PRJ301_SE1612

Group PASSED

Online Car Renting Website 1 report

1/ Case Study

OverView: Our newly established company wants to promote its brand by means of media and internet. Therefore, a special website with many functions allows customers to interact with the admin to rent cars that are suitable for their budget.

Challenge: We have the most important challenge when design a website that is seperated for Guest's interface and the other for Manager.

* Firstly, customers can only register one account to perform the following functions:

--- Search vehicle information, view vehicle details and price to rent that car.

--- Add place to cart, and also delete it

--- Modify the rental date, or any relevant details regarding the vehicle the customer needs to purchase.

--- View cart and total car rental price.

--- Confirm car rental with Customer's information.

* Next, the admin must also log in to perform the following steps:

--- Search rental car with specified information: Customer information.

--- CRUD (Create Delete Update Read) with vehicles.

--- View rental cars by member and by day.

Business Rules for Online Booking Website System

*Log in websites

1/Customer

- Customers must register as a member to log in (Type of Rule: Constrain)

- Successful login results will appear on a new page (Type of Rule: Fact)

2/Admin

-Daily website management (see how many people log on to the website that day to check rental car prices)

Type of Rule: Constrain

*Search in websites

1/ Customer

-Search Vehicle Information (Date, Price of Car Rental , car name)

Type of Rule: Constrain

- Customer's search results will show up on a new page

Type of Rule: Fact

2/Admin

- Search Vehicle Information (By car company, by year of buying a car)

Type of rule: Constrain

*Insert

-more information about the car (including the rental price, the car company and the date of importing that car every day)

Type of rule: Constrain

Type of user: Admin

*Update

-fix the ones that are available in the store, except for the vehicle ID

Type of rule: Constrain

Type of user: Admin

*Delete

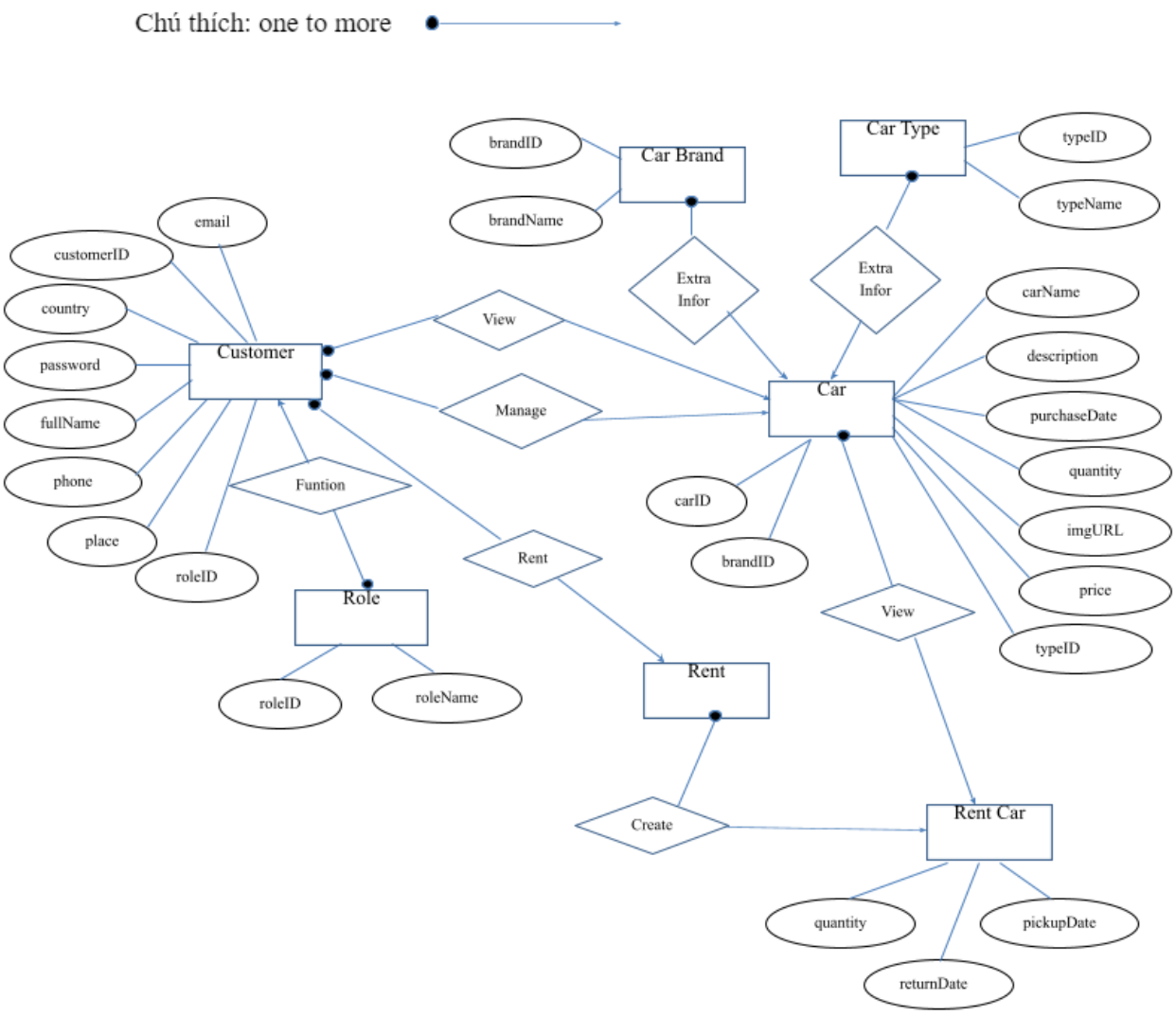
-Can delete the type of car that the customer has rented (if the customer has already rented a car, the customer cannot delete it)

Type of rule: Constrain

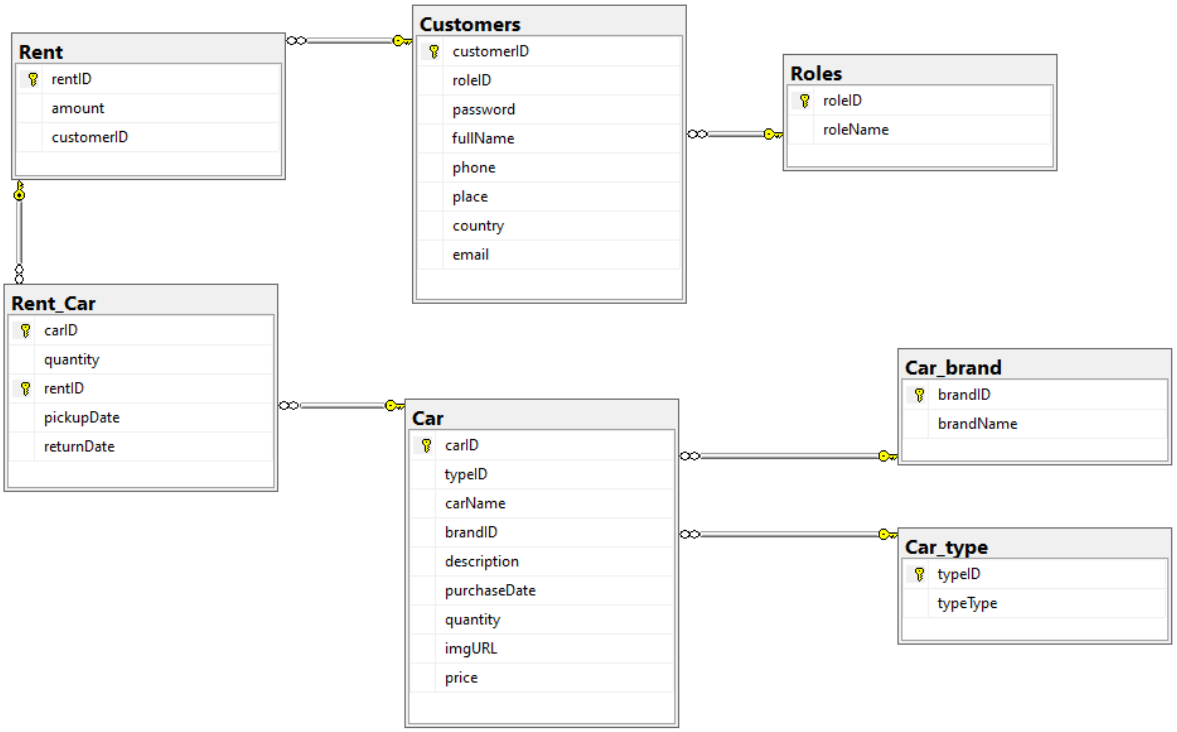
Type of user: Admin

2/A database design:

