OOP (JAVA) Final Project

Date : 21_May_2025

Vehicle Management System Project Report

Object Oriented Programing

BS - II SOFTWARE ENGINEERING

SECTION "A"

Instructor: Dr. Faheem Akhter Rajput

Student:

Talha Irfan

CMS:

053-24-0013

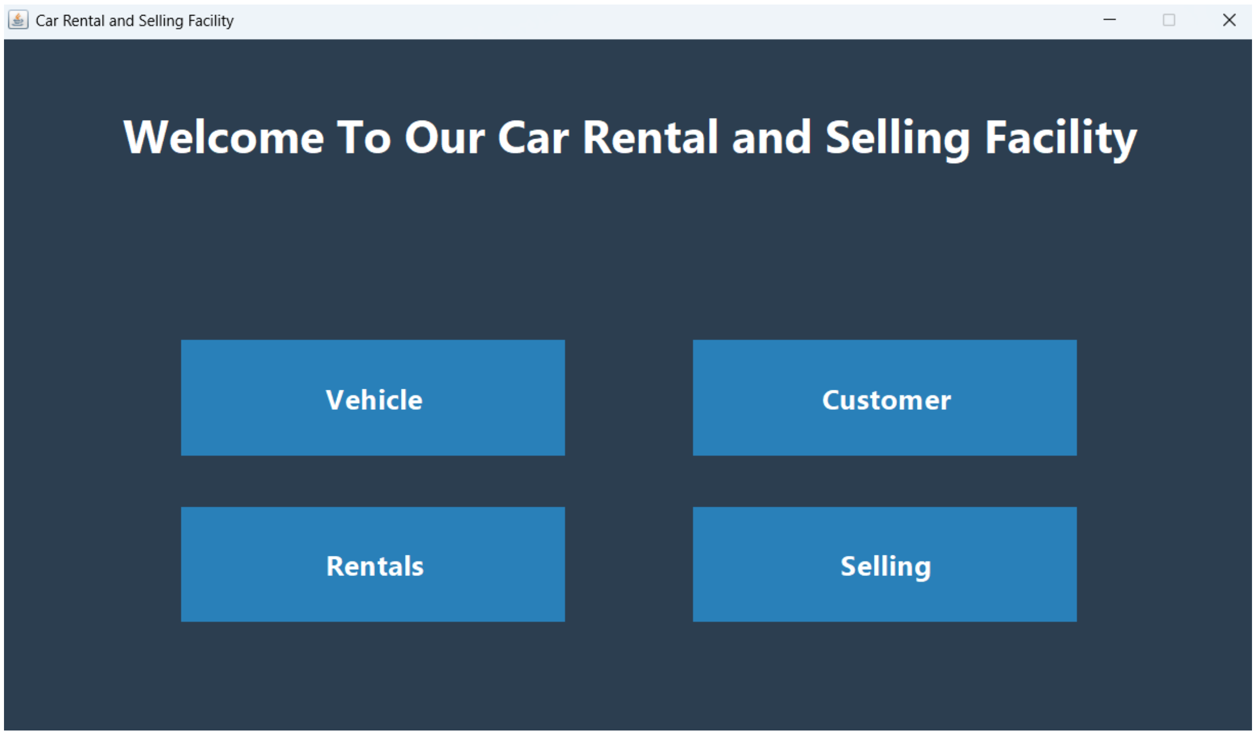
Table of Contents

Final Project Report of OOP (Java)

1. Introduction

This project is a **Vehicle Management System** designed using Java with a **Graphical User Interface (GUI)** and connected to a **MySQL database**. The system supports essential operations such as **Search, Insert, Update, Delete**, and **navigation of vehicle records, customers, and rental/selling transactions**.

The system is structured using Object-Oriented Programming (OOP) principles, including encapsulation, inheritance, and polymorphism, ensuring clean, modular, and reusable code.


Car Rental and Selling Facility
Welcome To Our Car Rental and Selling Facility
Vehicle
Customer
Rentals
Selling

