Database Final Project

[Sukkur Institute of Business Administration University]

Department of Computer Science

Vehicle Management System

(Project Report)

Database System (MYSQL)

BS - II SOFTWARE ENGINEERING

Spring 2025

SECTION “A”

Name: **Talha Irfan**

CMS ID: **053-24-0013**

Receiving Authority: **Dr Adil Khan**

Date : 21_May_2025

Project Components & Guidelines

s

Final Project Report of Database

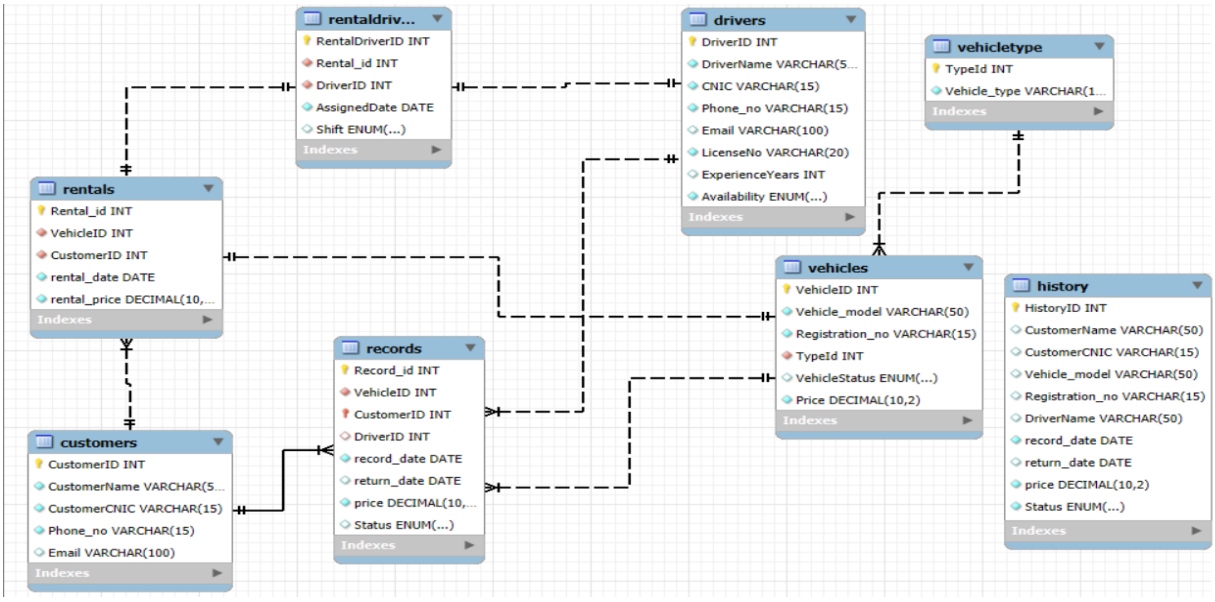
1. Database Design (Project):

The database consists of multiple tables:

Table	Description
Customers	Stores customer information
Vehicles	Stores vehicle details
Vehicle Type	Stores different vehicle types
Drivers	Stores driver data
Rentals	Keeps track of ongoing vehicle rentals
Rental Drivers	Maps drivers assigned to rentals
Records	Historical records of sales and rentals
History	Historical records of transactions(sales and rentals) 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:



Talha Irfan CMS : 053-24-0013

2. Normalization:

Unnormalized Form (UNF)

In the unnormalized form, the data is stored in a single table with repeating groups. This structure does not follow the rules of 1NF.

