CS393 - Project

Assignment Date : 5.12.2022
Deadline : 30.12.2022 23.59 PM

- This assessment must be prepared and finalized individually or as a team (3 people).
- You must **upload** your project into LMS as an archive file or you can upload a text file and write your project's GitHub URL in it. But in this case, you should add me as a collaborator. Never share your project with the public.
- I will **not** accept any meeting requests and will not answer questions after 6 pm the day before the deadline. (December 29)
- I will also share a document for grading rules.

PROJECT DETAILS

Suppose that you are working in a company as a software developer. Your company is developing a Car Rental Application. A web application and a mobile application will be developed. You are included in the team that is responsible for **developing server-side services**. Your back-end application will provide services to these applications.

Your team is responsible for developing all required Restful services which are required for web/mobile applications.

Base Technical Requirements and Rules

1. You should create a Spring Boot web application. You should use JPA and Spring Data JPA. You are not allowed to use pure JDBC or Spring JdbcTemplate. But you can use native SQL in the Data JPA Repositories.
2. Generate an API documentation for your Restful services using OpenAPI (Swagger). Be careful, you will not develop a web application, you will use Swagger and it will produce the required API documentation.
3. You are not allowed to use Entity Objects for the input and output of Restful services. Your Services should only accept and return Data Transfer Objects. You should use a mapping library like mapstruct (https://mapstruct.org) for such transformations.
4. Database selection is free of choice, but don't use any database specific command in your project. Switching to another database must be done easily by changing application.properties file and the dependencies.
5. You should initialize your database automatically with at least one record in each table. You are free to select any method you like for initialization.
6. Draw a UML class diagram for your model classes.
7. Develop seperate service classes for each entity. For example: ReservationService, CarService, MemberService etc.
    a. All service classes should contain base methods for related Entity. For example your MemberService should contain methods for creating, reading a member and returning all members
    b. If you need, you can add extra methods for updating entity.
    c. You should also add all other required methods according to the business requirements (see below)
8. Develop seperate controller classes according to the Resful Services requirements (listed below). For example: ReservationController, CarController, CarServiceController, MemberController etc
9. Use 3-layer architecture: Controller-Service-Repository. You can use Service beans in the Controller, you can use Repository in the Service beans. But you can not use Repository in the RestController classes.
10. Develop Junit tests for testing all of your service methods

11. Apply resource naming rules.
12. It is forbidden to use FetchType.EAGER
13. It is forbidden to use CascadeType.ALL and CascadeType.REMOVE. You can use the other cascade types depending on your requirements.

Required Restful Services

Develop all necessary Restful services to satisfy all the UI service requirements listed below.

Service short definition	Service input/output parameters and business rules
Search available cars	**Input parameters:** • car type (standard, suv, etc) • transmission type **Return Value :** available car list. Each object should contain following values: • Brand • Model • Car type • Mileage • Transmission type • Barcode number **Business Rules:** Your service should search only cars with has status Available **Return Http Status Code:** 200 => if available cars are found 404 => if no available cars
Make a reservation	**Input parameters:** • Car barcode number • Day count • Member id • Pick-up location code • Drop-off location code • Additional Equipment list • Additional Service list **Return Value :** • Reservation number • Pick-up date-time • Drop-off date-time • Pick-up location code and name • Drop-off location code and name • Total amount **Business Rules:** Before making reservation you should check the car status again

	1. If status is not "Available" you can't complete the reservation. You should send 406-Not Acceptable status code. In this case don't create & save any Reservation object. 2. If status is still Available you can complete reservation and send 200 as Http status code a. Generate a unique 8 digits reservation number in the Service layer. b. Update car's statu as Loaned c. Assign reservation create date as sysdate d. Assign pick-up date as sysdate+1 e. Assign drop-off date as pick-up date + dayCount f. Compute total amount : dayCount * car daily price + additional services price + additional equipment price g. Total amount should not save into the database, must be calculated when requested **Return Http Status Codes:** 200 – if reservation successful 206 – if selected car is not available
Get all rented cars	**Input parameters:** none **Return Value :** List of Loaned and Reserved cars. Each object should contain following values: • Brand • Model • Car type • Transmission type • Barcode number • Reservation Number • Member name • Drop-off date and time • Drop-off location • Reservation day count **Business Rules:** none **Return Http Status Codes:** 200 – OK 404 => if no rented/reserved cars
Add an additional service to a reservation	**Input parameters:** • Reservation number • Additional service code **Return Value :**

