

Neo Cars - Rental Booking Application

Objective:

Neo Cars is an application to be built as a product that can help customers to choose a Rental car.

Users of the System:

1. Super Admin
2. Admin
3. Customer

Functional Requirements:

- Build an application that customers can book Cars in company.
- The application should have signup, login, profile, dashboard page for user and login,signup,profile , dashboard, add Car page for the admin.
- This application should have a provision to maintain a database for customer information,Admin information,booking information and Car information.
- Also, an integrated platform required for customers, admin and super admin.
- Administration module to include options for adding / modifying / removing the existing Car(s) and customer management.
- **There should be a maximum of 4 people per car.**

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 Companies and Cars based on the customer's price range, specific Company 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 | React 16+ 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. As soon as the project mode is opened and set up, navigate to the settings of the Visual Studio Code by clicking on **File -> preferences -> Settings** or (**Ctrl + ,**).

- a. Click on the settings icon located as the first icon on the right-hand side to open the **settings.json** file.

- b. paste the following code inside it.

```
{ "settings": { "files.exclude": { "**/node_modules": true } } }
```

- c. Then proceed with running the react project.

Note: The above step has to be repeated each time, the project mode is opened up.

- The React app should run in port 8081. Do not run the react app in the port: 3000 or port: 4200.
- 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. The id provided should be defined strictly with the following name **data-testid="<your id>"**. Example : **<input data-testid="username" />**. The naming convention has to be followed for all the id's mentioned.
- 3.
4. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.

5. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
6. 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 Protected Routes from the react router package. For example, if the user enters as <http://localhost:3000/signup> or <http://localhost:3000/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 Cars in the company |
| Admin Add Car | /admin/addCar | POST-Sends Car Data | Car Added |

| | | | |
|------------------|--------------------|-----------------------------|------------------------------|
| Admin Edit Car | /admin/editCar | POST-Sends Car Data | Car Edited |
| Admin Delete Car | /admin/deleteCar | POST-Sends Car ID | Car 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 Companies available |
| Displaying Hotel Cars | /user/Cars | POST - Sends Company name and admin ID of that company | Return all the Cars of the company |
| Car Details | /user/CarDetails | POST-Sends Car ID | Return Car details |
| Booked Cars | /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. userrole
 - v. username
 - vi. age
 - vii. submitButton
 - viii. loginLink
 - b. Routing URL: <http://localhost:3000/user/signup>
 - c. Output screenshot:

The image shows a wireframe of a 'SIGN UP' form. The form is titled 'SIGN UP' and contains several input fields and a submit button. Each element is labeled with an ID, indicated by a red arrow pointing to the element and a label on the right. The elements and their IDs are:

- id = "signupBox" (the entire form container)
- id = "email" (the 'Enter Email' input field)
- id = "password" (the 'Enter Password' input field)
- id = "mobilenumber" (the 'Enter Mobile Number' input field)
- id = "userrole" (the 'User' dropdown menu)
- id = "username" (the 'Enter Username' input field)
- id = "age" (the 'Enter Age' input field)
- id = "submitButton" (the blue 'Submit' button)
- id = "loginLink" (the 'Click Here' link)

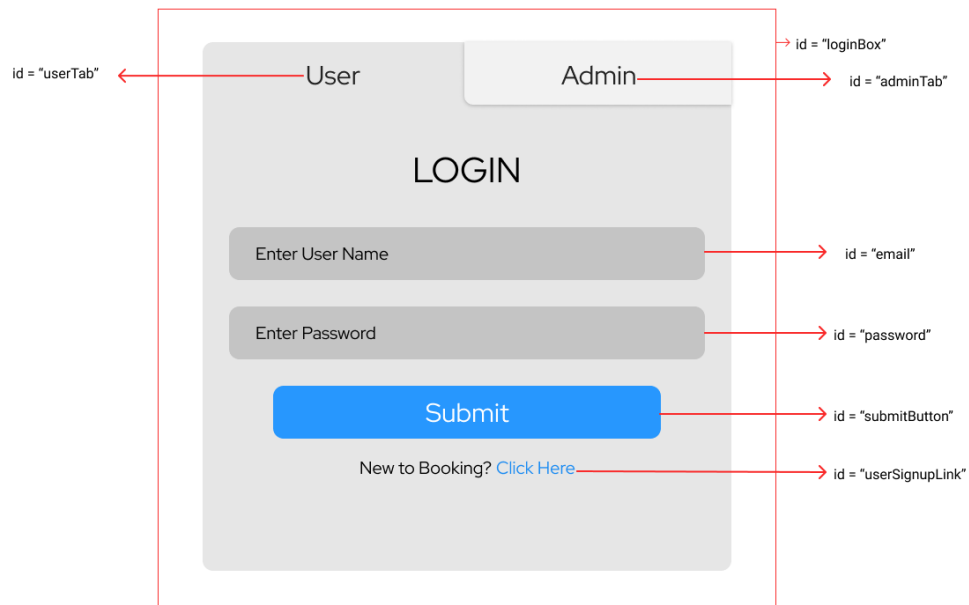
2. Login: Design a login page component where the existing customer can log in using the registered email id and password.

a. Ids:

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

b. Routing URL: <http://localhost:3000/login>

c. Output screenshot:



3. Dashboard / Home: Design a dashboard page component which provides a list of ~~cars~~ **cars** available where the user can book a particular Car.

