Neo Party Hall Booking Application

Objective:

Neo Party Hall is an application to be built as a product that can help customers to choose a Party Hall and book a Hall for Party for a specific date..

Users of the System:

1. Super Admin

2. Admin

3. Customer

Functional Requirements:

- Build an application that customers can book Party Hall.
- The application should have signup, login, profile, dashboard page for user and login, signup, profile, dashboard, add hall page for the admin.
- This application should have a provision to maintain a database for customer information, Admin information, Booking information and hall information.
- Also, an integrated platform required for customers, admin and super admin.
- Administration module to include options for adding / modifying / removing the existing product(s) and customer management.
- **There should be one user per booking.**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

☐ Filters for products like Low to High or showcasing halls based on the customer's price range, specific date etc.

☐ Email integration for intimating new personalized offers to customers.

☐ Multi-factor authentication for the sign-in process

☐ Payment Gateway

Output/ Post Condition:

☐ Records Persisted in Success & Failure Collections

☐ Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	● App Platform –UserName/Password-Based Credentials ● Sensitive data has to be categorized and stored in a secure manner ● Secure connection for transmission of any data
Performance	● Peak Load Performance (during Festival days, National holidays etc) ● eCommerce -< 3 Sec

	● Admin application < 2 Sec ● Non Peak Load Performance ● eCommerce < 2 Sec ● Admin Application < 2 Sec
Availability	● 99.99 % Availability
Standard Features	● Scalability ● Maintainability ● Usability ● Availability ● Failover
Logging & Auditing	● The system should support logging(app/web/DB) & auditing at all levels
Monitoring	● Should be able to monitor via as-is enterprise monitoring tools
Cloud	● The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	● IE 7+ ● Mozilla Firefox Latest – 15 ● Google Chrome Latest – 20 ● Mobile Ready

Technology Stack

Front End	Angular 7+ Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Prerequisites (Do's and Don'ts):

1. The angular app should run in port 8081. Do not run the angular app in the port: 4200.

2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.

2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.

3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.

4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.

2. Manual routing should be restricted by using Auth Guard by implementing the canActivate interface. For example, if the user enters as http://localhost:8080/signup or http://localhost:8080/home the page should not navigate to the corresponding page instead it should redirect to the login page.

3. Unless logged into the system, the user cannot navigate to any other pages.

4. Logging out must again redirect to the login page.

5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.

6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.

2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

Admin Side:

Action	URL	Method	Response
Admin Login	/admin/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/admin/signup	POST-Sends Admin Model data	Admin added
Admin Dashboard	/admin/dashboard	GET	Return All the halls in the resort

Admin Add Hall	/admin/addHall	POST-Sends Hall Data	Hall Added
Admin Edit Hall	/admin/editHall	POST-Sends Hall Data	Hall Edited
Admin Delete Hall	/admin/deleteHall	POST-Sends Hall ID	Hall Deleted
Admin Profile	/admin/profile	POST-Sends Admin ID	Return Admin Profile Details
Edit Profile	/admin/editProfile	GET-Sends Admin ID	Return Admin Profile Details
Edit Profile	/admin/editProfile	POST-Sends Admin Model data	Edits Admin Profile

User Side:

Action	URL	Method	Response
User Login	/user/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/user/signup	POST-Sends User Model data	User added
User Dashboard	/user/dashboard	GET	Return all the Halls available
Displaying Halls	/user/halls	POST - Sends Hall name and admin ID of that hall	Return all the availability of halls
Hall Details	/user/hallDetails	POST-Sends Halls ID	Return Hall details
Booked Halls	/user/bookings	POST-Sends User ID	Return user bookings

Super Admin:

Action	URL	Method	Response
Super Admin Login	/super/login	POST-Sends email ID and password	Return True/False
Delete an admin	/super/deleteAdmin	POST-Sends admin email ID	Delete an admin
Delete a User	/super/deleteUser	POST-Sends user email ID	Delete a user

Frontend:

Customer:

1. Signup: Design a signup page component where the new customer has options to sign up by providing their basic details.

    a. Ids:

        i. signupBox

        ii. email

        iii. password

        iv. mobilenumber

        v. userrole

        vi. username

        vii. age

        viii. submitButton

        ix. loginLink

    b. Routing Url: http://localhost:4200/user/signup

    c. Output screenshot:

SIGN UP

Enter Email → id = "email"

Enter Password → id = "password"

Enter Mobile Number → id = "mobilenumber"

User ⌄ → id = "userrole"

Enter Username → id = "username"

Enter Age → id = "age"

Submit → id = "submitButton"

Go to Login Click Here → id = "loginLink"
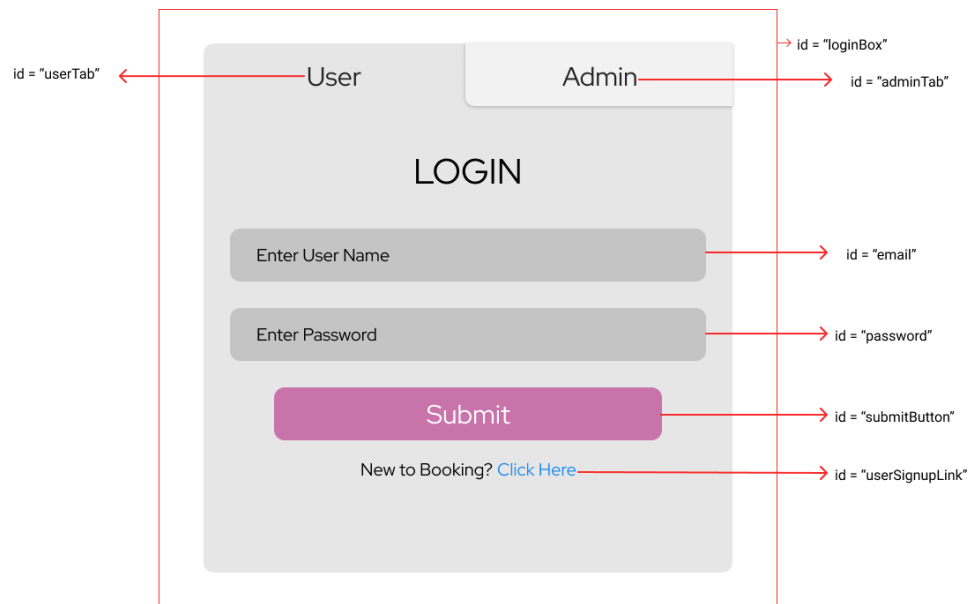
→ id = "signupBox"

