



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:

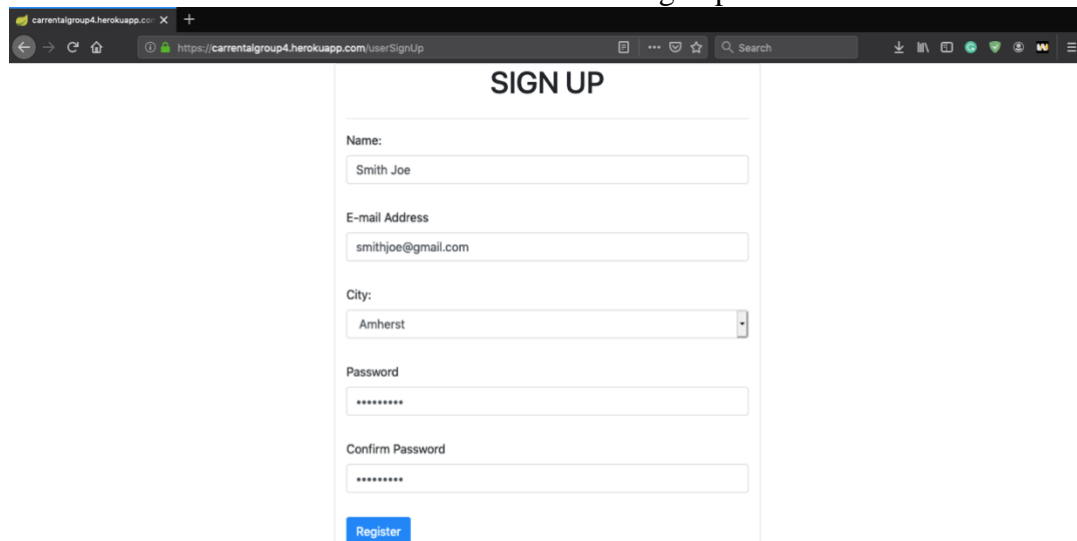
Username: admin

Password: adminPassword

Features

- Registration

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



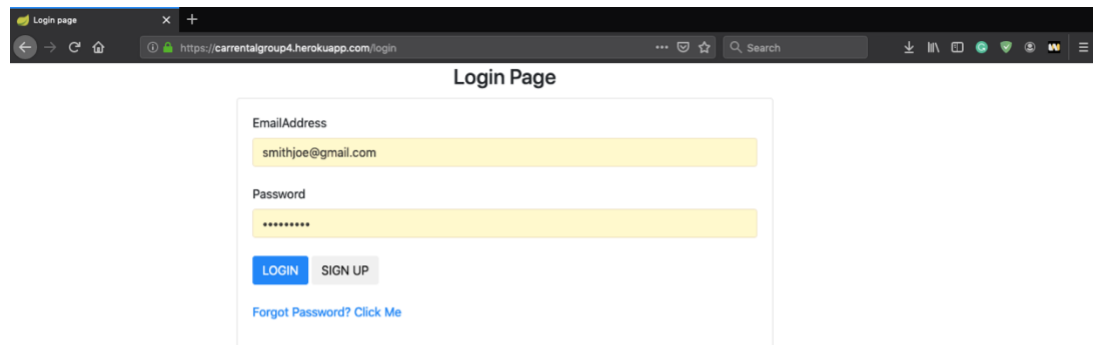
The screenshot shows a web browser window with the URL <https://carrentalgroup4.herokuapp.com/userSignUp>. The page displays a "SIGN UP" form with the following fields:

- Name:
- E-mail Address:
- City:
- Password:
- Confirm Password:

A blue "Register" button is located at the bottom of the form.