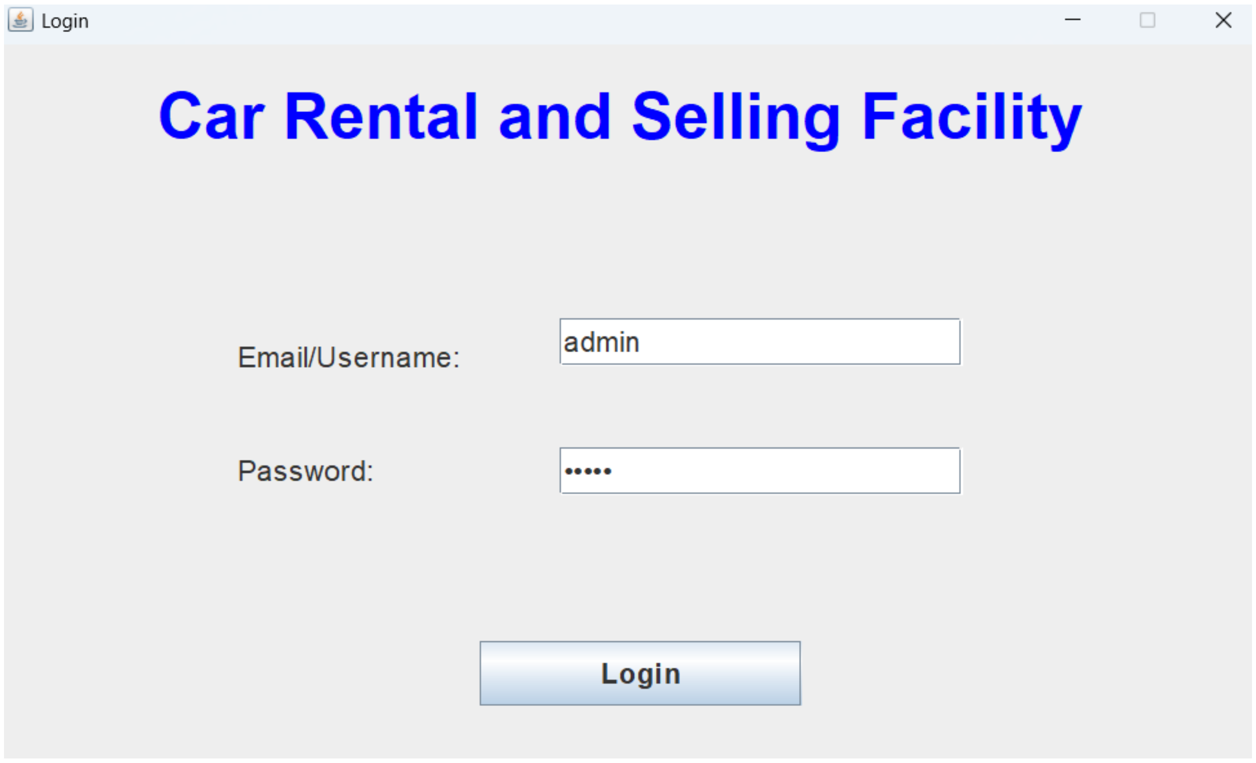
2. Objective

The objective of this project is to create a user-friendly management system for vehicle rental and sales, enabling administrators to efficiently manage vehicles, customers, rentals, and returns using a Java Swing GUI connected to a MySQL database.

3. Snapshots of the System

- Login page.
- Admin Menu.
- Vehicle Menu.
- Customer Menu.
- Rental Menu
- Sales Menu

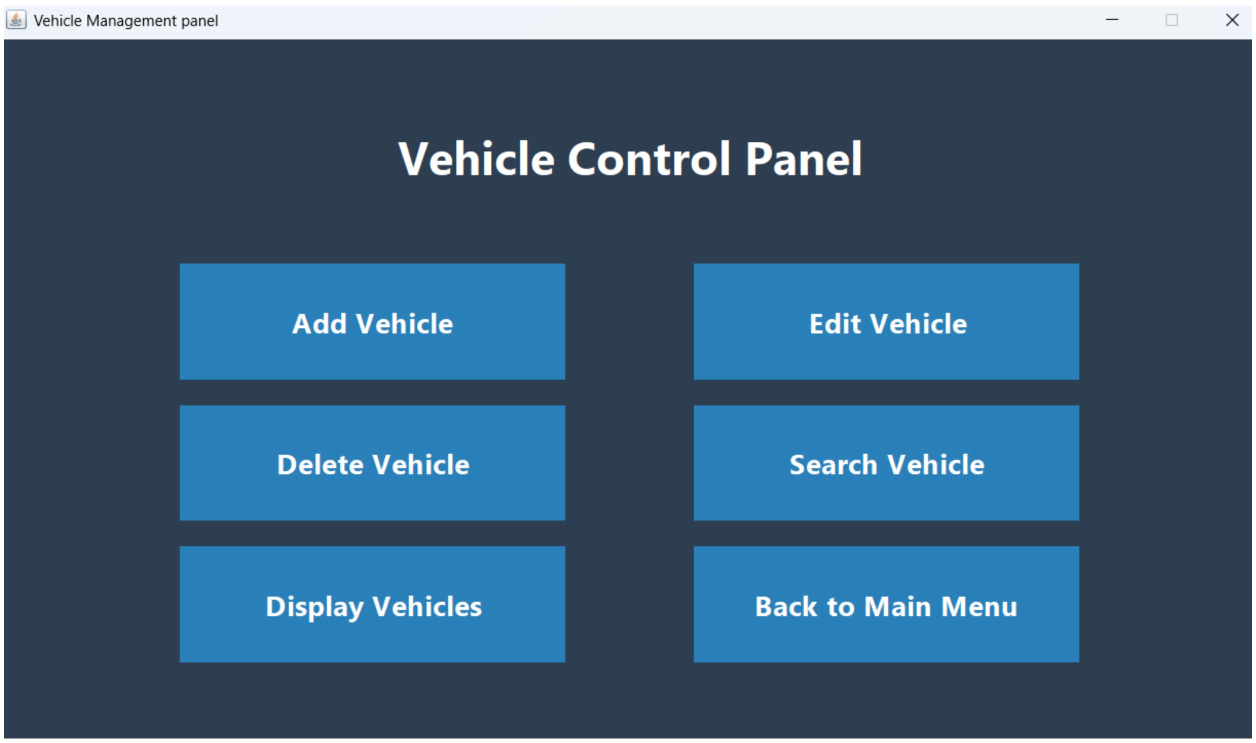
Login page : (username “admin” and Password “admin”)