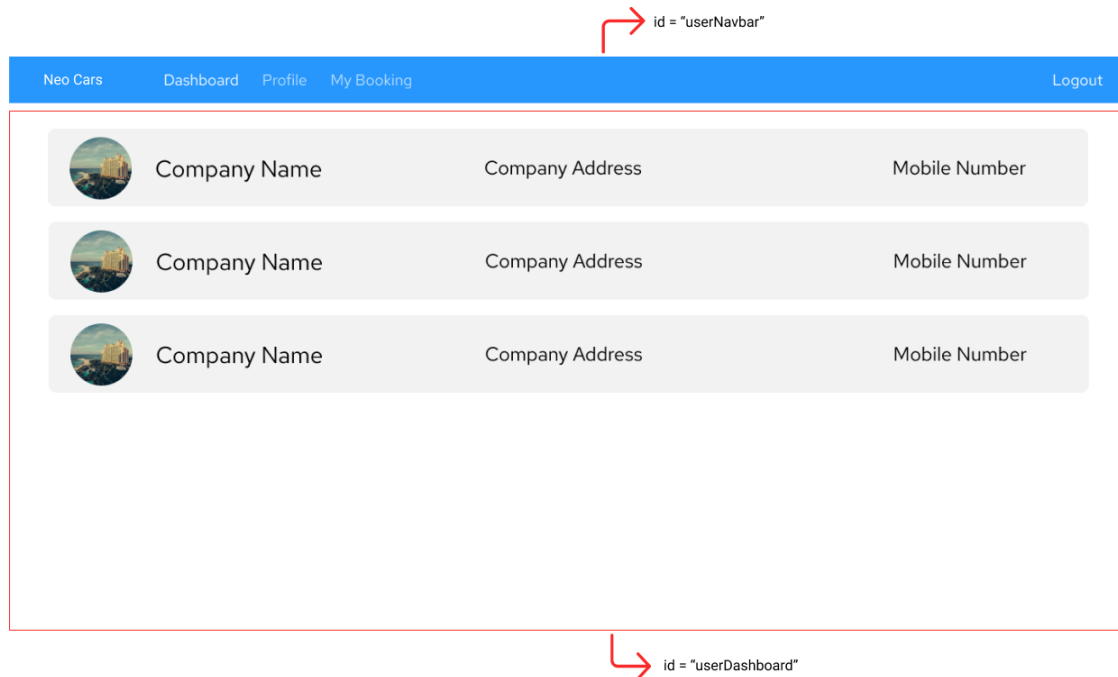
a. Ids:

i. userNavbar

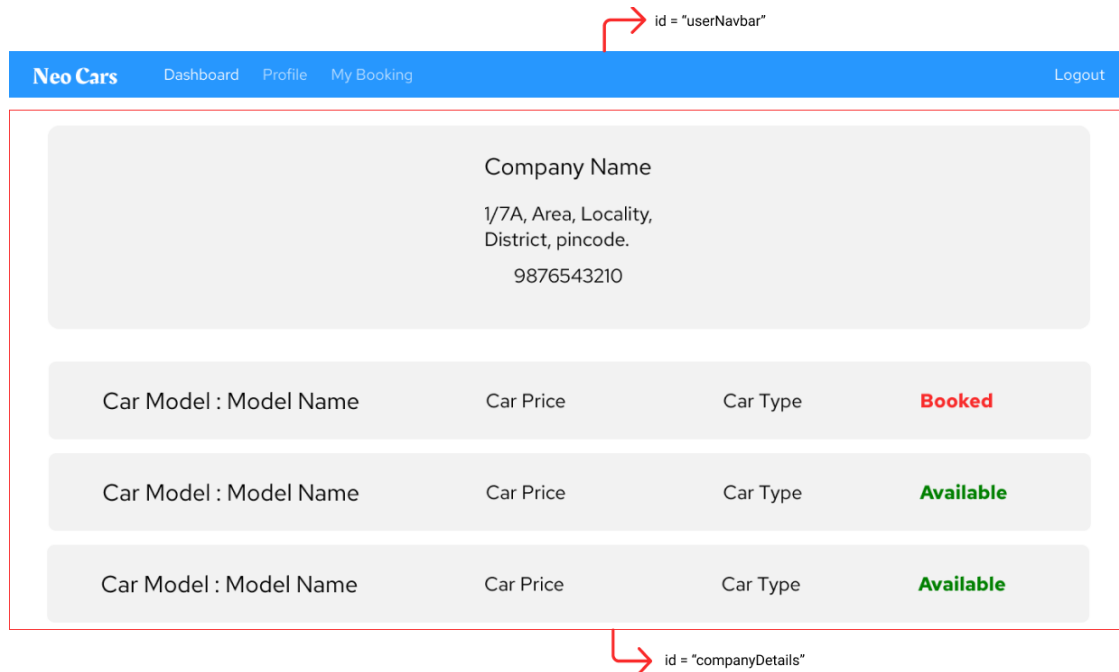
ii. userDashboard

b. Routing URL: <http://localhost:3000/user/dashboard>

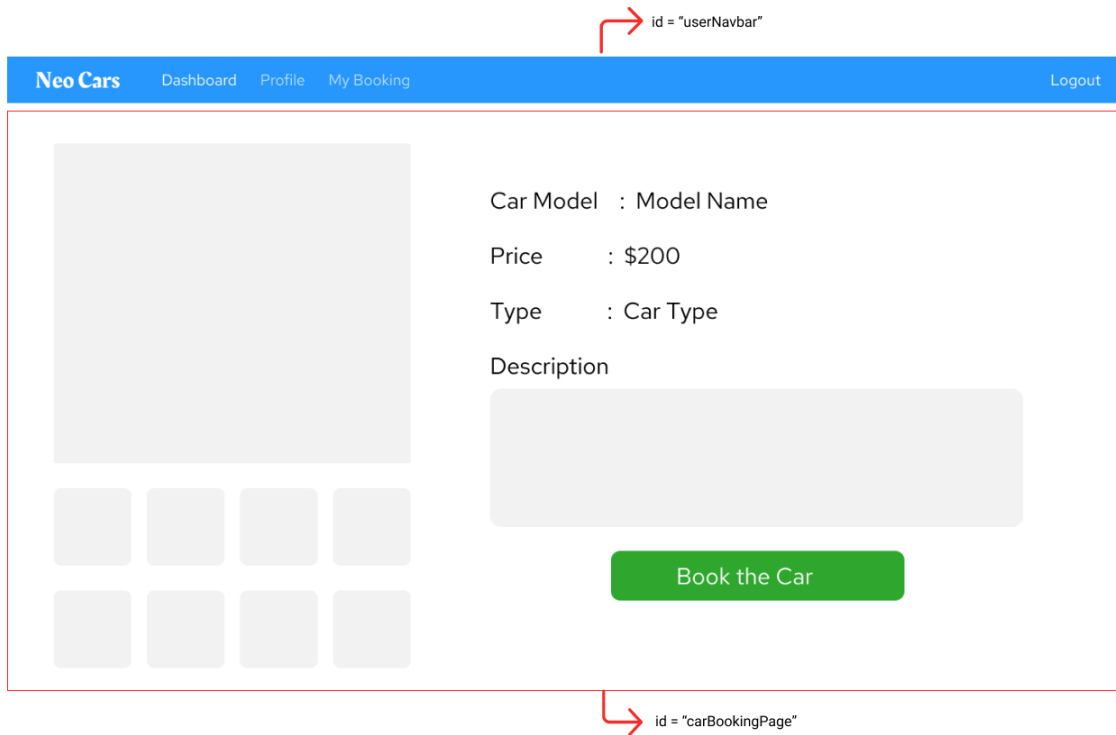
c. Screenshot



4. Company Details : Design a Hotel Details component which lists the total number of Cars available along with the individual status of each Car (available / Booked).
 - a. Ids
 - i. userNavbar
 - ii. companyDetails
 - b. Routing URL : <http://localhost:3000/user/companyDetail/{companyId}>
 - c. Screenshot



5. Car Details: Design a Car Booking Page that displays the details of the Car that the user is currently booking.
 - a. Ids
 - i. userNavbar
 - ii. carBookingPage
 - b. Routing URL: <http://localhost:3000/user/carDetail/{carId}>
 - c. Screenshot

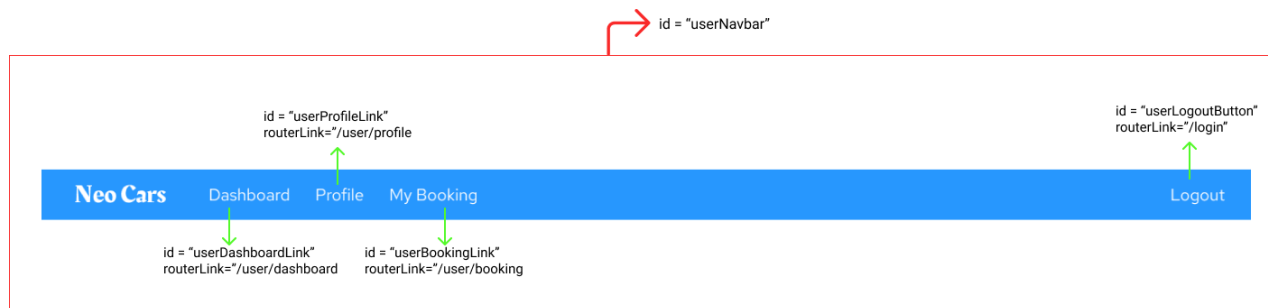


6. User Navigation : Design a Booking page component where the user can navigate to their profile and bookings page.

a. Ids

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

b. Screenshot



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

b. Routing URL : <http://localhost:3000/user/editProfile/{userId}>

c. Screenshot:

id = "userNavbar"

| Neo Cars | | | | | |
|------------------------------|------------|------|------|-------------|--------|
| Dashboard Profile My Booking | | | | | Logout |
| Company Name | Car 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 the user to view their Information.

a. Ids:

i. userNavbar

ii. userProfileBody

b. Routing URL : <http://localhost:3000/user/profile/{userId}>


c. Screenshot:

id = "userNavbar"

| Neo Cars | | | | | |
|------------------------------|--|--|--|--|--------|
| Dashboard Profile My Booking | | | | | Logout |


Name : Jennifer Aniston

Email : rachelgreen@iamneo.ai

Password : ***** 

Age : 27

Mobile Number : 1234567890



Edit Profile

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. mobilenumber
- v. userrole
- vi. adminname
- vii. companyname
- viii. companyimageURL
- ix. companyAddress
- x. submitButton
- xi. adminLoginLink

b. Routing URL : <http://localhost:3000/admin/signup>

c. Screenshot

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

- id = "signupBox" (points to the entire form container)
- id = "email" (points to the 'Enter Email' field)
- id = "password" (points to the 'Enter Password' field)
- id = "mobilenumber" (points to the 'Enter Mobile Number' field)
- id = "userrole" (points to the 'Admin' dropdown menu)
- id = "adminname" (points to the 'Enter Seller Name' field)
- id = "companyname" (points to the 'Enter Company Name' field)
- id = "companyimageURL" (points to the 'Enter Company Image Url' field)
- id = "companyAddress" (points to the 'Enter Company Address' field)
- id = "submitButton" (points to the blue 'Submit' button)
- id = "adminLoginLink" (points to the 'Click Here' link in the 'Go to Login Click Here' text)