- Login

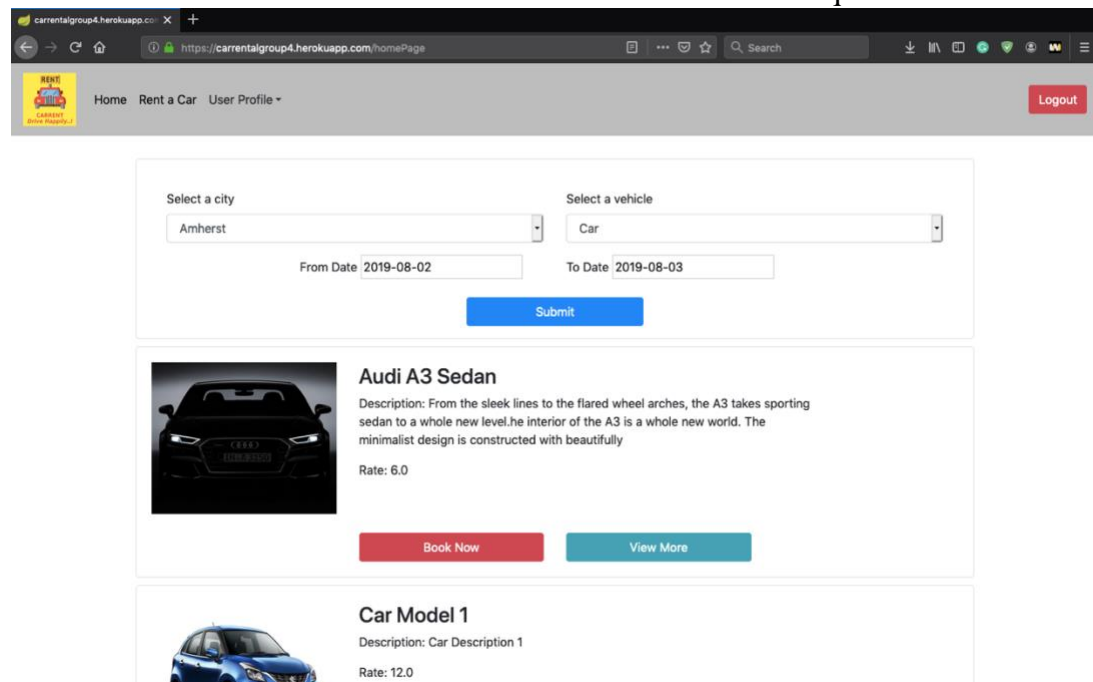
Once the user is registered the user can login and use the CarRental website.



The screenshot shows a web browser window with the URL `https://carrentalgroup4.herokuapp.com/login`. The page is titled "Login Page". It features a form with two input fields: "EmailAddress" containing "smithjoe@gmail.com" and "Password" containing "*****". Below the password field are two buttons: "LOGIN" (blue) and "SIGN UP" (grey). At the bottom of the form is a link that says "Forgot Password? Click Me".

- Search for Cars

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



The screenshot shows a web browser window with the URL `https://carrentalgroup4.herokuapp.com/homePage`. The page has a navigation bar with a logo, "Home", "Rent a Car", "User Profile", and a "Logout" button. The main content area contains a search form with the following fields: "Select a city" (dropdown menu showing "Amherst"), "Select a vehicle" (dropdown menu showing "Car"), "From Date" (text input showing "2019-08-02"), and "To Date" (text input showing "2019-08-03"). A "Submit" button is below the date fields. Below the search form, there are two car listings. The first listing is for an "Audi A3 Sedan" with a description: "From the sleek lines to the flared wheel arches, the A3 takes sporting sedan to a whole new level. The interior of the A3 is a whole new world. The minimalist design is constructed with beautifully". The rate is "6.0". There are "Book Now" and "View More" buttons. The second listing is for "Car Model 1" with a description: "Car Description 1". The rate is "12.0".

- Rent your car

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

The screenshot shows a web browser window with the URL `https://carrentalgroup4.herokuapp.com/carrent`. The page has a navigation bar with links for Home, Rent a Car, and User Profile, and a Logout button. The main form is titled 'Car Model' and contains the following fields:

- Car Model:
- Description:
- Select City:
- Select Car Type:
- Car Rate:
- Example file input: zie4zb.png

A blue Submit button is located at the bottom of the form.

