

CMPE 172 Enterprise Software Platforms

Final Project Report

Car Rental System

Team 5

Dylan Zhang

En-Ping Shih

Tsz Hin Chan

Professor: Babu Thomas

San Jose State University

Spring 2020

May 11th, 2020

Table of Contents

Project Overview	2
Project Description	2
HTML/CSS	2
Spring boot	3
RDS(MySQL)	3
EC2	4
Docker	4
System Diagram	5
Team Member Contributions	6
Next Steps/Lessons Learned	7
Relevant Links	7
Links to the GitHub upload	7
Public URL to the application.....	7
Test account credentials.....	8
References.....	8

Project Overview:

The car rental platform has become an exciting startup option and has a place in the world. Users who want to rent a car can search for the car according to their needs and rent the car after providing the necessary details and payment. The goal of our project is that the process of registering a "car rental" takes only a few seconds. The user only needs to fill in personal information and select a car(s) to fulfill the entire process.

Project Description:

Our project is a Three-tier architecture that we need a client, server, and database. For the frontend part, we used **HTML** and **CSS**. The reason why we use HTML is that based on our research, HTML is the most compatible with Spring Boot and Thymeleaf. Plus, all of our team members have some experiences in HTML.

→ HTML/CSS

The screenshot shows an IDE with a project named 'demo_car_rental'. The project structure on the left includes a 'main' directory with 'com.example.demo_car_rental' package, 'resources' directory, and 'templates' directory. The 'templates' directory contains several HTML files: 'booking_page.html', 'home_page.html', 'login_page.html', 'receipt_page.html', 'signup_page.html', and 'welcome_page.html'. The 'welcome_page.html' file is selected, and its content is displayed in the main editor. The code is written in HTML and uses Thymeleaf syntax for dynamic content. It includes a header with a title and a London image, a navigation bar with buttons for 'Car', 'Hotel', and 'Car', and a main content area with a 'Car' tab. The 'Car' tab contains a form for searching for a car, with fields for 'city', 'pick-up date', and 'drop-off date'. The form has a 'Search' button. The code also includes a discount message: 'Special offer if you book today: 25% off anywhere in the world!'. The code is as follows:

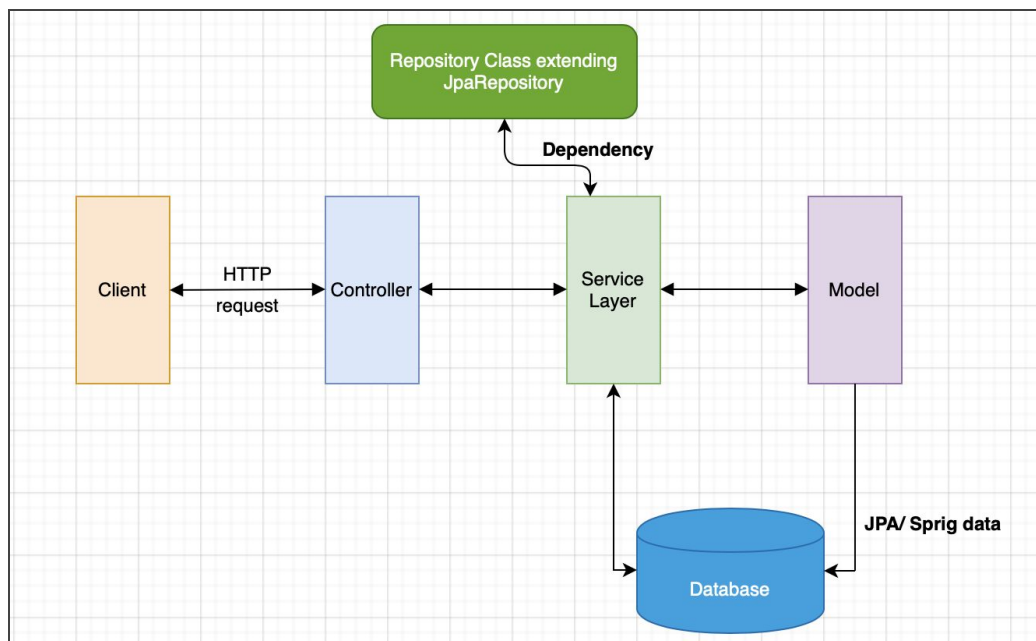
```

33  <!-- Title -->
34  
35  <div class="w3-display-middle" style="text-align: center;">
36  <div class="w3-bar w3-black">
37  <button class="w3-bar-item w3-button tablink" onclick="openLink(event, 'Car');"><i class="fa fa-car w3-margin-right">Car</i></button>
38  <button class="w3-bar-item w3-button tablink" onclick="openLink(event, 'Hotel');"><i class="fa fa-bed w3-margin-right">Hotel</i></button>
39  <button class="w3-bar-item w3-button tablink" onclick="openLink(event, 'Car');"><i class="fa fa-car w3-margin-right">Car</i></button>
40  </div>
41  </div>
42  <!-- Tabs -->
43  <div id="Car" class="w3-container w3-white w3-padding-16 myLink">
44  <h3>Best car rental in the world!</h3>
45  <p><span class="w3-tag w3-orange">DISCOUNT!</span> Special offer if you book today: 25% off anywhere in the world!</p>
46  <form action="#" th:action="@{/tempForm}" th:object="${tempRent}" method="post">
47  <div class="w3-row-padding" style="text-align: center;">
48  <div class="w3-half">
49  <label for="city"><i class="fa fa-map-marker"></i> Pick Up Location</label>
50  <input class="w3-input w3-border" type="text" th:field="*{pickup}" id="city" name="city" placeholder="City">
51  </div>
52  <div class="w3-half">
53  <label for="city"><i class="fa fa-map-marker"></i> Drop Off Location</label>
54  <input class="w3-input w3-border" type="text" th:field="*{dropOff}" id="city" name="city" placeholder="City">
55  </div>
56  <div class="w3-half">
57  <label class="pickup">Pick-Up Date</label><br>
58  <input type="date" th:field="*{startDate}" class="form-control" name="start_date" placeholder="Pick-Up Date">
59  </div>
60  <div class="w3-half">
61  <label class="dropoff">Drop-Off Date</label><br>
62  <input type="date" th:field="*{endDate}" class="form-control" name="end_date" placeholder="Drop-Off Date">
63  </div>
64  </div>
65  <p><button class="w3-button w3-dark-grey" type="submit">Search</button></p>
66  </form>
67  </div>

