**Documentation Employee Management System** **Problem Statement**

***Title: Development of an Employee Management Portal***

**Overview:**

The Employee Management System (EMS) is a web-based portal designed to streamline the management of employee information within the organization. The system aims to efficiently track employees assigned to various projects, identify employees currently not assigned to any project, and determine the respective managers for each employee. This system will significantly enhance the organization’s ability to manage human resources effectively and improve overall operational efficiency.

**Current Challenges:**

1. Manual Tracking: The current method of tracking employee assignments and projects is manual and prone to errors, leading to inefficiencies and mismanagement of resources.

2. Lack of Centralized Information: Information regarding employees, their skills, project assignments, and managers is scattered, making it difficult to access and manage.

3. Resource Allocation: There is no streamlined process for managers to request and allocate resources based on skills and availability.

4. Administrative Overheads: Admin tasks such as adding new employees, updating employee details, assigning projects, and managing requests are time-consuming and lack automation.

5. Limited Access Control: The current system does not provide differentiated access controls based on user roles, leading to potential security and data integrity issues.

**Objectives:**

1. Develop a user-friendly online portal to centralize employee, project, and manager information.

2. Implement role-based access control to ensure that Admins, Managers, and Employees have appropriate access to system functionalities.

3. Enable Admins to manage employees, projects, and resource allocation efficiently.

4. Provide Managers with tools to filter employees by skills, request unassigned employees, and manage project assignments.

5. Allow Employees to view their project assignments, update their skills, and access relevant information.

6. Ensure high code quality and maintainability through comprehensive unit testing and application logging.

**Functional Requirements:**

**User Roles and Permissions:**

**1. Admin:**

- Comprehensive access to all system functionalities.

- Add new employees and managers.

- Add new projects to the system.

- View all employees and projects.

- Assign/unassign projects to employees.

- Approve/reject manager requests for resources.

- Delete employees.

- Update employee details.

**2. Employee:**

- Limited access to system functionalities.

- Update personal skills.

- View personal information including project assignments and manager details.

- View all employees and managers after login.

**3. Manager:**

- Subtype of Admin with specific capabilities.

- View all employees, managers, and projects.

- Filter employees by skills and unassigned status.

- Request unassigned employees based on skills for projects.

- Request multiple employees for a project.

**Technical Specifications:**

- User Interface: Developed using React framework with HTML, CSS, and JavaScript.

- Backend: REST API developed in Python. I have used FastAPI

- Database: Used MySQL .

- Testing: Achieve a minimum of 85% code coverage through unit tests.

- Logging: Implement application logging for monitoring and debugging purposes.

**Delivered:**

1. A fully functional Employee Management Portal with the specified features.

2. Documentation detailing the system architecture, user guide, and maintenance procedures.

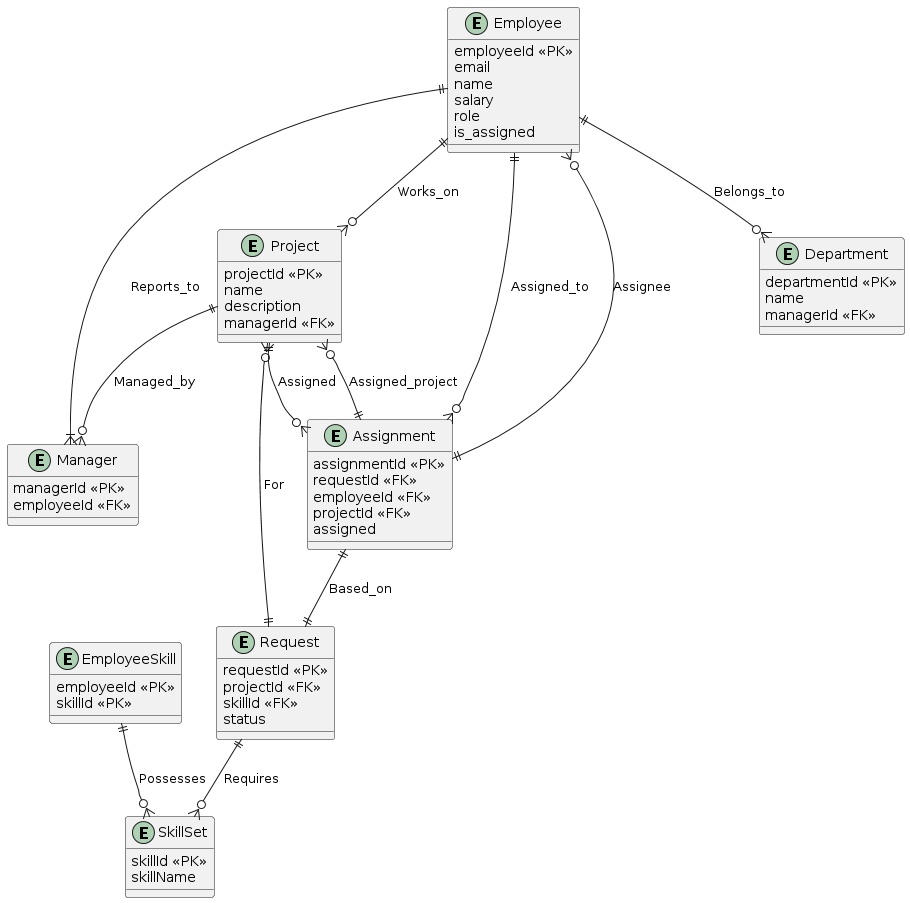
3. Unit tests ensuring at least 85% code coverage.