- Accept/Reject Car Request (Admin)

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

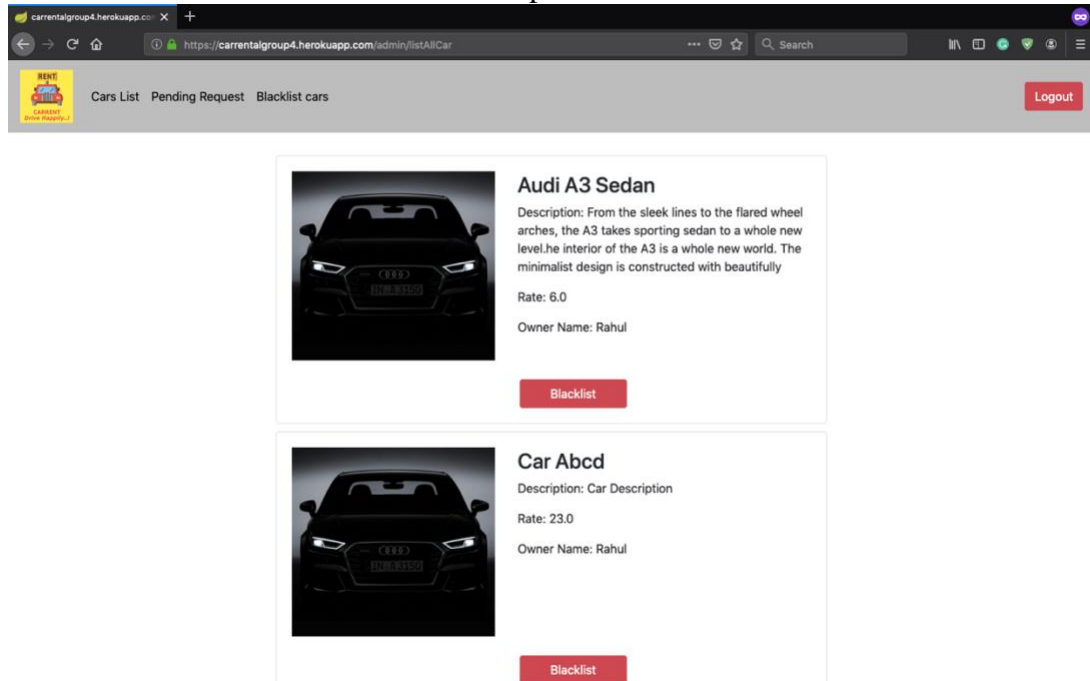
The screenshot shows a web browser window with the URL `https://carrentalgroup4.herokuapp.com/admin/ListPendingRequests`. The page has a navigation bar with links for Cars List, Pending Request, and Blacklist cars, and a Logout button. The main content area is titled 'PendingCarRequestsList' and displays the following information:

- Model: Car Model
- Description: Car Description
- Rate: 12.0
- Owner Name: Smith Joe
- Mail: smithjoe@gmail.com

At the bottom of the information box are two buttons: APPROVE (blue) and REJECT (grey).

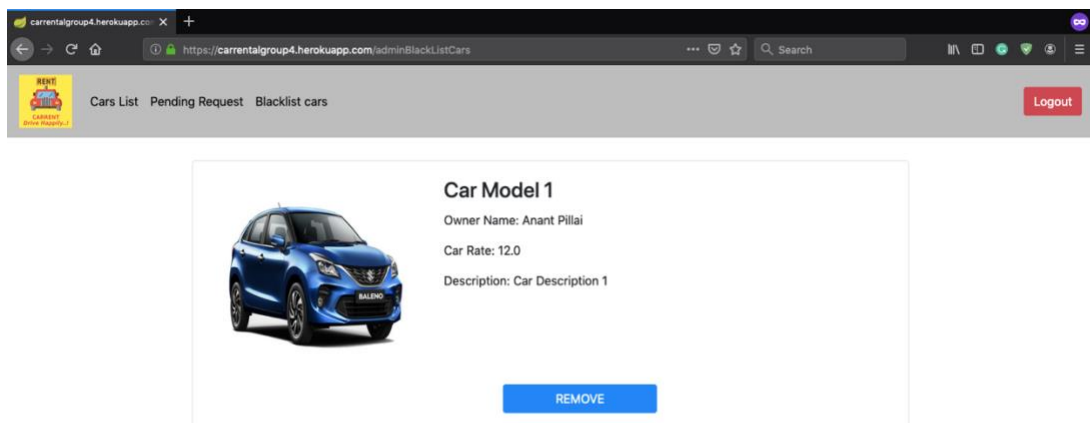
- **List All Cars (Admin)**

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)

