

## **Neo Rooms - Hotel Booking Application**

### **Objective:**

Neo Rooms is an application to be built as a product that can help customers to choose a hotel and book a room.

### **Users of the System:**

1. Super Admin
2. Admin
3. Customer

### **Functional Requirements:**

- Build an application that customers can book rooms in hotels.
- The application should have signup, login, profile, dashboard page for user and login,signup,profile , dashboard , add room page for the admin.
- This application should have a provision to maintain a database for customer information,Admin information,booking information and room 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 slot.**

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 hotels and rooms based on the customer's price range, specific hotel 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:**

<b>Security</b>	<ul style="list-style-type: none"><li>● App Platform –UserName/Password-Based Credentials</li><li>● Sensitive data has to be categorized and stored in a secure manner</li><li>● Secure connection for transmission of any data</li></ul>
<b>Performance</b>	<ul style="list-style-type: none"><li>● Peak Load Performance (during Festival days, National holidays etc)</li><li>● eCommerce -&lt; 3 Sec</li></ul>

	<ul style="list-style-type: none"> <li>• Admin application &lt; 2 Sec</li> <li>• Non Peak Load Performance</li> <li>• eCommerce &lt; 2 Sec</li> <li>• Admin Application &lt; 2 Sec</li> </ul>
<b>Availability</b>	<ul style="list-style-type: none"> <li>• 99.99 % Availability</li> </ul>
<b>Standard Features</b>	<ul style="list-style-type: none"> <li>• Scalability</li> <li>• Maintainability</li> <li>• Usability</li> <li>• Availability</li> <li>• Failover</li> </ul>
<b>Logging &amp; Auditing</b>	<ul style="list-style-type: none"> <li>• The system should support logging(app/web/DB) &amp; auditing at all levels</li> </ul>
<b>Monitoring</b>	<ul style="list-style-type: none"> <li>• Should be able to monitor via as-is enterprise monitoring tools</li> </ul>
<b>Cloud</b>	<ul style="list-style-type: none"> <li>• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure</li> </ul>
<b>Browser Compatible</b>	<ul style="list-style-type: none"> <li>• IE 7+</li> <li>• Mozilla Firefox Latest – 15</li> <li>• Google Chrome Latest – 20</li> <li>• Mobile Ready</li> </ul>

## 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 Rooms in the hotel

Admin Add Room	/admin/addRoom	POST-Sends Room Data	Room Added
Admin Edit Room	/admin/editRoom	POST-Sends Room Data	Room Edited
Admin Delete Room	/admin/deleteRoom	POST-Sends Room ID	Room 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 Hotels available
Displaying Hotel Rooms	/user/rooms	POST - Sends Hotel name and admin ID of that hotel	Return all the rooms of the hotel
Room Details	/user/roomDetails	POST-Sends Room ID	Return room details
Booked Rooms	/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 an 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. API endpoint Url: <http://localhost:8080/signup>
  - c. Output screenshot:

The image shows a 'SIGN UP' form with the following elements and their corresponding IDs:

- id = "signupBox"**: Points to the entire sign-up container.
- id = "email"**: Points to the 'Enter Email' input field.
- id = "password"**: Points to the 'Enter Password' input field.
- id = "mobilenumber"**: Points to the 'Enter Mobile Number' input field.
- id = "userrole"**: Points to the 'User' dropdown menu.
- id = "username"**: Points to the 'Enter Username' input field.
- id = "age"**: Points to the 'Enter Age' input field.
- id = "submitButton"**: Points to the red 'Submit' button.
- id = "loginLink"**: Points to the 'Click Here' link in the 'Go to Login' text.

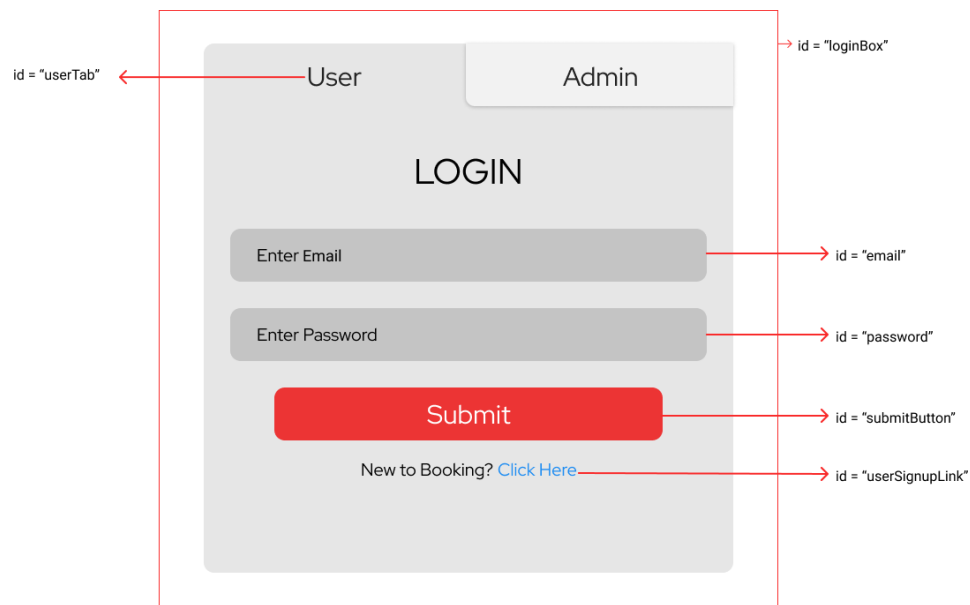
2. Login: Design a login page component where the existing customer can log in using the registered email id and password.

