

# Car Rental System

---

The Fast Lane to Convenient Car Rentals!

# OUR TEAM

**DHARSHANI  
MURUGAIYAN**



**SOBHIKA  
KAMALAVASA  
N**



**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.



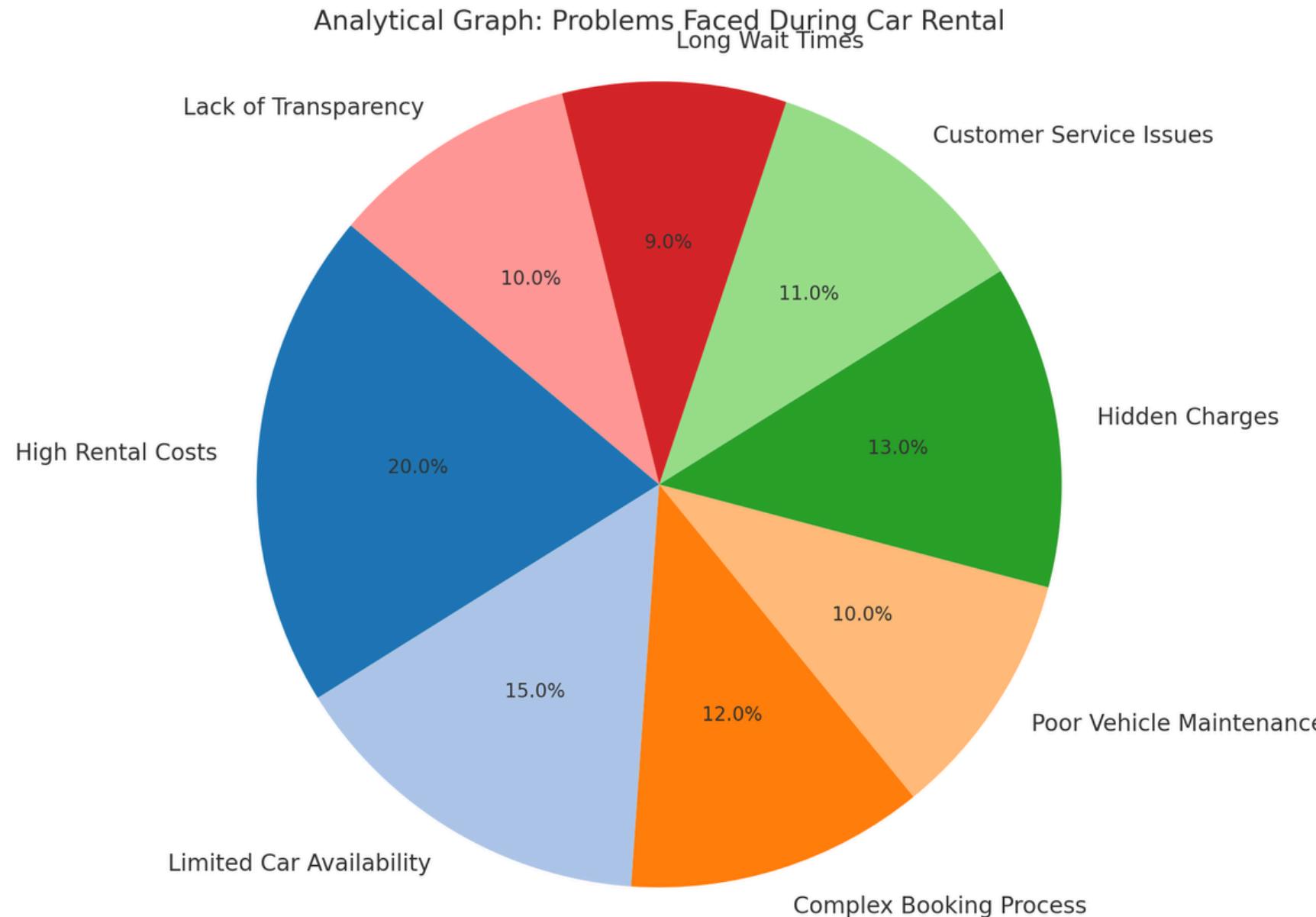
# INTRODUCTION

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.



