5s ease-in-out",
        opacity: visible ? 1 : 0,
        zIndex: 1000,
        backgroundColor: type === "error" ? "#ff4d4d" : "#28a745",
        boxShadow: "0px 4px 10px rgba(0,0,0,0.2)",
        display: "flex",
        alignItems: "center",
        gap: "8px",
      }}
    >
      <i
        className={`fa ${
          type === "error" ? "fa-times-circle" : "fa-check-circle"
        }`}
        style={{ fontSize: "18px" }}
      ></i>
      {message}
```

```
    </div>
  );
};

const Profile = () => {
  const navigate = useNavigate();
  const { theme, toggleTheme } = useTheme();
  const [isPopupOpen, setIsPopupOpen] = useState(false);
  const isMobile = useIsMobile();
  const togglePopup = () => setIsPopupOpen(!isPopupOpen);
  const [allclientPopupOpen, setallclientPopupOpen] = useState(false);
  const toggleallclientPopup = () => setallclientPopupOpen(!allclientPopupOpen);

  const [selectedTime, setSelectedTime] = useState("10:00");

  // Toast state
  const [toast, setToast] = useState({
    visible: false,
    message: "",
    type: "success",
  });

  // Show toast message function
  const showToast = (message, type = "success") => {
    setToast({
      visible: true,
      message,
      type,
    });

    // Auto hide after 3 seconds
    setTimeout(() => {
      setToast({
        visible: false,
        message: "",
        type: "success",
      });
    }, 3000);
  };

  const [profileData, setProfileData] = useState(null);
  const fetchProfileData = async () => {
    try {
      const res = await fetch("http://localhost:5000/profile");
      const data = await res.json();
      setProfileData(data);
    } catch (error) {
      console.error("Error fetching profile:", error);
      showToast("Failed to load profile", "error");
    }
  };
  useEffect(() => {
    fetchProfileData();
  }, []);
  if (!profileData)
    return (
      <div className="p-8 text-center text-lg font-medium">
        Loading profile...
      </div>
    );

  const handleUpdateProfile = () => {
    showToast("Profile updated successfully!");
  };

  const handleLogout = () => {
    localStorage.removeItem("token");
    sessionStorage.clear();
    showToast("You have been logged out.");
    navigate("/login");
  };

  // Added custom button styles
```

```
const buttonStyle = {
  display: "block",
  margin: "1rem auto",
  padding: "0.5rem 1.5rem",
};

// Added card style for consistent spacing
const cardStyle = {
  padding: "1.5rem",
  margin: "1rem",
  borderRadius: "8px",
  boxShadow: "0 2px 10px rgba(0,0,0,0.1)",
};

// Added section style for consistent section spacing
const sectionStyle = {
  marginBottom: "2rem",
};

// Responsive styles
const responsiveRowStyle = {
  display: "flex",
  flexDirection: window.innerWidth <= 768 ? "column" : "row",
  gap: "1.5rem",
  marginBottom: "2rem",
};

return (
  <div
    className="profile"
    style={{
      marginRight: isMobile ? "1.2rem" : "0rem",
    }}
  >
    {/* Toast Message Component */}
    {toast.visible && (
      <ToastMessage message={toast.message} type={toast.type} />
    )}
    {/* Personal Info + Client Info */}
    <div className="row" style={responsiveRowStyle}>
      <div
        className="col card"
        style={{
          ...cardStyle,
          flex: 1,
          border: "0.1px solid #cc5500",
        }}
      >
        <center>
          <h3>Personal Information</h3>
        </center>
        <p>
          <strong>Name:</strong> {profileData.personalInfo.name}
        </p>
        <p>
          <strong>Email:</strong> {profileData.personalInfo.email}
        </p>
        <p>
          <strong>Phone:</strong> {profileData.personalInfo.phone}
        </p>
        <p>
          <strong>DOB:</strong> {profileData.personalInfo.dob}
        </p>
        <div style={{ textAlign: "center" }}>
          <button
            className="btn btn-primary"
            onClick={togglePopup}
            style={buttonStyle}
          >
            View More
          </button>
        </div>
      </div>
```

```
  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "1px solid #cc5500",
    }}
  >
    <center>
      <h3>Client Info</h3>
    </center>
    <p>
      <strong>Total Clients:</strong>{" "}
      {profileData.clientData.totalClients}
    </p>
    <p style={{ fontSize: "1.5em", color: "var(--accent)" }}>
      <strong>Random Number:</strong>{" "}
      {profileData.clientData.randomNumber}
    </p>
  </div>
</div>
{/* Performance & Analytics + Weekly Patient Flow side-by-side */}
<div className="row" style={responsiveRowStyle}>
  {/* Performance Chart */}
  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "0.1px solid #cc5500",
    }}
  >
    <center>
      <h3>Performance & Analytics</h3>
    </center>
    <Chart
      chartType="PieChart"
      data={[
        ["Metric", "Value"],
        ["Patients Treated", profileData.performanceData.patientsTreated],
        [
          "Remaining",
          profileData.performanceData.totalPatients -
            profileData.performanceData.patientsTreated,
        ],
      ]}
      options={{
        title: "Treatment Overview",
        titleTextStyle: {
          fontSize: 18,
          bold: true,
          textAlign: "right",
          color: theme === "dark" ? "#f8f9fa" : "#212529",
        },
        is3D: true,
        pieHole: 0.3,
        colors: ["#cc5500", "#ffb380"],
        legend: {
          position: "bottom",
          textStyle: {
            color: theme === "dark" ? "#f8f9fa" : "#212529",
            fontSize: 13,
          },
        },
        pieSliceText: "value",
        pieSliceTextStyle: {
          color: "#fff",
          fontSize: 14,
          bold: true,
        },
        slices: {
          0: { offset: 0.05 },
```

```
      },
      backgroundColor: "transparent",
      chartArea: { width: "80%", height: "80%" }, // Better responsive chart
    }}
    width={"100%"}
    height={"300px"}
  />
  <center>
    {" "}
    <p>
      <strong>Success Rate:</strong>{" "}
      {profileData.performanceData.successRate}
    </p>
  </center>
</div>

{/* Weekly Bar Chart */}
<div
  className="col card"
  style={{
    ...cardStyle,
    flex: 1,
    border: "1px solid #cc5500",
  }}
>
  <center>
    <h3>Weekly Patient Flow</h3>
  </center>
  <Chart
    chartType="BarChart"
    data={[
      ["Week", "Appointments", "Patients Treated"],
      ["Week 1", 10, profileData.weeklyMetrics.patientsPerWeek[0]],
      ["Week 2", 12, profileData.weeklyMetrics.patientsPerWeek[1]],
      ["Week 3", 14, profileData.weeklyMetrics.patientsPerWeek[2]],
      ["Week 4", 16, profileData.weeklyMetrics.patientsPerWeek[3]],
    ]}
    options={{
      title: "",
      backgroundColor: "transparent",
      titleTextStyle: {
        fontSize: 18,
        bold: true,
        color: theme === "dark" ? "#f8f9fa" : "#212529",
      },
      legend: {
        position: "bottom",
        textStyle: {
          color: theme === "dark" ? "#f8f9fa" : "#212529",
          fontSize: 13,
        },
      },
      chartArea: { width: "60%" },
      hAxis: {
        title: "Count",
        minValue: 0,
        textStyle: {
          color: theme === "dark" ? "#f8f9fa" : "#212529",
        },
        titleTextStyle: {
          color: theme === "dark" ? "#f8f9fa" : "#212529",
          bold: true,
        },
        gridlines: {
          color: theme === "dark" ? "#444" : "#ccc",
        },
      },
      vAxis: {
        title: "Week",
        textStyle: {
          color: theme === "dark" ? "#f8f9fa" : "#212529",
        },
        titleTextStyle: {
```