a. Ids:

- i. loginBox
- ii. userTab
- iii. email
- iv. password
- v. submitButton
- vi. userSignupLink

b. API endpoint Url: <http://localhost:8080/login>

c. Output screenshot:



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

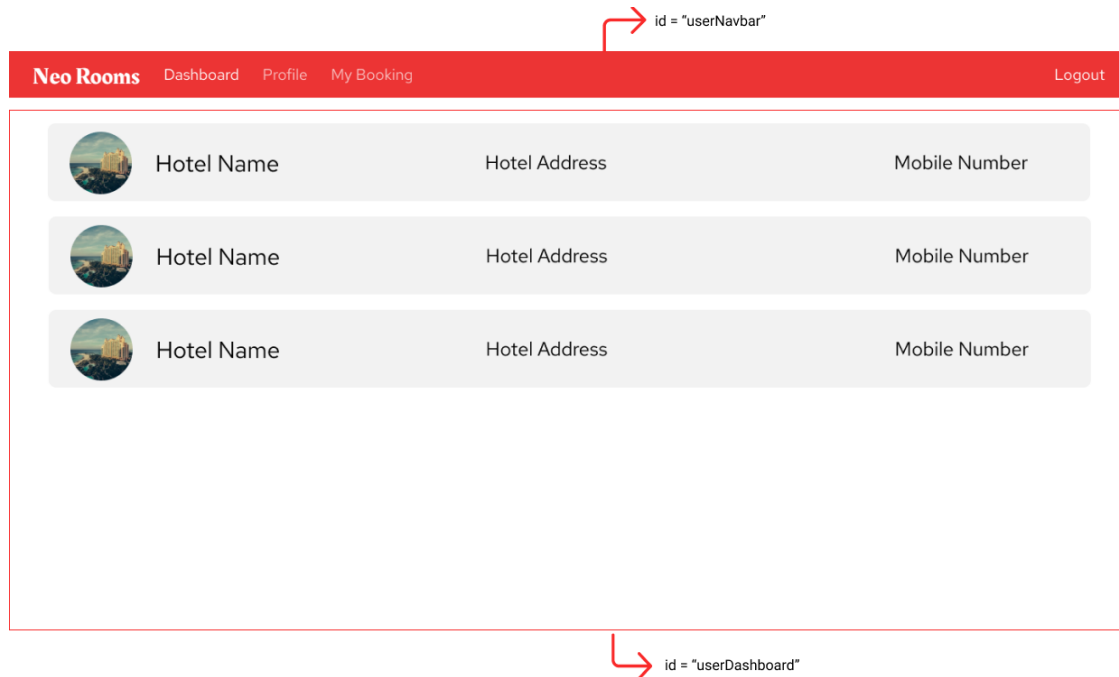
a. Ids:

i. userNavbar

ii. userDashboard

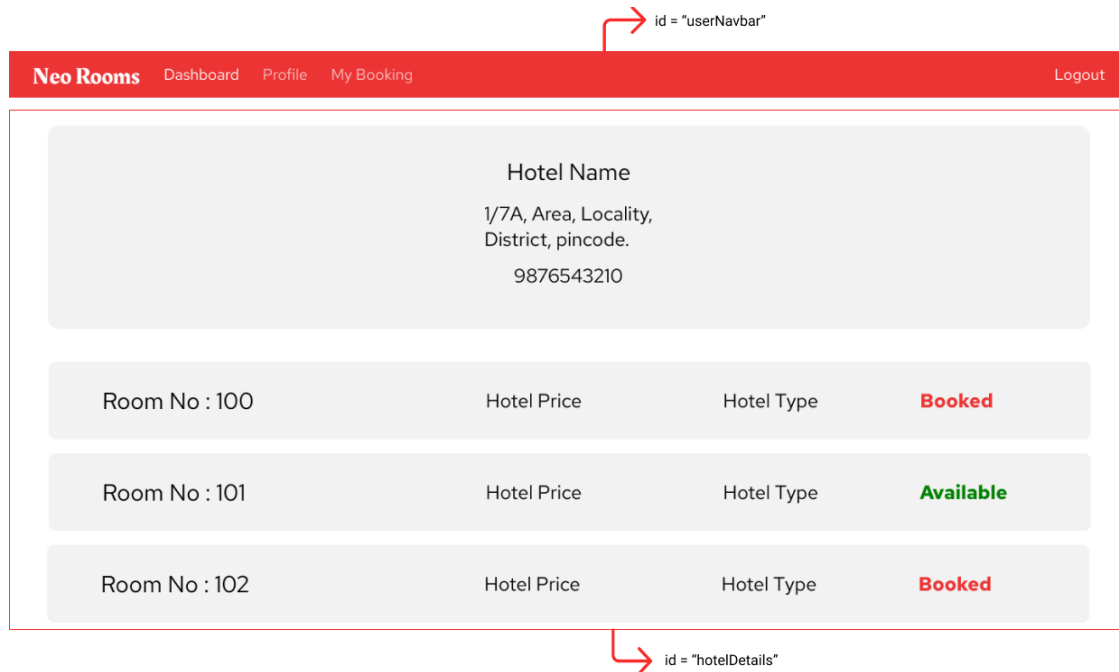
b. API endpoint Url: <http://localhost:8080/home>

