WRITE-UP FOR EVENT-MANAGEMENT

1. Set up an Angular development environment on your computer:

To set up an Angular development environment,

- Install Node.js from https://nodejs.org/en/

- Verify the installation by running `node -v` and `npm -v` on the terminal or command prompt.

- Install Angular CLI using `npm install -g @angular/cli`

- Verify the installation using `ng version`

2. Create a new Angular project using the Angular CLI:

To create a new Angular project, run `ng new my-app` on the terminal or command prompt. Replace `my-app` with a suitable project name.

3. Choose a suitable project name and directory location for your Employee management app:

When creating a new project using the Angular CLI, you can specify the project name and directory location (using the `-dir` option). Choose a suitable project name and directory location for your Employee management app.

4. Install the required Angular modules and packages for your app:

To install the required Angular modules and packages, navigate to the project directory and run `npm install`. This will install all the dependencies specified in the `package.json` file.

5. Create the components for Employee, Login, Logout, Home, and NavBar using the Angular CLI:

To create a new component using the Angular CLI, run `ng generate component component-name`. Replace `component-name` with the name of the component you want to create.

6. Set up the component templates and stylesheets for each of the components:

Each component created by the Angular CLI comes with its own HTML template and CSS stylesheet. Edit these files to create the desired look and feel.

7. Define the component properties and methods that are needed for each component:

Each component should have properties and methods that define its behavior. Define these in the component class.

8. Implement basic functionality for each component:

Add functionality to each component to implement its basic behavior. For example, the Employee component may display a list of employees, while the Login component may validate user input.

9. Create a service to manage employee data and perform CRUD operations on it:

Create a new service using `ng generate service service-name`. Replace `service-name` with a suitable name for your service. Implement methods in the service to manage employee data.

10. Define routes for each of the components using the Angular Router:

Define routes for each component in the `app-routing.module.ts` file using the Angular Router.

11. Set up a router outlet in your app's template to render the active component based on the current route:

Add a `<router-outlet>` element to your app's template. This element will render the active component based on the current route.

12. Create a navigation menu in your app's template using the NavBar component:

Create a new NavBar component using `ng generate component nav-bar`. Add the necessary HTML and CSS to create a navigation menu.

13. Configure the navigation menu to use the Angular Router to navigate between routes:

Add links to the navigation menu that navigate to the appropriate routes using the Angular Router.

14. Add authentication functionality to the Login and Logout components using Firebase Authentication or a similar service:

Follow the documentation of Firebase Authentication or a similar service to implement authentication functionality in the Login and Logout components.

15. Define guards to restrict access to certain parts of your app based on the user's login status:

Create new guards using `ng generate guard guard-name`. Replace `guard-name` with a suitable name for your guard. Implement the necessary functionality to restrict access based on the user's login status.

16. Implement logic to redirect the user to the appropriate page based on their login status:

Add the necessary logic to redirect the user to the appropriate page based on their login status.

17. Create a home page component to serve as the landing page for your app:

Create a new Home component using `ng generate component home`. Define the necessary HTML and CSS to create a landing page.

18. Add content and styling to the home page component as desired:

Add the necessary content and styling to the Home component to create a visually appealing landing page.

19. Implement functionality to allow users to log in and out of the app using Firebase Authentication or a similar service:

Implement the necessary functionality to allow users to log in and out of the app using Firebase Authentication or a similar service.

20. Define a user model to keep track of user information:

Create a new TypeScript interface to define the structure of the user model.

21. Create a service to manage user data and perform relevant operations:

Create a new UserService using `ng generate service user`. Implement the necessary functionality to manage user data.

22. Configure the NavBar component to display appropriate navigation links based on the user's login status:

Update the NavBar component to display links based on the user's login status. For example, display a "Logout" link when the user is logged in.

23. Add styling to the NavBar component to make it more visually appealing:

Add the necessary CSS to the NavBar component to create a visually appealing navigation menu.

24. Refactor your code to improve its structure and readability:

Clean up your code by removing duplicate or unnecessary code, renaming variables and functions to improve readability, and organizing your code into logical sections.

25. Add comments to your code to help other developers understand its functionality:

26. Test your app on different browsers and devices to ensure cross-browser compatibility and responsiveness:

27. Use debug tools to troubleshoot any errors or issues that arise during testing:

28. Optimize your app for performance by minimizing the number of HTTP requests, reducing the size of assets, and compressing data where possible:

29. Implement automated testing using unit tests, end-to-end tests, or a similar framework:

Implement automated testing using tools like Jasmine, Protractor, or a similar framework to ensure that your app works as expected.