```
            color: theme === "dark" ? "#f8f9fa" : "#212529",
            bold: true,
          },
        },
        colors: ["#cc5500", "#ff884d"],
        bar: { groupWidth: "60%" },
      }}
      width={"100%"}
      height={"300px"}
    />
  </div>
</div>
<div className="section" style={sectionStyle}>
  <div className="section-header mb-4 text-[#cc5500]">
    <h2>Client Management</h2>
  </div>
  <div className="row" style={responsiveRowStyle}>
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "0.1px solid #cc5500",
      }}
    >
      <center>
        <h3>Active Clients</h3>
      </center>
      <div
        className="metric-display"
        style={{ textAlign: "center", margin: "1.5rem 0" }}
      >
        <span
          className="metric-number"
          style={{
            fontSize: "2.5rem",
            fontWeight: "bold",
            display: "block",
          }}
        >
          {profileData.clientManagement.activeClients}
        </span>
        <span
          className="metric-label"
          style={{ display: "block", color: "#666" }}
        >
          Active
        </span>
      </div>
      <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
        Currently enrolled in treatment plans
      </p>
      <div style={{ textAlign: "center" }}>
        <button
          className="btn btn-primary"
          style={buttonStyle}
          onClick={() => {
            toggleallclientPopup();
          }}
        >
          View All Active Clients
        </button>
      </div>
    </div>

    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "1px solid #cc5500",
      }}
    >
```

```
  <center>
    <h3>Past Clients</h3>
  </center>
  <div
    className="metric-display"
    style={{ textAlign: "center", margin: "1.5rem 0" }}
  >
    <span
      className="metric-number"
      style={{
        fontSize: "2.5rem",
        fontWeight: "bold",
        display: "block",
      }}
    >
      {profileData.clientManagement.pastClients}
    </span>
    <span
      className="metric-label"
      style={{ display: "block", color: "#666" }}
    >
      Past
    </span>
  </div>
  <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
    Completed treatment programs
  </p>
  <div style={{ textAlign: "center" }}>
    <button
      className="btn btn-primary"
      style={buttonStyle}
      onClick={() => {
        toggleallclientPopup();
      }}
    >
      View Treatment History
    </button>
  </div>
</div>

<div
  className="col card"
  style={{
    ...cardStyle,
    flex: 1,
    border: "1px solid #cc5500",
  }}
>
  <center>
    <h3>Pending Consultations</h3>
  </center>
  <div
    className="metric-display"
    style={{ textAlign: "center", margin: "1.5rem 0" }}
  >
    <span
      className="metric-number"
      style={{
        fontSize: "2.5rem",
        fontWeight: "bold",
        display: "block",
      }}
    >
      {profileData.clientManagement.pendingConsultations}
    </span>
    <span
      className="metric-label"
      style={{ display: "block", color: "#666" }}
    >
      Pending
    </span>
  </div>
  <p style={{ textAlign: "center", marginBottom: "1.5rem" }}>
```

```
        Awaiting initial assessment
      </p>
      <div style={{ textAlign: "center" }}>
        <button
          className="btn btn-primary"
          style={buttonStyle}
          onClick={() => {
            togglePopup();
          }}
        >
          Schedule Consultation
        </button>
      </div>
    </div>
  </div>
</div>
{/* NEW SECTION: Exercise Plans */}
<div className="section" style={sectionStyle}>
  <div className="section-header" style={{ marginBottom: "1rem" }}>
    <h2>Exercise Plans</h2>
  </div>
  <div className="row" style={responsiveRowStyle}>
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "0.1px solid #cc5500",
      }}
    >
      <center>
        <h3>Create New Plan</h3>
      </center>
      <div className="plan-form" style={{ marginTop: "1.5rem" }}>
        <select
          className="form-control"
          style={{
            marginBottom: "1rem",
            padding: "0.5rem",
            width: "100%",
          }}
        >
          <option>Select Client...</option>
          <option>Alice Smith</option>
          <option>Bob Johnson</option>
          <option>Carol Davis</option>
        </select>
        <select
          className="form-control"
          style={{
            marginBottom: "1.5rem",
            padding: "0.5rem",
            width: "100%",
          }}
        >
          <option>Select Template...</option>
          <option>Lower Back Rehabilitation</option>
          <option>Knee Strengthening</option>
          <option>Post-Surgery Recovery</option>
        </select>
        <div style={{ textAlign: "center" }}>
          <button
            className="btn btn-primary"
            style={buttonStyle}
            onClick={() => {
              togglePopup();
            }}
          >
            Create Custom Plan
          </button>
        </div>
      </div>
    </div>
```

```
<div
  className="col card"
  style={{
    ...cardStyle,
    flex: 1,
    border: "1px solid #cc5500",
  }}
>
  <center>
    <h3>Manage Existing Plans</h3>
  </center>
  <p style={{ margin: "1rem 0" }}>
    <strong>Total Active Plans:</strong>{" "}
    {profileData.exercisePlans.totalPlans}
  </p>
  <ul
    className="plan-list"
    style={{ listStyle: "none", padding: 0, margin: "1.5rem 0" }}
  >
    <li
      style={{
        margin: "0.5rem 0",
        display: "flex",
        justifyContent: "space-between",
      }}
    >
      Lower Back Rehabilitation{" "}
      <span
        className="badge"
        style={{
          background: "#cc5500",
          color: "white",
          padding: "0.25rem 0.5rem",
          borderRadius: "4px",
        }}
      >
        12 Clients
      </span>
    </li>
    <li
      style={{
        margin: "0.5rem 0",
        display: "flex",
        justifyContent: "space-between",
      }}
    >
      Knee Strengthening{" "}
      <span
        className="badge"
        style={{
          background: "#cc5500",
          color: "white",
          padding: "0.25rem 0.5rem",
          borderRadius: "4px",
        }}
      >
        8 Clients
      </span>
    </li>
    <li
      style={{
        margin: "0.5rem 0",
        display: "flex",
        justifyContent: "space-between",
      }}
    >
      Shoulder Mobility{" "}
      <span
        className="badge"
        style={{
          background: "#cc5500",
          color: "white",
```

```
            padding: "0.25rem 0.5rem",
            borderRadius: "4px",
          }}
        >
          14 Clients
        </span>
      </li>
    </ul>
    <div style={{ textAlign: "center" }}>
      <button
        className="btn btn-primary"
        style={buttonStyle}
        onClick={() => {
          togglePopup();
        }}
      >
        View All Plans
      </button>
    </div>
  </div>

  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "1px solid #cc5500",
    }}
  >
    <center>
      <h3>Plan Templates</h3>
    </center>
    <p style={{ margin: "1rem 0" }}>
      <strong>Available Templates:</strong>{" "}
      {profileData.exercisePlans.templates}
    </p>

    <table
      style={{
        width: "100%",
        marginTop: "1rem",
        borderSpacing: "0 0.5rem",
      }}
    >
      <tbody>
        {[
          "Lower Back Rehabilitation",
          "Knee Strengthening",
          "Post-Surgery Recovery",
        ].map((template, index) => (
          <tr key={index} style={{ backgroundColor: "transparent" }}>
            <td style={{ padding: "0.5rem 0" }}>{template}</td>
            <td style={{ textAlign: "right" }}>
              <button
                className="btn btn-primary"
                style={{ padding: "0.25rem 0.75rem" }}
                onClick={() =>
                  showToast(`Editing template: ${template}`)
                }
              >
                Edit
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
    <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
      <button
        className="btn btn-primary"
        style={buttonStyle}
        onClick={() => {
          togglePopup();
```

```
          }}
        >
          Create New Template
        </button>
      </div>
    </div>
  </div>
</div>
{/* NEW SECTION: Appointments */}
<div className="section" style={sectionStyle}>
  <div className="section-header" style={{ marginBottom: "1rem" }}>
    <h2>Appointments</h2>
  </div>

  {/* Row 1: Schedule Form and Calendar */}
  <div className="row" style={responsiveRowStyle}>
    <div
      className="col card"
      style={{
        ...cardStyle,
        flex: 1,
        border: "0.1px solid #cc5500",
      }}
    >
      <center>
        <h3>Schedule Consultation</h3>
      </center>
      <form className="consultation-form" style={{ width: "100%" }}>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="clientName">Client Name:</label>
          <input
            type="text"
            id="clientName"
            className="form-control"
            style={{
              width: "100%",
              padding: "0.5rem",
              marginTop: "0.5rem",
            }}
          />
        </div>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="date">Date:</label>
          <input
            type="date"
            id="date"
            className="form-control"
            style={{
              width: "100%",
              padding: "0.5rem",
              marginTop: "0.5rem",
            }}
          />
        </div>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="time">Time:</label>
          <input
            type="time"
            id="time"
            className="form-control"
            style={{
              width: "100%",
              padding: "0.5rem",
              marginTop: "0.5rem",
            }}
          />
        </div>
        <div style={{ marginBottom: "1rem" }}>
          <label htmlFor="type">Consultation Type:</label>
          <select
            id="type"
            className="form-control"
            style={{
```