```

For the backend part, we used **Spring Boot MVC** to serve web content, and use Maven to run the spring boot application. We used **AWS RDS** for the remote database and **MySQL** workbench for the local database. **JPA** was used to connect the database. We used **Docker** for dockerizing the application so that the web application can run on any OS system. Finally, we used the **AWS EC2** for the server which allows our application to run on a public URL.

→ Spring boot Flow Diagram (Three-Tier)

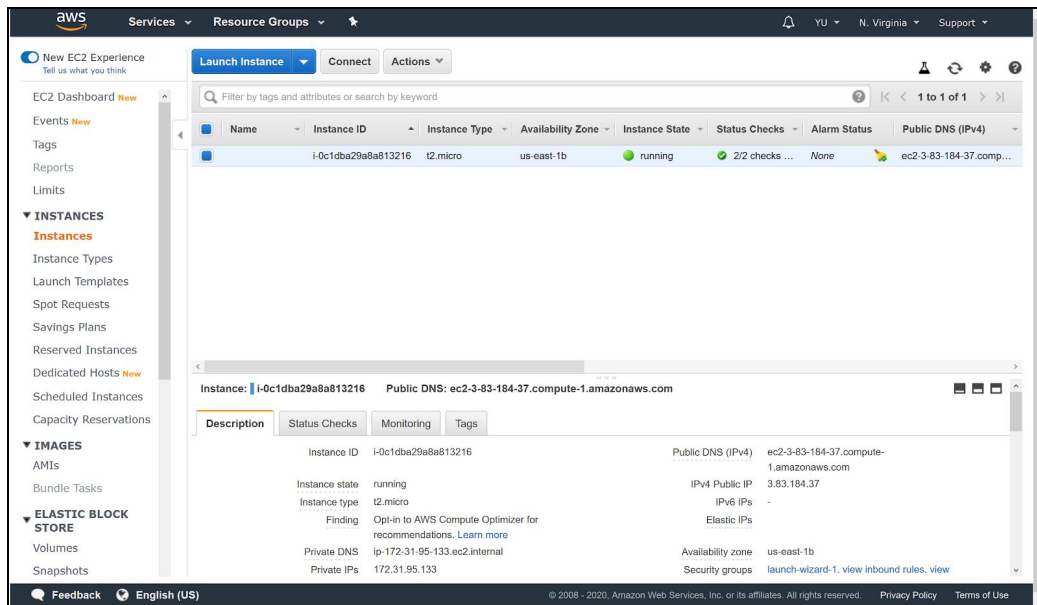


→ AWS RDS(MySQL)

```

application.properties  Car_controller.java  demo_car_rental/pom....  WebSecurityConfigura...  RentDAO.java
1 #debug=true
2 #spring.http.log-request-details=true
3 spring.jpa.hibernate.ddl-auto = create
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.datasource.url=jdbc:mysql://database-1.cp0nqf3tr05o.us-east-2.rds.amazonaws.com:3306/car
6 spring.datasource.username=admin
7 spring.datasource.password=test1234
8 server.port=5000
9
10 ## Hibernate Properties
11 # The SQL dialect makes Hibernate generate better SQL for the chosen database
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
  