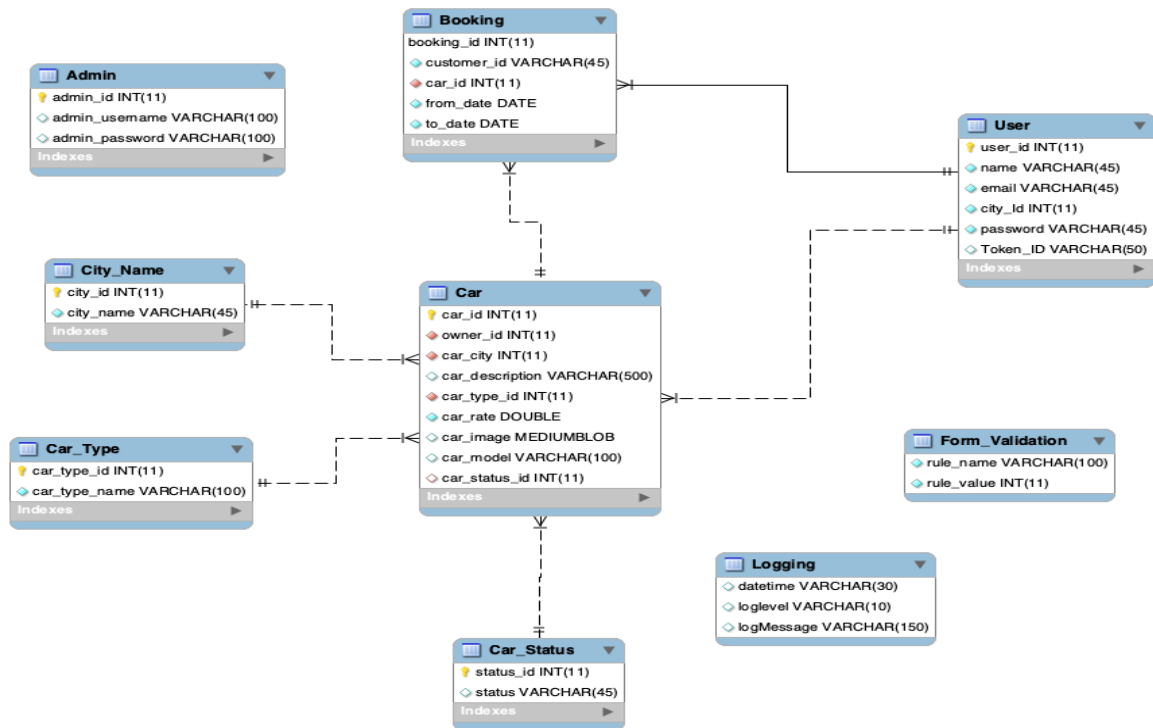
We have implemented Singleton in Authentication, Where Authentication check for user session. Instead of writing session validation code in every controller we used singleton ensure that to have one instance for a class in memory and no redundant code everywhere.

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.

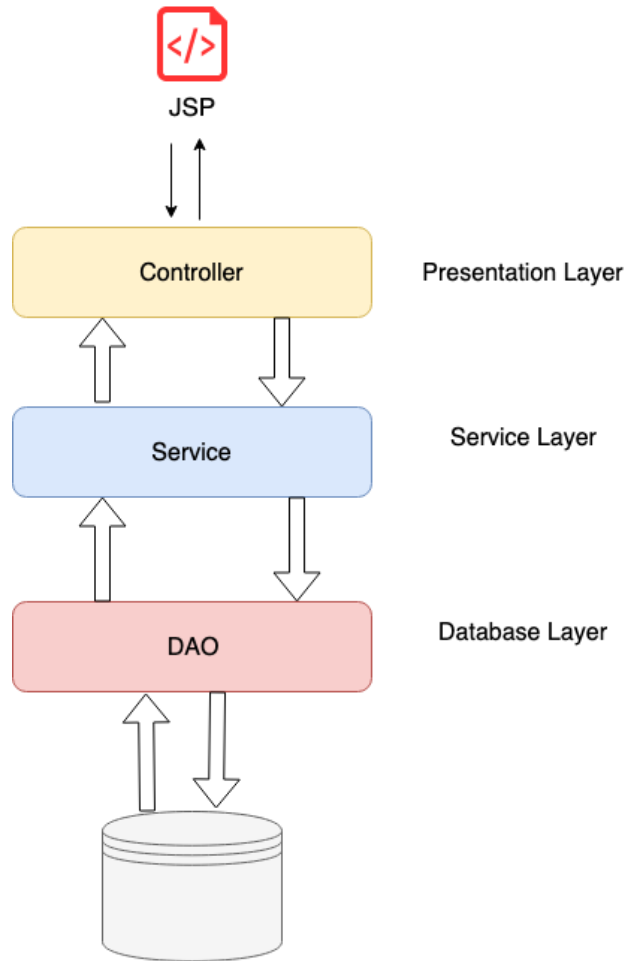
Identifier Type	Naming style	Example
Classes	We used upper camel case	class UserSignUpController class LoginService
Interfaces	For interface, we followed “InterfaceName” naming convention style	interface ILoginService interface ILoginDAO
Methods	For methods we used lower camel case style.	void updateCarStatus();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