```
            width: "100%",
            padding: "0.5rem",
            marginTop: "0.5rem",
          }}
        >
          <option>Initial Assessment</option>
          <option>Follow-up</option>
          <option>Treatment Session</option>
        </select>
      </div>
      <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
        <button
          type="button"
          className="btn btn-primary"
          style={buttonStyle}
          onClick={() =>
            showToast("Appointment scheduled successfully!")
          }
        >
          Schedule
        </button>
      </div>
    </form>
  </div>
</div>

{/* Row 2: Today's Appointments */}
<div className="row" style={responsiveRowStyle}>
  <div
    className="appointments-card card"
    style={{
      ...cardStyle,
      flex: 1,
      width: "100%",
      border: "0.1px solid #cc5500",
    }}
  >
    <center>
      <h3>Today's Appointments</h3>
    </center>
    <div style={{ overflowX: "auto" }}>
      {" "}
      {/* Makes table scrollable on small screens */}
      <table
        className="appointments-table"
        style={{
          width: "100%",
          borderCollapse: "separate",
          borderSpacing: "0 0.5rem",
          marginTop: "1rem",
        }}
      >
        <thead>
          <tr>
            <th
              style={{
                textAlign: "left",
                padding: "0.5rem",
                borderBottom: "1px solid #ddd",
              }}
            >
              Patient
            </th>
            <th
              style={{
                textAlign: "left",
                padding: "0.5rem",
                borderBottom: "1px solid #ddd",
              }}
            >
              Time
            </th>
            <th
```

```
      style={{
        textAlign: "left",
        padding: "0.5rem",
        borderBottom: "1px solid #ddd",
      }}
    >
      Status
    </th>
    <th
      style={{
        textAlign: "center",
        padding: "0.5rem",
        borderBottom: "1px solid #ddd",
      }}
    >
      Actions
    </th>
  </tr>
</thead>
<tbody>
  {profileData.todaysAppointments.map((appointment, index) => (
    <tr key={index}>
      <td style={{ padding: "0.75rem 0.5rem" }}>
        {appointment.patient}
      </td>
      <td style={{ padding: "0.75rem 0.5rem" }}>
        {appointment.time}
      </td>
      <td style={{ padding: "0.75rem 0.5rem" }}>
        <span
          className={`status-tag status-${appointment.status.toLowerCase()}`}
          style={{
            background:
              appointment.status === "Confirmed"
                ? "#28a745"
                : "#ffc107",
            color: "white",
            padding: "0.25rem 0.5rem",
            borderRadius: "4px",
            fontSize: "0.75rem",
          }}
        >
          {appointment.status}
        </span>
      </td>
      <td
        style={{
          textAlign: "center",
          padding: "0.75rem 0.5rem",
        }}
      >
        <div
          className="action-buttons"
          style={{
            display: "flex",
            flexDirection:
              window.innerWidth <= 500 ? "column" : "row",
            gap: "0.5rem",
            justifyContent: "center",
          }}
        >
          <button
            className="btn btn-primary"
            style={{
              padding: "0.25rem 0.75rem",
            }}
            onClick={() =>
              showToast(
                `Session started with ${appointment.patient}`
              )
            }
          >
            Start
```

```
                  </button>
                  <button
                    className="btn btn-success"
                    style={{
                      padding: "0.25rem 0.75rem",
                    }}
                    onClick={() =>
                      showToast(
                        `Rescheduling ${appointment.patient}'s appointment`
                      )
                    }
                  >
                    Reschedule
                  </button>
                </div>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>

    {/* Time Picker Component */}
    <div style={{ marginTop: "2rem" }}>
      <h4>Quick Time Selection</h4>
      <div
        style={{
          display: "flex",
          alignItems: "center",
          marginTop: "1rem",
          flexDirection: window.innerWidth <= 768 ? "column" : "row",
        }}
      >
        <label
          htmlFor="time"
          style={{
            marginRight: "1rem",
            marginBottom: window.innerWidth <= 768 ? "0.5rem" : 0,
          }}
        >
          Select Time:
        </label>
        <TimePicker
          id="time"
          onChange={(time) => {
            setSelectedTime(time);
            showToast(`Time selected: ${time}`);
          }}
          value={selectedTime}
          disableClock={true}
          clearIcon={null}
          className="form-control"
          format="hh:mm a"
        />
      </div>
    </div>
  </div>
</div>
<div
  className="row"
  style={{ display: "flex", gap: "1.5rem", marginBottom: "1.5rem" }}
>
  <div
    className="col card"
    style={{
      ...cardStyle,
      flex: 1,
      border: "1px solid #cc5500",
      overflow: "hidden",

      display: "flex",
      flexDirection: "column",
      maxHeight: "360px", // adjust as needed
```

```
      }}
    >
      <center>
        <h3>Upcoming Appointments</h3>
      </center>

      <div
        style={{
          overflowY: "auto",
          flex: 2,
          marginTop: "1rem",
          scrollbarColor: "#cc5500 transparent",
          scrollbarWidth: "thin",
        }}
      >
        <style>
          {`
  /* WebKit-based browsers */
  .col.card::-webkit-scrollbar {
    width: 8px;
  }
  .col.card::-webkit-scrollbar-track {
    background: transparent;
  }
  .col.card::-webkit-scrollbar-thumb {
    background-color: #cc5500;
    border-radius: 4px;
  }
`}
        </style>
        <table
          style={{
            width: "100%",
            borderSpacing: "0 0.75rem", // horizontal 0, vertical 0.75rem space between rows
            fontSize: "0.9rem",
            borderCollapse: "separate", // IMPORTANT: allows borderSpacing to work
          }}
        >
          <thead>
            <tr>
              <th
                style={{
                  borderBottom: "1px solid #ccc",
                  textAlign: "left",
                  padding: "8px",
                  backgroundColor: "var(--bg-primary)",
                  position: "sticky",
                  top: 0,
                  zIndex: 1,
                }}
              >
                Date
              </th>
              <th
                style={{
                  borderBottom: "1px solid #ccc",
                  textAlign: "left",
                  padding: "8px",
                  backgroundColor: "var(--bg-primary)",
                  position: "sticky",
                  top: 0,
                  zIndex: 1,
                }}
              >
                Patient
              </th>
              <th
                style={{
                  borderBottom: "1px solid #ccc",
                  textAlign: "left",
                  padding: "8px",
                  backgroundColor: "var(--bg-primary)",
                  position: "sticky",
```

```
                top: 0,
                zIndex: 1,
              }}
            >
              Time
            </th>
          </tr>
        </thead>
        <tbody>
          {profileData.appointments.map((appointment, index) => (
            <tr key={index}>
              <td
                style={{
                  padding: "8px",
                  borderBottom: "1px solid #eee",
                }}
              >
                {appointment.date}
              </td>
              <td
                style={{
                  padding: "8px",
                  borderBottom: "1px solid #eee",
                }}
              >
                {appointment.patient}
              </td>
              <td
                style={{
                  padding: "8px",
                  borderBottom: "1px solid #eee",
                }}
              >
                {appointment.time}
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  </div>
</div>{" "}
{isPopupOpen && (
  <div
    className="popup"
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      width: "100vw",
      height: "100vh",
      backgroundColor: "rgba(0, 0, 0, 0.5)",
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      zIndex: 1000,
    }}
  >
    <div
      className="popup-content card"
      style={{
        backgroundColor: "var(--bg-primary)",
        padding: "2rem",
        borderRadius: "8px",
        maxWidth: "600px",
        width: "90%",
        maxHeight: "90vh",
        overflow: "auto",
      }}
    >
      <h3>Physiotherapist Details</h3>
      <p>
```

```
        <strong>Specialization:</strong>{" "}
        {profileData.professionalInfo.specialization}
      </p>
      <p>
        <strong>Experience:</strong>{" "}
        {profileData.professionalInfo.experience}
      </p>
      <p>
        <strong>Certifications:</strong>{" "}
        {profileData.professionalInfo.certifications.join(", ")}
      </p>
      <Chart
        chartType="PieChart"
        data={[
          ["Metric", "Value"],
          [
            "Patients Treated",
            profileData.performanceData.patientsTreated,
          ],
          [
            "Remaining",
            profileData.performanceData.totalPatients -
              profileData.performanceData.patientsTreated,
          ],
        ]}
        options={{
          title: "Performance Insights",
          backgroundColor: "transparent",
          titleTextStyle: {
            color: theme === "dark" ? "white" : "#212529",
          },
          legend: {
            textStyle: {
              color: theme === "dark" ? "white" : "#212529",
            },
          },
          pieSliceTextStyle: {
            color: theme === "dark" ? "white" : "#212529",
          },
          tooltip: {
            textStyle: {
              color: theme === "dark" ? "white" : "#212529",
            },
          },
        }}
        width={"100%"}
        height={"300px"}
      />