4. Application logs for system monitoring and maintenance.

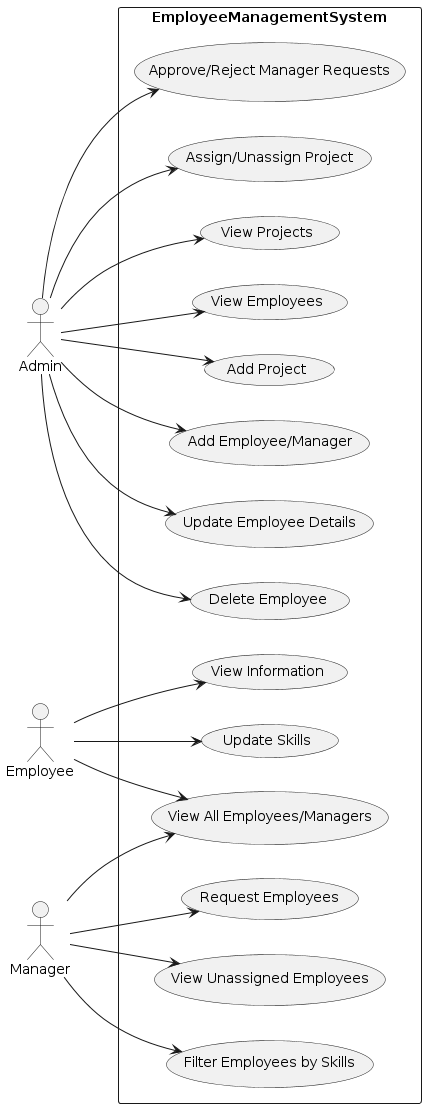
By addressing the above challenges and meeting the specified objectives, the Employee Management Portal will significantly enhance the efficiency and effectiveness of managing human resources within the organization.