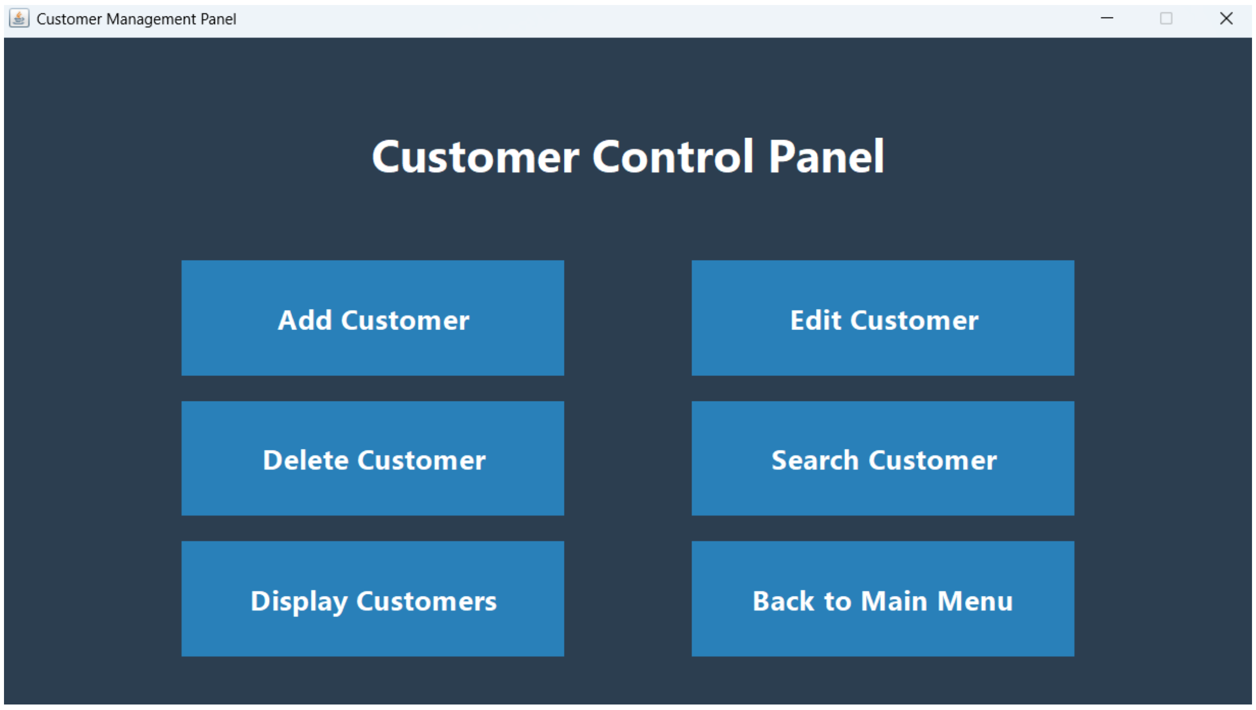
Email/Username: admin

Password: •••••

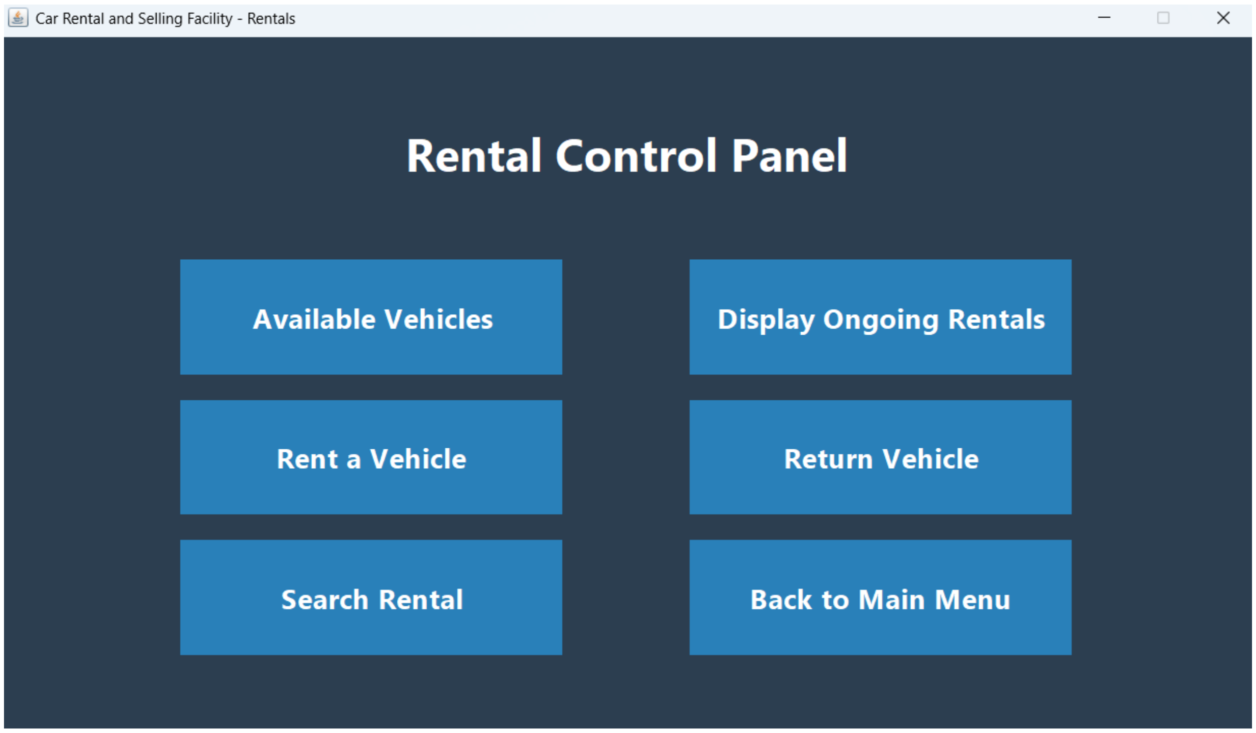
Vehicle Menu :



