■ Frontend File List:

- src\App.css
- src\App.js
- src\App.test.js
- src\index.css
- src\index.js
- src\reportWebVitals.js
- src\setupTests.js
- src\theme.css
- src\ThemeProvider.js
- src\auth\forgotpassword.css
- src\auth\forgotpassword.js
- src\auth\login.css
- src\auth\login.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\diet_assign.js
- src\dashboard\diet\diet_clientmanagement.js
- src\dashboard\diet\diet_physioo.js
- src\dashboard\diet\diet_profile.js
- src\dashboard\diet\diet_report.js
- src\dashboard\diet\diet_userdetails.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\doctor\DoctorAssignDecision .js
- src\dashboard\doctor\DoctorAssignmentDashboard.js
- src\dashboard\doctor\DoctorsAppointments.js
- src\dashboard\doctor\DoctorUpcomingAppointments .js
- src\dashboard\doctor\doctor_clientmanagement.js
- src\dashboard\doctor\doctor_physioo.js
- src\dashboard\doctor\doctor_profile.js
- src\dashboard\doctor\doctor_report.js
- src\dashboard\doctor\doctor_userdetails.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\pages\DoctorPasswordRequest.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\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\SecurityControls.js
- src\dashboard\master_admin\Services.css
- src\dashboard\master_admin\Services.js
- src\dashboard\master_admin\SystemOverview.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\physio\assign.css
- src\dashboard\physio\assign.js

- src\dashboard\physio\clientManagement.css
- src\dashboard\physio\clientManagement.js
- src\dashboard\physio\physio.css
- src\dashboard\physio\physioo.js
- src\dashboard\physio\PhysiosAppointments.js
- src\dashboard\physio\profile.css
- src\dashboard\physio\profile.js
- src\dashboard\physio\report.css
- src\dashboard\physio\report.js
- 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

```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 {
```

```css
  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%;
}
```

```css
/*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";

// =============================================
// 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 Assign from "./dashboard/physio/assign";
import ClientManagement from "./dashboard/physio/clientManagement";
import Report from "./dashboard/physio/report";
import UserDetails from "./dashboard/physio/userDetails";
import Physioo from "./dashboard/physio/physioo";
import PhysiosAppointments from "./dashboard/physio/PhysiosAppointments";
import PhysioPasswordRequest from "./dashboard/physio/pages/PhysioPasswordRequest";

// =============================================
// 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 DietClientManagement from "./dashboard/diet/diet_clientmanagement";
import DietReport from "./dashboard/diet/diet_report";
import DietUserDetails from "./dashboard/diet/diet_userdetails";
import DietPhysioo from "./dashboard/diet/diet_physioo";
import DietitiansAppointments from "./dashboard/diet/DietitiansAppointments";
import DietPasswordRequest from "./dashboard/diet/pages/DietPasswordRequest";
```

```javascript
// ===========================================
// 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 DoctorProfile from "./dashboard/doctor/doctor_profile";
import DoctorAssignDecision from "./dashboard/doctor/DoctorAssignDecision ";
import DoctorClientManagement from "./dashboard/doctor/doctor_clientmanagement";
import DoctorReport from "./dashboard/doctor/doctor_report";
import DoctorUserDetails from "./dashboard/doctor/doctor_userdetails";
import DoctorPhysioo from "./dashboard/doctor/doctor_physioo";
import DoctorUpcomingAppointments from "./dashboard/doctor/DoctorUpcomingAppointments ";
import DoctorAssignmentDashboard from "./dashboard/doctor/DoctorAssignmentDashboard";
import DoctorsAppointments from "./dashboard/doctor/DoctorsAppointments";
import DoctorPasswordRequest from "./dashboard/doctor/pages/DoctorPasswordRequest";

// ===========================================
// 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";

// 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";

// ===========================================
// 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">
```

```jsx
          <div className="content-wrapper">
            <Outlet />
          </div>
          <MasterAdminFooter />
        </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>
  );
};

// ===============================================
// 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 />} />

          {/* ===============================================
              PHYSIO DASHBOARD ROUTES
              =============================================== */}
          <Route element={<PhysioDashboardLayout />}>
            <Route path="/" element={<Navigate to="/profile" replace />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/assign" element={<Assign />} />
            <Route path="/clients" element={<ClientManagement />} />
            <Route path="/reports" element={<Report />} />
            <Route path="/user" element={<UserDetails />} />
            <Route path="/physio" element={<Physioo />} />
            <Route path="/user-details/:userId" element={<UserDetails />} />
            <Route path="/user/:id/report" element={<Report />} />
            <Route path="/PhysiosAppointments" element={<PhysiosAppointments />} />
            <Route path="/PhysioPasswordRequest" element={<PhysioPasswordRequest />} />
          </Route>

          {/* ===============================================
              DIET DASHBOARD ROUTES
```

```jsx
                       =========================================== */}
        <Route path="/diet" element={<DietDashboardLayout />}>
          <Route path="diet_profile" element={<DietProfile />} />
          <Route path="diet_assign" element={<DietAssign />} />
          <Route path="diet_clients" element={<DietClientManagement />} />
          <Route path="diet_reports" element={<DietReport />} />
          <Route path="user" element={<DietUserDetails />} />
          <Route path="diet_physio" element={<DietPhysioo />} />
          <Route path="user-details/:userId" element={<DietUserDetails />} />
          <Route path="user/:id/report" element={<DietReport />} />
          <Route path="DietitiansAppointments" element={<DietitiansAppointments />} />
          <Route path="DietPasswordRequest" element={<DietPasswordRequest />} />
        </Route>

        {/* ===========================================
            DOCTOR DASHBOARD ROUTES
            =========================================== */}
        <Route path="/doctor" element={<DoctorDashboardLayout />}>
          <Route index element={<Navigate to="doctor_profile" replace />} />
          <Route path="doctor_profile" element={<DoctorProfile />} />
          <Route path="DoctorAssignDecision" element={<DoctorAssignDecision />} />
          <Route path="doctor_clients" element={<DoctorClientManagement />} />
          <Route path="doctor_reports" element={<DoctorReport />} />
          <Route path="user" element={<DoctorUserDetails />} />
          <Route path="doctor_physioo" element={<DoctorPhysioo />} />
          <Route path="user/:id/report" element={<DoctorReport />} />
          <Route path="DoctorsAppointments" element={<DoctorsAppointments />} />
          <Route path="user-details/:userId" element={<DoctorUserDetails />} />
          <Route path="DoctorUpcomingAppointments" element={<DoctorUpcomingAppointments />} />
          <Route path="DoctorAssignmentDashboard" element={<DoctorAssignmentDashboard />} />
          <Route path="DoctorPasswordRequest" element={<DoctorPasswordRequest />} />
        </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 />} />

          {/* Appointment Management Routes */}
          <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>

        {/* ===========================================
            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>
      </Routes>
    </ThemeProvider>
   </BrowserRouter>
  );
};

export default App;
```

File: src\App.test.js
```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

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
```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';
```

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

```css
  --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;
}

button {
  cursor: pointer;
}
```

File: src\ThemeProvider.js

```javascript
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\auth\forgotpassword.css

```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

```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]);
```

```jsx
    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

```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=format
}

/* 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=format
}

/* 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;
```

```css
  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' vie
  background-repeat: no-repeat;
  background-position: right 16px center;
```

```css
    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;
```

```css
  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) {
```

```css
  .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

```javascript
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 [formData, setFormData] = useState({
    email: savedEmail,
    password: savedPassword,
  });

  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) => {
  setFormData({ ...formData, [e.target.id]: e.target.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);
    } else {
      localStorage.removeItem("savedEmail");
      localStorage.removeItem("savedPassword");
    }

    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/doctor_profile");
          break;
        case "masteradmin":
          navigate("/masteradmin");
          break;
        case "counselor":
          navigate("/counselor");
          break;
        default:
          navigate("/unauthorized");
      }
```

```jsx
      }, 1500);
    } 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 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\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)}
```

```jsx
                            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
```jsx
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");
  }

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

  const userRole = user.role?.toLowerCase();
  const allowed = allowedRoles.map(role => role.toLowerCase());

  if (!allowed.includes(userRole)) {
    return <Navigate to="/unauthorized" />;
```

```
  }

  return children;
};

export default ProtectedRoute;
```

File: src\dashboard\counselor\counselor.css
```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
```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("");
```

```jsx
      })
      .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>
           </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) => (
```

```jsx
                <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",
      }}
    >
```

```jsx
      >
        <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>
```

```jsx
          )}
        </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

```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

```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" : "");
```

```jsx
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" />
```

```jsx
                  <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

```jsx
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) => {
```

```jsx
      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">
        <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

```javascript
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) => {
```

```jsx
  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\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)}
```