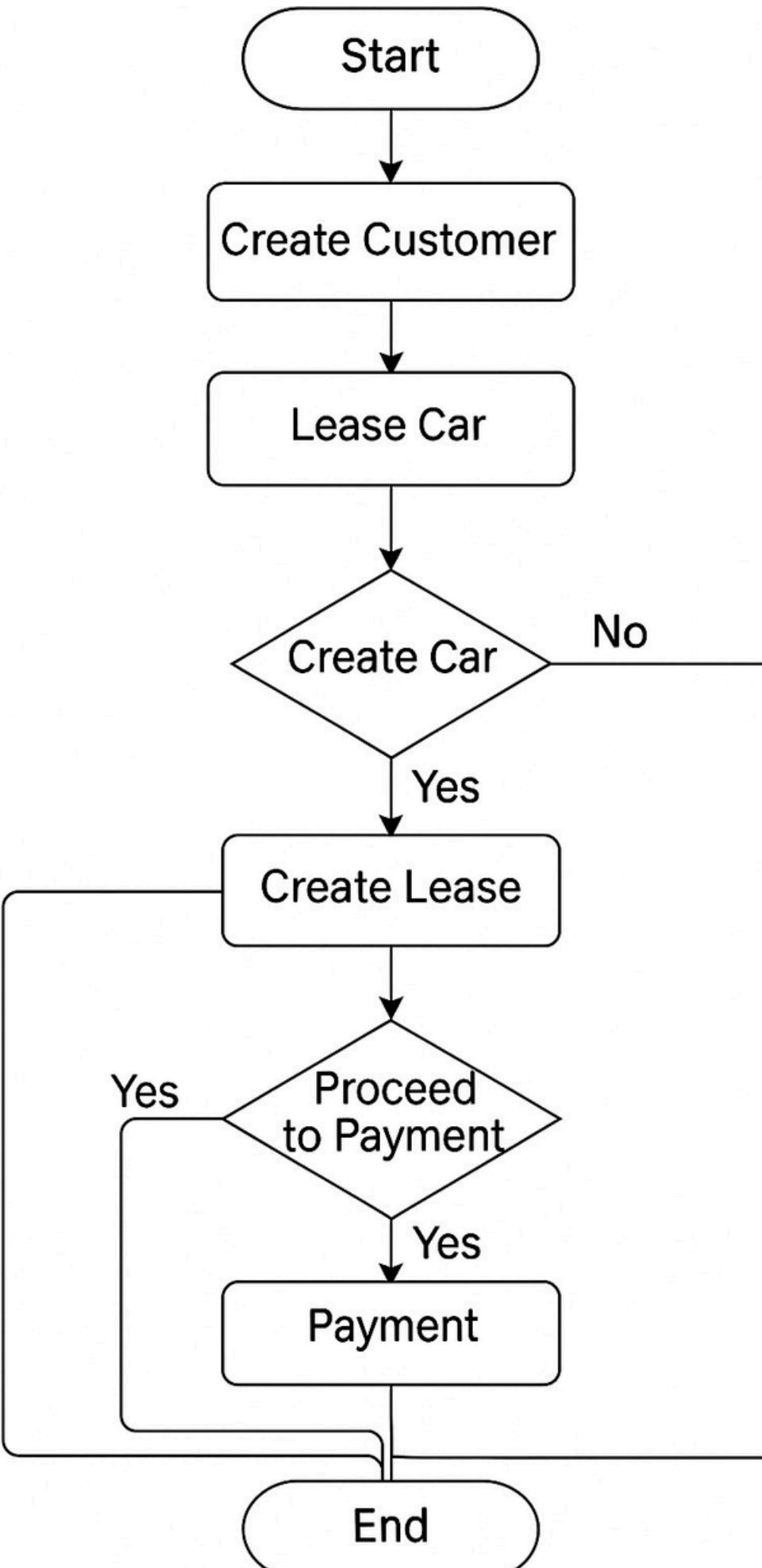


# ANALYSIS

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

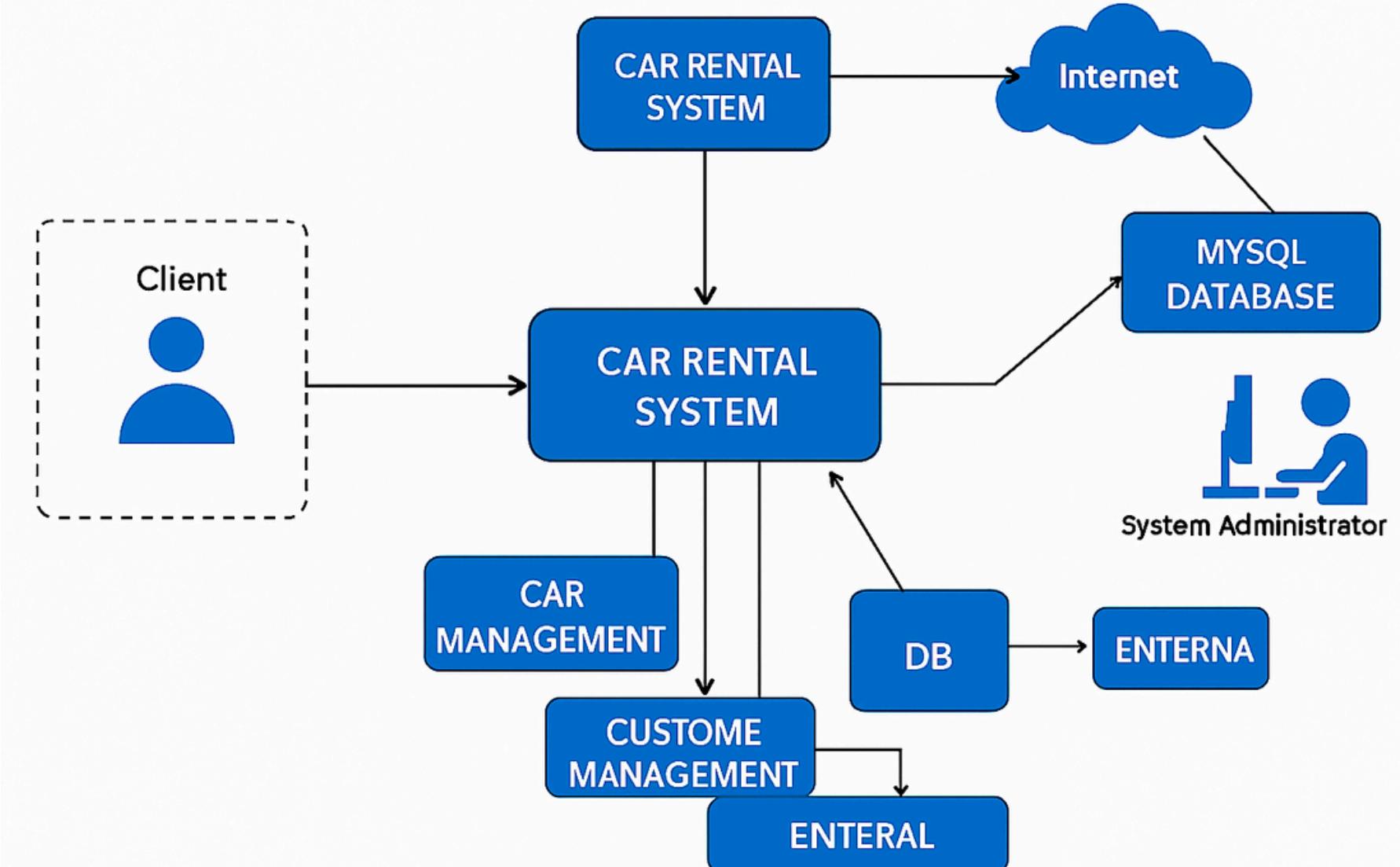
The flowchart illustrates the logical flow of the Car Rental System, starting from user input and authentication, followed by customer or admin actions like adding cars, managing leases, and recording payments. Each process leads to database interactions or exception handling, ensuring a smooth transition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



