Car Rental System

The Fast Lane to Convenient Car Rentals!

OUR TEAM

DHARSHANI MURUGAIYAN

SOBHIKA KAMALAVASAN

VINODHA R

ABSTRACT

The Car Rental System is a simple Python application that helps manage a car rental business. It lets users add cars and customers, create leases, and record payments — all through a clear, menu-based interface.

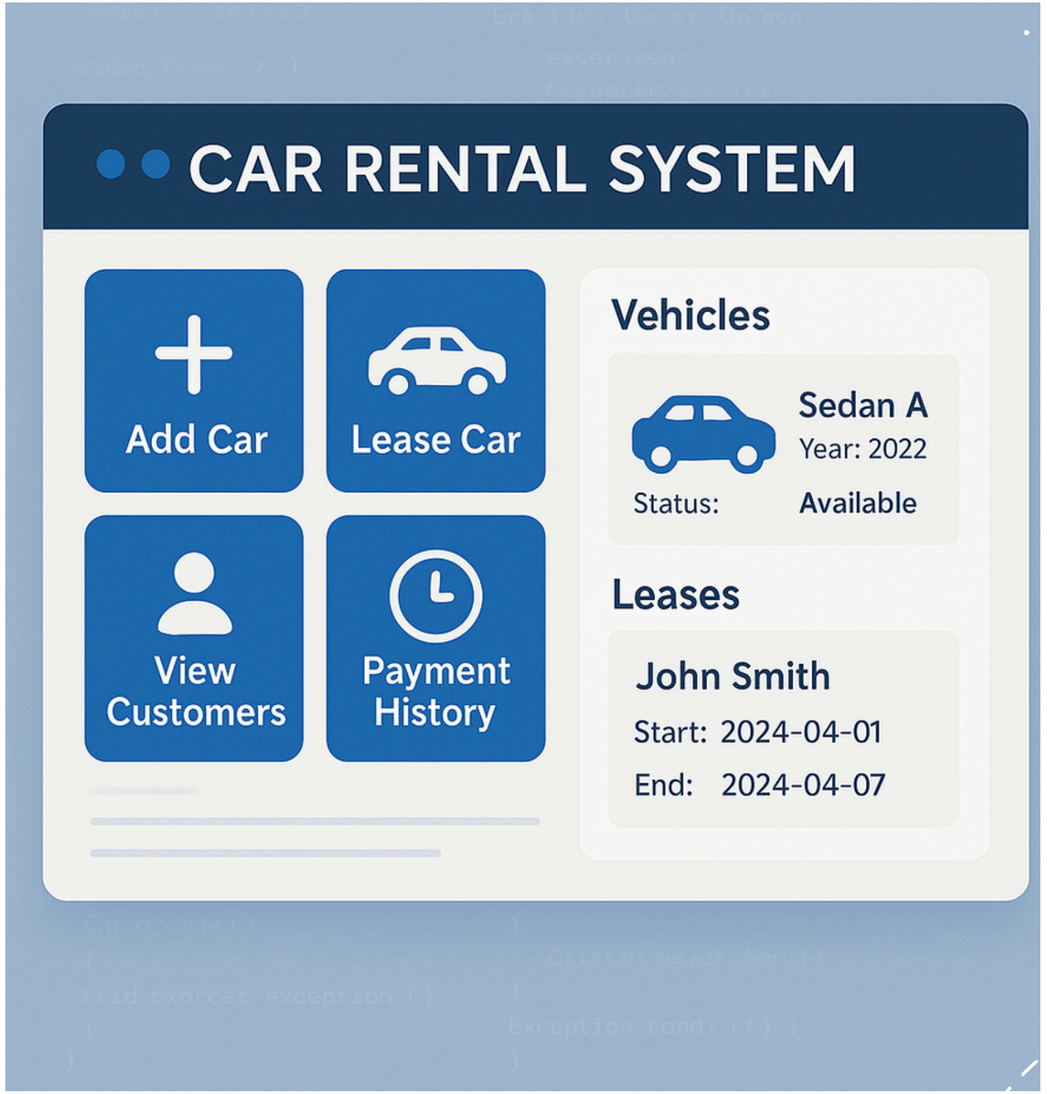
Built using object-oriented programming and connected to a MySQL database, the system keeps data organized and easy to access. It’s modular, easy to use, and ready for future upgrades like a web or GUI version.