```jsx
                />
                <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_clientmanagement.js

```javascript
import React, { useState, useEffect } from "react";
import { useTheme } from "../../ThemeProvider";
import axios from "axios";
import "../physio/clientManagement.css";

export default function ClientManagement() {
  const { theme } = useTheme();
  const [clients, setClients] = useState([]);
  const [loading, setLoading] = useState(true);
  const [showModal, setShowModal] = useState(false);
  const [editIndex, setEditIndex] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [activeFilter, setActiveFilter] = useState("ALL");
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    status: "Active",
    type: "Client",
    resolution: "In Progress",
  });

  useEffect(() => {
    fetchClients();
  }, []);

  const fetchClients = async () => {
    try {
      const response = await axios.get("http://localhost:3001/clients");
      setClients(response.data);
    } catch (error) {
      console.error("Error fetching clients:", error);
    } finally {
      setLoading(false);
    }
  };

  const openModal = (client = null, index = null) => {
    if (client) {
      setFormData({
        ...client,
        type: client.type || "Client",
        resolution: client.resolution || "In Progress",
      });
      setEditIndex(index);
    } else {
      setFormData({
        name: "",
        email: "",
```

```
        status: "Active",
        type: "Client",
        resolution: "In Progress",
      });
      setEditIndex(null);
    }
    setShowModal(true);
  };

  const closeModal = () => {
    setShowModal(false);
  };

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

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

  const handleFilterChange = (filter) => {
    setActiveFilter(filter);
  };

  const handleSave = async () => {
    if (!formData.name || !formData.email) {
      alert("Please fill in all required fields");
      return;
    }

    try {
      if (editIndex !== null) {
        const updatedClient = {
          ...formData,
          avatar: clients[editIndex].avatar,
        };
        await axios.put(
          `http://localhost:3001/clients/${clients[editIndex].id}`,
          updatedClient
        );
        const updated = [...clients];
        updated[editIndex] = updatedClient;
        setClients(updated);
      } else {
        const newClient = {
          ...formData,
          avatar: `https://i.pravatar.cc/150?img=${clients.length + 6}`,
        };
        const response = await axios.post(
          "http://localhost:3001/clients",
          newClient
        );
        setClients([...clients, response.data]);
      }
      closeModal();
    } catch (error) {
      console.error("Error saving client:", error);
    }
  };

  const handleDelete = async (index) => {
    if (window.confirm("Are you sure you want to delete this client?")) {
      try {
        const clientId = clients[index].id;
        await axios.delete(`http://localhost:3001/clients/${clientId}`);
        const updated = [...clients];
        updated.splice(index, 1);
        setClients(updated);
      } catch (error) {
        console.error("Error deleting client:", error);
      }
```

```jsx
    }
  };

  const markAsResolved = async (index) => {
    try {
      const updatedClient = {
        ...clients[index],
        resolution: "Resolved",
      };
      await axios.put(
        `http://localhost:3001/clients/${clients[index].id}`,
        updatedClient
      );
      const updated = [...clients];
      updated[index] = updatedClient;
      setClients(updated);
    } catch (error) {
      console.error("Error updating resolution status:", error);
    }
  };

  const filteredClients = clients.filter((client) => {
    // Filter by search term
    const matchesSearch =
      client.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
      client.email.toLowerCase().includes(searchTerm.toLowerCase());

    // Filter by status button
    const matchesFilter =
      activeFilter === "ALL" ||
      (activeFilter === "IN PROGRESS" && client.resolution === "In Progress") ||
      (activeFilter === "RESOLVED" && client.resolution === "Resolved");

    return matchesSearch && matchesFilter;
  });

  return (
    <div className={`client-page ${theme}`}>
      <div className="client-container">
        <div className="client-header">
          <h2>Client Management</h2>
          <div className="client-controls">
            {/* <div className="search-container">
              <input
                type="text"
                placeholder="Search clients..."
                value={searchTerm}
                onChange={handleSearchChange}
                className="search-input"
              />
            </div> */}
            <div className="filter-buttons">
              <button
                className={`filter-btn ${
                  activeFilter === "ALL" ? "active" : ""
                }`}
                onClick={() => handleFilterChange("ALL")}
              >
                ALL
              </button>
              <button
                className={`filter-btn ${
                  activeFilter === "IN PROGRESS" ? "active in-progress" : ""
                }`}
                onClick={() => handleFilterChange("IN PROGRESS")}
              >
                In Progress
              </button>
              <button
                className={`filter-btn ${
                  activeFilter === "RESOLVED" ? "active" : ""
                }`}
                onClick={() => handleFilterChange("RESOLVED")}
```

```
          >
            Resolved
          </button>
        </div>
        <div className="action-buttons-right">
          <button
            className="add-consultant-btn"
            onClick={() => {
              setFormData({
                name: "",
                email: "",
                status: "Active",
                type: "Consultant",
                resolution: "In Progress",
              });
              setShowModal(true);
            }}
          >
            Add Consultant
          </button>
          <button className="add-btn" onClick={() => openModal()}>
            <span className="btn-icon">+</span> Add Client
          </button>
        </div>
      </div>
    </div>

    <div className="table-responsive">
      {loading ? (
        <p>Loading clients...</p>
      ) : filteredClients.length > 0 ? (
        <table
          className="client-table"
          style={{ border: "1px solid #cc5500" }}
        >
          <thead>
            <tr>
              <th>Name</th>
              <th className="hide-sm">Email</th>
              <th>Status</th>
              <th>Type</th>
              <th>Resolution</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {filteredClients.map((client, i) => (
              <tr key={client.id}>
                <td className="client-name-cell">
                  <div className="client-info">
                    <img
                      src={client.avatar}
                      alt={client.name}
                      className="avatar"
                    />
                    <div className="client-details">
                      <span className="client-name">{client.name}</span>
                      <span className="client-email-mobile">
                        {client.email}
                      </span>
                    </div>
                  </div>
                </td>
                <td className="hide-sm">{client.email}</td>
                <td>
                  <span
                    className={`status-badge ${client.status.toLowerCase()}`}
                  >
                    {client.status}
                  </span>
                </td>
                <td>
                  <span
```

```jsx
                  className={`type-badge ${(
                    client.type || "Client"
                  ).toLowerCase()}`}
                >
                  {client.type || "Client"}
                </span>
              </td>
              <td>
                <span
                  className={`resolution-badge ${(
                    client.resolution || "In Progress"
                  )
                    .toLowerCase()
                    .replace(" ", "-")}`}
                >
                  {client.resolution || "In Progress"}
                </span>
              </td>
              <td>
                <div className="action-buttons">
                  <button
                    className="edit-btn"
                    onClick={() => openModal(client, i)}
                  >
                    <span className="btn-text">Edit</span>
                  </button>
                  {client.resolution !== "Resolved" && (
                    <button
                      className="resolve-btn"
                      onClick={() => markAsResolved(i)}
                    >
                      <span className="btn-text">Mark as Resolved</span>
                    </button>
                  )}
                  <button
                    className="delete-btn"
                    onClick={() => handleDelete(i)}
                  >
                    <span className="btn-text">Delete</span>
                  </button>
                </div>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    ) : (
      <div className="no-results">
        <p>No clients found matching your search.</p>
      </div>
    )}
  </div>

  {showModal && (
    <div className="modal-overlay" onClick={closeModal}>
      <div className="modal-box" onClick={(e) => e.stopPropagation()}>
        <div className="modal-header">
          <h3>
            {editIndex !== null
              ? "Edit Client"
              : `Add New ${formData.type}`}
          </h3>
          <button className="close-modal" onClick={closeModal}>
            ×
          </button>
        </div>
        <div className="modal-body">
          <div className="form-group">
            <label htmlFor="name">Name</label>
            <input
              type="text"
              id="name"
              name="name"
```

```jsx
                    placeholder={`Enter ${formData.type.toLowerCase()} name`}
                    value={formData.name}
                    onChange={handleInputChange}
                    required
                  />
                </div>
                <div className="form-group">
                  <label htmlFor="email">Email</label>
                  <input
                    type="email"
                    id="email"
                    name="email"
                    placeholder={`Enter ${formData.type.toLowerCase()} email`}
                    value={formData.email}
                    onChange={handleInputChange}
                    required
                  />
                </div>
                <div className="form-group">
                  <label htmlFor="status">Status</label>
                  <select
                    id="status"
                    name="status"
                    value={formData.status}
                    onChange={handleInputChange}
                  >
                    <option value="Active">Active</option>
                    <option value="Inactive">Inactive</option>
                  </select>
                </div>
                <div className="form-group">
                  <label htmlFor="type">Type</label>
                  <select
                    id="type"
                    name="type"
                    value={formData.type}
                    onChange={handleInputChange}
                  >
                    <option value="Client">Client</option>
                    <option value="Consultant">Consultant</option>
                  </select>
                </div>
                <div className="form-group">
                  <label htmlFor=`resolution">Resolution Status</label>
                  <select
                    id="resolution"
                    name="resolution"
                    value={formData.resolution}
                    onChange={handleInputChange}
                  >
                    <option value="In Progress">In Progress</option>
                    <option value="Resolved">Resolved</option>
                  </select>
                </div>
              </div>
              <div className="modal-actions">
                <button className="cancel-btn" onClick={closeModal}>
                  Cancel
                </button>
                <button className="save-btn" onClick={handleSave}>
                  Save
                </button>
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
}
```

File: src\dashboard\diet\diet_physioo.js

```javascript
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">
```

```jsx
        <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

```javascript
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>
  );
};
```

```jsx
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",
```

```jsx
};

// 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
          }}
```

```jsx
            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>
<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
```

```jsx
                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>
```

```jsx
          </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",
                }}
```

```jsx
            >
              <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={{
```

```jsx
                        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>
<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,
                }}
              >
```

```jsx
                    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>{" "}
    {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,
          }}
        >
```

```jsx
<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\diet_report.js

```javascript
import React, { useState } from "react";
import axios from "axios";
import "../physio/report.css";

const Report = () => {
  const [mrn, setMrn] = useState("");
  const [searchedMrn, setSearchedMrn] = useState("");
  const [pdfUrl, setPdfUrl] = useState("");
  const [loading, setLoading] = useState(false);
  const [notFound, setNotFound] = useState(false);

  const handleSearch = async () => {
    const formattedMrn = mrn.trim().toUpperCase();
    if (!formattedMrn) return;

    setLoading(true);
    setNotFound(false);
    setPdfUrl(""); // reset before new search

    try {
      const response = await axios.get(
        `http://localhost:3001/reports?mrn=${formattedMrn}`
      );
      if (response.data.length > 0) {
        // Get the relative path from the db.json file
        const relativePath = response.data[0].pdfUrl;

        // Create the full path by prepending the public path
        // The public folder is served at the root of your app
        const fullPath = relativePath.startsWith("/")
          ? relativePath
          : `/${relativePath}`;

        setPdfUrl(fullPath);
        setSearchedMrn(formattedMrn);
      } else {
        setNotFound(true);
      }
    } catch (error) {
      console.error("Error fetching report:", error);
      setNotFound(true);
    } finally {
      setLoading(false);
    }
  };

  const handleRemove = () => {
    setSearchedMrn("");
    setMrn("");
    setPdfUrl("");
    setNotFound(false);
  };
```

```
  const handleMrnChange = (e) => {
    setMrn(e.target.value.toUpperCase());
  };

  return (
    <div className="report-body standalone-report">
      <div className="report-card">
        <div className="content-area">
          <h2>Patient Reports</h2>
          <p className="page-description">
            This page allows you to view and download client reports by entering
            the MRN (Medical Record Number). Please ensure the MRN is accurate
            to retrieve the correct PDF report.
          </p>
          <div className="search-container">
            <input
              type="text"
              placeholder="Enter MRN"
              value={mrn}
              onChange={handleMrnChange}
              className="uppercase-input"
            />
            <button onClick={handleSearch}>Search</button>
          </div>

          {loading && <p>Loading report...</p>}

          {searchedMrn && (
            <div className="pdf-container">
              <h3>Report for MRN: {searchedMrn}</h3>
              {pdfUrl ? (
                <>
                  <iframe
                    src={pdfUrl}
                    width="100%"
                    height="600px"
                    title="Patient Report"
                    style={{ border: "1px solid #ccc", borderRadius: "6px" }}
                  />
                  <button onClick={handleRemove} className="remove-button">
                    Remove
                  </button>
                </>
              ) : (
                <p>PDF not found for this MRN.</p>
              )}
            </div>
          )}

          {notFound && !loading && (
            <p className="error-text">No report found for MRN: {mrn}</p>
          )}
        </div>
      </div>
    </div>
  );
};

export default Report;
```

