Machine Coding: Vehicle Rental System ()

Description

Implement a vehicle rental system to rent out vehicles to users and manage inventory.

Features

1. The system will support the renting of different automobiles like cars, trucks, SUVs, vans, and motorcycles.
2. Each vehicle should be added with a unique barcode and other details, including a parking stall number which helps to locate the vehicle.
3. The system should be able to retrieve information like which user took a particular vehicle or what vehicles have been rented out by a specific user.
4. Users should be able to search the vehicle inventory and reserve any available vehicle.
5. The system should collect a late-fee for vehicles returned after the due date.

Assumptions

1. Assume user accounts already exist in the system with email as a unique identifier.
2. Assume any rates to rent vehicles.

Requirements

1. **Design class Structures for all the entities required. Figure out all the entities in the application and design a data model / class structures for each one of them, storing essential information.**

2. **<u>List Available Vehicles</u>: From the inventory, return the vehicles which are free for a given time duration by user. (Average complexity)**

    ```
    // type = All/Car/SUV
    // isAvailable = true/false
    // timeFrame = Similar to hotel booking  (Define properly)
    ```

3. **<u>Book a Vehicle</u>: Given a vehicle, startDateTime and duration, book a vehicle for that particular time. The vehicle should be marked unavailable only for the given time frame and should be available for others. (Average complexity)**

4. **<u>Calculate amount to pay</u>:** Given a booking which contains vehicle, user, time frame for booking etc, calculate total amount user needs to pay. Assume any rate of the vehicle.

5. **<u>Return a vehicle</u>: After booking complete, mark the vehicle as returned/available and end the booking when the user returns the vehicle.**

---

6. <u>Calculate late fee:</u> If the vehicle is returned after the due date, charge some late fee. Assume any formula and rate to calculate the fee.

7. <u>List of rented out vehicles:</u> List all the rented out vehicles with their current tenant, due date, etc.

8. <u>Locate a vehicle:</u> Return current status of vehicle. If it is rented, return the booking and user. If free, return the parking lot number where the vehicle is parked.

---

9. **<u>List booking history of a user:</u> For a given user, return all the past and future bookings of the user with status of bookings.**

Other Notes

1. Write a driver class for demo purposes. Which will execute all the commands in one place in the code and test cases.
2. Do not use any database or NoSQL store, use in-memory data-structure for now.
3. Do not create any UI for the application.
4. Please prioritize code compilation, execution, and completion.
5. Work on the expected output first and then add good-to-have features of your own.

Expectations

1. Make sure that you have a working and demonstrable code.
2. Make sure that the code is functionally correct.
3. Use of proper abstraction, modeling, separation of concerns is required.
4. Code should be modular, readable and unit-testable.
5. Code should easily accommodate new requirements with minimal changes.

6. Proper exception handling is required.

Test Scenarios

addVehicle()
addUsers()

Vehicle RENT {vehicle_id} {user_id} duration
Vehicle Return {userId} duration
Show users
Show vehicles booked
Show vehicles available
Show vehicle {user}
Show booking history