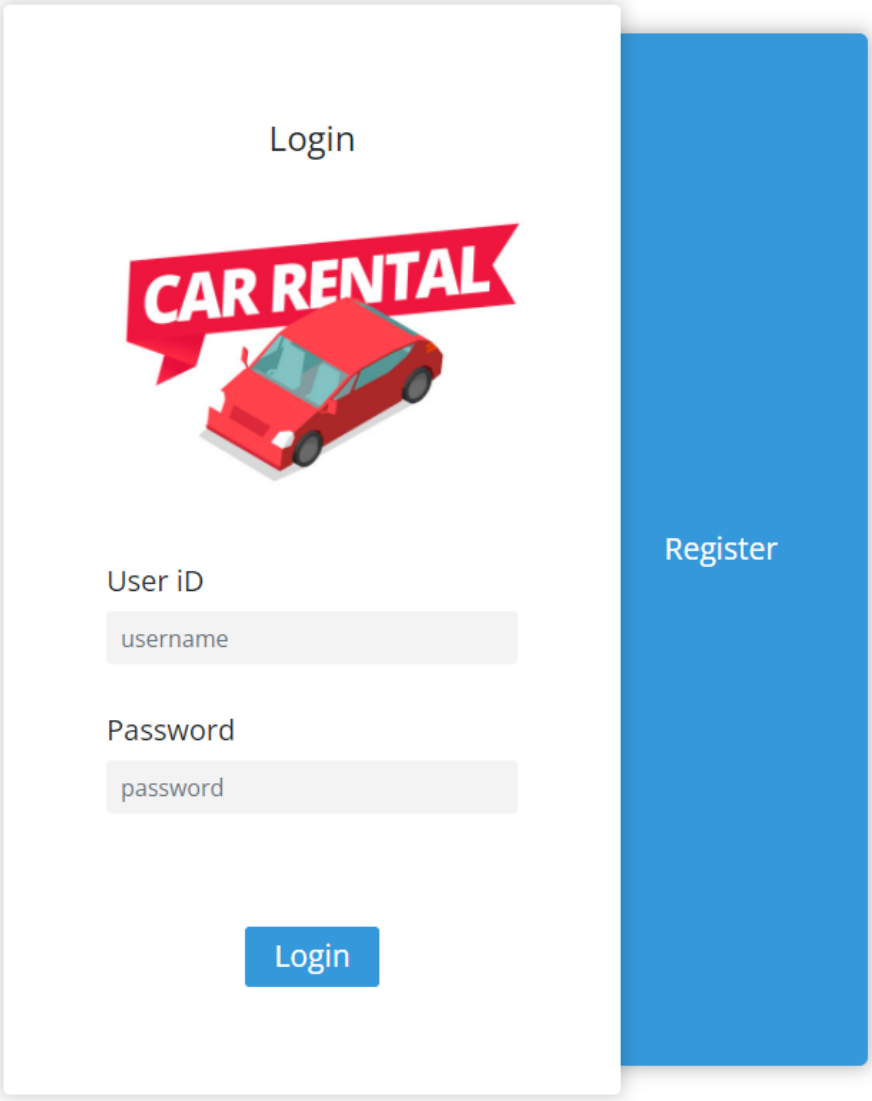
Database Report

Made by:
Abdallah Ashraf 6365
Islam Wael 6373
Aly Zaitoon 6368

GUI:

