System Requirements Specification Index

For

WorkPlace Admin System

Version 1.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

WORKPLACE ADMIN SYSTEM

System Requirements Specification

1 PROJECT ABSTRACT

The **WorkPlace Admin System** is a ASP.NET Web API 4.8 with MS SQL Server database connectivity. It enables users to manage various aspects of employee and Department information to manage and Tracking. It provides a user-friendly interface for administrators to perform CRUD (Create, Read, Update, Delete) operations on departments and employees. The system allows seamless navigation and interaction with organizational data through a well-structured UI. The system ensures secure authentication, seamless data management, and robust backend integration to maintain organizational records effectively..

Before access any data from API user must verified by token.

Following is the requirement specifications:

	Package Tracker Application
Modules	
1	Employee
2	Department
3	EmployeeLogin
Employee Module Functionalities	
1	Get the Employee by Id
2	Create an Employee Information Details
3	Update the existing Employee details by its ID
4	Get all Employee Details
5	Delete an employee
6	Retrieve all employees in a specific department
Department Module Functionalities	
1	Get all Department Details
2	Get the Department by Id

3	Create an Department Information Details
4	Update the existing Department data/details
5	Edit an existing Department by ID
6	Delete Department
Auth Module Functionalities	
1	User Need to access login method from AuthController and using EmailId and Password get token and use that token to work on any other controller.

2 Assumptions, Dependencies, Risks / Constraints

.

2.1 Employee Constraints

- When fetching an Employee by ID, if the Employee ID does not exist, the operation should throw a custom exception.
- When updating an Employee, if the Employee ID does not exist, the operation should throw a custom exception.
- When removing an Employee, if the Employee ID does not exist, the operation should throw a custom exception.
- If any required field is missing, salary is not a valid number, or department ID doesn't exist.

2.2 Department Constraints

- When fetching a Department by ID, if the Department ID does not exist, the operation should throw a custom exception.
- When updating a Department, if the Department ID does not exist, the operation should throw a custom exception.
- When removing a Department, if the Department ID does not exist, the operation should throw a custom exception.
- Retrieve all employees in a specific department with department ID must exists in database otherwise the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

Employee Table

- Id (Int) Key, Not Null
- Name (String),Not Null
- Email (String),Not Null
- Position (String), Not Null
- Salary (Int) positive number not null.
- DepartmentId (int), Not Null

.

Department Table –

- Id (Int) Key, Not Null
- Name (String),Not Null
- Description (String)

EmployeeLogin Table –

- Id (Int) Key, Not Null
- Email (Email) Not Null, must in email format
- Password (string), Not Null

4 Rest endpoints

.

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 EmployeesController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/employees/Id
<table>
<tr><td>Http Method</td><td>GET</td></tr>
<tr><td>Parameter</td><td>1</td></tr>
<tr><td>Return</td><td>Employee</td></tr>
</table></td><td>Fetches an employee</td></tr>
<tr><td>2. /api/employees/employees
<table>
<tr><td>Http Method</td><td>POST</td></tr>
<tr><td>Parameter 1</td><td>Employee</td></tr>
<tr><td>Return</td><td>Employee</td></tr>
</table></td><td>Add a new Employee</td></tr>
<tr><td>3. /api/employees/DeleteEmployee</td><td></td></tr>
</table>

Http Method	DELETE	Delete Employee with given Employee id
Parameter 1	Int (id)	
Return	-	

4./ api/employees/GetAllEmployee		Fetches the all Employee
Http Method	GET	
Parameter 1	-	
Return	<IEnumerable<Employee>>	

5. /api/ employees/GetEmployeesByDepartment/{departmentId}		Get an existing Employee by Department Id
Http Method	GET	
Parameter 1	Int (id)	
Parameter 2	Department ID	
Return	Employee	

6. /api/employees/UpdateEmployee/{empId}		Update an existing Employee by Employee Id and Employee Model
Http Method	PUT	
Parameter 1	Int (id)	
Parameter 2	Employee ID	
Return	Status = "Success"	

4.2 DepartmentsCONTROLLER

URL Exposed		**Purpose**
1. /api/departments/GetAllDepartments		Fetches all the Department
Http Method	GET	
Parameter	-	
Return	IEnumerable<Department>	
2. /api/departments/departments		Add a new Department
Http Method	POST	
Parameter 1	Department	
Return	department	
3. /api/departments/DeleteDepartment		Delete Department with given Department id
Http Method	DELETE	

Parameter 1	Int (id)	
Return	-	
4. /api/departments/{id}		Fetches the Department with the given id
Http Method	GET	
Parameter 1	Int (id)	
Return	Department	
5. /api/departments /UpdateDepartment/{deptId}		Updates existing Department
Http Method	PUT	
Parameter 1	Int (id)	
Parameter 2	Department	
Return	Department	

4.3 AuthCONTROLLER

URL Exposed		Purpose
1. /api/auth/login		Get New Token for access all controller method, Basic token that need to pass with emailId and password on postman after token received (Picture attached in below)
Http Method	POST	
Parameter 1	EmployeeLogin	
Return	string token	

5. TEMPLATE CODE STRUCTURE

5.1 Package: EmployeeeManagement

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	Employee Controller Department, Controller, Auth Controller	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented

Web.Config	Web.Config file	Contain all Services settings and SQL server Configuration.	Already Implemented

Interface	IEmployeeService, IDepartmentsServices, IEmployeeLoginService interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	EmployeeService, DepartmentsServices, EmployeeLoginService CS file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	IDepartmentsRepository, IEmployeesRepository CS file and interface.	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially Implemented
Models	Employee, Department, EmployeeLogi, Response cs file	All Entities/Domain attribute are used for pass the data in controller.	Already Implementation

5.2 Package: EmployeeManagement.Tests

Resources

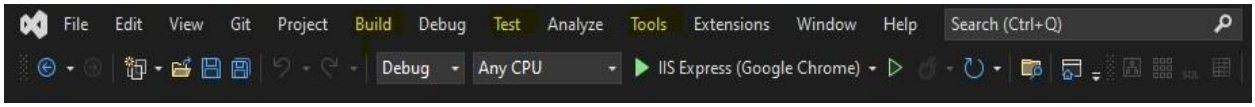
The EmployeeManagement.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

6. Execution Steps to Follow

1. **After successfully cloning the project template on desktop, you will be able to see folder named with your user id. (e.g. user@gmail.com)**

2. **Go to below path and open solution file with Visual Studio.**
   Path: user@gmail.com > EmployeeManagement_App.App> EmployeeManagement_API.sln

3. **All actions such as building, compiling, running the application, and executing test cases will be performed using the Visual Studio interface. Rather than using the command terminal, the necessary operations will be initiated through the buttons, menus, and features available within the Visual Studio IDE.**



4. **Press Ctrl + S to save your code.**

5. **Steps to Apply Migration(Code first approach):**
   - **Go to "Tools" -> "NuGet Package Manager" -> "Package Manager Console" from the top menu bar of Visual Studio.**

   - **After clicking on "Package Manager Console," a new tab should open at the bottom of the Visual Studio window, displaying the Package Manager Console.**

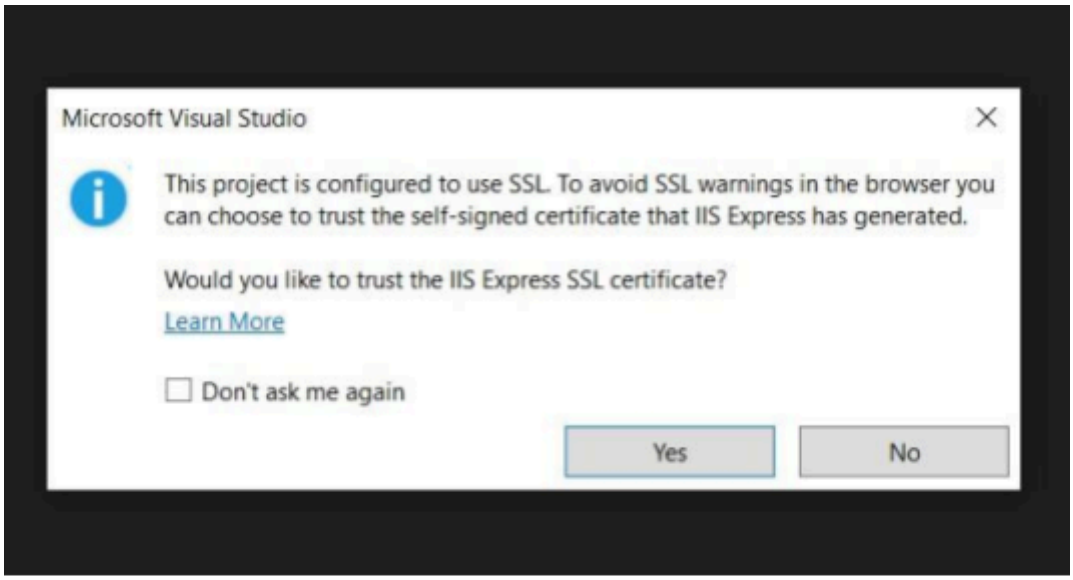
   - **Run following command to apply migration : update-database**

