

Neo Bike Rental Booking Application

Objective:

Neo Bike Rental is an application to be built as a product that can help customers to choose a Company and book a bike.

Users of the System:

1. Super Admin
2. Admin
3. Customer

Functional Requirements:

- Build an application that customers can book Bikes.
- The application should have signup, login, profile, dashboard page for user and login, signup, profile, dashboard, add bike page for the admin.
- This application should have a provision to maintain a database for customer information, Admin information, Booking information and bike 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 booking per bike.**

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 bikes based on the customer's price range, specific model 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 | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• Peak Load Performance (during Festival days, National holidays etc)• eCommerce -< 3 Sec |

| | |
|-------------------------------|---|
| | <ul style="list-style-type: none"> • Admin application < 2 Sec • Non Peak Load Performance • eCommerce < 2 Sec • Admin Application < 2 Sec |
| Availability | <ul style="list-style-type: none"> • 99.99 % Availability |
| Standard Features | <ul style="list-style-type: none"> • Scalability • Maintainability • Usability • Availability • Failover |
| Logging & Auditing | <ul style="list-style-type: none"> • The system should support logging(app/web/DB) & auditing at all levels |
| Monitoring | <ul style="list-style-type: none"> • Should be able to monitor via as-is enterprise monitoring tools |
| Cloud | <ul style="list-style-type: none"> • The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure |
| Browser Compatible | <ul style="list-style-type: none"> • 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:4200/signup> or <http://localhost:4200/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 Bikes in the Company |

| | | | |
|-------------------|--------------------|-----------------------------|------------------------------|
| Admin Add Bike | /admin/addBike | POST-Sends Bike Data | Bike Added |
| Admin Edit Bike | /admin/editBike | POST-Sends Bike Data | Bike Edited |
| Admin Delete Bike | /admin/deleteBike | POST-Sends Bike ID | Bike 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 Bikes available |
| Displaying Bikes | /user/bikes | POST - Sends Bike name and admin ID of that bike | Return all the bikes of the company |
| Bike Details | /user/bikeDetails | POST-Sends Bike ID | Return Bike details |
| Booked Bikes | /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. mobilenummer
 - v. userrole
 - vi. username
 - vii. age
 - viii. submitButton
 - ix. loginLink
 - b. Routing URL: <http://localhost:4200/user/signup>
 - c. Output screenshot:

Diagram illustrating the structure of a SIGN UP form component with associated IDs:

- Form Container: id = "signupBox"
- Input Fields:
 - Enter Email: id = "email"
 - Enter Password: id = "password"
 - Enter Mobile Number: id = "mobilenumber"
 - User (Dropdown): id = "userrole"
 - Enter Username: id = "username"
 - Enter Age: id = "age"
- Submit Button: id = "submitButton"
- Login Link: id = "loginLink"

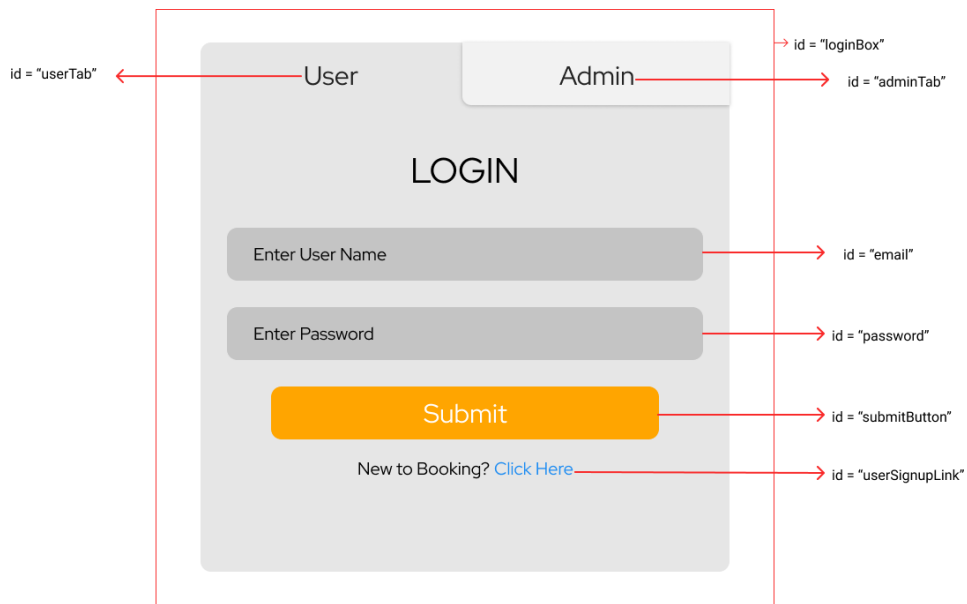
2. Login: Design a login page component where the existing customer can log in using the registered email id and password.

a. Ids:

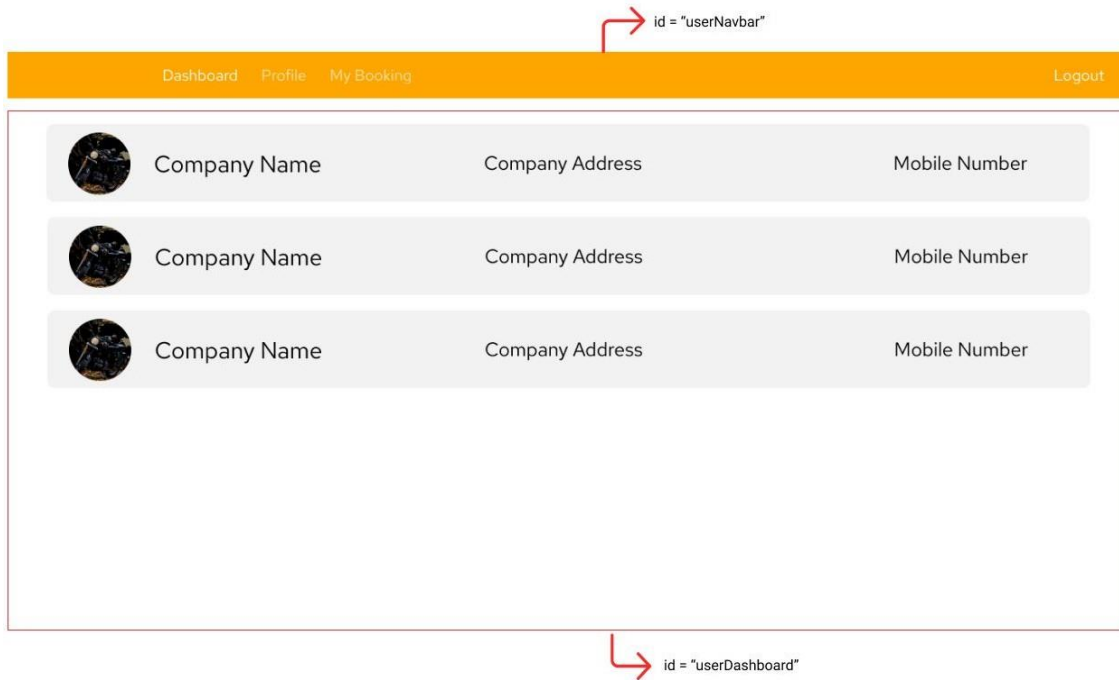
- i. loginBox
- ii. userTab
- iii. email
- iv. password
- v. submitButton
- vi. userSignupLink

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

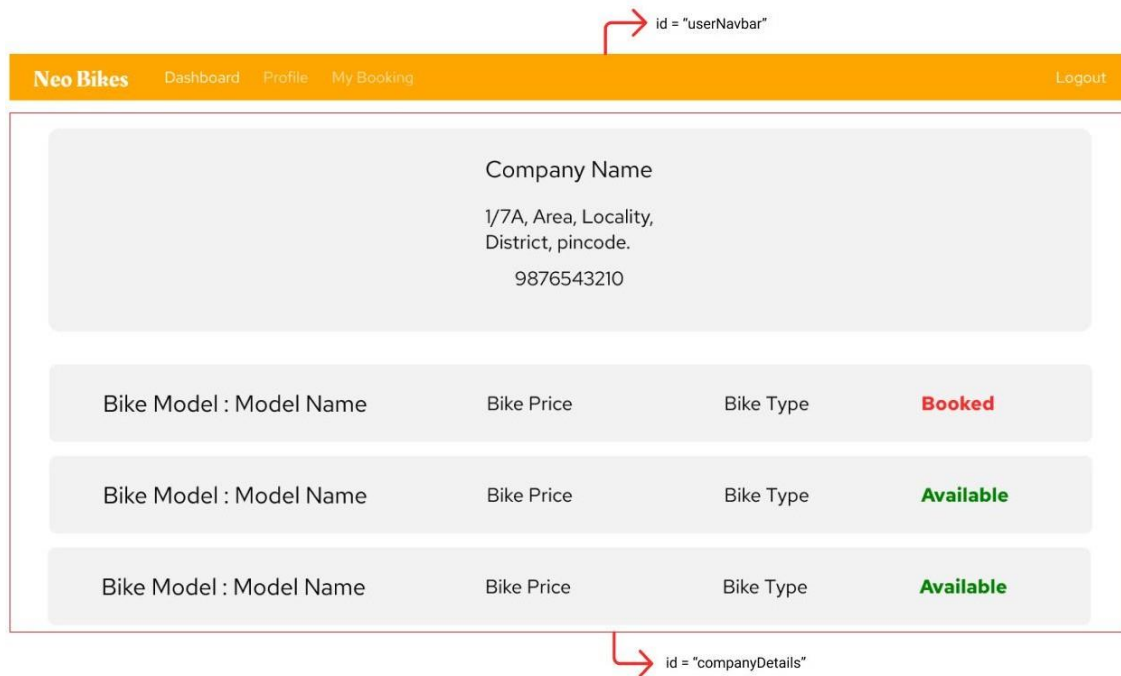
c. Output screenshot:



3. Dashboard / Home: Design a dashboard page component which provides a list of companies available where the user can book a particular bike.
 - a. Ids:
 - i. userNavbar
 - ii. userDashboard
 - b. Routing URL: <http://localhost:4200/user/dashboard>
 - c. Screenshot



4. Company Details: Design a Company Details component which lists the total number of bikes available along with the individual status of each bike (available / Booked).
 - a. Ids
 - i. userNavbar
 - ii. companyDetails
 - b. Routing URL : <http://localhost:4200/user/companyDetail/{companyId}>
 - c. Screenshot



