The Mean Squares

Pranav Lodha
Wasae Qureshi
Jeyasri Subramanian
Subarna Chowdhury Soma

Overview

The goal of this project was to develop a car rental service software that can be implemented at rental companies to organize their information much better and have easy access to data as well as customers.

Project Group

Name	SJSU ID	Role
Pranav Lodha	009468121	Team Member
Wasae Qureshi	014569880	Team Member
Jeyasri Subramanian	014510132	Team Member
Subarna Chowdhury Soma	014549587	Team Member

Introduction

Purpose of this document

The purpose of this document is to provide a detailed project report of the application TheMeanSquares Car Rental, which is designed to help car rental agencies have a web application to manage their rental business. This document includes details about project deliverables, project difficulties, deployment, and test cases

Intended Audience

This document shall be used to review if project deliverables have been met.

- Professor
- Teaching Assistant

- Project Team members

Project Repository and Deployment

Github: https://github.com/gopinathsjsu/sp20-cmpe-202-sec-49-team-project-themeansquares/tree/master

Deployment: https://app.wasaequreshi.com/home

Note: Deployment will be live until May 8th, 2020

Scope

This document defines the Project Plan of TheMeanSquares Car Rental application. The overview includes objectives of the project, organization of the project team, development process, difficulties faced, and other project related items.

Definitions and acronyms

Keyword	Definition
Project Leader	A person in-charge of organizing the team and communicating with the project supervisor
Team Member	An active member of the team responsible for making the job done
Git	Version Control system that will be used in this project
Spring Boot	Framework that helps create microservices
MySQL	Database that allows persistent queries
Angular	Framework that helps create frontend views

Background and Objective

Many Car Rental companies across the country are not very accommodating of users under the age of 25. Our service and software allows us to create a system where users under the age of 25 are able to rent a car without an underage fee for a total of 72 hours. This system allows them to reserve a call entirely online, pickup and dropoff with the touch of a button.

It also allows the admin to adjust pricing, locations, vehicles, and many other features with simplicity.

Design Patterns

Singleton

To ensure that we were safely calling our API's, we created a Singleton class. This makes sure that if an API can't be called in parallel to another, it will block and wait until it can process the next request.

Chain of Responsibility

In some places in our code, we used this design pattern to determine certain decisions in our code such as authorization to our api. We had a secure chain to help determine which type of role the user had which we returned in the response after authenticating.

Proxy

For our login page, we have to separate proxy services. This checks locally if there is a session already, if so, just reroute to the dashboard, otherwise ask the user to relogin. This saves the user time and frustration since they don't have to input their credentials each and every time.

Command and Observer

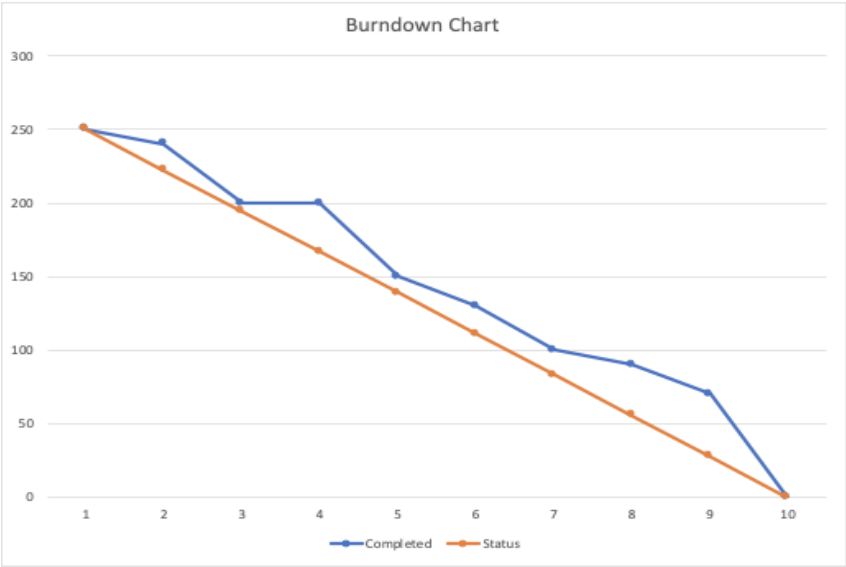
Angular provided a couple of design patterns for us to implement. The first was the command design pattern. We could easily assign a command to a button to one of our many methods and allow it to trigger the respective feature.

Observer was easy to implement as well. We could update the data in the backend and we could easily notify the frontend to update with the newly updated data.

DAO

To make it easy for the back end to work with the database, we used the DAO pattern. This made it very easy to create a class of a table object and use that in our backend to create, delete, update and read the data.

Burndown Chart



Story	Points	Assigned To	Sprint	Status	Date
Requirement Analysis	5	Jeyasri, Subarna, Pranav, Wasae	Sprint 1	Complete	3/7/20
Technology Design	5	Jeyasri, Subarna, Pranav, Wasae	Sprint 1	Complete	3/7/20
Database Design	5	Jeyasri, Subarna, Pranav, Wasae	Sprint 1	Complete	3/7/20
Class Design	5	Jeyasri, Subarna, Pranav, Wasae	Sprint 1	Complete	3/7/20
Basic design for backend	1	Jeyasri	Sprint 1	Complete	3/7/20

Backend Design	5	Pranav, Wasae,Subarna	Sprint 2	Complete	3/15/20
Setup AngularJS	2	Jeyasri	Sprint 2	Complete	3/15/20
Setup Spring Boot	2	Subarna	Sprint 2	Complete	3/15/20
Setup SQL script	3	Pranav	Sprint 2	Complete	3/15/20
Wireframe	5	Jeyasri	Sprint 2	Complete	3/15/20
Setup Infra	2	Wasae	Sprint 2	Complete	3/15/20
Infrastructure Design	5	Wasae	Sprint 2	Complete	3/15/20
Setup our GitHub	1	Jeyasri, Subarna, Pranav, Wasae	Sprint 2	Complete	3/15/20
User Authentication API	2	Wasae/Pranav	Sprint 3	Complete	3/22/20
Registration API	5	Wasae/Pranav	Spring 3	Complete	3/22/20
Add Vehicle API	3	Subarna	Spring 3	Complete	3/22/20
Update Vehicle API	3	Subarna	Spring 3	Complete	3/22/20
Get By Id-Vehicle API	1	Subarna	Spring 3	Complete	3/22/20
Get Vehicle API	1	Subarna	Sprint 4	Complete	3/29/20
Get Vehicle for a fixed VehicleType API	5	Subarna	Sprint 4	Complete	3/29/20
Get Vehicles from a Location Type	2	Subarna	Sprint 4	Complete	3/29/20
Remove Vehicle API	4	Subarna	Sprint 4	Complete	3/29/20
Add Location API	1	Subarna	sprint 5	Complete	4/5/20
Get Location API	1	Subarna	sprint 5	Complete	4/5/20
Update Location API	5	Subarna	sprint 5	Complete	4/5/20
Remove Location API	3	Subarna	sprint 5	Complete	4/5/20
Get By Id-Location API	5	Subarna	sprint 6	Complete	4/12/20