File: src\dashboard\diet\diet_userdetails.js

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import axios from "axios";
import "../physio/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:3001/users/${userId}`
          );

          const fitnessResponse = await axios.get(
            `http://localhost:3001/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>
  </div>
);

if (loading) {
  return (
```

```jsx
        <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\diet\components\diet_footer.js

```jsx
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_clients" className={navLinkClass}>
                <Users size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
                <span>Client Management</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/diet_reports" className={navLinkClass}>
                <BarChart2 size={18} style={{ marginRight: !collapsed || isHovering ? "10px" : "0" }} />
```

```
            <span>Reports</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">
        <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\doctor\DoctorAssignDecision .js

```
import React, { useState } from "react";
import { useTheme } from "../../ThemeProvider";
import { FileText, Search, CheckCircle } from "lucide-react";
import { toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "../physio/assign.css";

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

```
const [searchMRN, setSearchMRN] = useState("");
const [patient, setPatient] = useState(null);
const [assignment, setAssignment] = useState({});
const [loading, setLoading] = useState(false);
const [prescriptionSaved, setPrescriptionSaved] = useState(false);

const handleSearch = async () => {
  if (!searchMRN.trim()) {
    toast.warn("Please enter an MRN.");
    return;
  }

  setLoading(true);
  try {
    const res = await fetch(
      `http://localhost:3001/patients?mrn=${searchMRN}`
    );
    const data = await res.json();
    if (data.length > 0) {
      const foundPatient = data[0];
      setPatient(foundPatient);
      setAssignment(foundPatient.assignedTo || {});
      setPrescriptionSaved(false);
    } else {
      toast.error("Patient not found.");
      setPatient(null);
      setAssignment({});
      setPrescriptionSaved(false);
    }
  } catch (error) {
    console.error("Error fetching patient:", error);
    toast.error("Error fetching patient data.");
  } finally {
    setLoading(false);
  }
};

const handleAssignment = async (role) => {
  if (!patient || !prescriptionSaved) {
    toast.warn("Please save the prescription before assignment.");
    return;
  }

  const updatedAssignment = {
    ...assignment,
    [role.toLowerCase()]: true,
  };

  try {
    const res = await fetch(`http://localhost:3001/patients/${patient.id}`, {
      method: "PATCH",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        assignedTo: updatedAssignment,
        prescription: patient.prescription || {},
      }),
    });

    if (res.ok) {
      setAssignment(updatedAssignment);
      setPatient({
        ...patient,
        assignedTo: updatedAssignment,
      });
      toast.success(`Assigned to ${role}`);
    } else {
      toast.error("Failed to assign patient.");
    }
  } catch (error) {
    console.error("Assignment error:", error);
    toast.error("Error assigning patient.");
```

```jsx
    }
  };

  const handleFinish = () => {
    toast.success("All tasks completed for this patient.");
    setSearchMRN("");
    setPatient(null);
    setAssignment({});
    setPrescriptionSaved(false);
  };

  return (
    <div className={`assign-page ${theme}`}>
      <div className="assign-container">
        <h2 className="section-title">Doctor Assignment Interface</h2>

        <div className="input-group">
          <input
            type="text"
            className="search-inputs"
            placeholder="Enter MRN to fetch patient report"
            value={searchMRN}
            onChange={(e) => setSearchMRN(e.target.value.toUpperCase())}
            onKeyDown={(e) => e.key === "Enter" && handleSearch()}
          />
          <button className="search-button" onClick={handleSearch}>
            <Search size={16} /> Search
          </button>
        </div>

        {loading && <p>Loading...</p>}

        {patient && (
          <div className="row">
            <div className="col card">
              <h3>Patient Information</h3>
              <p>
                <strong>Name:</strong> {patient.name}
              </p>
              <p>
                <strong>Age:</strong> {patient.age}
              </p>
              <p>
                <strong>Gender:</strong> {patient.gender}
              </p>
              <p>
                <strong>MRN:</strong> {patient.mrn}
              </p>
              <p>
                <strong>Condition:</strong> {patient.condition}
              </p>
            </div>

            <div className="col card">
              <h3>Lab Report</h3>
              {patient.reportPdfUrl ? (
                <>
                  <div className="pdf-viewer-container">
                    <iframe
                      src={patient.reportPdfUrl}
                      title="Lab Report PDF"
                      width="100%"
                      height="400px"
                      style={{
                        border: "1px solid var(--border)",
                        borderRadius: "8px",
                      }}
                    ></iframe>
                    <div style={{ marginTop: "0.5rem" }}>
                      <a
                        href={patient.reportPdfUrl}
                        target="_blank"
                        rel="noopener noreferrer"
```

```
                  className="open-pdf-link"
                >
                  ? Open PDF in New Tab
                </a>
              </div>
            </div>
          </>
        ) : (
          <p>No PDF report available.</p>
        )}
      </div>
    </div>
)}

{patient && (
  <div className="card" style={{ marginTop: "2rem" }}>
    <h3>Doctor's Prescription</h3>
    <label htmlFor="prescriptionText">Write Prescription:</label>
    <textarea
      id="prescriptionText"
      rows="4"
      style={{ width: "100%", marginBottom: "1rem" }}
      value={patient.prescription?.text || ""}
      onChange={(e) =>
        setPatient({
          ...patient,
          prescription: {
            ...patient.prescription,
            text: e.target.value,
          },
        })
      }
    />
    <label htmlFor="prescriptionFile">
      Upload Prescription (PDF/Image):
    </label>
    <input
      type="file"
      accept=".pdf,.png,.jpg,.jpeg"
      onChange={(e) => {
        const file = e.target.files[0];
        if (file) {
          const fileUrl = `/prescriptions/${file.name}`;
          setPatient({
            ...patient,
            prescription: {
              ...patient.prescription,
              fileUrl,
            },
          });
          toast.info(
            "Prescription file URL set. Please place the file in /public/prescriptions/"
          );
        }
      }}
    />
    {patient.prescription?.fileUrl && (
      <p style={{ marginTop: "0.5rem" }}>
        ?{" "}
        <a
          href={patient.prescription.fileUrl}
          target="_blank"
          rel="noreferrer"
        >
          View Uploaded Prescription
        </a>
      </p>
    )}
    <button
      className="btn btn-prescription"
      style={{ marginTop: "1rem" }}
      onClick={async () => {
        if (
```

```jsx
                !patient.prescription?.text &&
                !patient.prescription?.fileUrl
              ) {
                toast.warn(
                  "Enter text or upload a file to save prescription."
                );
                return;
              }
              try {
                const res = await fetch(
                  `http://localhost:3001/patients/${patient.id}`,
                  {
                    method: "PATCH",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({
                      prescription: patient.prescription,
                    }),
                  }
                );
                if (res.ok) {
                  toast.success("Prescription saved.");
                  setPrescriptionSaved(true);
                } else {
                  toast.error("Failed to save prescription.");
                }
              } catch (error) {
                console.error("Save error:", error);
                toast.error("Error saving prescription.");
              }
            }}
          >
            ? Save Prescription
          </button>
        </div>
      )}

      {patient && (
        <div
          className="row"
          style={{ justifyContent: "center", marginTop: "1.5rem" }}
        >
          <button
            className="btn btn-physio"
            disabled={!prescriptionSaved || assignment.physio}
            onClick={() => handleAssignment("physio")}
          >
            <CheckCircle size={18} />
            {assignment.physio ? "Assigned to Physio" : "Assign to Physio"}
          </button>
          <button
            className="btn btn-dietitian"
            disabled={!prescriptionSaved || assignment.dietitian}
            onClick={() => handleAssignment("dietitian")}
          >
            <CheckCircle size={18} />
            {assignment.dietitian
              ? "Assigned to Dietitian"
              : "Assign to Dietitian"}
          </button>
        </div>
      )}

      {patient && (assignment.physio || assignment.dietitian) && (
        <>
          <div
            className="card"
            style={{ marginTop: "2rem", border: "2px solid var(--accent)" }}
          >
            <h3>? Patient Assigned</h3>
            <p>{patient.name} is currently assigned to:</p>
            <ul>
              {assignment.physio && (
                <li>
```

```
                    <strong>Physio</strong>
                  </li>
                )}
                {assignment.dietitian && (
                  <li>
                    <strong>Dietitian</strong>
                  </li>
                )}
              </ul>
            </div>
            <div style={{ marginTop: "1.5rem", textAlign: "center" }}>
              <button className="btn btn-primary" onClick={handleFinish}>
                Click To Proceed
              </button>
            </div>
          </>
        )}
      </div>
    </div>
  );
};

export default DoctorAssignDecision;
```

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);

  useEffect(() => {
    const fetchAssignedPatients = async () => {
      try {
        const res = await fetch("http://localhost:3001/patients");
        const data = await res.json();
        const filtered = data.filter(
          (p) => p.assignedTo?.physio || p.assignedTo?.dietitian
        );
        setAssignedPatients(filtered);
      } catch (error) {
        console.error("Error fetching assigned patients:", error);
      } finally {
        setLoading(false);
      }
    };
    fetchAssignedPatients();
  }, []);

  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>

        {assignedPatients.length === 0 ? (
          <div className="empty-state">
```

```jsx
                <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>Prescribed Medicine</th>
                      <th>Assigned To</th>
                      <th>Prescription File</th>
                    </tr>
                  </thead>
                  <tbody>
                    {assignedPatients.map((patient) => (
                      <tr key={patient.id}>
                        <td data-label="Name">{patient.name}</td>
                        <td data-label="MRN">{patient.mrn}</td>
                        <td data-label="Prescribed Medicine">
                          <span className="prescription-text">
                            {patient.prescription?.text || "-"}
                          </span>
                        </td>
                        <td data-label="Assigned To">
                          <div className="assignment-tags">
                            {patient.assignedTo.physio && (
                              <span className="assignment-tag physio">Physio</span>
                            )}
                            {patient.assignedTo.dietitian && (
                              <span className="assignment-tag dietitian">Dietitian</span>
                            )}
                          </div>
                        </td>
                        <td data-label="Prescription File">
                          {patient.prescription?.fileUrl ? (
                            <a
                              href={patient.prescription.fileUrl}
                              target="_blank"
                              rel="noopener noreferrer"
                              className="pdf-link"
                            >
                              <FileText size={18} /> View File
                            </a>
                          ) : (
                            <span className="no-file">No File</span>
                          )}
                        </td>
                      </tr>
                    ))}
                  </tbody>
                </table>
              </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, #6666
  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, #6666
  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-prima
}

.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, #6666
  font-size: 0.875rem;
}
```

```css
.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)'};
}

/* 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-pr
      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, #e0e
      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;
```

```css
      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\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>
```