UNF Table: RentalInfo

Un Normalized Table

CustomerName	CustomerCNIC	Phone_no	Email	Vehicle_model	Registration_no	Vehicle_type	Price	VehicleStatus	DriverName	DriverCNIC	DriverPhone	DriverEmail	LicenseNo	ExperienceYears	Availability	record_date	price	return_date
Ahmed Khan	42101-1234567-1	0300 1234567	ahmed.khan@example.com	Toyota Corolla	ABC 1234	Sedan	2500000	available										
Sara Malik	35202-7654321-9	0312 7654321	sara.malik@example.com	Honda Civic, Toyota Corolla	DEF 5678, ABC 1234	Sedan, Sedan	2800000, 2500000	rented, available	Imran Shaikh, Fahad Sheikh	42101-1122334-5, 42201-9999888-2	0300 1122334, 0304 3344556	imran.shaikh@example.com ,fahad.s@example.com	LIC12345, LIC33344	5,4	Available	4/1/2025, null	15000, null	
Zain Ali	61101-3344556-4	0321 3344556	zain.ali@example.com	Yamaha YBR 125	MNO 999	Bike	300000	rented	Farhan Aziz	61101-3344556-2	0321 3344556	farhan.aziz@example.com	LIC11122	7	Available	5/14/2025	4000	
Hina Shah	42201-2233445-2	0301 2233445	hina.shah@example.com	Suzuki Alto	GHI 4321	Hatchback	1600000	sold								3/10/2025	1600000	
Talha Rizvi	33102-9988776-6	0345 9988776	talha.rizvi@example.com	KIA Sportage	JKL 8765	SUV	5500000	available										
Nida Akbar	42201-3344556-2	0302 3344556	nida.akbar@example.com	Suzuki Cultus	PQR 786	Hatchback	2100000	sold								2/28/2025	2100000	

First Normal Form (1NF)

In 1NF, we eliminate repeating groups and ensure atomicity (single values per field). We separate vehicle and customer details into distinct rows

Split **multi-valued cells** (like Vehicle_model, DriverName, Price, etc.) into **multiple rows**.

First normal form

CustomerName	CustomerCNIC	Phone_no	Email	Vehicle_model	Registration_no	Vehicle_type	Price	VehicleStatus	DriverName	DriverCNIC	DriverPhone	DriverEmail	LicenseNo	ExperienceYears	Availability	record_date	Price	return_date
Ahmed Khan	42101-1234567-1	0300 1234567	ahmed.khan@example.com	Toyota Corolla	ABC 1234	Sedan	2500000	available										
Sara Malik	35202-7654321-9	0312 7654321	sara.malik@example.com	Honda Civic	DEF 5678	Sedan	2800000	rented	Imran Shaikh	42101-1122334-5	0300 1122334	imran.shaikh@example.com	LIC12345	5	Available	4/1/2025	15000	
Sara Malik	35202-7654321-9	0312 7654321	sara.malik@example.com	Toyota Corolla	ABC 1234	Sedan	2500000	available	Fahad Sheikh	42201-9999888-2	0304 3344556	fahad.s@example.com	LIC33344	4	Available			
Zain Ali	61101-3344556-4	0321 3344556	zain.ali@example.com	Yamaha YBR 125	MNO 999	Bike	300000	rented	Farhan Aziz	61101-3344556-2	0321 3344556	farhan.aziz@example.com	LIC11122	7	Available	5/14/2025	4000	
Hina Shah	42201-2233445-2	0301 2233445	hina.shah@example.com	Suzuki Alto	GHI 4321	Hatchback	1600000	sold								3/10/2025	1600000	
Talha Rizvi	33102-9988776-6	0345 9988776	talha.rizvi@example.com	KIA Sportage	JKL 8765	SUV	5500000	available										
Nida Akbar	42201-3344556-2	0302 3344556	nida.akbar@example.com	Suzuki Cultus	PQR 786	Hatchback	2100000	sold								2/28/2025	2100000	

Second Normal Form (1NF)

There are partial dependencies because:

- Customer fields depend only on CustomerID (or CNIC)
- Vehicle fields depend only on VehicleID (or Registration_no)
- Driver fields depend only on DriverID (or DriverCNIC)
- Rental fields depend on Rental or Record IDs

Customers Table: (Dat about customer)

Customers Table

Cusomer ID	Customer Name	Customer CNIC	Phone_no	Email
1	Ahmed Khan	42101-1234567-1	0300 1234567	ahmed.khan@example.com
2	Sara Malik	35202-7654321-9	0312 7654321	sara.malik@example.com
3	Zain Ali	61101-3344556-4	0321 3344556	zain.ali@example.com
4	Hina Shah	42201-2233445-2	0301 2233445	hina.shah@example.com
5	Talha Rizvi	33102-9988776-6	0345 9988776	talha.rizvi@example.com
6	Nida Akbar	42201-3344556-2	0302 3344556	nida.akbar@example.com