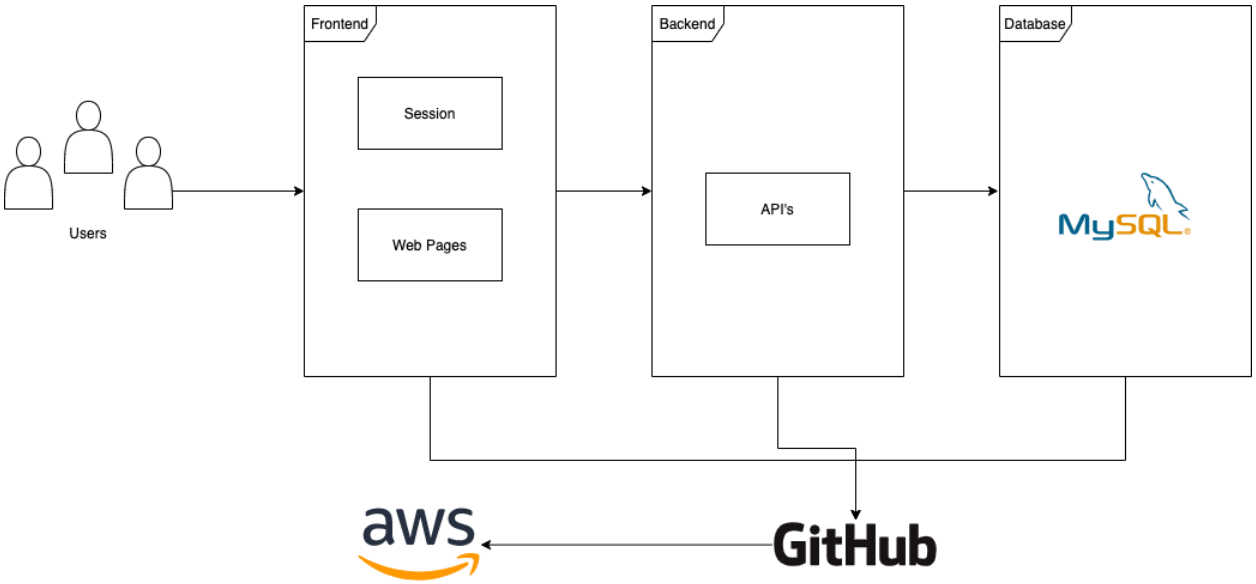
Add User API	3	Wasae/Pranav	sprint 6	Complete	4/12/20
Update User API	2	Wasae/Pranav	sprint 6	Complete	4/12/20
Remove User API	3	Wasae/Pranav	sprint 6	Complete	4/12/20
Add Column Status in reservation table	2	Pranav	sprint 7	Complete	4/19/20
Add column description to price	5	Pranav	sprint 7	Complete	4/19/20
Add CreditCard to Customer	5	Pranav	sprint 7	Complete	4/19/20
Get Price API	4	Wasae/Pranav	sprint 7	Complete	4/19/20
Add Price API	5	Wasae/Pranav	sprint 8	Complete	4/26/20
Update Price API	1	Wasae/Pranav	sprint 8	Complete	4/26/20
Remove Price API	3	Wasae/Pranav	sprint 8	Complete	4/26/20
Get Damage API	3	Wasae/Pranav	sprint 8	Complete	4/26/20
Add Damage API	3	Wasae/Pranav	sprint 9	Complete	5/7/20
Remove Damage API	3	Wasae/Pranav	sprint 9	Complete	5/7/20
Update Damage API	1	Wasae/Pranav	sprint 9	Complete	5/7/20
Add Reservation API	2	Subarna	sprint 9	Complete	5/7/20
Get Reservation API	2	Subarna	sprint 9	Complete	5/7/20
Get by Id Reservation API	5	Subarna	sprint 9	Complete	5/7/20
Update Reservation API	4	Subarna	sprint 9	Complete	5/7/20
Cancel Reservation API with lateFee computation	4	Subarna	sprint 9	Complete	5/7/20
Cancel Reservation API without lateFee	3	Subarna	sprint 9	Complete	5/7/20
Invoice computation API	3	Subarna	sprint 9	Complete	5/7/20

(PUT)					
Invoice get all API	5	Subarna	sprint 9	Complete	5/7/20
Invoice get by Id API	4	Subarna	sprint 9	Complete	5/7/20
Cancel Membership API	3	Wasae/Pranav	sprint 9	Complete	5/7/20
Renew Membership API	5	Wasae/Pranav	sprint 9	Complete	5/7/20
Get VehicleType API	3	Wasae/Pranav	sprint 9	Complete	5/7/20
Update VehicleType API	2	Wasae/Pranav	sprint 9	Complete	5/7/20
Add VehicleType API	5	Wasae/Pranav	sprint 9	Complete	5/7/20
Remove VehicleType API	5	Wasae/Pranav	sprint 9	Complete	5/7/20
Get Employee API	1	Wasae/Pranav	sprint 9	Complete	5/7/20
Add Employee API	3	Wasae/Pranav	sprint 9	Complete	5/7/20
UpdateEmployee API	5	Wasae/Pranav	sprint 9	Complete	5/7/20
DeleteEmployee API	2	Wasae/Pranav	sprint 9	Complete	5/7/20
Special api to compute the estimated prices for all the available vehicles in a location for all vehicle types	3	Subarna	sprint 9	Complete	5/7/20
get available vehicle list for a vehicleType Id, location, and non overlapping pickuptime, actualdropOfftime from reservation	5	Subarna	sprint 9	Complete	5/7/20
Update vehicle APIs and related code for new columns	2	Subarna	sprint 9	Complete	5/7/20

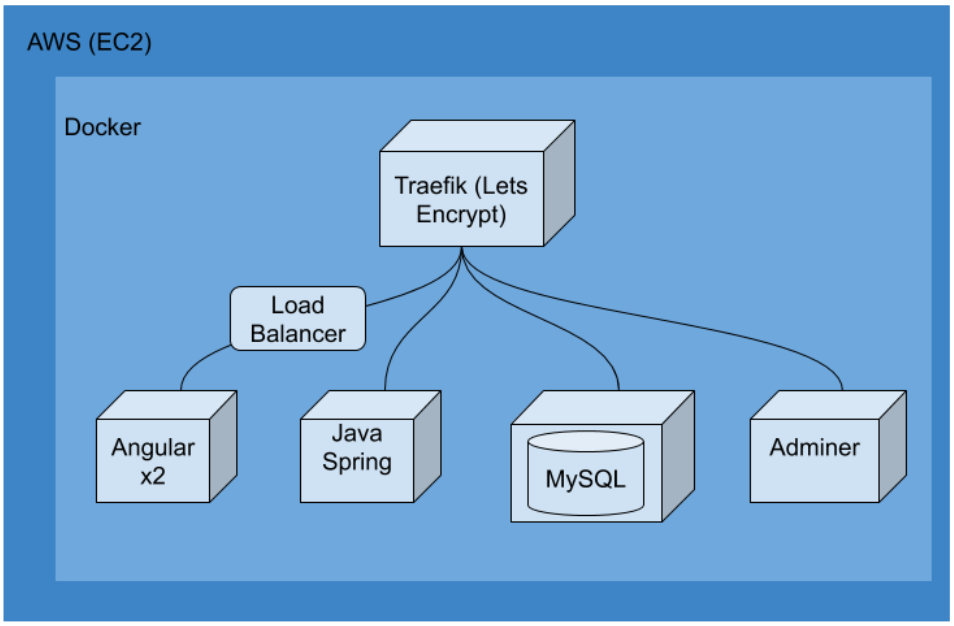
Setup front end routing	5	Jeyasri	sprint 9	Complete	5/7/20
Connect backend and frontend	4	Jeyasri	sprint 9	Complete	5/7/20
Save User Session token	3	Jeyasri	sprint 9	Complete	5/7/20
Login Page	4	Jeyasri	Spring 3	Complete	3/22/20
Signup Page	2	Jeyasri	Spring 3	Complete	3/22/20
Profile Page	3	Jeyasri	Spring 3	Complete	3/22/20
Booking Page	5	Jeyasri	Sprint 4	Complete	3/29/20
Reservation Page	2	Jeyasri	Sprint 4	Complete	3/29/20
Search Page	4	Jeyasri	Sprint 4	Complete	3/29/20
Book page (final page in booking process)	4	Jeyasri	Sprint 5	Complete	4/5/20
Different pages for customer/admin	3	Jeyasri	Sprint 5	Complete	4/5/20
List of Vehicles Page	4	Jeyasri	Sprint 5	Complete	4/5/20
Individual Vehicle Page	2	Jeyasri	Sprint 5	Complete	4/5/20
Add Vehicle Page	5	Jeyasri	sprint 6	Complete	4/12/20
List of User Page	5	Jeyasri	sprint 6	Complete	4/12/20
Individual User Page	3	Jeyasri	sprint 6	Complete	4/12/20
Add User Page	3	Jeyasri	sprint 6	Complete	4/12/20
List of Location Page	2	Jeyasri	sprint 7	Complete	4/19/20
Individual Location Page	5	Jeyasri	sprint 8	Complete	4/26/20
Add Location Page	1	Jeyasri	sprint 8	Complete	4/26/20
List of Reservation Page	5	Jeyasri	sprint 8	Complete	4/26/20
Add Reservation Page	3	Jeyasri	sprint 8	Complete	5/7/20