```jsx
          </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
```

```jsx
                                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\DoctorUpcomingAppointments .js

```
import React, { useEffect, useState } from "react";
import { useTheme } from "../../ThemeProvider";
import { Eye } from "lucide-react";
import "../physio/assign.css";

const DoctorUpcomingAppointments = () => {
  const { theme } = useTheme();
  const [appointments, setAppointments] = useState([]);

  useEffect(() => {
    const fetchAppointments = async () => {
      try {
        const res = await fetch("http://localhost:3001/doctorappointments");
        const data = await res.json();
        setAppointments(data);
      } catch (error) {
        console.error("Failed to load appointments:", error);
      }
    };

    fetchAppointments();
  }, []);

  return (
    <div className={`assign-page ${theme}`}>
      <div className="assign-container">
        <h2 className="section-title">Upcoming Patient Appointments</h2>

        <div className="table-wrapper">
          <table className="appointments-table">
            <thead>
              <tr>
                <th>Patient Name</th>
                <th>MRN</th>
                <th>Appointment Date</th>
                <th>Report</th>
                <th>Samples Collected</th>
              </tr>
            </thead>
            <tbody>
              {appointments.map((appt, index) => (
                <tr key={index}>
                  <td data-label="Patient Name">{appt.name}</td>
                  <td data-label="MRN">{appt.mrn}</td>
                  <td data-label="Appointment Date">
                    {new Date(appt.appointmentDate).toLocaleDateString()}
                  </td>
                  <td data-label="Report">
                    <a
                      href={appt.reportPdfUrl}
                      target="_blank"
                      rel="noopener noreferrer"
                      className="report-link"
                    >
                      <Eye size={16} /> View Report
                    </a>
```

```jsx
            </td>
            <td data-label="Samples Collected">
              {appt.samples.join(", ")}
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>
 <style jsx>{`
  .table-wrapper {
    width: 100%;
    overflow-x: auto;
    margin-top: 1.5rem;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  }

  .appointments-table {
    width: 100%;
    border-collapse: collapse;
    min-width: 600px;
    background-color: ${theme === "dark"
      ? "var(--background-secondary, #1a1a1a)"
      : "var(--background-primary, #ffffff)"};
  }

  .appointments-table th,
  .appointments-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;
  }

  .appointments-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;
  }

  .appointments-table td {
    color: ${theme === "dark"
      ? "var(--text-secondary-dark, #cccccc)"
      : "var(--text-secondary-light, #666666)"};
    font-size: 0.875rem;
  }

  .appointments-table tr:hover {
    background-color: ${theme === "dark"
      ? "var(--hover-dark, #2a2a2a)"
      : "var(--hover-light, #f5f5f5)"};
    transition: background-color 0.2s ease;
  }

  .report-link {
    display: inline-flex;
    align-items: center;
    gap: 0.5rem;
```

```css
    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;
}

.report-link:hover {
  background-color: ${theme === "dark"
      ? "rgba(100, 181, 246, 0.1)"
      : "rgba(25, 118, 210, 0.1)"};
  text-decoration: none;
}

/* Tablet Styles */
@media (max-width: 1024px) {
  .table-wrapper {
    margin-top: 1rem;
  }

  .appointments-table {
    min-width: 500px;
  }

  .appointments-table th,
  .appointments-table td {
    padding: 0.75rem 0.5rem;
    font-size: 0.8rem;
  }
}

/* Mobile Styles */
@media (max-width: 768px) {
  .table-wrapper {
    overflow-x: visible;
    box-shadow: none;
  }

  .appointments-table,
  .appointments-table thead,
  .appointments-table tbody,
  .appointments-table th,
  .appointments-table td,
  .appointments-table tr {
    display: block;
  }

  .appointments-table {
    min-width: auto;
    background-color: transparent;
  }

  .appointments-table thead {
    display: none;
  }

  .appointments-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);
  }
```

```css
    .appointments-table tr:hover {
      background-color: ${theme === "dark"
        ? "var(--hover-dark, #2a2a2a)"
        : "var(--hover-light, #f5f5f5)"};
    }

    .appointments-table td {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 0.75rem 0;
      border-bottom: 1px solid
        ${theme === "dark"
          ? "var(--border-dark, #333)"
          : "var(--border-light, #e0e0e0)"};
      font-size: 0.875rem;
    }

    .appointments-table td:last-child {
      border-bottom: none;
    }

    .appointments-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;
    }

    .report-link {
      justify-content: flex-end;
      padding: 0.5rem 0.75rem;
      border-radius: 6px;
      font-size: 0.8rem;
    }
  }

  /* Small Mobile Styles */
  @media (max-width: 480px) {
    .assign-container {
      padding: 0.5rem;
    }

    .section-title {
      font-size: 1.25rem;
      margin-bottom: 1rem;
    }

    .appointments-table tr {
      padding: 0.75rem;
    }

    .appointments-table td {
      flex-direction: column;
      align-items: flex-start;
      gap: 0.25rem;
      padding: 0.5rem 0;
    }

    .appointments-table td::before {
      width: 100%;
      margin-bottom: 0.25rem;
    }

    .report-link {
      align-self: flex-end;
      margin-top: 0.25rem;
```

```
          }
        }

        /* Large Desktop Styles */
        @media (min-width: 1200px) {
          .appointments-table th,
          .appointments-table td {
            padding: 1.25rem 1rem;
          }

          .appointments-table {
            font-size: 0.9rem;
          }
        }

        /* Print Styles */
        @media print {
          .table-wrapper {
            box-shadow: none;
          }

          .appointments-table {
            border: 1px solid #000;
          }

          .appointments-table th,
          .appointments-table td {
            border: 1px solid #000;
            color: #000;
            background-color: #fff;
          }

          .report-link {
            color: #000;
          }
        }
      `}</style>
    </div>
  );
};

export default DoctorUpcomingAppointments;
```

File: src\dashboard\doctor\doctor_clientmanagement.js

```javascript
import React, { useState, useEffect } from "react";
import { useTheme } from "../../ThemeProvider";
import axios from "axios";
import "../physio/clientManagement.css";

export default function ClientManagement() {
  const { theme } = useTheme();
  const [clients, setClients] = useState([]);
  const [loading, setLoading] = useState(true);
  const [showModal, setShowModal] = useState(false);
  const [editIndex, setEditIndex] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [activeFilter, setActiveFilter] = useState("ALL");
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    status: "Active",
    type: "Client",
    resolution: "In Progress",
  });

  useEffect(() => {
    fetchClients();
  }, []);

  const fetchClients = async () => {
    try {
      const response = await axios.get("http://localhost:3001/clients");
      setClients(response.data);
```

```
    } catch (error) {
      console.error("Error fetching clients:", error);
    } finally {
      setLoading(false);
    }
  };

  const openModal = (client = null, index = null) => {
    if (client) {
      setFormData({
        ...client,
        type: client.type || "Client",
        resolution: client.resolution || "In Progress",
      });
      setEditIndex(index);
    } else {
      setFormData({
        name: "",
        email: "",
        status: "Active",
        type: "Client",
        resolution: "In Progress",
      });
      setEditIndex(null);
    }
    setShowModal(true);
  };

  const closeModal = () => {
    setShowModal(false);
  };

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

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

  const handleFilterChange = (filter) => {
    setActiveFilter(filter);
  };

  const handleSave = async () => {
    if (!formData.name || !formData.email) {
      alert("Please fill in all required fields");
      return;
    }

    try {
      if (editIndex !== null) {
        const updatedClient = {
          ...formData,
          avatar: clients[editIndex].avatar,
        };
        await axios.put(
          `http://localhost:3001/clients/${clients[editIndex].id}`,
          updatedClient
        );
        const updated = [...clients];
        updated[editIndex] = updatedClient;
        setClients(updated);
      } else {
        const newClient = {
          ...formData,
          avatar: `https://i.pravatar.cc/150?img=${clients.length + 6}`,
        };
        const response = await axios.post(
          "http://localhost:3001/clients",
          newClient
        );
```

```
        setClients([...clients, response.data]);
      }
      closeModal();
    } catch (error) {
      console.error("Error saving client:", error);
    }
  };

  const handleDelete = async (index) => {
    if (window.confirm("Are you sure you want to delete this client?")) {
      try {
        const clientId = clients[index].id;
        await axios.delete(`http://localhost:3001/clients/${clientId}`);
        const updated = [...clients];
        updated.splice(index, 1);
        setClients(updated);
      } catch (error) {
        console.error("Error deleting client:", error);
      }
    }
  };

  const markAsResolved = async (index) => {
    try {
      const updatedClient = {
        ...clients[index],
        resolution: "Resolved",
      };
      await axios.put(
        `http://localhost:3001/clients/${clients[index].id}`,
        updatedClient
      );
      const updated = [...clients];
      updated[index] = updatedClient;
      setClients(updated);
    } catch (error) {
      console.error("Error updating resolution status:", error);
    }
  };

  const filteredClients = clients.filter((client) => {
    // Filter by search term
    const matchesSearch =
      client.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
      client.email.toLowerCase().includes(searchTerm.toLowerCase());

    // Filter by status button
    const matchesFilter =
      activeFilter === "ALL" ||
      (activeFilter === "IN PROGRESS" && client.resolution === "In Progress") ||
      (activeFilter === "RESOLVED" && client.resolution === "Resolved");

    return matchesSearch && matchesFilter;
  });

  return (
    <div className={`client-page ${theme}`}>
      <div className="client-container">
        <div className="client-header">
          <h2>Client Management</h2>
          <div className="client-controls">
            {/* <div className="search-container">
              <input
                type="text"
                placeholder="Search clients..."
                value={searchTerm}
                onChange={handleSearchChange}
                className="search-input"
              />
            </div> */}
            <div className="filter-buttons">
              <button
                className={`filter-btn ${
```

```
              activeFilter === "ALL" ? "active" : ""
            }`}
            onClick={() => handleFilterChange("ALL")}
          >
            ALL
          </button>
          <button
            className={`filter-btn ${
              activeFilter === "IN PROGRESS" ? "active in-progress" : ""
            }`}
            onClick={() => handleFilterChange("IN PROGRESS")}
          >
            In Progress
          </button>
          <button
            className={`filter-btn ${
              activeFilter === "RESOLVED" ? "active" : ""
            }`}
            onClick={() => handleFilterChange("RESOLVED")}
          >
            Resolved
          </button>
        </div>
        <div className="action-buttons-right">
          <button
            className="add-consultant-btn"
            onClick={() => {
              setFormData({
                name: "",
                email: "",
                status: "Active",
                type: "Consultant",
                resolution: "In Progress",
              });
              setShowModal(true);
            }}
          >
            Add Consultant
          </button>
          <button className="add-btn" onClick={() => openModal()}>
            <span className="btn-icon">+</span> Add Client
          </button>
        </div>
      </div>
    </div>

    <div className="table-responsive">
      {loading ? (
        <p>Loading clients...</p>
      ) : filteredClients.length > 0 ? (
        <table
          className="client-table"
          style={{ border: "1px solid #cc5500" }}
        >
          <thead>
            <tr>
              <th>Name</th>
              <th className="hide-sm">Email</th>
              <th>Status</th>
              <th>Type</th>
              <th>Resolution</th>
              <th>Actions</th>
            </tr>
          </thead>
          <tbody>
            {filteredClients.map((client, i) => (
              <tr key={client.id}>
                <td className="client-name-cell">
                  <div className="client-info">
                    <img
                      src={client.avatar}
                      alt={client.name}
                      className="avatar"
```

```jsx
                    />
                    <div className="client-details">
                      <span className="client-name">{client.name}</span>
                      <span className="client-email-mobile">
                        {client.email}
                      </span>
                    </div>
                  </div>
                </td>
                <td className="hide-sm">{client.email}</td>
                <td>
                  <span
                    className={`status-badge ${client.status.toLowerCase()}`}
                  >
                    {client.status}
                  </span>
                </td>
                <td>
                  <span
                    className={`type-badge ${(
                      client.type || "Client"
                    ).toLowerCase()}`}
                  >
                    {client.type || "Client"}
                  </span>
                </td>
                <td>
                  <span
                    className={`resolution-badge ${(
                      client.resolution || "In Progress"
                    )
                      .toLowerCase()
                      .replace(" ", "-")}`}
                  >
                    {client.resolution || "In Progress"}
                  </span>
                </td>
                <td>
                  <div className="action-buttons">
                    <button
                      className="edit-btn"
                      onClick={() => openModal(client, i)}
                    >
                      <span className="btn-text">Edit</span>
                    </button>
                    {client.resolution !== "Resolved" && (
                      <button
                        className="resolve-btn"
                        onClick={() => markAsResolved(i)}
                      >
                        <span className="btn-text">Mark as Resolved</span>
                      </button>
                    )}
                    <button
                      className="delete-btn"
                      onClick={() => handleDelete(i)}
                    >
                      <span className="btn-text">Delete</span>
                    </button>
                  </div>
                </td>
              </tr>
          ))}
        </tbody>
      </table>
    ) : (
      <div className="no-results">
        <p>No clients found matching your search.</p>
      </div>
    )}
  </div>

  {showModal && (
```

```
<div className="modal-overlay" onClick={closeModal}>
  <div className="modal-box" onClick={(e) => e.stopPropagation()}>
    <div className="modal-header">
      <h3>
        {editIndex !== null
          ? "Edit Client"
          : `Add New ${formData.type}`}
      </h3>
      <button className="close-modal" onClick={closeModal}>
        ×
      </button>
    </div>
    <div className="modal-body">
      <div className="form-group">
        <label htmlFor="name">Name</label>
        <input
          type="text"
          id="name"
          name="name"
          placeholder={`Enter ${formData.type.toLowerCase()} name`}
          value={formData.name}
          onChange={handleInputChange}
          required
        />
      </div>
      <div className="form-group">
        <label htmlFor="email">Email</label>
        <input
          type="email"
          id="email"
          name="email"
          placeholder={`Enter ${formData.type.toLowerCase()} email`}
          value={formData.email}
          onChange={handleInputChange}
          required
        />
      </div>
      <div className="form-group">
        <label htmlFor="status">Status</label>
        <select
          id="status"
          name="status"
          value={formData.status}
          onChange={handleInputChange}
        >
          <option value="Active">Active</option>
          <option value="Inactive">Inactive</option>
        </select>
      </div>
      <div className="form-group">
        <label htmlFor="type">Type</label>
        <select
          id="type"
          name="type"
          value={formData.type}
          onChange={handleInputChange}
        >
          <option value="Client">Client</option>
          <option value="Consultant">Consultant</option>
        </select>
      </div>
      <div className="form-group">
        <label htmlFor="resolution">Resolution Status</label>
        <select
          id="resolution"
          name="resolution"
          value={formData.resolution}
          onChange={handleInputChange}
        >
          <option value="In Progress">In Progress</option>
          <option value="Resolved">Resolved</option>
        </select>
      </div>
```

```
            </div>
            <div className="modal-actions">
              <button className="cancel-btn" onClick={closeModal}>
                Cancel
              </button>
              <button className="save-btn" onClick={handleSave}>
                Save
              </button>
            </div>
          </div>
        </div>
      )}
    </div>
  </div>
  );
}
```