Vehicles Table: (Holds data of Vehicel)

Vehicles table					
Vehicle ID	Vehicle Model	Registration No	Vehicle Type	Status	Price
1	Toyota Corolla	ABC 1234	1	available	2500000
2	Honda Civic	DEF 5678	1	rented	2800000
3	Suzuki Alto	GHI 4321	3	sold	1600000
4	KIA Sportage	JKL 8765	2	available	5500000
5	Yamaha YBR 125	MNO 999	4	rented	300000
6	Suzuki Cultus	PQR 786	3	sold	2100000

Vehicles Type Table: (Holds data of Vehicle type and rental price of that type)

Vehicel Type		
TypeId	Vehicle_type	Average_rent
1	Sedan	2650000
2	SUV	5500000
3	Hatchback	1850000
4	Bike	300000

Drivers Table: (Holds driver data)

Driver table							
DriverID	DriverName	CNIC	Phone_no	Email	LicenseNo	ExperienceYears	Availability
1	Imran Shaikh	42101-1122334-5	0300 1122334	imran.shaikh@example.com	LIC12345	5	Available
2	Farhan Aziz	61101-3344556-2	0321 3344556	farhan.aziz@example.com	LIC11122	7	Available
3	Adeel Qureshi	35202-2233445-7	0301 2233445	adeel.q@example.com	LIC67890	3	Assigned
4	Zeeshan Ahmed	42101-8888999-1	0303 1122334	zeeshan.a@example.com	LIC22233	6	Available
5	Fahad Sheikh	42201-9999888-2	0304 3344556	fahad.s@example.com	LIC33344	4	Available
6	Asim Baig	33102-1111000-3	0305 2233445	asim.b@example.com	LIC44455	2	Available

Rental Driver Table: (data about rentals who have taken drivers)

Rental Driver Table				
RentalDriverID	Rental_id	DriverID	AssignedDate	Shift
1	1	1	2025-04-01	Morning
2	2	2	2025-05-14	Evening
3	3	3	2025-03-01	Night
4	4	4	2025-05-15	Morning
5	5	5	2025-05-18	Evening
6	6	6	2025-05-20	Morning

Rentals Table: (Vehicles that are currently on rent)

Rentals Table				
Rental ID	Vehicle ID	Customer ID	Rental Date	Rental Price
1	2	1	2025-04-01	15000
2	5	3	2025-05-14	4000
3	4	2	2025-03-01	12000
4	5	5	2025-05-15	4500
5	1	4	2025-05-18	16000
6	2	6	2025-05-20	3500

Records Table: (Vehicles that are Rented but also returned and sold)

Records Table						
Record ID	Vehicle ID	Customer ID	Record Date	Return Date	Price	Status
01	3	02	4/15/2025	4/20/2025	10000	Rented
02	2	04	3/22/2025	NULL	2500000	Sold
03	5	05	5/1/2025	5/5/2025	22000	Rented
04	1	01	4/1/2025	4/10/2025	17000	Rented
05	6	03	3/15/2025	NULL	280000	Sold
06	4	01	5/10/2025	4/14/2025	15000	Rented

Third Normal Form (1NF)

No transitive dependency exists — i.e., no non-prime attribute should be functionally dependent on another non-prime attribute.

Final tables

1. Customers
2. Drivers
3. Vehicle types
4. Vehicles
5. Rentals
6. Rental Driver
7. Records
8. History (backup of all past and present transactions)

3. Database Design: (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}$"),
  check(Phone_no regexp "^[0-9]{4} [0-9]{7}$")
  check(Email like'%@%.%')
);
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. Drivers ==========
create table Drivers (
          DriverID INT primary key auto_increment,
          DriverName varchar(50) not null check((DriverName) != ''),
          CNIC varchar(15) unique not null,
          Phone_no varchar(15) unique not null,
          Email varchar(100) unique,
          LicenseNo varchar(20) unique not null,
          ExperienceYears INT check(ExperienceYears >= 0),
          Availability ENUM('Available', 'Assigned', 'On Leave') not null,
          check (CNIC regexp '^[0-9]{5}-[0-9]{7}-[0-9]{1}$'),
          check (Phone_no regexp '^[0-9]{4} [0-9]{7}$'),
          check (Email like '%@%.%')
);

desc Drivers;