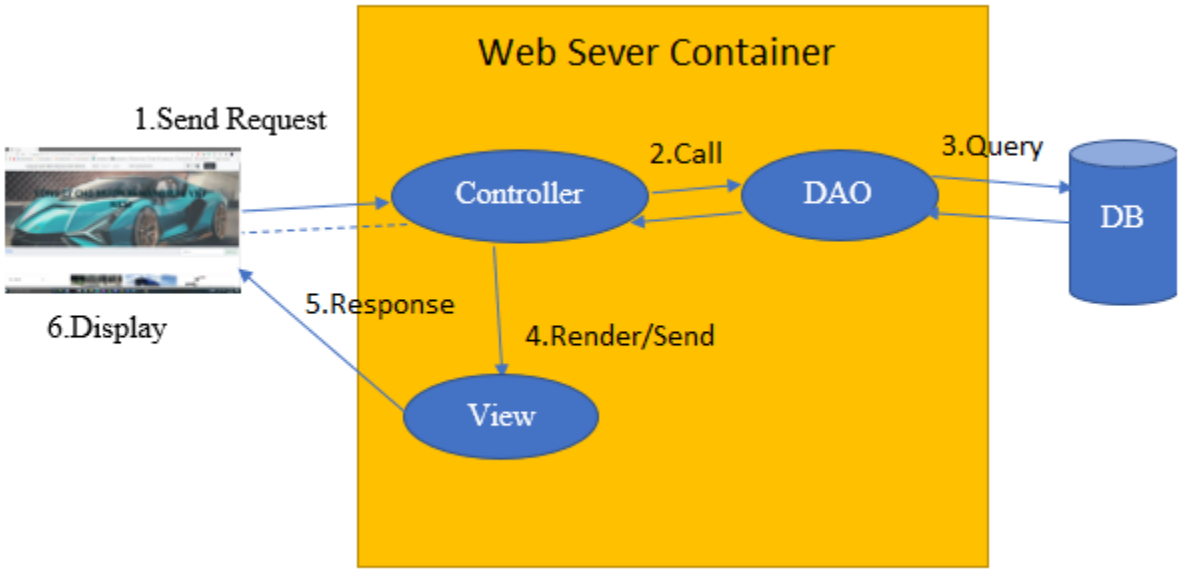
+Logical design:

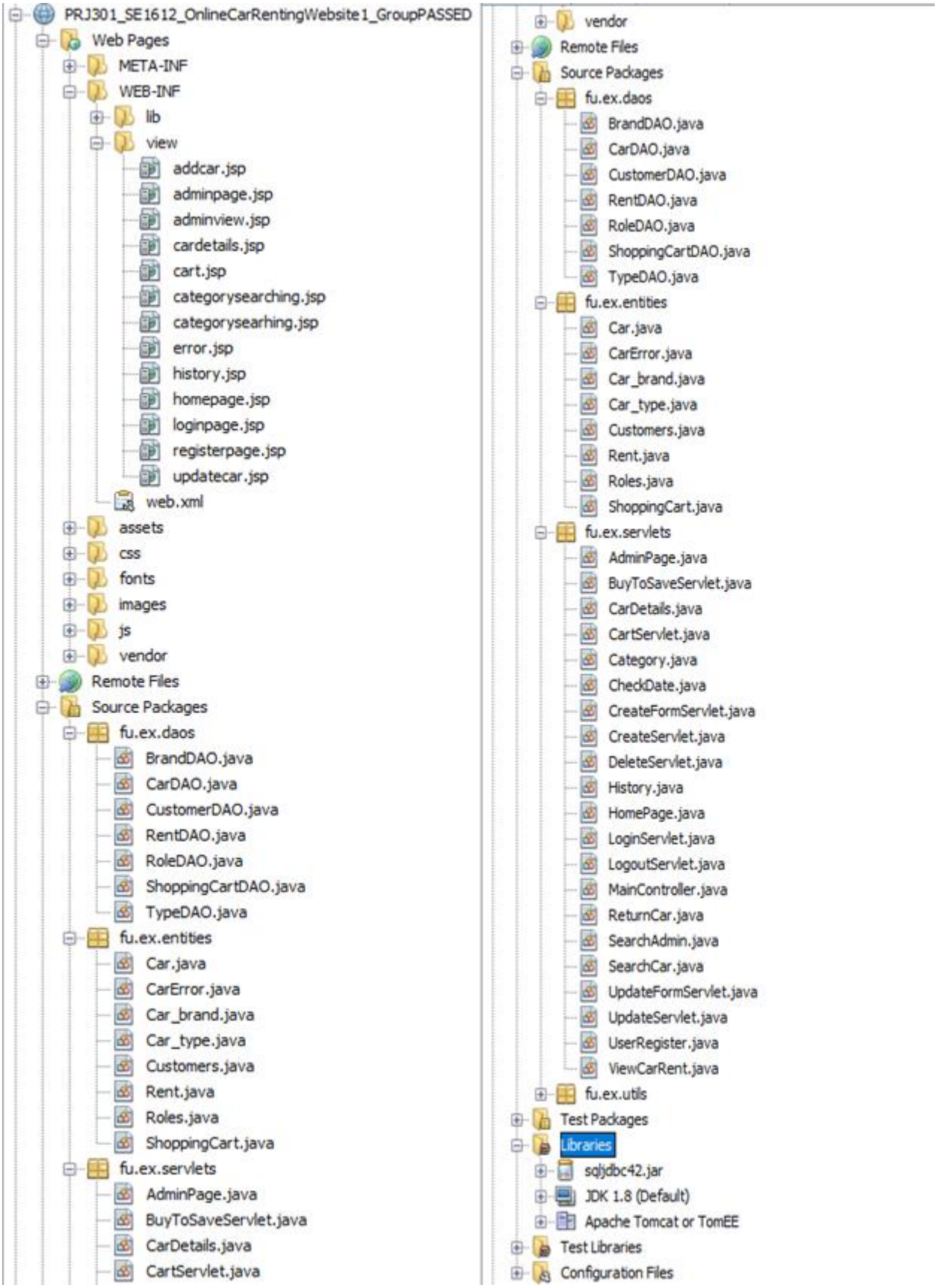


+Physical design:

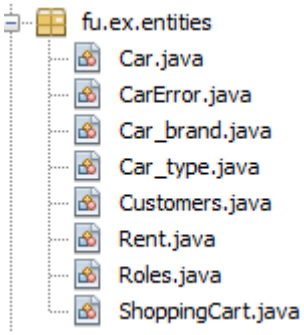


3/System design: Online Car Renting Website 1 System using MVC2. In MVC2, MVC stands for Model – View – Controller. MVC2 architecture:



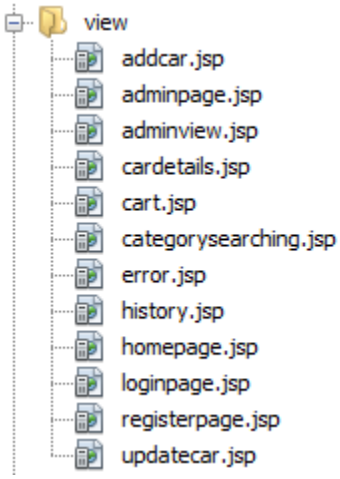

PRJ301_SE1612_OnlineCarRentingWebsite1_GroupPASSED
Web Pages
META-INF
WEB-INF
lib
view
addcar.jsp
adminpage.jsp
adminview.jsp
cardetails.jsp
cart.jsp
categorysearching.jsp
categorysearhing.jsp
error.jsp
history.jsp
homepage.jsp
loginpage.jsp
registerpage.jsp
updatecar.jsp
web.xml
assets
css
fonts
images
js
vendor
Remote Files
Source Packages
fu.ex.daos
BrandDAO.java
CarDAO.java
CustomerDAO.java
RentDAO.java
RoleDAO.java
ShoppingCartDAO.java
TypeDAO.java
fu.ex.entities
Car.java
CarError.java
Car_brand.java
Car_type.java
Customers.java
Rent.java
Roles.java
ShoppingCart.java
fu.ex.servlets
AdminPage.java
BuyToSaveServlet.java
CarDetails.java
CartServlet.java
vendor
Remote Files
Source Packages
fu.ex.daos
BrandDAO.java
CarDAO.java
CustomerDAO.java
RentDAO.java
RoleDAO.java
ShoppingCartDAO.java
TypeDAO.java
fu.ex.entities
Car.java
CarError.java
Car_brand.java
Car_type.java
Customers.java
Rent.java
Roles.java
ShoppingCart.java
fu.ex.servlets
AdminPage.java
BuyToSaveServlet.java
CarDetails.java
CartServlet.java
Category.java
CheckDate.java
CreateFormServlet.java
CreateServlet.java
DeleteServlet.java
History.java
HomePage.java
LoginServlet.java
LogoutServlet.java
MainController.java
ReturnCar.java
SearchAdmin.java
SearchCar.java
UpdateFormServlet.java
UpdateServlet.java
UserRegister.java
ViewCarRent.java
fu.ex.utils
Test Packages
Libraries
sqljdbc42.jar
JDK 1.8 (Default)
Apache Tomcat or TomEE
Test Libraries
Configuration Files

