Car Rentals

CSCI 5308 Quality Assurance

GROUP – 4

Anant Pillai (anant.pillai@dal.ca)

Bhumi Patel (bh842792@dal.ca)

Mohit Kacha (mohit@dal.ca)

Theja Manasa (manasa@dal.ca)

# Introduction

Car Rentals is a website which will allow a user to rent a car for a specific amount of time. User can also rent their own car if they want to.

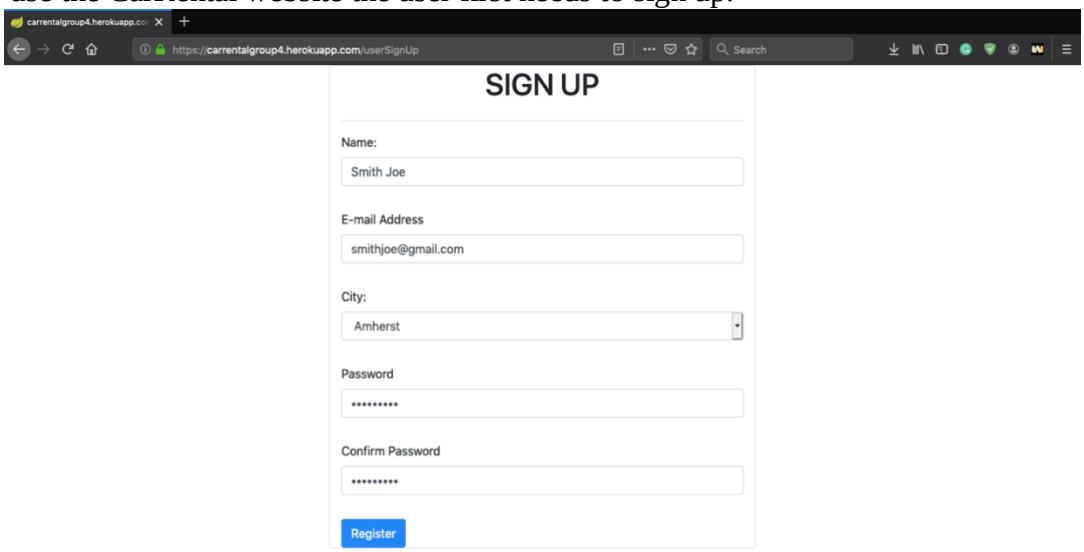
Link for the user: https://carrentalgroup4.herokuapp.com/login

Link for the admin: https://carrentalgroup4.herokuapp.com/admin/login

Credential for admin:
Username: admin
Password: adminPassword

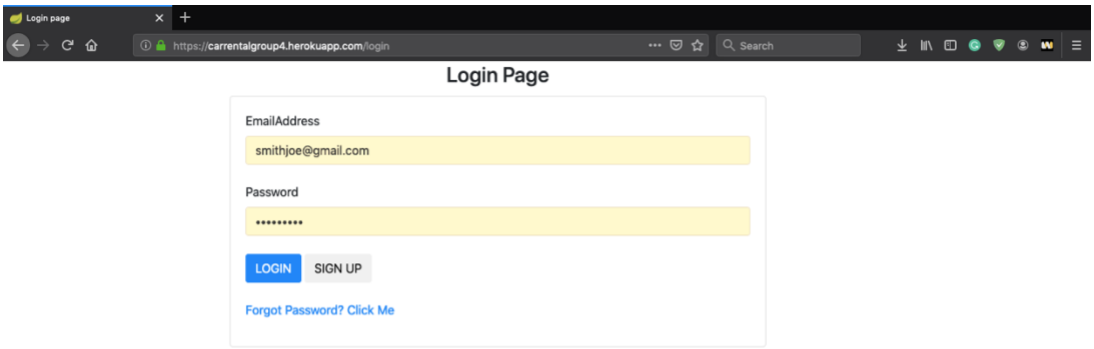
# Features

- Registration
  To use the CarRental website the user first needs to sign up.