5. Bike Details : Design a Bike Booking Page that displays the details of the bike that the user is currently booking.

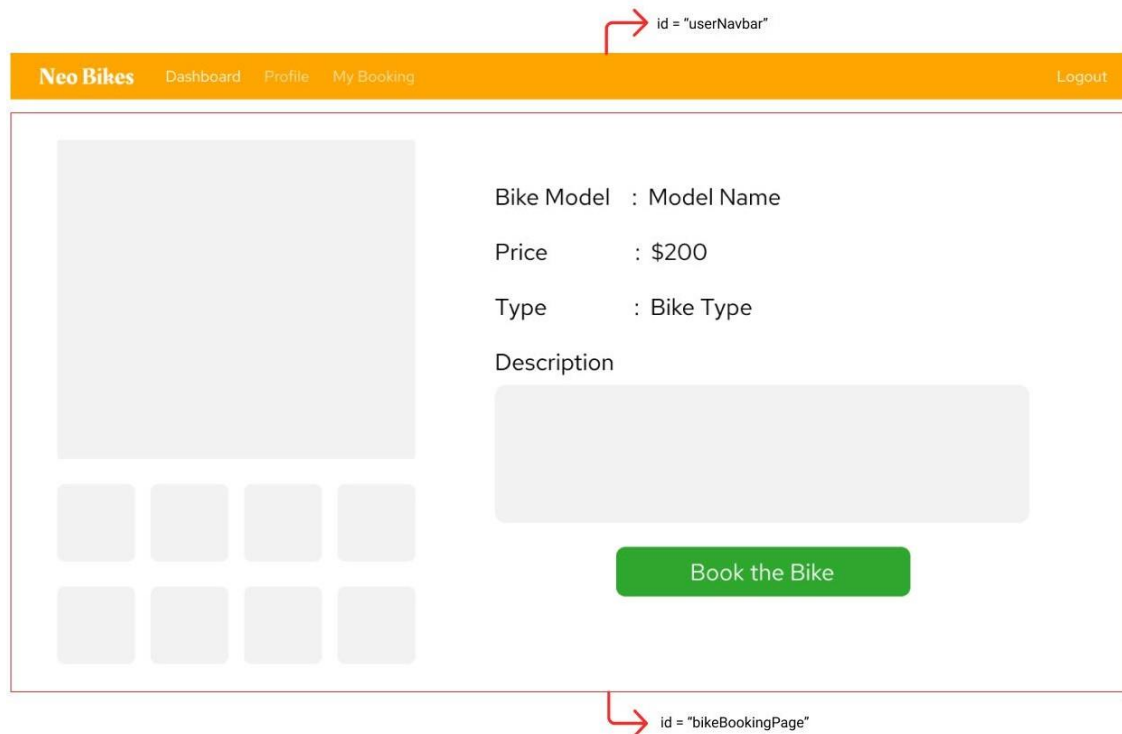
a. Ids

i. userNavbar

ii. bikeBookingPage

b. Routing URL : <http://localhost:4200/user/bikeDetail/{bikeId}>

c. Screenshot

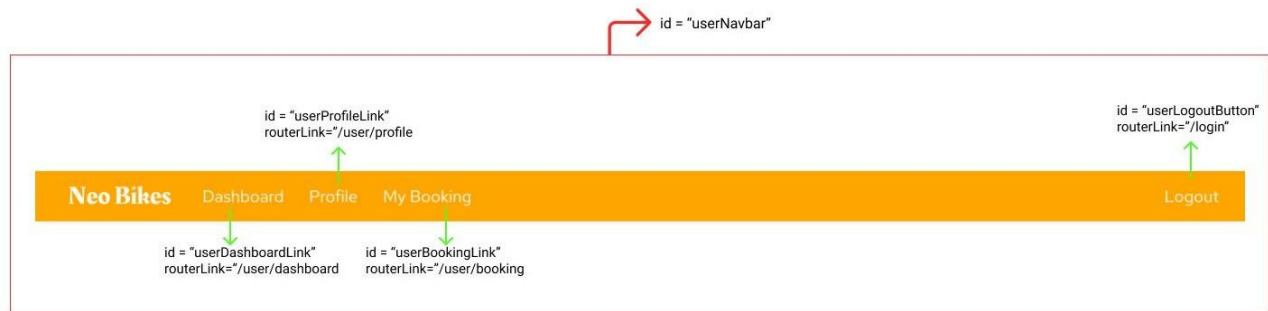


6. User Navigation: Design a Navbar component where the user can navigate to their profile and bookings page.

a. Ids

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

b. Screenshot



7. User Profile Edit: Design 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:

→ 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 Bikes Dashboard Profile My Booking Logout | | | | |
|--|------------|------|------|-------------|
| Company Name | Bike Model | Rent | Days | Total Price |
| Company Name | Model Name | Rent | Days | Total Price |
| Company Name | Model Name | Rent | Days | Total Price |
| Company Name | Model Name | Rent | Days | Total Price |
| Company Name | Model Name | Rent | Days | Total Price |
| Company Name | Model Name | Rent | Days | Total Price |

id = "userBookingBody"

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

a. Ids:


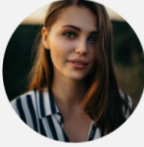
i. userNavbar

ii. userProfileBody

b. Routing URL: <http://localhost:4200/user/profile/{userId}>

c. Screenshot:

id = "userNavbar"

| Neo Bikes Dashboard Profile My Booking Logout | |
|---|--|
| <div> <div>Name : Jennifer Aniston</div> <div>Email : rachelgreen@iamneo.ai</div> <div>Password : ***** </div> <div>Age : 27</div> <div>Mobile Number : 1234567890</div> </div> <div>  <div>Edit Profile</div> </div> | |

id = "userProfileBody"

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. mobilenumbers
- v. userrole
- vi. adminname
- vii. companynames
- viii. companyimageURL
- ix. companyAddress
- x. submitButton
- xi. adminLoginLink

b. Routing URL: <http://localhost:4200/admin/signup>

c. Screenshot

The screenshot 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 'Admin' dropdown menu.
- id = "adminname"**: Points to the 'Enter Seller Name' input field.
- id = "companyname"**: Points to the 'Enter Company Name' input field.
- id = "companyimageURL"**: Points to the 'Enter Company Image Url' input field.
- id = "companyAddress"**: Points to the 'Enter Company Address' input field.
- id = "submitButton"**: Points to the orange 'Submit' button.
- id = "adminLoginLink"**: Points to the 'Go to Login [Click Here](#)' link.