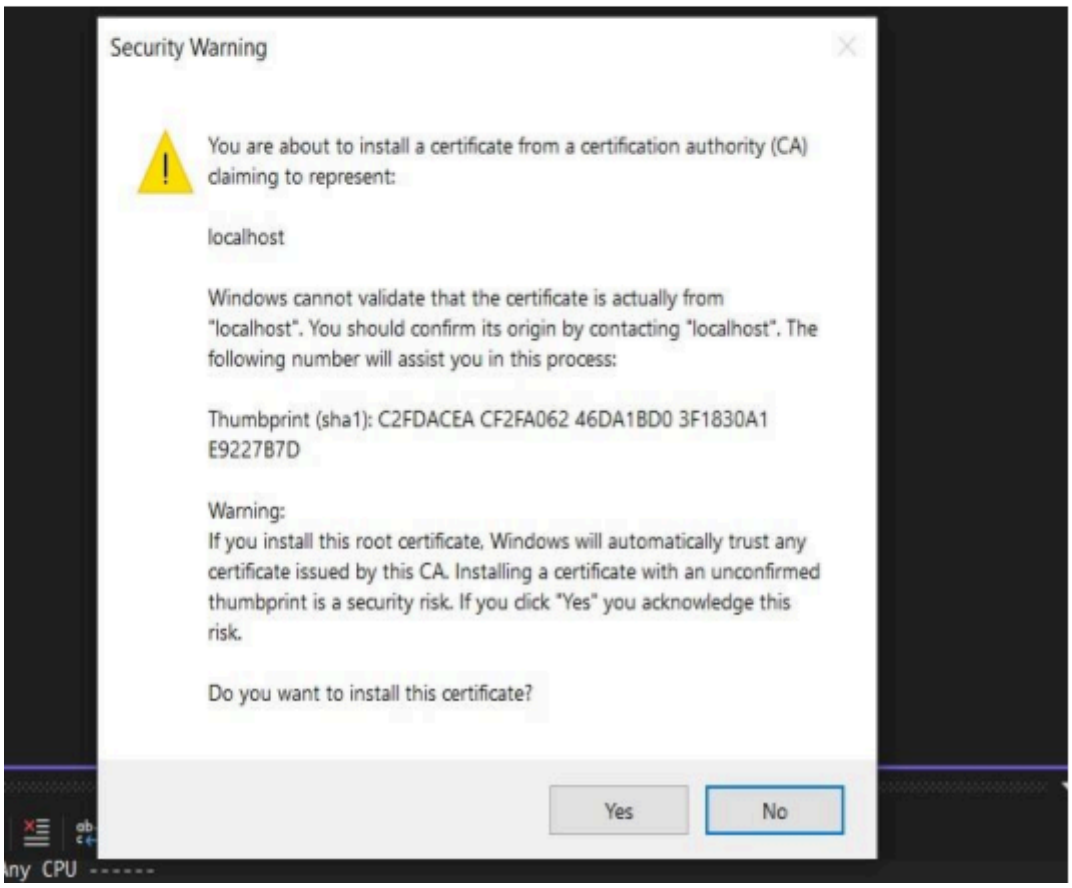
6. **Steps after applying migration :**
   - **Click on "Build"(Top Menu Bar) > "Clean Solution"**
   - **Click on "Build"(Top Menu Bar) > "Rebuild Solution"**
   - **Click on "Build" (Top Menu Bar) > "Build Solution"**

7. **To build your project in Visual Studio, click on "Build" in the top menu, then select "Build Solution" or press Ctrl + Shift + B.**

8. **To launch your application, press F5 or use Ctrl + F5 to start your application without debugging.**

(Note: If you get below screens regarding SSL certificate, Click on YES for both screens.)

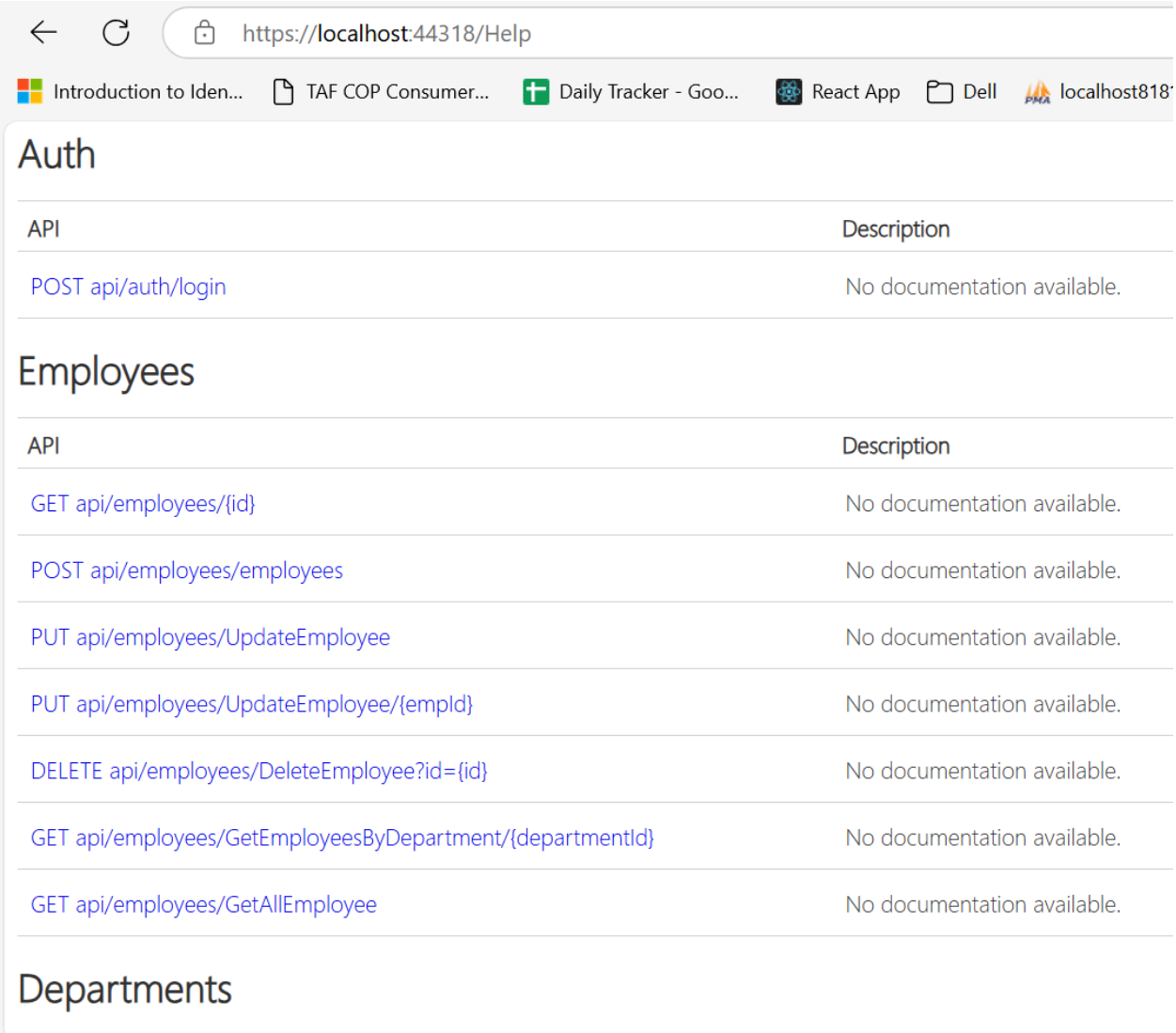




Note: The application will run in the local browser, Run Application again.