Architecture and High Level Design

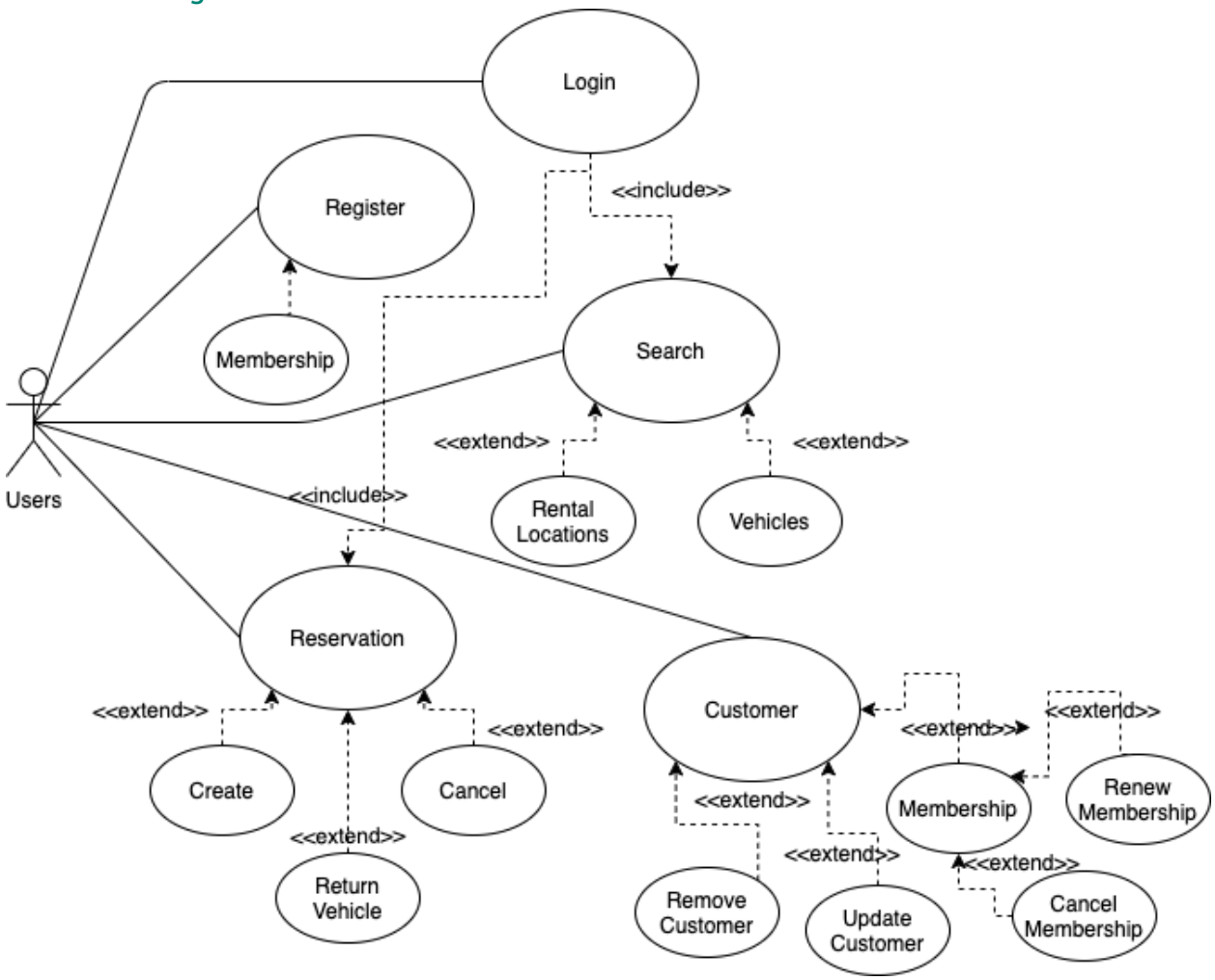
Architecture Diagram

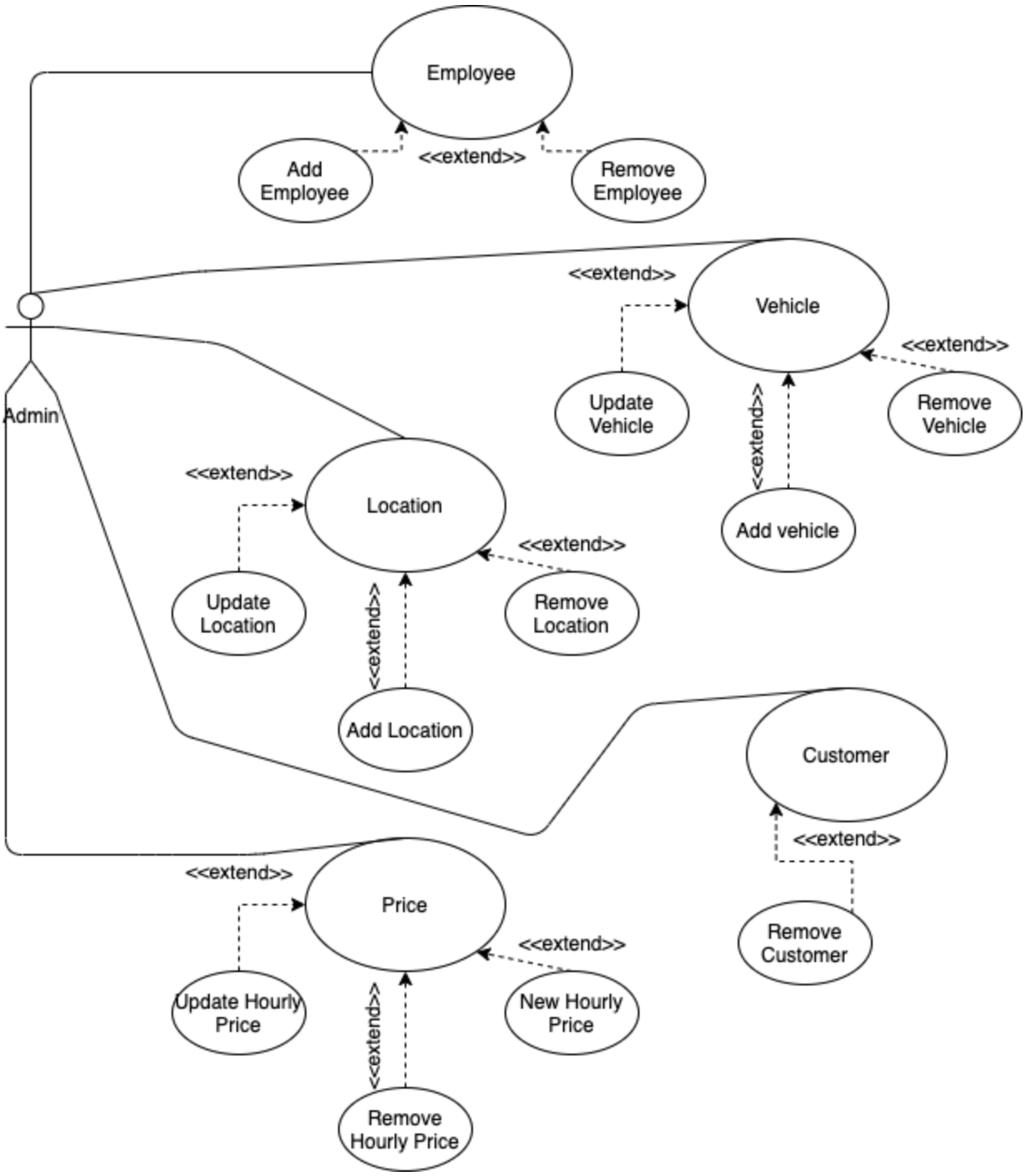


Deployment Diagram

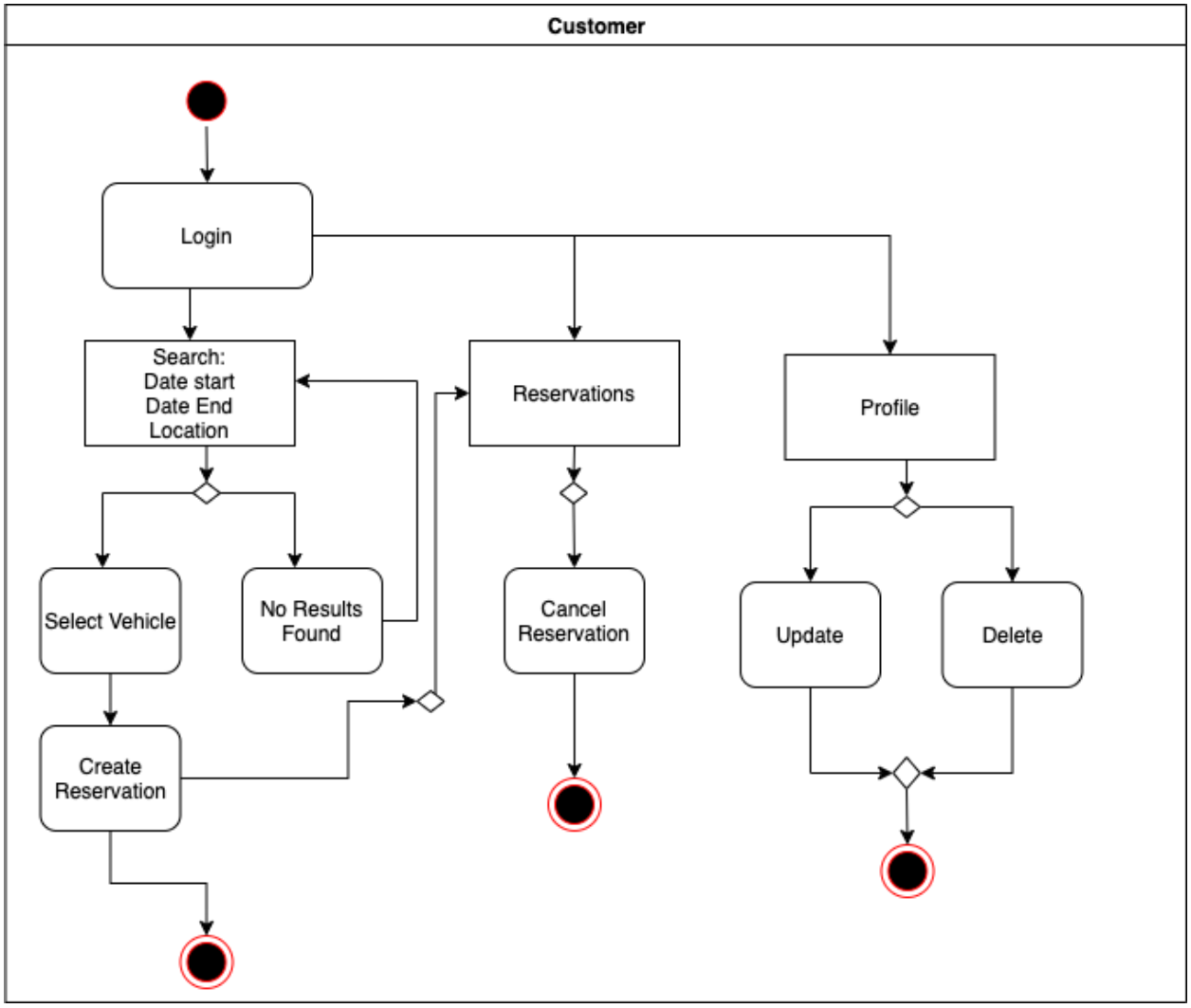

AWS (EC2)
Docker
Traefik (Lets Encrypt)
Load Balancer
Angular x2
Java Spring
MySQL
Adminer