2. Login: Design a login page component where the existing customer can log in using the registered email id and password.

    a. Ids:

        i. loginBox

        ii. userTab

        iii. email

        iv. password

        v. submitButton

        vi. userSignupLink

    b. Routing Url: http://localhost:4200/login

    c. Output screenshot:

3. Dashboard / Home: Design a dashboard page component which provides a list of companies available where the user can book a particular Hall.
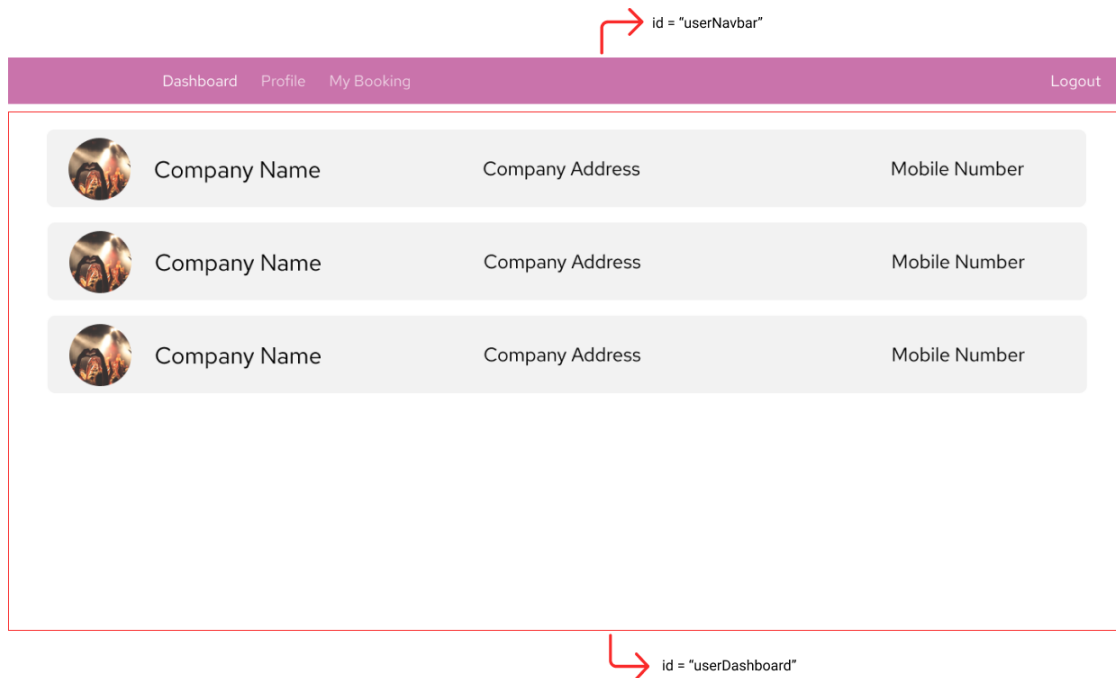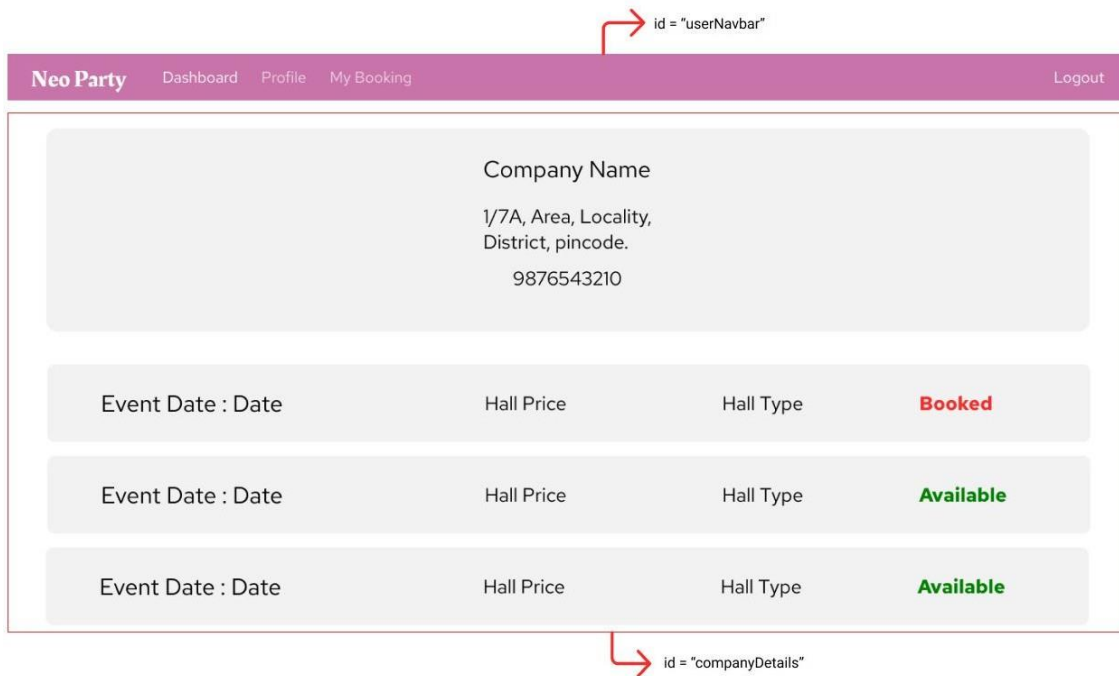
    a. Ids:

        i. userNavbar

        ii. userDashboard

    b. Routing Url: http://localhost:4200/user/dashboard

    c. Screenshot

id = "userNavbar"

| Dashboard | Profile | My Booking | | | Logout |

	Company Name	Company Address	Mobile Number
	Company Name	Company Address	Mobile Number
	Company Name	Company Address	Mobile Number

id = "userDashboard"