9. **To test any applications on a browser, use the internal browser in the workspace.**

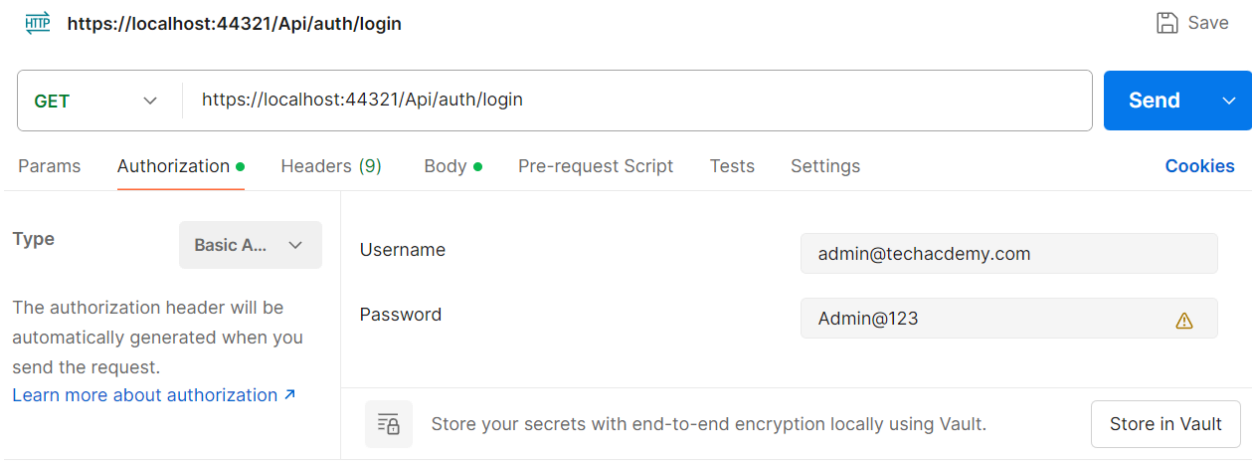
(Note: This screen shows that your application is running on local browser, and the base URL is https://localhost:44318) Make sure your output URL before access API



10. To test any Restful application, you can use POSTMAN.

(Note: Before sending the request from postman you need to disable "Enable SSL certificate verification" from settings.)

Step to get Basic Token – Put username and password as below screen, then you can use API like Employee and Department Controller



https://localhost:44318/Help

Introduction to Iden... | TAF COP Consumer... | Daily Tracker - Goo... | React App | Dell | localhost818

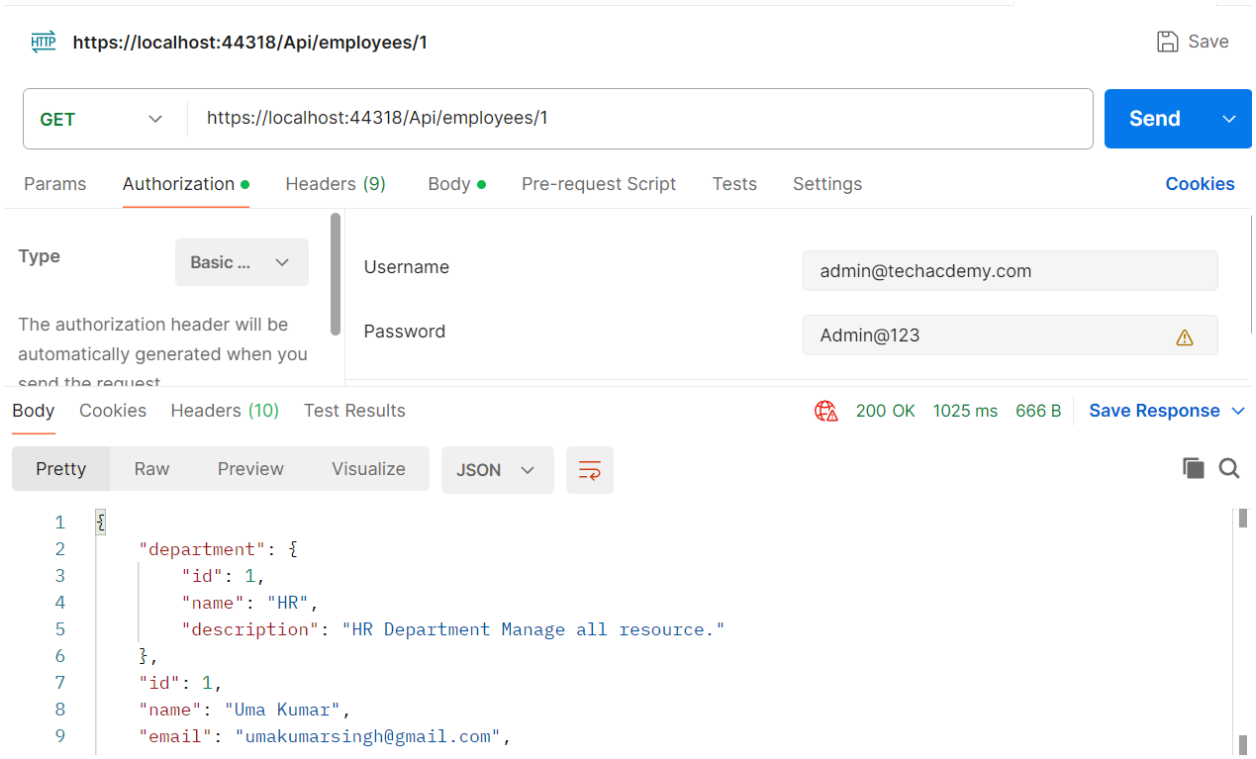
Auth

API	Description
POST api/auth/login	No documentation available.

Employees

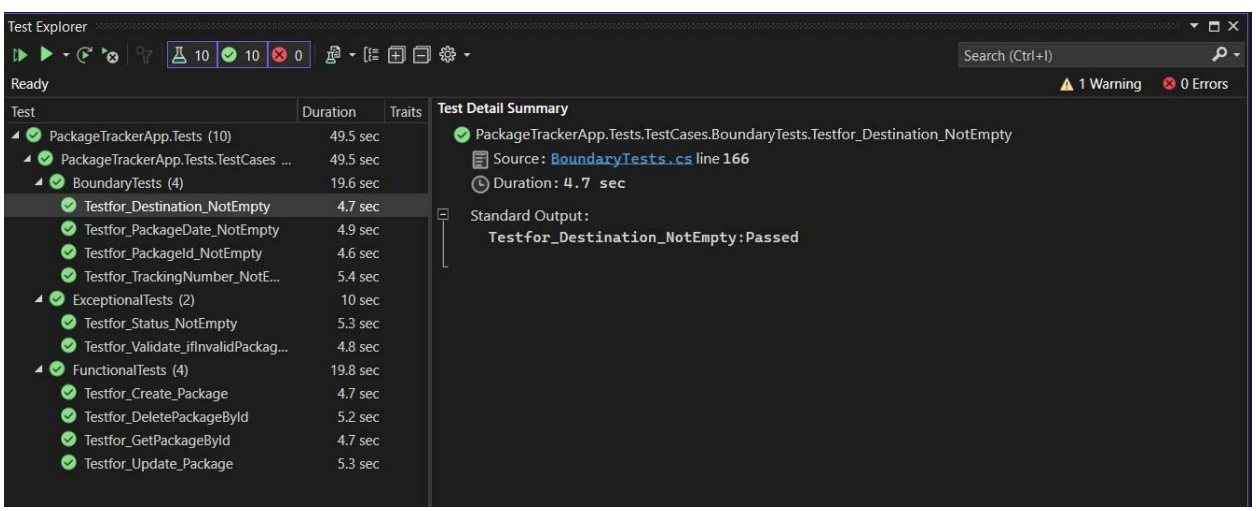
API	Description
GET api/employees/{id}	No documentation available.
POST api/employees/employees	No documentation available.
PUT api/employees/UpdateEmployee	No documentation available.
PUT api/employees/UpdateEmployee/{empId}	No documentation available.
DELETE api/employees/DeleteEmployee?id={id}	No documentation available.
GET api/employees/GetEmployeesByDepartment/{departmentId}	No documentation available.
GET api/employees/GetAllEmployee	No documentation available.

