Frontend Development Plan: In-House Employee Management System (EMS)

1. Project Overview

The In-House Employee Management System (EMS) is designed to manage employees, track hardware and software assets, handle licenses, provide a ticketing system, and integrate with Microsoft Graph API for user authentication and management. The frontend will be a responsive web application built with React JS and Tailwind CSS app structure.

2. Technology Stack

- **Platform:** Web (Responsive Desktop-First)
- **Framework:** React JS
- **Styling:** Tailwind CSS
- **Design System:** Reusable components (forms, tables, modals, tabs)
- **Typography:** Modern sans-serif (e.g., Inter, Roboto)
- **Icons:** Lucide
- **Authentication:** JWT-based authentication for all routes.

3. API Integration

The frontend will interact with the backend APIs via the following base URLs and functionalities:

- **Authentication:** `/api/auth` - Login, logout, token management.
- **Employee Management:** `/api/employees` - CRUD operations and bulk import.
- **Hardware Assets:** `/api/hardware` - Manage company hardware.
- **Software Assets:** `/api/software` - Manage software and versioning.
- **Licenses:** `/api/licenses` - Manage license keys and assignments.
- **Tickets:** `/api/tickets` - IT support ticket lifecycle.
- **Microsoft Graph:** `/api/msgraph` - Microsoft 365 integration.
- **Audit Logs:** `/api/audit-logs` - Access event logs.

4. UI Components and Pages

The following sections detail the UI components and pages, referencing the UI Design Deliverable Document.

4.1. Login Page

- **Elements:** Email and password fields, submit button, inline error messages, "Forgot password?" link, loading spinner. **(changed, review the Figma Design)**
- **Functionality:** Authenticate via MS OAuth using the `/api/auth/login` endpoint.

4.2. Dashboard (Role-Based)

- **Elements:** Top Navigation (Logo, User Profile, Notifications), Collapsible Side Navigation (Dashboard, Employees, Assets, Software, Licenses, Tickets, Audit Logs, Settings).
- **Widget Cards:**
    - Total Employees (Active/Inactive)
    - Total Assets (Available/Assigned)
    - Expiring Licenses (next 30 days)
    - Ticket Summary (Open/Critical)
- **Functionality:** Each widget must be clickable and link to the respective module.

4.3. Employee Management

4.3.1. List View

- **Elements:** Table with Name, Email, Department, Role, Status.
- **Filters:** Role, Status, Department (utilizing common query filters where applicable).
- **Actions:** Edit, deactivate (using `/api/employees/:id` PUT/DELETE).
- **Buttons:** "Add New Employee" (POST `/api/employees`), "Bulk Upload (CSV)" (POST `/api/employees/bulk`).

4.3.2. Detail View

- **Layout:** Left Panel (Employee details), Right Panel Tabs (Assets, Licenses, Tickets).
- **Functionality:**
    - Assets: Show assigned assets (GET `/api/employees/:id/assets`).
    - Licenses: Show active licenses (GET `/api/licenses/employee/:id`).
    - Tickets: Show raised & assigned tickets (GET `/api/tickets` with appropriate filters).

4.4. Hardware Asset Management

4.4.1. List View

- **Elements:** Table with Asset Type, Model, Status, Assigned To.
- **Filters:** Type, Status.
- **Button:** "Add New Asset" (POST `/api/hardware`).

4.4.2. Detail View

- **Elements:** Asset Details (Model, Serial, Specs), Assignment History (timeline component).
- **Buttons:** Assign (POST `/api/hardware/:id/assign`), Unassign.

4.5. Software Management

4.5.1. List View

- **Elements:** Name, Version, Vendor, License Required (Yes/No), Install Count.
- **Button:** "Add New Software" (POST `/api/software`).

4.5.2. Detail View

- **Elements:** Info Block (Name, Vendor), Version history, Linked Licenses list.

4.6. License Management

4.6.1. List View

- **Elements:** Software Name, Key (masked), Status, Assigned To, Expiry.
- **Filters:** Status, Software.
- **Button:** "Add New License" (POST `/api/licenses`).

4.6.2. Detail View

- **Elements:** Assignment dropdown (Employee or Hardware), License metadata.
- **Action Buttons:** Revoke, activate (PUT/DELETE `/api/licenses/:id`).

4.7. Ticketing System

4.7.1. User View

- **Elements:** "My Tickets" table with Title, Status, Priority, Last Updated.
- **Form:** "Raise Ticket" (Title, Description, Priority dropdown) (POST `/api/tickets`).

4.7.2. IT/Admin View

- **Filters:** Status, Assigned To, Priority.
- **Button:** "Assign to Me".
- **Elements:** Ticket detail with comment thread and status updates (PUT `/api/tickets/:id`).

4.8. Audit Logs (Admin/IT Only)

- **Elements:** Table with Action, User, Target, Timestamp.
- **Filters:** Action type, Date range.
- **Functionality:** Read-only (GET `/api/audit-logs`).

4.9. Settings (Admin Only)

- **Tabs:** Roles & Permissions, SMTP Configuration, Microsoft OAuth Credentials, Branding (Logo, App name).

5. Responsive Behaviour

- The layout must scale gracefully for smaller desktop screens (minimum 1024px width).
- Side navigation should collapse into a hamburger menu for smaller screens.
- Tables should implement horizontal scrolling for better display on smaller screens.
- Mobile-friendly view is not required in Phase 1.

6. Security and Authentication

- Implement JWT-based authentication for all routes, sending the token in the `Authorization: Bearer <token>` header.
- Implement role-based access checks on the route level using middleware to control access to specific functionalities (e.g., Admin, IT, Authenticated).

- Ensure all sensitive actions are audit-logged by the backend.
- Admin access is required for Microsoft Graph API operations.

7. Planned Enhancements (Post Phase 1)

- WebSocket support for real-time ticket status updates.
- Advanced reports module (asset usage, license trends).
- Notification centre with role-based alerts.