4. Company Details: Design a Company Details component which lists available dates along with the individual status of each day (available / Booked).

    a. Ids

        i. userNavbar

        ii. companyDetails

    b. Routing Url : http://localhost:4200/user/companyDetail/{companyId}

    c. Screenshot

5. **Hall Details:** Design a Hall Booking Page that displays the details of the hall that the useris currently booking.
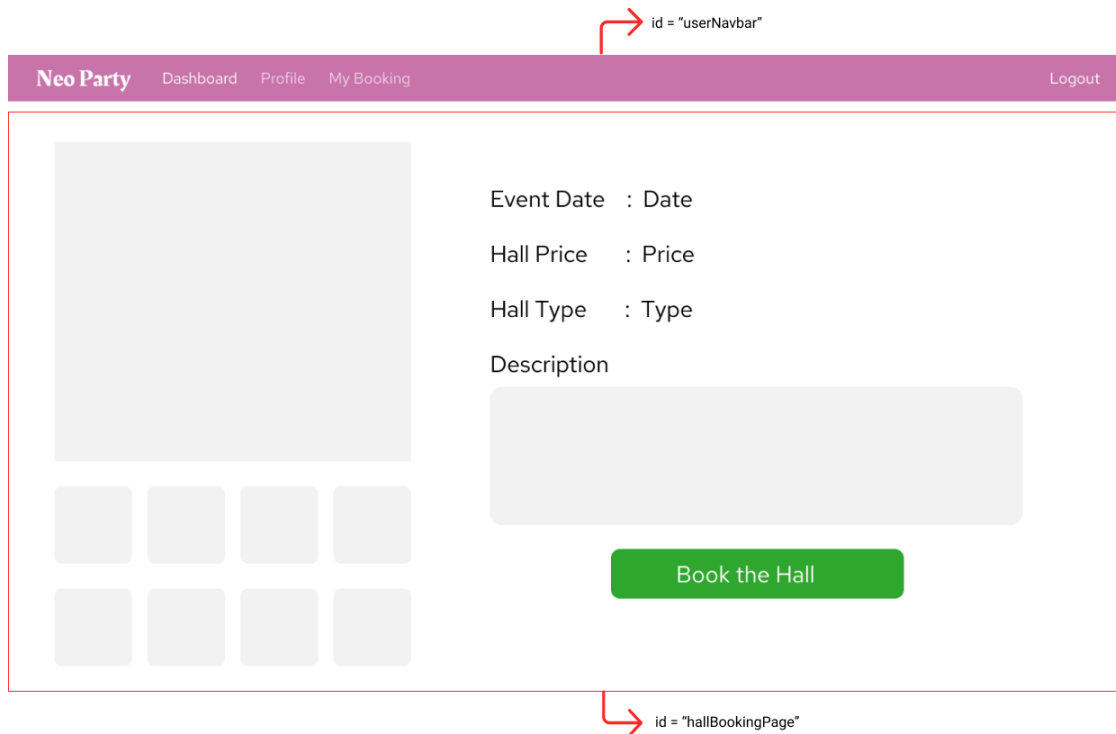
    a. Ids

        i. userNavbar

        ii. hallBookingPage

    b. Routing Url : http://localhost:4200/user/hallDetail/{hallId}

    c. Screenshot

Neo Party   Dashboard   Profile   My Booking   Logout

Event Date   : Date

Hall Price   : Price

Hall Type   : Type

Description

Book the Hall

id = "hallBookingPage"

6. User Navigation: Design a Booking page component where the user can navigate to their profile and bookings page.

    a. Ids

        i. userNavbar

        ii. userProfleLink

        iii. userDashboardLink

        iv. userBookingLink

        v. userLogoutButton

    b. Screenshot
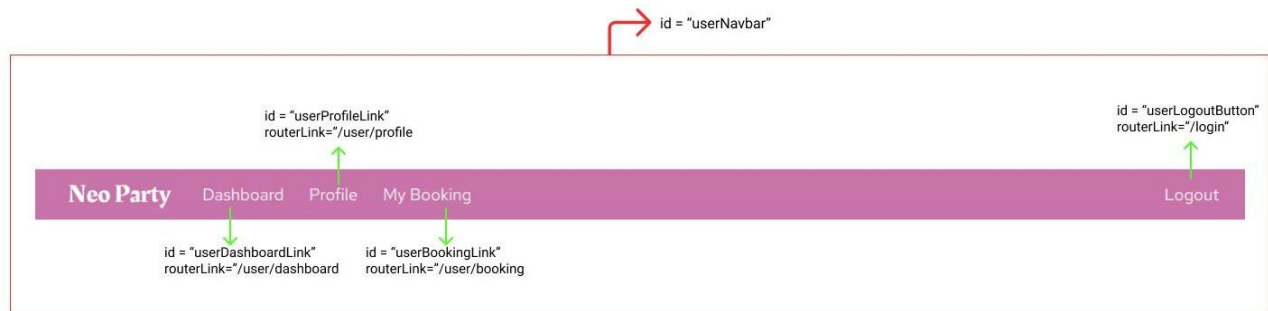
7. User Profile Edit: Design a Profile Edit Page where the user can edit the details of him/herself.

    a. Ids

        i. userNavbar

        ii. editProfileBox

        iii. username

        iv. email

        v. password

        vi. userAge

        vii. mobilenumber

        viii. editProfileButton

    b. Routing Url : http://localhost:4200/user/editProfile/{userId}

    c. Screenshot:

| Neo Party | Dashboard | Profile | My Booking | | Logout |

back

Name

| Jennifer Aniston | → id = "editProfileBox"
 → id = "username"

Email

| rachelgreen@iamneo.ai | → id = "email"

Password

| JoeyTribbiani | → id = "password"

Age

| 27 | → id = "userAge"