Departments



Use same base URL link from local browser to test on postman.

11. **To run test cases in your project in Visual Studio, click on "Test" -> "Run All Tests" in the top menu. (You can run this command multiple times to identify the test case status, and refactor code to make maximum test cases passed before final submission).**



12. Steps to push changes to GitHub:

- **Go to "View" -> "Git Changes" from the top menu bar of Visual Studio.**

- **In the "Changes" window on the right side of Visual Studio, you'll see the modified files.**

- **Enter any commit message in the "Message" box at the top of the window, and click on “Commit All” button.**

- **After committing your changes, Click the "Push" button (Up Arrow Button) to push your committed changes to the GitHub repository.**

13. If you want to exit (logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to follow step-12 compulsorily. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

14. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

15. You need to follow step-12 compulsorily, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

FRONTEND - ANGULAR SPA

1. PROBLEM STATEMENT

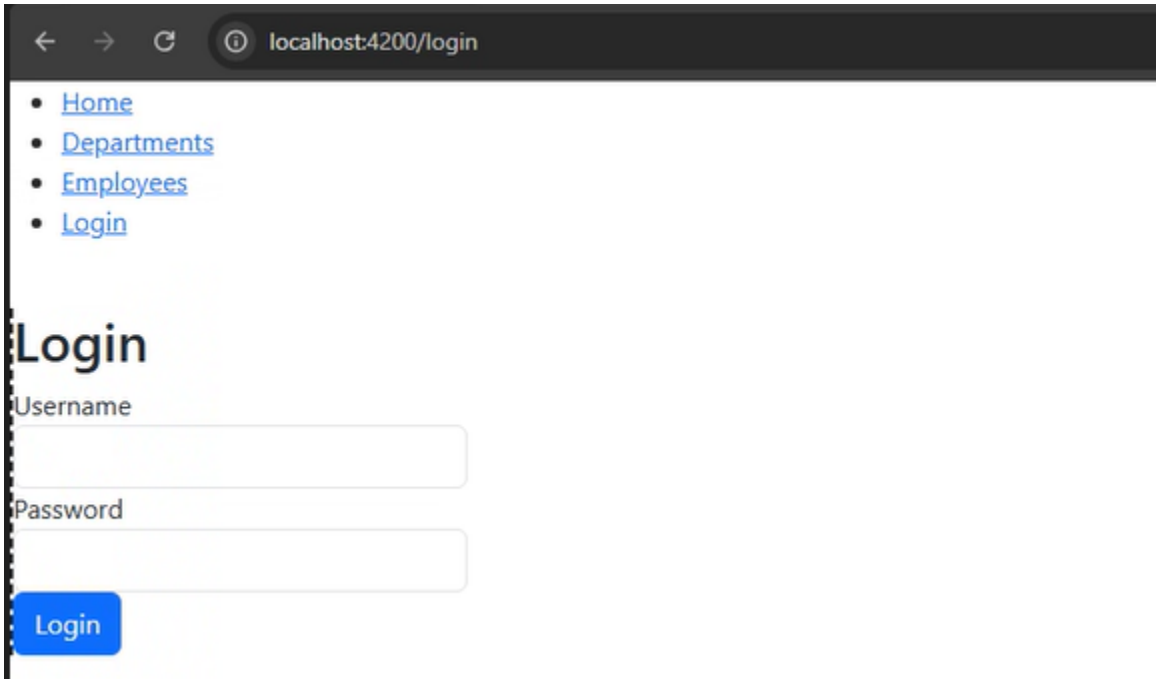
The **WorkPlace Admin System** is a **Single Page Application (SPA)** designed to manage departments and employees efficiently. It provides a user-friendly interface for administrators to perform CRUD (Create, Read, Update, Delete) operations on departments and employees. The system allows seamless navigation and interaction with organizational data through a well-structured UI. The system ensures secure authentication, seamless data management, and robust backend integration to maintain organizational records effectively.

2. PROPOSED WORKPLACE ADMIN SYSTEM APPLICATION WIREFRAME

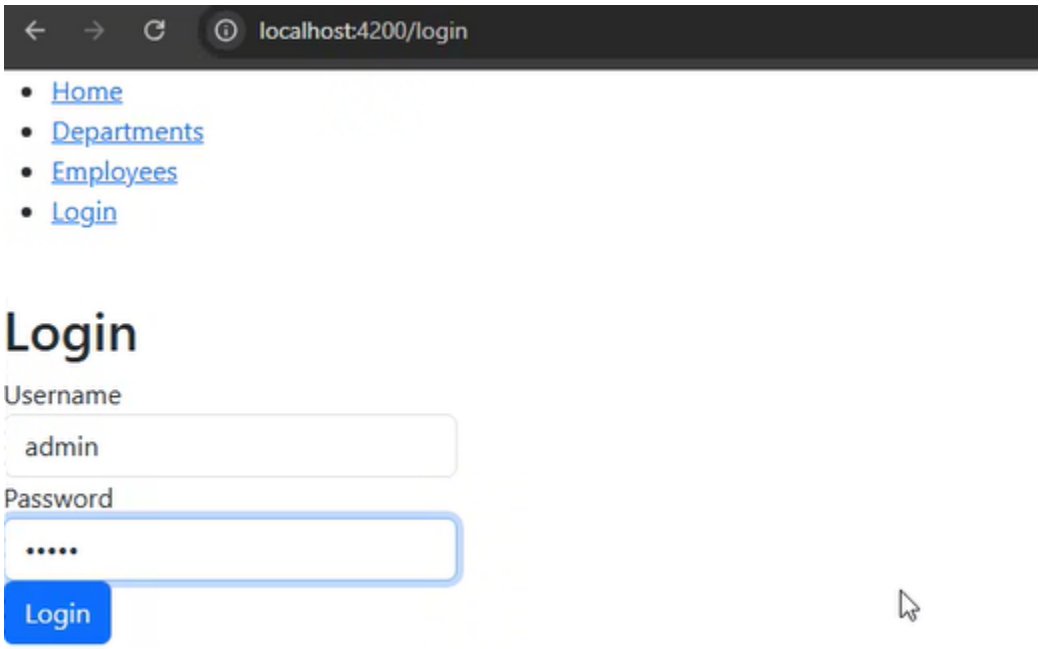
UI needs improvisation and modification as per given use case and to make test cases passed.