**Architecture Diagrams**

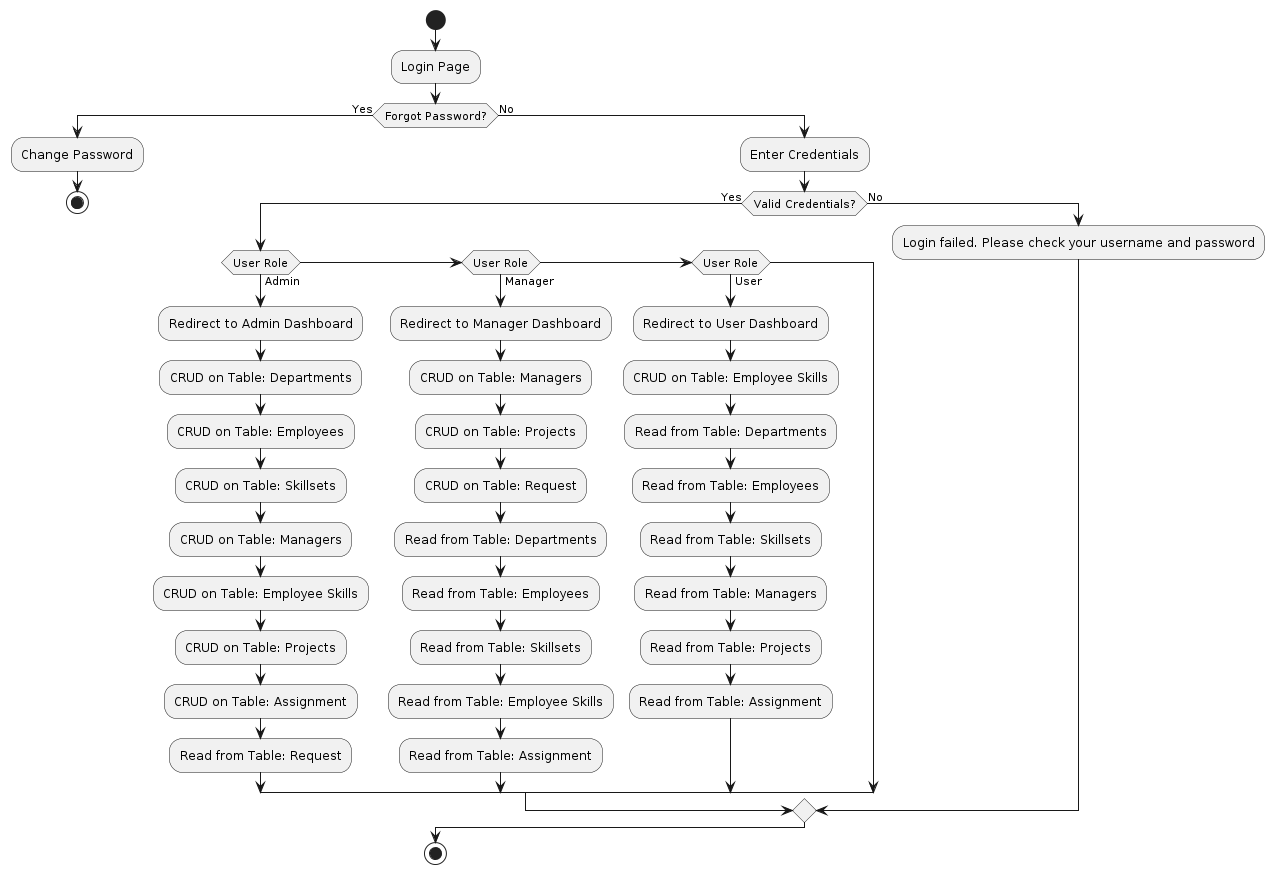
1. **ER Diagram**

****

**2. Use Case Diagram:**

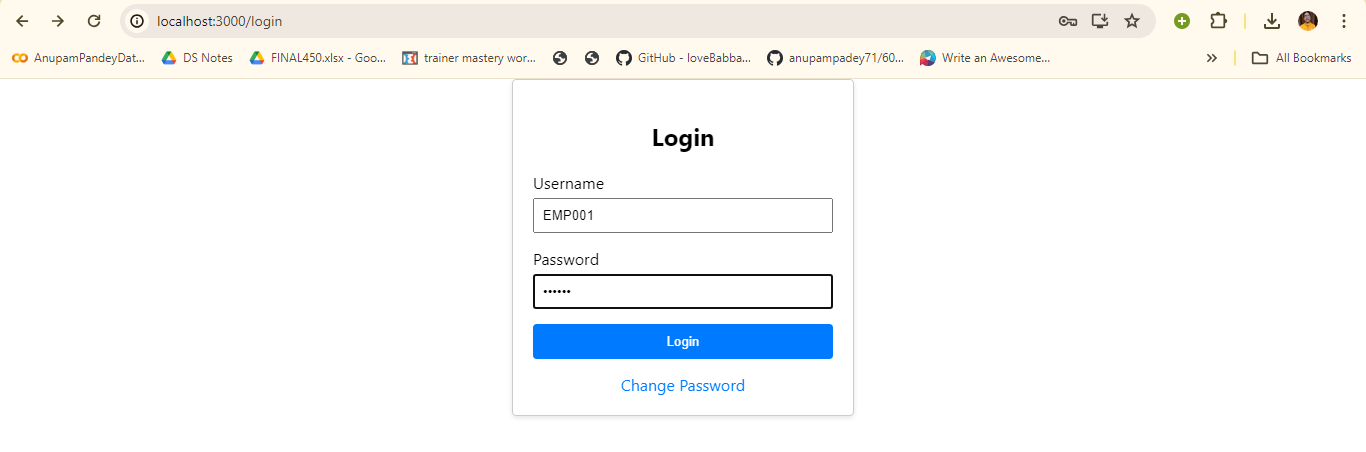
****

**3. Flowchart**

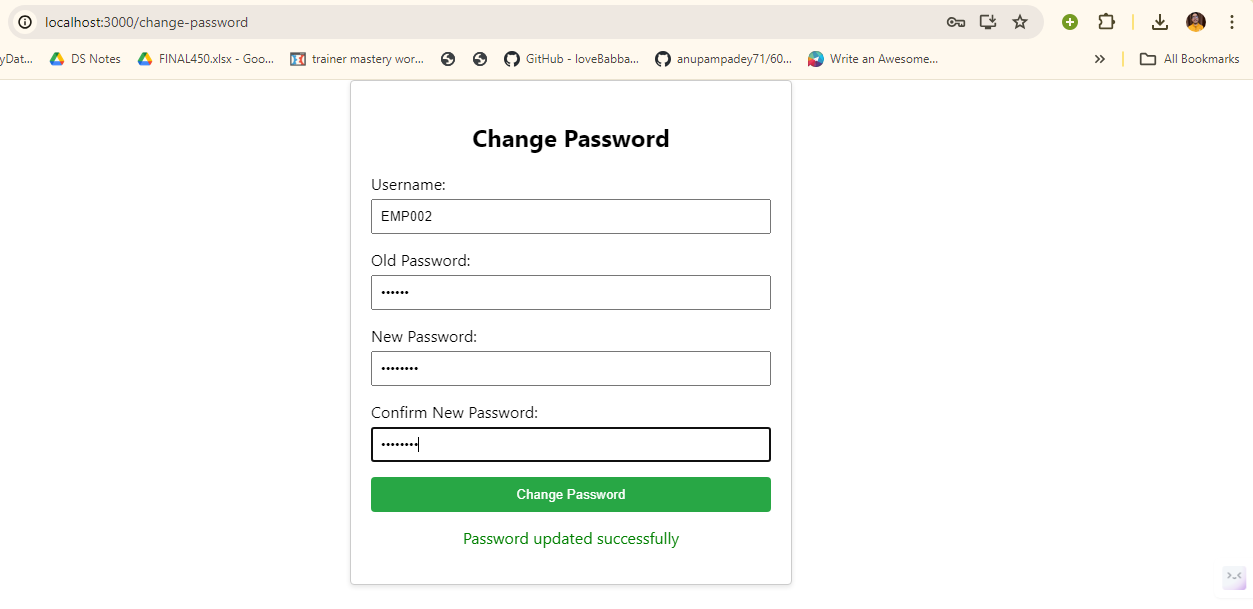
****

**Screenshots of frontend**

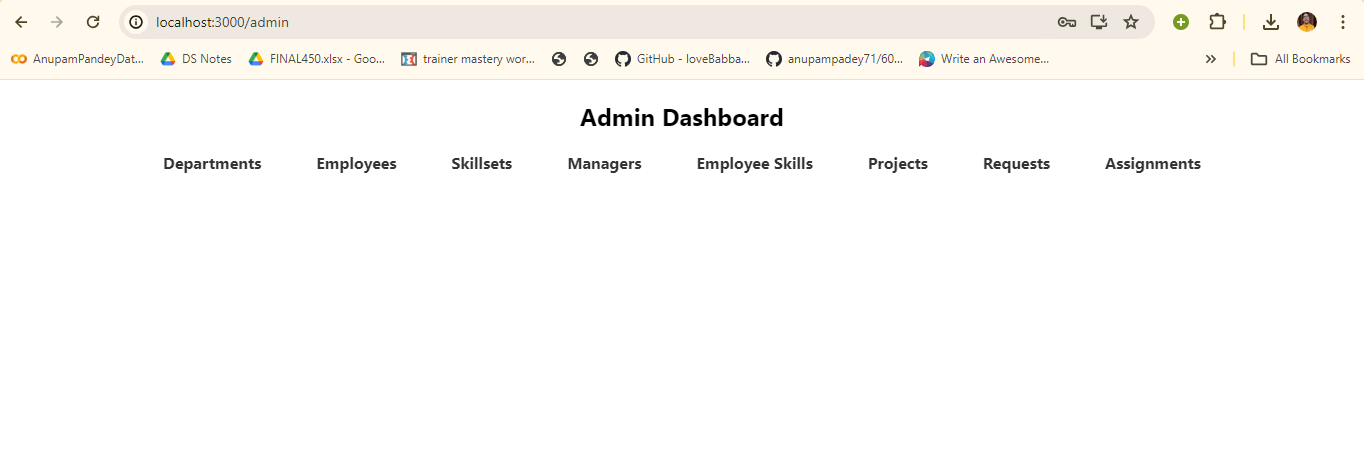
1. **Login**