Model contains these classes:



Classes	Description
Car	Contain attribute and present the table “Car” in DB
CarError	Contain attribute for catch error when create or update car
Car_band	Contain attribute and present the table “Car_brand” in DB
Car_type	Contain attribute and present the table “Car_type” in DB
Customers	Contain attribute and present the table “Customers” in DB
Rent	Contain attribute and present the table “Rent” in DB
Roles	Contain attribute and present the table “Roles” in DB
ShoppingCart	Contain attribute and present the table “Rent_Car” in DB

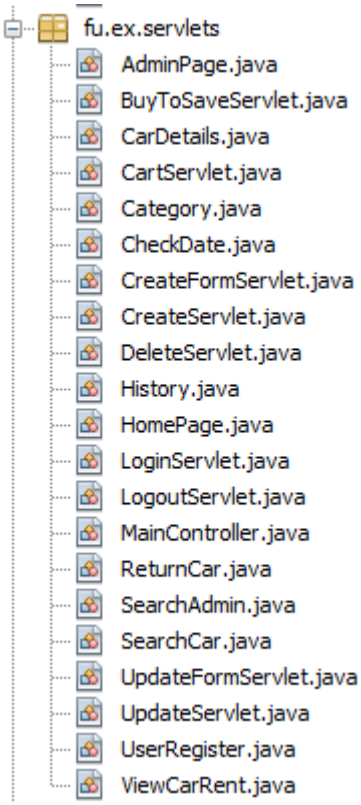
View contains jsp file:



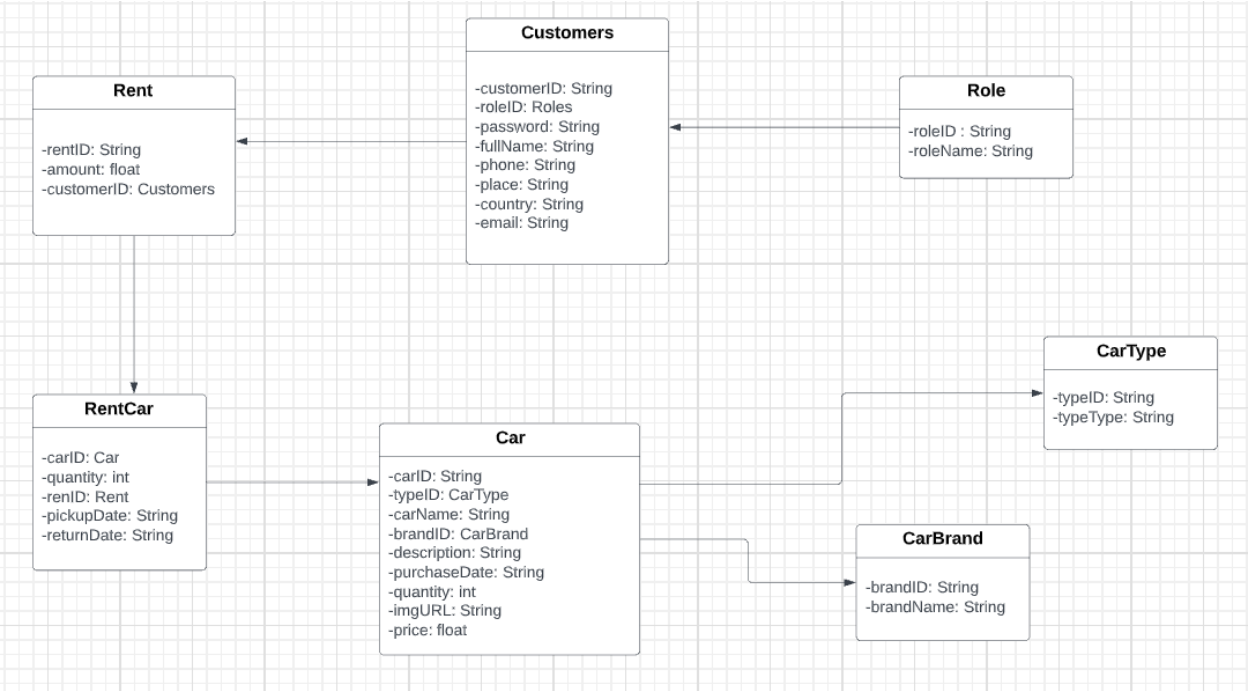
File JSP	Description
addcar	The form for manager to create a new car

adminpage	The page for manager to manage all car of the shop, they can add, update, delete.
adminview	The page for manager to search car, view rent history by user, by date
cardetails	The page for guests, users to view the detail of the car
cart	The page for user to view their cart
categorysearching	The page for guests, users to search car by type, brand or name
error	The page error
history	The page for users to view their history rent
homepage	Homepage and show all car
loginpage	The form for users and manager to login their account
registerpage	The form for guests who want to create new account
updatecar	The page for manager to update car

Controllers using Servlet to handle the request obtained from the webserver, process the request (GET or POST) then send a response back to the webserver to display.



-Class diagrams:



-Coding convention such as:

.Variable rule: using camelCase

.Method name rule: using camelCase

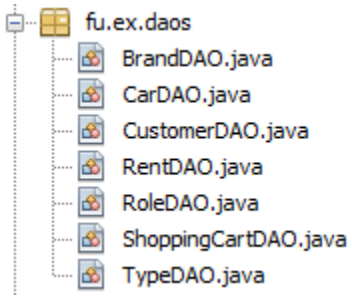
.Name class java rule: upercase first letter like: AdminPage

.Name jsp file rule: lowercase all letter like : adminpage.jsp

- Method in the model: contain 2 types of the constructor (No-Arg Constructor. Parameterized Constructor), full getter, and setter corresponding with the variables:

```
14  public class Customers  implements Serializable{
15      private String customerID;
16      private Roles role;
17      private String password;
18      private String fullname;
19      private String phone;
20      private String place;
21      private String country;
22      private String email;
23
24      public Customers(String customerID, Roles role, String password, String fullname, String phone, String place,
34
35      public String getCustomerID() {...3 lines }
38
39      public void setCustomerID(String customerID) {...3 lines }
42
43      public Roles getRole() {...3 lines }
46
47      public void setRole(Roles role) {...3 lines }
50
51      public String getPassword() {...3 lines }
54
55      public void setPassword(String password) {...3 lines }
58
59      public String getFullname() {...3 lines }
62
63      public void setFullname(String fullname) {...3 lines }
66
67      public String getPhone() {...3 lines }
70
71      public void setPhone(String phone) {...3 lines }
74
75      public String getPlace() {...3 lines }
78
79      public void setPlace(String place) {...3 lines }
82
83      public String getCountry() {...3 lines }
86
87      public void setCountry(String country) {...3 lines }
90
91      public String getEmail() {...3 lines }
94
95      public void setEmail(String email) {...3 lines }
98
99  }
```

Class DAO is a structure to help interact between JDBC and the code in project.