Mobile Number

| 1234567890 | → id = "mobilenumber"

| Save Changes | → id = "editProfileButton"

8. **User Bookings:** Design a User Bookings Page that allows the user to view the bookings that the user has made till date.

    a. Ids:

        i. userNavbar

        ii. userBookingBody

    b. Routing Url : http://localhost:4200/user/bookings/{userId}

    c. Screenshot:

id = "userNavbar"

| Neo Party | Dashboard | Profile | My Booking | Logout |

Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price
Company Name	Event Date	Hall Price	Days	Total Price

id = "userBookingBody"

9. **User Profile Page:** Design a User Profile Page that allows user to view their Information.

    a. Ids:

i. userNavbar

ii. userProfileBody

b. Routing Url : http://localhost:4200/user/profile/{userId}

c. Screenshot:
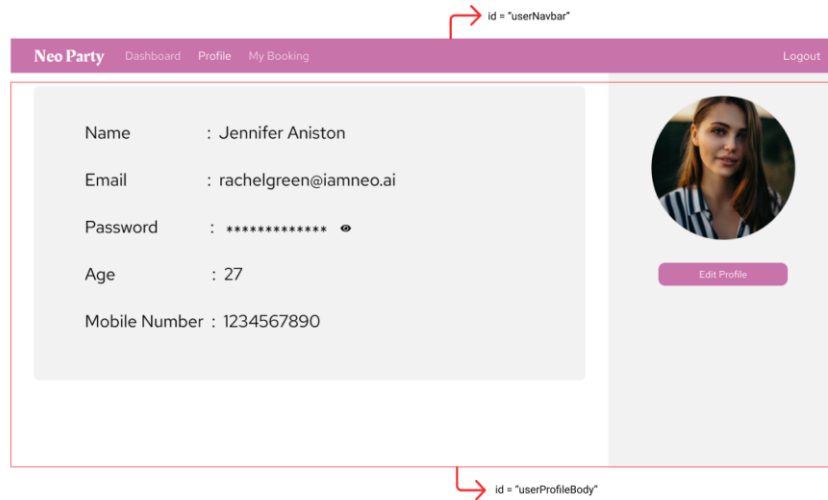
Admin:

10. Admin Signup: Design a Signup page that registers a particular user with the role of Admin.

    a. Ids

        i. signupbox

        ii. email

        iii. password

        iv. mobilenumber

        v. userrole

        vi. adminname

        vii. companyname

        viii. companyimageURL

        ix. companyAddress

        x. submitButton

        xi. adminLoginLink

    b. Routing Url : http://localhost:4200/admin/signup

    c. Screenshot

11. Admin Login: Design an Admin Login page that allows admin to Login

    a. Ids

        i. loginBox

        ii. userTab

        iii. email

        iv. password

        v. submitButton

        vi. adminSignupLink

    b. Routing URL: http://localhost:4200/login

    c. Screenshots

12. Admin Profile: Design an Admin Profile page that displays the details of the admincurrently logged in.

    a. Ids

        i. adminNavbar

        ii. adminProfile

        iii. editAdminProfile

    b. Routing Url: http://localhost:4200/admin/profile/{adminId}

    c. Screenshots

id = "adminNavbar"

id ="editAdminProfile"

id = "adminProfile"
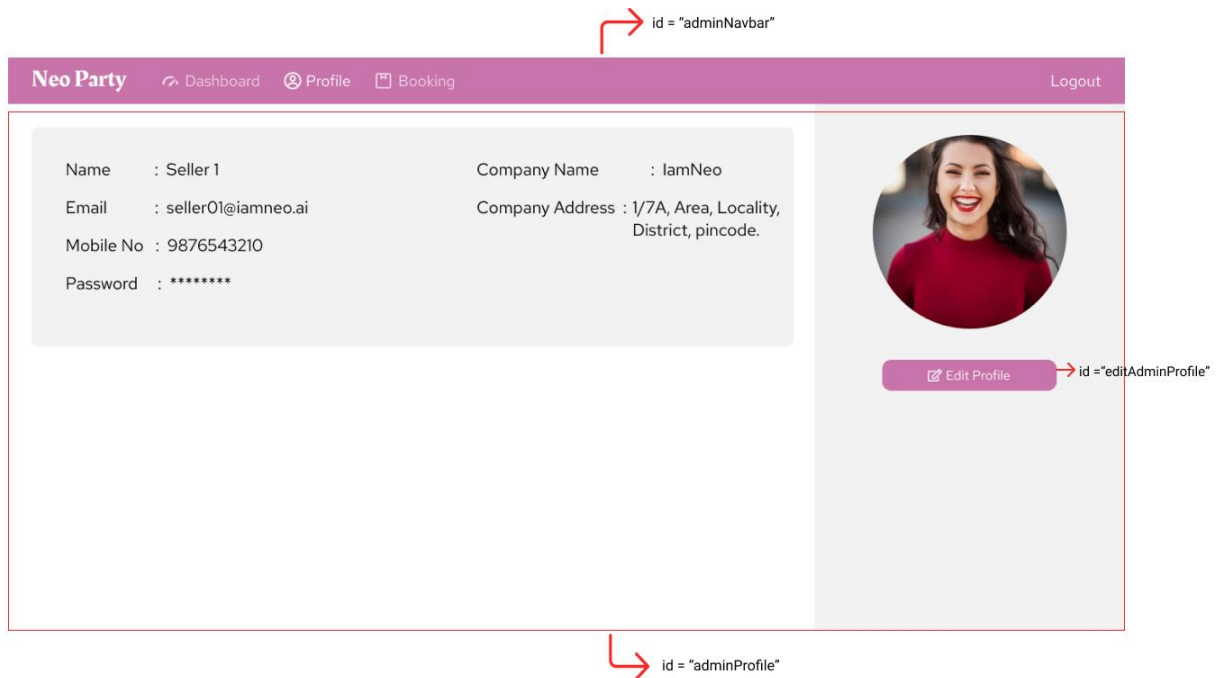
13. **Admin Dashboard:** Design a dashboard page where the list of products is displayed on the admin side.