- SCREENSHOTS

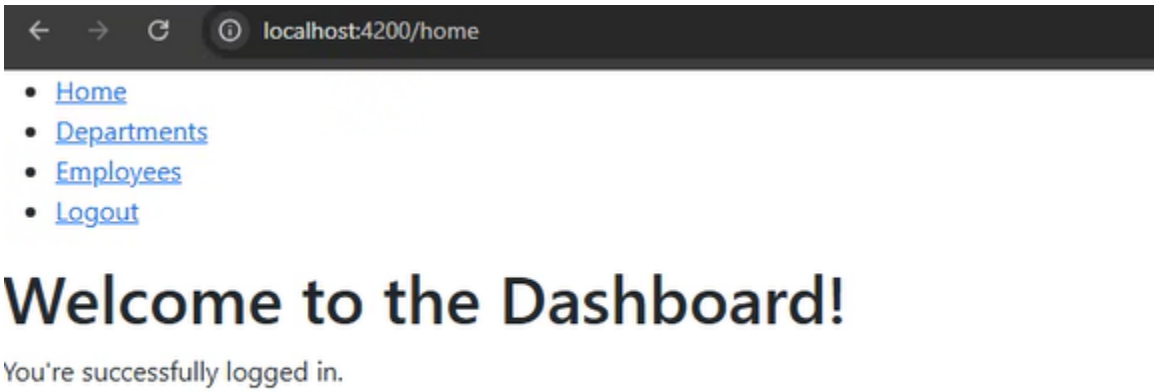
***** Login Page*****



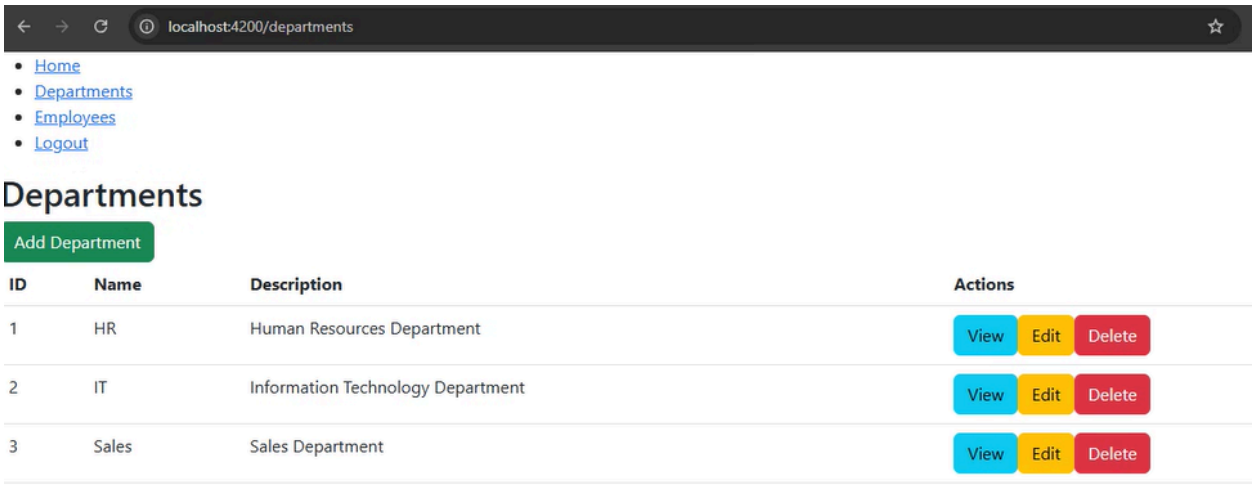
*** Users can log in using the already existing admin credentials (Username: `admin`, Password: `admin`) to access and manage the system.**