      <div style={{ textAlign: "center" }}>
        <button
          className="btn btn-primary"
          onClick={togglePopup}
          style={buttonStyle}
        >
          Close
        </button>
      </div>
    </div>
  </div>
)}
{allclientPopupOpen && (
  <div
    className="popup"
    style={{
      position: "fixed",
      top: 0,
      left: 0,
      width: "100%",
      height: "100%",
      backgroundColor: "rgba(0, 0, 0, 0.5)",
      display: "flex",
      justifyContent: "center",
```

```
    alignItems: "center",
    zIndex: 1000,
  }}
>
  <div
    className="popup-content card"
    style={{
      backgroundColor: "var(--bg-primary)",
      padding: "2rem",
      top: "50%",
      left: "50%",
      borderRadius: "8px",
      border: "1px solid #cc5500",
      maxWidth: "700px",
      width: "95%",
      maxHeight: "90vh",
      overflow: "auto",
      color: "var(--text-primary)",
    }}
  >
    <h3 style={{ marginBottom: "1rem" }}>Upcoming Appointments</h3>

    <table style={{ width: "100%", borderCollapse: "collapse" }}>
      <thead>
        <tr style={{ backgroundColor: "#cc5500" }}>
          <th style={{ padding: "0.75rem", textAlign: "left" }}>
            Date
          </th>
          <th style={{ padding: "0.75rem", textAlign: "left" }}>
            Patient
          </th>
          <th style={{ padding: "0.75rem", textAlign: "left" }}>
            Time
          </th>
          <th style={{ padding: "0.75rem", textAlign: "left" }}>
            Status
          </th>
          <th style={{ padding: "0.75rem", textAlign: "center" }}>
            Actions
          </th>
        </tr>
      </thead>
      <tbody>
        {profileData.appointments.map((appt, index) => (
          <tr
            key={index}
            style={{ borderBottom: "1px solid var(--border-color)" }}
          >
            <td style={{ padding: "0.75rem" }}>{appt.date}</td>
            <td style={{ padding: "0.75rem" }}>{appt.patient}</td>
            <td style={{ padding: "0.75rem" }}>{appt.time}</td>
            <td style={{ padding: "0.75rem" }}>Scheduled</td>
            <td style={{ padding: "0.75rem", textAlign: "center" }}>
              <button
                className="btn btn-secondary"
                style={{ marginRight: "0.5rem" }}
                onClick={() => showToast(`Viewing ${appt.patient}`)}
              >
                <Eye size={16} />
              </button>
              <button
                className="btn btn-secondary"
                style={{ marginRight: "0.5rem" }}
                onClick={() => showToast(`Editing ${appt.patient}`)}
              >
                <Pencil size={16} />
              </button>
              <button
                className="btn btn-secondary"
                onClick={() => showToast(`Deleting ${appt.patient}`)}
              >
                <Trash2 size={16} />
              </button>
```

```
                  </td>
                </tr>
              ))}
            </tbody>
          </table>

          <div style={{ textAlign: "center", marginTop: "1.5rem" }}>
            <button
              className="btn btn-primary"
              onClick={toggleallclientPopup}
              style={{ padding: "0.5rem 1.5rem", fontSize: "1rem" }}
            >
              Close
            </button>
          </div>
        </div>
      </div>
    )}
  </div>
  );
};
export default Profile;
```

File: src\dashboard\physio\report.css

```
.uppercase-input {
  text-transform: uppercase;
  background-color: transparent;
}

.report-body {
  /* padding: 2rem; */
  background-color: var(--bg-primary, #f9f9f9);
  min-height: 100vh;
  color: var(--text-primary, #111);
}
.page-description {
  font-size: 16px;
  color: #555;
  margin-bottom: 20px;
  max-width: 600px;
  line-height: 1.5;
  text-align: justify;
}

.report-card {
  max-width: 700px;
  margin: auto;
  background: var(--bg-secondary, #fff);
  padding: 1.5rem;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}
.content-area h2 {
  margin-bottom: 1rem; /* equivalent to mb-4 */
}

.search-container {
  display: flex;
  gap: 1rem;
  margin-bottom: 1rem;
}

.search-container input {
  flex-grow: 1;
  border: 1px solid var(--border-color, #ccc);
  padding: 0.5rem;
  color: var(--text-primary, #111);
  background-color: var(--bg-secondary, #f9f9f9);
  font-size: 1rem;
}

.search-container button {
  background-color: #cc5500;
```

```
  color: white;
  border: none;
  padding: 0 1rem;
  cursor: pointer;
  border-radius: 5px;
}

.search-container button:hover {
  background-color: #a24300;
}

.remove-button {
  margin: 1rem auto 0;
  background-color: #e74c3c;
  color: white;
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  display: block; /* make it a block element so margin auto works */
}

.remove-button:hover {
  background-color: #c0392b;
}

.error-text {
  color: #e74c3c;
  margin-top: 1rem;
}
.report-card {
  border: 1px solid #cc5500;
  padding: 1.5rem;
  border-radius: 8px;
  background-color: var(--bg-secondary, #fff);
}
```

File: src\dashboard\physio\userDetails.css

```
/* Main layout and containers */
.content-container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
  font-family: "Roboto", Arial, sans-serif;
}

.fitness-journey {
  display: flex;
  flex-direction: column;
  gap: 20px;
}

/* Card styling */
.card {
  background-color: #ffffff;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  padding: 20px;
  margin-bottom: 20px;
  transition: transform 0.2s ease, box-shadow 0.2s ease;
}

/* Heading styles - all bold */
h2,
h3,
h4,
h5,
h6 {
  font-weight: bold;
}

/* Removed hover effect as requested */
```

```
.user-info-card {
  background-color: #f8f9fa;
  border-left: 4px solid #cc5500;
}

.day-card {
  position: relative;
}

.day-card h4 {
  color: #cc5500;
  margin-top: 0;
  font-size: 1.5rem; /* Increased from 1.2rem */
  border-bottom: 1px solid #e0e0e0;
  padding-bottom: 10px;
  font-weight: bold;
  text-align: center; /* Added center alignment */
  margin-bottom: 20px; /* Added margin bottom */
}

/* Day selector styling */
.day-selector {
  margin: 15px 0;
  display: flex;
  align-items: center;
  flex-wrap: wrap;
  gap: 10px;
}

.day-selector label {
  font-weight: 500;
  margin-right: 10px;
}

.day-selector select {
  padding: 8px 12px;
  border-radius: 4px;
  border: 1px solid #d1d1d1;
  background-color: #fff;
  font-size: 0.9rem;
  min-width: 120px;
  cursor: pointer;
}

.day-selector select:focus {
  outline: none;
  border-color: #cc5500;
  box-shadow: 0 0 0 2px rgba(204, 85, 0, 0.2);
}

/* Exercise sections */
.exercise-section {
  margin-bottom: 25px;
}

.exercise-section h5 {
  color: #ffffff; /* Changed from #333 to white for better visibility */
  font-size: 1.2rem; /* Increased from 1rem */
  font-weight: bold;
  margin-bottom: 15px; /* Increased from 10px */
  display: block;
  padding: 8px 12px;
  border-radius: 4px;
  border-left: 3px solid #cc5500;
  background-color: rgba(
    204,
    85,
    0,
    0.2
  ); /* Added semi-transparent background */
  text-shadow: 0px 0px 1px rgba(255, 255, 255, 0.5); /* Added text shadow */
}
```

```
/* Table styling */
.table-responsive {
  overflow-x: auto;
  margin-bottom: 15px;
}

.followup-table {
  width: 100%;
  border-collapse: collapse;
  font-size: 0.9rem;
}

.followup-table th {
  background-color: var(--bg-secondary);
  color:--text-primary;
  text-align: left;
  padding: 12px;
  font-weight: bold;
}

.followup-table td {
  padding: 12px;
  border-bottom: 1px solid #e0e0e0;
  text-align: left;
}