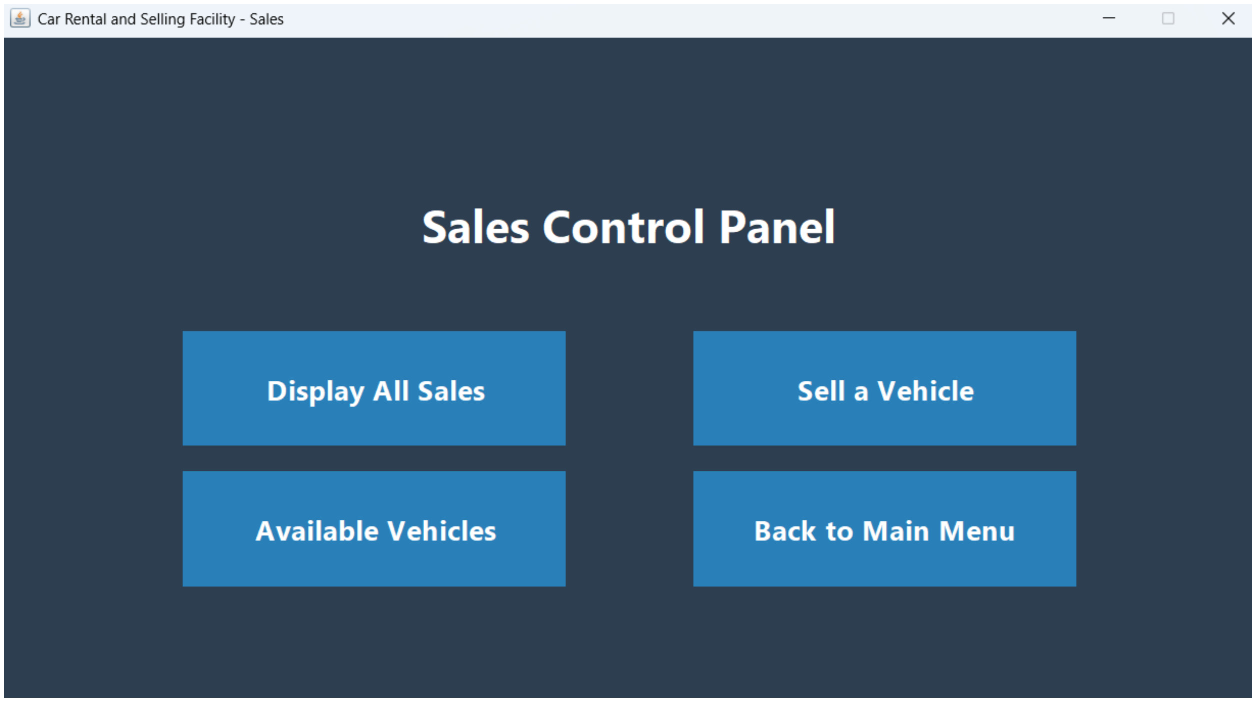
Customer Menu :



Rental Menu :



Sales Menu :



4. Project Features

- Add, Search, Update, Delete and Display vehicles

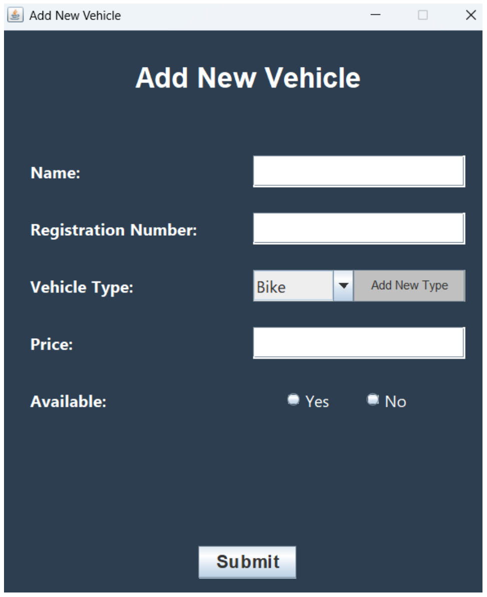
- Add, Search, Update, Delete and Display Customers

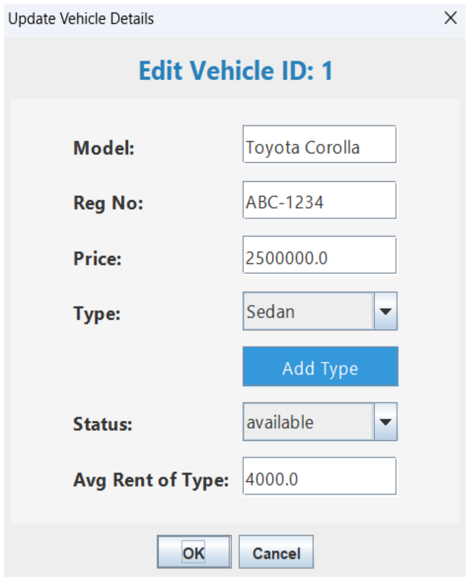
- Rent and Return Vehicles with database tracking

- Display records (rented, sold, ongoing rentals) with Customer and Vehicle details in formatted GUI tables

- Login system with Admin (username “admin” and Password “admin”)