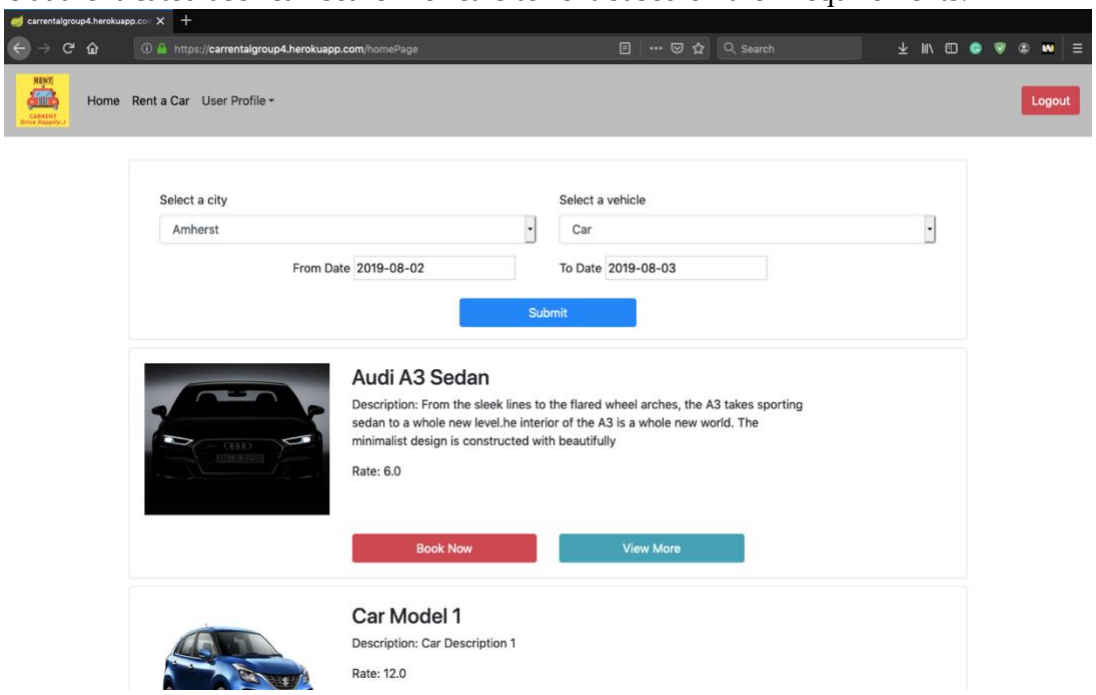
- Login

  Once the user is registered the user can login and use the CarRental website.



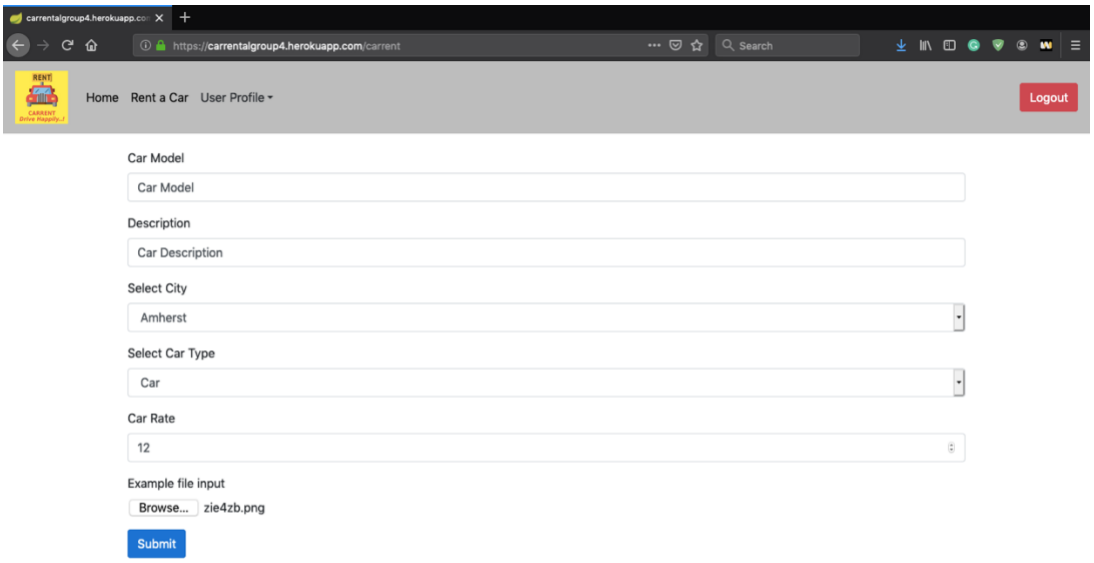

- Search for Cars

  The authenticated user can search for cars to rent based on their requirements.