    a. Ids

        i. adminNavbar

        ii. addHallButton

        iii. adminDashboard

    b. Routing Url: http://localhost:4200/admin/dashboard

    c. Screenshot
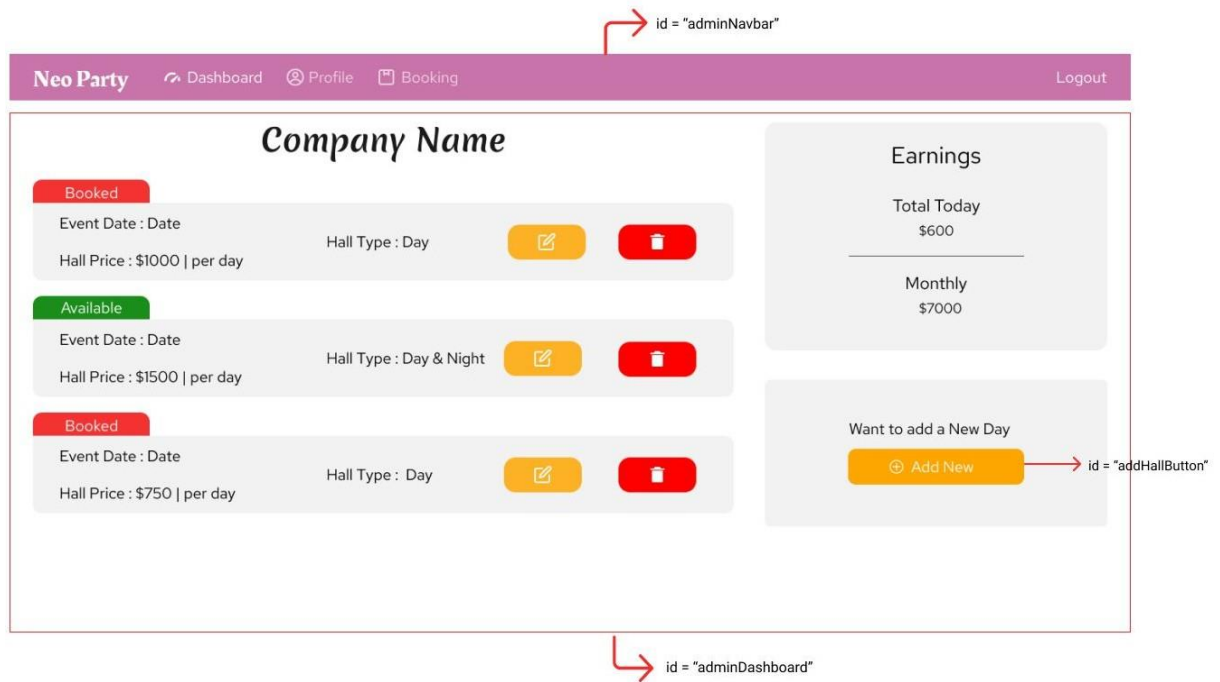
14. Admin Navigation: Design an Admin navigation component that can navigate to the Dashboard, profile and bookings done by the users

   a. Ids:

      i. adminNavbar

      ii. adminProfileLink

      iii. adminDashboardLink

      iv. adminBookingLink

      v. adminLogoutButton

   b. Screenshot:
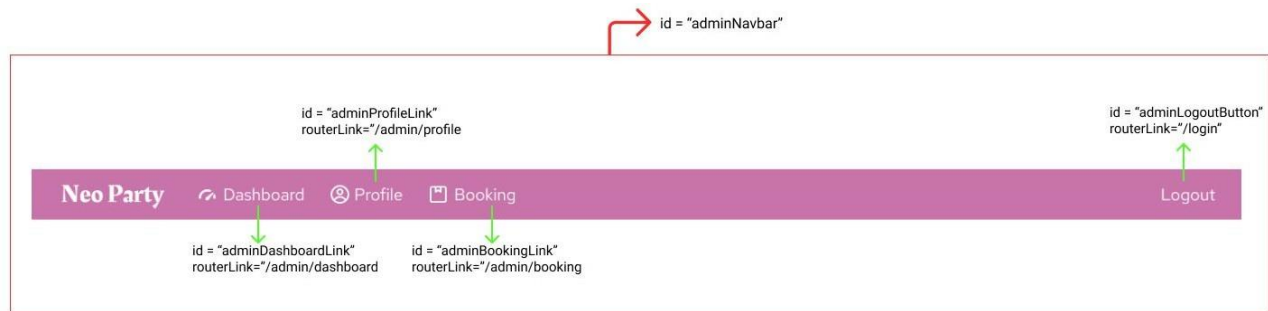
15. Admin Add Date: Design an Add Date component in which the admin can add new dates to the inventory.

    a. Ids:

        i. adminNavbar

        ii. addHallBox

        iii. hallNo

        iv. hallPrice

        v. hallType

        vi. addHallButton

    b. Routing Url: http://localhost:4200/admin/addHall
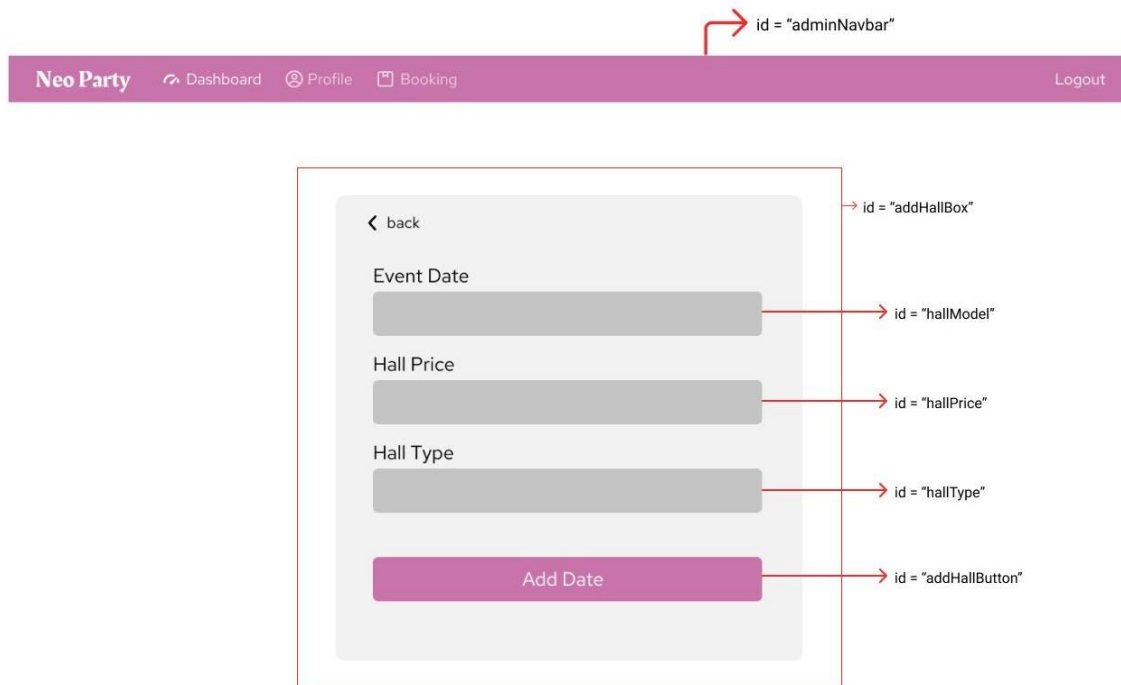
    c. Screenshot

16. Admin Edit Hall: Design an Edit Hall component in which the admin can edit details of an already existing hall in the inventory.

    a. Ids:

        i. adminNavbar

        ii. editHallBox

        iii. hallNo

        iv. hallPrice

        v. hallType

        vi. editHallButton

    b. Routing Url: http://localhost:4200/admin/editHall/{hallId}
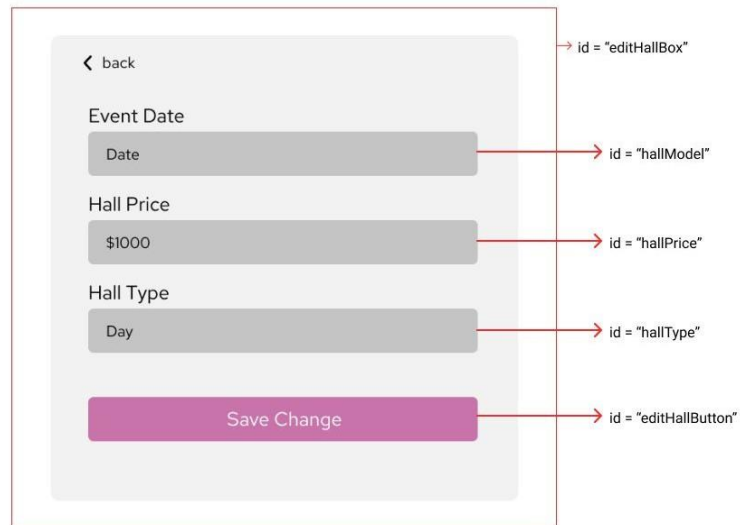
    c. Screenshot

17. Admin Bookings: Design an Admin Bookings component where the admin can seethe booking details that all the users have made till present date.

    a. Ids

        i. adminNavbar

        ii. adminBookings

        iii. bookingList

    b. Routing Url : http://localhost:4200/admin/bookings/{adminId}

    c. Screenshot:
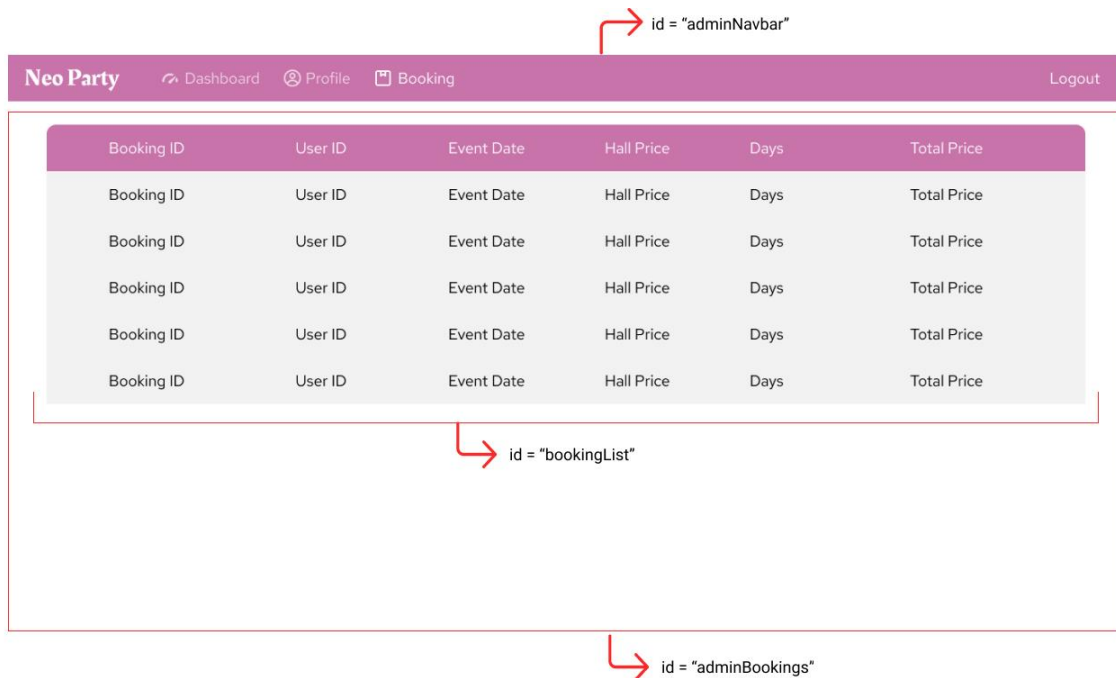
18. Admin Profile Edit: Design an Admin Profile Edit page where the admin can edit the details of himself.