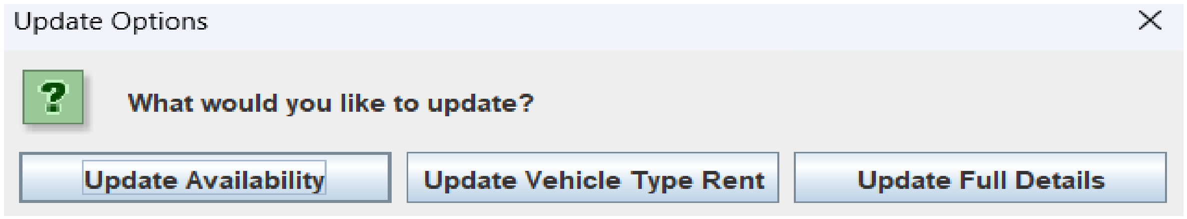
- Database integration with MySQL



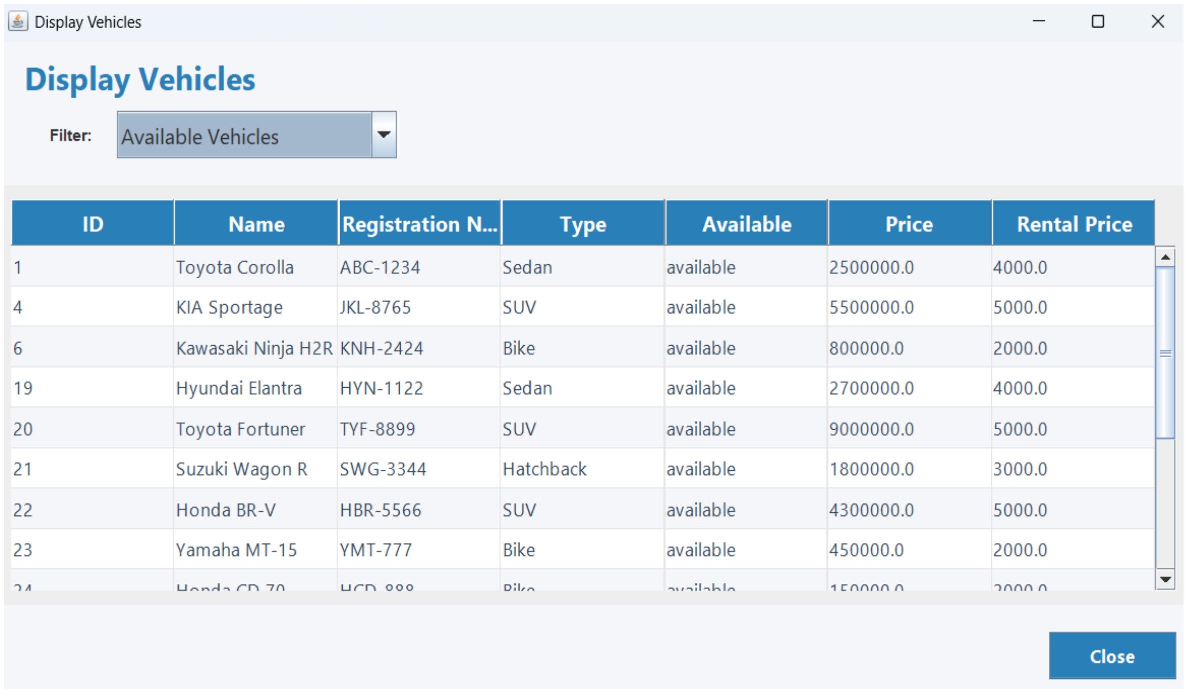

Update Vehicle Details

Edit Vehicle ID: 1

Model:	Toyota Corolla
Reg No:	ABC-1234
Price:	2500000.0
Type:	Sedan
Status:	available
Avg Rent of Type:	4000.0


Update Options

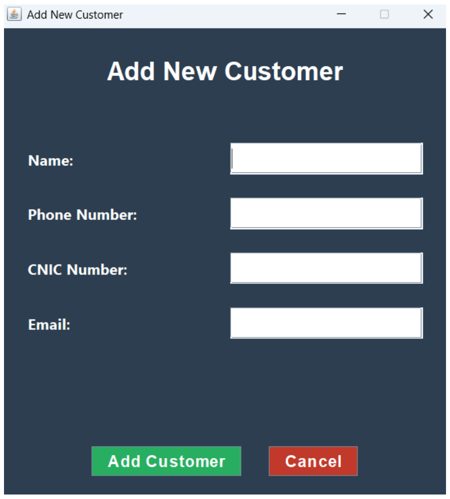
What would you like to update?


Display Vehicles

Display Vehicles

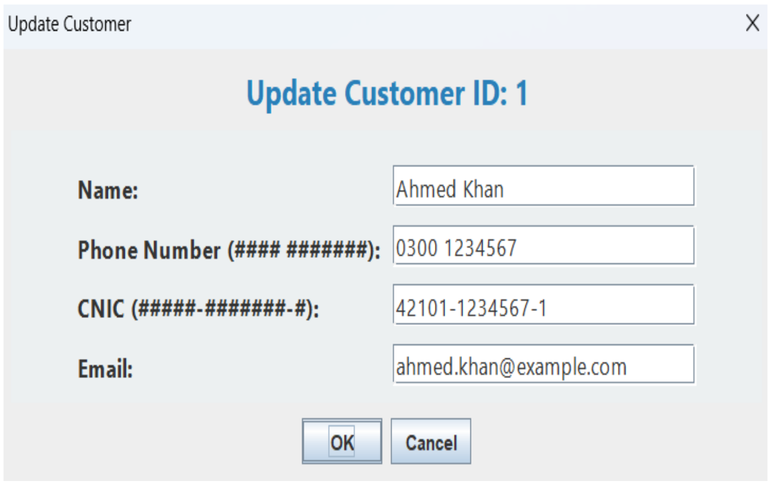
Filter: Available Vehicles

ID	Name	Registration N...	Type	Available	Price	Rental Price
1	Toyota Corolla	ABC-1234	Sedan	available	2500000.0	4000.0
4	KIA Sportage	JKL-8765	SUV	available	5500000.0	5000.0
6	Kawasaki Ninja H2R	KNH-2424	Bike	available	800000.0	2000.0
19	Hyundai Elantra	HYN-1122	Sedan	available	2700000.0	4000.0
20	Toyota Fortuner	TYF-8899	SUV	available	9000000.0	5000.0
21	Suzuki Wagon R	SWG-3344	Hatchback	available	1800000.0	3000.0
22	Honda BR-V	HBR-5566	SUV	available	4300000.0	5000.0
23	Yamaha MT-15	YMT-777	Bike	available	450000.0	2000.0
24	Honda CD 70	HCD-888	Bike	available	150000.0	2000.0