File: src\dashboard\doctor\doctor_physioo.js

```javascript
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
```

```javascript
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>
```

```jsx
        </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\doctor\doctor_profile.js

```javascript
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)",
```

```jsx
          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: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>
```

```jsx
              <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
```

```jsx
      </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" }}>
```

```jsx
      <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",
                  }}
                >
```

```jsx
                    <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" }}
```

```jsx
            >
            <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>{" "}
{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>
  )}
```

```jsx
{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
```

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

export default Profile;
```

File: src\dashboard\doctor\doctor_report.js

```jsx
import React, { useState } from "react";
import axios from "axios";
import "../physio/report.css";

const Report = () => {
  const [mrn, setMrn] = useState("");
  const [searchedMrn, setSearchedMrn] = useState("");
  const [pdfUrl, setPdfUrl] = useState("");
  const [loading, setLoading] = useState(false);
  const [notFound, setNotFound] = useState(false);

  const handleSearch = async () => {
    const formattedMrn = mrn.trim().toUpperCase();
    if (!formattedMrn) return;

    setLoading(true);
    setNotFound(false);
    setPdfUrl(""); // reset before new search

    try {
      const response = await axios.get(
        `http://localhost:3001/reports?mrn=${formattedMrn}`
      );
      if (response.data.length > 0) {
        // Get the relative path from the db.json file
        const relativePath = response.data[0].pdfUrl;

        // Create the full path by prepending the public path
        // The public folder is served at the root of your app
        const fullPath = relativePath.startsWith("/")
          ? relativePath
          : `/${relativePath}`;

        setPdfUrl(fullPath);
        setSearchedMrn(formattedMrn);
      } else {
```

```
        setNotFound(true);
      }
    } catch (error) {
      console.error("Error fetching report:", error);
      setNotFound(true);
    } finally {
      setLoading(false);
    }
  };

  const handleRemove = () => {
    setSearchedMrn("");
    setMrn("");
    setPdfUrl("");
    setNotFound(false);
  };

  const handleMrnChange = (e) => {
    setMrn(e.target.value.toUpperCase());
  };

  return (
    <div className="report-body standalone-report">
      <div className="report-card">
        <div className="content-area">
          <h2>Patient Reports</h2>
          <p className="page-description">
            This page allows you to view and download client reports by entering
            the MRN (Medical Record Number). Please ensure the MRN is accurate
            to retrieve the correct PDF report.
          </p>
          <div className="search-container">
            <input
              type="text"
              placeholder="Enter MRN"
              value={mrn}
              onChange={handleMrnChange}
              className="uppercase-input"
            />
            <button onClick={handleSearch}>Search</button>
          </div>

          {loading && <p>Loading report...</p>}

          {searchedMrn && (
            <div className="pdf-container">
              <h3>Report for MRN: {searchedMrn}</h3>
              {pdfUrl ? (
                <>
                  <iframe
                    src={pdfUrl}
                    width="100%"
                    height="600px"
                    title="Patient Report"
                    style={{ border: "1px solid #ccc", borderRadius: "6px" }}
                  />
                  <button onClick={handleRemove} className="remove-button">
                    Remove
                  </button>
                </>
              ) : (
                <p>PDF not found for this MRN.</p>
              )}
            </div>
          )}

          {notFound && !loading && (
            <p className="error-text">No report found for MRN: {mrn}</p>
          )}
        </div>
      </div>
    </div>
  );
```

```
};

export default Report;
```

File: src\dashboard\doctor\doctor_userdetails.js
```javascript
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import axios from "axios";
import "../physio/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:3001/users/${userId}`
        );

        const fitnessResponse = await axios.get(
          `http://localhost:3001/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" },
```

```jsx
    };

    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>
</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\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/doctor_profile"> <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,
} from "lucide-react";
```

```javascript
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);
  };
```

```jsx
    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">Doctor</span>
            </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",
                    }}
                  />
```

```jsx
              <span>Dashboard</span>
          </NavLink>
      </li>
      <li>
          <NavLink to="/doctor/doctor_clients" className={navLinkClass}>
              <Users
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Client Management</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/DoctorAssignDecision"
            className={navLinkClass}
          >
              <Hospital
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Assign Patient</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/DoctorUpcomingAppointments"
            className={navLinkClass}
          >
              <BarChart2
                size={18}
                style={{
                  marginRight: !collapsed || isHovering ? "10px" : "0",
                }}
              />
              <span>Report</span>
          </NavLink>
      </li>
      <li>
          <NavLink
            to="/doctor/DoctorPasswordRequest"
```

```
                          className={navLinkClass}
                        >
                          <BarChart2
                            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 DoctorSidebar;
```

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();
```

```jsx
    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">
        <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\master_admin\ActivityLogs.js

```javascript
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,
```

```jsx
        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>
```

```jsx
              <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

```jsx
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
```

```jsx
              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

```javascript
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;
```

```css
    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

```javascript
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} />
```

```jsx
<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={{
```

```jsx
                              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\DietitianAppointments.js

```jsx
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" ? (
```

```jsx
                        <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);
  };
```

```jsx
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;
}
```

```css
.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);
```

```css
}

.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;
```

```css
  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;
}
/* 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;
}
```

```css
.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;
```

```css
  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,
```

```css
.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);
```

```css
    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;
```

```css
}

/* .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;
```

```css
  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 {
```

```css
  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;
```

```css
  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;
```

```css
}

.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 {
```

```css
  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;
}
```

```css
.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;
}
```

File: src\dashboard\master_admin\master_admin_dashboard.js

```javascript
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>
```

```jsx
          <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"
          style={{
            display: "flex",
            flexWrap: "wrap",
            gap: "20px",
            justifyContent: "flex-start",
            marginBottom: "20px",
          }}
        >
          {metrics.roleCounts.map((r, i) => (
            <div className="card"
              key={i}
              style={{
                padding: "12px",
                minWidth: "246px",
                maxWidth: "160px",
                border: "1px solid #cc5500",
                backgroundColor: "transparent",
                textAlign: "center",
                fontSize: "20px",
                borderRadius: "8px",
```

```
        flex: "0 0 auto",
      }}
    >
      <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>
  </div>
  );
};

export default BasicPage;
```