c. Screenshot

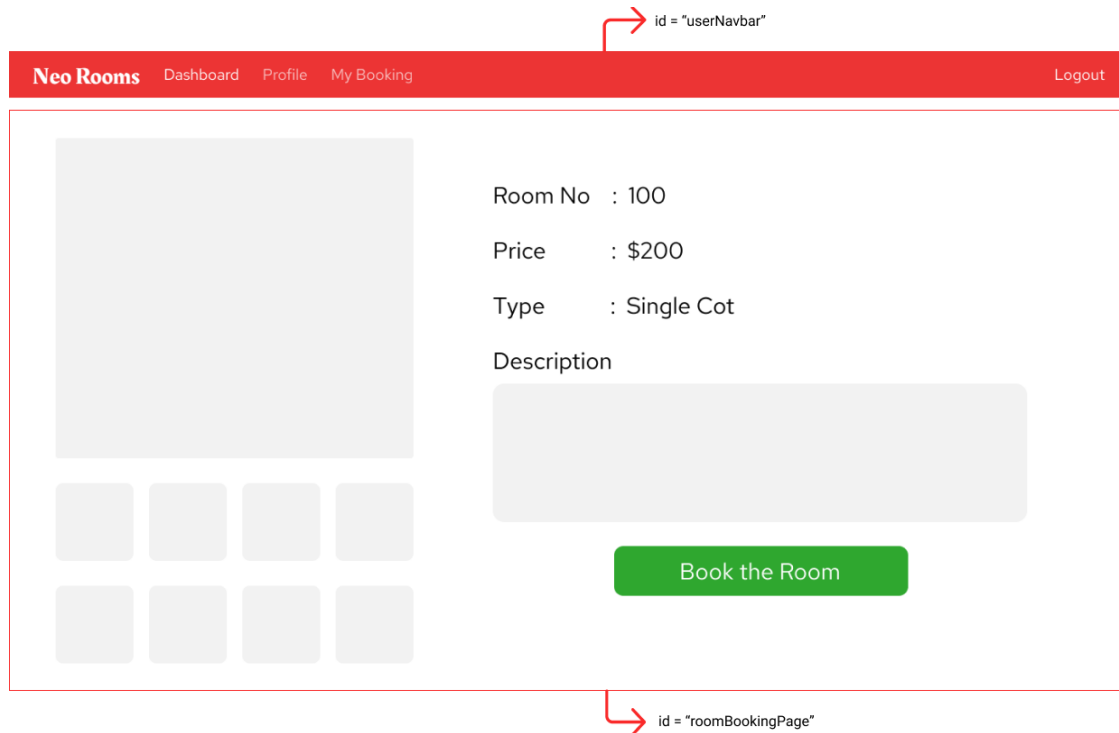


4. Hotel Details : Design a Hotel Details component which lists the total number of rooms available along with the individual status of each room ( available / Booked ).
  - a. Ids
    - i. userNavbar
    - ii. hotelDetails
  - b. API endpoint URL : <http://localhost:8080/hotelDetail/{id}>
  - c. Screenshot





5. Bookings : Design a Booking Page that displays the details of the room that the user is currently booking.
  - a. Ids
    - i. userNavbar
    - ii. roomBookingPage
  - b. API endpoint URL : <http://localhost:8080/booking/{id}>
  - c. Screenshot

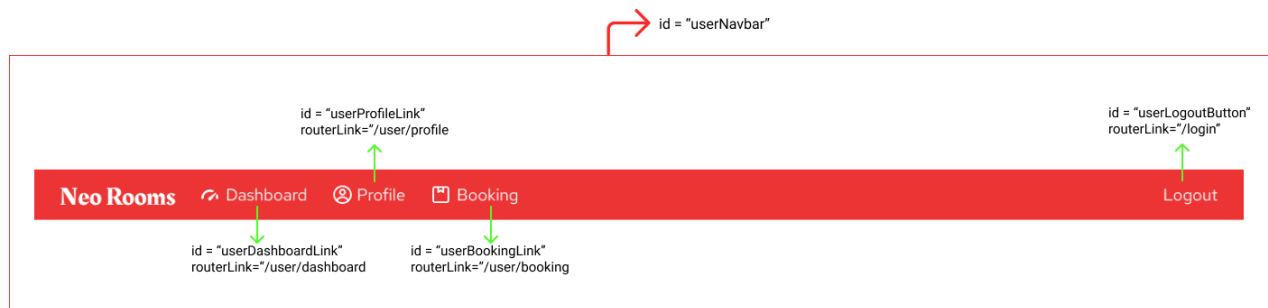


6. User Navigation : Design a Booking page component where the user can navigate to their profile and bookings page.

a. Ids

- i. userNavbar
- ii. userProfileLink
- iii. userDashboardLink
- iv. userBookingLink
- v. userLogoutButton

b. Screenshot



7. User Profile Edit : Design an 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. mobilenummer
- viii. editProfileButton

b. API endpoint URL : <http://localhost:8080/editProfile/{id}>

c. Screenshot:

**Neo Rooms** [Dashboard](#) [Profile](#) [My Booking](#) [Logout](#)

< back

Name

Jennifer Aniston

Email

rachelgreen@iamneo.ai

Password

JoeyTribbiani

Age

27

Mobile Number

1234567890

Save Changes

id = "editProfileBox"

id = "username"

id = "email"

id = "password"

id = "userAge"

id = "mobilenumber"

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. API endpoint URL : <http://localhost:8080/bookings/{id}>

c. Screenshot:

Hotel Name	Room No	Price	Quantity	Total Price
Hotel Name	Room No	Price	Quantity	Total Price
Hotel Name	Room No	Price	Quantity	Total Price
Hotel Name	Room No	Price	Quantity	Total Price
Hotel Name	Room No	Price	Quantity	Total Price
Hotel Name	Room No	Price	Quantity	Total Price