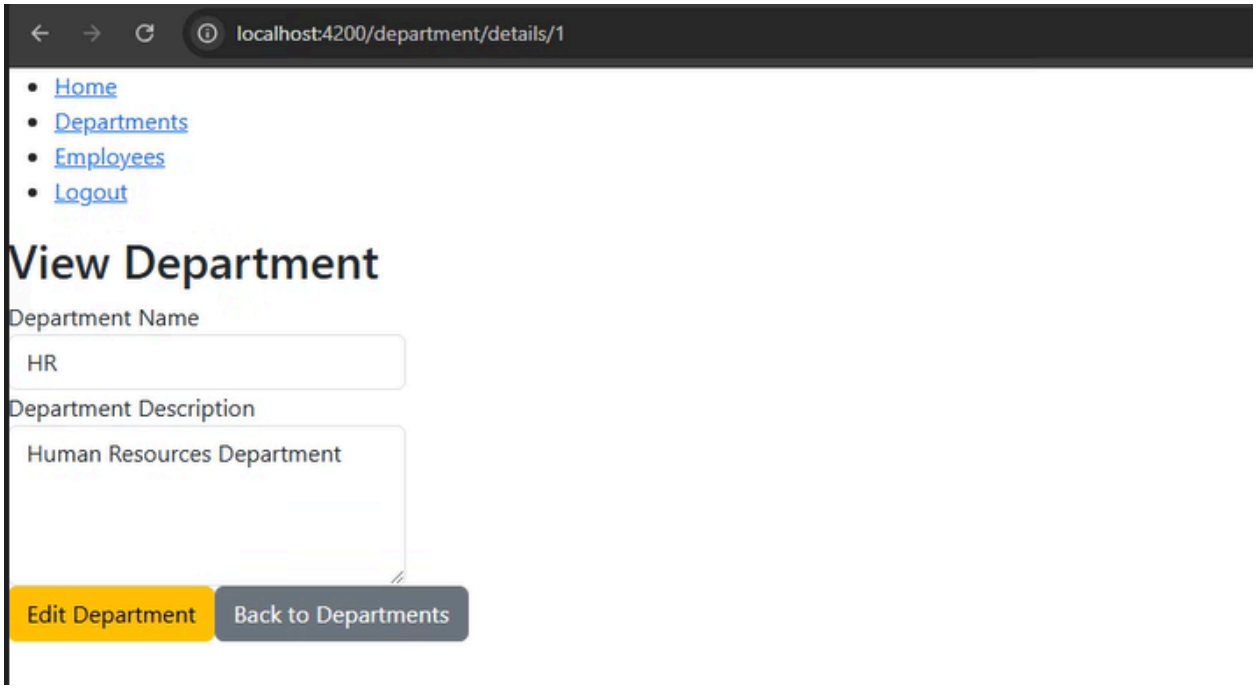
***** Home Page*****



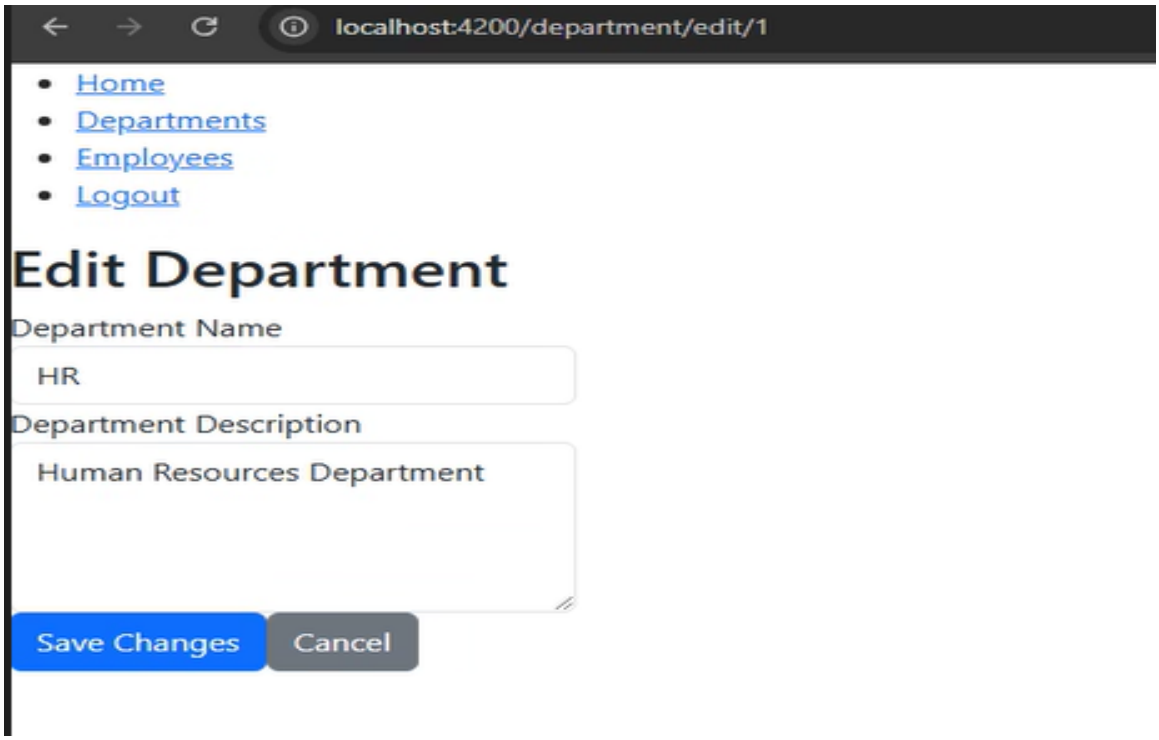
***** Departments Page*****



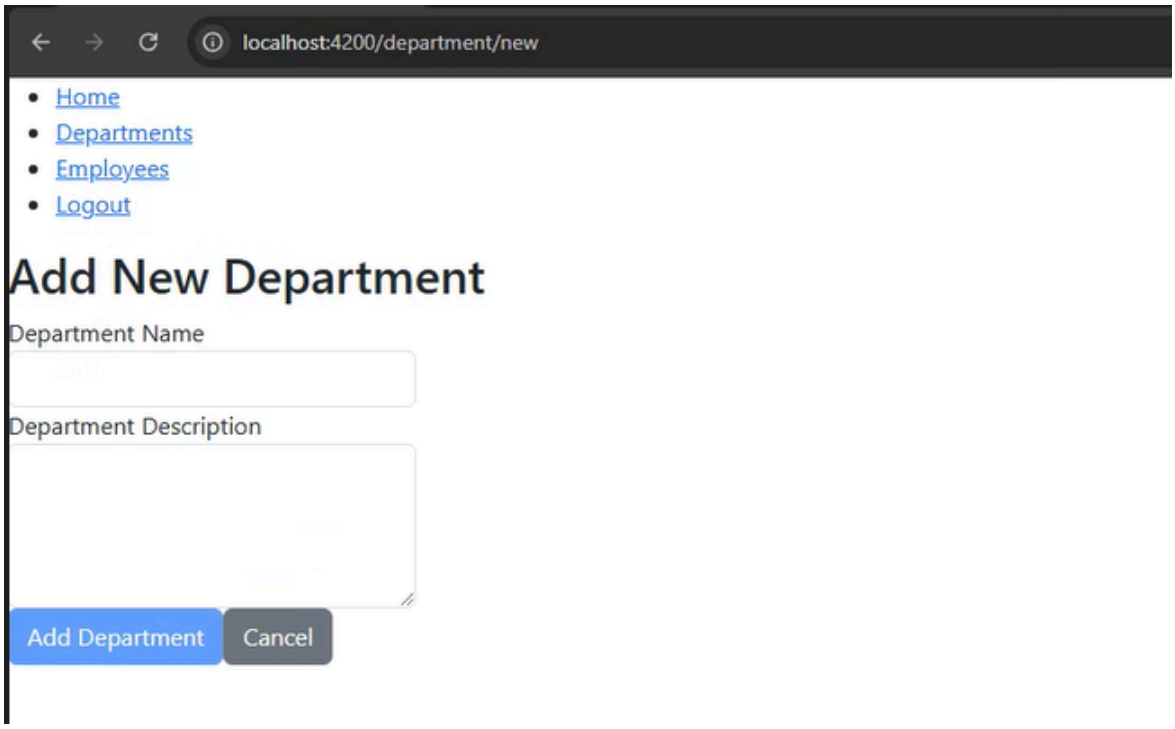
***** View Departments *****



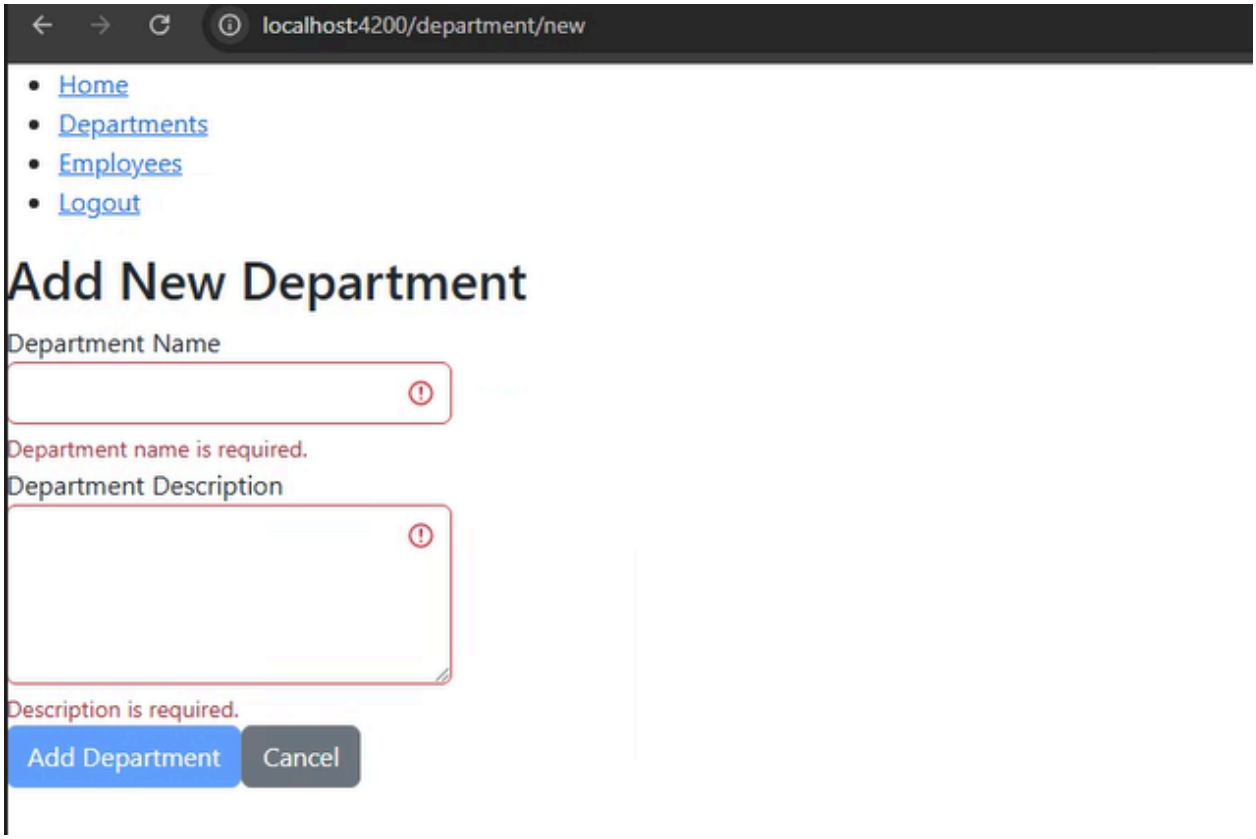
***** Edit Departments *****



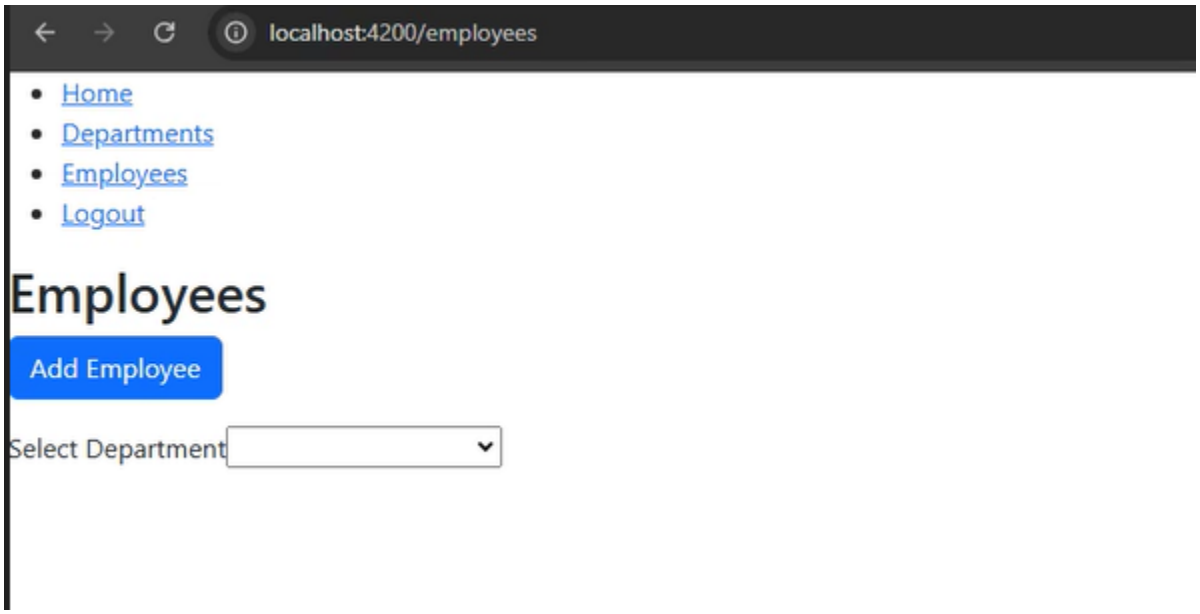
***** Add New Department *****



*** If any field (Department Name and Department Description) is not filled, the respective validation error should be displayed, and the "Add Department" button should remain disabled; it should only be enabled once all fields are correctly filled.**