Add New Customer

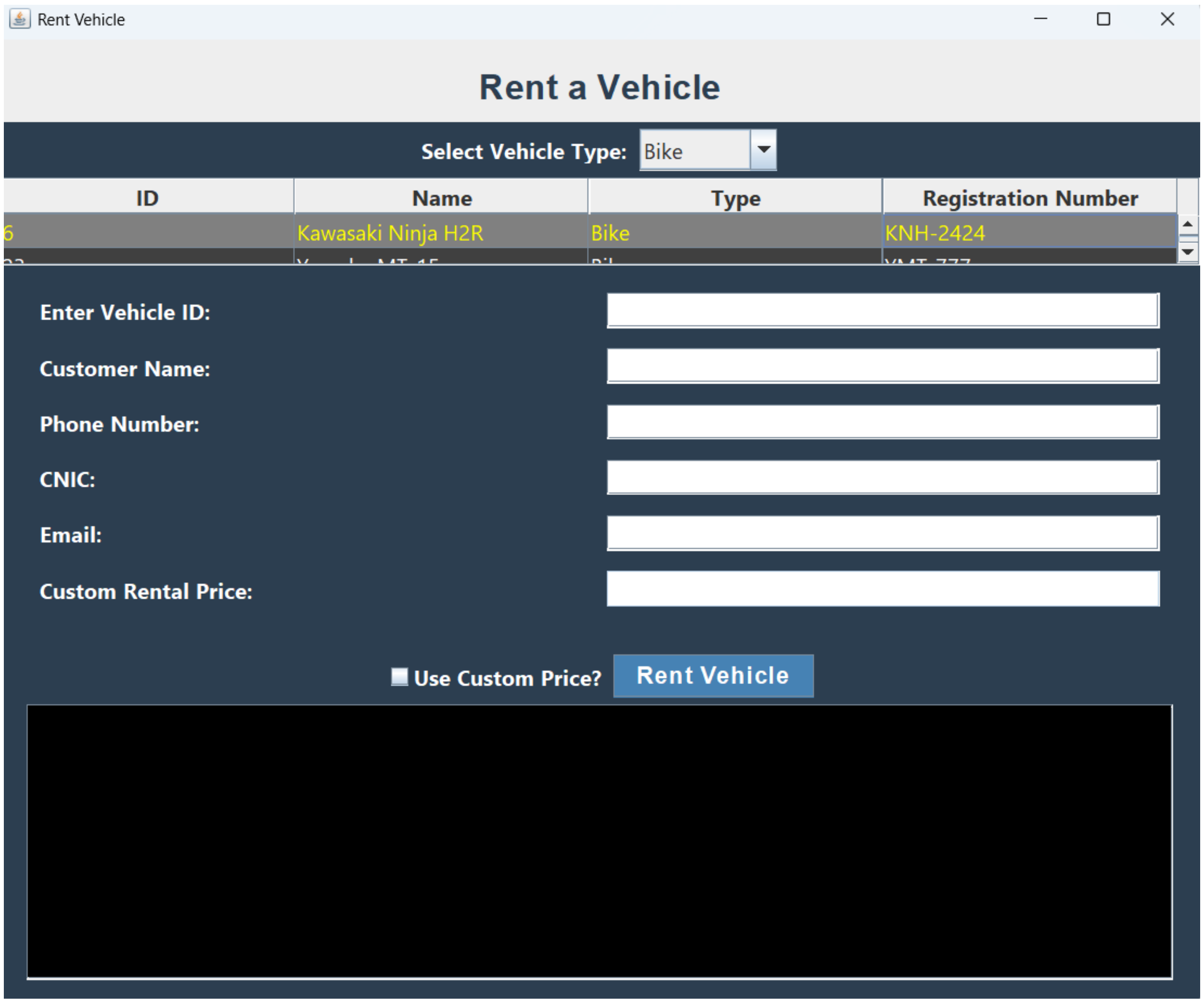
Add New Customer

Name:	
Phone Number:	
CNIC Number:	
Email:	


Update Customer

Update Customer ID: 1

Name:	Ahmed Khan
Phone Number (#### #######):	0300 1234567
CNIC (#####-#######-#):	42101-1234567-1
Email:	ahmed.khan@example.com



Rent Vehicle

Rent a Vehicle

Select Vehicle Type: Bike

ID	Name	Type	Registration Number
6	Kawasaki Ninja H2R	Bike	KNH-2424

Enter Vehicle ID:

Customer Name:

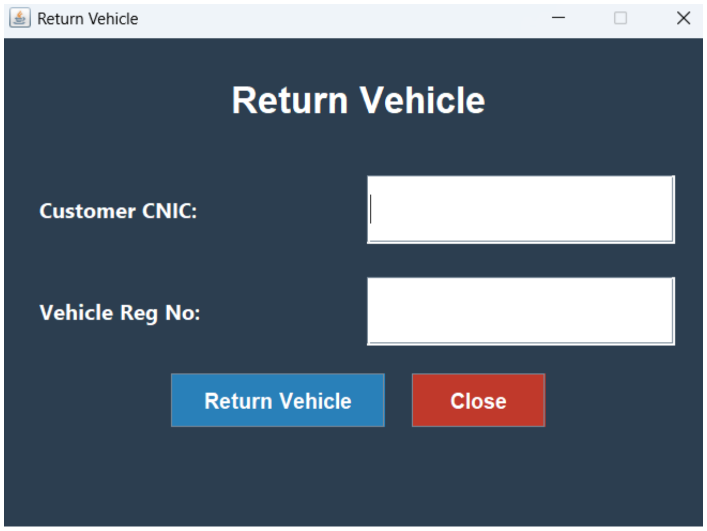
Phone Number:

CNIC:

Email:

Custom Rental Price:

☐ Use Custom Price?