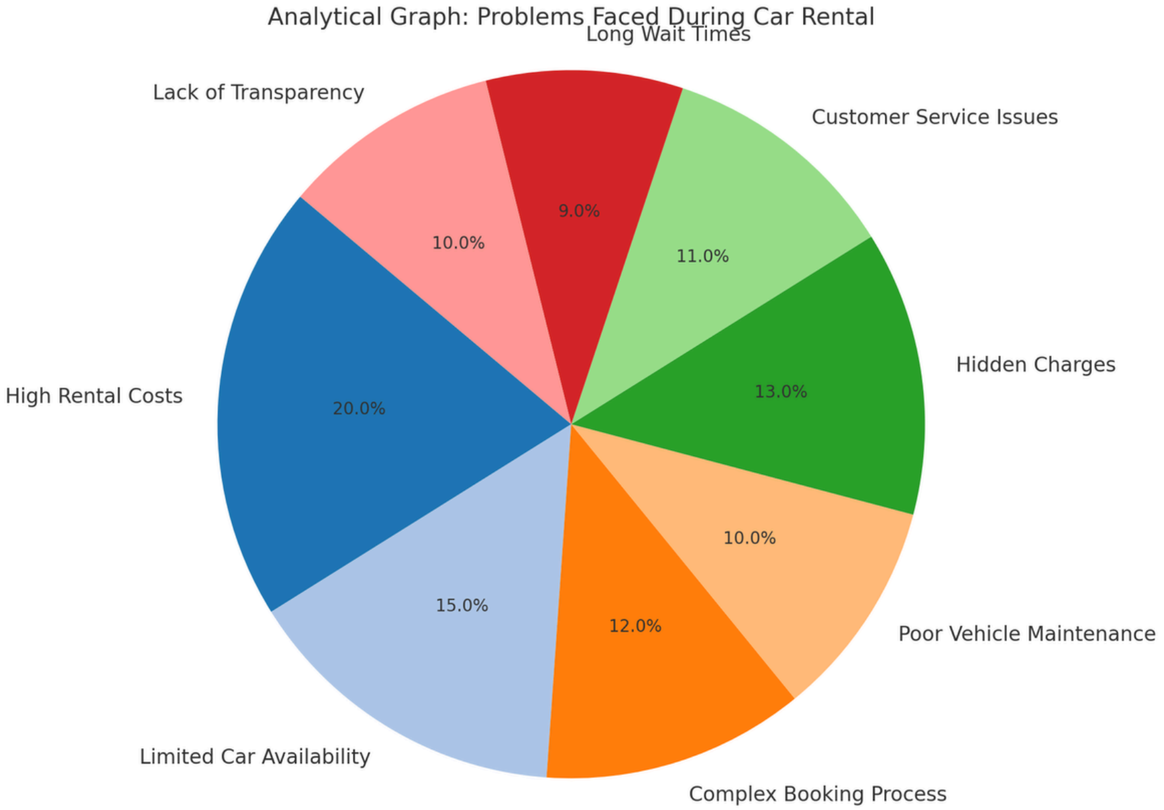
INTRODUCTION

Managing a car rental business involves keeping track of vehicles, customer bookings, return dates, and payments With a clean, menu-driven interface, users can:

- Add and manage car details
- Register new customers
- Create and track lease agreements
- View all available cars and customer lists
- Search for cars or customers by their ID
- Exit the system gracefully

This project addresses that challenge by building a structured Car Rental System using Python and MySQL.



