

Project: Engineering Resource Management System

Overview

Build a full-stack application to manage engineering team assignments across projects. Track who's working on what, their capacity allocation, and when they'll be available for new projects.

Duration: 2 days (16 hours)

Core Features

1. Authentication & User Roles

- Login system with two roles: **Manager** and **Engineer**
- Engineers can view their assignments
- Managers can assign people to projects

2. Engineer Management

- **Engineer Profile:** Name, skills (React, Node.js, Python, etc.), seniority level
- **Employment Type:** Full-time (100% capacity) or Part-time (50% capacity)
- **Current Status:** Available percentage (e.g., 60% allocated, 40% available)

3. Project Management

- **Basic Project Info:** Name, description, start/end dates, required team size
- **Required Skills:** What technologies/skills the project needs
- **Project Status:** Active, Planning, Completed

4. Assignment System

- **Assign Engineers to Projects:** Select engineer, project, allocation percentage
- **View Current Assignments:** Who's working on which project and for how long
- **Capacity Tracking:** Visual indicator of each engineer's current workload

5. Dashboard Views

- **Manager Dashboard:** Team overview, who's overloaded/underutilized
- **Engineer Dashboard:** My current projects and upcoming assignments
- **Availability Planning:** When will engineers be free for new projects

6. Search & Analytics

- **Search & Filter:** Find engineers by skills or projects by status
 - **Analytics:** Simple charts showing team utilization
-

AI-Powered Development Approach





Important: We strongly encourage leveraging AI development tools throughout this assignment. As the engineering landscape evolves rapidly, we seek team members who can effectively use AI to accelerate development while maintaining code quality and deep technical understanding.

Expected AI Tool Usage

- **AI IDEs:** Cursor, Windsurf, or similar AI-powered editors
- **AI Assistants:** Claude, ChatGPT, GitHub Copilot for code generation and debugging
- **Code Review:** Use AI for code optimization and best practices suggestions
- **Problem Solving:** Leverage AI for architectural decisions and implementation strategies

Critical Requirement

While we encourage AI usage, **you must understand every line of code you implement.** Simply copying AI-generated code without comprehension will not meet our standards. You should:

-  Review and understand all AI-generated code
-  Modify and optimize AI suggestions to fit your specific needs
-  Be able to explain your implementation decisions
-  Test and validate all AI-assisted implementations

Documentation Requirement

In your README, include a section on:

- Which AI tools you used and how
 - Specific examples of how AI accelerated your development
 - Any challenges you faced with AI-generated code and how you resolved them
 - Your approach to validating and understanding AI suggestions
-

Technical Requirements

Frontend (React + TypeScript)

- **Components:** Use ShadCN UI components with Tailwind CSS
- **Forms:** React Hook Form for project/assignment creation
- **Data Display:** Tables showing assignments, charts showing capacity
- **State Management:** React Context or Zustand

Backend (Node.js or NestJS preferred)

- **Database:** MongoDB or Any Database with proper schemas
 - **Authentication:** JWT tokens
 - **API Design:** RESTful endpoints
 - **Business Logic:** Capacity calculations
-

Database Schemas

User

```
{
  email: String,
  name: String,
  role: 'engineer' | 'manager',
  // Engineer fields
  skills: ['React', 'Node.js', 'Python'], // Array of strings
  seniority: 'junior' | 'mid' | 'senior',
  maxCapacity: Number, // 100 for full-time, 50 for part-time
  department: String
}
```

Project

```
{
  name: String,
  description: String,
  startDate: Date,
  endDate: Date,
  requiredSkills: [String],
  teamSize: Number,
  status: 'planning' | 'active' | 'completed',
  managerId: ObjectId
}
```

Assignment

```
{
  engineerId: ObjectId,
  projectId: ObjectId,
  allocationPercentage: Number, // 0-100
  startDate: Date,
  endDate: Date,
  role: String // 'Developer', 'Tech Lead', etc.
}
```

Key Calculations

1. Available Capacity

```
// How much capacity an engineer has left
function getAvailableCapacity(engineerId) {
  const engineer = getEngineer(engineerId);
  const activeAssignments = getActiveAssignments(engineerId);
  const totalAllocated = activeAssignments.reduce((sum, a) => sum + a.allocationPercentage,
0);
  return engineer.maxCapacity - totalAllocated;
}
```

2. Skill Matching

```
// Find engineers with required skills for a project
function findSuitableEngineers(project) {
  return engineers.filter(engineer =>
    project.requiredSkills.some(skill => engineer.skills.includes(skill))
  );
}
```

API Endpoints

Authentication:

POST /api/auth/login

GET /api/auth/profile

Engineers:

GET /api/engineers
GET /api/engineers/:id/capacity

Projects:
GET /api/projects
POST /api/projects
GET /api/projects/:id

Assignments:
GET /api/assignments
POST /api/assignments
PUT /api/assignments/:id
DELETE /api/assignments/:id

UI Requirements

Manager Pages

- **Team Overview:** List of engineers with current capacity (e.g., "John: 80% allocated")
- **Create Assignment:** Form to assign engineer to project with percentage
- **Project Management:** Create/edit projects with required skills

Engineer Pages

- **My Assignments:** Current and upcoming projects
- **Profile:** Update skills and basic info

Key UI Elements

- **Capacity Bars:** Visual representation of workload (progress bars)
 - **Skill Tags:** Display engineer skills and project requirements
 - **Assignment Timeline:** Simple calendar view of assignments
-

Evaluation Criteria

1. **Functionality (20%):** Core features work correctly
2. **UI/UX (20%):** Intuitive interface with good visual design
3. **Code Quality (30%):** Clean, readable, well-structured code

4. **Technical Implementation (30%):** Proper API design, database structure, code structure
-

Sample Data

Include seed data with:

- 3-4 Engineers with different skills and capacity
 - 3-4 Projects with various requirements
 - 6-8 Assignments showing different scenarios
 - Mix of full-time and part-time engineers
-

Deliverables

1. **GitHub Repository** with:

- Frontend and backend code
- README with setup instructions
- Database seed script

2. **Working Demo** with:


- All core features functional
 - Sample data pre-loaded
 - Responsive design
-

Bonus Features

- **Timeline View:** Calendar showing assignment duration
 - **Skill Gap Analysis:** Identify missing skills in team
-

Success Metrics

Your system should:

-  Accurately calculate and display engineer capacity

- ☒ Allow managers to easily assign people to projects
 - ☒ Show clear overview of team workload
 - ☒ Handle basic CRUD operations smoothly
 - ☒ Have intuitive UI that non-technical managers can use
-

Focus on building a practical tool that solves real resource allocation problems. Keep it simple but functional!