## **Admin:**

9. 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. mobilenummer
- v. userrole
- vi. adminname
- vii. hotelname
- viii. hotelimageURL
- ix. hotelAddress
- x. submitButton
- xi. adminLoginLink

b. API endpoint Url : <http://localhost:8080/admin/signup>

c. Screenshot

The screenshot shows a 'SIGN UP' form with the following elements and their IDs:

- Form container: `id = "signupBox"`
- Email input: `id = "email"`
- Password input: `id = "password"`
- Mobile Number input: `id = "mobilenumber"`
- User Role dropdown: `id = "userrole"`
- Seller Name input: `id = "adminname"`
- Hotel Name input: `id = "hotelName"`
- Hotel Image URL input: `id = "hotelimageURL"`
- Hotel Address input: `id = "hotelAddress"`
- Submit button: `id = "submitButton"`
- Go to Login link: `id = "adminLoginLink"`

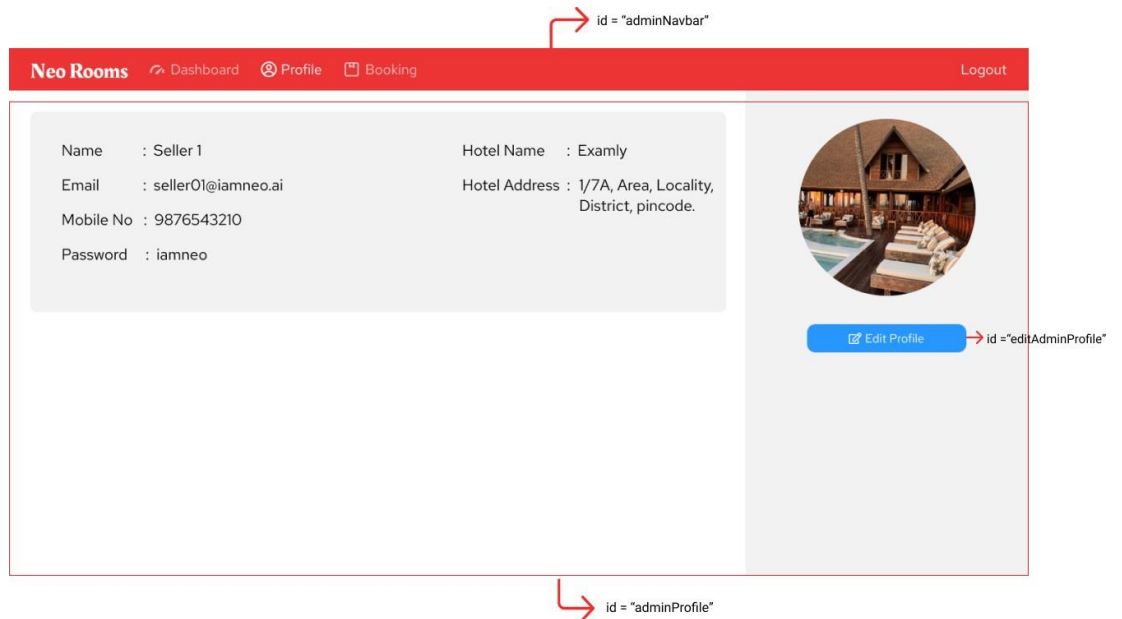
10. Admin Profile: Design a Admin Profile page that displays the details of the admin currently logged in.

a. Ids

- i. `adminNavbar`
- ii. `adminProfile`
- iii. `editAdminProfile`

b. API endpoint URL: <http://localhost:8080/admin/profile>

c. Screenshots



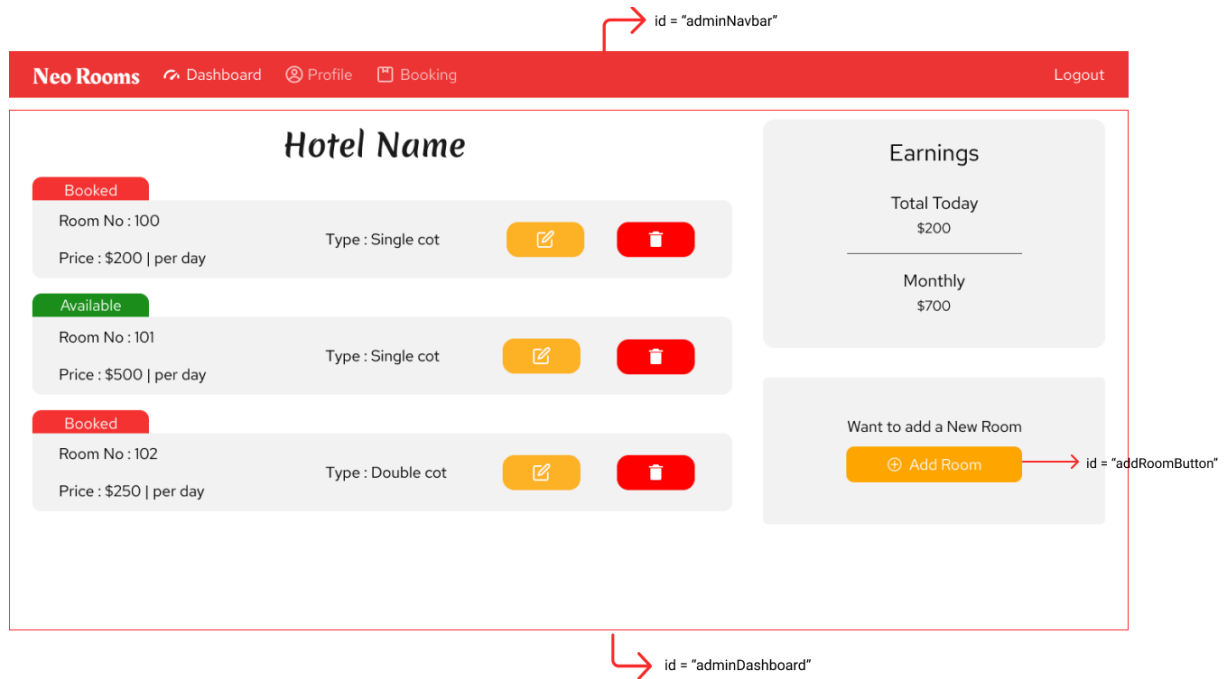
11. Admin Dashboard: Design a dashboard page where the list of products is displayed on the admin side.