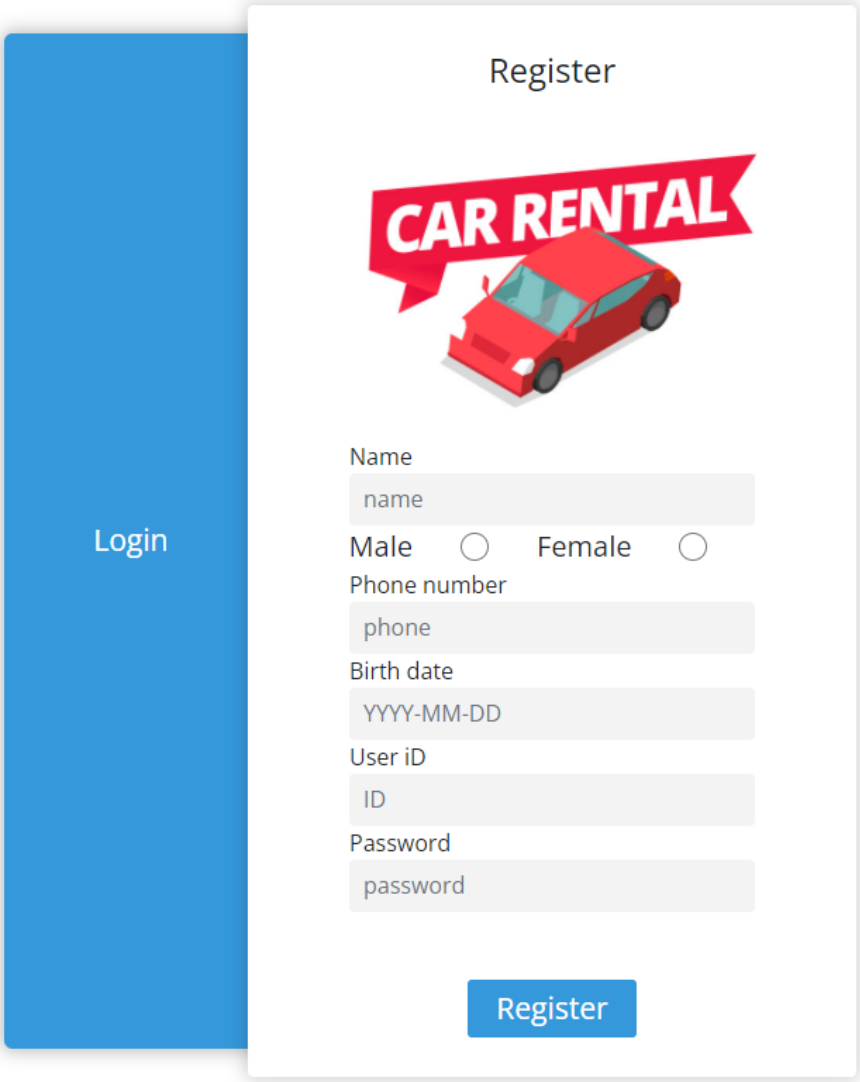
1) Login Page:



This page talks about a login page where the user needs to write his user ID and password in order to proceed to the next window. The user may be a customer or admin. If the user is a customer, he will be directed to a reservation for a car page. If the user is admin, he will be directed to the admin page where he can configure the data from the database and check for daily reports. After putting user ID and password and pressing login , the website connects to the database and checks if the user exists. If it fails then an alert message will display that the user is not valid. If the user

omits user ID or password , an alert message will display that there is a field required.

2) Register Page:

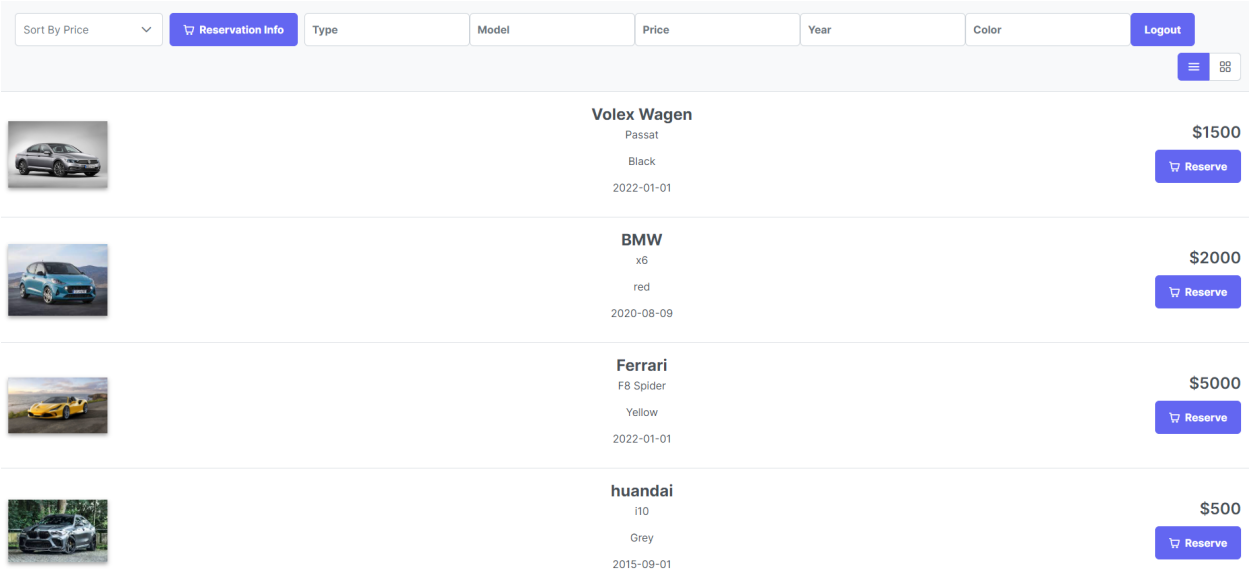


If a user does not have an account , he will press the register button to redirect to the registration page. The user must fill in the required fields in order to create an account. When a user successfully fills the form, the website connects to the database and inserts all the values in the fields in the database. However, if the user writes a user ID which is included in the database an alert message will appear showing that user ID already exists. Omitting any fields will display an alert message. After successful

registration, the user will be directed to the login page again and fill in the user ID and password that he wrote in the register page.The registered user will always be the type of customer.

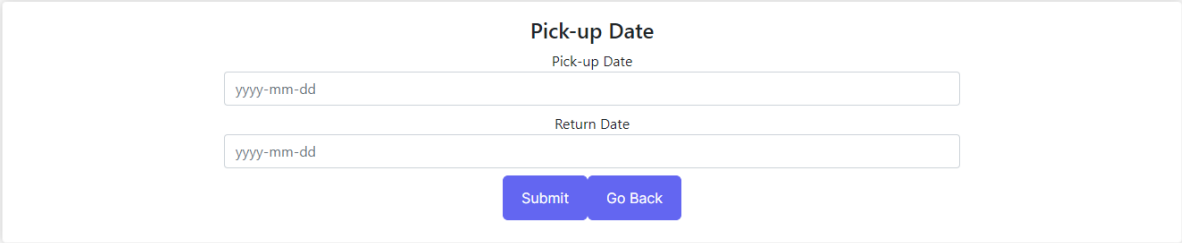
3) Customer Pages:

Reserving a car:



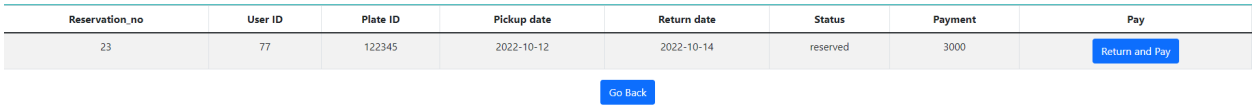
The customer will be directed to reserving a car , the customer must select what cars he likes with its specs. The customer also can use filters to get the required car more conveniently. The customer can also search the required car by typing its name and the car will show up. After selecting the car needed, he should press the reserve button and he is directed to the pick up and return dates. If the customer wants to pay for reserving a car he will press on the payment button. If the customer did not like anything he could simply logout from his account to redirect from the login page.