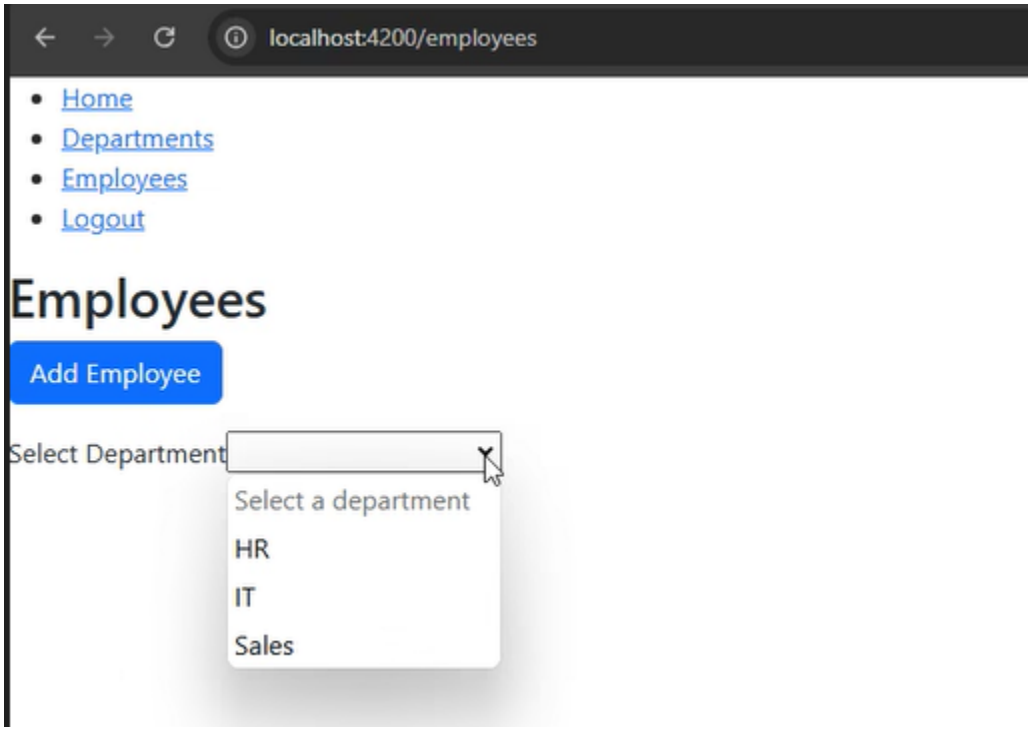


***** Employees Page *****

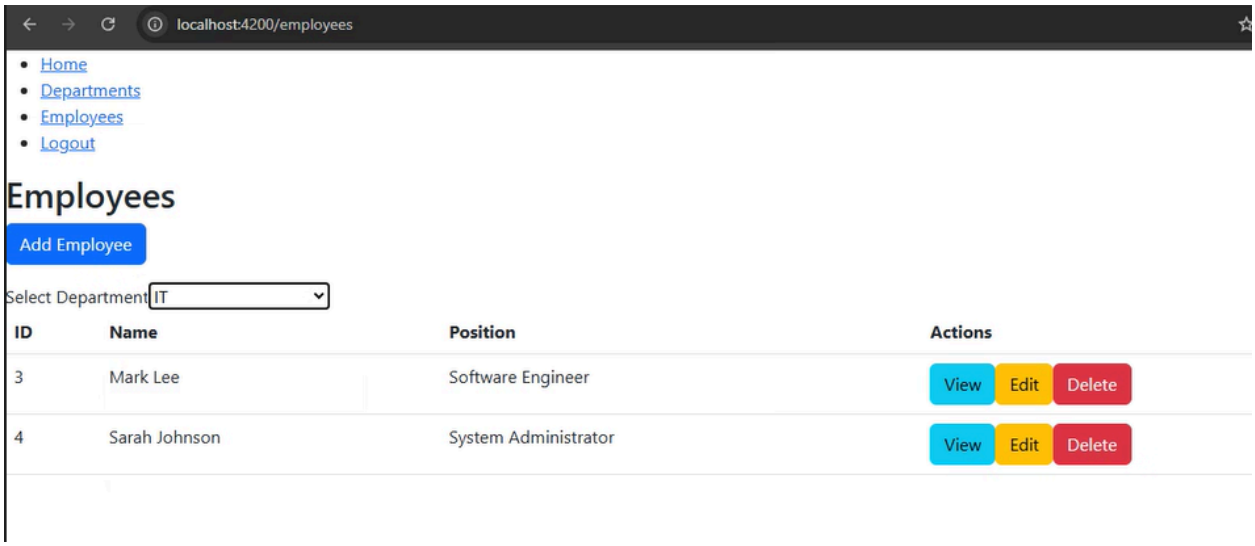


***** Select Department *****

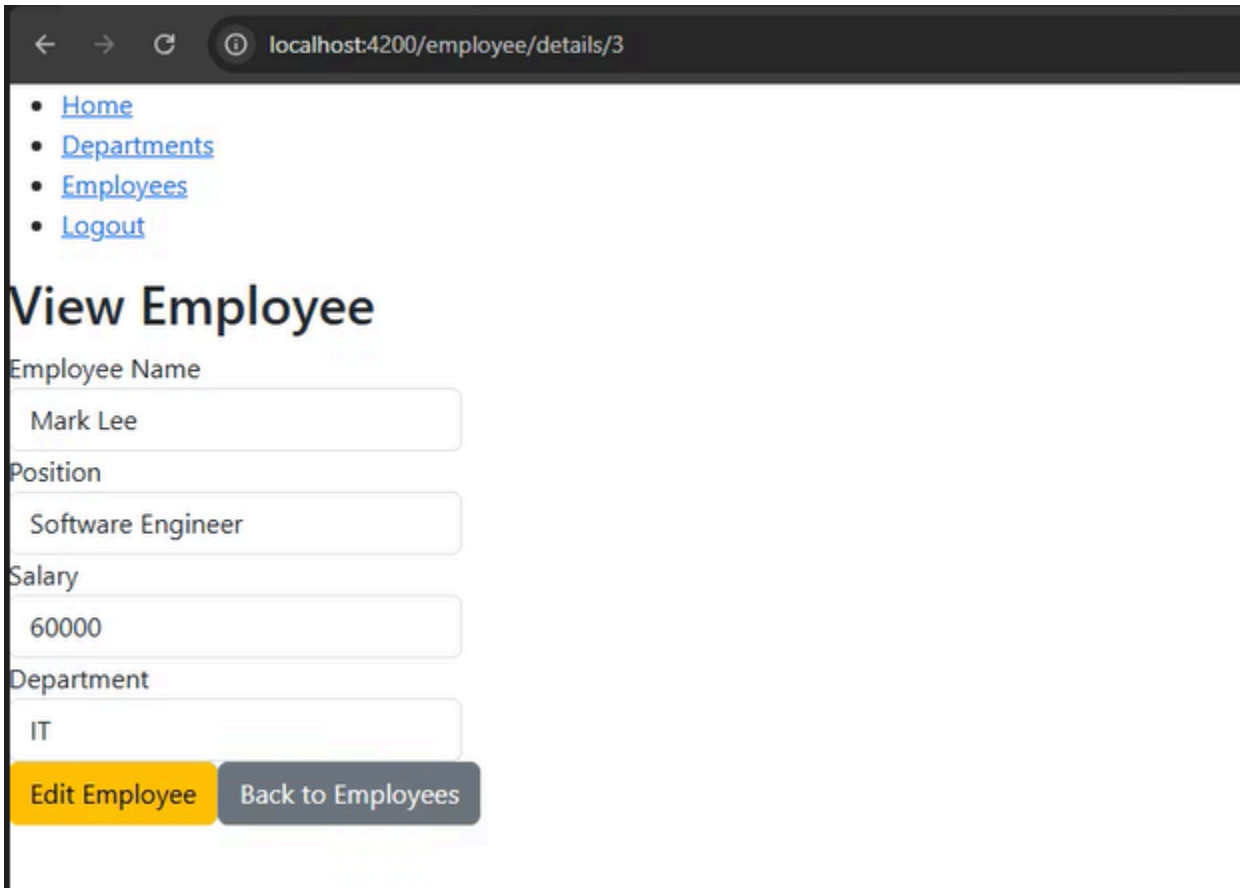
*** Employees will only be displayed after selecting a department from the dropdown; initially, the list remains hidden, and once a department is chosen, the relevant employees are fetched and shown dynamically.**