Database Schema:



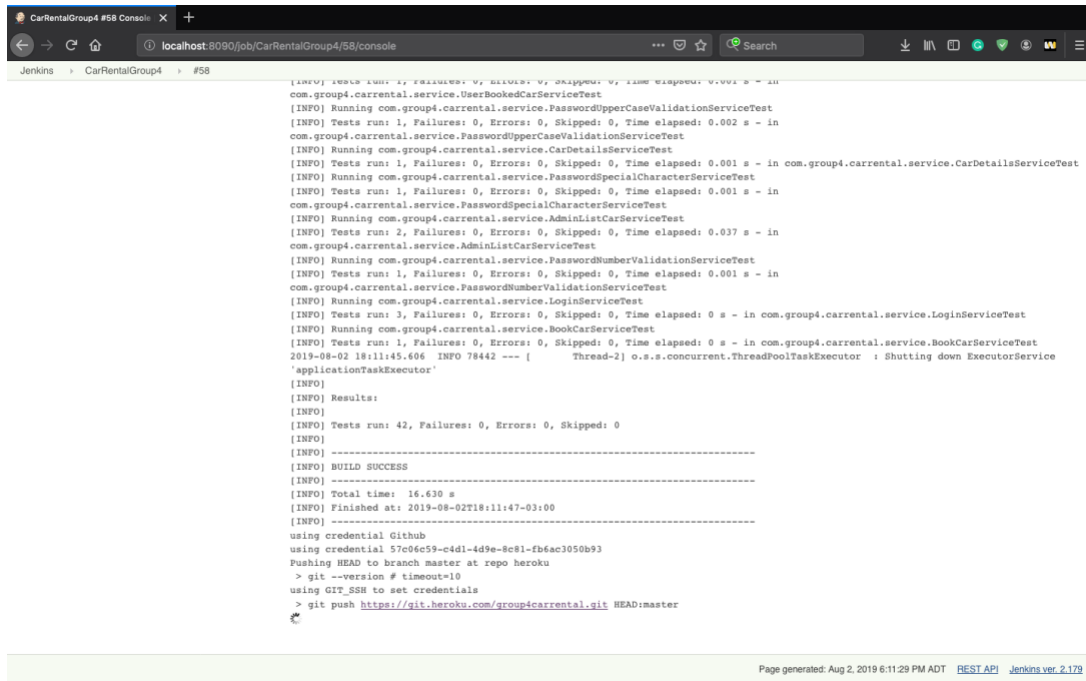
MVC Architecture:

- 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.



```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in com.group4.carrental.service.PasswordUpperCaseValidationServiceTest
[INFO] Running com.group4.carrental.service.PasswordUpperCaseValidationServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.group4.carrental.service.CarDetailsServiceTest
[INFO] Running com.group4.carrental.service.PasswordSpecialCharacterServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.group4.carrental.service.PasswordSpecialCharacterServiceTest
[INFO] Running com.group4.carrental.service.AdminListCarServiceTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 s - in com.group4.carrental.service.AdminListCarServiceTest
[INFO] Running com.group4.carrental.service.PasswordNumberValidationServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.group4.carrental.service.PasswordNumberValidationServiceTest
[INFO] Running com.group4.carrental.service.LoginServiceTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in com.group4.carrental.service.LoginServiceTest
[INFO] Running com.group4.carrental.service.BookCarServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in com.group4.carrental.service.BookCarServiceTest
2019-08-02 18:11:45.606 [INFO] 78442 --- [ Thread-2] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 42, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.630 s
[INFO] Finished at: 2019-08-02T18:11:47-03:00
[INFO] -----
using credential Github
using credential 57c06e59-c4d1-4d9e-8c81-fb6ac3050b93
Pushing HEAD to branch master at repo heroku
> git --version # timeout=10
using GIT_SSH to set credentials
> git push https://git.heroku.com/group4carrental.git HEAD:master
[INFO]
```

Page generated: Aug 2, 2019 6:11:29 PM ADT [REST API](#) [Jenkins ver. 2.179](#)

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)	<ul style="list-style-type: none">• LoginController• LogoutController• AdminApproveController• ForgotPasswordController
Service Interfaces (.java)	<ul style="list-style-type: none">• ILoginService• IAdminCarApproveService• IEmailApproveReject• IForgotPasswordService
Service Implementation (.java)	<ul style="list-style-type: none">• LoginService• AdminCarApproveService• EmailApproveReject• ErrorLogger• InfoLogger• WarningLogger• LoggerInstance• LoggerService• ForgotPasswordService
DAO Interfaces (.java)	<ul style="list-style-type: none">• ILoginDAO• IAdminResponseDAO• IForgotPasswordDAO• ILoggerDAO
DAO Implementation (.java)	<ul style="list-style-type: none">• LoggerDAO• LoginDAO• AdminResponseDAO• ForgotPasswordDAO
JSP pages (.jsp)	<ul style="list-style-type: none">• adminPendingRequests• forgotPassword• login• resetPassword
Service Test	<ul style="list-style-type: none">• LoginServiceTest

	<ul style="list-style-type: none"> ForgotPasswordServiceTest
Mock DAO	<ul style="list-style-type: none"> LoginDAOMock ForgotPasswordDAOMock
Stored Procedures	<ul style="list-style-type: none"> getPassword getUserID carApproval carReject getAllPendingRequests logging validateEmail addToken getToken updatePassword validateToken
Design Pattern	<ul style="list-style-type: none"> LoggerInstance (Singleton) Logging (Chain of responsibility)

Mohit Kacha (B00804879)

Controller (.java)	<ul style="list-style-type: none">• CarDetailsController• CarEditController• UpdatePasswordController• UserBookedCarController• UserListedCarController
Service Interfaces (.java)	<ul style="list-style-type: none">• IUserListedCarService• IUserBookedCarService• IUpdatePasswordService• ICarEditService• ICarDetailsService
Service Implementation (.java)	<ul style="list-style-type: none">• UserListedCarService• UserBookedCarService• UpdatePasswordService• CarEditService• CarDetailsService
DAO Interfaces (.java)	<ul style="list-style-type: none">• IUserListedCarsDAO• IUserBookedCarsDAO• IUpdatePasswordDAO• ICarEditDAO• ICarDetailsDAO
DAO Implementation (.java)	<ul style="list-style-type: none">• UserListedCarsDAO• UserBookedCarsDAO• UpdatePasswordDAO• CarEditDAO• CarDetailsDAO
JSP pages (.jsp)	<ul style="list-style-type: none">• userListedCar• userBookedCar• UpdatePassword• carDetails• carEdit
Service Test	<ul style="list-style-type: none">• UserListedCarServiceTest• UserBookedCarServiceTest• UpdatePasswordServiceTest• CarDetailsServiceTest
Mock DAO	<ul style="list-style-type: none">• UserListedCarDAOMock

	<ul style="list-style-type: none"> • UserBookedCarDAOMock • UpdatePasswordDAOMock • CarDetailsDAOMock
Stored Procedures	<ul style="list-style-type: none"> • getListedCar • removeCarById • getBookedCarDetails • getBookedCars • removeBookedCarById • getUserPassword • updatePasswordQuery • updateCarQuery • updateCarImage
Design Pattern	<ul style="list-style-type: none"> • Authentication (Singleton) • IAuthentication (Singleton)