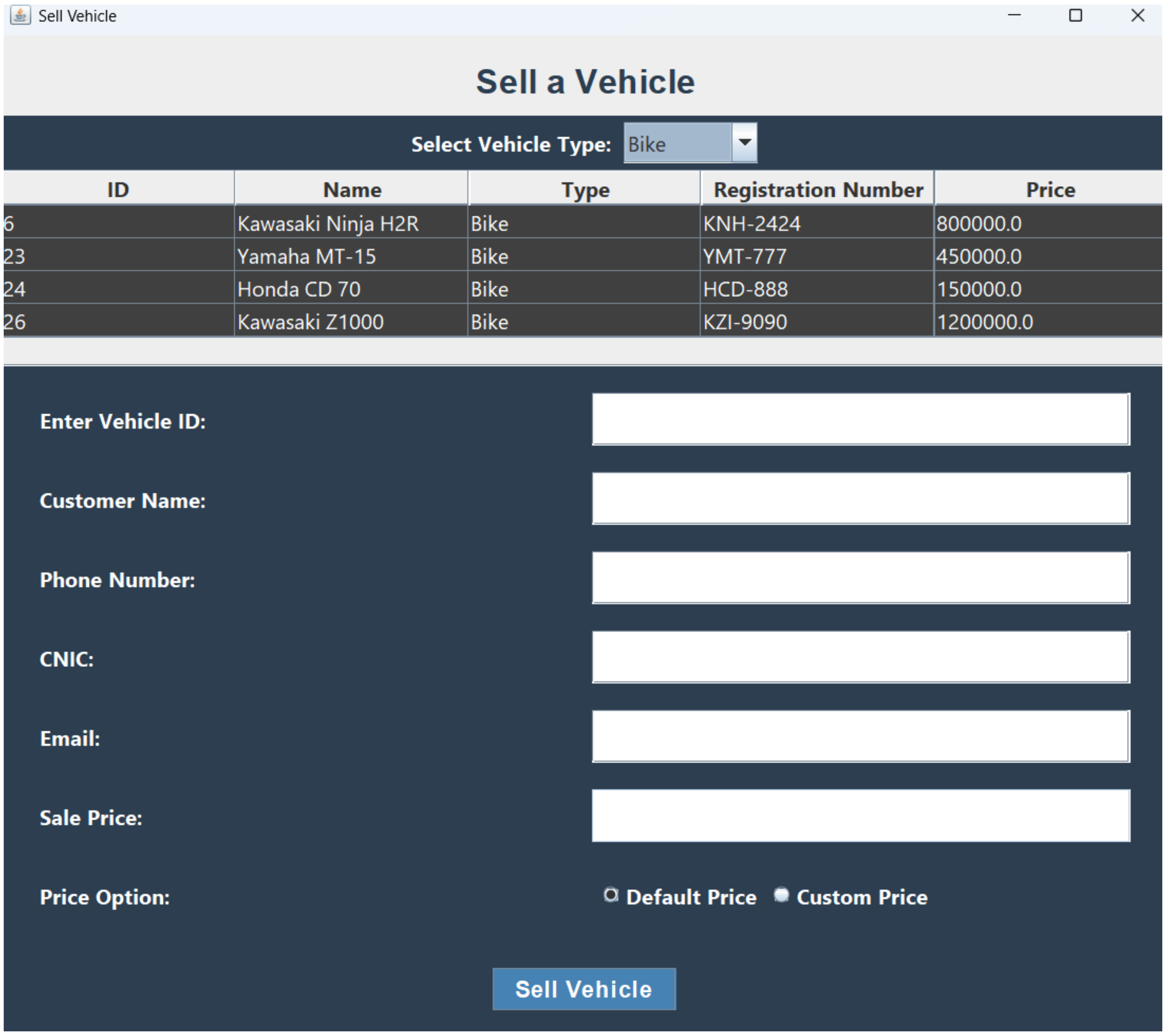


Return Vehicle

Return Vehicle

Customer CNIC:

Vehicle Reg No:


Sell Vehicle

Sell a Vehicle

Select Vehicle Type: Bike

ID	Name	Type	Registration Number	Price
6	Kawasaki Ninja H2R	Bike	KNH-2424	800000.0
23	Yamaha MT-15	Bike	YMT-777	450000.0
24	Honda CD 70	Bike	HCD-888	150000.0
26	Kawasaki Z1000	Bike	KZI-9090	1200000.0

Enter Vehicle ID:	
Customer Name:	
Phone Number:	
CNIC:	
Email:	
Sale Price:	
Price Option:	Default Price Custom Price

5. Tools and Technologies

- **Programming Language**: **Java** with **Swing** and **AWT** for GUI development

- **Database: MySQL** as the relational database management system

- **Database Connectivity: JDBC** for connecting java code with MySQL Database

- Object-Oriented Programming principles

- **IDE: (Notepad++** for code, **VS code** for GUI and **IntelliJ IDEA** for Database connectivity)

6. Project Structure:

Main Java File: Menue.java o Contains two classes: menueFrames and Menu

Vehicle manager file : handles Add, Search, Update, Delete and Display vehicles

Customer manager file : Add, Search, Update, Delete and Display Customers

Database Connectivity file : handles database connection

Dealing manager file: handles Rent and Return Vehicles, selling of vehicles and displaying sold and rented vehicles with database tracking

7. Features Implemented

- **Search Vehicles**: By vehicle type or ID, showing results in a formatted JTable.
- **Insert Vehicles**: Adding new vehicles with details like model, registration, price.
- **Sell Vehicle**: Selecting a vehicle to sell, capturing customer data, and recording sales.
- **Update Vehicle Status**: Mark vehicles as available or sold.

- **Same things for Customers**

- **Display Sold and Rented Vehicles**: Showing all sold and Rented vehicles in a table.
- **Input Validation**: Ensures valid CNIC, phone numbers, and proper formats.
- **Database Integration**: All CRUD operations reflect immediately in the MySQL database.

8. How to Run the Project:

1. Import the project into IntelliJ IDEA.

2. Make sure MySQL server is running.

3. Add your MySQL URL, username, and password in DatabaseConnectivity.java to connect to your server.

4. Restore the provided SQL dump file into MySQL or import the given database **(**VRS (Vehicle Rental & Sales) Database.sql **).**

5. Run the Menu.java file

9. How to Use the System

1. Login: Enter your credentials to access admin (username “admin” and Password “admin”).
2. Manage Vehicles: Add, search, update, and delete vehicles.
3. Sell Vehicle: Select available vehicles by type, enter customer details, and confirm sale.
4. View Sold Vehicles: Check the list of vehicles sold along with customer info.
5. Exit: Properly close the application.

