Car Pooling

Objective:

Car Pooling is an online application to be built as a product that enables the corporate employees within an organization to avail the facility of car pooling effectively.

Users of the System:

1. Admin
2. Employees

Functional Requirements:

- A system for an Admin who can enter the employee details like name, contact number, vehicle details etc
- Corporate employees can register the details to the website.
- The facility to check whether the vehicle and driver is authorized or not
- Admin can view the report of the car pooling process to improve the system
- Employees can report suggestions/complaints in the website
- Employees receive SMS alerts regarding the route and timings.
- **Maximum 4 Person allowed for a car.**

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

- Secure access of confidential data (car pooling details).
- Flexible service based architecture will be highly desirable for future extension

Output/ Post Condition:

- System will generate Monthly Reports and Weekly Reports

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 • Car Pooling -< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance
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	React 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 Pre-requisites (Do's and Don'ts):

1. The React app should run in port 8081. Do not run the React 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 AuthGaurd 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:

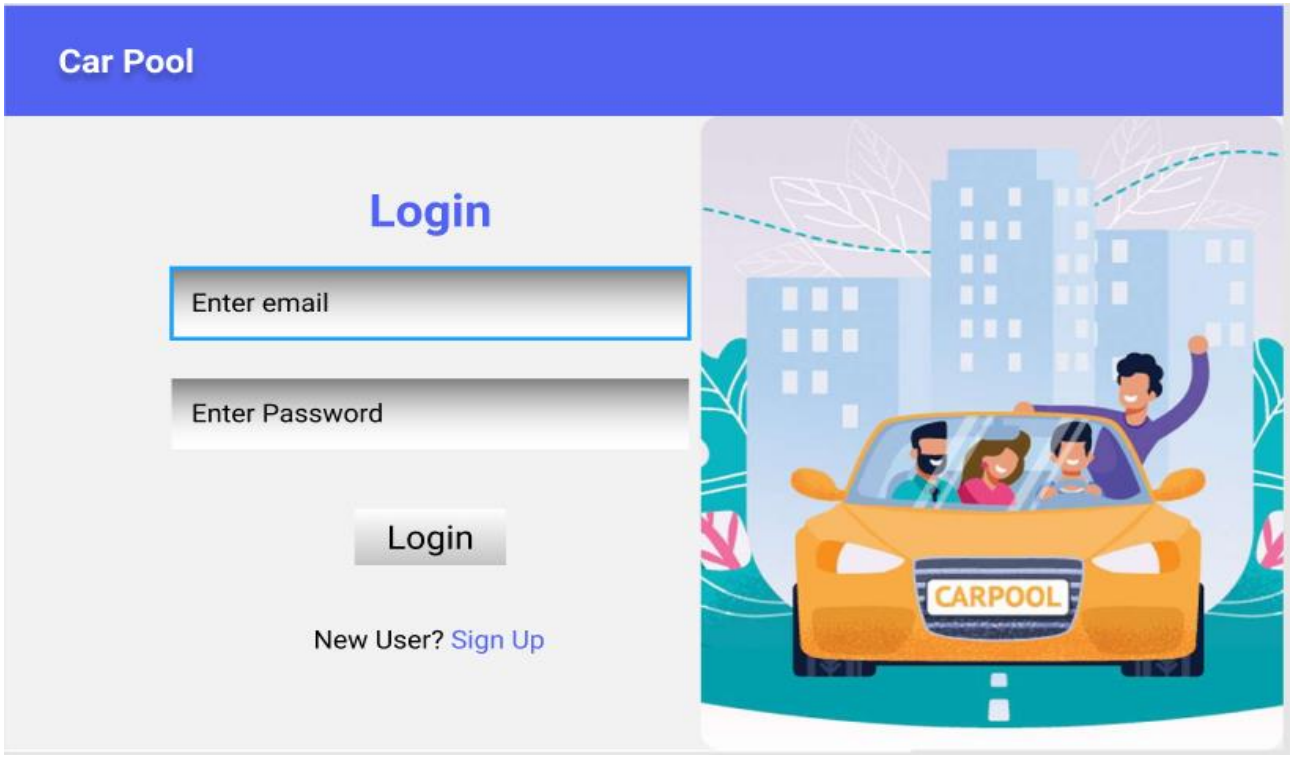
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get Routes	/route	GET	Array of Route
Get RouteById	/route/{id}	POST	Route Details
Edit Customer	/editCustomer/{id}	PUT	Success
New Booking	/saveBooking	POST	Cart items added to the Orders list
Remove Booking	/deleteBooking	DELETE	Booking removed
ADMIN			
Action	URL	Method	Response
Get Employee	/admin/getEmployee	GET	Array of Employee
Get EmployeeById	/admin/getEmployeeById/{id}	GET	Employee Details
Delete Employee	/admin/delete/{id}	DELETE	Employee Delete
Edit Employee	/admin/editEmployee/{id}	GET	Get All details of Particular id
Save Employee	/admin/saveEmployee/{id}	POST	Save new employees
Get All routes	/admin/routes	GET	Array of Routes
Add Route	/admin/addRoutes	POST	Routes added successfully
Edit Routes	/admin/editRoutes/{id}	PUT	Successfully routes edited
Delete Route	/admin/deleteRoutes	DELETE	Routes deleted successful