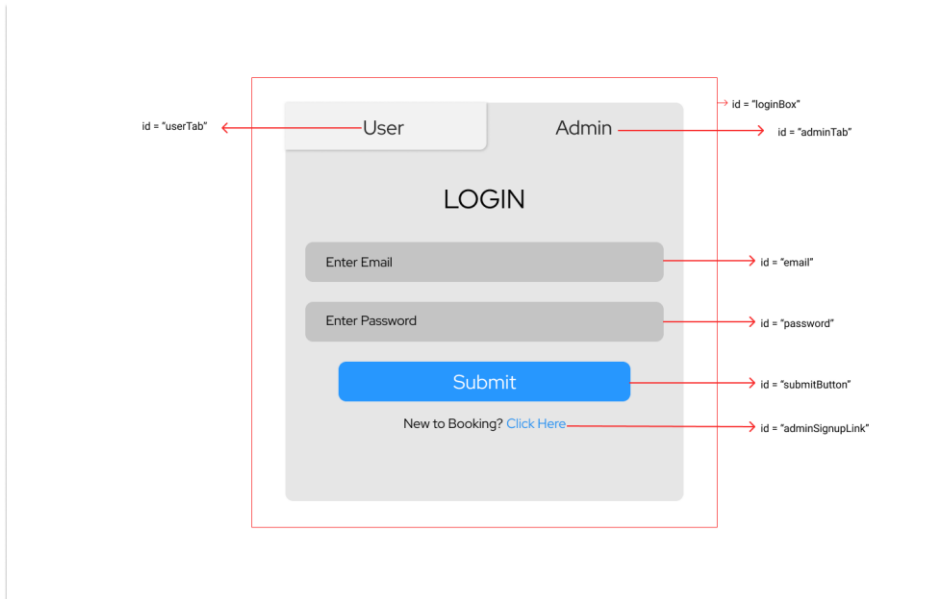
11. Admin Login: Design an Admin Login page that allows admin to login.

a. Ids

- i. loginBox
- ii. adminTab
- iii. email
- iv. password
- v. submitButton
- vi. adminSignupLink

b. Routing URL: <http://localhost:3000/login>

c. Screenshots



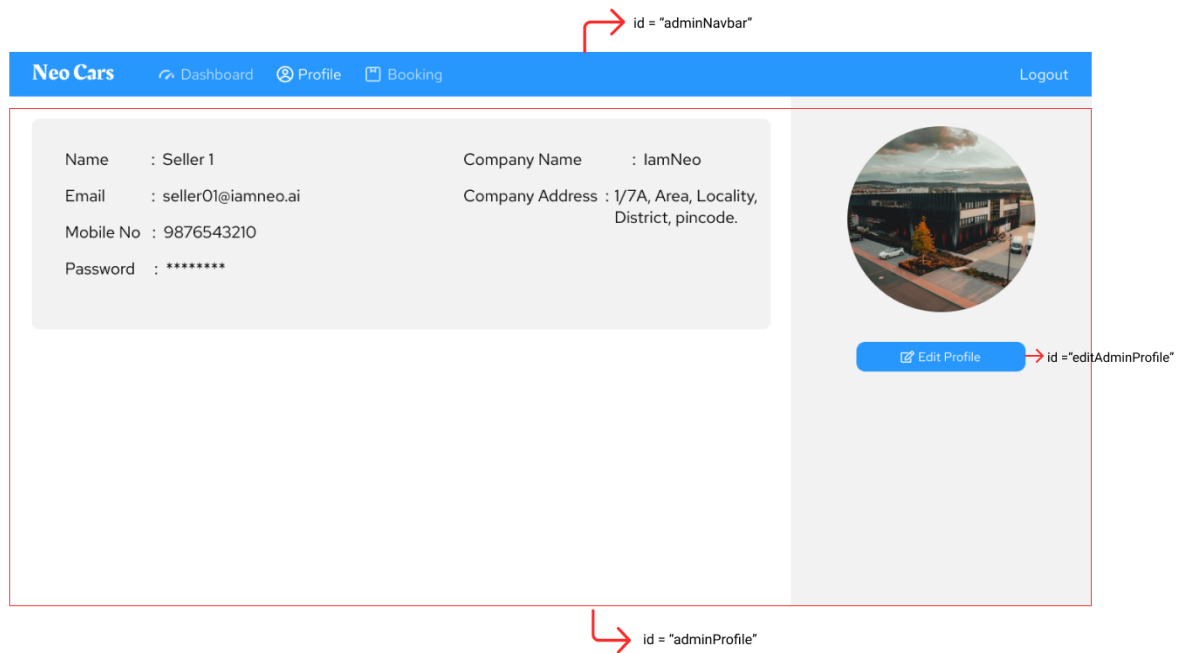
12. Admin Profile: Design an Admin Profile page that displays the details of the admin currently logged in.