a. Ids

- i. adminNavbar
- ii. addRoomButton
- iii. adminDashboard

b. API endpoint Url: <http://localhost:8080/admin/dashboard>

c. Screenshot



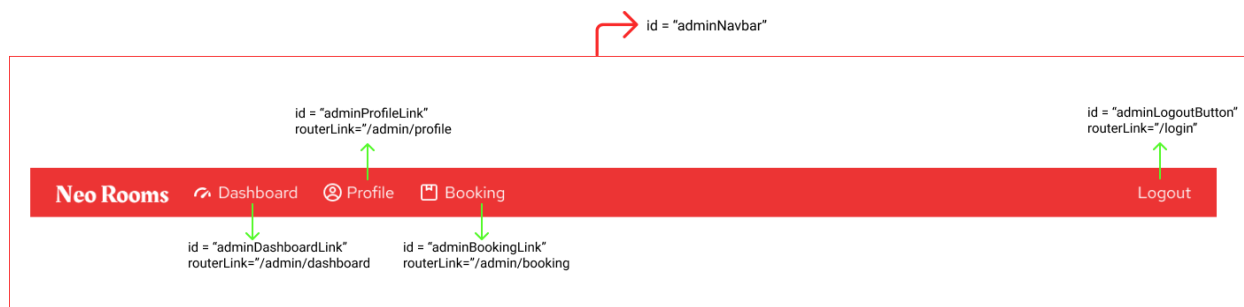
12. Admin Navigation: Design an Admin navigation component that can navigate to the Dashboard, profile and bookings done by the users

a. Ids:

- i. adminNavbar
- ii. adminProfileLink
- iii. adminDashboardLink
- iv. adminBookingLink
- v. adminLogoutButton

b. Screenshot:





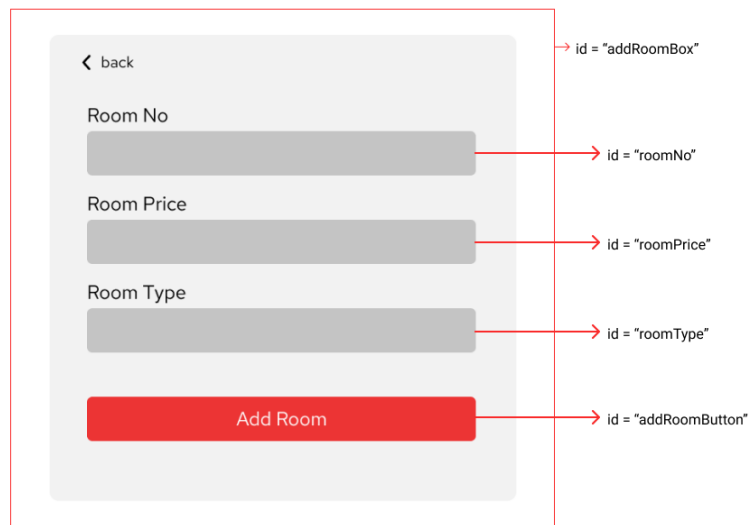
13. Admin Add Room: Design an Add Room component in which the admin can add new rooms to the inventory.

a. Ids:

- i. adminNavbar
- ii. addRoomBox
- iii. roomNo
- iv. roomPrice
- v. roomType
- vi. addRoomButton

b. API endpoint Url: <http://localhost:8080/admin/addRoom>

c. Screenshot



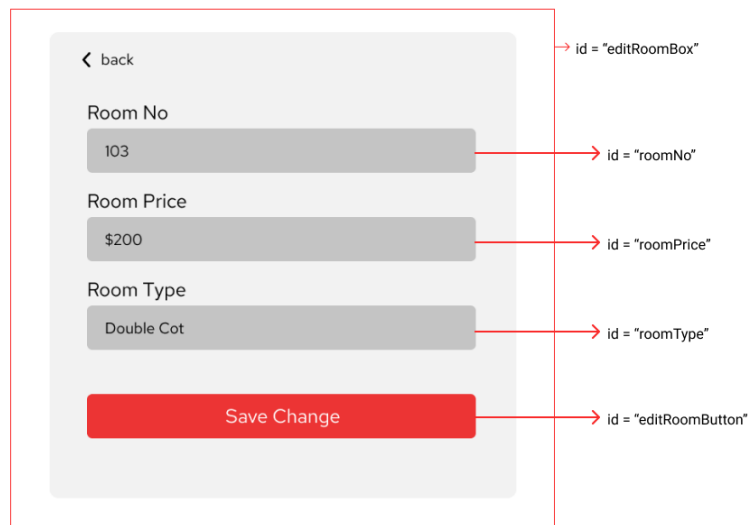
14. Admin Edit Room: Design an Edit Room component in which the admin can edit details of an already existing room in the inventory.

a. Ids:

- i. adminNavbar
- ii. editRoomBox
- iii. roomNo
- iv. roomPrice
- v. roomType
- vi. editRoomButton

b. API endpoint Url: <http://localhost:8080/admin/editRoom>

c. Screenshot



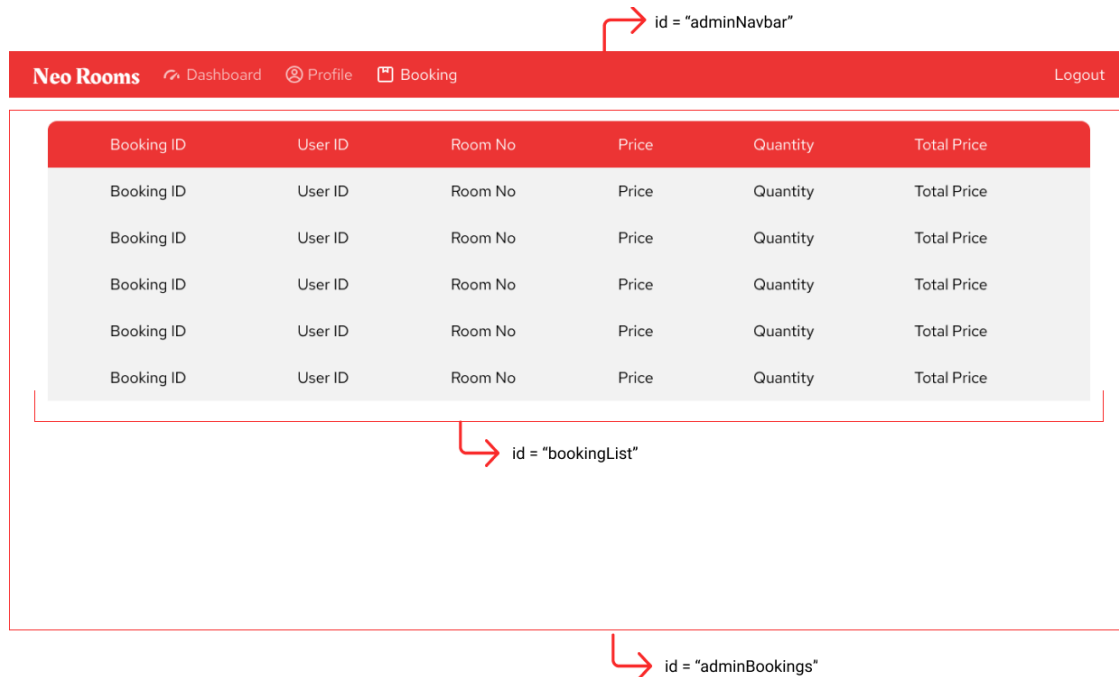
15. Admin Bookings : Design a Admin Bookings component where the admin can see the booking details that all the users have made till present date.

a. Ids

- i. adminNavbar
- ii. adminBookings
- iii. bookingList

b. API endpoint URL : <http://localhost:8080/admin/bookings>

c. Screenshot:



The screenshot shows the 'Neo Rooms' admin dashboard. The top navigation bar is red and contains the 'Neo Rooms' logo, a 'Dashboard' link with a home icon, a 'Profile' link with a user icon, a 'Booking' link with a calendar icon, and a 'Logout' link. Below the navbar is a table with 6 columns: 'Booking ID', 'User ID', 'Room No', 'Price', 'Quantity', and 'Total Price'. The table has 5 data rows, each containing placeholder text. Red arrows indicate the following IDs: 'id = "adminNavbar"' points to the navbar, 'id = "bookingList"' points to the table, and 'id = "adminBookings"' points to the container surrounding the table.

Booking ID	User ID	Room No	Price	Quantity	Total Price
Booking ID	User ID	Room No	Price	Quantity	Total Price
Booking ID	User ID	Room No	Price	Quantity	Total Price
Booking ID	User ID	Room No	Price	Quantity	Total Price
Booking ID	User ID	Room No	Price	Quantity	Total Price
Booking ID	User ID	Room No	Price	Quantity	Total Price

16. 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. hotelName
- vii. hotelAddress
- viii. profileEditButton

b. API endpoint URL : <http://localhost:8080/admin/editProfile>

c. Screenshot:

Neo Rooms [Dashboard](#) [Profile](#) [Booking](#) [Logout](#)

< back

Name  
Seller 1

Email  
seller01@iamneo.ai

Mobile Number  
9876543210

Password  
iamneo

Hotel Name  
Examly

Hotel Address  
1/7A, Area, Locality, District, pincode

Save Changes

→ id = "adminEditBox"

→ id = "adminName"

→ id = "adminEmail"

→ id = "adminMobilenumber"

→ id = "adminPassword"

→ id = "hotelName"

→ id = "hotelAddress"

→ id = "profileEditButton"

### **Super Admin:**

17. 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. API endpoint URL : <http://localhost:8080/superadmin/login>

c. Screenshot:

id = "superAdminLoginBox"

Super Admin

Enter Super Admin Email

id = "email"

Enter Password

id = "password"

Submit

id = "submitButton"

18. Super Admin Users page : Design a Super-Admin users page where the super admin has authority to view and delete any user currently registered into the system.