```

→ AWS EC2



→ Docker installed with AWS EC2

```

Desktop — ec2-user@ip-172-31-95-133:~/demo_car_rental — ssh -i cmpe172.pem ec2-user@ec2-3-83-184-37.compute-1.amazonaws.com — 139...
containerd version: ff48f57fc83a8c44cf4ad5d672424a98ba37ded6
runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
init version: fec3683
Security Options:
  seccomp
   Profile: default
Kernel Version: 4.14.173-137.229.amzn2.x86_64
Operating System: Amazon Linux 2
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 989.4MiB
Name: ip-172-31-95-133.ec2.internal
ID: 6DGS:DN86:73QM:OEJC:JHS2:NN7R:TWDN:C7UP:DRM7:REJS:YBGZ:4KQ6
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

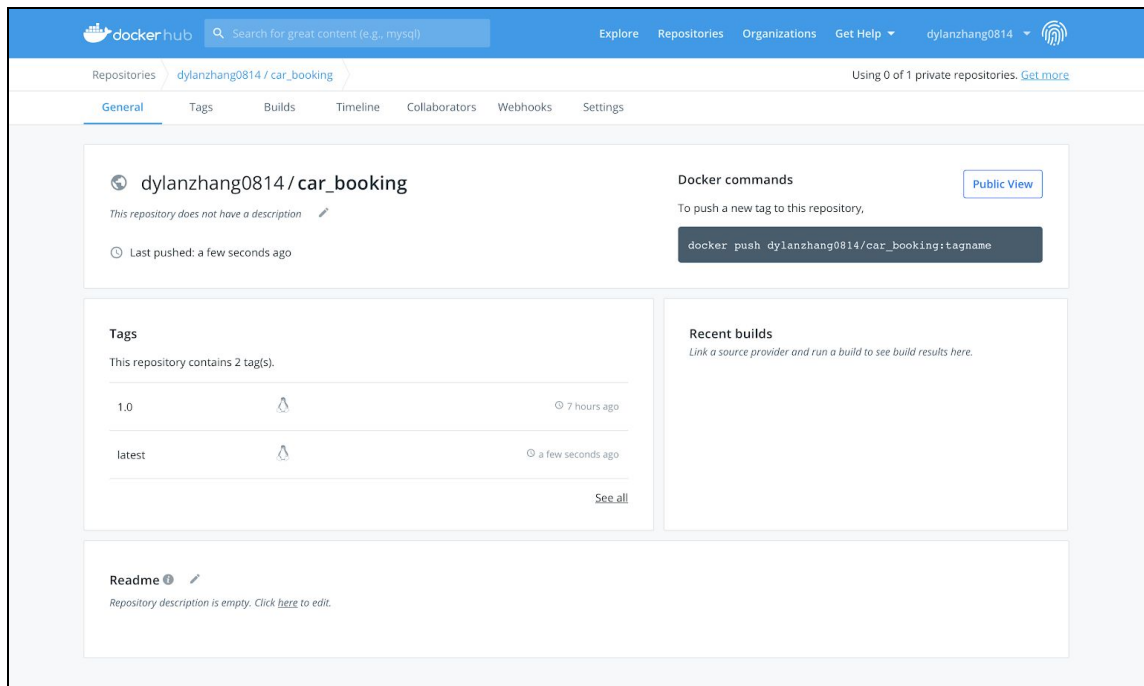
[ec2-user@ip-172-31-95-133 ~]$ build -t cmpe172/carbooking
-bash: build: command not found
[ec2-user@ip-172-31-95-133 ~]$ ls
demo_car_rental
[ec2-user@ip-172-31-95-133 ~]$ cd demo_car_rental
[ec2-user@ip-172-31-95-133 demo_car_rental]$ docker build -t cmpe172/carbooking
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage: docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[ec2-user@ip-172-31-95-133 demo_car_rental]$ sudo docker build -t carbooking:1.0 .
