* React Hooks
* State management
* Routing
* Protected routes
* Conditional rendering
* API handling
* Form validation
* Component reusability
* UI libraries (like Tailwind or Material UI)
* Notifications, loading states, and real-time updates

**🎯 Goals for the React Frontend**

Build a clean, component-driven, scalable React app that:

* Works for both **Manager** and **Employee** (role-based interface)
* Integrates with all the microservices through APIs
* Implements best practices for modern React development

**📁 Suggested Project Structure**

/src

│

├── assets/ # Static files like images, icons

├── components/ # Reusable UI components (Buttons, Modals, Navbar)

├── layouts/ # Role-based layouts (ManagerLayout, EmployeeLayout)

├── pages/ # All pages (Dashboard, Leaves, Attendance, etc.)

│ ├── manager/

│ └── employee/

├── services/ # API service handlers (authService, taskService, etc.)

├── hooks/ # Custom hooks (useAuth, useFetch, etc.)

├── context/ # Context API for global state (auth, theme, etc.)

├── utils/ # JWT helpers, date formatters, etc.

├── routes/ # React Router config (private + public routes)

├── App.jsx

└── main.jsx

**🔧 Key Topics and Where to Use Them**

**✅ 1. React Router + Role-based Routing**

* Use react-router-dom v6+
* Implement PrivateRoute and RoleRoute components

📍 Used In:

* Routing access based on login and role
* Example:

<Route path="/dashboard" element={<PrivateRoute />}>

<Route path="manager" element={<ManagerDashboard />} />

<Route path="employee" element={<EmployeeDashboard />} />

</Route>

**✅ 2. JWT Authentication + Protected Routes**

* Store JWT in localStorage or httpOnly cookie
* On login → decode role → redirect accordingly
* Attach token in all API calls using Axios interceptors

📍 Used In:

* Login page
* Global API setup
* Route protection

**✅ 3. Hooks**

* useState, useEffect for logic
* Custom hooks like:
  + useAuth() – Auth token/role
  + useFetch() – For API calls

📍 Used In:

* Every interactive page (task, feedback, etc.)

**✅ 4. Global State Management**

Options:

* Start with **Context API** for auth/user data
* Optionally move to **Redux Toolkit** later

📍 Used In:

* User login state
* Theme toggle or global messages

**✅ 5. API Integration (with Axios)**

* Create a central Axios instance with interceptors
* All services in services/ folder

const axiosInstance = axios.create({

baseURL: import.meta.env.VITE\_API\_URL,

});

axiosInstance.interceptors.request.use((config) => {

config.headers.Authorization = `Bearer ${localStorage.getItem("token")}`;

return config;

});

📍 Used In:

* Task CRUD, Leave Management, Feedback, etc.

**✅ 6. Conditional Rendering**

* Render manager/employee components conditionally

{user.role === "MANAGER" ? <ManagerLayout /> : <EmployeeLayout />}

📍 Used In:

* Navbar, Sidebar, Dashboards

**✅ 7. Form Handling + Validation**

Use:

* react-hook-form + yup for schema validation

📍 Used In:

* Leave request form
* Login form
* Task assignment form

**✅ 8. Loading States + Toast Notifications**

Use:

* react-toastify for alerts (success, error, etc.)
* react-spinners or custom loaders for UX

📍 Used In:

* After API calls
* During form submission

**✅ 9. Reusable Components**

Examples:

* <FormInput />, <Loader />, <Modal />, <Avatar />, <TaskCard />

📍 Used In:

* Everywhere 😄

**✅ 10. Responsive UI with Tailwind CSS**

* Fast styling
* Role-specific dashboards
* Sidebar + header navigation
* Use @headlessui/react or Material UI for modals, toggles

📍 Used In:

* Layouts
* Cards, Dashboards, Forms

**🛠️ Tools / Libraries to Use**

| **Purpose** | **Tool/Lib** |
| --- | --- |
| Routing | react-router-dom |
| Forms & Validation | react-hook-form, yup |
| API | axios |
| Styling | tailwindcss, clsx |
| Alerts | react-toastify |
| Icons | react-icons, lucide-react |
| Loading UI | react-loader-spinner |

**📈 Optional Advanced Topics (Phase 2)**

| **Topic** | **Where to Try It** |
| --- | --- |
| WebSocket Chat | Chat between manager/employee |
| Lazy Loading | Route/code splitting |
| Debouncing | Search filters (Employee, Task) |
| Optimistic UI | Leave request, task update |
| PWA Support | Add offline caching |