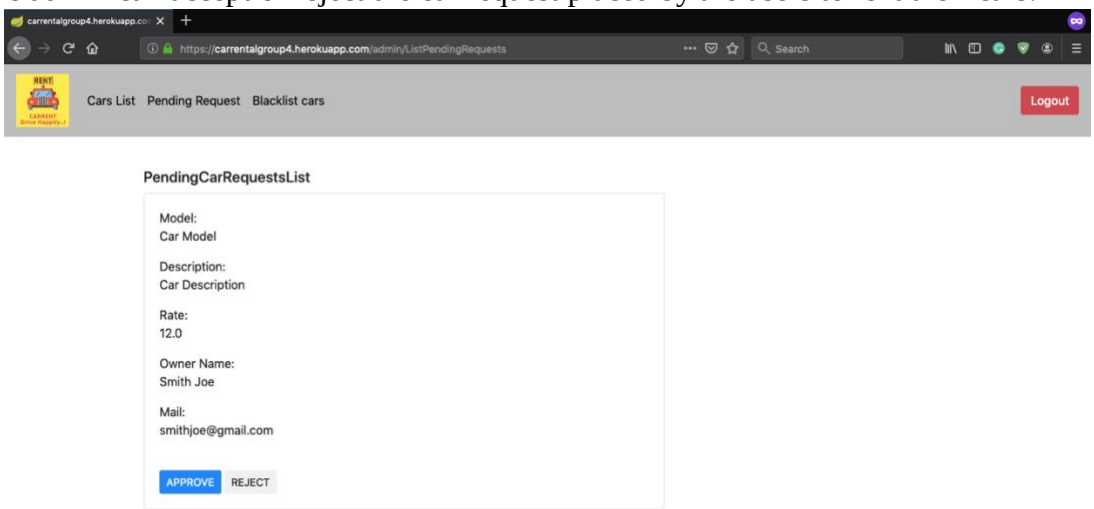
- Rent your car

  The user can place their own car to be rented by other users.