Frontend:

Employees:

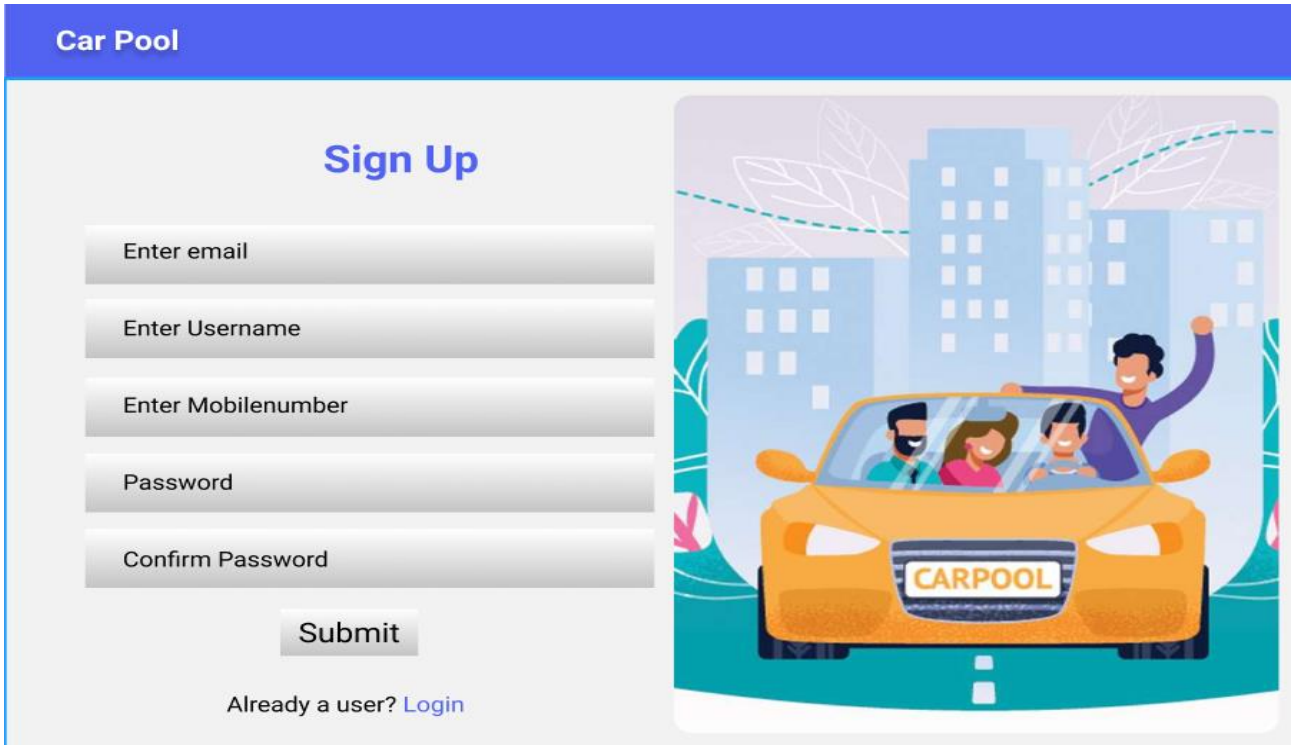
Login:

Output Screenshot:



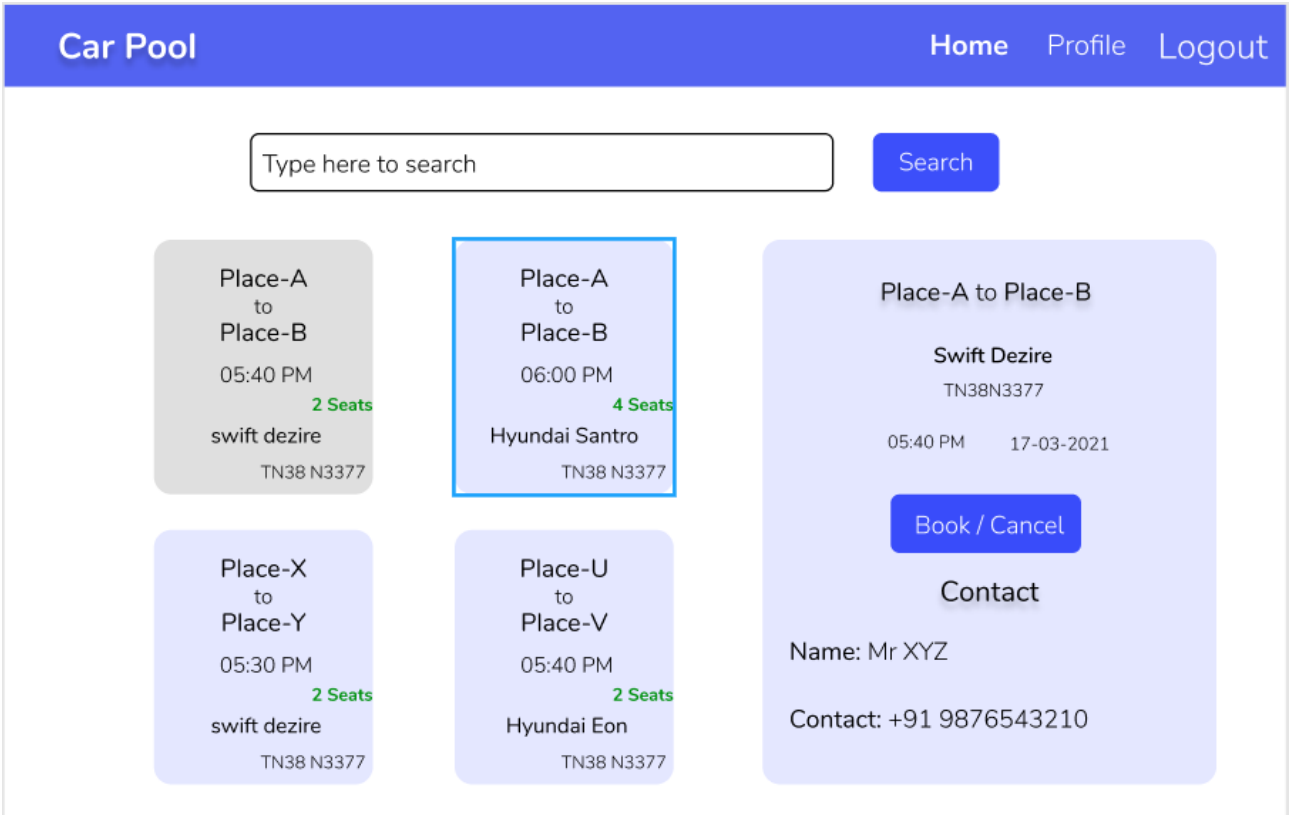
Signup:

Output Screenshot:



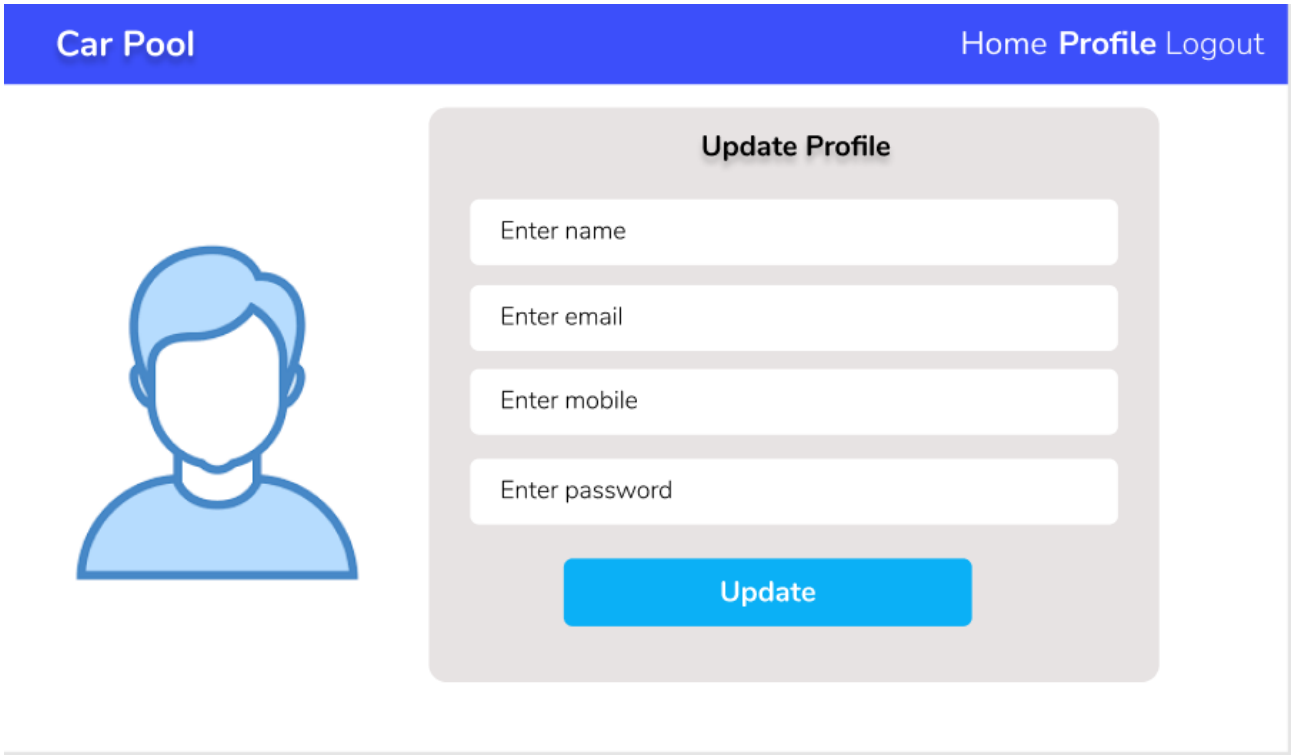
Home:

Output Screenshot:



Profile:

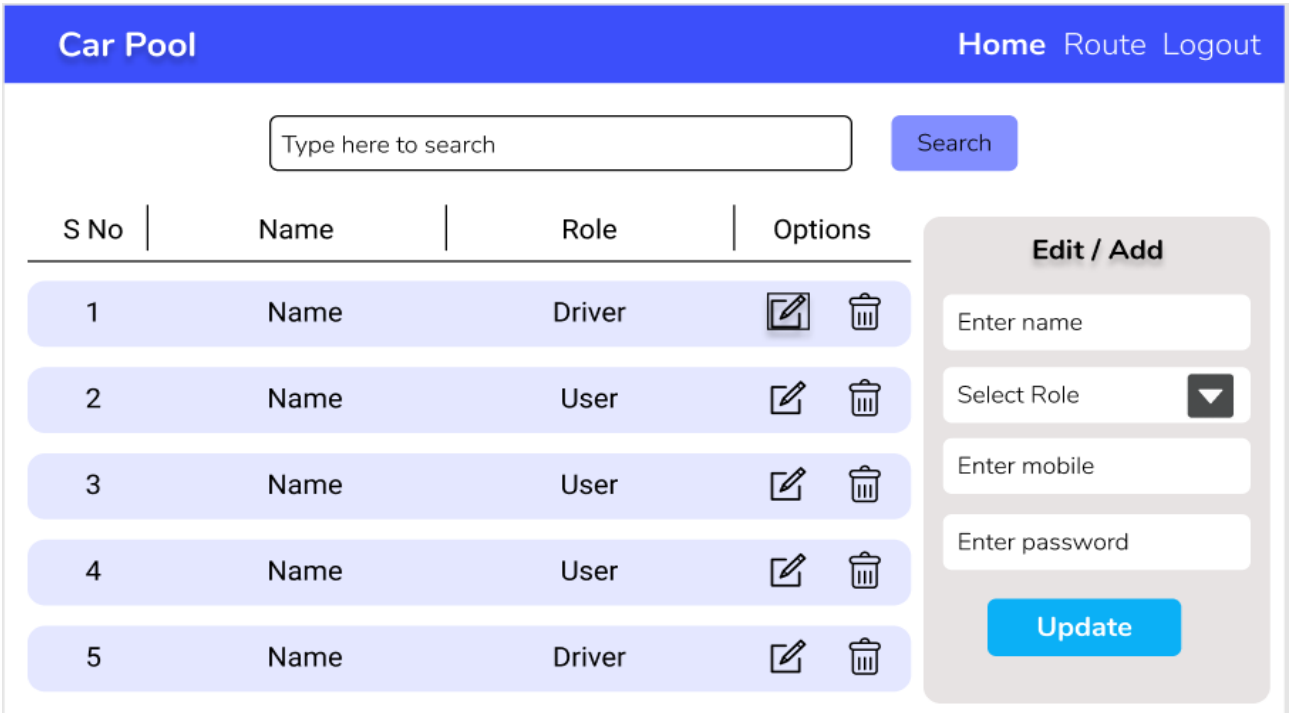
Output Screenshot:



Admin:

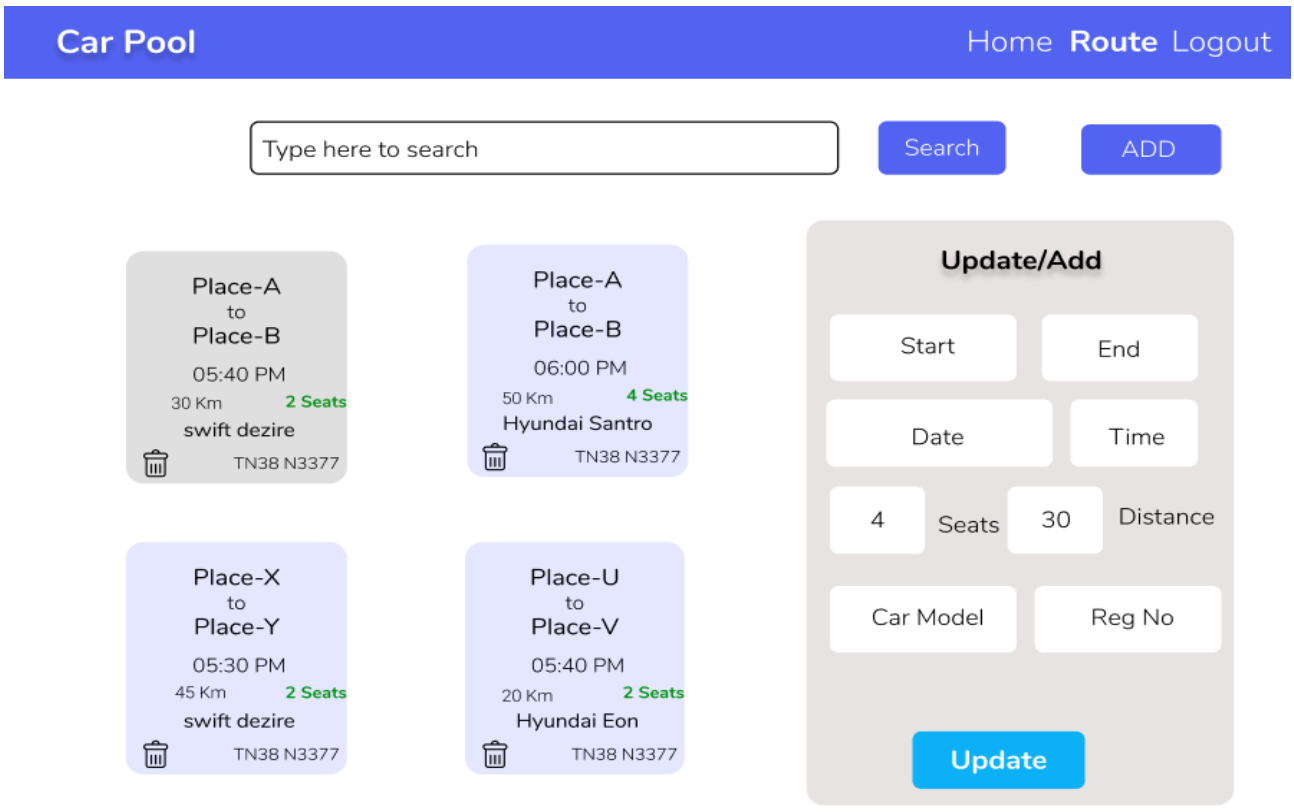
Home:

Output Screenshot:



Route:

Output Screenshot:



Backend:

Class and Method description:

Model Layer:

1. EmployeesModel: This class stores the details of the Employees information.
    a. Attributes:
        i. username: String
        ii. password: String
        iii. email: String
        iv. mobileNumber: Long
        v. vehicleModel: String
        vi. vehicleNumber: String
        vii. verified: Boolean
        viii. active: Boolean
    b. Methods: -
2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. CustomerModel: This class stores the details of the Corporate Employees.
    a. Attributes:
        i. customerId: String
        ii. customerName: String
        iii. emailId: String
        iv. mobileNumber: Long
        v. status: Boolean
    b. Methods: -
4. RouteModel: This class stores the route details.
    a. Attributes:
        i. routeId: int
        ii. startPoint: String

iii. endPoint: String

iv. distance: int

b. Methods: -

Controller Layer:

5. SignupController: This class control the user signup
    a. Attributes: -
    b. Methods:
        i. saveCustomer(CustomerModel user): This method helps to store Customer in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
    a. Attributes: -
    b. Methods:
        i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. EmployeeController: This class controls the add/edit/update/view Employees.
    a. Attributes: -
    b. Methods:
        i. List<EmployeModel> getEmployee(): This method helps the admin to fetch all employees from the database.
        ii. EmployeModel getEmployeeById(String id): This method helps to retrieve a Employee from the database based on the employee id.
        iii. EmployeeModel editEmployee(EmployeeModel data): This method helps to edit a employee and save it to the database.
        iv. saveEmployee(EmployeeModel data): This method helps to add a new employee to the database.
        v. deleteEmployee(String id): This method helps to delete a Employee from the database.
8. RouteController: This class helps in add, delete, update, delete the Route.
    a. Attributes: -
    b. Methods:
        i. List<RouteModel> getRoute(): This method helps the Customer to fetch all route from the database.
        ii. RouteModel getRouteById(String id): This method helps to retrieve a Route from the database based on the route id.

iii. RouteModel editRoute(RouteModel data): This method helps to edit a route and save it to the database.

iv. saveRoute(RouteModel data): This method helps to add a new Route to the database.

v. deleteRoute(String id): This method helps to delete a Route from the database.

9. BookingController: This class helps to booking the car.
    a. Attributes: -
    b. Methods:
        i. List<BookingModel> getBooking (): This method helps the admin to fetch all booking from the database.
        ii. BookingModel getBookingById(String id): This method helps to retrieve a Booking from the database based on the userId.
        iii. saveBooking(BookingModel): This method helps the customer to add a new Bookingto the database.