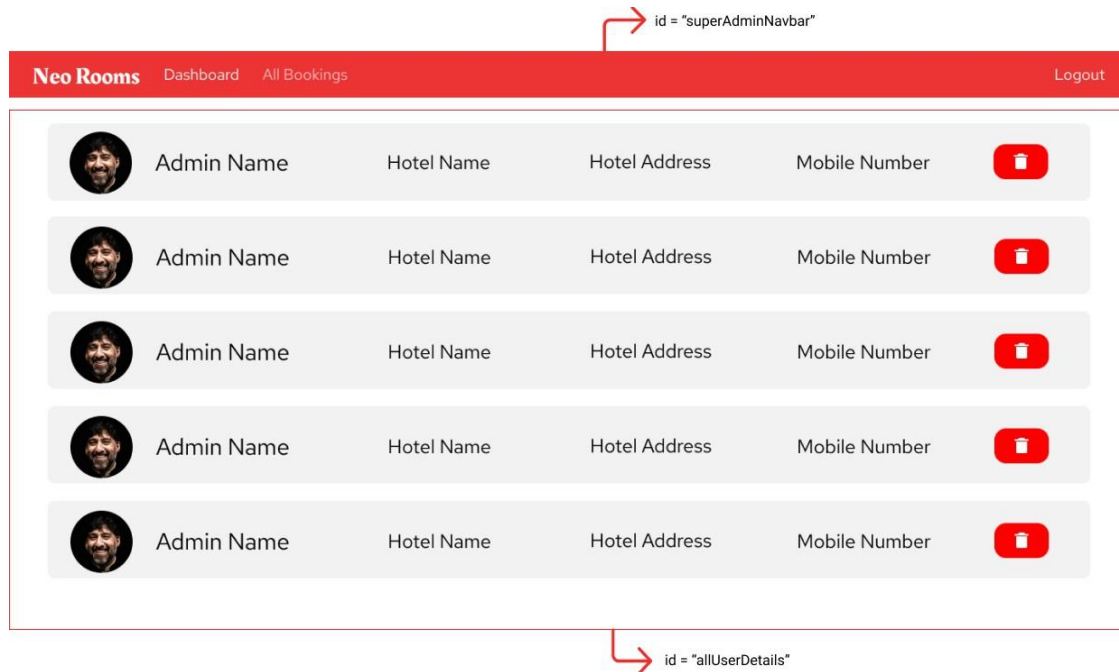
a. Ids:

i. superAdminNavbar

ii. allUserDetails

b. API endpoint URL : <http://localhost:8080/superadmin/users>

c. Screenshot



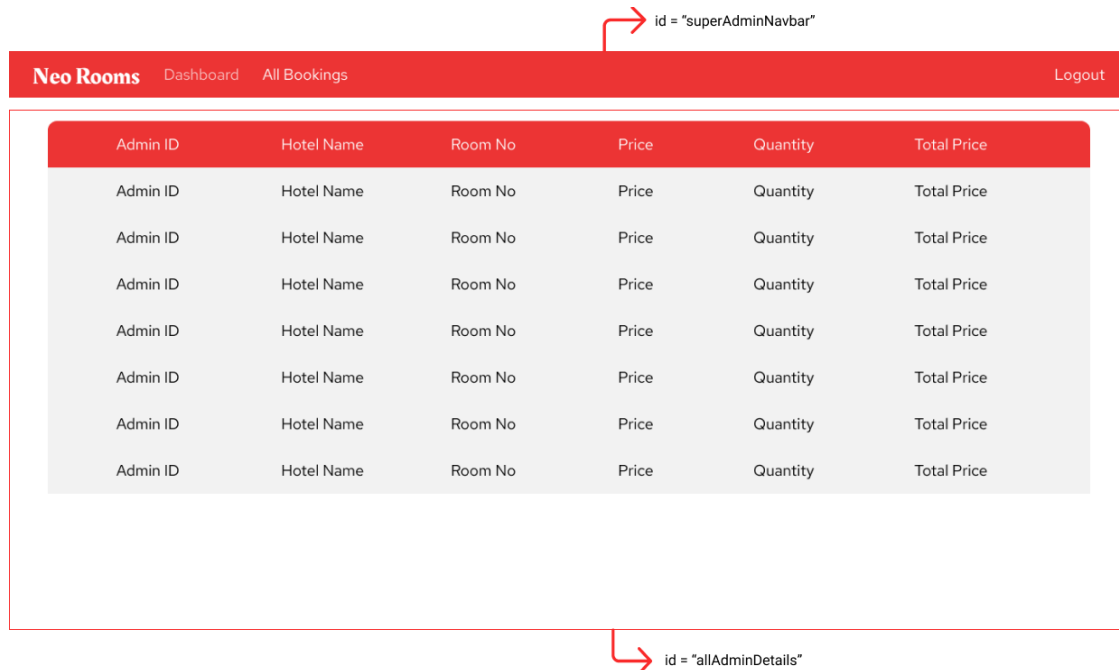
19. Super Admin Bookings page : Design a Super-Admin Booking component where the super admin has authority to view all the bookings.

a. Ids:

- i. superAdminNavbar
- ii. allAdminDetails

b. API endpoint URL : <http://localhost:8080/superadmin/admin>

c. Screenshot



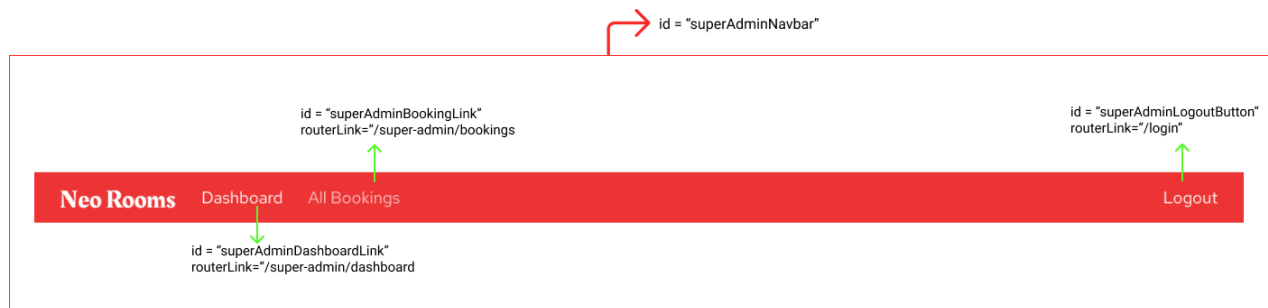
20. Super Admin Navigation : Design a Navigation page for the super user where he can navigate to the dashboard, all the bookings and logout.