.followup-table tr:last-child td {
  border-bottom: none;
}

/* Status indicators */
.status-indicator {
  display: flex;
  align-items: center;
}

.status-dot {
  display: inline-block;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  margin-right: 8px;
}

.missed {
  color: #f44336;
  font-weight: 500;
}

.completed {
  color: #4caf50;
  font-weight: 500;
}

.notes-section {
  background-color: var(--bg-secondary);
  border-left: 4px solid #ffc107;
}

.notes-section h5 {
  color: #333;
  margin-top: 0;
  font-weight: bold;
}

.notes-section ul {
  padding-left: 0;
  margin-bottom: 0;
  list-style-type: none;
}

.notes-section li {
  margin-bottom: 8px;
```

```
  padding-left: 0;
}

.notes-section li:last-child {
  margin-bottom: 0;
}

/* Loading and error states */
.loading-spinner {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 200px;
  color: #cc5500;
  font-size: 1.2rem;
}

.error-message {
  background-color: #ffebee;
  color: #d32f2f;
  padding: 15px;
  border-radius: 4px;
  margin: 20px 0;
  text-align: center;
}

/* Responsive adjustments */
@media (max-width: 768px) {
  .content-container {
    padding: 15px;
  }

  .card {
    padding: 15px;
  }

  .followup-table th,
  .followup-table td {
    padding: 10px 8px;
    font-size: 0.85rem;
  }

  h2 {
    font-size: 1.5rem;
    font-weight: bold;
  }
}

@media (max-width: 576px) {
  .content-container {
    padding: 0; /* Removed padding to allow cards to reach the edge */
    width: 100%;
    max-width: 100%; /* Ensure container is full width */
  }

  .fitness-journey {
    width: 100%;
    gap: 10px; /* Reduced gap between elements */
  }

  .card {
    padding: 12px;
    width: 100%;
    margin-left: 0;
    margin-right: 0;
    border-radius: 0; /* Optional: Remove rounded corners on mobile for full-width feel */
    margin-bottom: 10px; /* Reduced margin between cards */
  }

  .day-card {
    width: 100%; /* Ensure full width */
    margin: 0; /* Remove any margin */
    border-radius: 0; /* Remove border radius for full-width look */
```

```
  }

  /* Center text in mobile view */
  .card h2,
  .card h4,
  .card h5,
  .exercise-section h5,
  .notes-section h5 {
    text-align: center;
  }

  .day-selector {
    flex-direction: column;
    align-items: flex-start;
    width: 100%;
  }

  .day-selector select {
    width: 100%;
  }

  .followup-table {
    font-size: 0.8rem;
  }

  .followup-table th,
  .followup-table td {
    padding: 8px 6px;
  }

  h2 {
    font-size: 1.3rem;
    font-weight: bold;
  }

  h4 {
    font-size: 1.1rem;
    font-weight: bold;
  }

  h5 {
    font-size: 1rem;
    font-weight: bold;
  }
}

/* Fix to ensure section headings are visible when toggling views */
.day-card .exercise-section {
  display: block !important;
  visibility: visible !important;
  opacity: 1 !important;
}

/* Ensure exercise section headings remain visible */
.day-card .exercise-section h5 {
  display: block !important;
  visibility: visible !important;
  opacity: 1 !important;
}

/* Animation for loading */
@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

.loading-spinner::before {
  content: "";
  display: inline-block;
```

```
  width: 20px;
  height: 20px;
  margin-right: 10px;
  border: 3px solid rgba(204, 85, 0, 0.2);
  border-top-color: #cc5500;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}
```

File: src\dashboard\physio\userDetails.js

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import axios from "axios";
import "./userDetails.css";

const UserDetails = () => {
  const { userId } = useParams();
  const [userData, setUserData] = useState({
    clientName: "",
    mrnId: "",
    days: [],
  });
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [viewMode, setViewMode] = useState(null); // will be set dynamically

  useEffect(() => {
    const fetchUserData = async () => {
      try {
        setLoading(true);

        const userResponse = await axios.get(
          `http://localhost:5000/users/${userId}`
        );

        const fitnessResponse = await axios.get(
          `http://localhost:5000/fitnessJourney?userId=${userId}`
        );

        if (fitnessResponse.data && fitnessResponse.data.length > 0) {
          const fitnessData = fitnessResponse.data[0];

          setUserData({
            ...fitnessData,
            clientName: userResponse.data?.name || "Unknown User",
            mrnId: userResponse.data?.mrn || "Unknown MRN",
          });

          // Set default viewMode to first available day
          if (fitnessData.days?.length > 0) {
            setViewMode(fitnessData.days[0].day);
          } else {
            setViewMode("all");
          }
        } else {
          setError("No fitness journey data found for this user");
        }
      } catch (error) {
        console.error("Error fetching data:", error);
        setError("Failed to load user or fitness journey data");
      } finally {
        setLoading(false);
      }
    };

    if (userId) {
      fetchUserData();
    } else {
      setError("No user ID provided");
      setLoading(false);
    }
  }, [userId]);
```

```
const renderSessionStatus = (status) => {
  const statusMapping = {
    Completed: { color: "#4CAF50", label: "Completed" },
    Missed: { color: "#F44336", label: "Missed" },
    Partial: { color: "#FFC107", label: "Partial" },
  };

  const { color, label } = statusMapping[status] || {};
  return (
    <span className="status-indicator">
      <span className="status-dot" style={{ backgroundColor: color }}></span>
      <span className="status-text">{label}</span>
    </span>
  );
};

const handleDaySelect = (dayValue) => {
  if (dayValue === "all") {
    setViewMode("all");
  } else {
    setViewMode(Number(dayValue));
  }
};

const renderDayCard = (day) => (
  <div key={day.day} className="card day-card">
    <h4>DAY {day.day}</h4>

    <div className="exercise-section">
      <h5>Yoga Exercises</h5>
      <div className="table-responsive">
        <table className="followup-table">
          <thead>
            <tr>
              <th>Yoga</th>
              <th>Times</th>
              <th>Status</th>
            </tr>
          </thead>
          <tbody>
            {day.yoga?.length > 0 ? (
              day.yoga.map((exercise, index) => (
                <tr key={index}>
                  <td>{exercise.name}</td>
                  <td>{exercise.times}</td>
                  <td>{renderSessionStatus(exercise.status)}</td>
                </tr>
              ))
            ) : (
              <tr>
                <td colSpan="3">No yoga exercises found</td>
              </tr>
            )}
          </tbody>
        </table>
      </div>
    </div>

    <div className="exercise-section">
      <h5>Resistance Training</h5>
      <div className="table-responsive">
        <table className="followup-table">
          <thead>
            <tr>
              <th>Exercise</th>
              <th>Times</th>
              <th>Missed</th>
            </tr>
          </thead>
          <tbody>
            {day.resistanceTraining?.length > 0 ? (
              day.resistanceTraining.map((exercise, index) => (
                <tr key={index}>
```

```
                <td>{exercise.name}</td>
                <td>{exercise.times}</td>
                <td className={exercise.missed ? "missed" : "completed"}>
                  {exercise.missed ? "Yes" : "No"}
                </td>
              </tr>
            ))
          ) : (
            <tr>
              <td colSpan="3">No resistance training exercises found</td>
            </tr>
          )}
        </tbody>
      </table>
    </div>
  </div>
);

if (loading) {
  return (
    <div className="content-container">
      <div className="loading-spinner">Loading...</div>
    </div>
  );
}

if (error) {
  return (
    <div className="content-container">
      <div className="error-message">{error}</div>
    </div>
  );
}

return (
  <div className="content-container">
    <div className="fitness-journey">
      <div className="card user-info-card">
        <h2>Fitness Journey Schedule</h2>

        <div className="day-selector">
          <label htmlFor="day-select">View Day: </label>
          <select
            id="day-select"
            value={viewMode}
            onChange={(e) => handleDaySelect(e.target.value)}
          >
            <option value="all">All Days</option>
            {userData.days?.map((day) => (
              <option key={day.day} value={day.day}>
                Day {day.day}
              </option>
            ))}
          </select>
        </div>
      </div>

      {userData.days?.length > 0 ? (
        viewMode === "all" ? (
          userData.days.map((day) => renderDayCard(day))
        ) : (
          renderDayCard(userData.days.find((day) => day.day === viewMode))
        )
      ) : (
        <div className="card">
          <p>No fitness journey data available for this user.</p>
        </div>
      )}

      <div className="card notes-section">
        <h5>Notes</h5>
        <ul>
```

```
            <li>Select a suitable weight of choice.</li>
            <li>Progress slowly.</li>
            <li>
              <strong>Take rest for 30?60 seconds between sets.</strong>
            </li>
            <li>Journal any difficulty faced in completing the session.</li>
          </ul>
        </div>
      </div>
    </div>
  );
};
export default UserDetails;
```