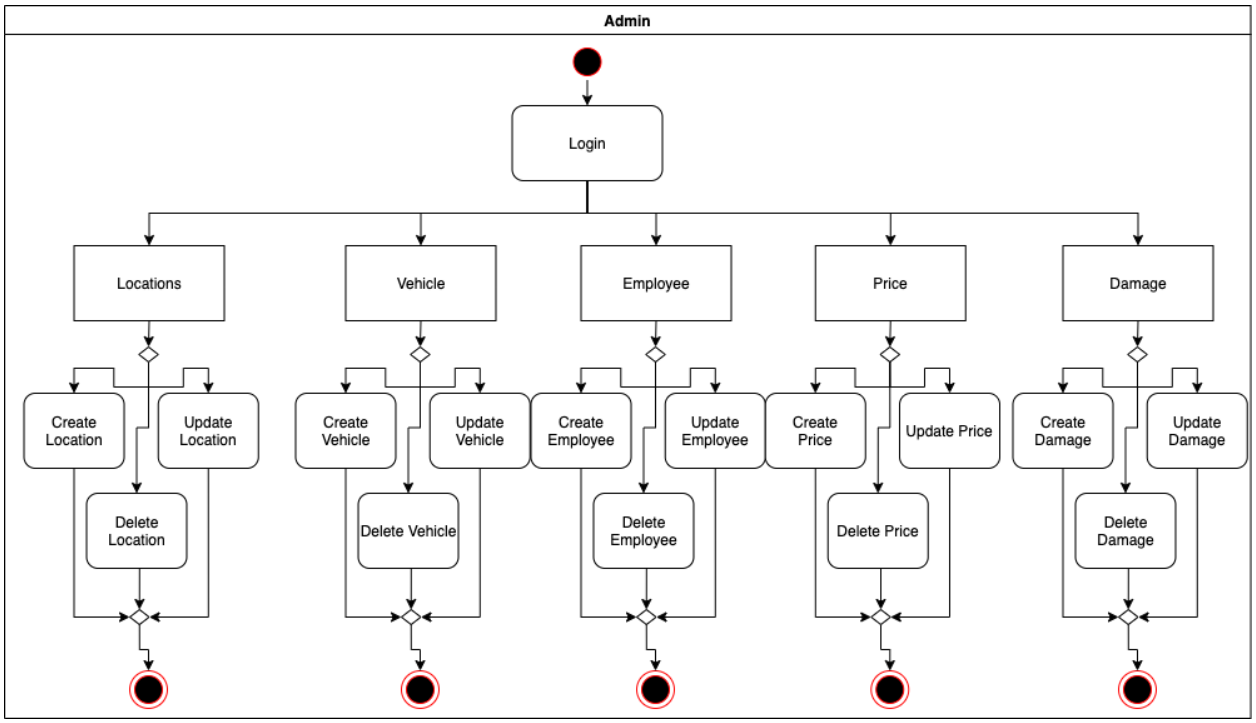
Use-Case Diagram



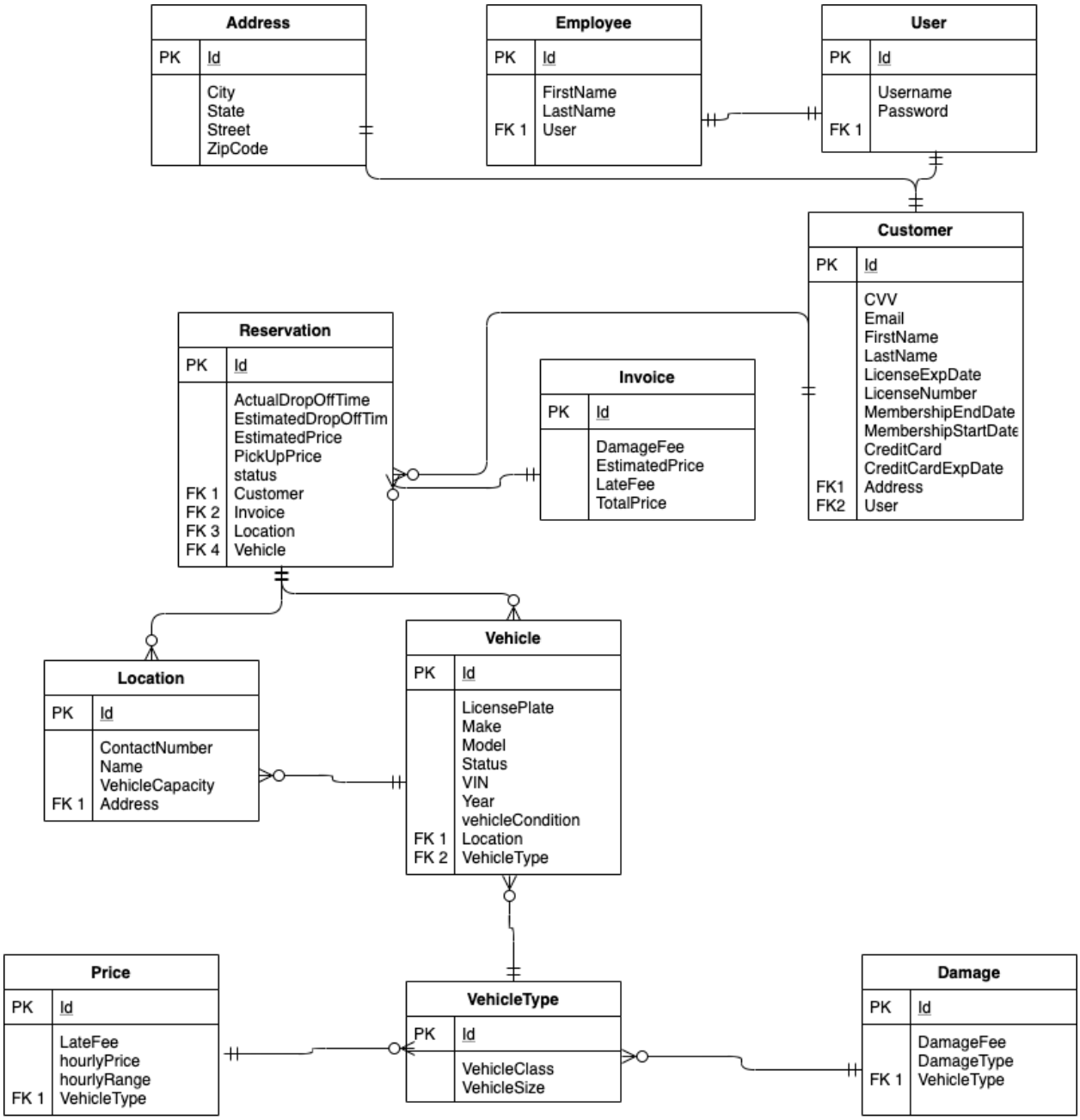

Employee
Add Employee
<<extend>>
Remove Employee
<<extend>>
Vehicle
<<extend>>
Update Vehicle
<<extend>>
Remove Vehicle
Admin
<<extend>>
Add vehicle
<<extend>>
Location
<<extend>>
Update Location
<<extend>>
Remove Location
Add Location
Customer
<<extend>>
Remove Customer
<<extend>>
Price
<<extend>>
Update Hourly Price
<<extend>>
New Hourly Price
Remove Hourly Price