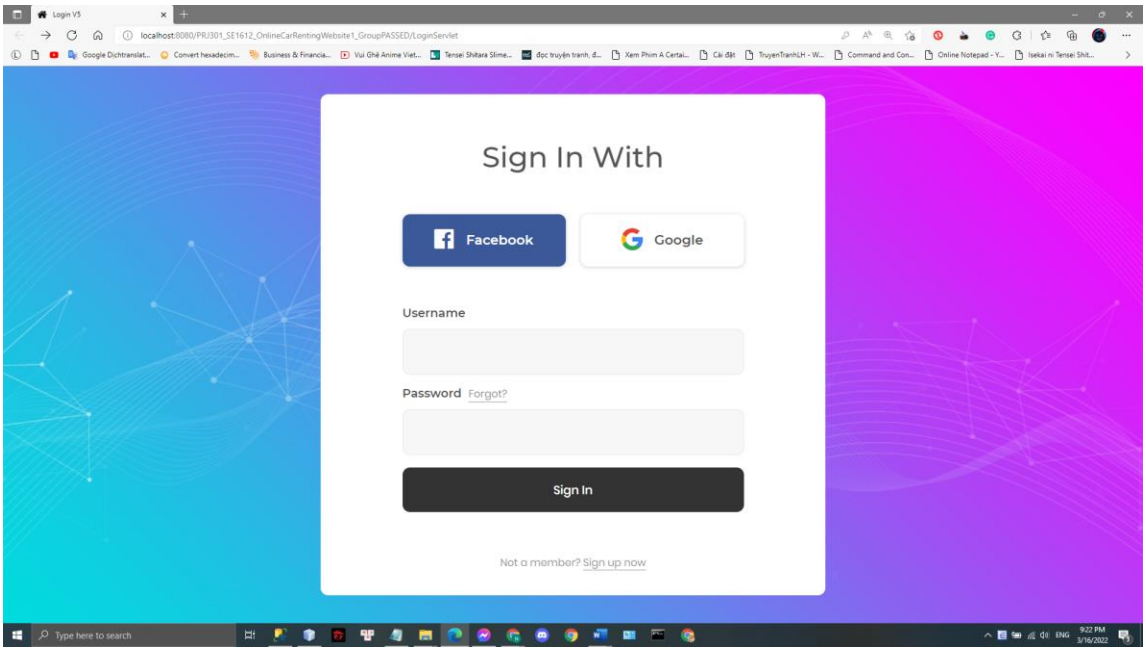
JSP using expression language to get data and view the page.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="container px-4 px-lg-5">
        <a class="navbar-brand" href="HomePage">Công ty trách nhiệm hữu hạn một mình tôi</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedC
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0 ms-lg-4">
                <li class="nav-item"><a class="nav-link active" aria-current="page" href="HomePage">Home</a></li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Abou
                    <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                        <li><a target="_blank" rel="noopener noreferrer" class="dropdown-item" href="https://www.facebook.com/tuanblep">Nguyen Anh Tuan
                        <li><a target="_blank" rel="noopener noreferrer" class="dropdown-item" href="https://www.facebook.com/trinhhuu.truong.1">Trinh
                        <li><a target="_blank" rel="noopener noreferrer" class="dropdown-item" href="https://www.facebook.com/nttung.1406">Nguyen Thanh
                        <li><a target="_blank" rel="noopener noreferrer" class="dropdown-item" href="https://www.facebook.com/profile.php?id=1000560658
                    </ul>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Shop
                    <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                        <li><a class="dropdown-item" href="#!">All Products</a></li>
                        <li><hr class="dropdown-divider" /></li>
                        <li><a class="dropdown-item" href="#!">Popular Items</a></li>
                        <li><a class="dropdown-item" href="#!">New Arrivals</a></li>
                    </ul>
                </li>
                    