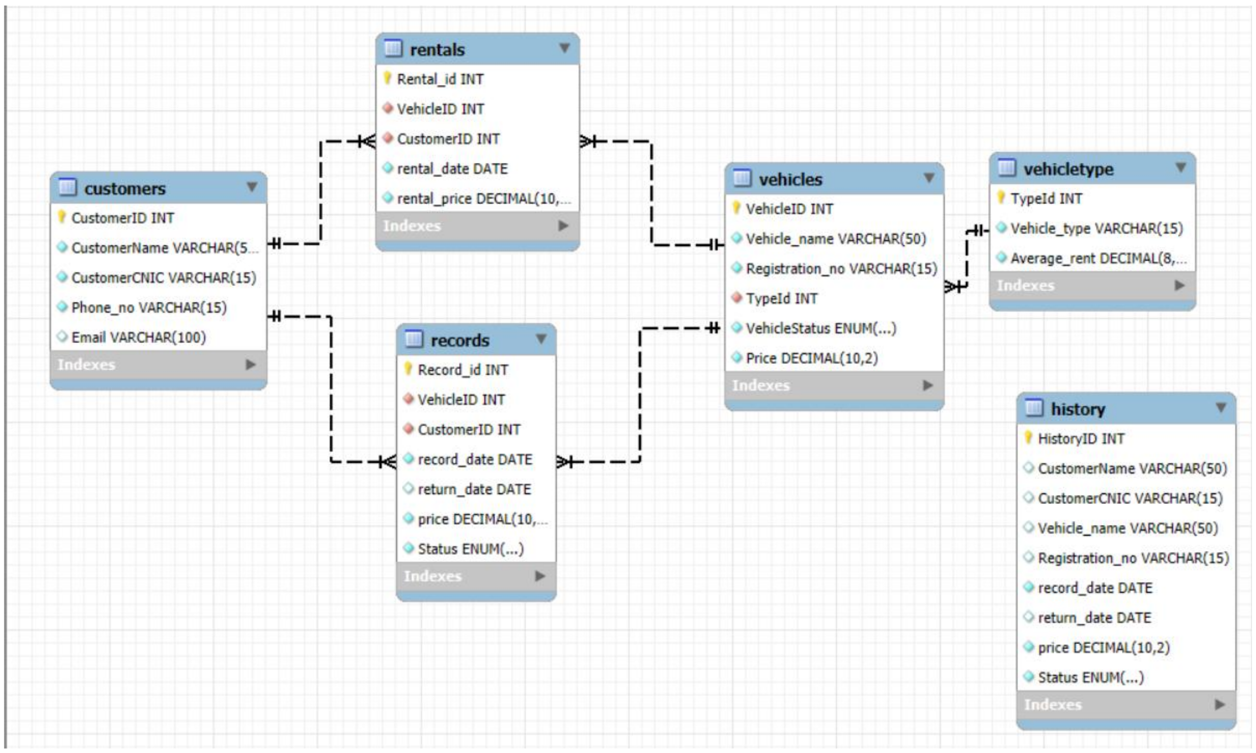
7. Database Design (Project):

The database consists of multiple tables:

Table	**Description**
Customers	Stores customer information
Vehicles	Stores vehicle details
VehicleType	Stores different vehicle types
Rentals	Keeps track of ongoing vehicle rentals
Records	Historical records of sales and rentals
History	Historical records of transactions in past and present too

Constraints such **as NOT NULL, UNIQUE and format validations on CNIC and phone numbers** ensure data integrity.

ER Diagram of Database:



Creation of database and tables in Database:

```
create database Project;
use Project;

-- ========== 1. Customers ==========
create table Customers(
            CustomerID INT primary key auto_increment,
             CustomerName varchar(50) not null check (CustomerName != ''),
             CustomerCNIC varchar(15) unique not null,
             Phone_no varchar(15) not null unique,
             Email varchar(100) unique,
    check(CustomerCNIC regexp "^[0-9]{5}-[0-9]{7}-[0-9]{1}$"),
    check(Phone_no regexp "^[0-9]{4} [0-9]{7}$")
);
desc Customers;



-- ========== 2. Vehicle Types ==========
create table VehicleType(
            TypeId INT primary key auto_increment,
            Vehicle_type varchar(15) not null unique,
            Average_rent decimal(8,2) not null
);
desc VehicleType;


-- ========== 3. Vehicles ==========
create table Vehicles(
            VehicleID INT primary key auto_increment,
               Vehicle_name varchar(50) not null check(Vehicle_name != " "),         -- vehicle name
               Registration_no varchar(15) unique not null,
               TypeId INT not null,
               VehicleStatus ENUM('available', 'rented', 'sold', 'not_available') not null,
               Price decimal(10, 2) not null,
               check (Registration_no regexp '^[A-Z]{1,3}-[0-9]{1,4}$'
            foreign key (TypeId ) references VehicleType(TypeId) on update cascade
);
desc Vehicles;



-- ========== 4. Rentals ==========
create table Rentals(     -- vehicles that are on rent
            Rental_id int  primary key auto_increment,
               VehicleID INT not null unique,
               CustomerID INT not null,
               rental_date Date not null default(current_date),
               rental_price decimal(10, 2) not null,
               foreign key (VehicleID) references Vehicles(VehicleID) on update cascade,
               foreign key (CustomerID) references Customers(CustomerID) on update cascade
);
desc Rentals;



-- ========== 5. Records ==========
create table Records(
            Record_id int  primary key auto_increment,
            VehicleID INT not null,
            CustomerID INT not null,
            record_date Date not null default(current_date),
            return_date Date ,
            price decimal(10, 2) not null,
            Status ENUM('Rented' , 'Sold') not null,
            foreign key (VehicleID) references Vehicles(VehicleID) on update cascade on delete cascade,
   foreign key (CustomerID) references Customers(CustomerID) on update cascade on delete cascade
);
desc Records;


-- ========== 6. History Table (keep records all transactions in past and present too) ==========

create table History (
               HistoryID INT primary key auto_increment,
               CustomerName varchar(50),
               CustomerCNIC varchar(15),
               Vehicle_name varchar(50),
               Registration_no varchar(15),
               record_date date not null,    -- DEFAULT (CURRENT_DATE)
               return_date date,
               price DECIMAL(10,2) not null,
               Status ENUM('Rented', 'Sold') not null
);
desc history;
```

10. Conclusion

This **Vehicle Management System** project successfully demonstrates a functional Vehicle Selling and Renting System using Java OOP concepts, Swing and MySQL database management System. The system is easy to navigate and ensures data integrity while providing all necessary management operations. The project can be extended for further functionalities such as reporting or multi-user support.

THE END