Activity Diagram

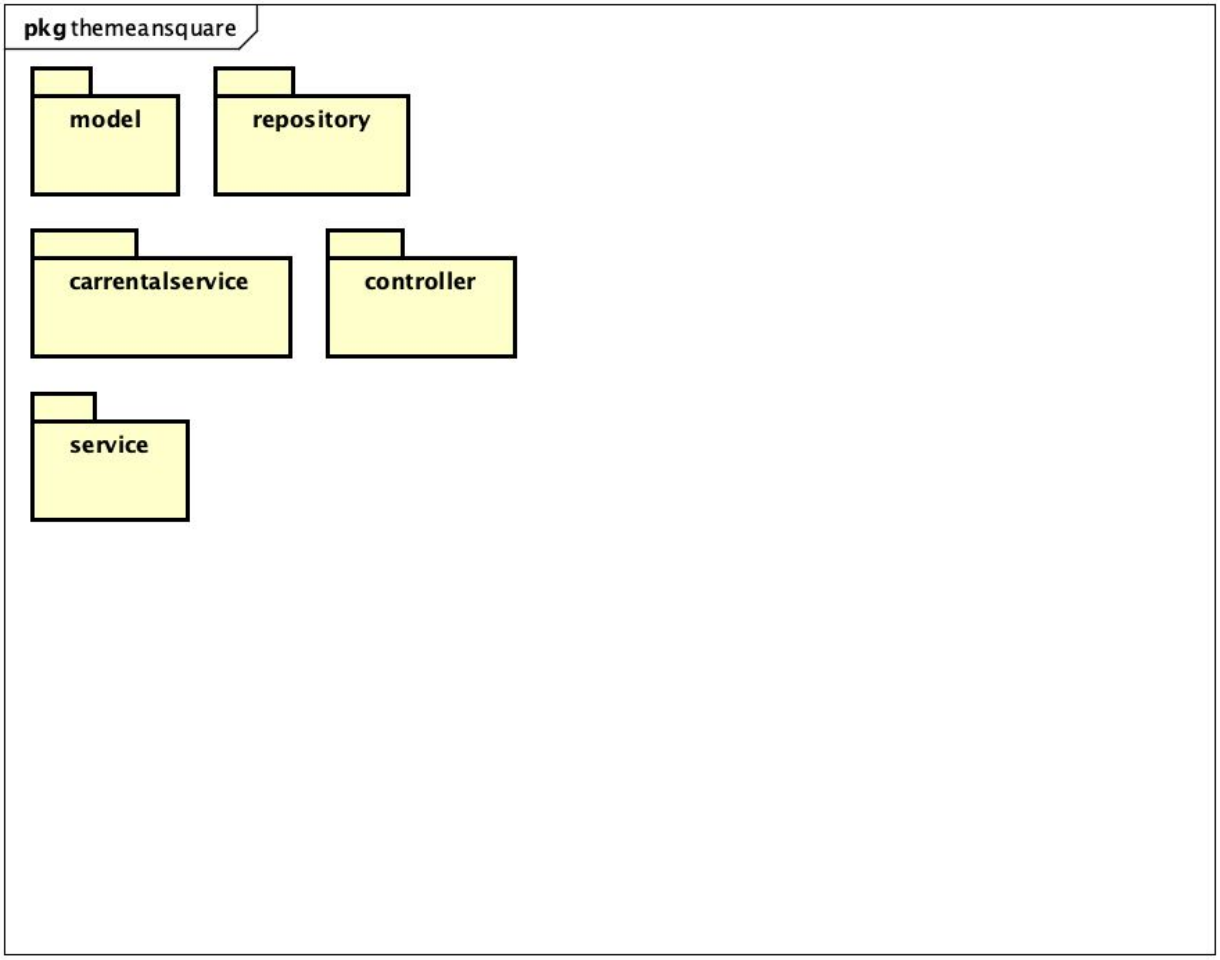


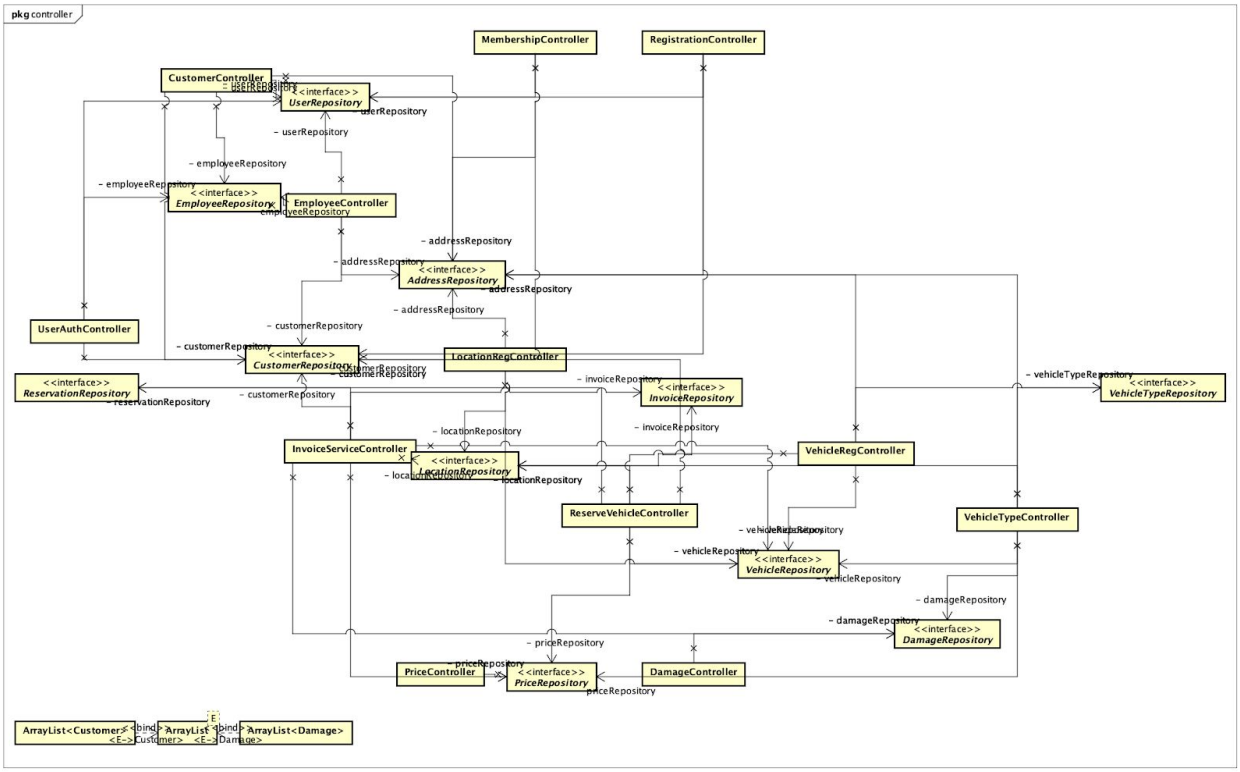


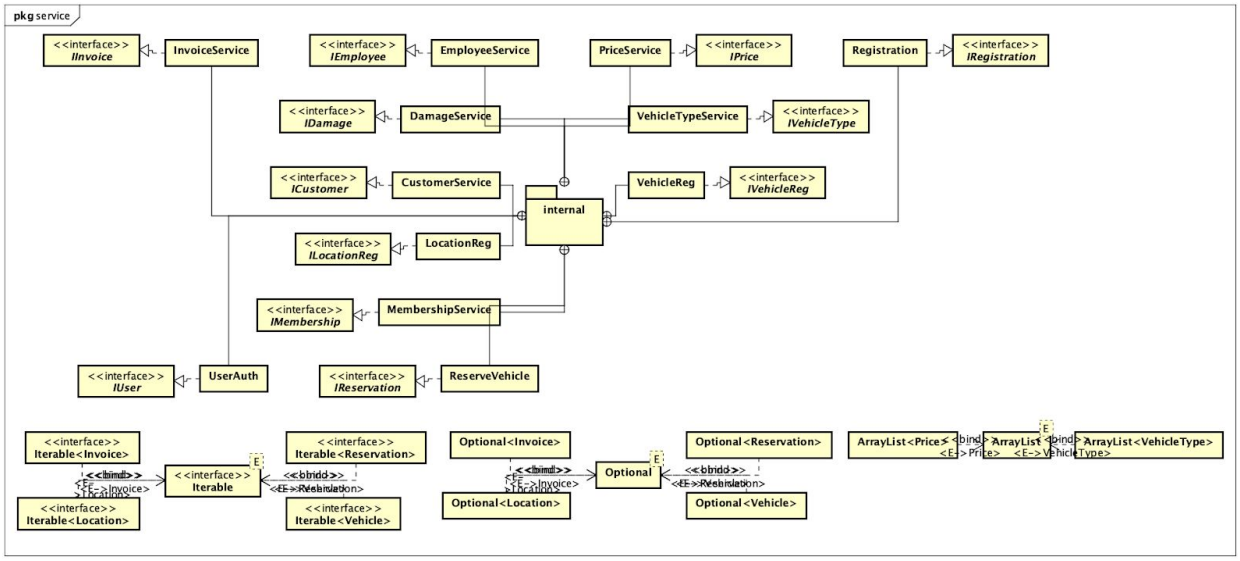
Database Diagram

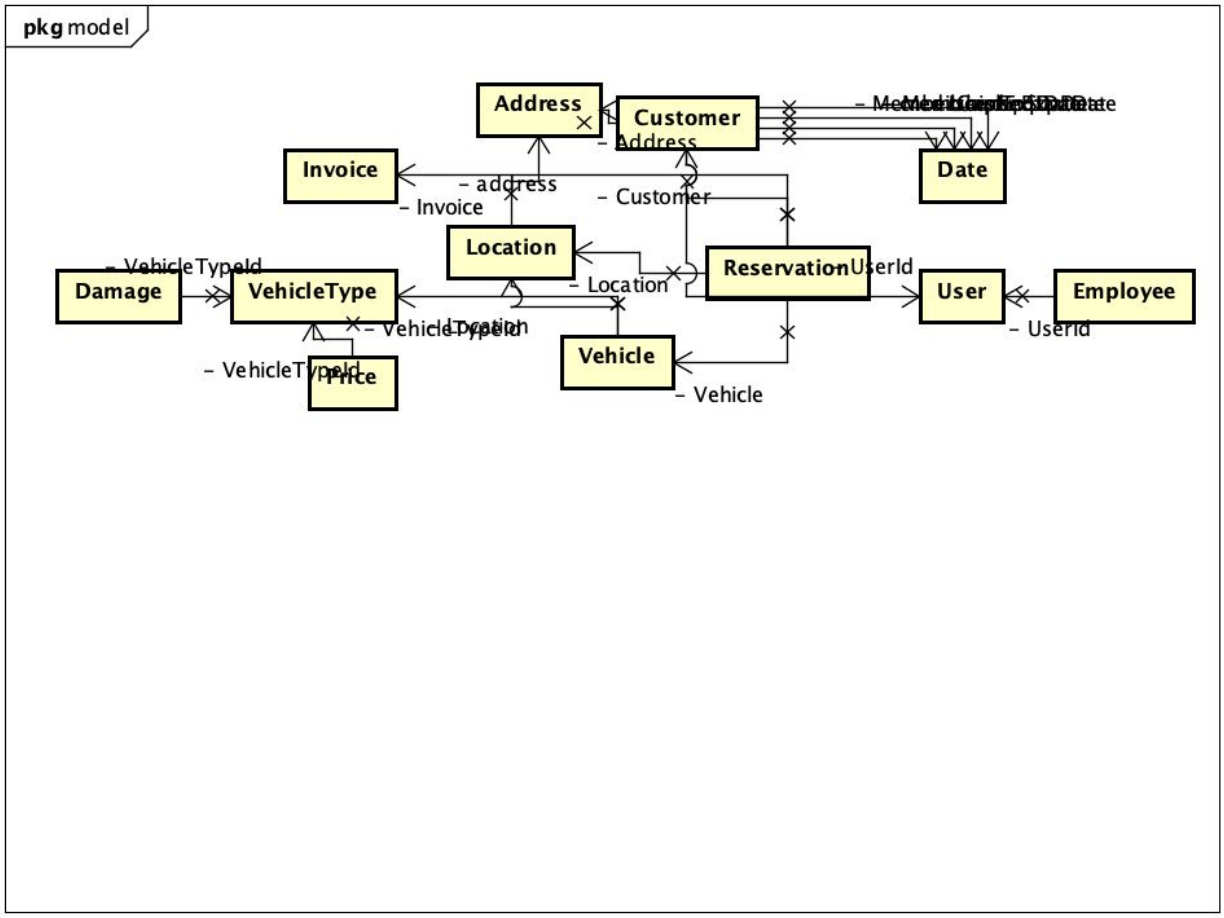


Class Diagram


pkg themeansquare
model
repository
carrentalservice
controller
service


pkg controller
MembershipController
RegistrationController
CustomerController
<<interface>>
UserRepository
EmployeeRepository
EmployeeController
AddressRepository
UserAuthController
CustomerRepository
LocationRegController
ReservationRepository
InvoiceRepository
VehicleTypeRepository
InvoiceServiceController
LocationRepository
VehicleRegController
ReserveVehicleController
VehicleTypeController
VehicleRepository
DamageRepository
PriceController
PriceRepository
DamageController
ArrayList<Customer>
ArrayList<Damage>


pkg service
<<interface>>
IInvoice
InvoiceService
IEmployee
EmployeeService
PriceService
IPrice
Registration
IRegistration
IDamage
DamageService
VehicleTypeService
IVehicleType
ICustomer
CustomerService
VehicleReg
IVehicleReg
internal
ILocationReg
LocationReg
IMembership
MembershipService
IUser
UserAuth
IReservation
ReserveVehicle
Iterable<Invoice>
Iterable
Iterable<Reservation>
Optional<Invoice>
Optional
Optional<Reservation>
ArrayList<Price>
ArrayList
ArrayList<VehicleType>
Iterable<Location>
Iterable<Vehicle>
Optional<Location>
Optional<Vehicle>



Project Development

Project Stack

Area	Tools
Backend	Java Spring-boot
Frontend	AngularJs
Database	MySQL
Containers	Docker
Deployment	Amazon Web Services

Project Deliverables

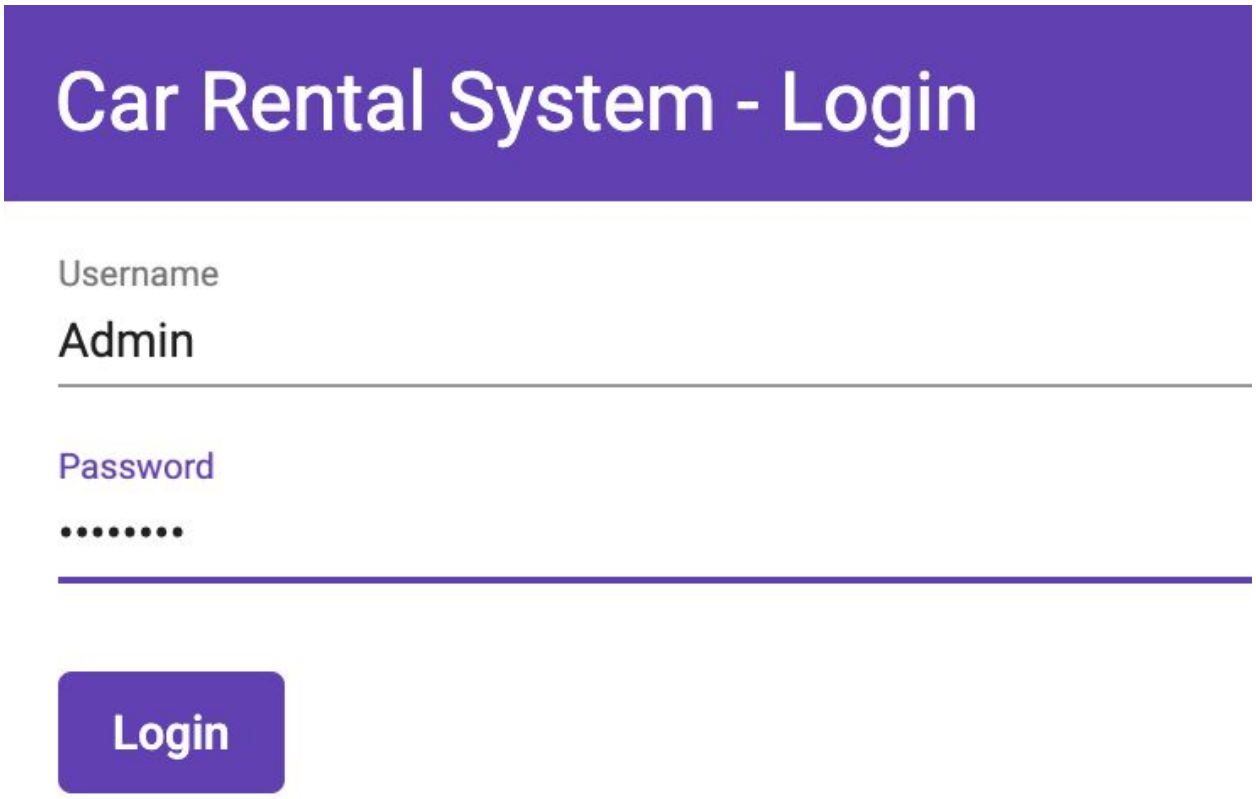
1. The system must allow the system administrator to define and enter into the system vehicle types, such as a small car, full-size car, truck, or a luxury car. Since vehicles are rented per hour, the administrator must be able to set an hourly rental price for each vehicle type. Furthermore, the price should be settable for hourly ranges, for example, 1-5, 6-10hours, etc. The administrator should be able to set a late return fee and a 6-month membership price, as well.