File: src\dashboard\master_admin\PatientJourney.js
```
import React, { useState } from "react";
import axios from "axios";
```

```javascript
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>
```

```jsx
{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>
```

```jsx
                    <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

```jsx
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>
```

```jsx
                {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" : "badg
                        }`}
                        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\SecurityControls.js

```jsx
import React, { useState, useEffect } from "react";
import axios from "axios";
import { toast, ToastContainer } from "react-toastify";
import { useTheme } from "../../ThemeProvider";
```

```javascript
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

```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;
}
```

```css
/* 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"],
```

```css
.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;
}
```

```css
/* 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;
}
```

```css
.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%);
```

```css
  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 {
```

```css
    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

```javascript
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 Services = () => {
  const { theme } = useTheme();
  const [services, setServices] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [selectedService, setSelectedService] = useState(null);
  const [editingServiceId, setEditingServiceId] = useState(null);
  const formRef = useRef(null);

  const [formData, setFormData] = useState({
    banner: "",
    title: "",
    subtitle: "",
    priceMonthly: "",
    priceQuarterly: "",
    priceYearly: "",
    rating: "",
    description: "",
    previewVideo: "",
    features: [""],
  });

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

  useEffect(() => {
    fetch(API_URL)
      .then((res) => res.json())
      .then((data) => setServices(data))
      .catch((err) => console.error("Error fetching services:", err));
  }, []);

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

  const handleFeatureChange = (index, value) => {
    const updated = [...formData.features];
    updated[index] = value;
    setFormData((prev) => ({ ...prev, features: updated }));
  };

  const addFeature = () =>
    setFormData((prev) => ({ ...prev, features: [...prev.features, ""] }));
```

```
const removeFeature = (index) => {
  const updated = formData.features.filter((_, i) => i !== index);
  setFormData((prev) => ({ ...prev, features: updated }));
};

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 method = editingServiceId ? "PUT" : "POST";
  const url = editingServiceId
    ? `${API_URL}/${editingServiceId}`
    : API_URL;

  fetch(url, {
    method,
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(formData),
  })
    .then((res) => res.json())
    .then((data) => {
      if (editingServiceId) {
        setServices((prev) =>
          prev.map((s) => (s.id === editingServiceId ? data : s))
        );
        toast.success("Service updated successfully!");
      } else {
        setServices([...services, data]);
        toast.success("Service added successfully!");
      }
      resetForm();
      setShowForm(false);
    })
    .catch(() => toast.error("Error saving service."));
};

const resetForm = () => {
  setFormData({
    banner: "",
    title: "",
    subtitle: "",
    priceMonthly: "",
    priceQuarterly: "",
    priceYearly: "",
    rating: "",
    description: "",
    previewVideo: "",
    features: [""],
  });
  setEditingServiceId(null);
};

const handleDelete = (id) => {
  toast.info(
    <div>
      <p style={{ color: "black" }}>Are you sure you want to delete this service?</p>
      <div style={{ display: "flex", gap: "10px", marginTop: "10px" }}>
        <button
          className="btn btn-primary"
          onClick={() => {
            fetch(`${API_URL}/${id}`, { method: "DELETE" }).then(() => {
              setServices((prev) => prev.filter((s) => s.id !== id));
              toast.dismiss();
              toast.error("Service deleted.");
            });
          }}
        >
```

```jsx
            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,
      }
    );
  };

  const handleEdit = (service) => {
    setFormData(service);
    setEditingServiceId(service.id);
    setShowForm(true);
    setTimeout(() => {
      formRef.current?.scrollIntoView({ behavior: "smooth" });
    }, 100);
  };

  return (
    <div className={`dashboard-main ${theme}`}>
      <ToastContainer position="top-center" autoClose={3000} />
      <h1>Manage Services</h1>

      <button className="btn btn-primary" onClick={() => {
        setShowForm(!showForm);
        if (!showForm) resetForm();
      }}>
        {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 && (
          <form onSubmit={handleSubmit} className="card" style={{ marginTop: "20px" }}>
            <label>Service Banner</label>
            <input type="file" accept="image/*" onChange={handleBannerUpload} required />
            {formData.banner && (
              <img src={formData.banner} alt="preview" style={{ maxHeight: "150px", marginTop: "10px" }} />
            )}

            <label>Title</label>
            <input name="title" value={formData.title} onChange={handleChange} required />
            <label>Sub-Title</label>
            <input name="subtitle" value={formData.subtitle} onChange={handleChange} required />

            <label>Pricing Options</label>
            <div className="pricing-row">
              <input name="priceMonthly" type="number" placeholder="1 Month" value={formData.priceMonthly} onChan
              <input name="priceQuarterly" type="number" placeholder="3 Month" value={formData.priceQuarterly} o
              <input name="priceYearly" type="number" placeholder="6 Month" value={formData.priceYearly} onChange
            </div>

            <label>Rating</label>
```

```jsx
              <input name="rating" type="number" step="0.1" max="5" value={formData.rating} onChange={handleChange
              <label>Description</label>
              <textarea name="description" value={formData.description} onChange={handleChange} rows={4} required
              <label>Preview Video (YouTube)</label>
              <input name="previewVideo" value={formData.previewVideo} onChange={handleChange} required />

              <label>Program Features</label>
              {formData.features.map((f, i) => (
                <div key={i} className="d-flex">
                  <input value={f} onChange={(e) => handleFeatureChange(i, e.target.value)} placeholder={`Feature
                  {formData.features.length > 1 && (
                    <button type="button" className="btn btn-primary" onClick={() => removeFeature(i)}>Remove</but
                  )}
                </div>
              ))}
              <button type="button" className="btn btn-primary" onClick={addFeature}>Add Feature</button>

              <div className="center-btn">
                <button type="submit" className="btn btn-primary">
                  {editingServiceId ? "Update Service" : "Submit Service"}
                </button>
              </div>
          </form>
        )}
      </div>

      <h2 style={{ marginTop: "30px" }}>Added Services</h2>
      {services.length === 0 ? (
        <p>No services added</p>
      ) : (
        <div className="blog-grid">
          {services.map((s) => (
            <div key={s.id} className="card blog-card">
              <img src={s.banner} alt="banner" style={{ height: "100px", objectFit: "cover", width: "100%" }} />
              <h3>{s.title}</h3>
              <p><strong>{s.description}</strong></p>
              <p><em>{s.subtitle}</em></p>
              <div className="center-btn">
                <button className="btn btn-primary" onClick={() => setSelectedService(s)}>View</button>
                <button className="btn btn-primary" onClick={() => handleEdit(s)}>Edit</button>
                <button className="btn btn-primary" onClick={() => handleDelete(s.id)}>Delete</button>
              </div>
            </div>
          ))}
        </div>
      )}

      {selectedService && (
        <div className="modal-overlay" style={{ border: "1px solid #cc5500" }} onClick={() => setSelectedService
          <div className="modals-box" style={{ border: "1px solid #cc5500" }} onClick={(e) => e.stopPropagation(
            <h2>{selectedService.title}</h2>
            <img src={selectedService.banner} alt="service-banner" style={{ width: "100%", maxHeight: "200px", o

            <table className="popup-table">
              <tbody>
                <tr><td><strong>Sub-Title:</strong></td><td>{selectedService.subtitle}</td></tr>
                <tr><td><strong>1 Month:</strong></td><td>?{selectedService.priceMonthly}</td></tr>
                <tr><td><strong>3 Month:</strong></td><td>?{selectedService.priceQuarterly}</td></tr>
                <tr><td><strong>6 Month:</strong></td><td>?{selectedService.priceYearly}</td></tr>
                <tr><td><strong>Rating:</strong></td><td>{selectedService.rating}</td></tr>
                <tr><td><strong>Description:</strong></td><td>{selectedService.description}</td></tr>
              </tbody>
            </table>

            <iframe
              width="100%"
              height="200"
              src={selectedService.previewVideo.replace("watch?v=", "embed/")}
              title="Preview Video"
              frameBorder="0"
              allowFullScreen
              style={{ margin: "20px 0" }}
            />
```

```
            <strong>Program Features:</strong>
            <ul style={{ paddingLeft: "10px" }}>
              {selectedService.features.map((f, i) => (
                <li key={i}>{f}</li>
              ))}
            </ul>

            <div className="center-btn" style={{ marginTop: "20px" }}>
              <center><button className="btn btn-primary" onClick={() => setSelectedService(null)}>Close</button
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default Services;
```

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"
```

```jsx
    ).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>
```

```jsx
        </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>
```

```jsx
              <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\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 [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 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());
```

```jsx
    return matchSearch && matchRole;
  });

  return (
    <div className={`dashboard-main ${theme}`}>
      <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>
```

```jsx
      </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

```javascript
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

```javascript
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

```javascript
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">
```

```jsx
              <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/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\physio\assign.css

```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;
}
```

```css
/* 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 */
```

```css
  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);
}
```

```css
.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%;
}
```

```css
.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);
```

```css
    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;
```

```css
}

/* 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);
}

/* 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%;
  }
```

```css
  }

  @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);
```

```css
    }

    .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;
  }
```

```css
  .assign-page {
    background: white;
  }

  .table {
    border: 1px solid #ddd;
  }
}
```

File: src\dashboard\physio\assign.js

```javascript
import React, { useState } from "react";
import { useTheme } from "../../ThemeProvider";
import {
  Search,
  Upload,
  Eye,
  Trash2,
  Calendar,
  FileCheck,
  X,
  Download,
  FileType,
  CheckCircle,
  Repeat,
} from "lucide-react";
import { toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import "./assign.css";

const Assign = () => {
  const { theme } = useTheme();
  const [searchMRN, setSearchMRN] = useState("");
  const [patient, setPatient] = 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 [confirmAction, setConfirmAction] = useState(null);

  const handleSearch = async () => {
    if (!searchMRN.trim()) {
      toast.warn("Please enter an MRN.");
      return;
    }

    try {
      const res = await fetch(
        `http://localhost:8080/api/patients?mrn=${searchMRN}`
      );
      if (!res.ok) throw new Error("Fetch failed");

      const found = await res.json();
      setPatient(found);

      const chartData = found.charts || {};
      const newFiles = Array(15).fill(null);
      const newFeedback = Array(15).fill("");
      const newPdfs = Array(15).fill(null);

      Object.entries(chartData).forEach(([index, value]) => {
        const [filename, fb] = value.split("|");
        const idx = parseInt(index);
        newFiles[idx] = { name: filename };
        newFeedback[idx] = fb || "";
        newPdfs[idx] = `/prescriptions/${filename}`;
      });

      setUploadedFiles(newFiles);
      setFeedback(newFeedback);
      setPdfs(newPdfs);
      toast.success("Patient data loaded.");
    } catch (err) {
```