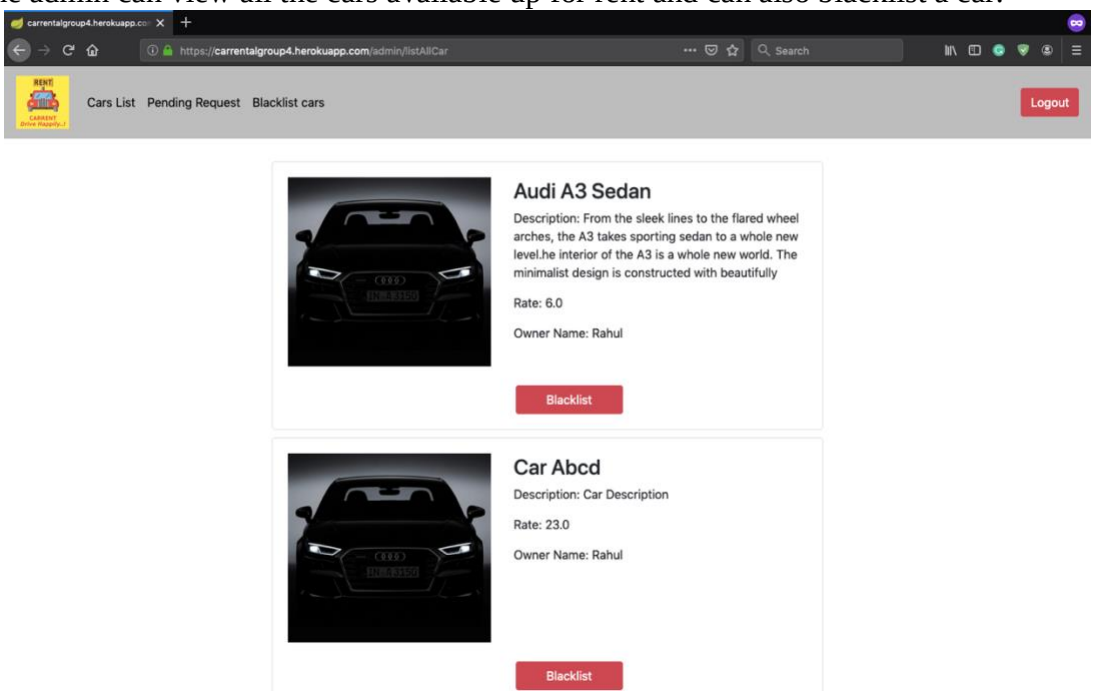

- Accept/Reject Car Request (Admin)

  The admin can accept or reject the car request placed by the users to rent their cars.



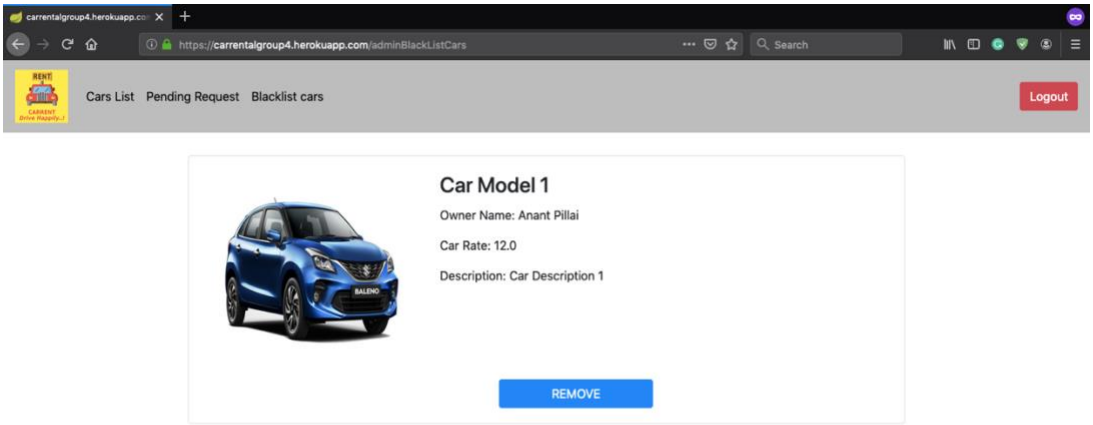
- List All Cars (Admin)

  The admin can view all the cars available up for rent and can also blacklist a car.




- Blacklisted Cars (Admin)

  The admin can view all the blacklisted car and remove them from the blacklist also.



## Tools and Technologies Used

| | |
|---|---|
| Backend Technology | Spring Boot and Java |
| Frontend | HTML, CSS, JSP |
| Database | MySQL |
| Continuous Integration | Jenkins |
| Deployment Server | Heroku |
| Version Control | Github |
| Test Cases | Junit and Mockito |
| Task Tracker | Trello |

## Design Pattern Implementation

- **Chain of responsibility:** Logging

To enable more than one receiver to handle the request based on the runtime conditions (i.e.., either error, warn or info logger) instead of coupling the request/class to a specific receiver.

- **Singleton**: Database connection

Singleton ensures to have only one instance of a class in the memory at a given point of time. So, we implemented Singleton in database connection class, as it avoids resource clogging by instantiating only one object for the class.

- **Singleton** : Authentication (Validate User Session)