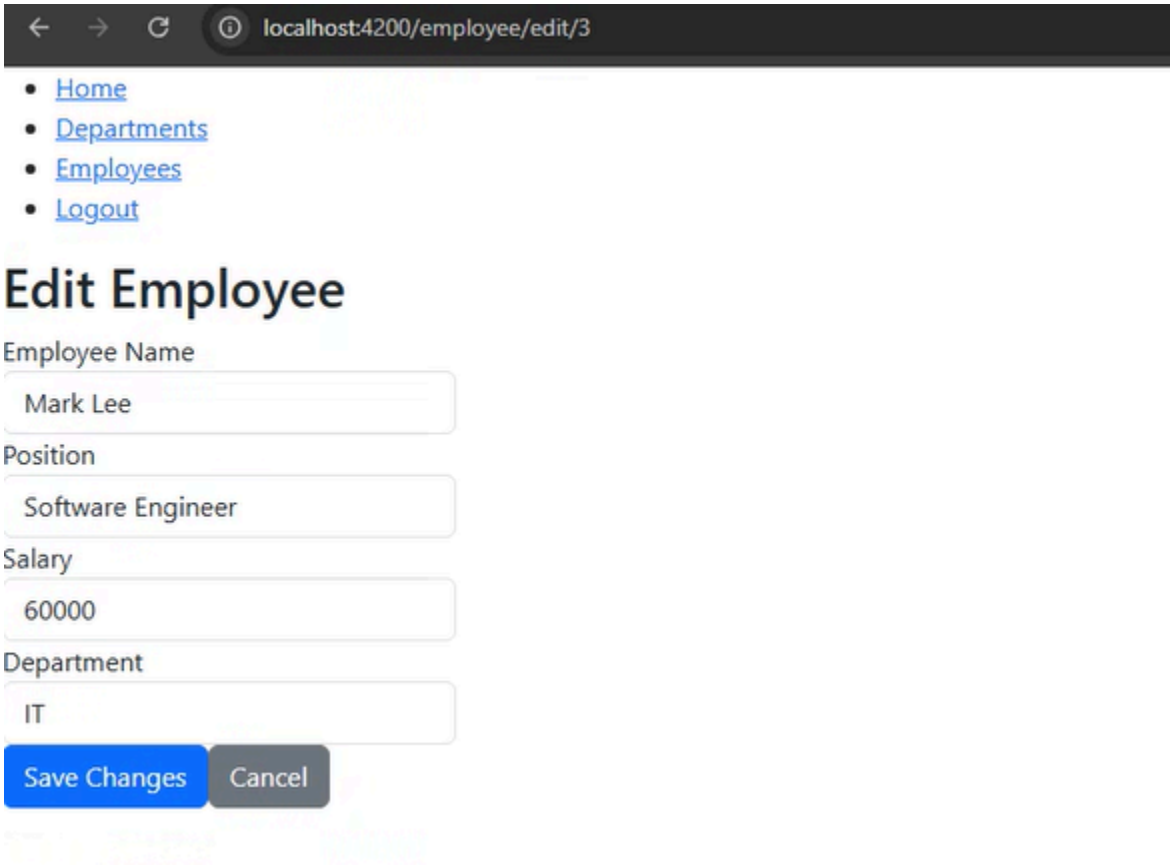
***** Employees List *****



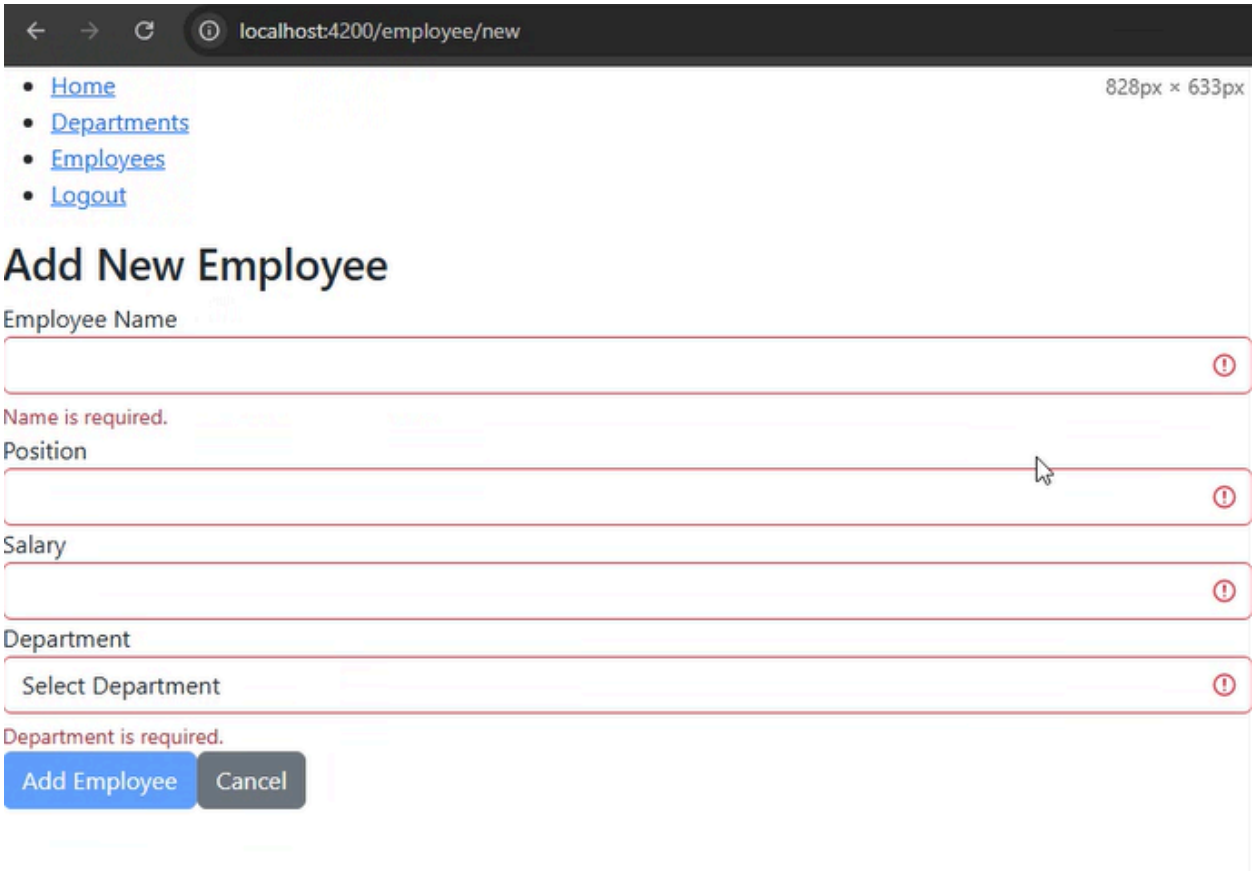
***** View Employee *****



***** Edit Employee *****



***** Add New Employee *****



*** If any field (Employee Name, Position, Salary, or Department) is not filled, the respective validation error should be displayed, and the "Add Employee" button should remain disabled; it should only be enabled once all fields are correctly filled.**

*** Once the Logout button is clicked, the user should be logged out and automatically redirected to the Login Page to ensure secure session management.**

*** The Home, Departments, Employees, and Logout buttons should be available on all pages, ensuring seamless navigation until the user logs out, after which they should be redirected to the Login Page.**

---

5. Execution Steps To Follow For Frontend

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

3. This is a web-based application, to run the application on a browser, use the internal browser in the environment.
4. You can follow series of command to setup Angular environment once you are in your project-name folder:
   a. npm install -> Will install all dependencies -> takes 10 to 15 min
   b. npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min
   c. npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min
5. You need to push your code compulsorily on code IDE, before final submission as well.
   1. git add .
   2. git commit -m "adding files"
   3. git push.

   This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.