2. The administrator should be able to enter rental locations into the system. Each rental location should have a name, address, and a vehicle capacity (the maximum number of vehicles it can hold). A number of vehicles (see below) are assigned to each rental location.
3. The system must allow the administrator to define and enter into the system individual vehicles. A vehicle should have a defined vehicle type, and a number of properties, such as the make and model, year, registration tag, current mileage, and the time it was last serviced. Also, each vehicle's condition is specified (good, needs cleaning, needs maintenance, etc.). Each vehicle should be assigned to a rental location.
4. The administrator should be able to make changes to any of the information currently stored in the system. For example, it should be possible to change rental prices, reassign vehicles to different locations, modify vehicle properties, etc. It should be possible to remove vehicles, rental locations, etc.
5. A rental system user (a customer) should be able to register with the system. To do that, the user must establish the user name and password, and then provide his/her driver's license state and number, email address, residence address, and a credit card information to be used for payments. The user must pay the initial 6-month membership fee. The user should be able to modify this information and extend his/ her membership.
6. It should be possible to browse and search rental locations and vehicles there, as well as vehicles alone.
7. The user should be able to place a reservation for a vehicle at a selected rental location. The reservation must specify a vehicle type, vehicle pickup time and the length of the rental. The system should check if the requested vehicle would be available at the requested time and place and create a reservation. If a request cannot be granted, the system should suggest a similar rental vehicle at a different location.