                <ul class="navbar-nav me-auto mb-2 mb-lg-0 ms-lg-4">
                    <li class="nav-item"><a class="nav-link active" aria-current="page" href=""><% if (session.getAttribute("userdata") == null) {%>
                            <%} else if (session.getAttribute("userdata") != null) {%>
                            <footer style="">
                                <% Customers cus = (Customers) request.getSession().getAttribute("userdata");%>
                                Hello <%= cus.getFullname()%>
                            </footer>
                            <%}%></a></li>
                </ul>
            </ul>
```

JSTL

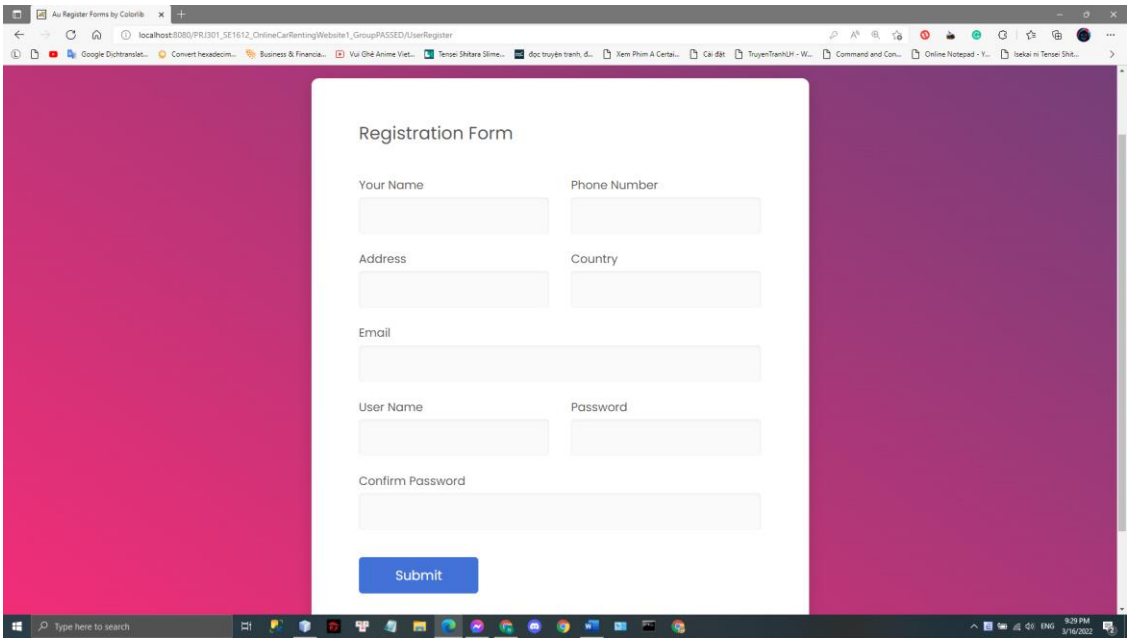
4/Screenshot:

+Login Page:



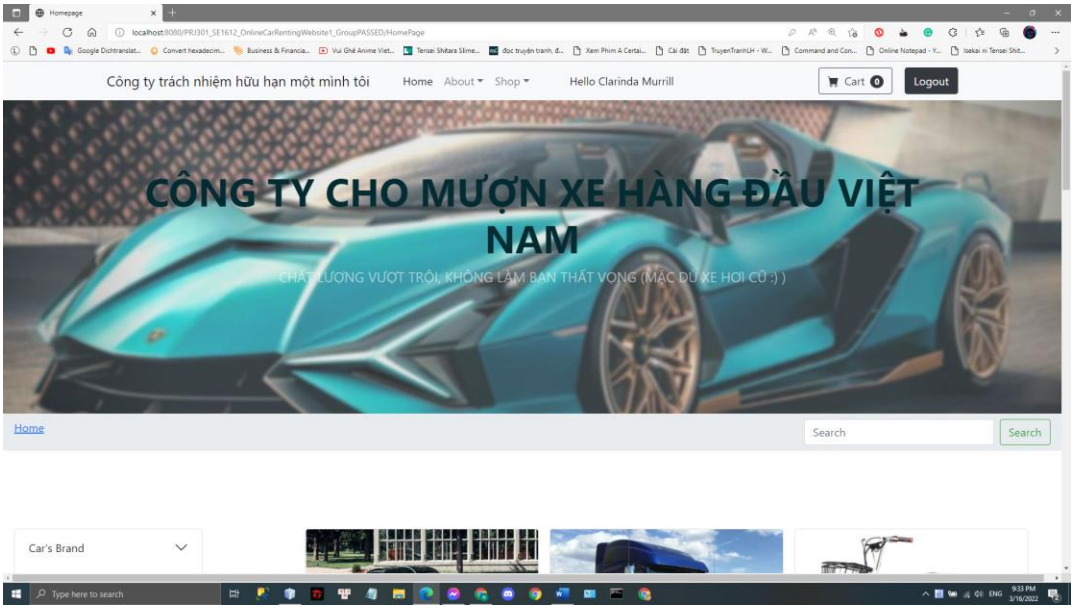
- The login page help guest can access the page with their username and password, then they can rent car(s) for their moving or traveling. If they haven't been registered yet, we have a register page to help them register their new account.

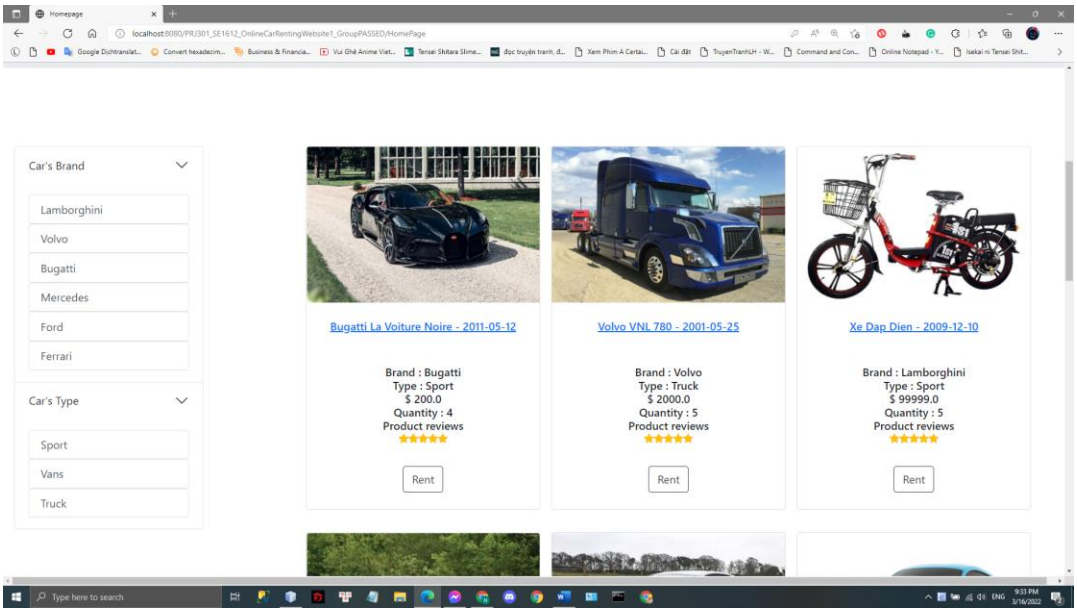
+Register page:



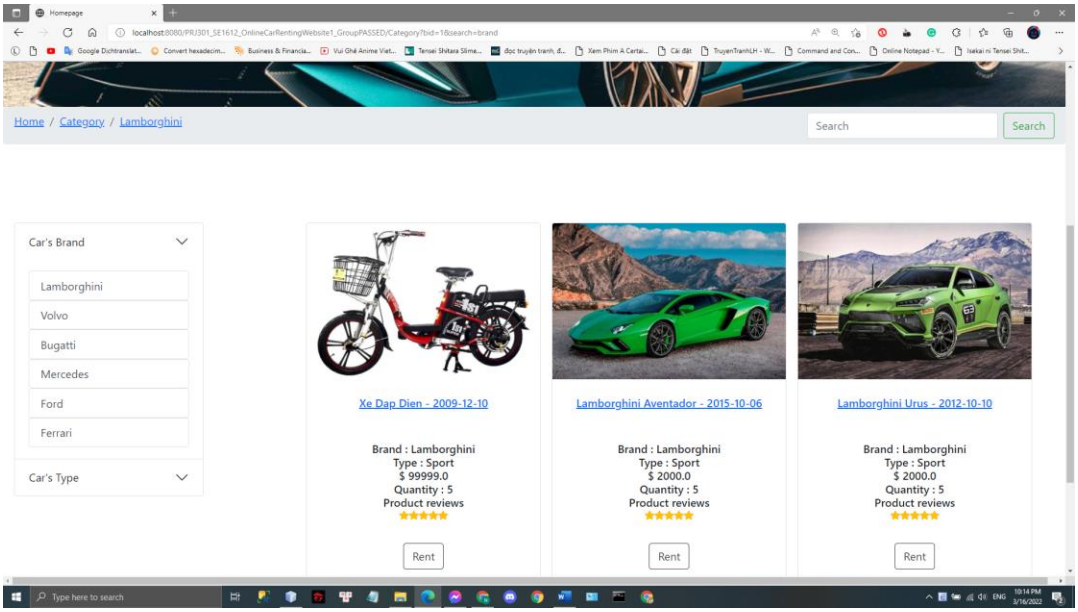