    a. Ids

        i. adminEditBox

        ii. adminName

        iii. adminEmail

        iv. adminMobileNumber

        v. adminPassword

        vi. companyName

        vii. companyAddress

        viii. profileEditButton

    b. Routing Url : http://localhost:4200/admin/editProfile/{adminId}

    c. Screenshot:

Super Admin:

19. Super Admin Login Page: Design a Login page through which the super user can log in.

   a. Ids:
      i. superAdminLoginBox
      ii. email
      iii. password
      iv. submitButton

   b. Routing Url : http://localhost:4200/superadmin/login

   c. Screenshot:

id = "superAdminLoginBox"

Super Admin

Enter Super Admin Email → id ="email"

Enter Password → id ="password"

Submit → id ="submitButton"

20. Super Admin Admins page : Design a Super-Admin admin page where the superadmin has authority to view and delete any admin currently registered into the system.

    a. Ids:

        i. superAdminNavbar

        ii. allUserDetails

    b. Routing Url : http://localhost:4200/superadmin/adminList

    c. Screenshot

id = "superAdminNavbar"

id = "allUserDetails"
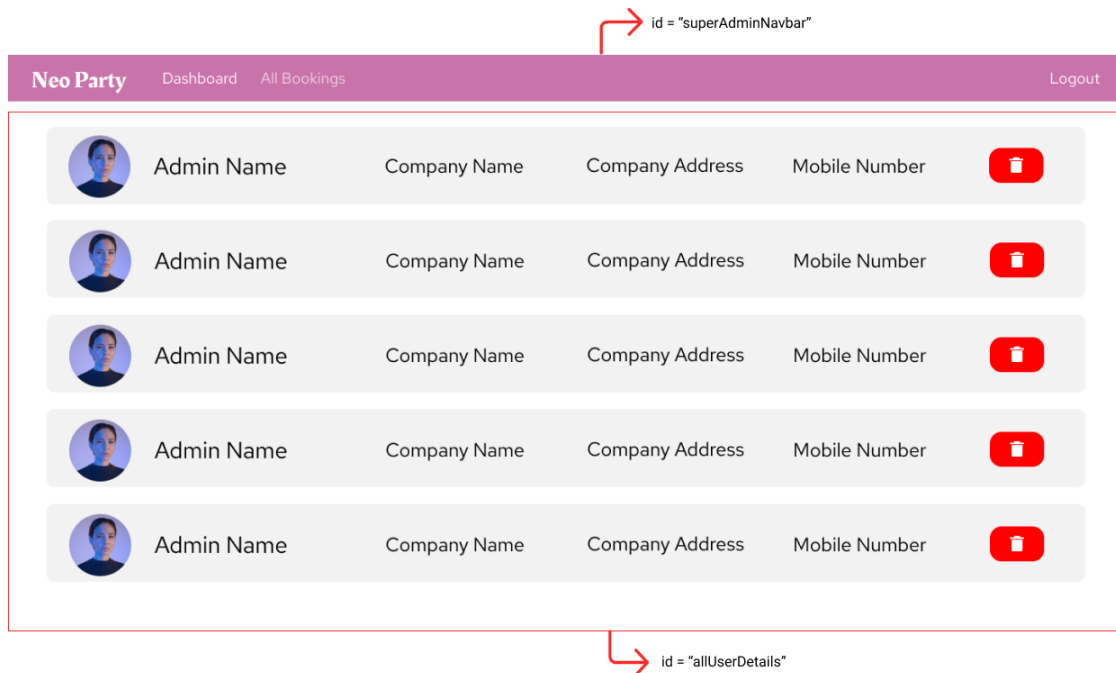
21. Super Admin Bookings page: Design a Super-Admin Booking component where the super admin has authority to view the bookings.

    a. Ids:

        i. superAdminNavbar

        ii. allAdminDetails

    b. Routing Url : http://localhost:8080/superadmin/adminBookings

    c. Screenshot

id = "superAdminNavbar"

Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price
Admin ID	Company Name	Event Date	Hall Price	Days	Total Price

id = "allAdminDetails"

22. Super Admin Navigation: Design a Navigation page for the super admin where hecan navigate to the dashboard, all the bookings and logout.

   a. Ids:

      i. superAdminNavbar

      ii. superAdminBookingLink

      iii. superAdminLogoutButton

      iv. superAdminDashboardLink

   b. Screenshot:

Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the customer) and all user information.

   a. Attributes:

      i. email: String

      ii. password: String

      iii. username: String

      iv. mobileNumber: String

      v. age: int

      vi. userRole: String

2. LoginModel: This class contains the email and password of the user.

   a. Attributes:

      i. email: String

      ii. password: String

3. AdminModel: This class stores the details of the product.
   a. Attributes:
      i. email:String
      ii. password:String
      iii. mobileNumber:String
      iv. sellerName:String
      v. userRole:String
      vi. companyName:String
      vii. companyImageURL:String
      viii. companyAddress: String
      ix. earnings:int
4. HallModel: This class stores the cart items.
   a. Attributes:
      i. hallID:String
      ii. hallNo:String
      iii. adminID:String
      iv. status:String
      v. price:String
      vi. type:String

Controller Layer:

5. AuthController: This class control the user /admin signup and signin
   a. Methods:
      i. isUserPresent(LoginModel data): This method helps to check whether the user present or not and check the email and password are correct and return the boolean value.
      ii. isAdminPresent(LoginModel data): This method helps to check whether the admin present or not and check the email and password are correct and return the boolean value.
      iii. saveUser(UserModel user): This method helps to save the user data in the database.
      iv. saveAdmin(UserModel user): This method helps to save the admin data in the database.
6. HallController: This class controls the save/edit/delete/view halls.

a. Methods:

    i. saveHall(HallModel data): This method helps the admin to savenew halls in the database.

    ii. editHall(HallModel data): This method helps the admin to edit the details of the halls and save it again in the database.

    iii. deleteHall(String Hall ID): This method helps the admin to deletea hall from the company and as well as in the database.

    iv. getHalls(String Admin_Email_ID): This method helps the admin to get all the halls in the company.

    v. bookHall(String Hall_ID): This method helps the user to book the hall by changing the hall status and add the record in the bookingstable.