File: src\dashboard\physio\components\Footer.css

```
.app-footer {
  background-color: var(--bg-secondary);
  color: var(--text-secondary);
  text-align: center;
  border-top: 1px solid var(--border-color);
  /* margin-top: auto; */
  font-size: 0.9rem;
  transition: background-color 0.3s ease, color 0.3s ease;
}

.footer-content {
  max-width: 1200px;
  margin: 0 auto;
}
```

File: src\dashboard\physio\components\Footer.js

```
import React from 'react';
import './Footer.css';

const Footer = () => {
  return (
    <footer className="app-footer">
      <div className="footer-content">
        <p>© {new Date().getFullYear()} PhysioCare Dashboard. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

File: src\dashboard\physio\components\Navbar.css

```
/* Navbar.css */
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem 2rem;
  background-color: var(--bg-primary);
  color: var(--text-primary);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
  position: sticky;
  top: 0;
  z-index: 1000;
  backdrop-filter: blur(10px);
  -webkit-backdrop-filter: blur(10px);
  border-bottom: 1px solid var(--border-color);
}

.navbar-logo {
  font-size: 1.5rem;
  font-weight: bold;
  color: var(--accent);
  display: flex;
  align-items: center;
  gap: 0.5rem;
}
```

```
.navbar-logo::before {
  content: '';
  display: inline-block;
  width: 12px;
  height: 12px;
  background-color: var(--accent);
  border-radius: 50%;
}

.navbar-links {
  display: flex;
  gap: 2rem;
  align-items: center;
}

.navbar-links a {
  color: var(--text-primary);
  text-decoration: none;
  font-weight: 500;
  transition: all 0.2s ease;
  position: relative;
  padding: 0.5rem 0;
}

.navbar-links a::after {
  content: '';
  position: absolute;
  width: 0;
  height: 2px;
  bottom: 0;
  left: 0;
  background-color: var(--accent);
  transition: width 0.3s ease;
}

.navbar-links a:hover {
  color: var(--accent);
}

.navbar-links a:hover::after {
  width: 100%;
}

.theme-toggle {
  background: none;
  border: none;
  cursor: pointer;
  color: var(--text-primary);
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 0.5rem;
  border-radius: 50%;
  transition: all 0.3s ease;
  margin-left: 1rem;
}

.theme-toggle:hover {
  background-color: var(--bg-hover);
  transform: rotate(15deg);
}

.mobile-menu-icon {
  display: none;
}

.mobile-menu {
  display: none;
}

/* Mobile responsive styles */
@media (max-width: 992px) {
  .navbar-links {
```

```
    gap: 1.5rem;
  }
}

@media (max-width: 768px) {
  .navbar {
    padding: 1rem;
  }

  .navbar-links {
    display: none;
  }

  .mobile-menu-icon {
    display: flex;
    align-items: center;
    gap: 1rem;
  }

  .mobile-menu-icon button {
    background: transparent;
    border: none;
    color: var(--text-primary);
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 0.5rem;
    border-radius: 50%;
    transition: background-color 0.2s ease;
  }

  .mobile-menu-icon button:hover {
    background-color: var(--bg-hover);
  }

  .mobile-menu {
    display: flex;
    flex-direction: column;
    position: absolute;
    top: 70px;
    left: 0;
    right: 0;
    background-color: var(--bg-primary);
    padding: 0.5rem;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    z-index: 10;
    border-radius: 0 0 8px 8px;
    overflow: hidden;
    animation: slideIn 0.3s ease forwards;
  }

  @keyframes slideIn {
    from {
      opacity: 0;
      transform: translateY(-10px);
    }
    to {
      opacity: 1;
      transform: translateY(0);
    }
  }

  .mobile-menu a {
    color: var(--text-primary);
    text-decoration: none;
    padding: 1rem;
    border-bottom: 1px solid var(--border-color);
    transition: all 0.2s ease;
    display: flex;
    align-items: center;
  }

  .mobile-menu a::before {
```

```
    content: '';
    display: inline-block;
    width: 8px;
    height: 8px;
    background-color: var(--accent);
    border-radius: 50%;
    margin-right: 10px;
    opacity: 0;
    transition: opacity 0.2s ease;
  }

  .mobile-menu a:last-child {
    border-bottom: none;
  }

  .mobile-menu a:hover {
    background-color: var(--bg-secondary);
    color: var(--accent);
    padding-left: 1.5rem;
  }

  .mobile-menu a:hover::before {
    opacity: 1;
  }
}

/* Extra small devices */
@media (max-width: 480px) {
  .navbar {
    padding: 0.75rem;
  }

  .navbar-logo {
    font-size: 1.25rem;
  }
}
```

File: src\dashboard\physio\components\navbar.js

```
import React from "react";
import "./Navbar.css";
import { Moon, Sun } from "lucide-react";
import { useTheme } from "../../../ThemeProvider";

const Navbar = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <nav className="navbar">
     <a href="../profile"> <div className="navbar-logo">PhysioCare</div></a>
      <button
        className="theme-toggle"
        onClick={toggleTheme}
        aria-label="Toggle dark mode"
      >
        {theme === "dark" ? <Sun size={20} /> : <Moon size={20} />}
      </button>
    </nav>
  );
};

export default Navbar;
```

File: src\dashboard\physio\components\Sidebar.css

```
/* Sidebar.css */

.sidebar {
  overflow-y: auto;
  scrollbar-width: thin;
  scrollbar-color: #cc5500 #f5f5f5;
}

.sidebar::-webkit-scrollbar {
  width: 8px;
```

```
}

.sidebar::-webkit-scrollbar-track {
  background: #f5f5f5;
  border-radius: 4px;
}

.sidebar::-webkit-scrollbar-thumb {
  background: #cc5500;
  border-radius: 4px;
  transition: background-color 0.2s ease;
}

.sidebar::-webkit-scrollbar-thumb:hover {
  background: #cc5500;
}

.sidebar.light-orange::-webkit-scrollbar-thumb {
  background: #cc5500;
}

.sidebar.light-orange::-webkit-scrollbar-thumb:hover {
  background: #cc5500;
}

.sidebar.dark-orange::-webkit-scrollbar-thumb {
  background: #cc5500;
}

.sidebar.dark-orange::-webkit-scrollbar-thumb:hover {
  background: #cc5500;
}

.sidebar.collapsed::-webkit-scrollbar {
  width: 4px;
}

.sidebar {
  --sidebar-width: 280px;
  --sidebar-collapsed-width: 70px;
  --transition: all 0.3s ease;
  --transition-fast: all 0.2s ease;
  width: var(--sidebar-width);
  height: 100%;
  background-color: var(--bg-secondary);
  color: var(--text-primary);
  padding-top: 1.5rem;
  border-right: 1px solid var(--border-color);
  transition: var(--transition);
  overflow-y: auto;
  position: sticky;
  top: 0;
  display: flex;
  flex-direction: column;
}
/* Collapsed state */
.sidebar.collapsed {
  width: var(--sidebar-collapsed-width);
}
.sidebar-header {
  padding: 0 1.5rem;
  margin-bottom: 2rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.sidebar.collapsed .sidebar-header {
  padding: 0 0.5rem;
  margin-bottom: 1rem;
  justify-content: center;
}
.sidebar-profile {
  display: flex;
```