- On the register page, they must fill in all text-box to register their new account, it contains their name, phone, country, address, email, username, and password. If their information is valid, they will have their new account.

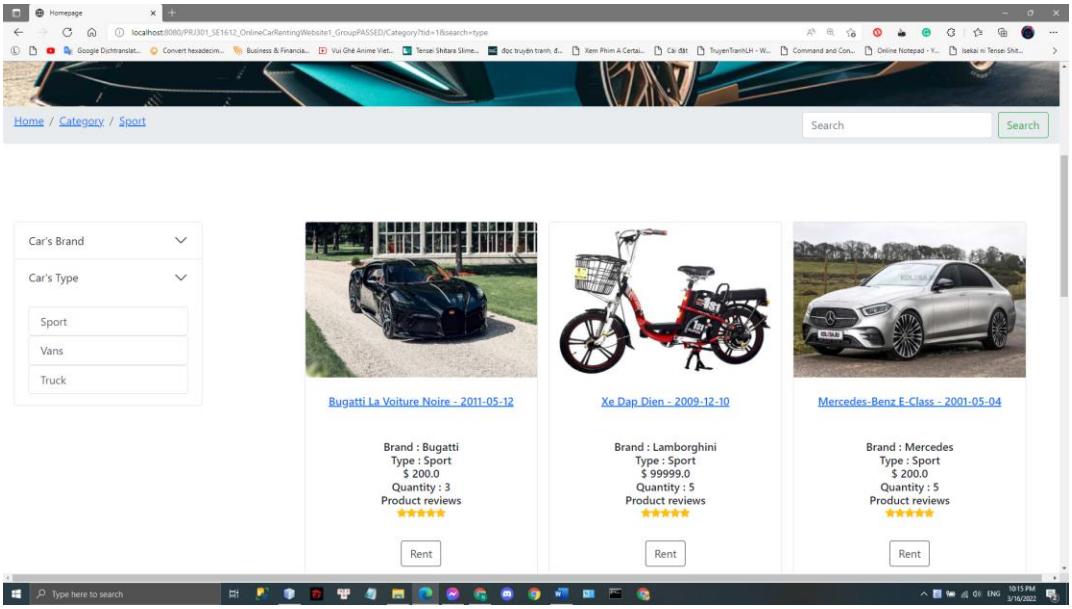
+Home page:





- On the home page, guests can view a list of all of the cars, or they can choose by brand or by type of car. In the type of car, we have three types, they are sport, vans, and truck. In brand, we have some famous brands like Lamborghini, Volvo, Bugatti, Mercedes, Ford, and Ferrari. But if they want to rent car, they must be login with their account. Then they can rent a car by pressing the button "Rent". Or to see car detail by pressing on the car's name. They can view their cart by pressing the button "Cart" on the top of the page, it shows the number of car which in their cart. If they want to choose any type of car or brand:

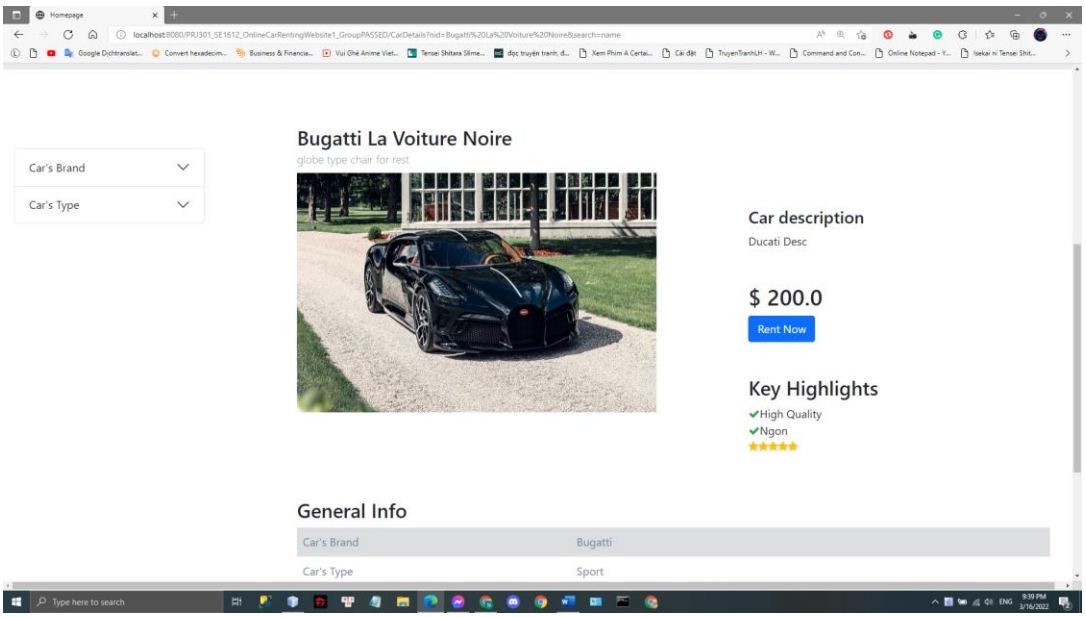




We will show them cars which are suitable for their choice. Or they can search a car by correct name or search all cars that contain which text their type in the search box.

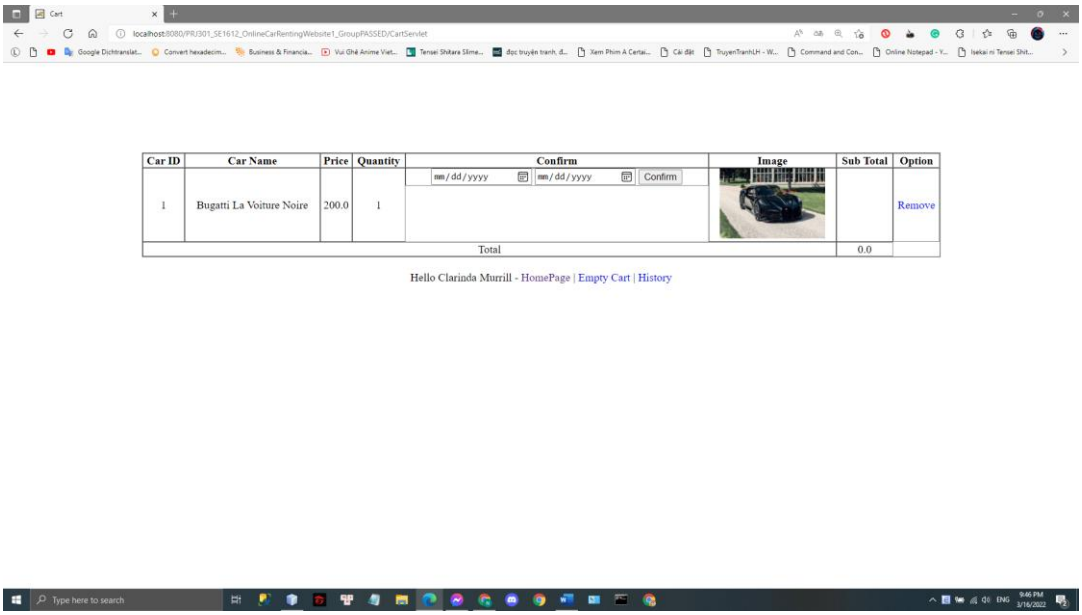
Or they can logout by pressing the button “Logout” at the top of the page.

+Car detail:



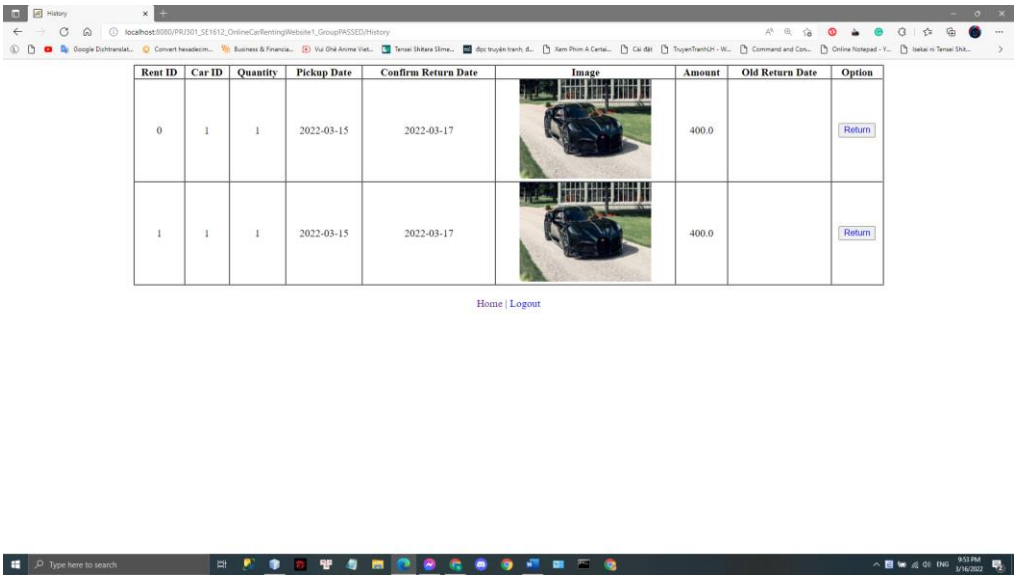