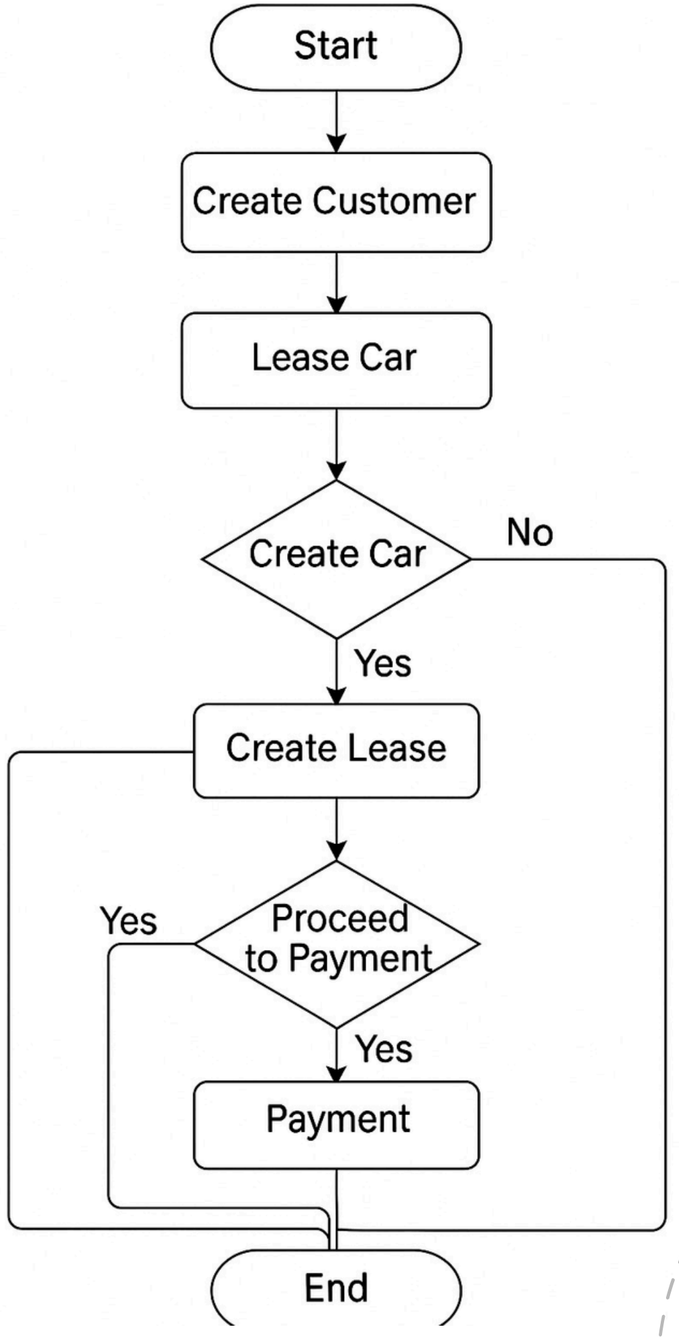


ANALYSIS

The graph highlights key challenges in the car rental experience, with high rental costs and limited car availability being the most frequent issues. Other common problems include hidden charges, complex booking processes, and poor customer service, indicating the need for a more transparent and user-friendly system.