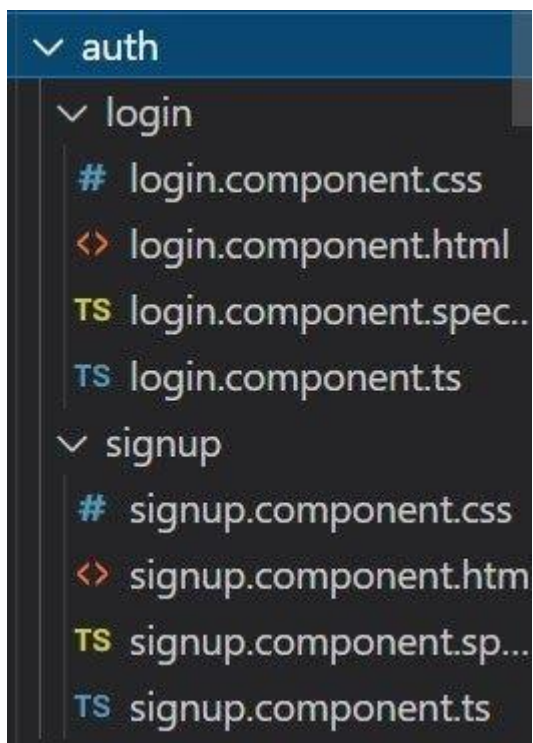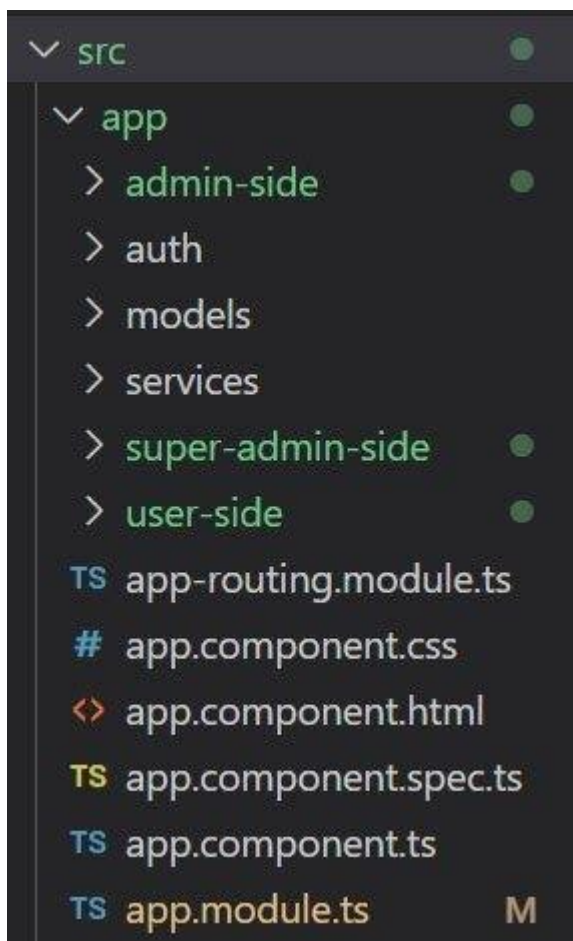7. AdminController: This class controls the edit/view admin details.

    a. Methods：

        i. editAdmin(AdminModel data): This method helps the admin to edit the profile details and save it in the database.

        ii. getProfile(String Admin_Email_ID): This method helps admin to get the profile details.

8. UserController: This class helps to get the halls.

    a. Methods:

        i. userBookings(String User_ID):This method helps users to get all the booking details done by them.

        ii. userProfileEdit(String User ID): This method helps users to edit their own details.

        iii. userBookHall(): This method helps user to book a particular date.

        iv. userGetBookings(String UserID): This method helps to get all the bookings of the particular user.

Angular Folder Structure:

```
∨ src
  ∨ app
    > admin-side
    > auth
    > models
    > services
    > super-admin-side
    > user-side
    TS app-routing.module.ts
    #  app.component.css
    <> app.component.html
    TS app.component.spec.ts
    TS app.component.ts
    TS app.module.ts                M
```

```
∨ auth
  ∨ login
    #  login.component.css
    <> login.component.html
    TS login.component.spec..
    TS login.component.ts
  ∨ signup
    #  signup.component.css
    <> signup.component.htm
    TS signup.component.sp...
    TS signup.component.ts
```

admin-side
- admin-dashboard
- admin-navbar
- admin-profile
- \# admin-side.comp... U
- <> admin-side.comp... U
- TS admin-side.comp... U
- TS admin-side.comp... U

user-side
- user-dashboard
- user-navbar
- user-profile
- \# user-side.compon... U
- <> user-side.compon... U
- TS user-side.compon... U
- TS user-side.compon... U

super-admin-side
- super-admin-dashboard
- super-admin-navbar
- \# super-admin-side.compone... U
- <> super-admin-side.compone... U
- TS super-admin-side.compone... U
- TS super-admin-side.compone... U

NOTE:

You should create the above folder structure mandatorily to pass the test cases and you can also create extra components if you need.

Workflow Prototypes:

Admin Flow
https://www.figma.com/proto/5QVhQYuw1tYAKFFmtVksCc/Neo-Party-Hall-Admin-Flow?node-id=1%3A482&viewport=556%2C197%2C0.03661709651350975&scaling=scale-down

User Flow
https://www.figma.com/proto/f9BG3VvxHTOTpzXNi9C5zy/Neo-Party-Hall-User-Flow?node-id=1%3A33&viewport=476%2C203%2C0.04019034653902054&scaling=scale-down

Super Admin Flow
https://www.figma.com/proto/B15Jf2uQhqqufVwEgbfrFv/Neo-Party-Hall-Super-Admin-Flow?node-id=1%3A2&viewport=418%2C363%2C0.03856455534696579&scaling=scale-down