-On the car detail page, we show the guests the information of the car such as price per date, car's description, brand, type, date of production. If they want to rent our car they must be login with their account, then they can press the button "Rent Now", then it will be sent to the cart page.

+Cart Page:



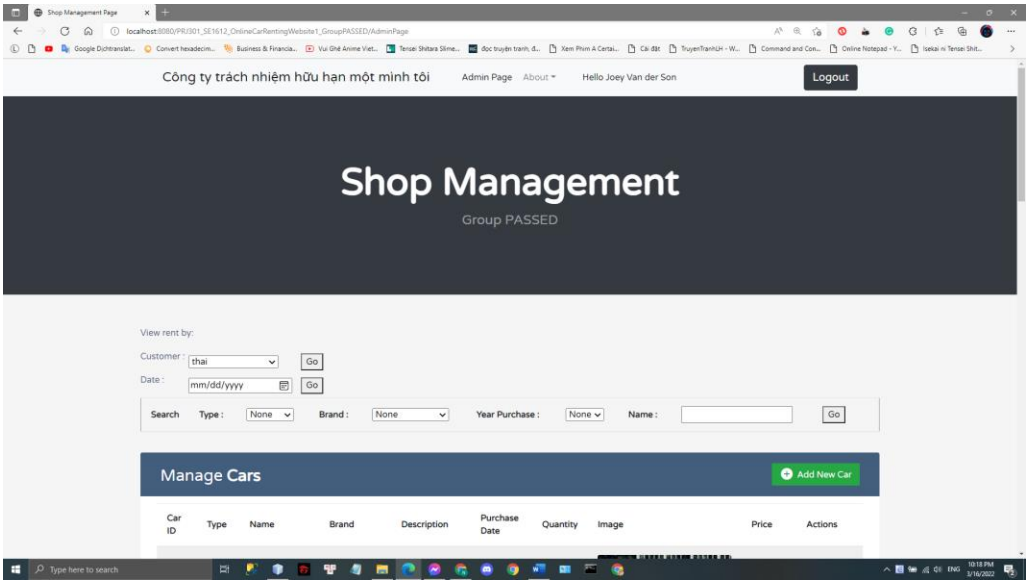
On the cart page, customers must be login with their account then they can view their renting car, they must choose the date start to rent and the date return to can be rent car. If they want to rent car(s), there is a button "Rent" which will appear after they confirm the start and return date. It has a button "History" at the bottom of the page, which will be sending them to a history page to see their history of renting car.

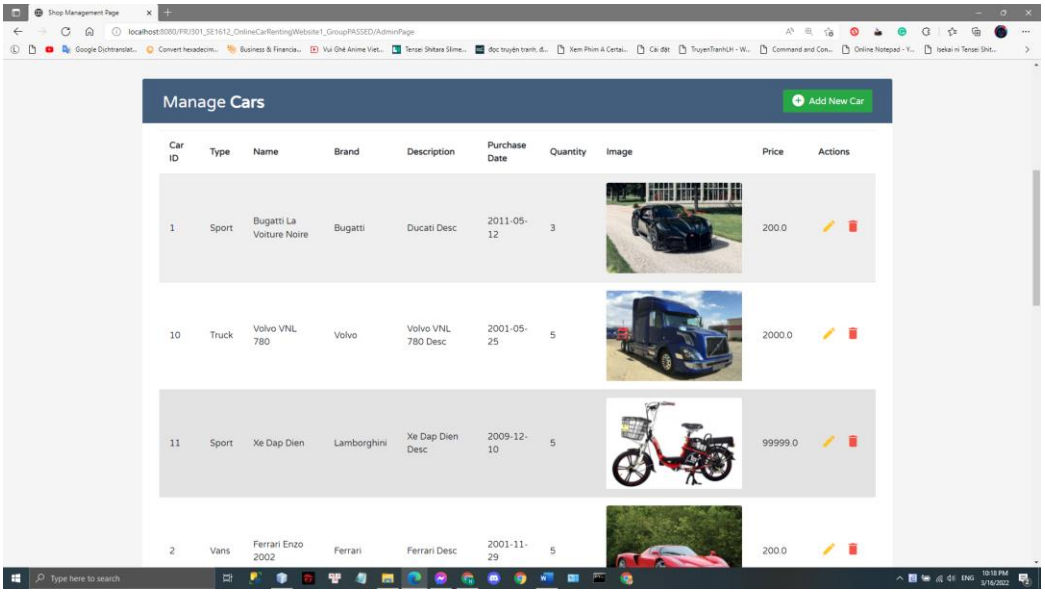
+History Page:



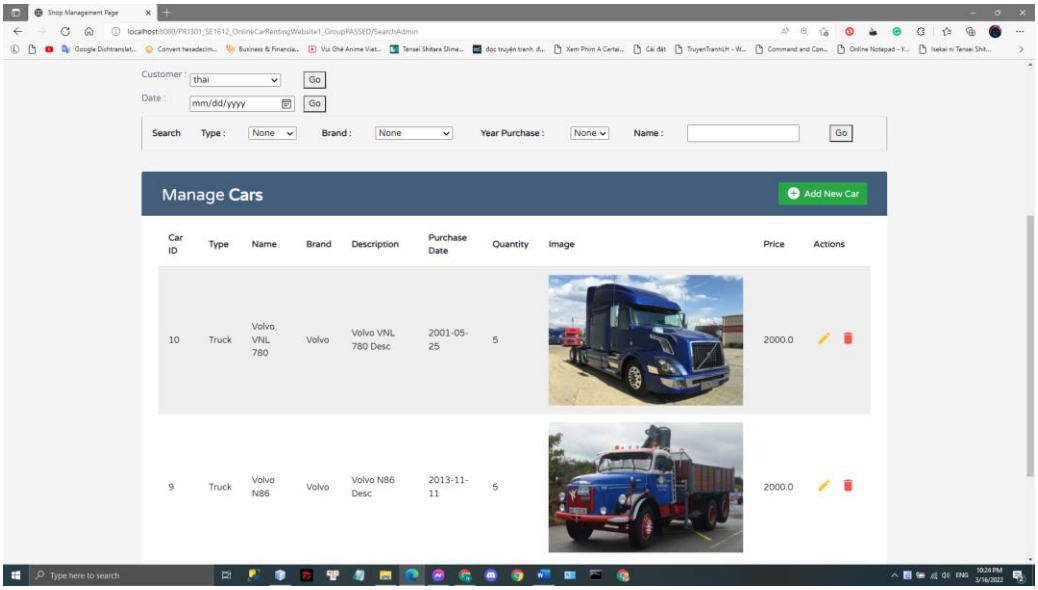
On the history page, customers can see their history of renting car, it shows the start and returns dates, amount, and quantity of the rented car. Besides, they can return the car before the return date by pressing the button "Return", which will update their return car date.

+Admin page:

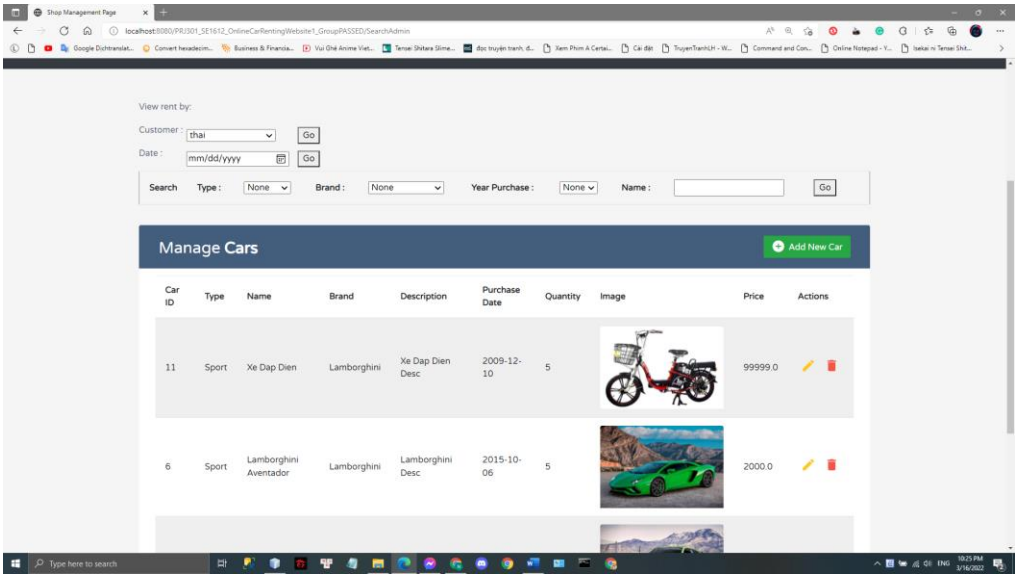