Sending build context to Docker daemon 44.82MB
Error response from daemon: Dockerfile parse error line 1: FROM requires either one or three arguments
[ec2-user@ip-172-31-95-133 demo_car_rental]$ sudo docker build -t carbooking:1.0 .
Sending build context to Docker daemon 44.82MB
Step 1/3 : FROM amazoncorretto:11
11: Pulling from library/amazoncorretto
a3f8e652bdc4: Pull complete
d195fa3929b2: Pull complete
Digest: sha256:23b3da2fac6d1802226298da373a6c5d1560a621309c7dd2c72a7e26686190f
Status: Downloaded newer image for amazoncorretto:11
--> dcce9e8293ec
Step 2/3 : COPY target/demo_car_rental-0.0.1-SNAPSHOT.jar app.jar
--> 70e5baa0ae0e
Step 3/3 : ENTRYPOINT ["java","-jar","/app.jar"]
--> Running in 758f0da33909
Removing intermediate container 758f0da33909
--> 95397471ac87
Successfully built 95397471ac87
Successfully tagged carbooking:1.0
[ec2-user@ip-172-31-95-133 demo_car_rental]$ docker run -d -p 8080:8080 -t carbooking:1.0
60b540e5f773e483ard693800e3237489875ce31574969169aa78a7ae60b52cf
[ec2-user@ip-172-31-95-133 demo_car_rental]$

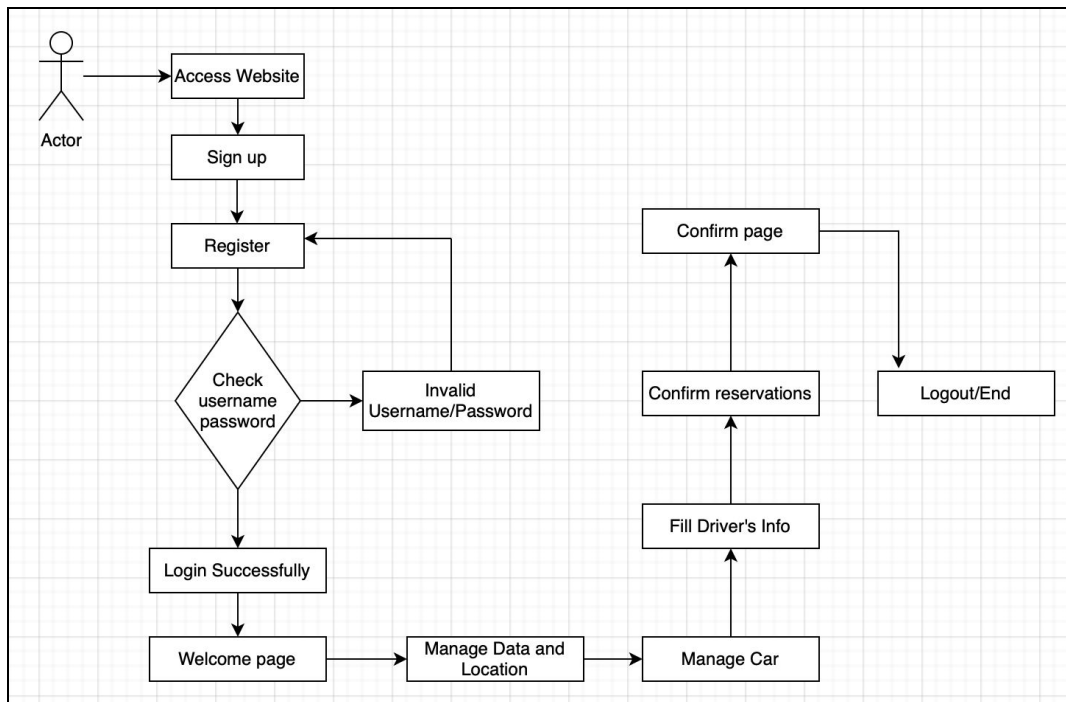
```