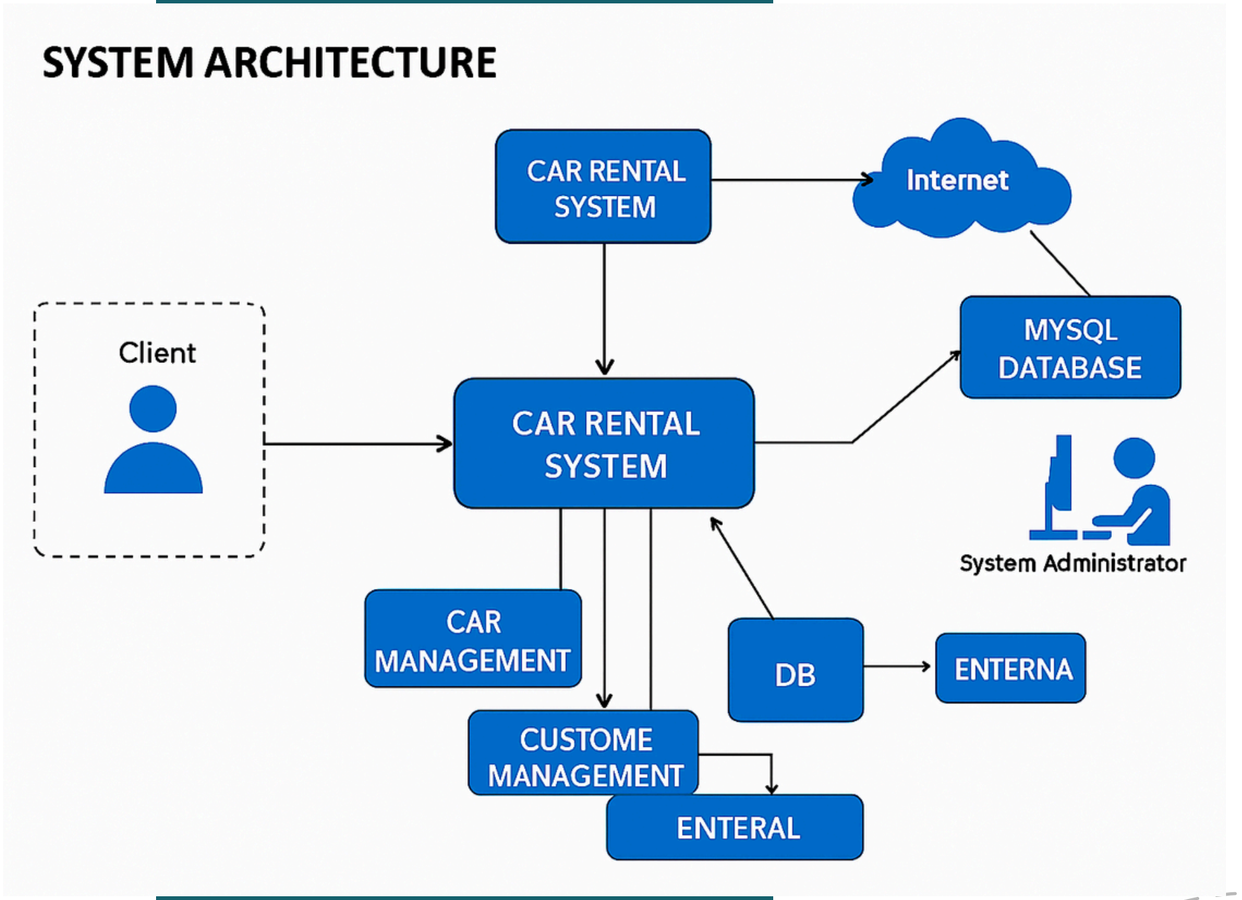
FLOWCHART

The flowchart illustrates the logical flow of the Car Rental System, starting from user input and authentication, followed by customer or admin actions like adding cars, managing leases, and recording payments. Each process leads to database interactions or exception handling, ensuring a smooth transition between modules and robust control over system operations.