# SYSTEM ARCHITECTURE DIAGRAM

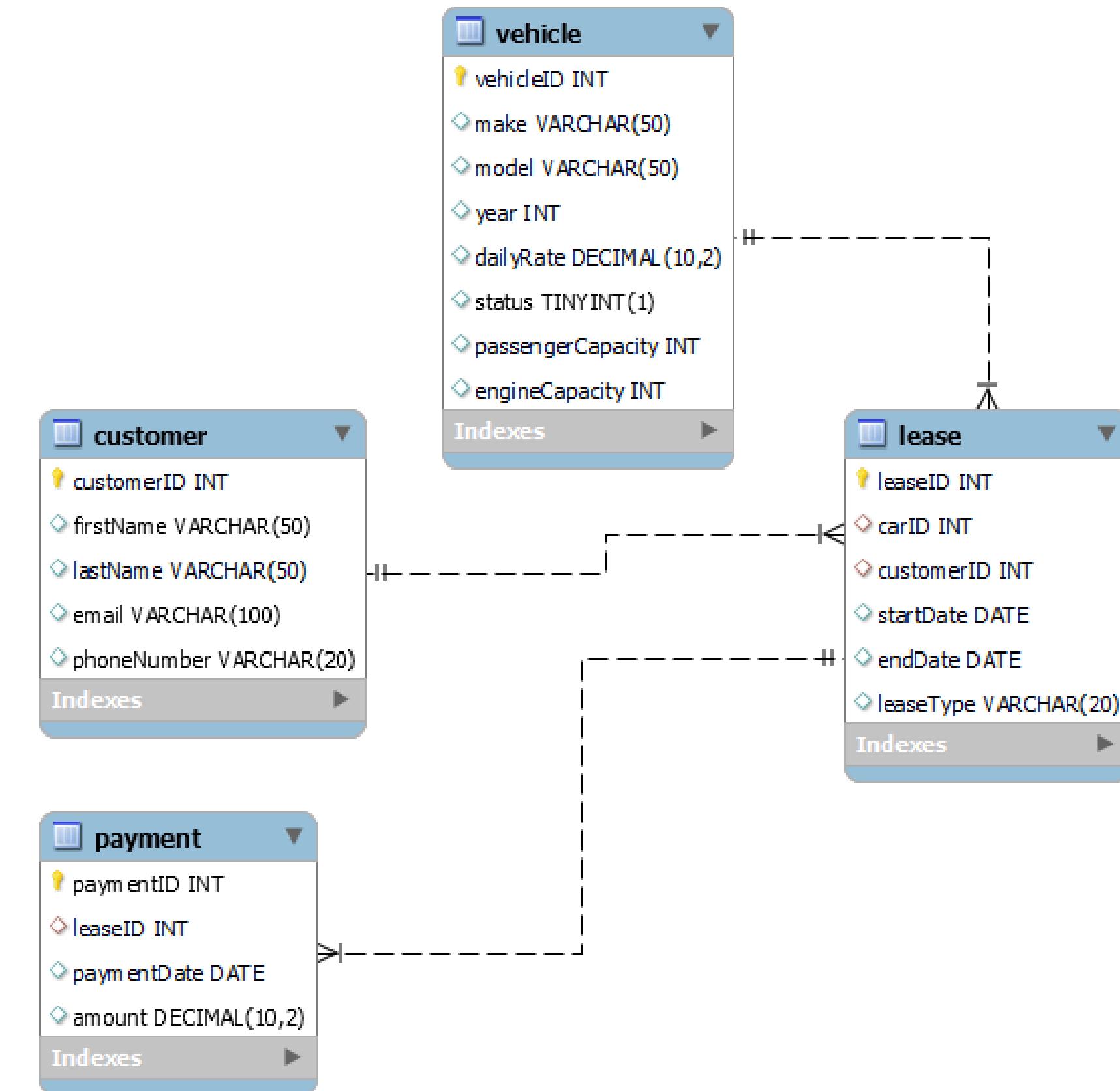
The system architecture diagram of the Car Rental System showcases a layered structure involving the User Interface, and MySQL Database. It clearly outlines the interaction between the entity models, DAO layer, custom exception handling, and REST endpoints, ensuring seamless data flow and modular functionality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