Pick up and return dates:



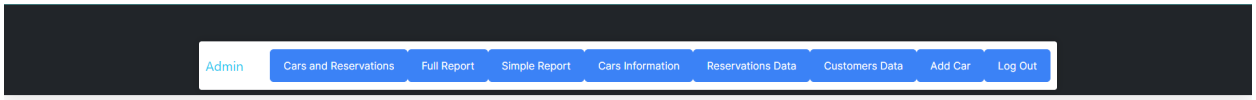
After the customer selects a reserve for the car he liked to reserve, another page will open . The user must select the pick up date and return date. Then, the website connects to the database and inserts the customer ID and the car he reserved and the amount he needed to be paid on the car. If the customer selected the return date less than pick up date an alert message will appear showing that the date picked in return date is less than pick up date.

Reservations Information:



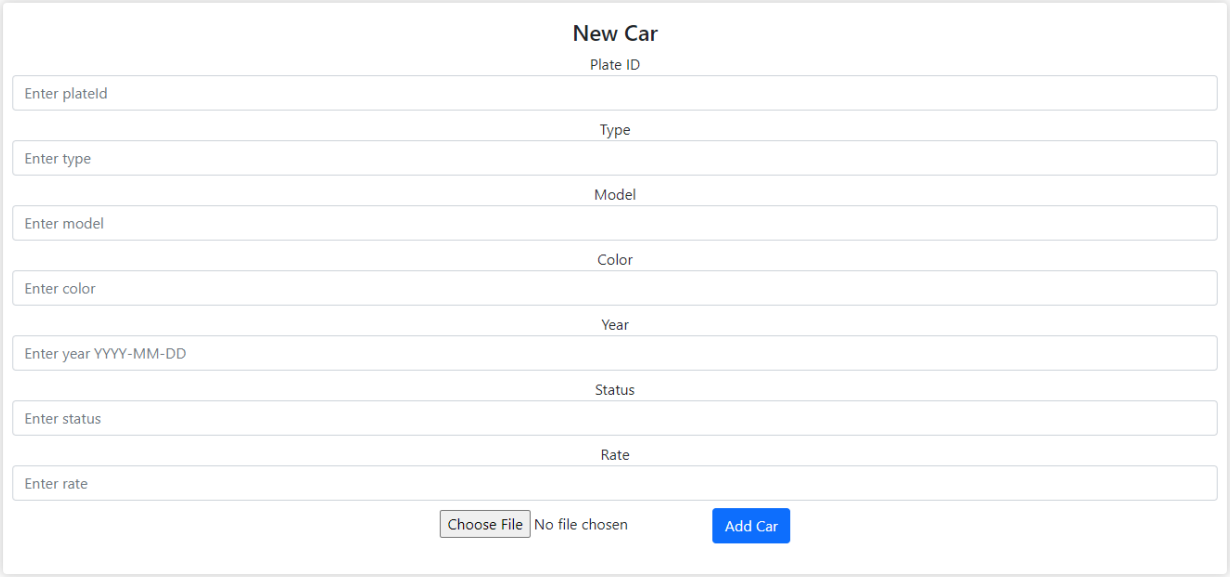
After the customer pressed the pay button , a table will appear showing all his reservations and for each row there is a pay and return button in order to pay the reservation and return the car. If he does so, then the payment will be zero and the row will be deleted from the database and the page itself.

4) Admin page:



When admin logins he will be directed to a page where there is a navigation bar. The navigation bar contains adding new cars, viewing reservations, all cars in the market itself including out of service cars, customers data, all cars reserved by customer , a full report that consists of all everything , a simple report that consists of customer information and car model and plate id. There is a log out button of admin which is in the navigation bar that redirects admin back to the login page.