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

a. Ids

i. loginBox

ii. userTab

iii. email

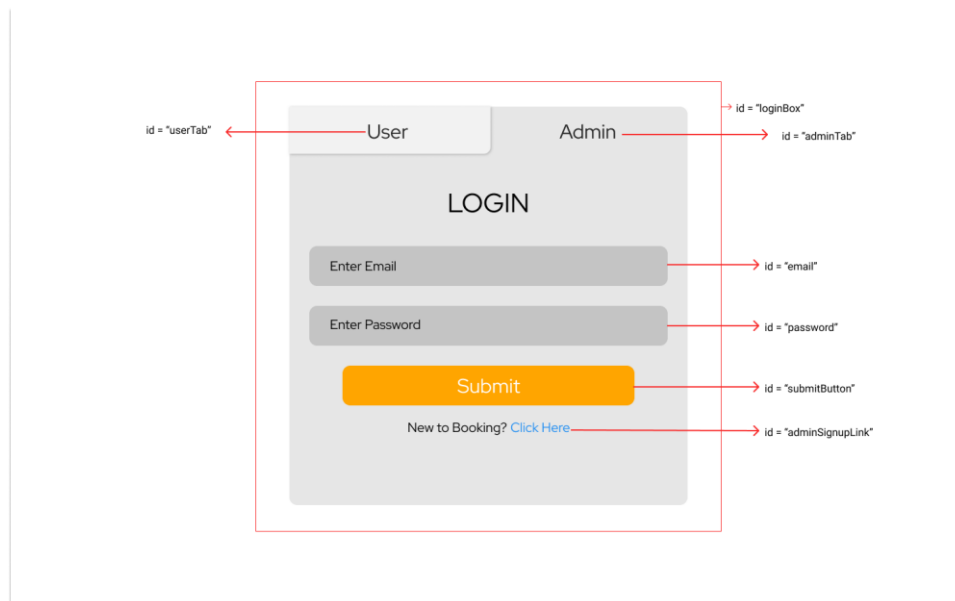
iv. password

v. submitButton

vi. adminSignupLink

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

c. Screenshots



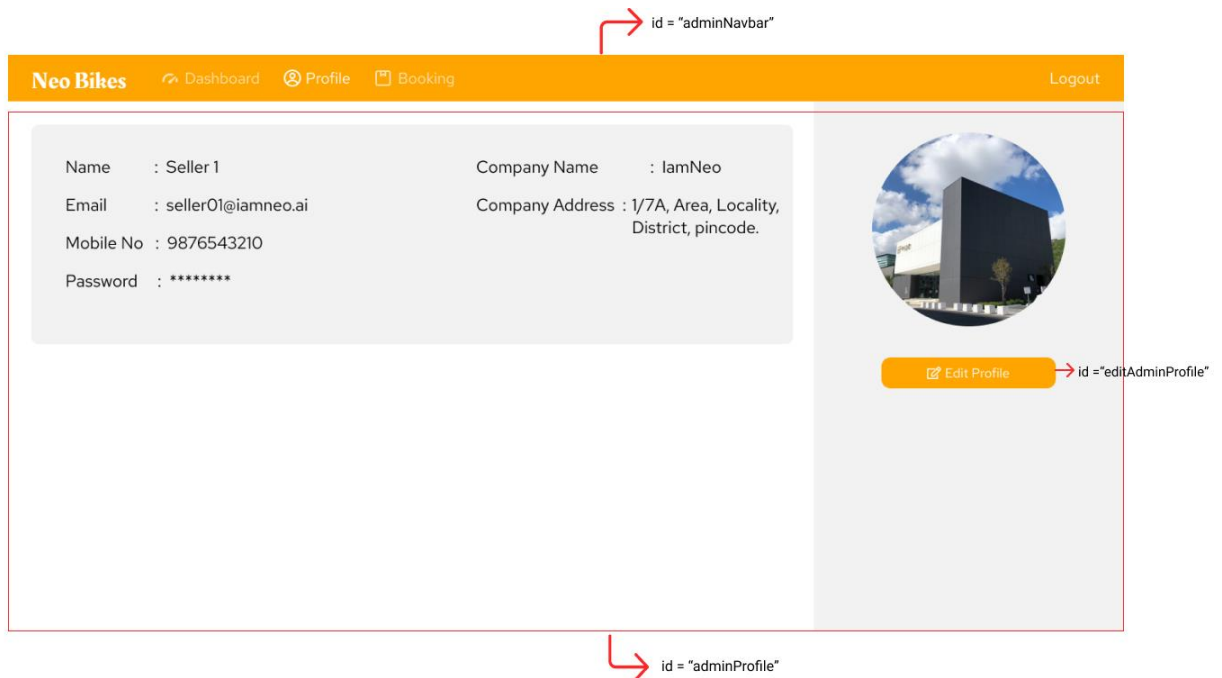
12. Admin Profile: Design an Admin Profile page that displays the details of the admin currently logged in.