a. Ids

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

b. Routing URL: <http://localhost:3000/admin/profile/{adminId}>

c. Screenshots



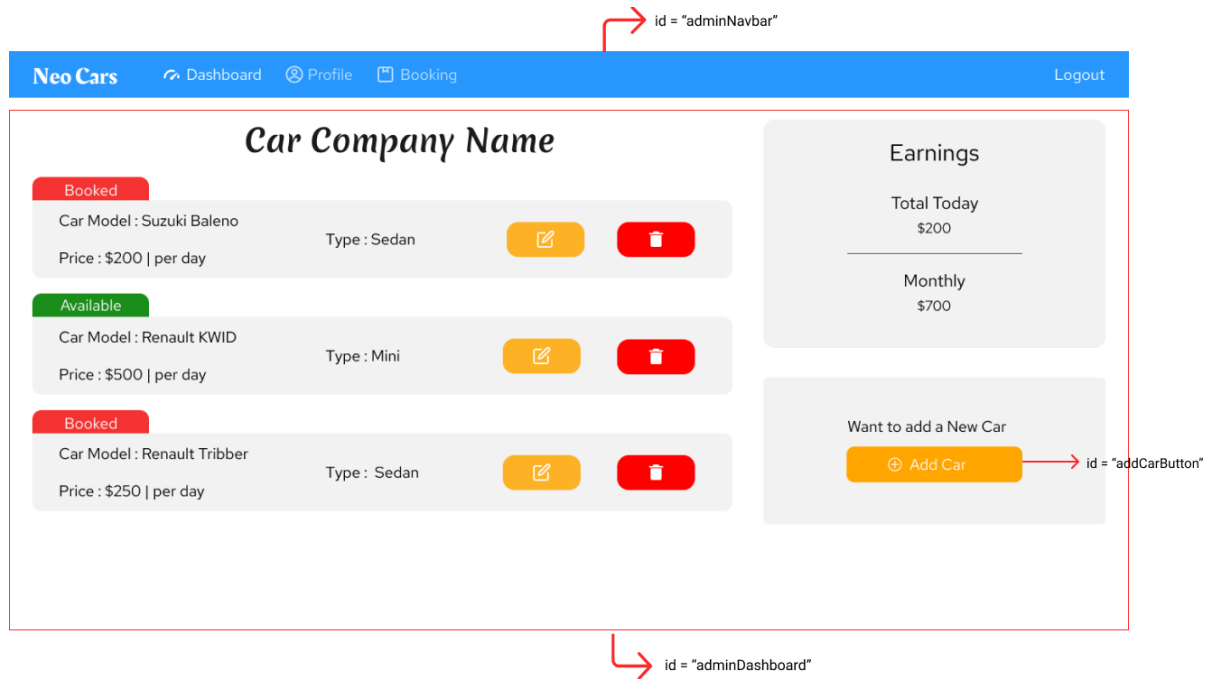
13. Admin Dashboard: Design a dashboard page where the list of cars is displayed on the admin side.

a. Ids

- i. adminNavbar
- ii. addCarButton
- iii. adminDashboard

b. Routing URL: <http://localhost:3000/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:



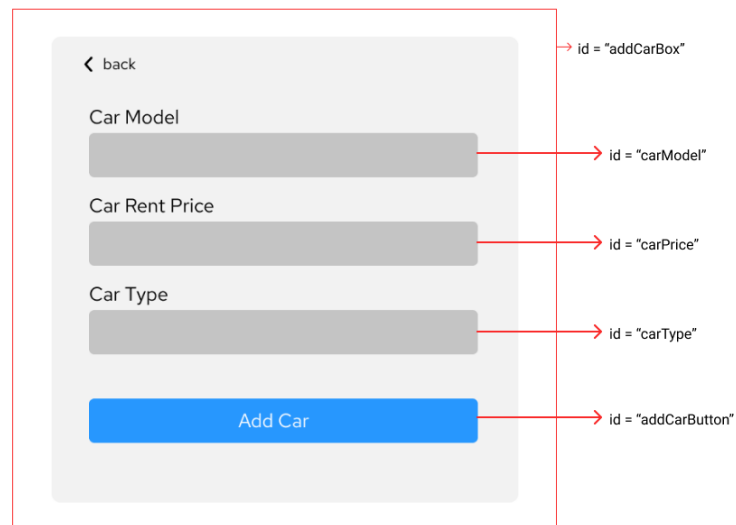
15. Admin Add Car: Design an Add Car component in which the admin can add new Cars to the inventory.

a. Ids:

- i. adminNavbar
- ii. addCarBox
- iii. carNo
- iv. carPrice
- v. carType
- vi. addCarButton

b. Routing URL: <http://localhost:3000/admin/addCar>

c. Screenshot



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

a. Ids:

- i. adminNavbar
- ii. editCarBox
- iii. carNo
- iv. carPrice
- v. carType
- vi. editCarButton

b. Routing URL: <http://localhost:3000/admin/editCar/{carId}>

c. Screenshot

The screenshot shows a form for editing car details. It includes a back button, input fields for Car Model, Car Rent Price, and Car Type, and a Save Change button. Red arrows point from text labels to the corresponding UI elements.

- id = "adminNavbar" (points to the top navigation bar)
- id = "editCarBox" (points to the form container)
- id = "carModel" (points to the Car Model input field)
- id = "carPrice" (points to the Car Rent Price input field)
- id = "carType" (points to the Car Type input field)
- id = "editCarButton" (points to the Save Change button)