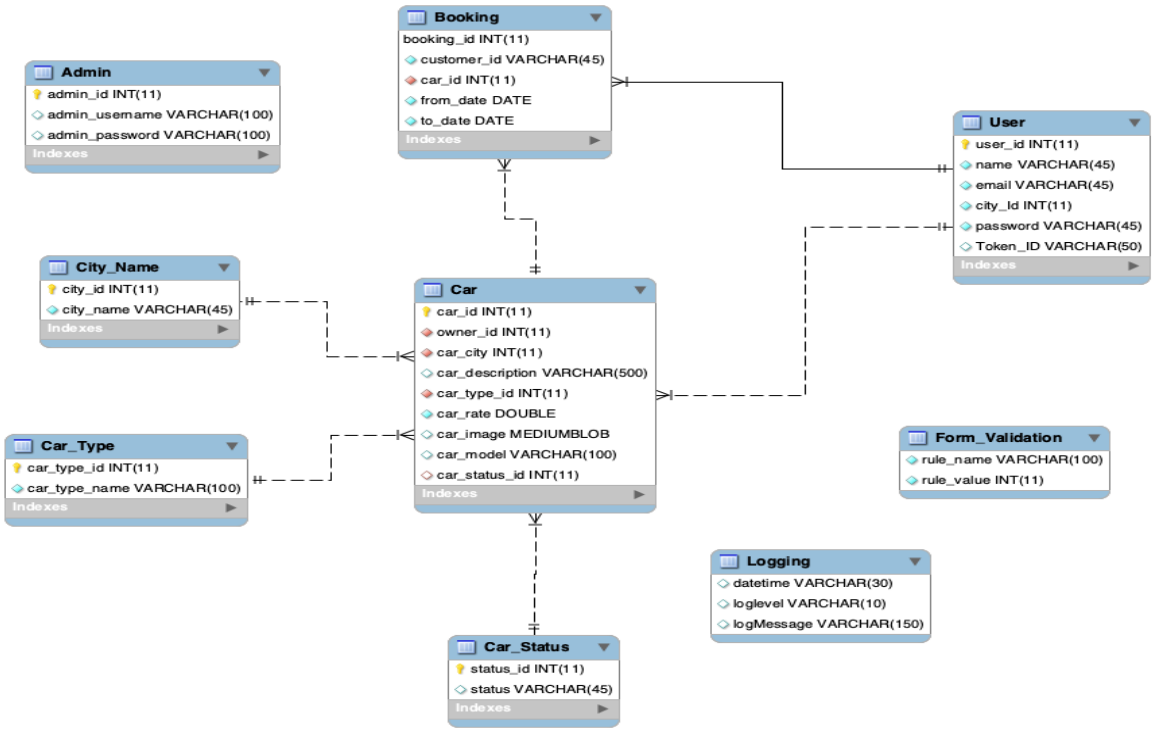
We have implemented Singleton in Authentication, Where Authentication check for user session. Instead of writing session validation code in every controller we used singleton ensure that to have one instance for a class in memory and no redundant code everywhere.

## Naming Strategy/Convention

- In our project, we used Camel Case for naming strategy. We used “Upper Camel Case” for all java class naming convention, where first letter of each word is capitalized. We used “Lower Camel Case” naming convention for methods, jsp files and stored procedure names.

| Identifier Type | Naming style | Example |
|---|---|---|
| Classes | We used upper camel case | class UserSignUpController<br>class LoginService |
| Interfaces | For interface, we followed “IInterfaceName” naming convention style | interface ILoginService<br>interface ILoginDAO |
| Methods | For methods we used lower camel case style. | void updateCarStatus();<br>void saveUserSignUpDetails(); |
| Variables | For variables we used Lower Camel Case naming convention style. | private String userName;<br>private String errorMessage; |

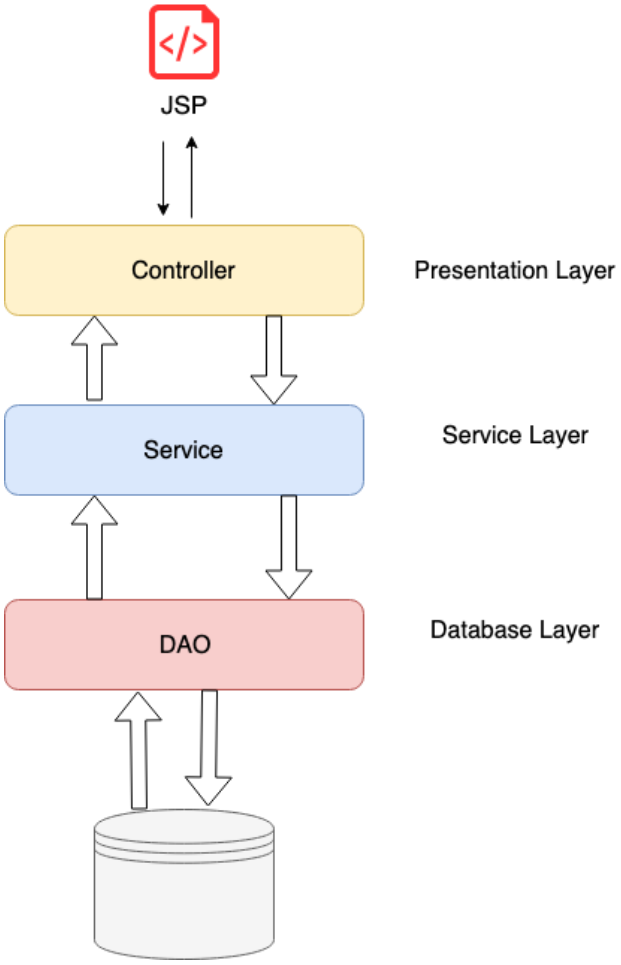
## Database Schema:




## MVC Architecture:

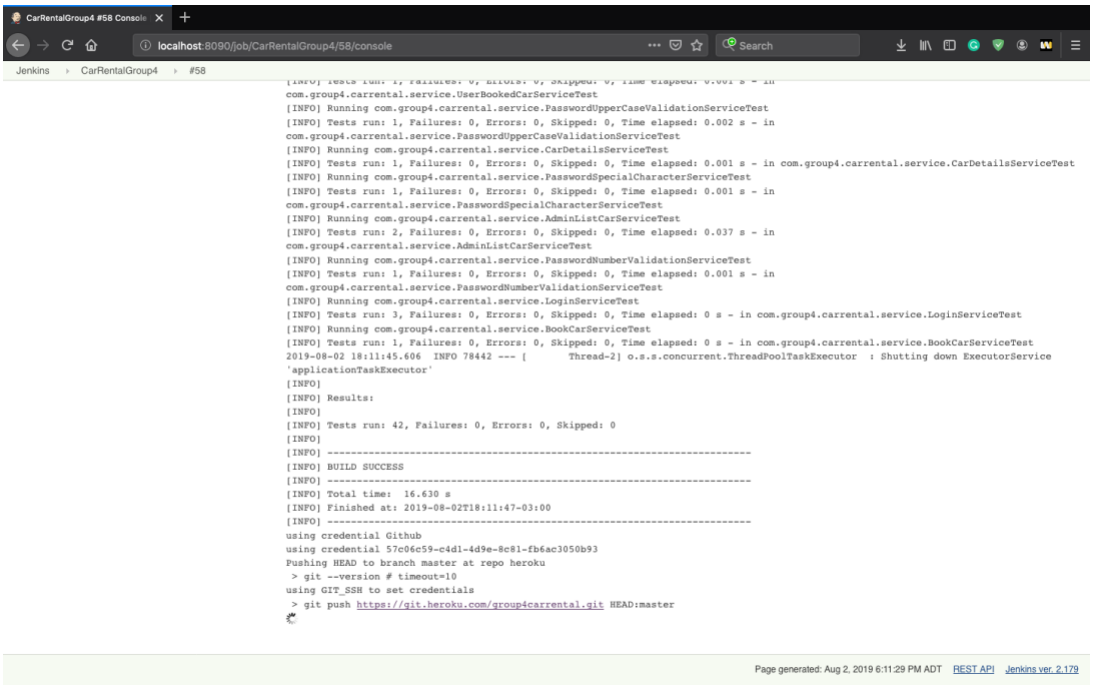
- We followed MVC (Model, View, Controller) layered architecture (JSP -> controller -> service -> DAO ->Model).

    a. Model – Model maintains the application data[2].
    b. View – View represent the User Interface and presentation of the application. In our project, JSP pages create the view of the application [2].
    c. Controller – Controller handles the user request. It controls the data flow and acts between the Model and the View [2].




## Testing

For testing we have used Junit and Mockito. We have written 42 test cases and all the test cases are passing.




# Refactoring

**Refactoring** is a technique for restructuring an existing code and altering its internal structure without altering its external behavior. Refactoring reduce the code bug and it makes code more readable. We have used following refactoring techniques to remove the bad smell.

1. **Preserve Whole Object**

   In the signUpFormValidation method of UserSignUpService.java. We passed whole object of user instead of long parameter list in argument.