```
  align-items: center;
  gap: 0.75rem;
}
.sidebar.collapsed .sidebar-profile {
  justify-content: center;
}
.sidebar-avatar {
  width: 48px;
  height: 48px;
  border-radius: 50%;
  object-fit: cover;
  border: 2px solid var(--accent);
  transition: var(--transition);
}
.sidebar.collapsed .sidebar-avatar {
  width: 40px;
  height: 40px;
}
.sidebar-username {
  font-size: 1rem;
  color: var(--text-primary);
  font-weight: 500;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  transition: var(--transition);
}
.sidebar.collapsed .sidebar-username {
  display: none;
}
.sidebar-menu-wrapper {
  display: flex;
  flex-direction: column;
  height: calc(100vh - 120px);
  justify-content: space-between;
}
.sidebar-menu {
  list-style: none;
  padding: 0;
  margin: 0;
}
.sidebar-menu li {
  margin-bottom: 0.25rem;
}
.sidebar-menu li a {
  display: flex;
  align-items: center;
  padding: 0.875rem 1.5rem;
  color: var(--text-primary);
  text-decoration: none;
  transition: var(--transition-fast);
  border-left: 4px solid transparent;
  position: relative;
  overflow: hidden;
}
.sidebar.collapsed .sidebar-menu li a {
  padding: 0.875rem;
  justify-content: center;
}
.sidebar-menu li a::before {
  content: "";
  position: absolute;
  top: 0;
  left: -100%;
  width: 100%;
  height: 100%;
  background-color: var(--accent);
  opacity: 0.1;
  transition: transform 0.3s ease;
  z-index: 0;
}
.sidebar-menu li a:hover {
  color: var(--accent);
```

```
  background-color: var(--bg-hover);
}
.sidebar-menu li a:hover::before {
  transform: translateX(100%);
}
/* Active link styling */
.sidebar-menu li a.active {
  background-color: var(--bg-hover);
  color: var(--accent);
  border-left: 4px solid var(--accent);
  font-weight: 500;
}
.sidebar.collapsed .sidebar-menu li a.active {
  border-left: none;
  border-right: 4px solid var(--accent);
}
.sidebar-menu li a.active::before {
  left: 0;
  opacity: 0.05;
}
/* Hide text when collapsed */
.sidebar.collapsed .sidebar-menu li a span {
  display: none;
}
/* Settings at bottom */
.sidebar-settings {
  margin-top: auto;
}
/* Collapse toggle button */
.collapse-toggle {
  width: 100%;
  padding: 0.875rem 1.5rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
  background: none;
  border: none;
  color: var(--text-primary);
  cursor: pointer;
  margin-top: auto;
  border-top: 1px solid var(--border-color);
  transition: var(--transition-fast);
  height: 50px;
}
.sidebar.collapsed .collapse-toggle {
  padding: 0.875rem;
  justify-content: center;
}

.collapse-toggle:hover {
  background-color: var(--bg-hover);
  color: var(--accent);
}
.collapse-toggle-icon {
  transition: transform 0.3s ease;
}
.sidebar.collapsed .collapse-toggle-icon {
  transform: rotate(180deg);
}
.collapse-toggle-text {
  font-weight: 500;
}
.sidebar.collapsed .collapse-toggle-text {
  display: none;
}
/* Sidebar toggle for mobile */
.sidebar-toggle {
  display: none;
  background: none;
  border: none;
  color: var(--text-primary);
  cursor: pointer;
  font-size: 1.5rem;
```

```
}
/* Responsive styles */
@media (max-width: 1200px) {
  .sidebar:not(.collapsed) {
    width: 250px;
  }
}
@media (max-width: 992px) {
  .sidebar:not(.collapsed) {
    width: 220px;
  }
}
/* Body padding when sidebar is expanded or collapsed */
@media (min-width: 769px) {
  body {
    padding-left: var(--sidebar-width);
    transition: var(--transition);
  }
  body.sidebar-collapsed {
    padding-left: var(--sidebar-collapsed-width);
  }
}
@media (max-width: 768px) {
  .sidebar {
    position: fixed;
    left: -280px;
    z-index: 1001;
    box-shadow: 2px 0 10px rgba(0, 0, 0, 0.1);
    height: 100vh;
    padding-top: 1rem;
  }
  .sidebar::-webkit-scrollbar {
    width: 6px;
  }
  .sidebar.open {
    left: 0;
  }
  .sidebar.collapsed {
    width: var(--sidebar-width); /* On mobile, collapsed state is ignored */
  }
  .sidebar-toggle {
    display: block;
    position: fixed;
    bottom: 20px;
    left: 20px;
    background-color: var(--accent);
    color: white;
    border-radius: 50%;
    width: 50px;
    height: 50px;
    display: flex;
    align-items: center;
    justify-content: center;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    z-index: 1002;
    transition: transform 0.3s ease;
  }
  .sidebar-toggle:hover {
    transform: scale(1.05);
  }
  /* Hide collapse toggle on mobile */
  .sidebar .collapse-toggle {
    display: none;
  }
  .sidebar-overlay {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
```

```
    animation: fadeIn 0.3s ease forwards;
  }
  .sidebar-overlay.show {
    display: block;
  }
  @keyframes fadeIn {
    from {
      opacity: 0;
    }
    to {
      opacity: 1;
    }
  }
}
@media (max-width: 480px) {
  .sidebar {
    width: 260px;
  }
}

/* Modal styles for logout confirmation */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1050;
  animation: fadeIn 0.3s ease forwards;
}

.modal-container {
  background-color: var(--bg-secondary, white);
  border-radius: 8px;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.15);
  width: 90%;
  max-width: 400px;
  overflow: hidden;
  animation: slideIn 0.3s ease forwards;
}

@keyframes slideIn {
  from {
    transform: translateY(-20px);
    opacity: 0;
  }
  to {
    transform: translateY(0);
    opacity: 1;
  }
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 16px 20px;
  /* border-bottom: 1px solid var(--border-color, #eaeaea); */
}

.modal-header h3 {
  margin: 0;
  font-size: 18px;
  font-weight: 600;
  color: var(--text-primary, #333);
}

.modal-close {
  background: none;
```

```
  border: none;
  cursor: pointer;
  padding: 4px;
  color: var(--text-secondary, #666);
  transition: var(--transition-fast);
  border-radius: 4px;
}

.modal-close:hover {
  background-color: var(--bg-hover, #f5f5f5);
  color: var(--accent, #4263eb);
}

.modal-body {
  padding: 24px 20px;
  text-align: center;
  color: var(--text-primary, #333);
}

.modal-footer {
  display: flex;
  justify-content: flex-end;
  padding: 16px 20px;
  /* border-top: 1px solid var(--border-color, #eaeaea); */
  gap: 12px;
}

.btn {
  padding: 10px 16px;
  border-radius: 6px;
  cursor: pointer;
  font-weight: 500;
  border: none;
  transition: background-color 0.2s, transform 0.1s;
}

.btn:active {
  transform: translateY(1px);
}

.btn-primary {
  background-color: var(--accent, #4263eb);
  color: white;
}

.btn-primary:hover {
  background-color: var(--accent-hover, #364fc7);
}

.btn-secondary {
  background-color: var(--bg-hover, #e9ecef);
  color: var(--text-secondary, #495057);
}

.btn-secondary:hover {
  background-color: var(--border-color, #dee2e6);
}

/* Toast message styles - centered */
.toast-overlay {
  position: fixed;
  top: 20px;
  left: 50%;
  transform: translateX(-50%);
  display: flex;
  align-items: center;
  justify-content: center;
  pointer-events: none;
  z-index: 1050;
}

.toast-container {
  padding: 14px 24px;
```

```
  border-radius: 8px;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
  max-width: 500px;
  width: 100%;
  text-align: center;
  pointer-events: all;
  animation: toastIn 0.3s ease forwards;
}

@keyframes toastIn {
  from {
    transform: translateY(20px);
    opacity: 0;
  }
  to {
    transform: translateY(0);
    opacity: 1;
  }
}

.toast-container.success {
  background-color: #4bb543;
  color: white;
  max-width: 400px;
}

.toast-content {
  font-size: 16px;
  font-weight: 500;
}

/* Ensure modals and toasts are above sidebar on mobile */
@media (max-width: 768px) {
  .modal-overlay,
  .toast-overlay {
    z-index: 1100;
  }
}

/* Style for Claude toggle button */
.claude-toggle {
  display: flex;
  align-items: center;
  justify-content: center;
}

.toggle-icon {
  transition: var(--transition);
}
```

File: src\dashboard\physio\components\sidebar.js

```
import React, { useEffect, useState } from "react";
import { NavLink, useNavigate } from "react-router-dom";
import ProfileImage from "../../../image/1.png";
import "./Sidebar.css";
import {
  Home,
  Users,
  Dumbbell,
  BarChart2,
  LogOut,
  Menu,
  Workflow,
  ChevronLeft,
  ChevronRight,
  X,
  Check,
} from "lucide-react";

const Sidebar = () => {
  const [isOpen, setIsOpen] = useState(window.innerWidth > 768);
  const [collapsed, setCollapsed] = useState(false);
  const [isHovering, setIsHovering] = useState(false);
```