8. The user should be able to cancel an existing reservation up to one hour ahead of the scheduled pickup time. Otherwise, a minimum charge of one-hour rental should be applied.
9. The user should notify the system as soon as the car is returned to the rental location. The user is charged for the vehicle time starting with the reservation time and ending at the return time. If a vehicle is returned late, a late return fee may be applied in addition to the rental charge. The user may enter information about the condition of the returned vehicle. Also, the user should be able to provide comments about the vehicle and the rental service in general, if desired.
10. The user should be able to terminate the membership at any time. The membership fee is not refundable.
11. The administrator should be able to terminate the membership of a user, if necessary.
12. The system must be accessible from a common Web browser (assume Google Chrome for now).
13. The system should provide multi-user access, assuring correct concurrent behavior. The system should maintain suitable authorization information and validate access. User authentication should be implemented (by checking user id and password).

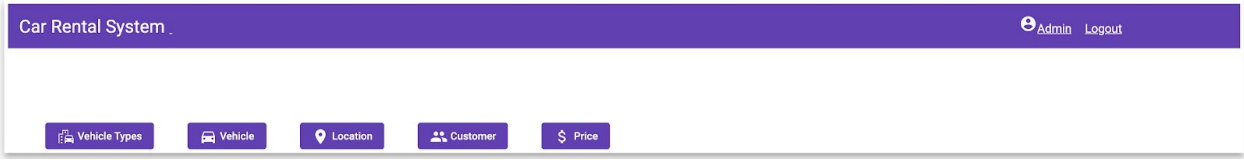
14. The system must have an easy-to-use user interface (UI) with screens designed for each part of the system's functionality and suitable for different types of users (customers, administrators, managers).
15. The system should use a persistent data store.
16. You may use any Tech stack of your choice

Screenshots

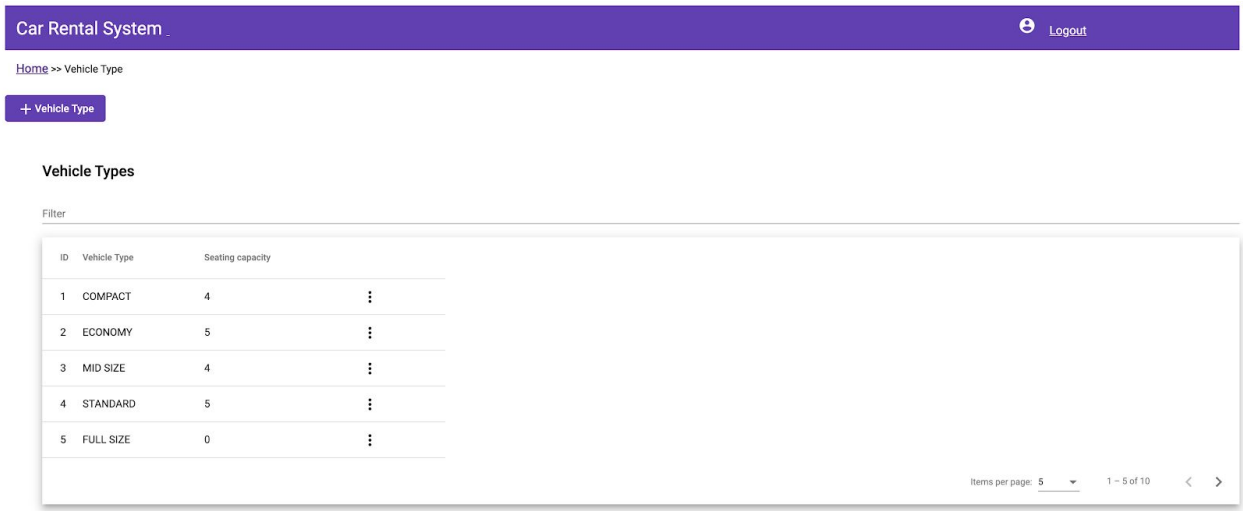
Home Page



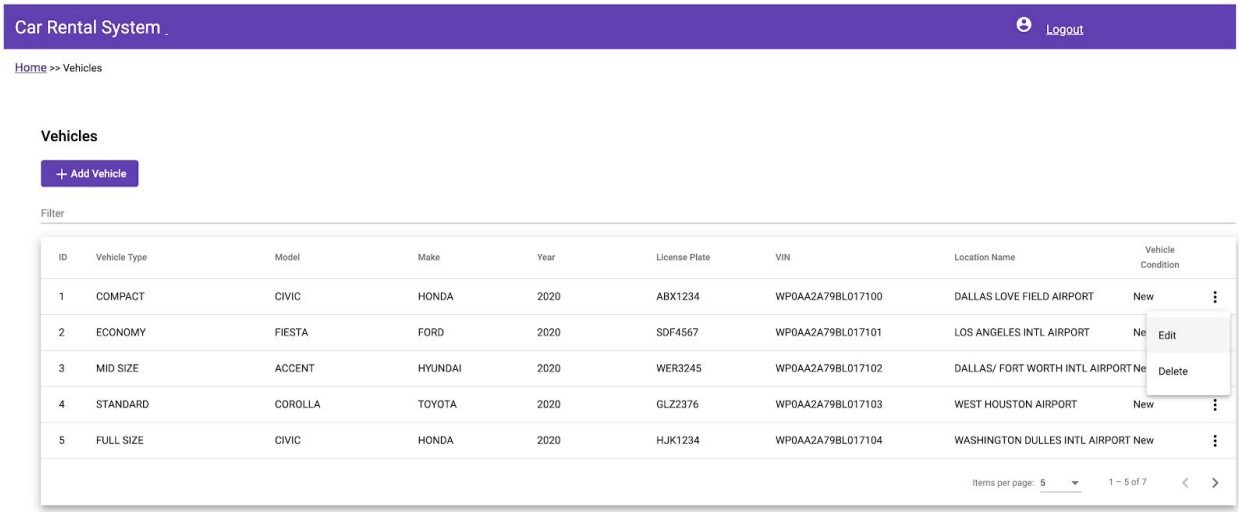
Admin View



Vehicle Type



Vehicles



Locations

Car Rental System Logout

Home >> Locations

+ Add Location

Locations

Filter

ID	Name	Address Line	City	State	Zip Code	Contact Number	Vehicle Capacity
1	DALLAS LOVE FIELD AIRPORT	50 Herb Kelleher Way	Dallas	TX	75235	1234567890	100
2	LOS ANGELES INTL AIRPORT	12 World Way	Los Angeles	CA	90045	1234567891	50
3	DALLAS/ FORT WORTH INTL AIRPORT	10 International Pkwy	Dallas	TX	75261	1234567892	75
4	WEST HOUSTON AIRPORT	15 Groschke Rd	Houston	TX	77094	1234567893	30
5	WASHINGTON DULLES INTL AIRPORT	15 Saarinen Cir	Dulles	VA	20166	1234567894	35

Items per page: 5 1 – 5 of 8

Customer

Car Rental System Logout

Home >> Customers

Customers

Filter

ID	Name	Address Line	License Information	License Expiration Date	Start Date	End Date	Email
1	KEERTHIKUMAR RAVICHANDRAN	700 CAMPBELL RD, RICHARDSON, TX, 75080	F1234554	2021-12-12	2020-05-05	2021-05-05	keerthi@gmail.com
2	SURESH KUMAR	800 RENNER RD, RICHARDSON, TX, 75080	F2345611	2021-12-12	2020-05-05	2021-05-05	suresh2234@gmail.com
3	VARADHA CHANDRASEKARAN NIVEDITHA	6547 CANOGA AVE, CANOGA PARK, CA, 91303	F9764521	2021-12-12	2020-05-05	2021-05-05	nivi07@gmail.com
4	Wasae Qureshi	random street, random city, ca, 94086	747324	1994-05-22	02020-05-07	02020-11-07	different5@gmail.com