2. **Extract Method**

   We put validation redundant code into single service class and using it wherever required

# Technical Debt

We had hard coded the text in the separate Java files created for our project. Instead, we could've created a text file containing all the hardcoded texts.

# Member's Contribution

## Theja Manasa Thatiparti (B00813459)

| | |
|---|---|
| Controller (.java) | • LoginController<br>• LogoutController<br>• AdminApproveController<br>• ForgotPasswordController |
| Service Interfaces (.java) | • ILoginService<br>• IAdminCarApproveService<br>• IEmailApproveReject<br>• IForgotPasswordService |
| Service Implementation (.java) | • LoginService<br>• AdminCarApproveService<br>• EmailApproveReject<br>• ErrorLogger<br>• InfoLogger<br>• WarningLogger<br>• LoggerInstance<br>• LoggerService<br>• ForgotPasswordService |
| DAO Interfaces (.java) | • ILoginDAO<br>• IAdminResponseDAO<br>• IForgotPasswordDAO<br>• ILoggerDAO |
| DAO Implementation (.java) | • LoggerDAO<br>• LoginDAO<br>• AdminResponseDAO<br>• ForgotPasswordDAO |
| JSP pages (.jsp) | • adminPendingRequests<br>• forgotPassword<br>• login<br>• resetPassword |
| Service Test | • LoginServiceTest |

| | |
|---|---|
| | • ForgotPasswordServiceTest |
| Mock DAO | • LoginDAOMock<br>• ForgotPasswordDAOMock |
| Stored Procedures | • getPassword<br>• getUserID<br>• carApproval<br>• carReject<br>• getAllPendingRequests<br>• logging<br>• validateEmail<br>• addToken<br>• getToken<br>• updatePassword<br>• validateToken |
| Design Pattern | • LoggerInstance (Singleton)<br>• Logging (Chain of responsibility) |

## Mohit Kacha (B00804879)

| | |
|---|---|
| Controller (.java) | • CarDetailsController<br>• CarEditController<br>• UpdatePasswordController<br>• UserBookedCarController<br>• UserListedCarController |
| Service Interfaces (.java) | • IUserListedCarService<br>• IUserBookedCarService<br>• IUpdatePasswordService<br>• ICarEditService<br>• ICarDetailsService |
| Service Implementation (.java) | • UserListedCarService<br>• UserBookedCarService<br>• UpdatePasswordService<br>• CarEditService<br>• CarDetailsService |
| DAO Interfaces (.java) | • IUserListedCarsDAO<br>• IUserBookedCarsDAO<br>• IUpdatePasswordDAO<br>• ICarEditDAO<br>• ICarDetailsDAO |
| DAO Implementation (.java) | • UserListedCarsDAO<br>• UserBookedCarsDAO<br>• UpdatePasswordDAO<br>• CarEditDAO<br>• CarDetailsDAO |
| JSP pages (.jsp) | • userListedCar<br>• userBookedCar<br>• UpdatePassword<br>• carDetails<br>• carEdit |
| Service Test | • UserListedCarServiceTest<br>• UserBookedCarServiceTest<br>• UpdatePasswordServiceTest<br>• CarDetailsServiceTest |
| Mock DAO | • UserListedCarDAOMock |

| | |
|---|---|
| | • UserBookedCarDAOMock<br>• UpdatePasswordDAOMock<br>• CarDetailsDAOMock |
| Stored Procedures | • getListedCar<br>• removeCarById<br>• getBookedCarDetails<br>• getBookedCars<br>• removeBookedCarById<br>• getUserPassword<br>• updatePasswordQuery<br>• updateCarQuery<br>• updateCarImage |
| Design Pattern | • Authentication (Singleton)<br>• IAuthentication (Singleton) |