a. Ids

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

b. Routing URL: <http://localhost:4200/admin/profile/{adminId}>

c. Screenshots



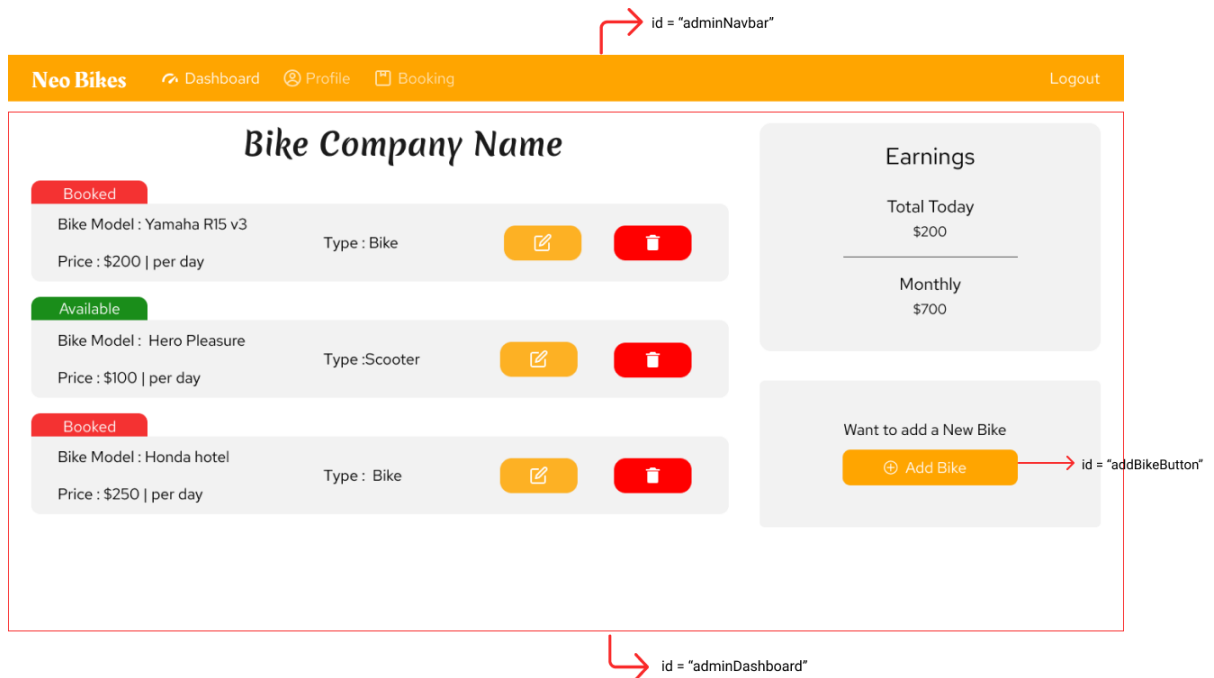
13. Admin Dashboard: Design a dashboard page where the list of products is displayed on the admin side.

a. Ids

- i. adminNavbar
- ii. addBikeButton
- iii. adminDashboard

b. Routing URL: <http://localhost:4200/admin/dashboard>

c. Screenshot

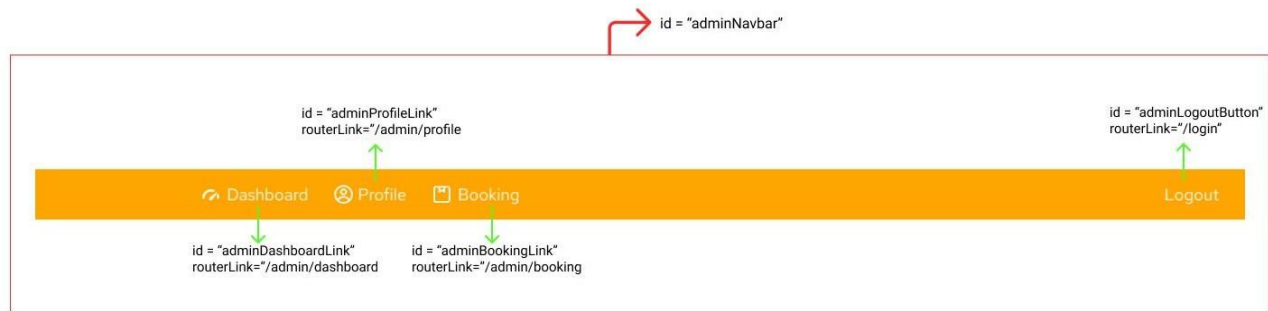


14. Admin Navigation: Design an Admin navigation component that can navigate to the Dashboard, profile and bookings done by the users

a. Ids:

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

b. Screenshot:



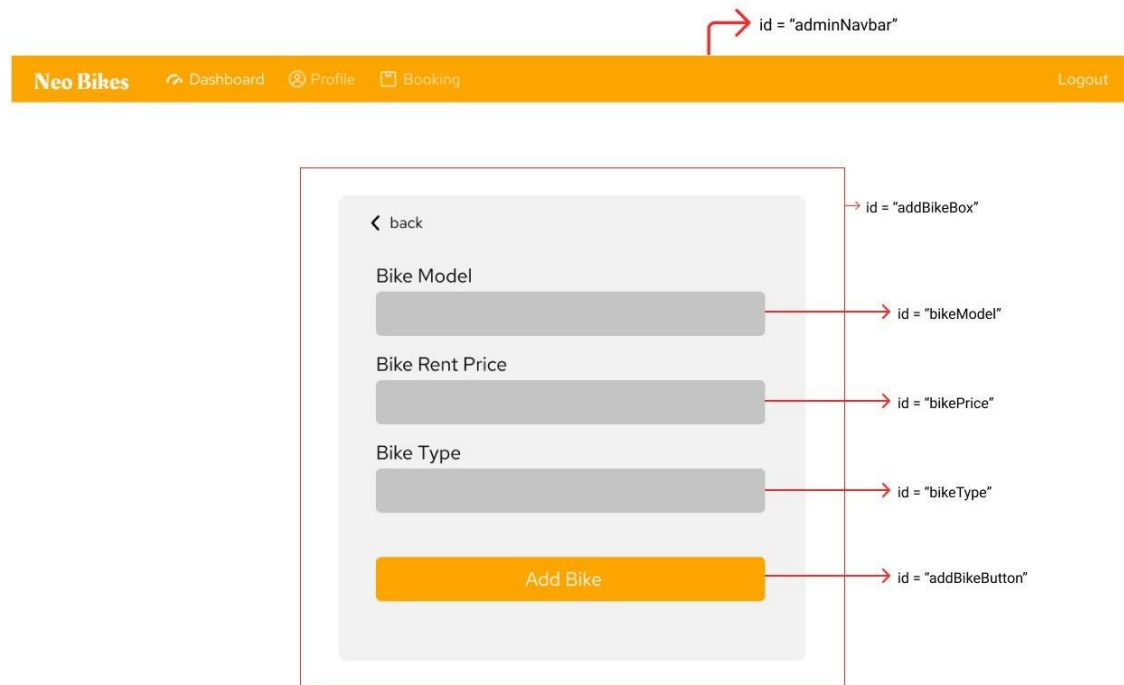
15. Admin Add Bike: Design an Add Bike component in which the admin can add new bikes to the inventory.