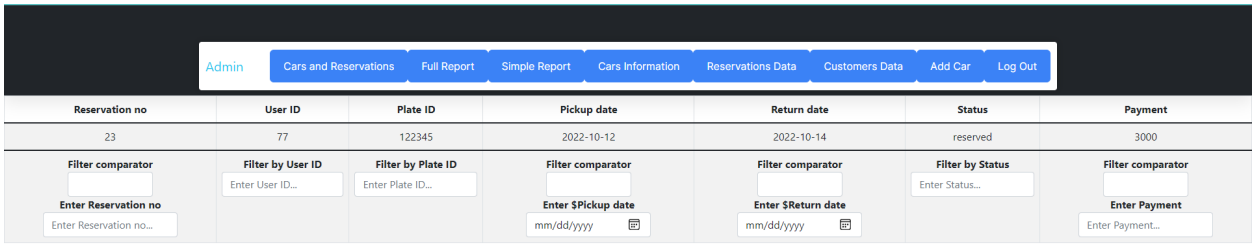
Adding a new car:



Admin can create a new car after clicking the add car button in the navigation. The page includes all the car information needed to insert this car in the database. After filling in the form, the website connects to the database and checks whether the new car is already in the list. If the plate id is the same as one of the existing cars it will display an alert message displaying that another car has the same plate id. If the admin omitted a

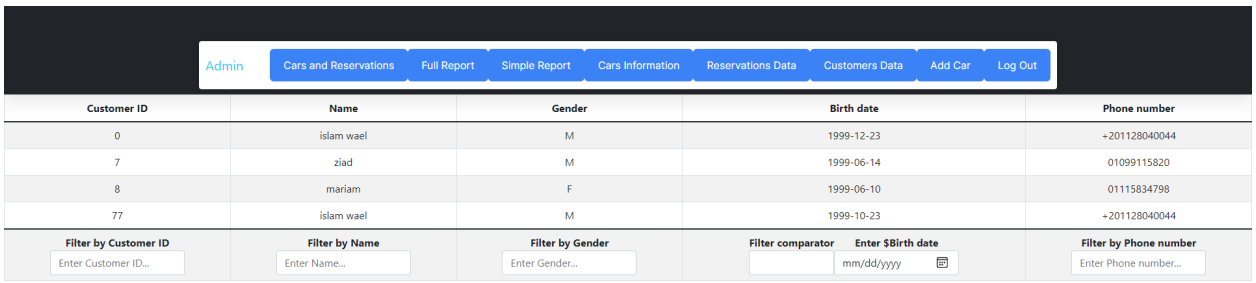
field then an alert message will appear that the omitted field needs to be filled. The admin can also put a picture to the car so it can be displayed on reserving a car.

Reservations Data:



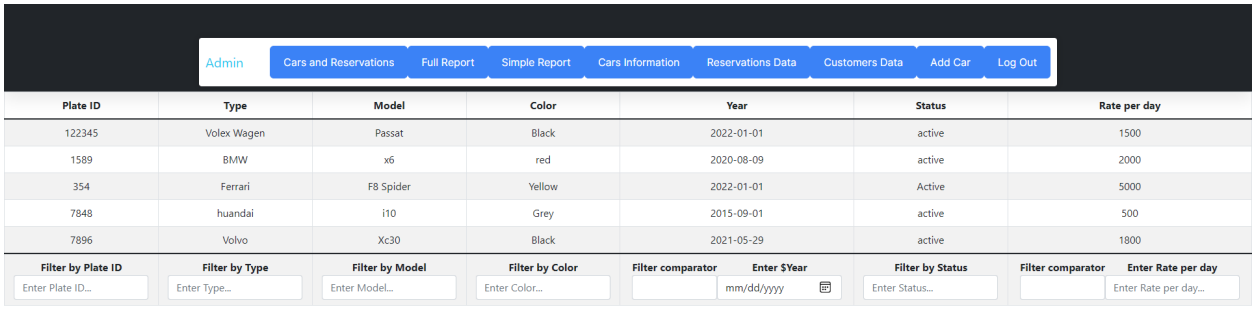
If the admin wants to know all reservations he can press the reservations data button. The page displays a table which includes customer id and plate id with its returning date and pick up date and so on. If the admin wants to check for a specific customer id then he can filter and write the customer id in the filter section. The admin can use multiple searches to find a specific reservation. The admin can also edit,sort and filter any other data he needs.