****

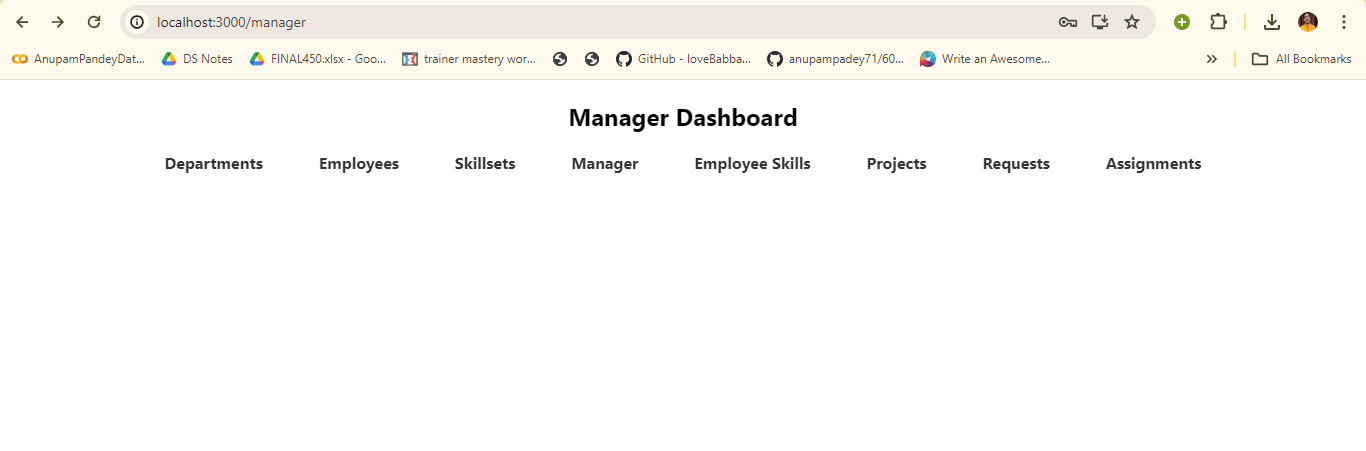
1. **Change Password**

****

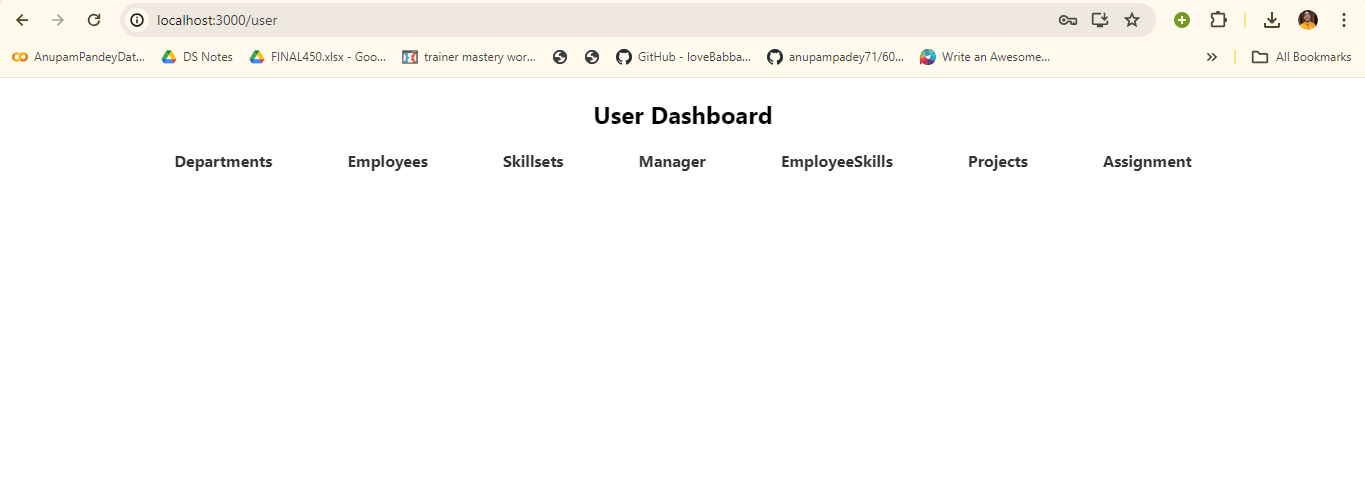
1. **Admin Dashboard**

****

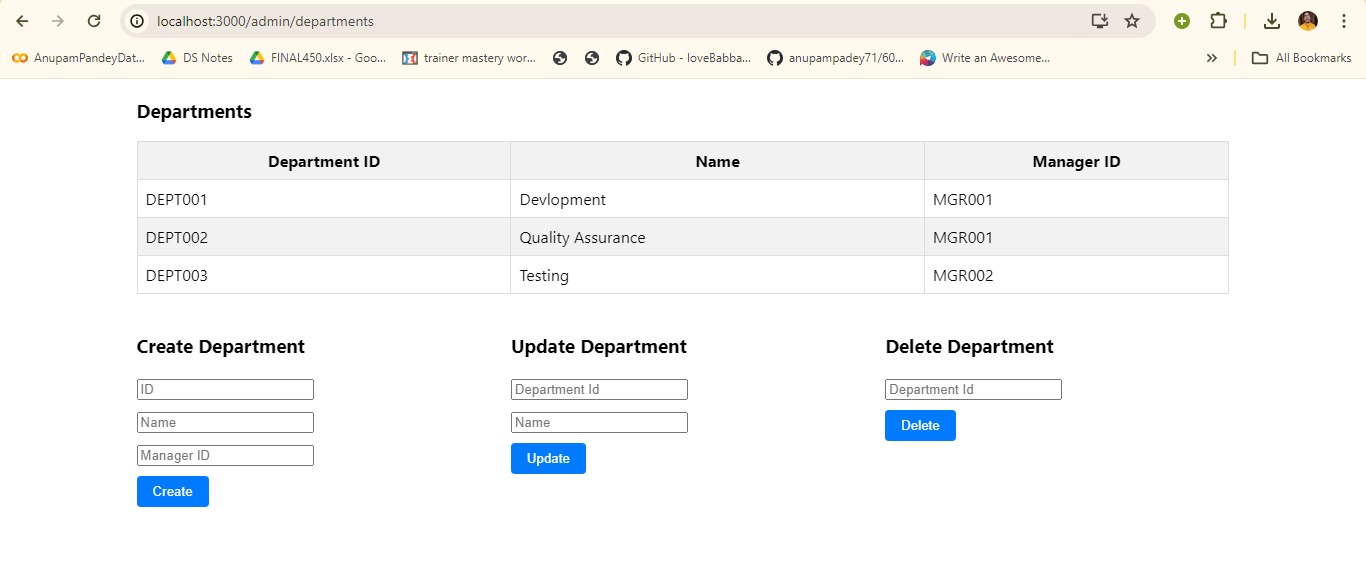
1. **Manager Dashboard**

****

1. **User Dashboard**

****

**7. Admin Department**

****

Above is sample of one table that is Department table in admin dashboard crud operation, similarly i have done for all tables in all 3 roles dashboard

**Challenges I faced**

1. Create Schema Architecture:
2. To add role based authentication like to give what crud operation permission to which user for which table. So I created users table to store user its password in encrypted from and role and based on it write the authentication code
3. To integrate FastAPI with Frontend for that i have coded apiService.js to integrate my frontend with my backend
4. To create flow for the frontend
5. For styling my frontend and I have added css in my code to make it in proper format