a. Ids:

- i. adminNavbar
- ii. addBikeBox
- iii. bikeNo
- iv. bikePrice
- v. bikeType
- vi. addBikeButton

b. Routing URL: <http://localhost:4200/admin/addBike>

c. Screenshot



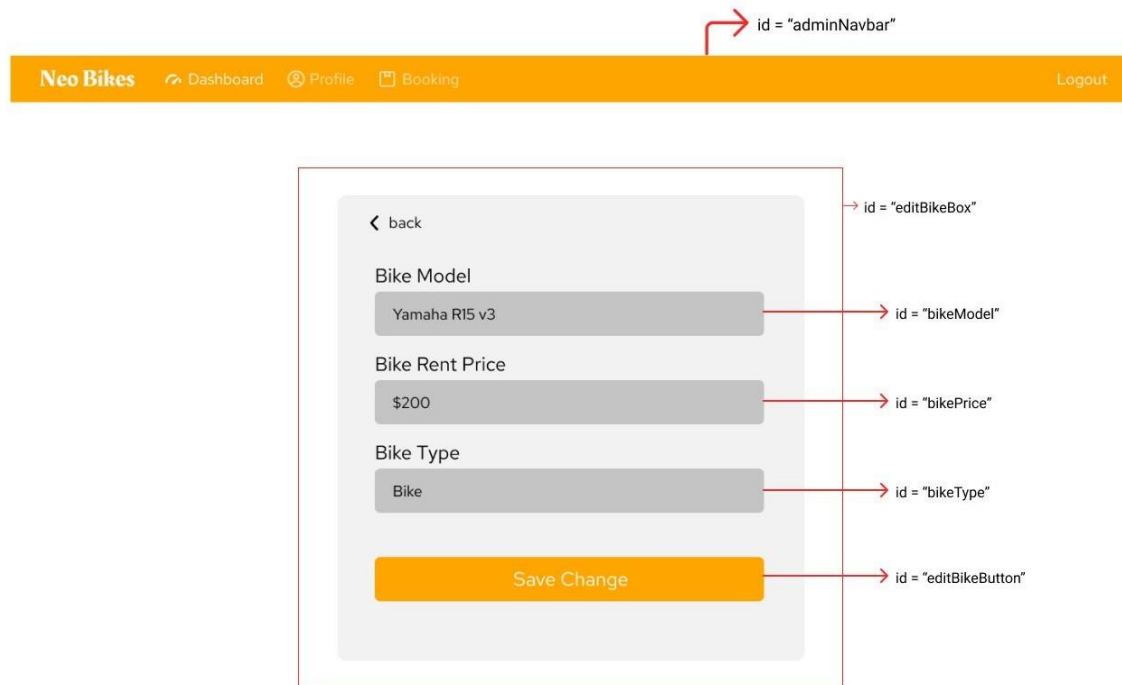
16. Admin Edit Bike: Design an Edit Bike component in which the admin can edit details of an already existing bike in the inventory.

a. Ids:

- i. adminNavbar
- ii. editBikeBox
- iii. bikeNo
- iv. bikePrice
- v. bikeType
- vi. editBikeButton

b. Routing URL: <http://localhost:4200/admin/editBike/{bikeId}>

c. Screenshot



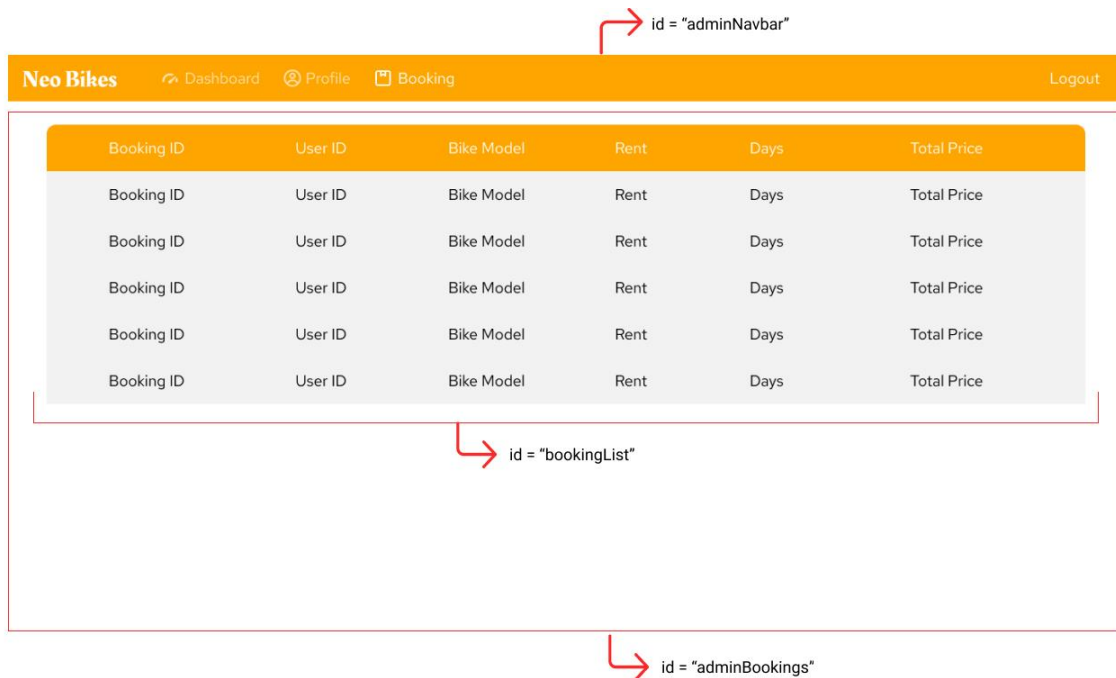
17. Admin Bookings: Design an Admin Bookings component where the admin can see the booking details that all the users have made till present date.

