# Overview

Car Rental Application has been written with the help of Spring Boot Starter project. I am using JPA with in memory h2 database. It is built on java 8 and maven 4.

# Business Requirement

Car rental application which helps you to handle

1. Register User who can the tell car availability
2. Rental user who rented the car
3. Car registration
4. Car booking

# Endpoints and their information

As per requirement there are 9 endpoints for my application which are mentioned

|  |  |  |
| --- | --- | --- |
| Endpoint | Usage | Example of Request |
| <http://localhost:9090/cars/registerCar> | For Registering Car | {      "carNumberPlate": "ABO129",      "carRegisterUser":"888888888"  } |
| http://localhost:9090/cars/registerAvailability | For register availability of Car | {      "numberplate": "ABO199",      "pricePerHour":11,      "fromDate":"2025-09-04T18:06:22.000Z",      "toDate":"2015-09-04T18:06:22.000Z"  } |
| <http://localhost:9090/cars/searchCar> | Search the car with required date and maximum price | {      "maximumPricePerHour": "229.9",      "fromDate":"2020-09-04T18:06:22.000Z",      "toDate":"2021-09-04T18:08:22.000Z"  } |
| <http://localhost:9090/registerusers/createCarRegisterUser> | Creating Car Register User | {      "userName": "User1",      "firstName":"Jphn",      "lastName":"Dk",      "password":"pas12345"  } |
| http://localhost:9090/rentalusers/createCarRentalUser | Creating Car Rental User | {      "userName": "User1",      "firstName":"Jphn",      "lastName":"Dk",      "password":"pas12345"  } |
| <http://localhost:9090/bookings/createCarBooking> | For Creating Car Booking User | {      "carBookingCost": "29.6",  "fromDate":"2020-09-04T18:06:22.000Z",  "toDate":"2020-09-18T18:06:22.000Z",  "carNumberPlate":"ABO124",  "carBookingUserName":"User1"  } |
| http://localhost:9090/bookings/searchBookedCars | Search Booked Cars for given dates, Car, Rental User | {  "fromDate":"2003-04-25T14:05:15.953Z",  "toDate":"2021-04-25T14:05:15.953Z",  "carNumberPlate":"ABO124",  "carRentalUser":"User1"  } |
| http://localhost:9090/bookings/searchTotalPayments | Search Total Payment for given dates | {  "fromDate":"2020-08-17T06:00:15.953Z",  "toDate":"2020-09-20T19:08:00.953Z"  } |
| http://localhost:9090/bookings/searchBookedCarsPerHour | Booked Cars Per Hour | {  "fromDate":"2021-09-17T06:00:15.953Z",  "toDate":"2020-09-18T16:08:00.953Z"  } |

There are many validations in system including normal validation, hibernate validations to validate the models

# Logical Data Model

userName: string

password: string

firstName: string

lastName: string

**Car\_Register\_User**

numberPlate: string

fromDate: date

toDate: date

pricePerHour: double

**Car**

**1**

**.**

userName: string

password: string

firstName: string

lastName: string

**Car\_Rental\_User**

id: int

numberPlate: string

fromDate: date

toDate: date

bookingDate: date

bookingLost: date

**Car Booking**

**1**

**.**

**1**

**.**

# Testing

I am junit Jupiter together with springbootstartertest and mockito to test my application.There are around **110** testcase which checks all different aspects including basic, negative, exception etc. Currently Unit Testing coverage currently stands at approx. 85%. Here is the attached report



# Some Facts About Application

1. Global Exception handler has been used to send the error back gracefully
2. Transactions are marked with Serializable isolation level. This has the highest level of isolation with Read&Write Locks and with the range locks. The transactions which are read only are marked with @Transactional(readOnly = **true**)
3. The code is written by following Clean Code Principles and many other principles such as SOLID, DRY,DIE etc
4. Some of the Design Patterns which have been used are/
   * 1. Builder Pattern
     2. Singleton Pattern
     3. Strategy Pattern
     4. Bridge Pattern
     5. Template Method Pattern
5. Sonarlint warnings are taken care.

# How to run the Application

There no additional configuration is required to run the application

## Build the Application

You can build the project by using maven. Below are the steps

* 1. Unzip the attached project
  2. Go to the location of where project is unzipped
  3. Run the command (ensure mvn is available)

mvn clean install

## Run the Car Application

To run the application, there are below 2 ways

1. Using mvn spring boot
   1. Unzip the attached project
   2. Go to the location of where project is unzipped
   3. Run the command (ensure mvn is available)

mvn spring-boot:run

1. Build the project and run the executable jar
   1. Unzip the attached project
   2. Go to the location of where project is unzipped
   3. Build the project by running below command

mvn clean install

* 1. Go to the target folder and run below to start the application

Java -jar carrentalapplication-0.0.1-SNAPSHOT.jar

## Run the test application

To run the testapplication you have to run the TestApplication.java present at package “com \testapplication”. I have used WebClient and flux to parallely call the different endpoints. It will right now run for 1 minutes which can be increased by changing *CONCURRENT\_TESTING\_MIN in TestApplication.java*

## Run the tests

Test will be running only with separate profile named as testProfile. Below are the steps

1. Unzip the attached project
2. Go to the location of where project is unzipped
3. Please ensure that this profile “testProfile” is there in <activeProfiles> in settings.xml. If not please add
4. Run the test by below command

mvn test -PtestProfile

Tests will not be run in other profiles, so this makes build faster.

## Containerize the application

Please follow below steps

1. Go to src folder where the project is unzipped.
2. Build the project
3. There is DockerFile which will be used for building the image
4. Ensure docker is already installed, if not install it
5. Build the image by opening cmd from src folder and running below command

docker build -t carrentalapp .

1. Run the container by opening cmd from src folder

docker run -p 8080:8080 -t carrentalapp