-- ========== 5. 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;
```

```
-- ========== 6. Drivers Assigned for rentals ==========

create table RentalDrivers (
        RentalDriverID int  primary key auto_increment,
        Rental_id INT not null unique,
        DriverID INT not null unique
        AssignedDate DATE not null default(CURRENT_DATE),
        Shift ENUM('Morning', 'Evening', 'Night') DEFAULT 'Morning',
        foreign key (Rental_id) references Rentals(Rental_id) on update cascade on delete cascade,
        foreign key (DriverID) references Drivers(DriverID) on update cascade
);


-- ========== 7. Records ==========
create table Records(
        Record_id int  primary key auto_increment,
        VehicleID INT not null,
        CustomerID INT not null,
        DriverID Int default 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,
        foreign key (DriverID) references Drivers(DriverID) on update cascade on delete set null
);
desc Records;


-- ========== 8. 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),
        DriverName varchar(50),
        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;
```

4. Final Database View

Customers Table: (Dat about customer)

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	auto_increment
CustomerName	varchar(50)	NO		NULL	
CustomerCNIC	varchar(15)	NO	UNI	NULL	
Phone_no	varchar(15)	NO	UNI	NULL	
Email	varchar(100)	YES	UNI	NULL	

Vehicles Table: (Holds data of Vehicel)

Field	Type	Null	Key	Default	Extra
VehicleID	int	NO	PRI	NULL	auto_increment
Vehicle_model	varchar(50)	NO		NULL	
Registration_no	varchar(15)	NO	UNI	NULL	
TypeId	int	NO	MUL	NULL	
VehicleStatus	enum('available','rented','sold')	YES		NULL	
Price	decimal(10,2)	NO		NULL	

Vehicles Type Table: (Holds data of Vehicle type and rental price of that type)

Field	Type	Null	Key	Default	Extra
TypeId	int	NO	PRI	NULL	auto_increment
Vehicle_type	varchar(15)	NO	UNI	NULL	
Average_rent	decimal(8,2)	NO		NULL	

Drivers Table: (Holds driver data)

Field	Type	Null	Key	Default	Extra
DriverID	int	NO	PRI	NULL	auto_increment