a. Ids

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

b. 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:

The screenshot shows the 'Neo Bikes' application interface with a top navigation bar containing 'Dashboard', 'Profile', 'Booking', and 'Logout'. The 'Profile' page is active, displaying a form for editing user information. The form includes a 'back' link, and fields for Name, Email, Mobile Number, Password, Company Name, and Company Address, each followed by a 'Save Changes' button. Red arrows point from text labels to specific form elements, indicating their IDs for testing purposes.

| Form Field | Annotation ID |
|-----------------------|--------------------------|
| Back Link | id = "adminEditBox" |
| Name Input | id = "adminName" |
| Email Input | id = "adminEmail" |
| Mobile Number Input | id = "adminMobilenumber" |
| Password Input | id = "adminPassword" |
| Company Name Input | id = "companyName" |
| Company Address Input | id = "companyAddress" |
| Save Changes Button | id = "profileEditButton" |

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

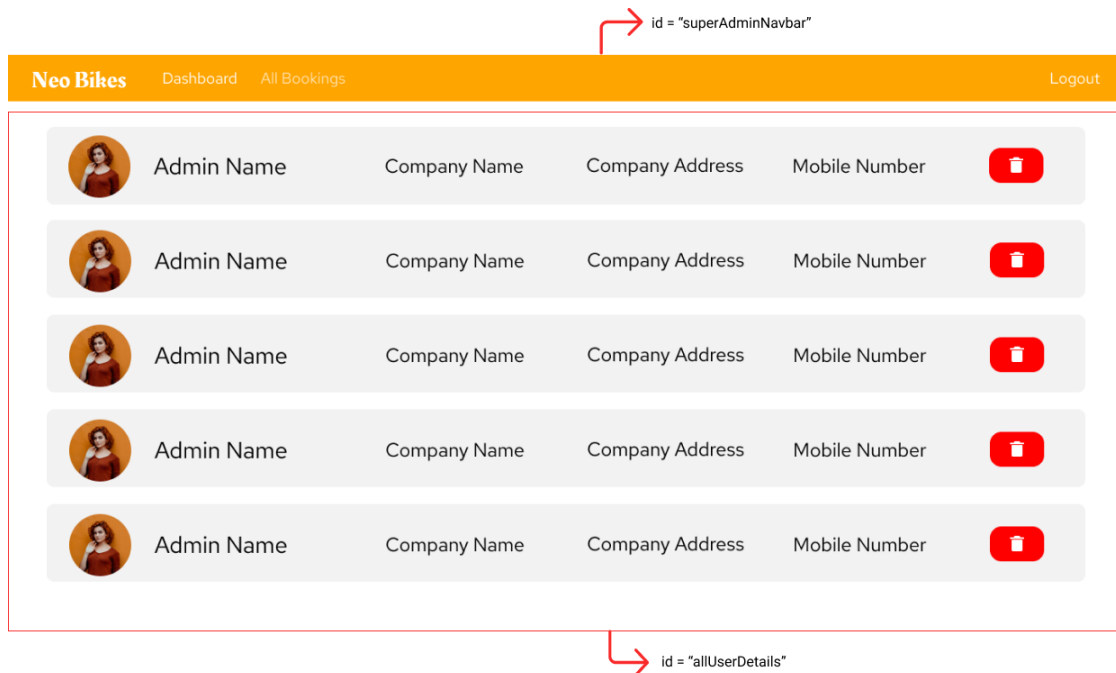
a. Ids:

i. superAdminNavbar

ii. allUserDetails

b. Routing URL: <http://localhost:4200/superadmin/adminList>

c. Screenshot



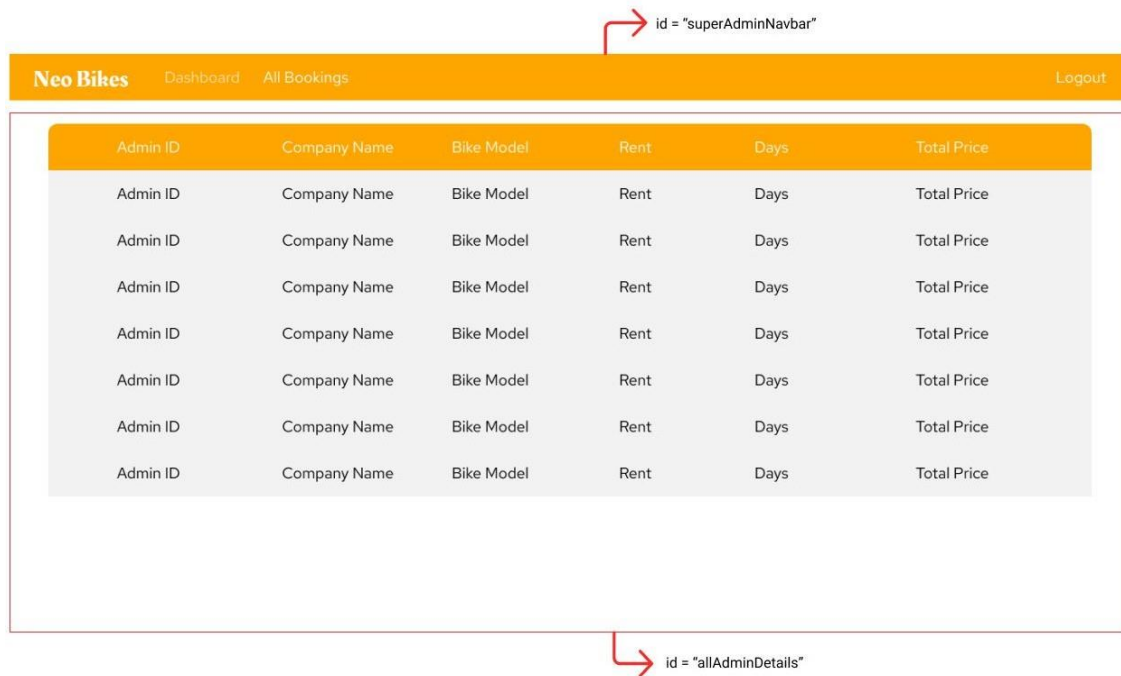
21. Super Admin Bookings page: Design a Super-Admin Booking component where the super admin has authority to view the bookings.