Customers Data:



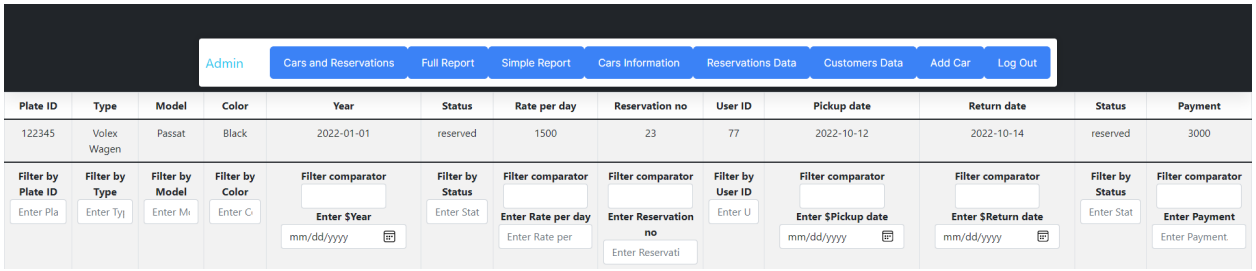
If the admin wants to know a specific customer information then he can press the customers data button. The page displays a table which includes all customer data. By getting a specific customer, the admin can filter by writing his name on the name column to get the customer name and all his details. The admin can also change any customer information at any time.

Cars Information:



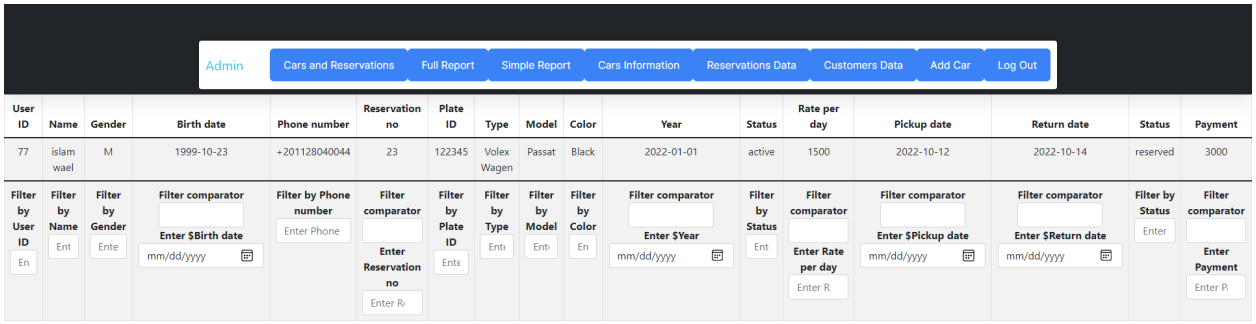
If the admin wants to know a specific car information then he can press the cars information button on the navigation bar. The page displays a table which includes all information on each car. By getting a specific car , the admin can filter by writing its model or plate ID to get all car details. The admin can also change any car information at any time.

