CarConnect, a Car Rental Platform

Instructions:

- Submitting assignments should be a single file or through git hub link shared with trainer and hexavarsity.
- Each assignment builds upon the previous one, and by the end, you will have a comprehensive application implemented in Java/C#/Python with a strong focus on SQL schema design, control flow statements, loops, arrays, collections, and database interaction.
- Follow object-oriented principles throughout the Java programming assignments. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exception from method and handle in the main method.
- The following Directory structure is to be followed in the application.
    - **entity/model**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider interface/abstract class to showcase functionalities.
        - Create the implementation class for the above interface/abstract class with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
        - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object.
    - **main**
        - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

User Authentication:

- Secure user authentication and authorization mechanisms.

Vehicle Management:

- CRUD operations for vehicles, including details such as model, make, availability, and pricing.

Reservation System:

- Real-time reservation handling with conflict resolution.
- Email/SMS notifications for reservation confirmation and reminders.

Reporting:

- Generation of reports for administrators, including reservation history, vehicle utilization, and revenue.

Create following tables in SQL Schema with appropriate class and write the unit test case for the application.

SQL Tables:

1. Customer Table:

- **CustomerID (Primary Key):** Unique identifier for each customer.
- **FirstName:** First name of the customer.
- **LastName:** Last name of the customer.
- **Email:** Email address of the customer for communication.
- **PhoneNumber:** Contact number of the customer.
- **Address:** Customer's residential address.
- **Username:** Unique username for customer login.
- **Password:** Securely hashed password for customer authentication.
- **RegistrationDate:** Date when the customer registered.

2. Vehicle Table:

- **VehicleID (Primary Key):** Unique identifier for each vehicle.
- **Model:** Model of the vehicle.
- **Make:** Manufacturer or brand of the vehicle.
- **Year:** Manufacturing year of the vehicle.
- **Color:** Color of the vehicle.
- **RegistrationNumber:** Unique registration number for each vehicle.
- **Availability:** Boolean indicating whether the vehicle is available for rent.
- **DailyRate:** Daily rental rate for the vehicle.

3. Reservation Table:

- **ReservationID (Primary Key):** Unique identifier for each reservation.
- **CustomerID (Foreign Key):** Foreign key referencing the Customer table.
- **VehicleID (Foreign Key):** Foreign key referencing the Vehicle table.
- **StartDate:** Date and time of the reservation start.
- **EndDate:** Date and time of the reservation end.
- **TotalCost:** Total cost of the reservation.
- **Status:** Current status of the reservation (e.g., pending, confirmed, completed).

4. Admin Table:

- **AdminID (Primary Key):** Unique identifier for each admin.
- **FirstName:** First name of the admin.
- **LastName:** Last name of the admin.
- **Email:** Email address of the admin for communication.
- **PhoneNumber:** Contact number of the admin.
- **Username:** Unique username for admin login.
- **Password:** Securely hashed password for admin authentication.
- **Role:** Role of the admin within the system (e.g., super admin, fleet manager).
- **JoinDate:** Date when the admin joined the system.

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors (default and parametrized) and getters,setters)

Classes:

- **Customer:**
  - Properties: **CustomerID, FirstName, LastName, Email, PhoneNumber, Address, Username, Password, RegistrationDate**
  - Methods: **Authenticate(password)**
- **Vehicle:**
  - Properties: **VehicleID, Model, Make, Year, Color, RegistrationNumber, Availability, DailyRate**
- **Reservation:**
  - Properties: **ReservationID, CustomerID, VehicleID, StartDate, EndDate, TotalCost, Status**
  - Methods: **CalculateTotalCost()**
- **Admin:**
  - Properties: **AdminID, FirstName, LastName, Email, PhoneNumber, Username, Password, Role, JoinDate**
  - Methods: **Authenticate(password)**
- **CustomerService (implements ICustomerService):**
  - Methods: **GetCustomerById, GetCustomerByUsername, RegisterCustomer, UpdateCustomer, DeleteCustomer**
- **VehicleService (implements IVehicleService):**
  - Methods: **GetVehicleById, GetAvailableVehicles, AddVehicle, UpdateVehicle, RemoveVehicle**
- **ReservationService (implements IReservationService):**
  - Methods: **GetReservationById, GetReservationsByCustomerId, CreateReservation, UpdateReservation, CancelReservation**
- **AdminService (implements IAdminService):**
  - Methods: **GetAdminById, GetAdminByUsername, RegisterAdmin, UpdateAdmin, DeleteAdmin**
- **DatabaseContext:**
  - A class responsible for handling database connections and interactions.
- **AuthenticationService:**
  - A class responsible for handling user authentication.
- **ReportGenerator:**
  - A class for generating reports based on reservation and vehicle data.

Interfaces:

- **ICustomerService:**
  - GetCustomerById(customerId)
  - GetCustomerByUsername(username)
  - RegisterCustomer(customerData)
  - UpdateCustomer(customerData)
  - DeleteCustomer(customerId)
- **IVehicleService:**
  - GetVehicleById(vehicleId)

- GetAvailableVehicles()
- AddVehicle(vehicleData)
- UpdateVehicle(vehicleData)
- RemoveVehicle(vehicleId)

- **IReservationService:**
  - GetReservationById(reservationId)
  - GetReservationsByCustomerId(customerId)
  - CreateReservation(reservationData)
  - UpdateReservation(reservationData)
  - CancelReservation(reservationId)
- **IAdminService:**
  - GetAdminById(adminId)
  - GetAdminByUsername(username)
  - RegisterAdmin(adminData)
  - UpdateAdmin(adminData)
  - DeleteAdmin(adminId)

Connect your application to the SQL database:

- Create a connection string that includes the necessary information to connect to your SQL Server database. This includes the server name, database name, authentication credentials, and any other relevant settings.
- Use the **SqlConnection** class to establish a connection to the SQL Server database.
- Once the connection is open, you can use the **SqlCommand** class to execute SQL queries.

Custom Exceptions:

AuthenticationException:
- Thrown when there is an issue with user authentication.
- Example Usage: Incorrect username or password during customer or admin login.

ReservationException:
- Thrown when there is an issue with reservations.
- Example Usage: Attempting to make a reservation for a vehicle that is already reserved.

VehicleNotFoundException:
- Thrown when a requested vehicle is not found.
- Example Usage: Trying to get details of a vehicle that does not exist.

AdminNotFoundException:
- Thrown when an admin user is not found.
- Example Usage: Attempting to access details of an admin that does not exist.

InvalidInputException:
- Thrown when there is invalid input data.
- Example Usage: When a required field is missing or has an incorrect format.

DatabaseConnectionException:

- Thrown when there is an issue with the database connection.
- Example Usage: Unable to establish a connection to the database.

Unit Testing:

Create NUnit test cases for car rental System are essential to ensure the correctness and reliability of your system. Below are some example questions to guide the creation of NUnit test cases for various components of the system:

1. Test customer authentication with invalid credentials.
2. Test updating customer information.
3. Test adding a new vehicle.
4. Test updating vehicle details.
5. Test getting a list of available vehicles.
6. Test getting a list of all vehicles.