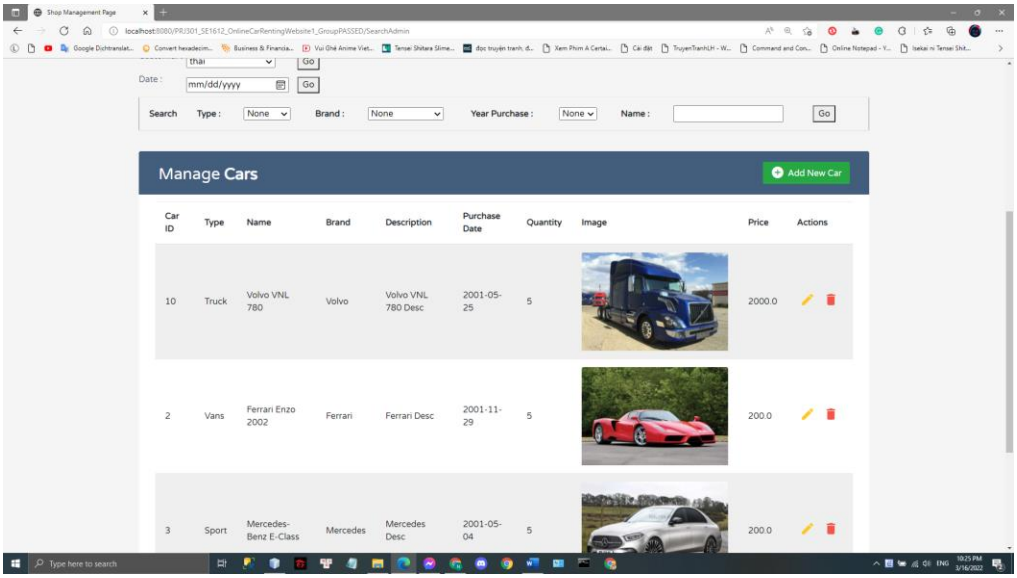
On this page, the manager will manage all the cars in the store, they can add, delete, edit with them.



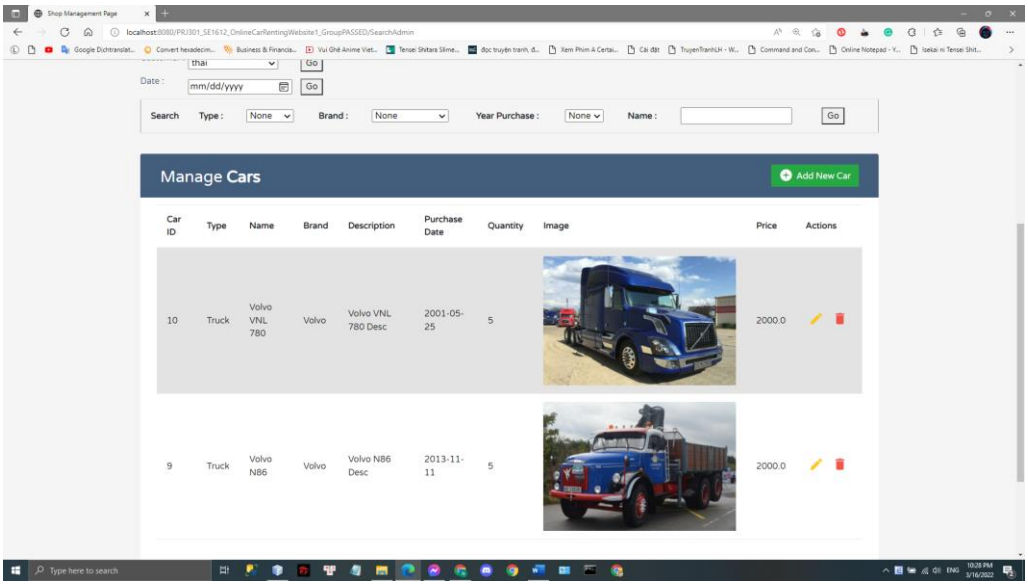
Also, they can search for cars by type



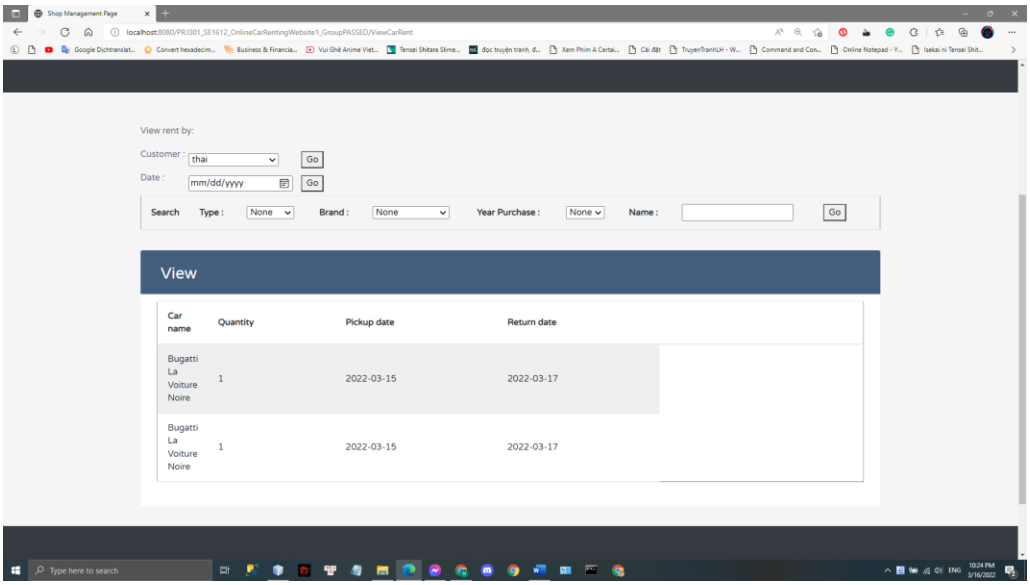
By brand



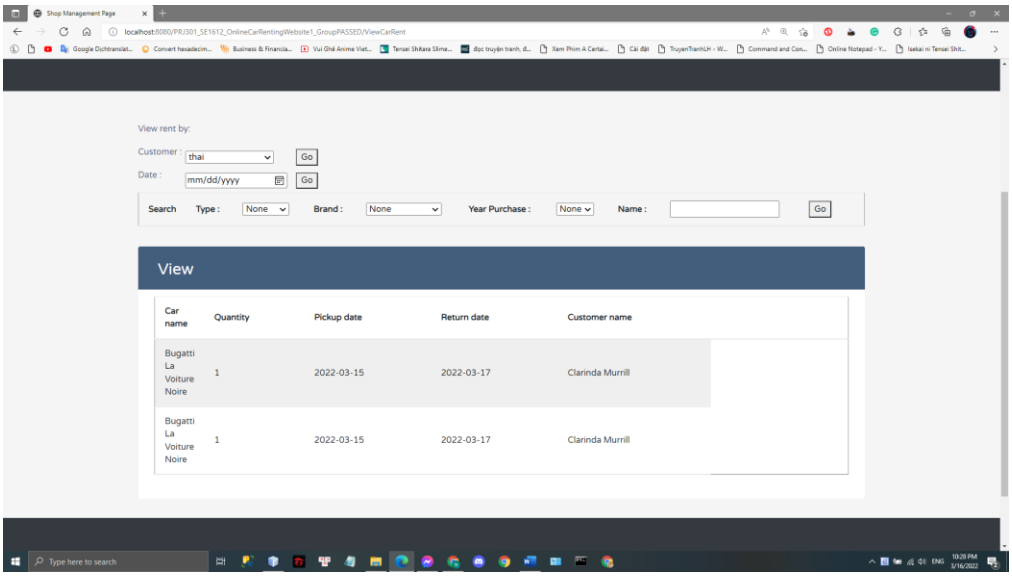
By purchase year



By car name.

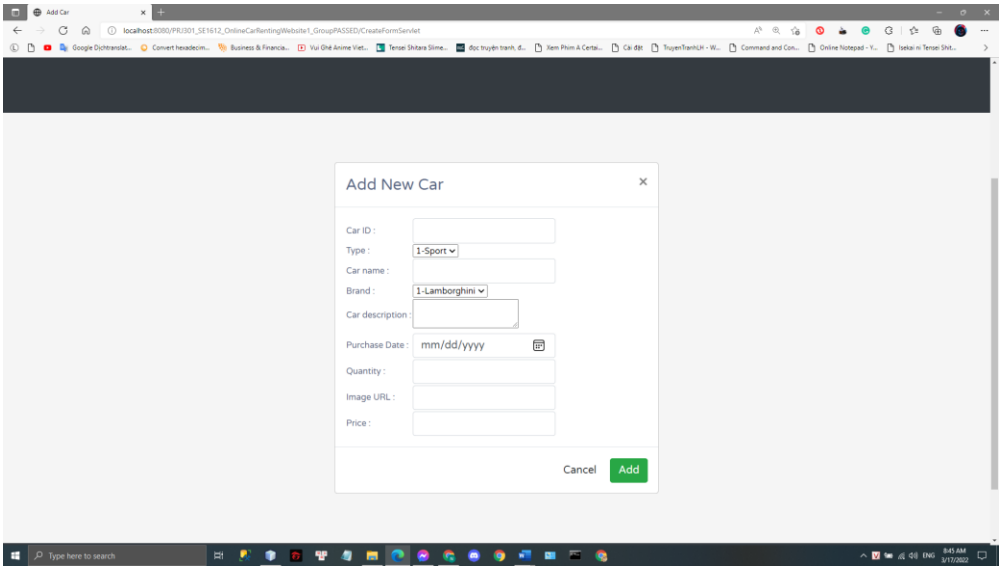


They can also see a list of car rental history by customer



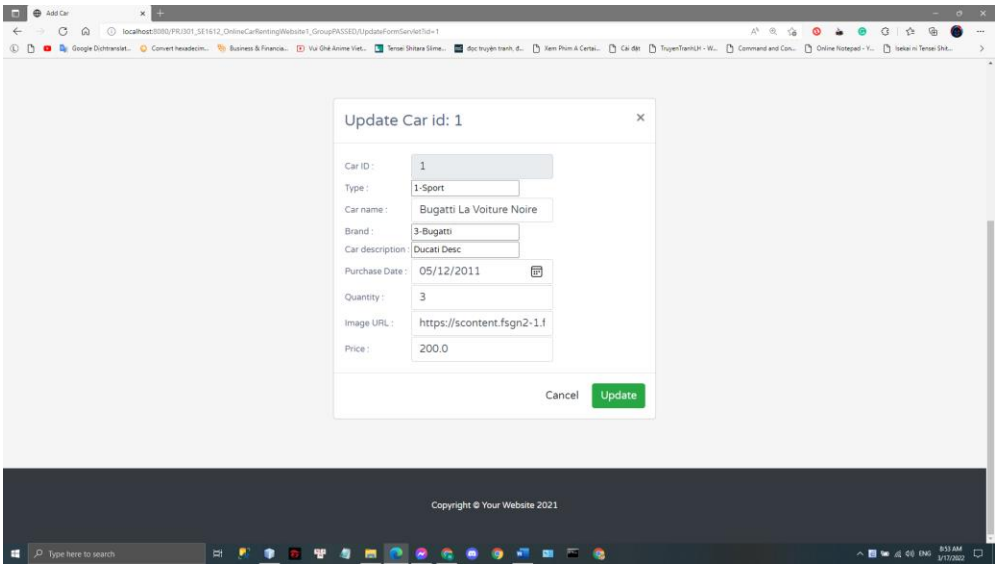
By date.

+Create Car page:



On this page, managers can add a new car by fill in the text box all information of the new car like Id, type, brand, name, description, day purchase, link image, quantity, and price per day.

+Update Car page:



On this page, managers can edit car which is available in shop, they can edit name, purchase date, quantity, link image, price per day.

5/ Conclusion and Discussion

* Advantage

- Have a basic database, project, report.

- Design project with correct MVC2.

- Front-end interface using CSS, Bootstrap.

- Design the right functions according to the description.

* Defect

- The function “Return car” has not finished. It cannot update car quantity correctly.