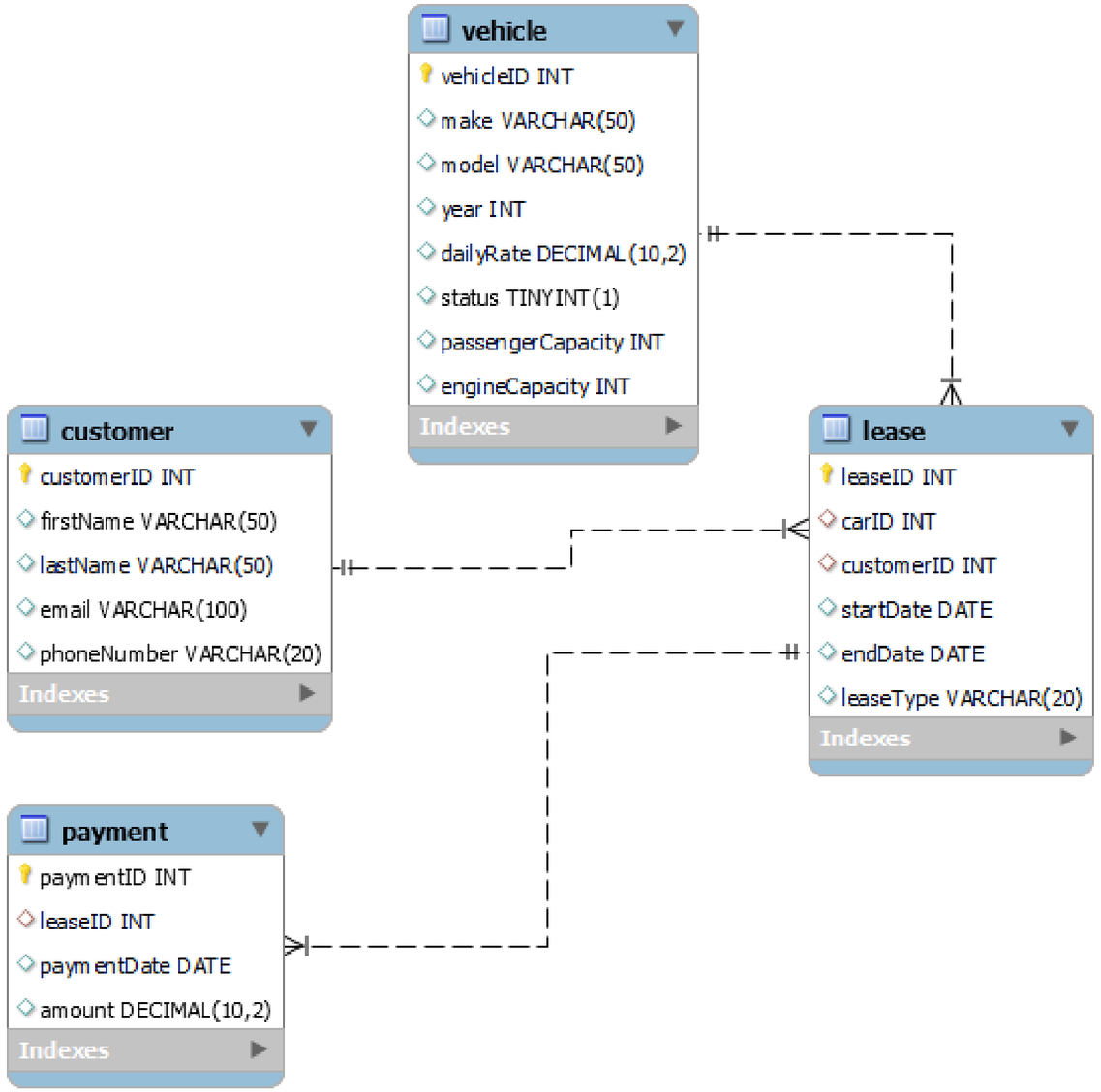
SYSTEM ARCHITECTURE DIAGRAM

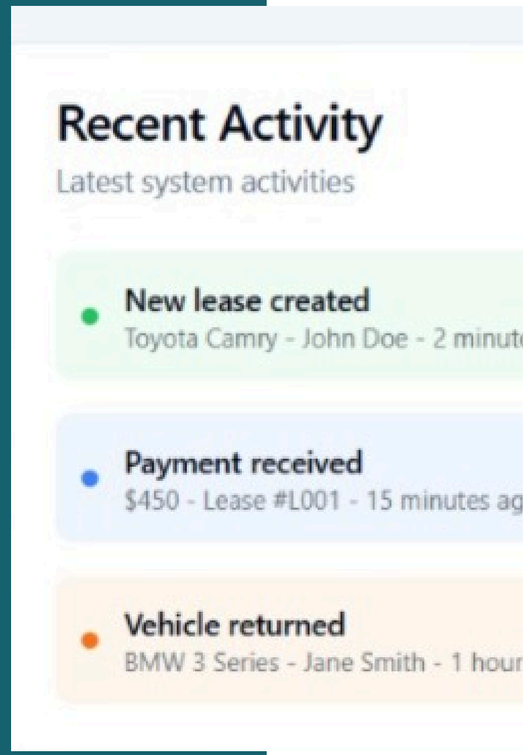
The system architecture diagram of the Car Rental System showcases a layered structure involving the User Interface, and MySQL Database. It clearly outlines the interaction between the entity models, DAO layer, custom exception handling, and REST endpoints, ensuring seamless data flow and modular functionality across the system.