```
const [showToast, setShowToast] = useState(false);
const [toastMessage, setToastMessage] = useState("");
const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);
const navigate = useNavigate();

const toggleSidebar = () => {
  setIsOpen(!isOpen);
};

const toggleCollapse = () => {
  setCollapsed(!collapsed);
};

const handleResize = () => {
  setIsOpen(window.innerWidth > 768);
};

const closeSidebar = () => {
  if (isOpen && window.innerWidth <= 768) {
    setIsOpen(false);
  }
};

// Hover handlers
const handleMouseEnter = () => {
  if (collapsed && window.innerWidth > 768) {
    setIsHovering(true);
  }
};

const handleMouseLeave = () => {
  if (collapsed && window.innerWidth > 768) {
    setIsHovering(false);
  }
};

useEffect(() => {
  window.addEventListener("resize", handleResize);
  return () => window.removeEventListener("resize", handleResize);
}, []);

// Add body class for content margin adjustment
useEffect(() => {
  document.body.classList.toggle(
    "sidebar-collapsed",
    collapsed && !isHovering
  );
}, [collapsed, isHovering]);

const navLinkClass = ({ isActive }) => (isActive ? "active" : "");

// Show logout confirmation modal
const handleLogoutClick = (e) => {
  e.preventDefault();
  setShowLogoutConfirm(true);
};

// Handle confirm logout
const confirmLogout = () => {
  // Perform logout actions here
  localStorage.clear(); // or remove auth tokens etc.

  // Close the confirmation modal
  setShowLogoutConfirm(false);

  // Show toast message
  setToastMessage("Logged out successfully.");
  setShowToast(true);

  // After a delay, navigate to login page
  setTimeout(() => {
    setShowToast(false);
    navigate("/login");
```

```
  }, 3000);
};

// Handle cancel logout
const cancelLogout = () => {
  setShowLogoutConfirm(false);
};

return (
  <>
    {/* Mobile Toggle Button */}
    <button
      className="sidebar-toggle"
      onClick={toggleSidebar}
      aria-label="Toggle sidebar"
    >
      <Menu size={24} />
    </button>

    {/* Overlay for mobile */}
    {isOpen && window.innerWidth <= 768 && (
      <div className="sidebar-overlay show" onClick={closeSidebar}></div>
    )}

    {/* Center Modal Logout Confirmation */}
    {showLogoutConfirm && (
      <div className="modal-overlay">
        <div className="modal-container">
          <div className="modal-header">
            <center>
              <h3>Logout Confirmation</h3>
            </center>
            <button className="modal-close" onClick={cancelLogout}>
              <X size={18} />
            </button>
          </div>
          <div className="modal-body">
            <p>Are you sure you want to log out?</p>
          </div>
          <div className="modal-footer">
            {/* <button className="btn btn-secondary" onClick={cancelLogout}>
              Cancel
            </button> */}
            <button className="btn btn-primary" onClick={confirmLogout}>
              Yes, Logout
            </button>
          </div>
        </div>
      </div>
    )}

    {/* Centered Toast Message */}
    {showToast && (
      <div className="toast-overlay">
        <div className="toast-container success">
          <div className="toast-content">
            <Check
              size={18}
              style={{ marginRight: "8px", verticalAlign: "middle" }}
            />
            {toastMessage}
          </div>
        </div>
      </div>
    )}

    <aside
      className={`sidebar ${isOpen ? "open" : ""} ${
        collapsed ? "collapsed" : ""
      } ${isHovering ? "hovering" : ""}`}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
    >
```

```
<div className="sidebar-header">
  <div className="sidebar-profile">
    <img src={ProfileImage} alt="Profile" className="sidebar-avatar" />
    <span className="sidebar-username">John Doe</span>
  </div>
</div>

<nav className="sidebar-menu-wrapper">
  <ul className="sidebar-menu">
    <li>
      <NavLink to="/profile" className={navLinkClass}>
        <Home
          size={18}
          style={{
            marginRight: !collapsed || isHovering ? "10px" : "0",
          }}
        />
        <span>Dashboard</span>
      </NavLink>
    </li>

    <li>
      <NavLink to="/PhysiosAppointments" className={navLinkClass}>
        <Users
          size={18}
          style={{
            marginRight: !collapsed || isHovering ? "10px" : "0",
          }}
        />
        <span>Appointments</span>
      </NavLink>
    </li>



    <li>
      <NavLink to="/PhysioPlanAssign" className={navLinkClass}>
        <Workflow
          size={18}
          style={{
            marginRight: !collapsed || isHovering ? "10px" : "0",
          }}
        />
        <span>PhysioPlanAssign</span>
      </NavLink>
    </li>
    <li>
      <NavLink to="/PhysioPasswordRequest" className={navLinkClass}>
        <Workflow
          size={18}
          style={{
            marginRight: !collapsed || isHovering ? "10px" : "0",
          }}
        />
        <span>Password Request</span>
      </NavLink>
    </li>
    {/* Updated logout item */}
    <li>
      {/* Use a regular clickable element here instead of NavLink */}
      <a
        href="#"
        onClick={handleLogoutClick}
        className={navLinkClass({ isActive: false })}
        style={{
          cursor: "pointer",
          display: "flex",
          alignItems: "center",
        }}
      >
        <LogOut
          size={18}
          style={{
```

```
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Log Out</span>
            </a>
          </li>
        </ul>

        {/* Claude-style toggle button at the bottom */}
        <button
          className="collapse-toggle"
          onClick={toggleCollapse}
          aria-label="Toggle collapse"
        >
          <div className="claude-toggle">
            {collapsed ? (
              <ChevronRight size={16} className="toggle-icon" />
            ) : (
              <ChevronLeft size={16} className="toggle-icon" />
            )}
          </div>
          {/* <span className="collapse-toggle-text">Collapse</span> */}
        </button>
      </nav>
    </aside>
  </>
 );
};

export default Sidebar;
```

File: src\dashboard\physio\pages\PhysioPasswordRequest.js

```
import React, { useState } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../../ThemeProvider"; // Adjust the import based on your project structure
import "react-toastify/dist/ReactToastify.css";
import "../../master_admin/master_admin.css"; // Ensure this path is correct

const PhysioPasswordRequest = () => {
  const { theme } = useTheme(); // Access the current theme
  const [formData, setFormData] = useState({
    counselorName: "",
    email: "",
    reason: "",
  });

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (!formData.counselorName || !formData.email || !formData.reason) {
      toast.error("All fields are required!");
      return;
    }

    try {
      await axios.post("http://localhost:5000/passwordChangeRequests", {
        ...formData,
        status: "Pending",
        requestedAt: new Date().toISOString(),
      });

      toast.success("Request submitted to admin!");
      setFormData({ counselorName: "", email: "", reason: "" });
    } catch (err) {
      toast.error("Failed to submit request.");
    }
  };
```

```
  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Password Change Request</h1>
      <div className="card" style={{border:"1px solid #cc5500"}}>
        <form onSubmit={handleSubmit} className="form">
          <div className="form-group">
            <label htmlFor="counselorName">Counselor Name:</label>
            <input
              type="text"
              id="counselorName"
              name="counselorName"
              placeholder="Enter your name"
              value={formData.counselorName}
              onChange={handleChange}
              className="search-input"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="email">Email ID:</label>
            <input
              type="email"
              id="email"
              placeholder="Enter your email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="search-input"
              required
            />
          </div>
          <div className="form-group">
            <label htmlFor="reason">Reason for Change:</label>
            <input
              type="text"
              id="reason"
              placeholder="Enter reason for password change"
              name="reason"
              value={formData.reason}
              onChange={handleChange}
              className="search-input"

              required
            />
          </div>
          <div className="center-btn">
           <center> <button type="submit" className="btn btn-primary">
              Submit Request
            </button></center>
          </div>
        </form>
      </div>
    </div>
  );
};

export default PhysioPasswordRequest;
```

File: src\pages\Unauthorized.js

```
import React from "react";

const Unauthorized = () => {
  return (
    <div style={{ padding: "2rem", textAlign: "center" }}>
      <h1>403 - Unauthorized</h1>
      <p>You do not have permission to access this page.</p>
    </div>
  );
};

export default Unauthorized;
```