	boolean **Business Rules:** If adding process is successful return true. If service is already added or service is not found return false **Return Http Status Codes:** 200 – OK – if service added successfully 404 – Not Found – if selected service is not found 500 – if an exception thrown
Add an additional equipment to a reservation	**Input parameters:** • Reservation number • Additional equipment code **Return Value :** boolean **Business Rules:** If adding process is successful return true. If equipment is already added or equipment is not found return false **Return Http Status Codes:** 200 – OK – if equipment added successfully 404 – Not Found – if selected equipment is not found 500 – if an exception thrown
Return the car	**Input parameters:** • Reservation number **Return Value :** boolean **Business Rules:** • Return true if vehicle return completed successfully o Update reservation as follows: ▪ Status=COMPLETED ▪ Return date = system date o Update reservation's car as follows: ▪ Statu=Available • Otherwise, do nothing and return false **Return Http Status Codes:** 200 – OK – if the car return is completed successfully 404 – Not Found – if the car is not found 500 – if an exception thrown
Cancel a reservation	**Input parameters:** • Reservation number **Return Value :** boolean **Business Rules:** • If reservation cancelled successfully return true and o Update reservation as follows:

	▪ Status=CANCELLED ○ Update reservation's car as follows: ▪ Statu=Available • If reservation is not cancelled successfully return false **Return Http Status Codes:** 200 – OK – if cancel is successfull 404 – Not Found – if reservation number is not found 500 – if an exception thrown
Delete a car	**Input parameters:** • Car barcode number **Return Value :** boolean **Business Rules:** • If car status is not Available or car is used for a reservation don't delete the car and return false. Otherwise delete the car • Don't delete car related objects **Return Http Status Codes:** 200 – OK – if delete successfull 404 – Not Found – if car is not found 406 – Not Acceptable – if car is not available or used in a reservation 500 – if an exception thrown

Main Classes:

Member	All members can search the catalog, as well as reserve, pick-up, and return a car. Members can have • Name • Address • Email • Phone • Driving license number A member can reserve more than one cars
Reservation	Responsible for managing reservations for a car. Reservation will have • Reservation number (must be autogenerated 8 digit string) • Car • Creation date • Pick-up date and time • Drop-off date and time • Pick-up location • Drop-off location • Return date • Statu

	• Member Reservation can have different status: `ACTIVE, PENDING, CONFIRMED, COMPLETED, CANCELLED, NONE` If reservation completed, same car can be used for another reservation
Location	The car rental system will have multiple locations, each location will have attributes like code, 'Name' to distinguish it from any other locations and 'Address' which defines the address of the rental location. Each location has a code and a name. Suck as: • 1-İstanbul Airport • 2-İstanbul Sabiha Gökçen Airport • 3-İstanbul Kadıköy • 4-İzmir City Center
Car	The basic building block of the system. Every car will have a barcode, license plate number, passenger capacity, brand, model, mileage, transmission type, dailyPrice etc. Cars can be of multiple types like: • Economy • People Carrier • Estate • SUV • Standard • Convertible • Luxury A car also can have different status: • `AVAILABLE` • `RESERVED` • `LOANED` • `LOST` • `BEING_SERVICED`
Equipment	Stores details about the various types of equipment that members can add to their reservation. Some of them are: • Snow Tyres • Child Seat • Baby Seat • Roof Box • WIFI • GPS Each equipment has a name and price. Price is not daily
Service	Stores details about the various types of service that members can add to their reservation, such as : • Additional Driver • Towing Service • Roadside assistance Each service has a name and price. Price is not daily