## Bhumi Patel (B00824756)

| | |
|---|---|
| Controller (.java) | • AdminBlacklistCarsController<br>• PaymentController<br>• UserSignUpController |
| Service Interfaces (.java) | • IAdminBlacklistCarsService<br>• IBookCarService<br>• IPasswordValidationService<br>• IPaymentValidationService<br>• ISendMailService<br>• ISignFormRuleService<br>• IUserSignUpService |
| Service Implementation (.java) | • AdminBlacklistCarsService<br>• BookCarService<br>• PasswordLengthValidationService<br>• PasswordLowerCaseValidationService<br>• PasswordNumberValidationService<br>• PasswordSpecialCharacterService<br>• PasswordUpperCaseValidationService<br>• PaymentValidationService<br>• SendMailService<br>• SignUpformRuleService<br>• UserSignUpService |
| DAO Interfaces (.java) | • IAdminBlacklistCarsDAO<br>• IBookCarDAO<br>• IUserSignUpDAO |
| DAO Implementation (.java) | • AdminBlacklistCarsDAO<br>• UserSignUpDAO<br>• BookCarDAO |
| JSP pages (.jsp) | • blackListedCars.jsp<br>• error.jsp<br>• error404.jsp<br>• error500.jsp<br>• paymentPage.jsp<br>• userSignUp.jsp<br>• userUpdateProfile.jsp |
| Service Test | • AdminBlacklistCarsServiceTest<br>• BookCarServiceTest |

| | |
|---|---|
| | • PasswordLengthValidationServiceTest<br>• PasswordLowerCaseValidationServiceTest<br>• PasswordNumberValidationServiceTest<br>• PasswordSpecialCharacterServiceTest<br>• PasswordUpperCaseValidationServiceTest<br>• PaymentValidationServiceTest<br>• SignUpFormRuleServiceTest<br>• UserSignUpServiceTest |
| Mock DAO | • AdminBlacklistCarsDAOMock<br>• BookCarDAOMock<br>• SignUpFormRuleDAOMock<br>• UserSignUpDAOMock |
| Stored Procedures | • saveUserSignUpDetails<br>• getUserDetails<br>• updateUserProfileDetails<br>• isEmailExist<br>• getCityList<br>• getBlacklistCars<br>• UpdateCarStatus<br>• SaveCarBookingDetails |
| Design Pattern | • I implemented observer design pattern for email sending service, but as suggested by Rob during presentation observer pattern for web application is not appropriate.<br>• I removed observer pattern. You can find code with observer pattern on (feature/resolvedIssues_bhumi) branch on github. |

## Anant Pillai (B00826642)

| | |
|---|---|
| Controller (.java) | • AdminListCarController<br>• AdminLoginController<br>• CarRentController<br>• HomeController |
| Service Interfaces (.java) | • IAdminListCarService<br>• IAdminLoginService<br>• ICarRentService<br>• IHomeService |
| Service Implementation (.java) | • AdminListCarService<br>• AdminLoginService<br>• CarRentService<br>• HomeService |
| DAO Interfaces (.java) | • IAdminListCarDAO<br>• IAdminLoginDAO<br>• ICarRentDAO<br>• IHomeDAO |
| DAO Implementation (.java) | • AdminListCarDAO<br>• AdminLoginDAO<br>• CarRentDAO<br>• HomeDAO |
| JSP pages (.jsp) | • adminListAllCar<br>• adminLogin<br>• carrent<br>• homePage |
| Service Test | • AdminListCarServiceTest<br>• AdminLoginServiceTest<br>• CarRentServiceTest<br>• HomeServiceTest |
| Mock DAO | • AdminListCarDAOMock<br>• AdminLoginDAOMock<br>• CarRentDAOMock<br>• HomeDAOMock |
| Stored Procedures | • listAllCarsAdmin<br>• blackListCar<br>• getOwnerEmail |

| | |
|---|---|
| | • validateAdmin<br>• getCarType<br>• addCar<br>• getCarById<br>• getAvailableCar<br>• getCurrentBooking |
| Design Pattern | • DatabaseConnection (Singleton)<br>• IDatabaseConnection (Singleton) |
| Other | • Config |

# Reference

[1]"fengyuanchen/datepicker", *GitHub*, 2019. [Online]. Available: https://github.com/fengyuanchen/datepicker. [Accessed: 25- Jul- 2019].

[2] "MVC Architecture", *Tutorialsteacher.com*, 2019. [Online]. Available: https://www.tutorialsteacher.com/mvc/mvc-architecture. [Accessed: 02- Aug- 2019]”

[3]"Introduction", *Getbootstrap.com*, 2019. [Online]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/