DriverName	varchar(50)	NO		NULL	
CNIC	varchar(15)	NO	UNI	NULL	
Phone_no	varchar(15)	NO	UNI	NULL	
Email	varchar(100)	YES	UNI	NULL	
LicenseNo	varchar(20)	NO	UNI	NULL	
ExperienceYears	int	YES		NULL	
Availability	enum('Available','Assigned','On Leave')	NO		NULL	

Rental Driver Table: (data about rentals who have taken drivers)

Field	Type	Null	Key	Default	Extra
RentalDriverID	int	NO	PRI	NULL	auto_increment
Rental_id	int	NO	UNI	NULL	
DriverID	int	NO	UNI	NULL	
AssignedDate	date	NO		curdate()	DEFAULT_GENERATED
Shift	enum('Morning','Evening','Night')	YES		Morning	

Rentals Table: (Vehicles that are currently on rent)

Field	Type	Null	Key	Default	Extra
Rental_id	int	NO	PRI	NULL	auto_increment
VehicleID	int	NO	UNI	NULL	
CustomerID	int	NO	MUL	NULL	
rental_date	date	NO		curdate()	DEFAULT_GENERATED
rental_price	decimal(10,2)	NO		NULL	

Records Table: (Vehicles that are Rented but also returned and sold)

Field	Type	Null	Key	Default	Extra
Record_id	int	NO	PRI	NULL	auto_increment
VehicleID	int	NO	MUL	NULL	
CustomerID	int	NO	MUL	NULL	