Cars and Reservations:



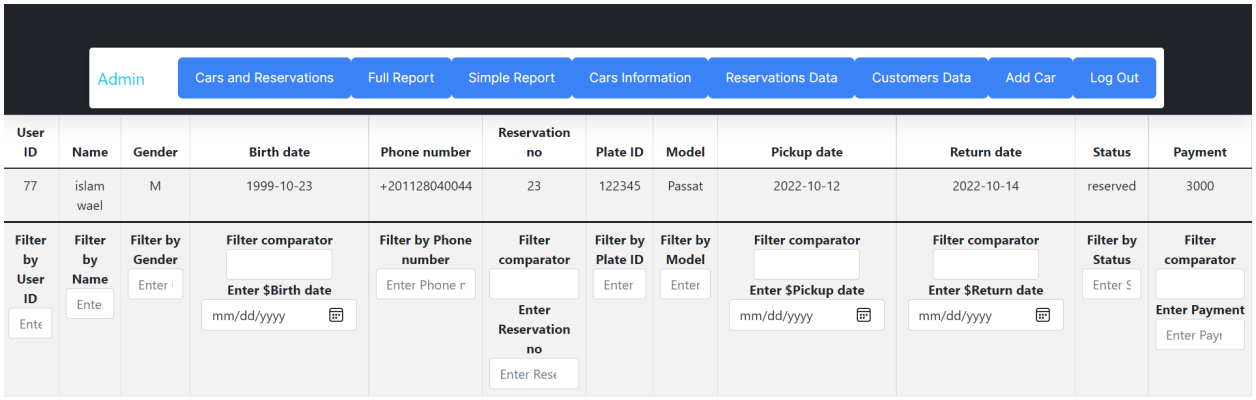
If the admin wants to know all the cars reserved then he can press the cars and reservations button on the navigation bar. The page displays a table which includes all information about reservation number, plate id and so on. By getting a specific reservation, the admin can filter by writing plate ID or reservation number to get the required reservation the admin needs. The admin can also change any reservations at any time.

Full Report:



If the admin wants to know every detail in reservation the he can press the full report button. The page displays a table which includes all information about cars and customers in each reservation. The admin can only view the report and it cannot be edited or can be sorted but it can be filtered.

Simple Report:



If the admin wants to know customer information and plate ID and model of car only he can press the simple report button.The page displays a table which includes all customer information about each customer and plate ID and model of the car.The admin can only view the report and it cannot be edited or can be sorted but it can be filtered.

Backend

We implemented the backend for our car rental system using a spring boot application and we implemented a REST API providing all the functionalities that the front need(supplying the front end with all data needed by axios ajax calls), connect to database (using jdbc) and retrieving data from database.

In Our Backend we have 3 main sets of classes

1. Entities

This set of classes contains all the database entities with its corresponding attributes, Objects of these classes act as a buffer in memory between frontend and database , we construct these objects by POST calls to be written to the database and we send them to frontend by GET calls to show info in frontend.For example we have an entity called admin which corresponds exactly to the admin table in the database.

2. Queries

This set of classes contains functions that executes all the queries we need in SQL of insert , update and delete and also it applies complex queries like joining multiple tables to give us the facility to gather more data in a single table and be able to publish reports to the admin portal.

2.1 Examples

- select * from customer as c1 join Reservation as r1 on c1.customer_id=r1.user_id join car as c2 on r1.plate_id = c2.plate_id;

This query gets a full report of all reservations with the customers who made it and the cars they reserved

- "update reservation set payment=? where reservation_number=reservation_number";

This query updates reservation record according to its reservation number (we use it to set payment to 0 when customer pays for its reservation

3.Controllers

In our implementation we had one Controller Class which represents the API and this class has instances from the Queries classes and executes them according to the received mapping (url).

3.1Example function for GET request

```
@GetMapping(value="/getReservations")
public List<Reservation> getReservations()
{

   return reservationQuery.getReservations(jdbc);
}
```

This an example function of the API that gets a request at url http://localhost:8080/getCars

Then it responds to this GET Request with all reservations.

3.2Example function for POST request

```
@PostMapping(value = "/addCustomer")
public String addCustomer (@RequestBody Customer customer)
{
   String ret;
   ret=this.customerQuery.addCustomer(customer,jdbc);
   return ret;
}
```

This an example function of the API that gets a request at url http://localhost:8080/addCustomer

Then it responds to this POST Request by creating the customer and inserting it to the database.