```javascript
      console.error(err);
      toast.error("Patient not found or error fetching data.");
      resetAll();
    }
  };

  const resetAll = () => {
    setPatient(null);
    setUploadedFiles(Array(15).fill(null));
    setFeedback(Array(15).fill(""));
    setPdfs(Array(15).fill(null));
    setViewedPdf(null);
    setViewedFileType(null);
    setConfirmAction(null);
  };

  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 viewChartData = (index) => {
    setViewedPdf(pdfs[index]);
    setViewedFileType(
      uploadedFiles[index]?.type?.includes("pdf") ? "pdf" : "excel"
    );
  };

  const removeChartData = (index) => {
    setUploadedFiles((prev) => {
      const updated = [...prev];
      updated[index] = null;
      return updated;
    });
    setPdfs((prev) => {
      const updated = [...prev];
      updated[index] = null;
      return updated;
    });
    setFeedback((prev) => {
      const updated = [...prev];
      updated[index] = "";
      return updated;
    });
    setViewedPdf(null);
  };

  const getDisplayFileName = (file) => {
    if (!file) return "Upload";
    const maxLength = 15;
    return file.name.length > maxLength
      ? file.name.substring(0, maxLength - 3) + "..."
      : file.name;
```

```jsx
};

const showConfirmationToast = (index, type) => {
  setConfirmAction({ type, index });

  toast.info(
    ({ closeToast }) => (
      <div>
        <p>
          Are you sure you want to {type} Chart {index + 1}?
        </p>
        <div style={{ marginTop: "0.5rem", display: "flex", gap: "0.5rem" }}>
          <button
            className="btn btn-primary"
            onClick={() => {
              if (type === "remove") handleConfirmedRemove(index);
              if (type === "reassign") handleConfirmedReassign(index);
              toast.dismiss();
            }}
          >
            Confirm
          </button>
          <button
            className="btn btn-primary"
            onClick={() => {
              toast.dismiss();
              toast.info("Action cancelled.");
            }}
          >
            Cancel
          </button>
        </div>
      </div>
    ),
    { autoClose: false }
  );
};

const handleRemove = (index) => {
  showConfirmationToast(index, "remove");
};

const handleReassign = (index) => {
  showConfirmationToast(index, "reassign");
};

const handleConfirmedRemove = (index) => {
  removeChartData(index);
  toast.success(`Chart ${index + 1} removed.`);
};

const handleConfirmedReassign = (index) => {
  removeChartData(index);
  toast.success(`Chart ${index + 1} is ready for reassignment.`);
};

const handleConfirmAssignment = async () => {
  if (!patient) return;

  try {
    const chartMap = {};
    uploadedFiles.forEach((file, i) => {
      if (file && file.name) {
        chartMap[i.toString()] = `${file.name}|${feedback[i] || ""}`;
      }
    });

    const patchData = {
      charts: chartMap,
      assignedTo: {
        physio: true,
      },
    };
```

```
      const res = await fetch(
        `http://localhost:8080/api/patients/${patient.id}`,
        {
          method: "PATCH",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify(patchData),
        }
      );

      if (res.ok) {
        toast.success("Patient assignment saved successfully.");
        resetAll();
        setSearchMRN("");
      } else {
        toast.error("Failed to save assignment.");
      }
    } catch (err) {
      console.error(err);
      toast.error("Error saving data.");
    }
  };

  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.toUpperCase())}
                onKeyDown={(e) => e.key === "Enter" && handleSearch()}
              />
              <button className="search-button" onClick={handleSearch}>
                <Search size={16} />
                <span className="button-text">Search</span>
              </button>
            </div>

            {patient && (
              <div className="user-info">
                <p>
                  <strong>Name:</strong> {patient.name}
                </p>
                <p>
                  <strong>Age:</strong> {patient.age}
                </p>
                <p>
                  <strong>Gender:</strong> {patient.gender}
                </p>
                <p>
                  <strong>Weight:</strong> {patient.weight}
                </p>
                <p>
                  <strong>Product:</strong> {patient.product}
                </p>
              </div>
            )}
          </div>

          {patient?.reportPdfUrl && (
            <div className={`assign-card ${theme === "dark" ? "dark" : ""}`}>
              <h2 className="section-title">Client Report</h2>
              <div className="pdf-viewer-container">
                <iframe
                  src={patient.reportPdfUrl}
                  title="Lab Report"
                  width="100%"
```

```
              height="400px"
              style={{
                backgroundColor: theme === "dark" ? "#1e1e1e" : "#fff",
                border: "1px solid var(--border)",
                borderRadius: "8px",
              }}
            ></iframe>

            <div style={{ marginTop: "0.5rem" }}>
              <a
                href={patient.reportPdfUrl}
                target="_blank"
                rel="noopener noreferrer"
                className="open-pdf-link"
              >
                ? Open PDF in New Tab
              </a>
            </div>
          </div>
        </div>
      )}
    </div>
  </div>

  {patient && (
    <>
      <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-${i}`}
                        className="upload-label"
                        title={uploadedFiles[i]?.name}
                      >
                        <Upload size={16} />
                        <span className="file-name">
                          {getDisplayFileName(uploadedFiles[i])}
                        </span>
                      </label>
                      <input
                        id={`upload-${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-primary"
                        onClick={() => viewChartData(i)}
                        disabled={!uploadedFiles[i]}
                      >
                        <Eye size={16} />
                      </button>
                      <button
                        className="btn btn-primary"
                        onClick={() => handleRemove(i)}
                        disabled={!uploadedFiles[i]}
                      >
                        <Trash2 size={16} />
                      </button>
                      <button
                        className="btn btn-primary"
                        onClick={() => handleReassign(i)}
                        disabled={!uploadedFiles[i]}
                      >
                        <Repeat size={16} />
                      </button>
                    </div>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>

      <div style={{ textAlign: "center", marginTop: "2rem" }}>
        <button
          className="btn btn-primary"
          onClick={handleConfirmAssignment}
        >
          <CheckCircle size={18} style={{ marginRight: "8px" }} />
          Confirm Assignment
        </button>
      </div>
    </>
  )}

  {viewedPdf && (
    <div className={`modal ${theme}`}>
      <div
        className="modal-content"
        style={{
          border: `1px solid #cc5500`,
          backgroundColor: theme === "dark" ? "#1e1e1e" : "#fff",
          color: theme === "dark" ? "#f1f1f1" : "#000",
        }}
      >
        <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"
                      style={{
                        backgroundColor: theme === "dark" ? "#2a2a2a" : "#fff",
                        borderRadius: "6px",
                        border: "1px solid var(--border)",
                      }}
                    />
                  ) : (
                    <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\physio\clientManagement.css

```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;
```

```css
    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;
}
```

```css
.filter-btn.active {
  background-color: var(--brand-color);
  color: white;
  border-color: var(--brand-color);
}

/* 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;
}
```

```css
.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;
```

```css
    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;
```

```css
  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,
```

```css
.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;
```

```css
    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\clientManagement.js

```javascript
import React, { useState, useEffect } from "react";
import { useTheme } from "../../ThemeProvider";
import axios from "axios";
import "./clientManagement.css";

export default function ClientManagement() {
  const { theme } = useTheme();
  const [clients, setClients] = useState([]);
  const [loading, setLoading] = useState(true);
  const [showModal, setShowModal] = useState(false);
  const [editIndex, setEditIndex] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [activeFilter, setActiveFilter] = useState("ALL");
```

```javascript
const [formData, setFormData] = useState({
  name: "",
  email: "",
  status: "Active",
  type: "Client",
  resolution: "In Progress",
});

useEffect(() => {
  fetchClients();
}, []);

const fetchClients = async () => {
  try {
    const response = await axios.get("http://localhost:5000/clients");
    setClients(response.data);
  } catch (error) {
    console.error("Error fetching clients:", error);
  } finally {
    setLoading(false);
  }
};

const openModal = (client = null, index = null) => {
  if (client) {
    setFormData({
      ...client,
      type: client.type || "Client",
      resolution: client.resolution || "In Progress",
    });
    setEditIndex(index);
  } else {
    setFormData({
      name: "",
      email: "",
      status: "Active",
      type: "Client",
      resolution: "In Progress",
    });
    setEditIndex(null);
  }
  setShowModal(true);
};

const closeModal = () => {
  setShowModal(false);
};

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

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

const handleFilterChange = (filter) => {
  setActiveFilter(filter);
};

const handleSave = async () => {
  if (!formData.name || !formData.email) {
    alert("Please fill in all required fields");
    return;
  }

  try {
    if (editIndex !== null) {
      const updatedClient = {
        ...formData,
        avatar: clients[editIndex].avatar,
      };
```

```
      await axios.put(
        `http://localhost:5000/clients/${clients[editIndex].id}`,
        updatedClient
      );
      const updated = [...clients];
      updated[editIndex] = updatedClient;
      setClients(updated);
    } else {
      const newClient = {
        ...formData,
        avatar: `https://i.pravatar.cc/150?img=${clients.length + 6}`,
      };
      const response = await axios.post(
        "http://localhost:5000/clients",
        newClient
      );
      setClients([...clients, response.data]);
    }
    closeModal();
  } catch (error) {
    console.error("Error saving client:", error);
  }
};

const handleDelete = async (index) => {
  if (window.confirm("Are you sure you want to delete this client?")) {
    try {
      const clientId = clients[index].id;
      await axios.delete(`http://localhost:5000/clients/${clientId}`);
      const updated = [...clients];
      updated.splice(index, 1);
      setClients(updated);
    } catch (error) {
      console.error("Error deleting client:", error);
    }
  }
};

const markAsResolved = async (index) => {
  try {
    const updatedClient = {
      ...clients[index],
      resolution: "Resolved",
    };
    await axios.put(
      `http://localhost:5000/clients/${clients[index].id}`,
      updatedClient
    );
    const updated = [...clients];
    updated[index] = updatedClient;
    setClients(updated);
  } catch (error) {
    console.error("Error updating resolution status:", error);
  }
};

const filteredClients = clients.filter((client) => {
  // Filter by search term
  const matchesSearch =
    client.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
    client.email.toLowerCase().includes(searchTerm.toLowerCase());

