Enterprise Architecture Project

Implement a **car rental application** with the following requirements:

Functional requirements:

Car functionality:
We can add, remove, update and find cars in the car rental application. You can search cars on car type, brand, price. We can have multiple cars of the same brand and type. For example we have 15 Ford F1 Pickup trucks. Every individual car has a unique license plate.
The system keeps track of how many cars of a certain brand and type are available in the car fleet.

Customer functionality:
We can add, remove, update and find customers in the car rental application. You can search customers on customernumber, name and email address. Every customers has an unique customernumber.

Reservation functionality:
A customer can reserve a certain car type for a certain period.

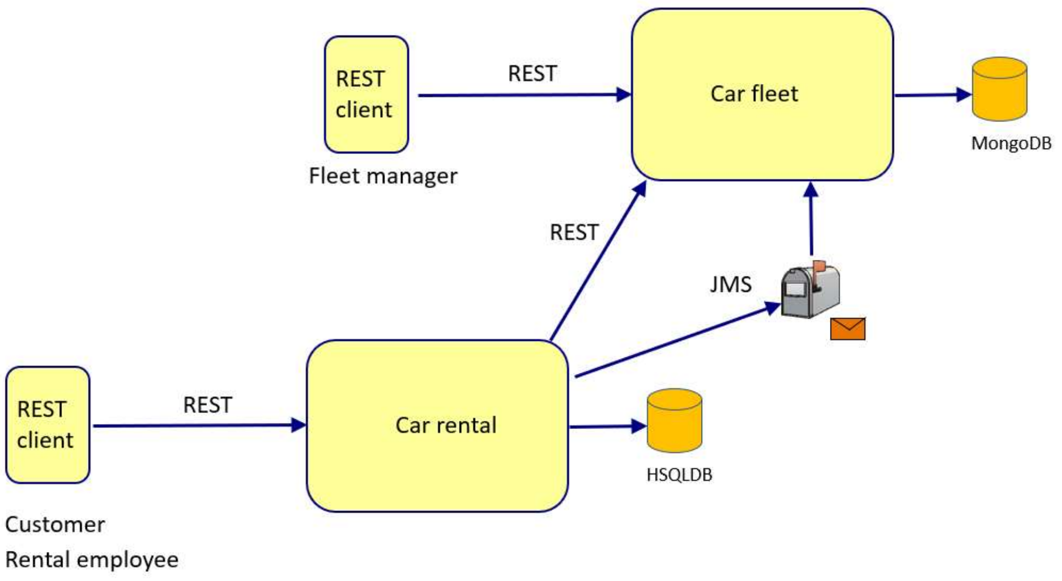
Borrowing functionality.
A customer can pickup the car at the car renting location
A customer can return the car to the car renting location
The customer pays for the car when the car is returned.
Rentals can only be paid by creditcard.

We can retrieve the following information:

- Get all data from a customer(including reserved cars, rental history, payments made)
- Get all data from a particular car including the rental history.
-

Architecture requirements

The Car rental system consist of 2 separate applications:



Application 1: Car fleet application

- Contains the collection of cars with all car functionality described above. This functionality can be accessed through a REST interface. Write a REST client that calls the functionality of this application.
- Uses a Mongo database

Application 2: Car rental application

- Contains all functionality that a customer needs in order to search and reserve a car. It also contains the functionality for the rental employee to handle car pickups and car returns. This functionality can be accessed through a REST interface. Write a REST client that calls the functionality of this application.
- Uses a HSQLDB database
- Use JPA for database access
- Add logging (console and file) to this application at different logging levels.
- This application should be monitored with Grafana.
- Use scheduling so that the application prints every 20 seconds an overview of all cars including their state (rented out, available).
- The following data can be configured in *application.properties*:
    - Maximum number of cars a customer can reserve.
    - The URL to the car fleet application
- Use @ConfigurationProperties to read these properties
- All domain logic is tested in unit tests
- All queries in the repositories are tested.
- Rental functionality is tested with RestAssured
- Anytime a car is reserved, the car rental application sends a JMS message using ActiveMQ to the car fleet application. The car fleet keeps track of how many cars are available of a certain brand/type.
- The car fleet application keeps the data of all cars. It is important that the car rental application always calls the car fleet application if it needs car data.
- When there are less than 3 cars available of a certain brand/type combination, the system should send an email to the fleet manager (Do net send a real email, but use System.out.println())

Use the best practices we learned in this course. (DTO's, REST interface, test, etc.)

<u>**Deliverables:**</u>

- All code that you write. Your code should be in 4 separate Maven projects (Car fleet, Car rental and their corresponding REST clients) that can be opened with IntelliJ.
- A video presentation that demos your application.

Presentation

- Maximal 20 minutes
- Your presentation should only demo your application (Do NOT show and explain your code)
  - Demo of the following
    - Show running your rest client
    - Show the content of the database
    - Show that a message is sent
    - Show running your different tests (unit, integration, acceptance)
    - Show your configuration file(s)
    - Show your logging file(s)
    - Show your Grafana dashboard.

This project is done individually. You can discuss with your fellow students, but all your demo code and material needs to be created by yourself.

The due date for this project is **Thursday at 12:30 PM** (projects that are submitted later will lose points).

On Tuesday and Wednesday everyone needs to be in class in the morning and in the afternoon..

How to submit your project?

The project needs to be submitted in the assignment section is sakai. You can submit the code and the slides, but you cannot upload your video (size is too large). Upload your video to google drive, make it **available to everyone**, and submit the link as part of your assignment. **It is important that the link is available to everyone.**

The grading of this project is based on:

- The quality of your design and code
- The quality of the demo presentation.
- Did you implement all requirements?

Extra credit

Add functionality that customers can create a frequent renter account. Every time a customer rents a car, this customer gets frequent renter points. There are 3 different frequent renter accounts: Bronze, Silver and Gold. If you have more than 50 points in Bronze, you get an upgrade to Silver. . If you have more than 100 points in Silver, you get an upgrade to Bronze. In Silver, you get twice the number of points that you normally would get in bronze. In gold, you get three times the number of points that you normally would get in bronze. Implement the necessary functionality like add rental points, get customer account info, login to your account, create account.