Bhumi Patel (B00824756)

Controller (.java)	<ul style="list-style-type: none">• AdminBlacklistCarsController• PaymentController• UserSignUpController
Service Interfaces (.java)	<ul style="list-style-type: none">• IAdminBlacklistCarsService• IBookCarService• IPasswordValidationService• IPaymentValidationService• ISendMailService• ISignFormRuleService• IUserSignUpService
Service Implementation (.java)	<ul style="list-style-type: none">• AdminBlacklistCarsService• BookCarService• PasswordLengthValidationService• PasswordLowerCaseValidationService• PasswordNumberValidationService• PasswordSpecialCharacterService• PasswordUpperCaseValidationService• PaymentValidationService• SendMailService• SignUpformRuleService• UserSignUpService
DAO Interfaces (.java)	<ul style="list-style-type: none">• IAdminBlacklistCarsDAO• IBookCarDAO• IUserSignUpDAO
DAO Implementation (.java)	<ul style="list-style-type: none">• AdminBlacklistCarsDAO• UserSignUpDAO• BookCarDAO
JSP pages (.jsp)	<ul style="list-style-type: none">• blackListedCars.jsp• error.jsp• error404.jsp• error500.jsp• paymentPage.jsp• userSignUp.jsp• userUpdateProfile.jsp
Service Test	<ul style="list-style-type: none">• AdminBlacklistCarsServiceTest• BookCarServiceTest

	<ul style="list-style-type: none"> • PasswordLengthValidationServiceTest • PasswordLowerCaseValidationServiceTest • PasswordNumberValidationServiceTest • PasswordSpecialCharacterServiceTest • PasswordUpperCaseValidationServiceTest • PaymentValidationServiceTest • SignUpFormRuleServiceTest • UserSignUpServiceTest
Mock DAO	<ul style="list-style-type: none"> • AdminBlacklistCarsDAOMock • BookCarDAOMock • SignUpFormRuleDAOMock • UserSignUpDAOMock
Stored Procedures	<ul style="list-style-type: none"> • saveUserSignUpDetails • getUserDetails • updateUserProfileDetails • isEmailExist • getCityList • getBlacklistCars • UpdateCarStatus • SaveCarBookingDetails
Design Pattern	<ul style="list-style-type: none"> • I implemented observer design pattern for email sending service, but as suggested by Rob during presentation observer pattern for web application is not appropriate. • I removed observer pattern. You can find code with observer pattern on (feature/resolvedIssues_bhumi) branch on github.

Anant Pillai (B00826642)

Controller (.java)	<ul style="list-style-type: none">• AdminListCarController• AdminLoginController• CarRentController• HomeController
Service Interfaces (.java)	<ul style="list-style-type: none">• IAdminListCarService• IAdminLoginService• ICarRentService• IHomeService
Service Implementation (.java)	<ul style="list-style-type: none">• AdminListCarService• AdminLoginService• CarRentService• HomeService
DAO Interfaces (.java)	<ul style="list-style-type: none">• IAdminListCarDAO• IAdminLoginDAO• ICarRentDAO• IHomeDAO
DAO Implementation (.java)	<ul style="list-style-type: none">• AdminListCarDAO• AdminLoginDAO• CarRentDAO• HomeDAO
JSP pages (.jsp)	<ul style="list-style-type: none">• adminListAllCar• adminLogin• carrent• homePage
Service Test	<ul style="list-style-type: none">• AdminListCarServiceTest• AdminLoginServiceTest• CarRentServiceTest• HomeServiceTest
Mock DAO	<ul style="list-style-type: none">• AdminListCarDAOMock• AdminLoginDAOMock• CarRentDAOMock• HomeDAOMock
Stored Procedures	<ul style="list-style-type: none">• listAllCarsAdmin• blackListCar• getOwnerEmail

	<ul style="list-style-type: none">• validateAdmin• getCarType• addCar• getCarById• getAvailableCar• getCurrentBooking
Design Pattern	<ul style="list-style-type: none">• DatabaseConnection (Singleton)• IDatabaseConnection (Singleton)
Other	<ul style="list-style-type: none">• 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/>