## SYSTEM ARCHITECTURE



# 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.





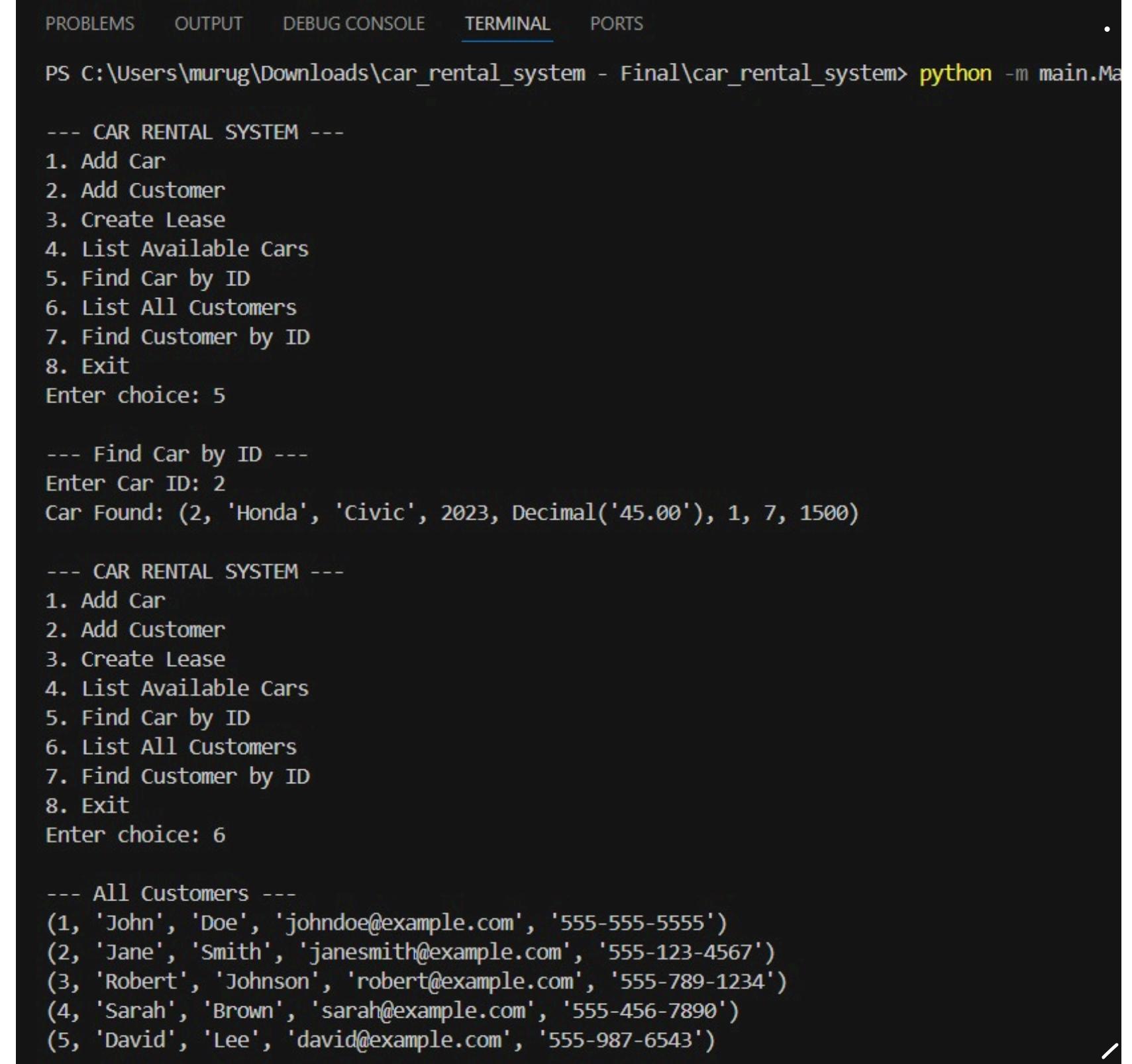
A screenshot of the CarRental Pro dashboard. At the top, it displays "Recent Activity" with three items: "New lease created" (Toyota Camry - John Doe - 2 minutes ago), "Payment received" (\$450 - Lease #L001 - 15 minutes ago), and "Vehicle returned" (BMW 3 Series - Jane Smith - 1 hour ago). Below this is a summary section with metrics: Total Revenue (\$111,000), Total Vehicles (85), Active Customers (234), and Active Leases (40). The bottom left shows a line chart for "Monthly Revenue" with a data point for June at \$25000. The bottom right shows a donut chart for "Vehicle Status".

# 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

```
def main():
    repo = CarLeaseRepositoryImpl()
    try:
        car_id = int(input("Enter Car ID to search: "))
        car = repo.findCarById(car_id)
        print(f"Car Found: {car.get_make()} {car.get_model()}")
    except CarNotFoundException as e:
        print(f"[Error] {e}")
    try:
        customer_id = int(input("Enter Customer ID to search: "))
        customer = repo.findCustomerById(customer_id)
        print(f"Customer Found: {customer.get(firstName)}
{customer.getLastName()}")
    except CustomerNotFoundException as e:
        print(f"[Error] {e}")
    try:
        lease_id = int(input("Enter Lease ID to return car: "))
        lease = repo.returnCar(lease_id)
        print(f"Returned Vehicle for Lease ID: {lease_id}")
    except LeaseNotFoundException as e:
        print(f"[Error] {e}")
    if __name__ == '__main__':
        main()
```



The screenshot shows a terminal window with the following content:

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

--- 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', 'janeshmith@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/Hetaware-P2-Training/tree/main/casestudy>