Product Requirements Document (PRD)

Feature: Job Scheduling & Employee Allocation System

Prepared for: Development Team

Prepared by: Project Owner

Date: November 2025

1. Objective

Integrate a calendar-based scheduling and employee allocation module into the existing application. The feature must allow authorized users to create, edit, and manage job schedules through a visual calendar, assign existing employees to jobs, and maintain permission control via the app’s permission dashboard. The system must be highly professional, error-free, user-friendly, and fully tested before deployment.

2. Access Control & Permissions

This feature must respect the app’s existing role-based permission system. Only users who are explicitly granted access under the Permissions Dashboard can view or use the scheduling feature. Default access: Admins/Managers have full access; Employees can only view their assigned jobs. A new permission flag named 'can_manage_schedule' must be added and integrated into the permission dashboard. Permission checks must be implemented both on frontend routes and backend endpoints. Unauthorized users should see a clean 'Access Restricted' message.

3. Feature Requirements

A. Calendar Interface

A 'Schedule' tab will be added to the main navigation. The calendar should use FullCalendar.js or React Big Calendar. It must support month, week, and day views with features like create, edit, delete, and drag-and-drop job scheduling. Jobs are color-coded: Blue for Upcoming, Green for In Progress, Gray for Completed. Clicking a job opens a modal with full details and assigned employees.

B. Employee Allocation

A multi-select dropdown lists active employees for job assignment. Inactive/unavailable employees are automatically excluded. Supports multiple assignments and blocks double-bookings. Assigned employees appear on both job detail modals and calendar tooltips.

C. User Experience

Clean, minimalist UI consistent with the current application’s styling (Tailwind/Shadcn aesthetic). Responsive design for desktop and tablet, fast load times, consistent icons, and dynamic updates without refresh.

D. Logic & Automation

Completed jobs auto-hide from default view (with toggle). Upcoming jobs appear on the dashboard. Statuses update dynamically. Optional phase: notifications for assigned employees.

4. Technical Implementation (High-Level)

Integrate seamlessly into existing frontend and backend structures. Follow current authentication and API patterns. Enforce permission checks both in frontend route guards and backend middleware. Endpoints: GET /schedule, POST /schedule, PUT /schedule/:id, DELETE /schedule/:id, POST /schedule/:id/assign. All routes must return clean JSON responses with proper error handling.

5. Testing & Quality Assurance

Systematic testing is required before deployment. Acceptance criteria: feature visible only to authorized users, zero errors, proper data persistence, flawless double-booking prevention, and full UI integration. Automated tests must achieve ≥ 80% coverage. Tools: Jest, Mocha, Cypress, Playwright.

6. Deployment & Workflow

Development will occur on branch 'feature/schedule-module' in incremental phases (Calendar → Job CRUD → Employee Assignment → Permission Integration → Testing). All stages must pass QA validation before merging. Deployment to staging for 24-hour stability validation is required prior to production release.

7. Completion Definition

Calendar scheduling system live and functional. Access controlled via permission dashboard. Fully responsive and bug-free upon delivery.