  // Filter by status button
  const matchesFilter =
    activeFilter === "ALL" ||
    (activeFilter === "IN PROGRESS" && client.resolution === "In Progress") ||
    (activeFilter === "RESOLVED" && client.resolution === "Resolved");

  return matchesSearch && matchesFilter;
});

return (
  <div className={`client-page ${theme}`}>
```

```jsx
<div className="client-container">
  <div className="client-header">
    <h2>Client Management</h2>
    <div className="client-controls">
      {/* <div className="search-container">
        <input
          type="text"
          placeholder="Search clients..."
          value={searchTerm}
          onChange={handleSearchChange}
          className="search-input"
        />
      </div> */}
      <div className="filter-buttons">
        <button
          className={`filter-btn ${
            activeFilter === "ALL" ? "active" : ""
          }`}
          onClick={() => handleFilterChange("ALL")}
        >
          ALL
        </button>
        <button
          className={`filter-btn ${
            activeFilter === "IN PROGRESS" ? "active in-progress" : ""
          }`}
          onClick={() => handleFilterChange("IN PROGRESS")}
        >
          In Progress
        </button>
        <button
          className={`filter-btn ${
            activeFilter === "RESOLVED" ? "active" : ""
          }`}
          onClick={() => handleFilterChange("RESOLVED")}
        >
          Resolved
        </button>
      </div>
      <div className="action-buttons-right">
        <button
          className="btn btn-primary"
          onClick={() => {
            setFormData({
              name: "",
              email: "",
              status: "Active",
              type: "Consultant",
              resolution: "In Progress",
            });
            setShowModal(true);
          }}
        >
          Add Consultant
        </button>
        <button className="btn btn-primary" onClick={() => openModal()}>
          <span className="btn-icon">+</span> Add Client
        </button>
      </div>
    </div>
  </div>

  <div className="table-responsive">
    {loading ? (
      <p>Loading clients...</p>
    ) : filteredClients.length > 0 ? (
      <table
        className="client-table"
        style={{ border: "1px solid #cc5500" }}
      >
        <thead>
          <tr>
            <th>Name</th>
```

```
                <th className="hide-sm">Email</th>
                <th>Status</th>
                <th>Type</th>
                <th>Resolution</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {filteredClients.map((client, i) => (
                <tr key={client.id}>
                  <td className="client-name-cell">
                    <div className="client-info">
                      <img
                        src={client.avatar}
                        alt={client.name}
                        className="avatar"
                      />
                      <div className="client-details">
                        <span className="client-name">{client.name}</span>
                        <span className="client-email-mobile">
                          {client.email}
                        </span>
                      </div>
                    </div>
                  </td>
                  <td className="hide-sm">{client.email}</td>
                  <td>
                    <span
                      className={`status-badge ${client.status.toLowerCase()}`}
                    >
                      {client.status}
                    </span>
                  </td>
                  <td>
                    <span
                      className={`type-badge ${(
                        client.type || "Client"
                      ).toLowerCase()}`}
                    >
                      {client.type || "Client"}
                    </span>
                  </td>
                  <td>
                    <span
                      className={`resolution-badge ${(
                        client.resolution || "In Progress"
                      )
                        .toLowerCase()
                        .replace(" ", "-")}`}
                    >
                      {client.resolution || "In Progress"}
                    </span>
                  </td>
                  <td>
                    <div className="action-buttons">
                      <button
                        className="edit-btn"
                        onClick={() => openModal(client, i)}
                      >
                        <span className="btn-text">Edit</span>
                      </button>
                      {client.resolution !== "Resolved" && (
                        <button
                          className="resolve-btn"
                          onClick={() => markAsResolved(i)}
                        >
                          <span className="btn-text">Mark as Resolved</span>
                        </button>
                      )}
                      <button
                        className="delete-btn"
                        onClick={() => handleDelete(i)}
                      >
```

```
                  <span className="btn-text">Delete</span>
                </button>
              </div>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  ) : (
    <div className="no-results">
      <p>No clients found matching your search.</p>
    </div>
  )}
</div>

{showModal && (
  <div className="modal-overlay" onClick={closeModal}>
    <div className="modal-box" onClick={(e) => e.stopPropagation()}>
      <div className="modal-header">
        <h3>
          {editIndex !== null
            ? "Edit Client"
            : `Add New ${formData.type}`}
        </h3>
        <button className="close-modal" onClick={closeModal}>
          ×
        </button>
      </div>
      <div className="modal-body">
        <div className="form-group">
          <label htmlFor="name">Name</label>
          <input
            type="text"
            id="name"
            name="name"
            placeholder={`Enter ${formData.type.toLowerCase()} name`}
            value={formData.name}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="email">Email</label>
          <input
            type="email"
            id="email"
            name="email"
            placeholder={`Enter ${formData.type.toLowerCase()} email`}
            value={formData.email}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="status">Status</label>
          <select
            id="status"
            name="status"
            value={formData.status}
            onChange={handleInputChange}
          >
            <option value="Active">Active</option>
            <option value="Inactive">Inactive</option>
          </select>
        </div>
        <div className="form-group">
          <label htmlFor="type">Type</label>
          <select
            id="type"
            name="type"
            value={formData.type}
            onChange={handleInputChange}
          >
```

```
                        <option value="Client">Client</option>
                        <option value="Consultant">Consultant</option>
                      </select>
                    </div>
                    <div className="form-group">
                      <label htmlFor="resolution">Resolution Status</label>
                      <select
                        id="resolution"
                        name="resolution"
                        value={formData.resolution}
                        onChange={handleInputChange}
                      >
                        <option value="In Progress">In Progress</option>
                        <option value="Resolved">Resolved</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>
        </div>
      );
    }
```

File: src\dashboard\physio\physio.css

```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 {
```

```css
    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;
```

```css
    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;    /* Prevent text from wrapping */
    min-width: 80px;         /* Ensure a minimum width */
    text-align: center;      /* Center the button text */
  }


  /* .btn:hover {
    background-color: var(--accent-hover);
  } */

  .content-container {
    padding: 2rem;
    text-align: center;
  }

  .loading-spinner,
  .error-message {
    font-size: 1.2rem;
    color: var(--accent);
    font-weight: bold;
  }

  /* Responsive adjustments */
  @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\physioo.js

```javascript
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.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:5000/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;
      }
```

```jsx
    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\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);
    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);
    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))
                )
                .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>
    );

    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>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}
                        </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"
                                    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 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

```javascript
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>
```

```jsx
        </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>

<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>
</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"
```

```jsx
          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" }}>
```

```jsx
          <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`
                  )
                }
```

```jsx
                  >
                    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>{" "}
    {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>
```

```jsx
            <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

```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;
}
```

```css
.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\report.js

```javascript
import React, { useState } from "react";
import axios from "axios";
import "./report.css";

const Report = () => {
  const [mrn, setMrn] = useState("");
  const [searchedMrn, setSearchedMrn] = useState("");
  const [pdfUrl, setPdfUrl] = useState("");
  const [loading, setLoading] = useState(false);
  const [notFound, setNotFound] = useState(false);

  const handleSearch = async () => {
    const formattedMrn = mrn.trim().toUpperCase();
    if (!formattedMrn) return;

    setLoading(true);
    setNotFound(false);
    setPdfUrl(""); // reset before new search

    try {
      const response = await axios.get(
        `http://localhost:5000/reports?mrn=${formattedMrn}`
      );
      if (response.data.length > 0) {
        // Get the relative path from the db.json file
        const relativePath = response.data[0].pdfUrl;

        // Create the full path by prepending the public path
        // The public folder is served at the root of your app
        const fullPath = relativePath.startsWith("/")
          ? relativePath
          : `/${relativePath}`;

        setPdfUrl(fullPath);
        setSearchedMrn(formattedMrn);
      } else {
        setNotFound(true);
      }
    } catch (error) {
      console.error("Error fetching report:", error);
      setNotFound(true);
    } finally {
      setLoading(false);
    }
  };
```

```
  const handleRemove = () => {
    setSearchedMrn("");
    setMrn("");
    setPdfUrl("");
    setNotFound(false);
  };

  const handleMrnChange = (e) => {
    setMrn(e.target.value.toUpperCase());
  };

  return (
    <div className="report-body standalone-report">
      <div className="report-card">
        <div className="content-area">
          <h2>Client Reports</h2>
          <p className="page-description">
            This page allows you to view and download client reports by entering
            the MRN (Medical Record Number). Please ensure the MRN is accurate
            to retrieve the correct PDF report.
          </p>
          <div className="search-container">
            <input
              type="text"
              placeholder="Enter MRN"
              value={mrn}
              onChange={handleMrnChange}
              className="uppercase-input"
            />
            <button onClick={handleSearch}>Search</button>
          </div>

          {loading && <p>Loading report...</p>}

          {searchedMrn && (
            <div className="pdf-container">
              <h3>Report for MRN: {searchedMrn}</h3>
              {pdfUrl ? (
                <>
                  <iframe
                    src={pdfUrl}
                    width="100%"
                    height="600px"
                    title="Patient Report"
                    style={{ border: "1px solid #ccc", borderRadius: "6px" }}
                  />
                  <button onClick={handleRemove} className="remove-button">
                    Remove
                  </button>
                </>
              ) : (
                <p>PDF not found for this MRN.</p>
              )}
            </div>
          )}

          {notFound && !loading && (
            <p className="error-text">No report found for MRN: {mrn}</p>
          )}
        </div>
      </div>
    </div>
  );
};

export default Report;
```

File: src\dashboard\physio\userDetails.css
```
/* Main layout and containers */
.content-container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
```

```css
    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;
}
```

```css
.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;
}
```

```css
.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;
```

```css
  }
}

/* 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

```javascript
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({
```

```jsx
          ...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}>
```

```jsx
                    <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>
    </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>
```

```jsx
          <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

```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

```jsx
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
```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%;
}
```

```css
.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);
```

```css
    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

```javascript
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);
```

```css
  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;
```

```css
}
.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;
```

```css
}
.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;
```

```css
      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;
```

```css
}

@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;
```

```css
}

.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;
```

```css
    justify-content: center;
}

.toggle-icon {
  transition: var(--transition);
}
```

File: src\dashboard\physio\components\sidebar.js

```javascript
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> */}
```

```jsx
          <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="/clients" className={navLinkClass}>
          <Users
            size={18}
            style={{{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}}
          />
          <span>Client Management</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/PhysiosAppointments" className={navLinkClass}>
          <Users
            size={18}
            style={{{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}}
          />
          <span>Appointments</span>
        </NavLink>
      </li>
      <li>
```

```jsx
        <NavLink to="/assign" className={navLinkClass}>
          <Dumbbell
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Assign Exercise</span>
        </NavLink>
      </li>
      <li>
        <NavLink to="/reports" className={navLinkClass}>
          <BarChart2
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Reports</span>
        </NavLink>
      </li>

      <li>
        <NavLink to="/physio" className={navLinkClass}>
          <Workflow
            size={18}
            style={{
              marginRight: !collapsed || isHovering ? "10px" : "0",
            }}
          />
          <span>Physio</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">
        <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;
```