Phase-End Project

Event Management App

- **Algorithm**

Step 1: Start the application.

Step 2: Define the necessary routes in the app-routing.module.ts file to navigate between different components.

Step 3: Create the database JSON file (db.json) with an initial list of employees.

Step 4: Implement the login functionality in the LoginComponent:

- Bind the input fields to the username and password properties using two-way data binding.
- On form submission, check if the entered username and password match the predefined admin credentials.
- If the credentials are correct, navigate to the main employee management page (/employees).
- If the credentials are incorrect, display an error message.

Step 5: Implement the employee management functionality in the EmployeeManagementComponent:

- Fetch the list of employees from the JSON database using the HttpClient in the ngOnInit method.
- Display the list of employees in a table using *ngFor directive.
- Provide options to add, view, update, and delete employees.

Step 6: Implement the add employee functionality in the AddEmployeeComponent:

- Create a form to enter the employee details (first name, last name, email).
- On form submission, send an HTTP POST request to add the new employee to the database.
- Clear the form and refresh the employee list.

Step 7: Implement the employee details functionality in the EmployeeDetailsComponent:

- Retrieve the employee ID from the route parameters.
- Fetch the employee details from the JSON database based on the ID.
- Display the employee details in a table.
- Provide options to go back, update, and delete the employee.

Step 8: Implement the update employee functionality in the UpdateEmployeeComponent:

- Retrieve the employee ID from the route parameters.
- Fetch the employee details from the JSON database based on the ID.
- Populate the form fields with the current employee details.
- On form submission, send an HTTP PUT request to update the employee details in the database.

- Navigate back to the employee details page.

Step 9: Implement the delete employee functionality in the EmployeeDetailsComponent:

- Provide a button to delete the employee.
- On button click, send an HTTP DELETE request to remove the employee from the database.
- Show a success message and navigate back to the employee management page.
- Style the components using CSS and Bootstrap classes to achieve the desired UI.
- Test the application for various scenarios and ensure proper functionality.

- **Sprint Planning**

✓ **Sprint Goal:** 2 weeks

✓ **Sprint Duration:** Implement a functional employee management application with login, CRUD operations, and proper routing.

✓ **User Stories:**

1. As a user, I want to be able to log in with my credentials and access the event management page.

Tasks:

- Implement the LoginComponent with username and password fields.
- Create the necessary routes in the app-routing.module.ts file.
- Add validation for login credentials.
- Handle successful login and navigation to the employee management page.
- Handle incorrect login credentials and display an error message.

2. As a user, I want to view the list of employees and perform CRUD operations on them.

Tasks:

- Create the database JSON file (db.json) with an initial list of employees.
- Implement the EmployeeManagementComponent.
- Use the HttpClient to fetch the list of employees from the JSON database.
- Display the list of employees in a table using *ngFor directive.
- Add options to add, view, update, and delete employees.
- Implement the add employee functionality in the AddEmployeeComponent.
- Implement the delete employee functionality in the EmployeeDetailsComponent.
- Implement the update employee functionality in the UpdateEmployeeComponent.

3. As a user, I want to view the details of each employee.

Tasks:

- Implement the EmployeeDetailsComponent.
- Retrieve the employee ID from the route parameters.
- Fetch the employee details from the JSON database based on the ID.
- Display the employee details in a table.
- Add options to go back, update, and delete the employee.

4. As a user, I want to update employee details.

Tasks:

- Implement the UpdateEmployeeComponent.
- Retrieve the employee ID from the route parameters.
- Fetch the employee details from the JSON database based on the ID.
- Populate the form fields with the current employee details.

- On form submission, send an HTTP PUT request to update the employee details in the database.
- Navigate back to the employee details page.

5. As a user, I want to navigate between different components and have a well-styled UI.

Tasks:

- Implement proper routing between components.
- Style the components using CSS and Bootstrap classes to achieve the desired UI.

6. As a developer, I want to test the application for various scenarios and ensure proper functionality.

Tasks:

- Write unit tests for each component and service.
- Test the application for edge cases and handle any potential issues.