a. Ids:

- i. superAdminNavbar
- ii. allAdminDetails

b. Routing URL : <http://localhost:4200/superadmin/adminBookings>

c. Screenshot



22. Super Admin Navigation: Design a Navigation page for the super admin 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. companyName:String
- vii. companyImageUrl:String
- viii. companyAddress: String
- ix. earnings:int

4. BikeModel: This class stores the cart items.

a. Attributes:

- i. bikeID:String
- ii. bikeNo: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. BikeController: This class controls the save/edit/delete/view bikes.

a. Methods:

- i. saveBike(BikeModel data): This method helps the admin to save new bikes in the database.
- ii. editBike(BikeModel data): This method helps the admin to edit the details of the bikes and save it again in the database.
- iii. deleteBike(String Bike ID): This method helps the admin to delete a bike from the company and as well as in the database.
- iv. getBikes(String Admin_Email_ID): This method helps the admin to get all the bikes in the company.
- v. bookBikes(String Bike_ID): This method helps the user to book the Bike by changing the bike status and add the record in the bookings table.

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

a. Methods:

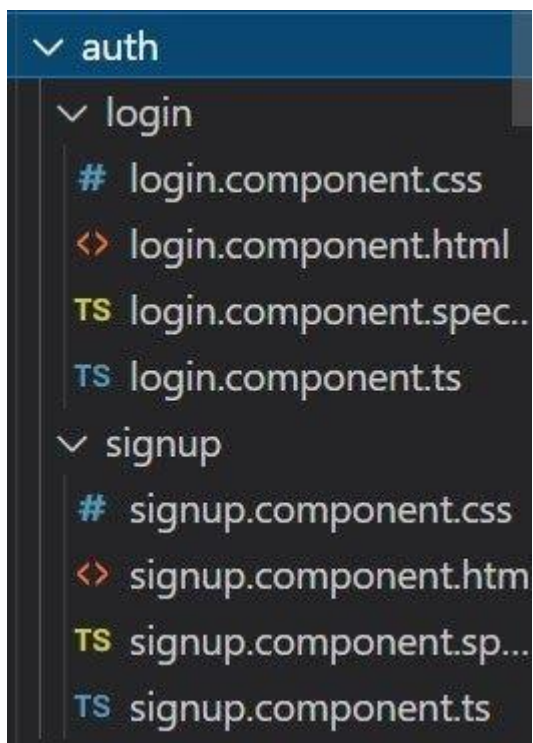
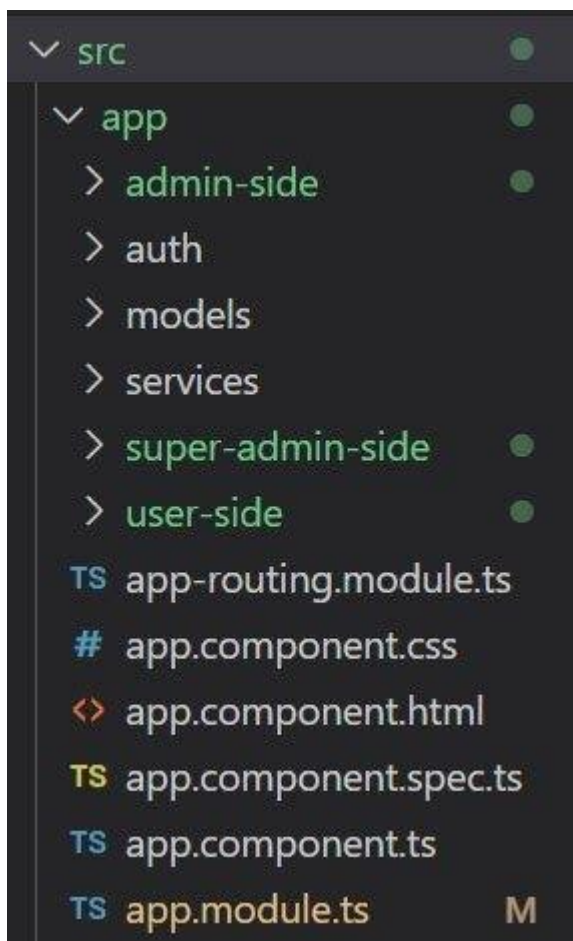
- i. editAdmin(AdminModel data): This method helps the admin to edit the profile details and save it in the database.
- ii. getProfile(String Admin_Email_ID): This method helps admin to get the profile details.

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

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. userBookBike(): This method helps user to book a particular bike.
- iv. userGetBookings(String UserID): This method helps to get all the bookings of the particular user.

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/e2SmP8Xofn2thePbejUqYZ/Neo-Bike-Rental-Admin-Flow?node-id=1%3A482&viewport=410%2C422%2C0.06292035430669785&scaling=scale-down>

User Flow

<https://www.figma.com/proto/7PiGV0IPBtapGvRHVTSvaz/Neo-Bike-Rental-User-Flow?node-id=2%3A31&viewport=638%2C532%2C0.14213642477989197&scaling=scale-down>

Super Admin Flow

<https://www.figma.com/proto/StwdqeHpE472w0MozUukT7/Neo-Bike-Rental-Super-Admin-Flow?node-id=2%3A0&viewport=424%2C446%2C0.027425706386566162&scaling=scale-down>