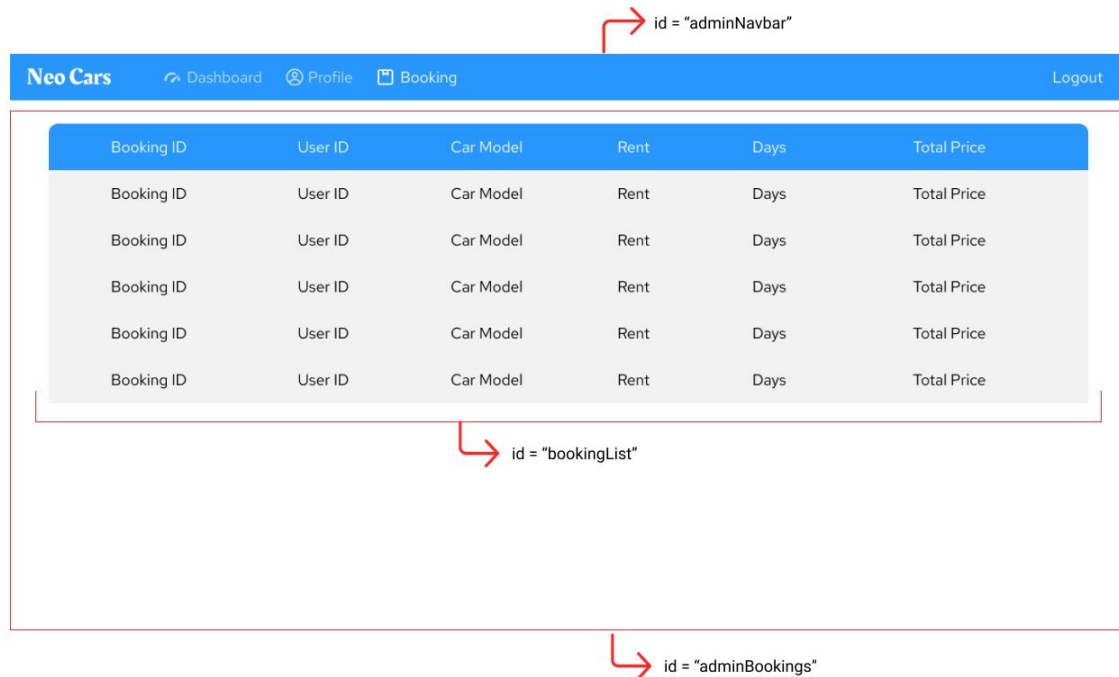
17. Admin Bookings: Design an Admin Bookings component where the admin can see the booking details that all the users have made till present date.

a. Ids

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

b. Routing URL: <http://localhost:3000/admin/bookings/{adminId}>

c. Screenshot:



The screenshot shows the Neo Cars Admin Dashboard. At the top is a blue navigation bar with the logo 'Neo Cars' and links for 'Dashboard', 'Profile', 'Booking', and 'Logout'. Below the navbar is a table with 6 columns: 'Booking ID', 'User ID', 'Car Model', 'Rent', 'Days', and 'Total Price'. The table contains 5 rows of placeholder data. Red arrows indicate the following IDs: 'id = "adminNavbar"' points to the navigation bar, 'id = "bookingList"' points to the table, and 'id = "adminBookings"' points to the container surrounding the table.

| Booking ID | User ID | Car Model | Rent | Days | Total Price |
|------------|---------|-----------|------|------|-------------|
| Booking ID | User ID | Car Model | Rent | Days | Total Price |
| Booking ID | User ID | Car Model | Rent | Days | Total Price |
| Booking ID | User ID | Car Model | Rent | Days | Total Price |
| Booking ID | User ID | Car Model | Rent | Days | Total Price |
| Booking ID | User ID | Car Model | Rent | Days | Total Price |

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:3000/admin/editProfile/{adminId}>

c. Screenshot:

The screenshot shows a user profile edit form with the following fields and annotations:

- Name:** Seller 1 (id = "adminName")
- Email:** seller01@iamneo.ai (id = "adminEmail")
- Mobile Number:** 9876543210 (id = "adminMobilenumber")
- Password:** iamneo (id = "adminPassword")
- Company Name:** Examly (id = "companyName")
- Company Address:** 1/7A, Area, Locality, District, pincode (id = "companyAddress")
- Save Changes:** (id = "profileEditButton")

The form is enclosed in a red box, and a red arrow points to the top right corner of the box with the label id = "adminEditBox".

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:3000/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 users page where the super admin has authority to view and delete any admin currently registered into the system.

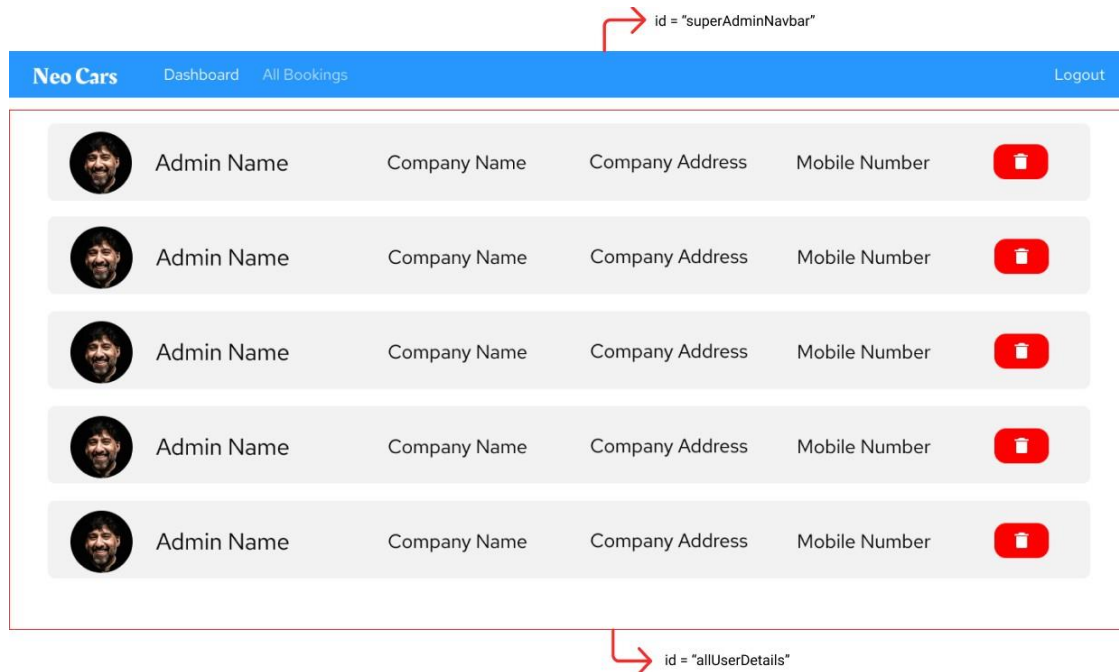
a. Ids:

i. superAdminNavbar

ii. allUserDetails

b. Routing URL: <http://localhost:3000/superadmin/adminList>

c. Screenshot



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

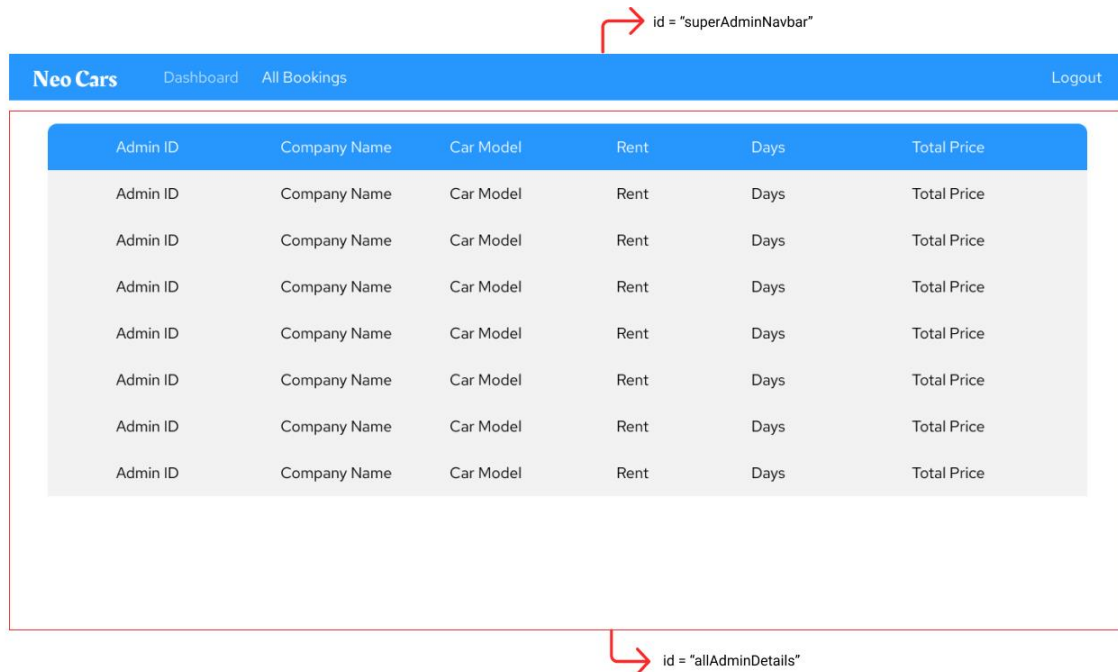
a. Ids:

i. superAdminNavbar

ii. allAdminDetails

b. Routing URL : <http://localhost:3000/superadmin/adminBookings>

c. Screenshot

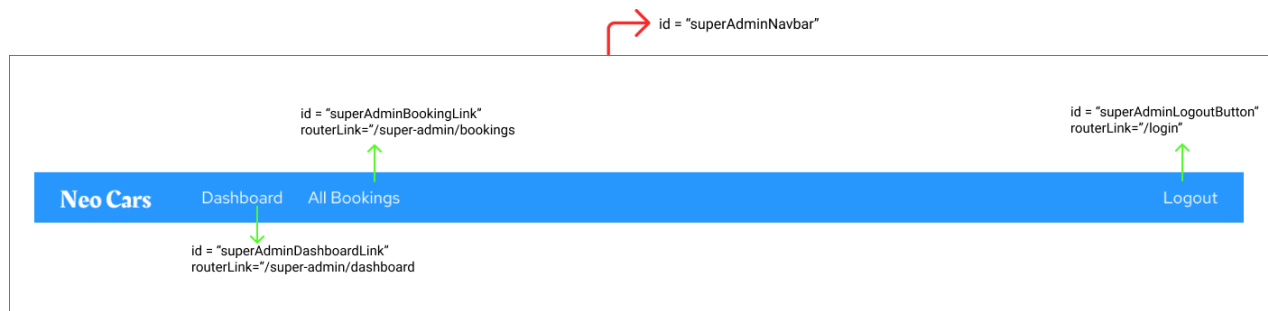


22. Super Admin Navigation: Design a Navigation page for the super admin where he can navigate to the dashboard, all the bookings and logout.

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. CarModel: This class stores the cart items.

a. Attributes:

- i. carID:String
- ii. carNo: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. CarController: This class controls the save/edit/delete/view Cars.

a. Methods:

- i. `saveCar(CarModel data)`: This method helps the admin to save new Cars in the database.
- ii. `editCar(CarModel data)`: This method helps the admin to edit the details of the Cars and save it again in the database.
- iii. `deleteCar(String CarID)`: This method helps the admin to delete a Car from the hotel and as well as in the database.
- iv. `getCars(String Admin_Email_ID)`: This method helps the admin to get all the Cars in the hotel.
- v. `bookCar(String Car_ID)`: This method helps the user to book the Car by changing the Car 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.