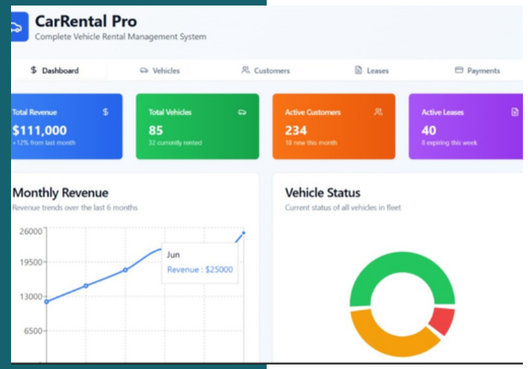


ER DIAGRAM

The ER (Entity-Relationship) Diagram of the developed Car Rental System illustrates the core entities: Car, Customer, Lease, and Payment. Each entity is interconnected—Customer books a Car via a Lease, and each lease is linked to a Payment, effectively capturing the rental lifecycle and relational flow of data within the system.







FEATURES OF THE SYSTEM

- Vehicle Management: Add, update, view, and delete vehicle details including model, type, and availability.
- Customer & Lease Handling: Manage customer data, create leases, and associate customers with specific cars for defined durations.
- Exception Handling: Custom exceptions (CarNotFoundException, LeaseNotFoundException, CustomerNotFoundException) ensure robust error management.

CODE AND OUTPUT

```
STATUS_AVAILABLE = 'available'
STATUS_NOT_AVAILABLE = 'notAvailable'
repo = CarLeaseRepositoryImpl()
def get_date_input(prompt):
    while True:
        try:
            return datetime.strptime(input(prompt + " (YYYY-MM-DD): "), "%Y-%m-%d").date()
        except ValueError:
            print("Invalid date format. Please enter in YYYY-MM-DD.")
while True:
    print("\n--- CAR RENTAL SYSTEM ---")
    print("1. Add Car")
    print("2. Add Customer")
    print("3. Create Lease")
    print("4. List Available Cars")
    print("5. Find Car by ID")
    print("6. List All Customers")
    print("7. Find Customer by ID")
    print("8. Exit")
    try:
        choice = int(input("Enter choice: "))
    except ValueError:
        print("Invalid input. Enter a number.")
        continue
    if choice == 1:
        print("\n--- Add Car ---")
        make = input("Enter make: ")
        model = input("Enter model: ")
        year = int(input("Enter year: "))
        rate = float(input("Enter daily rate: "))
        status = input("Enter status (available / notAvailable): ").strip()
        passenger_capacity = int(input("Enter passenger capacity: "))
        engine_capacity = float(input("Enter engine capacity (in cc): "))
        car = Car(None, make, model, year, rate, status, passenger_capacity, engine_capacity)
        repo.addCar(car)
        print("Car added.")
    elif choice == 2:
        print("\n--- Add Customer ---")
        fname = input("Enter first name: ")
        lname = input("Enter last name: ")
        email = input("Enter email: ")
        phone = input("Enter phone number: ")
        customer = Customer(None, fname, lname, email, phone)
        repo.addCustomer(customer)
        print("Customer added.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\murug\Downloads\car_rental_system - Final\car_rental_system> python -m main.Ma

--- CAR RENTAL SYSTEM ---
1. Add Car
2. Add Customer
3. Create Lease
4. List Available Cars
5. Find Car by ID
6. List All Customers
7. Find Customer by ID
8. Exit
Enter choice: 5

--- Find Car by ID ---
Enter Car ID: 2
Car Found: (2, 'Honda', 'Civic', 2023, Decimal('45.00'), 1, 7, 1500)

--- CAR RENTAL SYSTEM ---
1. Add Car
2. Add Customer
3. Create Lease
4. List Available Cars
5. Find Car by ID
6. List All Customers
7. Find Customer by ID
8. Exit
Enter choice: 6

--- All Customers ---
(1, 'John', 'Doe', 'johndoe@example.com', '555-555-5555')
(2, 'Jane', 'Smith', 'janesmith@example.com', '555-123-4567')
(3, 'Robert', 'Johnson', 'robert@example.com', '555-789-1234')
(4, 'Sarah', 'Brown', 'sarah@example.com', '555-456-7890')
(5, 'David', 'Lee', 'david@example.com', '555-987-6543')
```

THANK YOU

Github link

https://github.com/SobhikaKamalavasan/Hexaware-P2-Training/tree/main/casestudy