→ Docker Hub

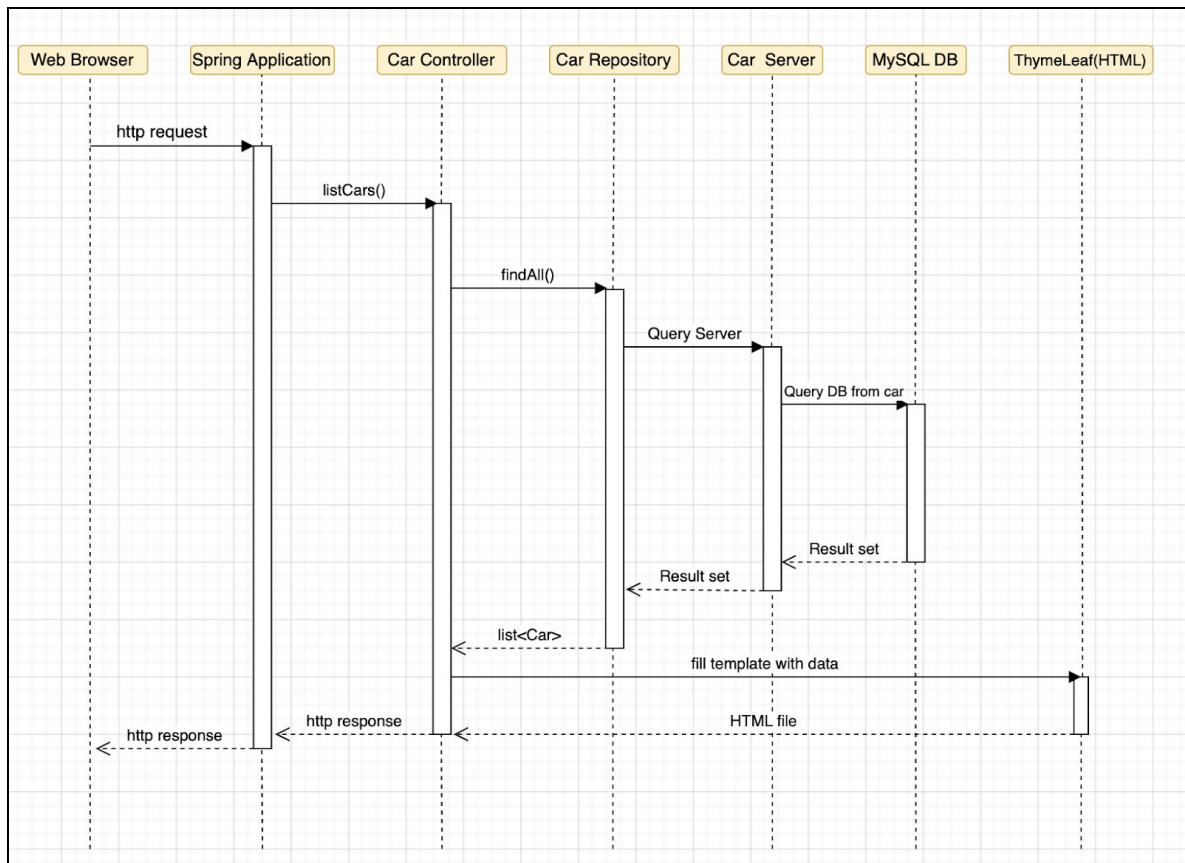


System Diagram:

- Car Rental Sequence Diagram



- System Class Template (UML)



Team Member Contributions:

Components	Team member
Spring boot, RDS, Docker, PPT	Dylan Zhang
HTML/CSS, Project Report, Demo Video, README.md	En-Ping Shih
Spring boot, EC2, HTML(ThymeLeaf), Docker	Tsz Hin Chan

Next Steps/Lessons Learned:

- Next Step

Features we would like to implement in the future:

- Enable registration via social media accounts for quick registration.
- Forget/Reset password
- Profile page
- Add more car types
- Search by title Function
- Add customer service message box

- Lesson learned

- Spring boot
- AWS RDS
- AWS EC2
- Amazon Simple Storage Service (S3)
- RESTful Web Services
- Docker

Relevant Links:

- Links to the GitHub upload:
<https://github.com/SharonShih/Project-CarRental>
- Public URL to the application:
<http://ec2-3-91-214-15.compute-1.amazonaws.com:8080/welcome>
(*NOTE: The Link is now **disabled** since we want to avoid any extra charges by AWS. But it was running when we did the video & live demo in class on 5/4/2020.)

- Test account credentials

There's no test account in the database since the app is only runnable on the localhost and local database. We recommend creating a new account and use that account to sign in.

The data we already have will be under the SQL file in our repo.

File Path: *"Project-CarRental/demo_car_rental/src/main/resources/import.sql"*

References:

- All CMPE 172 Slides
- Spring boot guide: <https://spring.io/guides/gs/serving-web-content/>
- AWS EC2: <https://aws.amazon.com/ec2/>
- AWS RDS: <https://aws.amazon.com/rds/>