High-Level Architecture - Telecom Mission Control SaaS

This document provides a high-level overview of the frontend and backend architecture of the Telecom SaaS platform, designed to be scalable, a11y-friendly, and mobile responsive.

# 1. Frontend Architecture

Built using React, Redux, CSS Modules, and a11y best practices, the frontend is modular, responsive, and accessible.  
  
Core layers include:

• App Shell: Persistent Navbar, Layout, and Routes  
• ProtectedRoute: Guards role-based access per route  
• Screens: Dashboard, Policies, Towers, Devices, Users  
• Redux Store: Stores auth, device, policy, user, tower states  
• Actions/Reducers: Handle async API communication using redux-thunk  
• Axios Client: For RESTful communication with backend  
• Service Worker (PWA): Enables offline access  
• Manifest: Makes app installable on mobile devices  
• CSS Modules: Used with `:root` theming for dark/light mode and accessibility

# 2. Backend Architecture

Node.js with Express powers the RESTful API server. MongoDB Atlas is used as the primary data store.  
  
Key components:

• Express Routes: `/api/auth`, `/api/users`, `/api/devices`, `/api/policies`, `/api/towers`  
• JWT Auth: Stateless token-based authentication  
• Middleware: Auth middleware for route protection, error handlers, CORS  
• MongoDB Models: Mongoose schemas for User, Policy, Device, Tower  
• Security: bcrypt password hashing, helmet for headers, input validation

# 3. Logical Layered Design

User → UI (React) → Redux Action → Axios → Express API → Controller → Mongoose → MongoDB  
 ↑ Redux Reducer ← API Response

# 4. Deployment Stack

• Client: React app   
• API Server: Node.js  
• DB: MongoDB Atlas (cloud-hosted)  
• PWA: Service worker registered on the client

# 5. Scalability & A11y Highlights

• Role-based route access and dynamic menus  
• Lazy-loaded routes and components  
• Responsive grid/flex UI layouts  
• Keyboard navigation and screen reader roles  
• Error boundaries and global error handling