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. hotelName:String
- vii. hotelImageURL:String
- viii. hotelAddress: String
- ix. earnings:int

4. RoomModel: This class stores the cart items.

a. Attributes:

- i. roomID:String
- ii. roomNo: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. RoomController: This class controls the save/edit/delete/view rooms.

a. Methods:

- i. saveRoom(RoomModel data): This method helps the admin to save new rooms in the database.
- ii. editRoom(RoomModel data): This method helps the admin to edit the details of the rooms and save it again in the database.
- iii. deleteRoom(String Room ID): This method helps the admin to delete a room from the hotel and as well as in the database.
- iv. getRooms(String Admin\_Email\_ID): This method helps the admin to get all the rooms in the hotel.
- v. bookRoom(String Room\_ID): This method helps the user to book the room by changing the room status and add the record in the bookings table.

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

a. Methods:

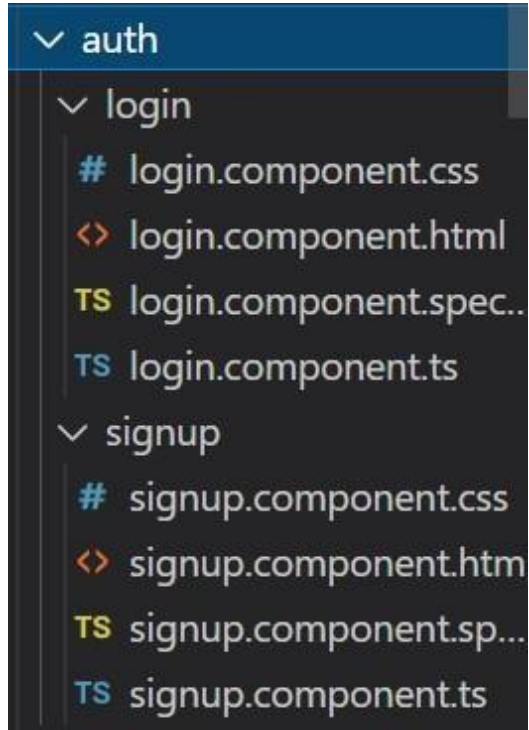
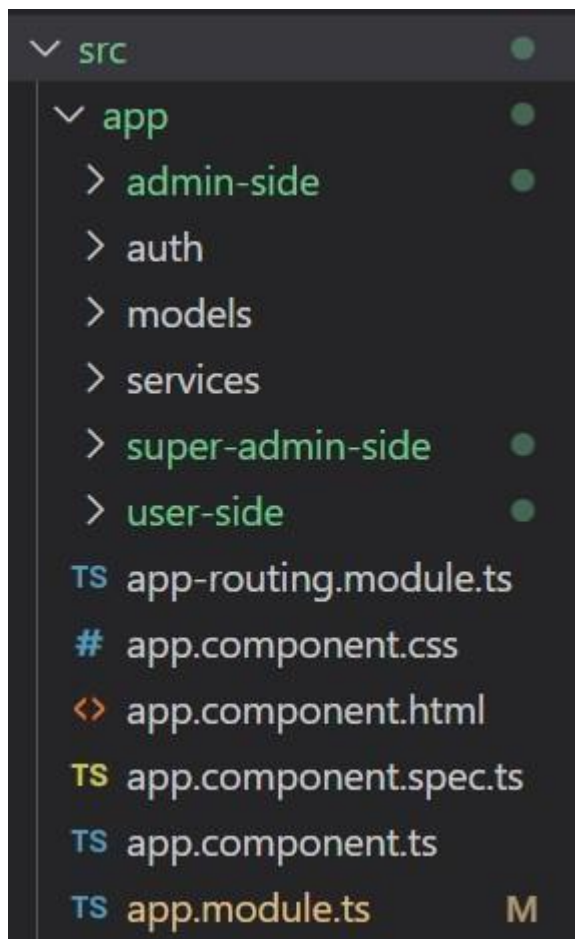
- i. editAdmin(AdminModel data): This method helps the admin to edit the profile details and save it in the database.
- ii. getProfile(String Admin\_Email\_ID): This method helps admin to get the profile details.

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

a. Methods:

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

## **Angular Folder Structure:**



```

  ✓ 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/Hq13FkhAJskiFvizpXbNiE/Neo-Rooms-Admin-Flow?node-id=30%3A53&viewport=700%2C675%2C0.08671894669532776&scaling=scale-down>

### **User Flow**

<https://www.figma.com/proto/PsLcbvAahDsWBCUJcrM1pW/Neo-Rooms-User-Flow?node-id=2%3A12&viewport=153%2C142%2C0.043812938034534454&scaling=scale-down>

### **Super Admin Flow**

<https://www.figma.com/proto/aLvkpVlp2he09tNwQxj5FR/Neo-Rooms-SuperAdmin-Flow?node-id=1%3A2&viewport=467%2C246%2C0.10300646722316742&scaling=scale-down>