Items per page: 5 1 – 4 of 4

Pricing

Car Rental System Logout

Home >> Price List

+ Add Price

Price List

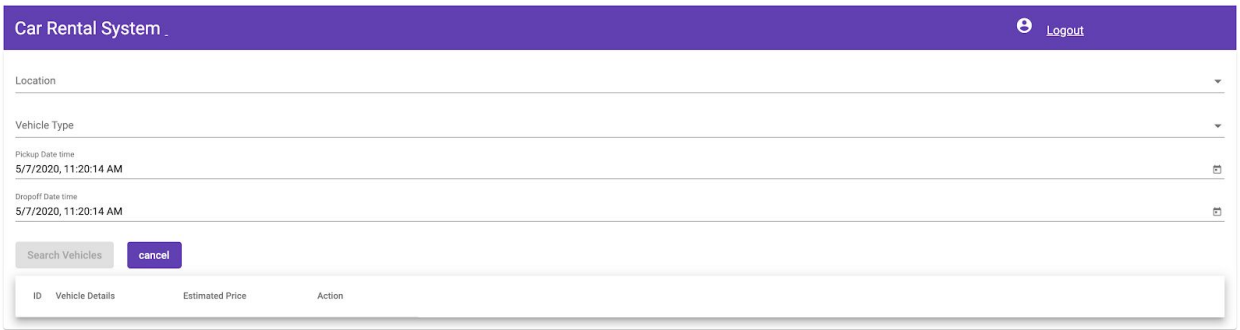
Filter

ID	Vehicle Class	Seating capacity	Late Fee	Hourly Fee	Discount Range (in hours)
1	COMPACT	4	$50	$50	upto 5 hours
2	COMPACT	4	$50	$40	upto 10 hours
3	ECONOMY	5	$20	$30	upto 5 hours
4	ECONOMY	5	$20	$20	upto 10 hours
5	MID SIZE	4	$30	$10	upto 5 hours

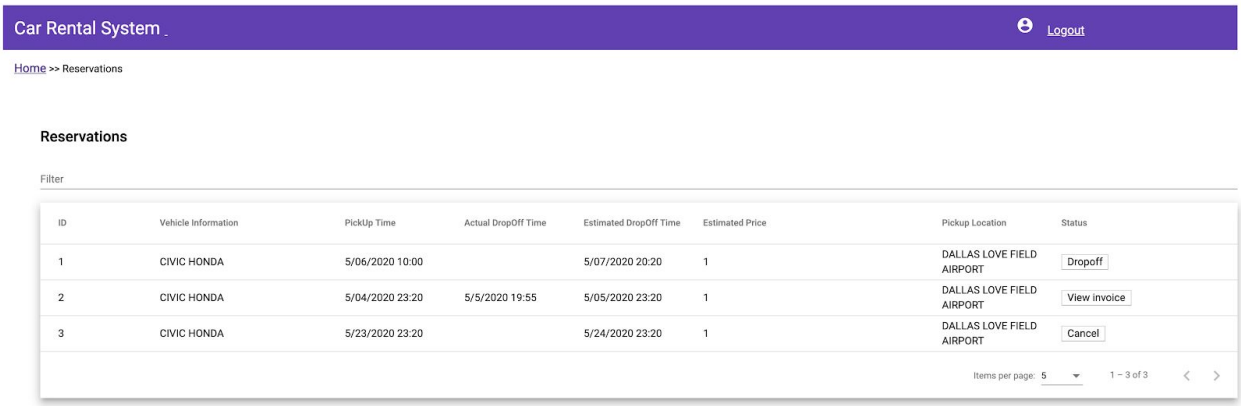
Items per page: 5 1 – 5 of 8

Customer View

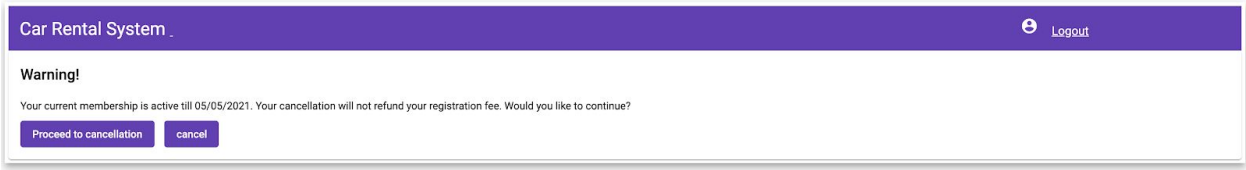
Reserve Vehicle



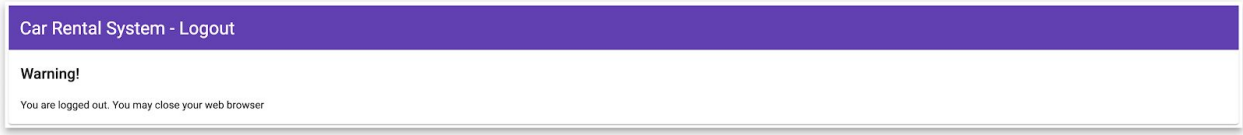
View Reservations



Cancel Membership



Logout



Project Difficulties

1. Difficulties with using JAVA and building out API's. This was solved with the use of Java Spring-boot
2. We had difficulties with AWS load balancer, we ran into issues of it having issues with out custom domain.
3. A prominent difficulty was with our frontend. We ran into a lot of issues with spring and angular having Cross Origin errors
4. Difficulty designing frontend, as none of us were very good at frontend development

XP Core Values

Communication

During this project we had various forms of communications. We would utilize WhatsApp for quick communication between each other, as well as having a group chat.

We also utilized a Google spreadsheet that allowed us to track our stories/ tasks that needed to be completed.

And Finally we would always confirm with another team member before we pushed our work and merged with the master branch of our github repository.

Communication was a key aspect to us building out a car rental application in the short time that was given.

Deployment

Our project is deployed on Amazon web services

The frontend, backend, database have been containerized and deployed on an EC2 instance. t2.large

	i-0c570b4a6c38b08f5	t2.large	us-west-2b	running	Unable to d…	*None*	ec2-35-155-65-155.us-…	35.155.65.155

Each component have been containerized for easy and quick deployment

A copy of the docker yaml file can be found here

```
ubuntu@ip-172-31-23-227:~$ docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS                                          NAMES
32a864d9fd52        adminer                "entrypoint.sh docke…"   5 hours ago         Up 5 hours          0.0.0.0:8010->8010/tcp, 8080/tcp               prod_adminer_1
5ecb05634673        prod_java_spring       "java -jar car-renta…"   5 hours ago         Up 5 hours          0.0.0.0:8020->8020/tcp                         prod_java_spri
ng_1
12841e04f7da        mysql                  "docker-entrypoint.s…"   5 hours ago         Up 5 hours          0.0.0.0:3306->3306/tcp, 33060/tcp              prod_db_1
fc7a375e50c1        prod_angular-service   "nginx -g 'daemon of…"   5 hours ago         Up 5 hours          80/tcp, 0.0.0.0:8030->8030/tcp                 mn-app
35f09e3f4f4b        traefik:1.7            "/traefik"               5 hours ago         Up 5 hours          0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp       traefik
```

Test Cases

- CustomerControllerTest
    - getAllCustomers
    - getCustomerInfo
    - updateCustomer
    - removeCustomer
- DamageControllerTest
    - getDamageForVehicleType
    - addDamage
    - updateDamage
    - deleteDamage
- EmployeeControllerTest
    - createEmployee
- InvoiceSearchControllerTest
    - computeInvoiceById
    - getInvoice
    - getInvoiceById
- LocationRegControllerTest
    - addLocation
    - getLocation
    - getLocationById
    - DeleteLocation
    - updateLocation
- MembershipControllerTest
    - cancelMembership
    - renewMembership
- PriceControllerTest
    - addPrice
    - deletePrice
    - cancelPrice

    - getPrice
- RegistrationControllerTest
    - register
- ReserveVehicleControllerTest
    - Reserve
    - getReservation
    - getReservationById
    - cancelReservation
    - updateReservation
- UserAuthControllerTest
    - getEstimatePriceForVehicle
- VehicleRegControllerTest
    - getVehicle
    - addVehicle
    - updateVehicle
    - removeVehicle
- VehicleTypeControllerTest
    - getVehicleType
    - updateVehicleType
    - removeVehicleType

Project Contributions

Name	Contribution
Pranav Lodha	Backend API's, Database, Documentation
Wasae Qureshi	Backend API's, Infrastructure, Documentation
Jeyasri Subramanian	Frontend Development, Integrations, Documentation, Wireframes
Subarna Chowdhury Soma	Backend Framework Setup, Backend Developer, Documentation

Future Updates

- Currently we would want to move our project stack to python and flask.
- Reduce Complexity of Database Design
- Adjust Frontend Design

References

- https://bezkoder.com/angular-spring-boot-crud/
- https://www.javatpoint.com/angular-spring-crud-example
- https://www.javaguides.net/2019/06/spring-boot-angular-7-crud-example-tutorial.html
- By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=17119753
- By GitHub - https://github.com/logos, Public Domain, https://commons.wikimedia.org/w/index.php?curid=41223578
- By Amazon.com Inc. - Amazon, Apache License 2.0, https://commons.wikimedia.org/w/index.php?curid=62382835