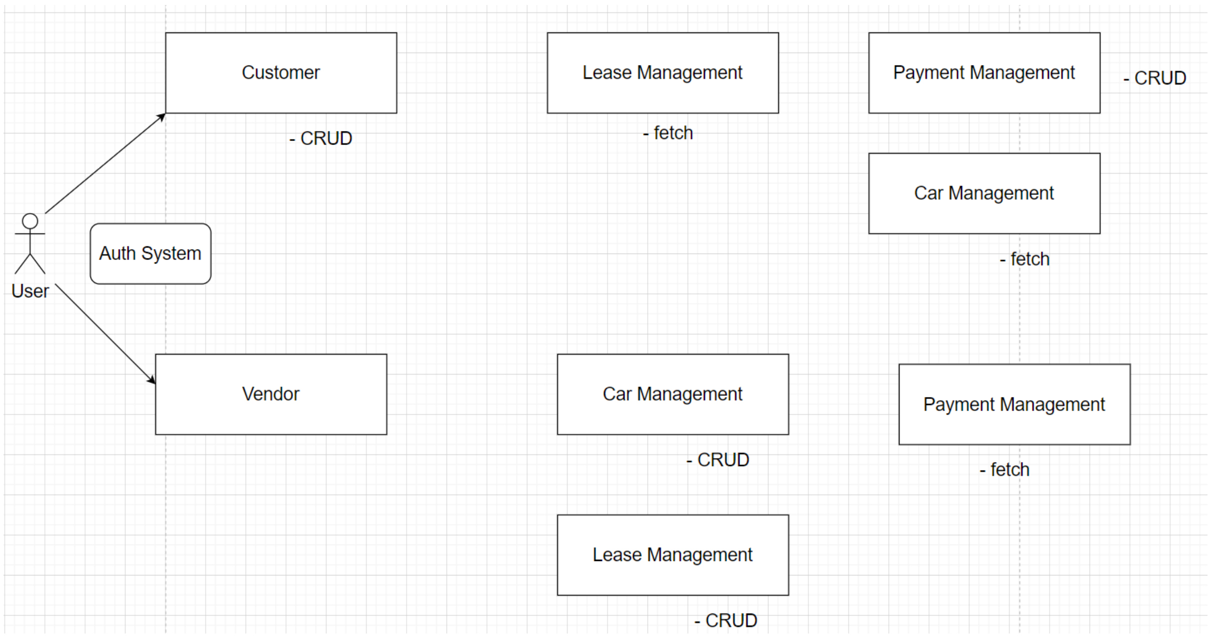
Car Rental



Instructions

- Project submissions should be done through the partcipants' Github repository, and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive **Ecommerce** implemented with a strong focus on **SQL**, **control flow statements, loops, arrays, collections, exception handling**, **database interaction** and **Unit Testing**.
- Follow **object-oriented principles** throughout the project. Use classes and objects to model real-world entities, **encapsulate data and behavior**, and **ensure code reusability**.
- Throw **user defined exceptions** from corresponding methods and handled**.**
- 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 to showcase functionalities.
        - Create the implementation class for the above interface 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(Use method defined in DBPropertyUtil class to get the connection String)**.
- **main**
    - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

1. **Customer Management**
    - Add new customers, Update customer information, Retrieve customer details.
2. **Car Management:**
    - Add new cars to the system, Update car availability, Retrieve car information.
3. **Lease Management**
    - Create daily or monthly leases for customers.
    - Calculate the total cost of a lease based on the type (Daily or Monthly) and the number of days or months.
4. **Payment Handling:**
    - Record payments for leases.
    - Retrieve payment history for a customer.
    - Calculate the total revenue from payments.

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

Schema Design:

1. **Vehicle Table:**
    - vehicleID (Primary Key)
    - make
    - model
    - year
    - dailyRate
    - status (available, notAvailable)
    - passengerCapacity
    - engineCapacity
2. **Customer Table:**
    - customerID (Primary Key)
    - firstName
    - lastName
    - email
    - phoneNumber

3. **Lease Table:**
    - leaseID (Primary Key)
    - vehicleID (Foreign Key referencing Vehicle Table)
    - customerID (Foreign Key referencing Customer Table)
    - startDate
    - endDate
    - type (to distinguish between DailyLease and MonthlyLease)
4. **Payment Table:**
    - paymentID (Primary Key)
    - leaseID (Foreign Key referencing Lease Table)
    - paymentDate
    - amount

5. **Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)**
6. **Service Provider Interface/Abstract class:**
   Keep the interfaces and implementation classes in package dao
   - Create Interface for **ICarLeaseRepository** and add following methods which interact with database.
   - **Car Management**
     1. **addCar(Car car)**
        parameter : Car
        return type : void
     2. **removeCar()**
        parameter : carID

        return type : void
     3. listAvailableCars() -
        parameter: NIL
        return type: return List of Car
     4. listRentedCars() – return List of Car
        parameter: NIL
        return type: return List of Car
     5. findCarById(int carID) – return Car if found or throw exception
        parameter: NIL
        return type: return List of Car
   - **Customer Management**
     1. addCustomer(Customer customer)
        parameter : Customer
        return type : void
     2. void removeCustomer(int customerID)
        parameter : CustomerID
        return type : void
     3. listCustomers()
        parameter : NIL
        return type : list of customer
     4. findCustomerById(int customerID)
        parameter : CustomerID
        return type : Customer

- **Lease Management**
    1. createLease()
        parameter : int customerID, int carID, Date startDate, Date endDate
        return type : Lease
    2. void returnCar();
        parameter : int leaseID
        return type : Lease info
    3. List<Lease> listActiveLeases();
        parameter : NIL
        return type : Lease list
    4. listLeaseHistory();
        parameter : NIL
        return type : Lease list
- **Payment Handling**
    1. void recordPayment();
        parameter : Lease lease, double amount
        return type : void

9. Create the exceptions in package **myexceptions** and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,
    - **CarNotFoundException**: throw this exception when user enters an invalid car id which doesn't exist in db.
    - **LeaseNotFoundException**: throw this exception when user enters an invalid lease id which doesn't exist in db.
    - **CustomerrNotFoundException**: throw this exception when user enters an invalid customer id which doesn't exist in db.

Unit Testing:

10. Create Unit test cases for **Ecommerce System** are essential to ensure the correctness and reliability of your system. Following questions to guide the creation of Unit test cases:
    - Write test case to test car created successfully or not.
    - Write test case to test lease is created successfully or not.
    - Write test case to test lease is retrieved successfully or not.
    - write test case to test exception is thrown correctly or not when customer id or car id or lease id not found in database.