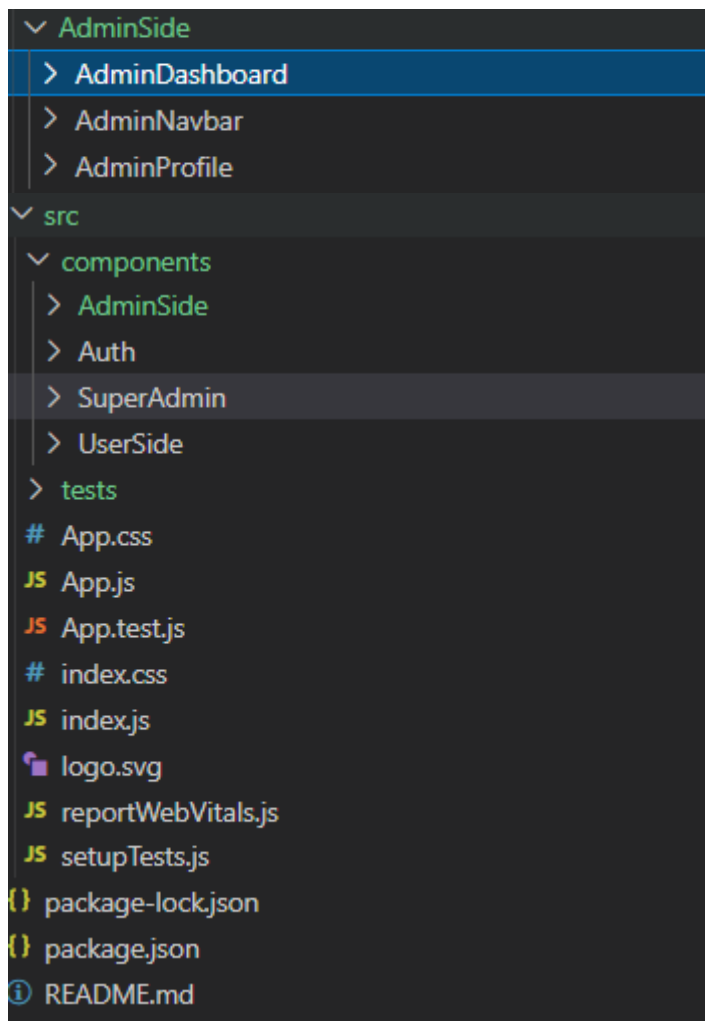
8. UserController: This class helps to get the Cars.

a. Methods:

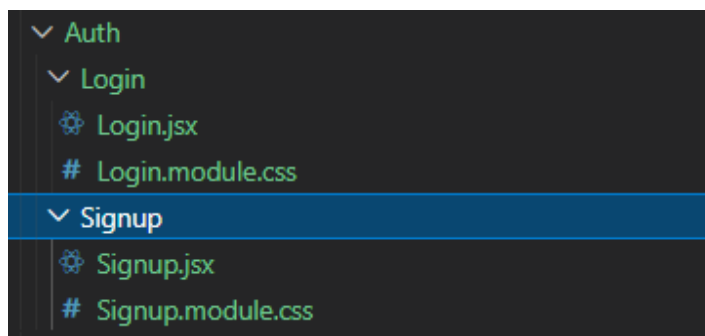
- i. `userBookings(String User_ID)`: This method helps users to get all the booking details done by them.
- ii. `userProfileEdit(String User ID)`: This method helps users to edit their own details.
- iii. `userBookCar()`: This method helps user to book a particular car.
- iv. `userGetBookings(String UserID)`: This method helps to get all the bookings of the particular user.

React Folder Structure:

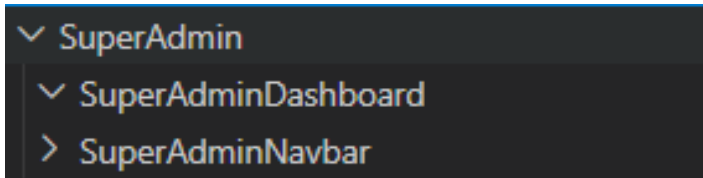
Admin Side:



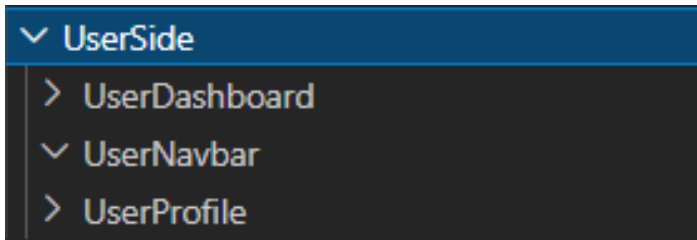
Auth Side:



Super Admin Side:



User Side:



React component Structure:

- **src (folder)**
 - **components (folder)**
 - **Navbar (folder)**
 - **Navbar.jsx (jsx file)**
 - **Navbar.module.css (css file)**

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

<https://www.figma.com/proto/65gAU93Fw4R351v2NnzHIC/Neo-Car-Rental-Admin-Flow?node-id=1%3A482&viewport=587%2C493%2C0.1219395324587822&scaling=scale-down>

User

<https://www.figma.com/proto/U5BYMfRAOfRxZkhOgO04ie/Neo-Car-Rental-User-Flow?node-id=1%3A33&viewport=632%2C540%2C0.07334188371896744&scaling=scale-down>

Super Admin

<https://www.figma.com/proto/lxLNSqPprebmhIC3pdaxBy/Neo-Car-Rental-Super-Admin-Flow?node-id=1%3A4&viewport=369%2C302%2C0.04095541685819626&scaling=scale-down>