Machine Coding: Vehicle Rental System

Description

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

Assumptions

1. Assume user accounts already exist in the system with email as a unique identifier.
2. Assume any rates to rent vehicles.
3. < Add more assumptions here if you have any >
4. Duration is in hours, vehicle to be free for that day

Requirements (Implement in order)

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

2. List Available Vehicles: From the inventory, return the vehicles which are free for a given time duration by user.
   // type = All/Car/SUV
   // timeFrame = Similar to hotel booking (Define properly)

3. Book a Vehicle: The system will support the renting of different automobiles like cars, trucks, SUVs, vans, and motorcycles. Each vehicle should be added with a unique barcode and other details, including a parking stall number which helps to locate the vehicle. 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.

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

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

6. List of rented out vehicles: List all the rented out vehicles with their current tenant, due date, etc.

7. Locate a vehicle: 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.

Other Notes

1. Use local IDE to code. You can use any language, any IDE/editor.
2. Write a driver class for demo purposes. Which will execute all the commands in one place in the code and implement test cases.
3. Recommended not to use any database or NoSQL store, use in-memory data-structure for now i.e store data in global variables.
4. Do not create any UI for the application.
5. **Prioritize code compilation, execution, and completion. We expect a working solution in the end.**
6. Work on the expected output first and then add good-to-have features of your own.
7. Hard wire 2 test cases in the driver class and run your solution on them.
8. Don't use controller or router based input or not even custom input from terminal instead give hard coded input object initialize approach in driver class for testing the functional correctness.

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. (Do not write unit tests)
5. Code should easily accommodate new requirements with minimal changes.
6. Proper exception handling is required.