DriverID	int	YES	MUL	NULL	
record_date	date	NO		curdate()	DEFAULT_GENERATED
return_date	date	YES		NULL	
price	decimal(10,2)	NO		NULL	
Status	enum('Rented','Sold')	NO		NULL	

History Table:

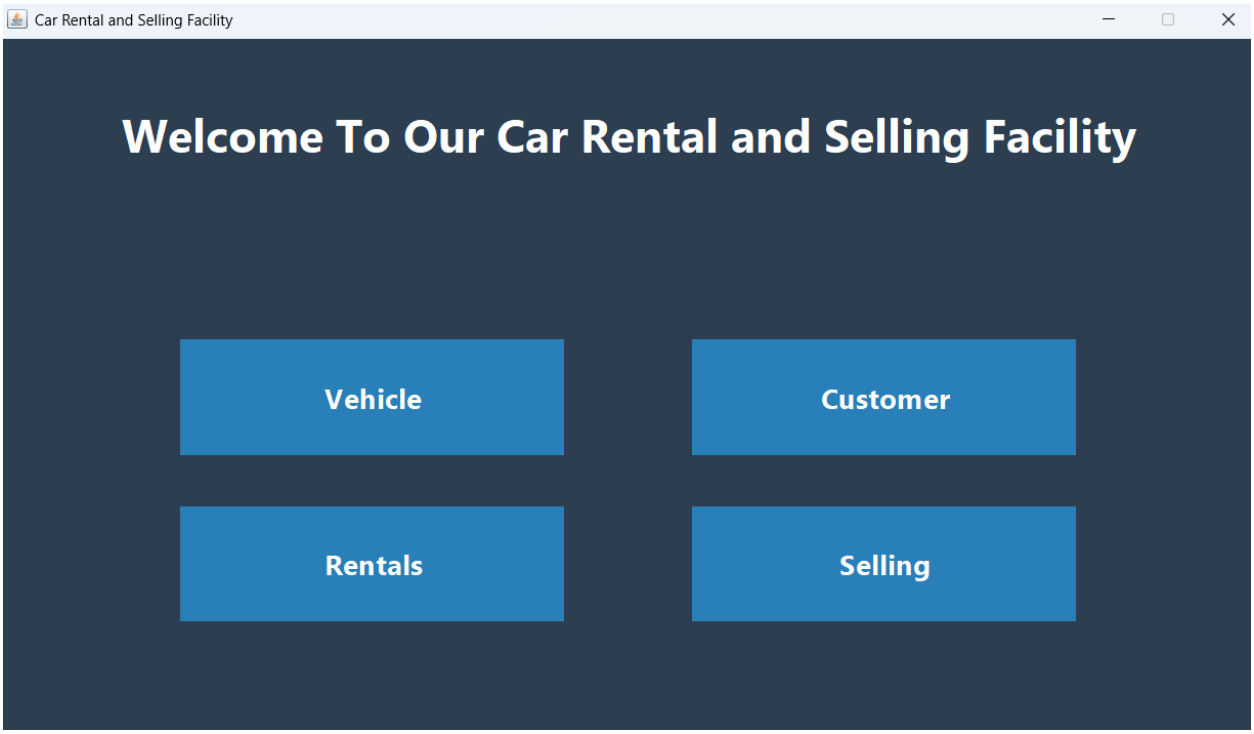
Field	Type	Null	Key	Default	Extra
HistoryID	int	NO	PRI	NULL	auto_increment
CustomerName	varchar(50)	YES		NULL	
CustomerCNIC	varchar(15)	YES		NULL	
Vehicle_model	varchar(50)	YES		NULL	
Registration_no	varchar(15)	YES		NULL	
DriverName	varchar(50)	YES		NULL	
record_date	date	NO		NULL	
return_date	date	YES		NULL	
price	decimal(10,2)	NO		NULL	
Status	enum('Rented','Sold')	NO		NULL	

5. DBMS and Programming Language Integration

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.



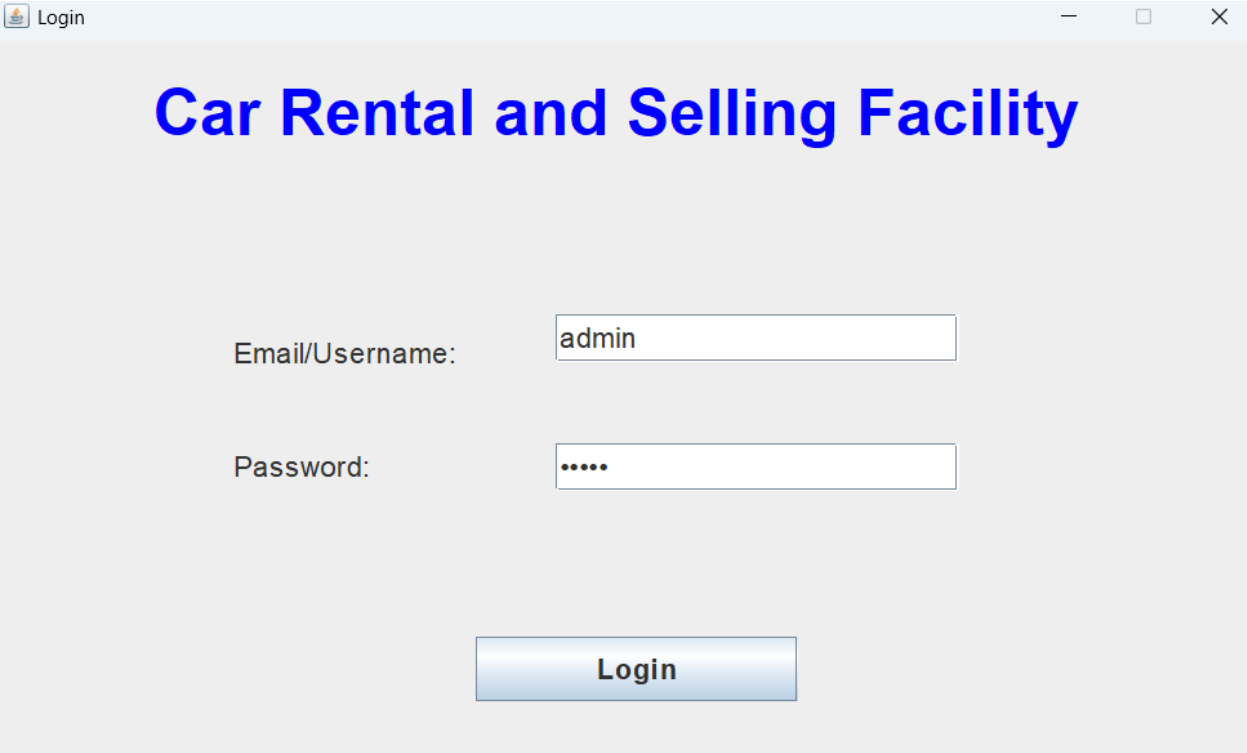
2. Objective

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.

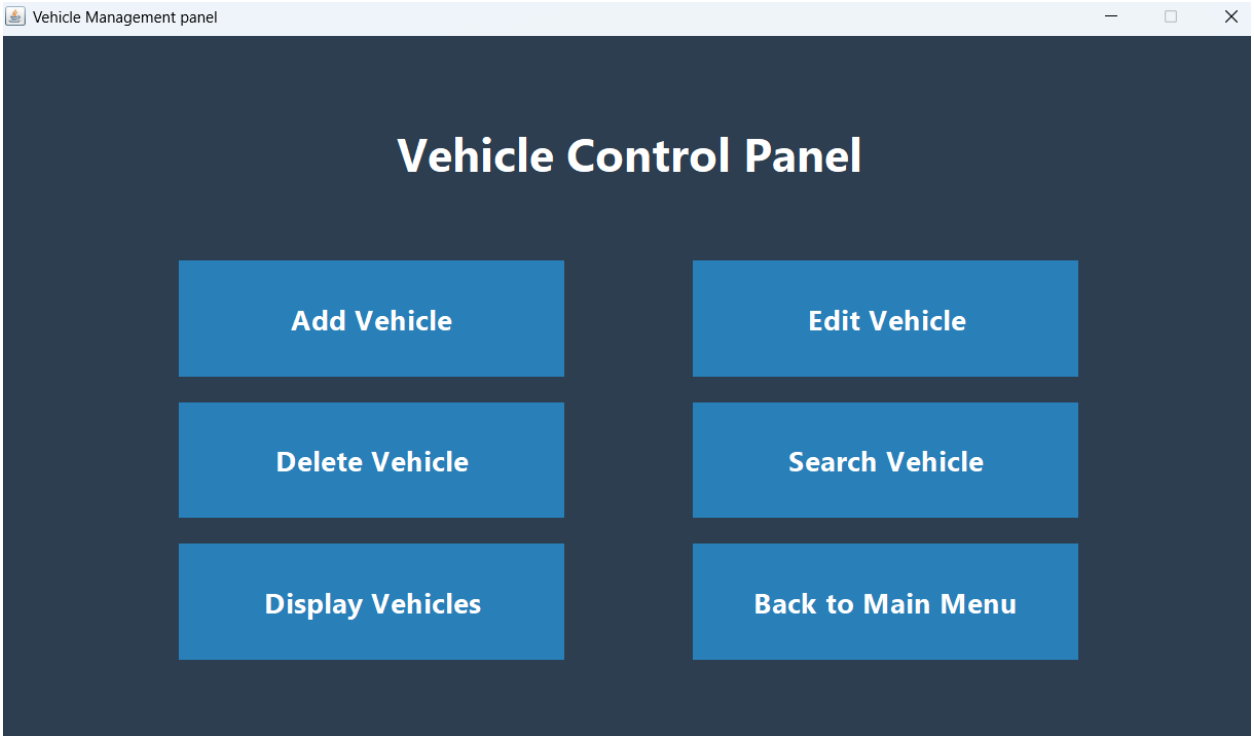
3. Snapshots of the System

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

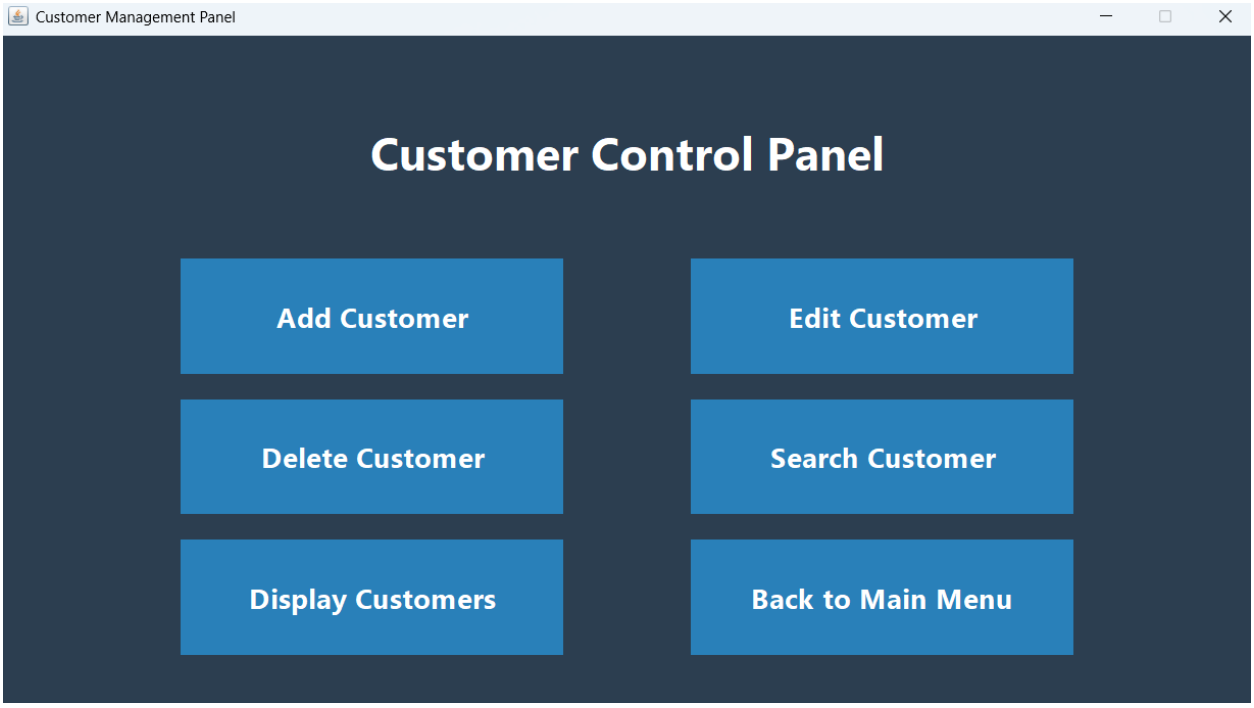
Login page : (username “admin” and Password “admin”)



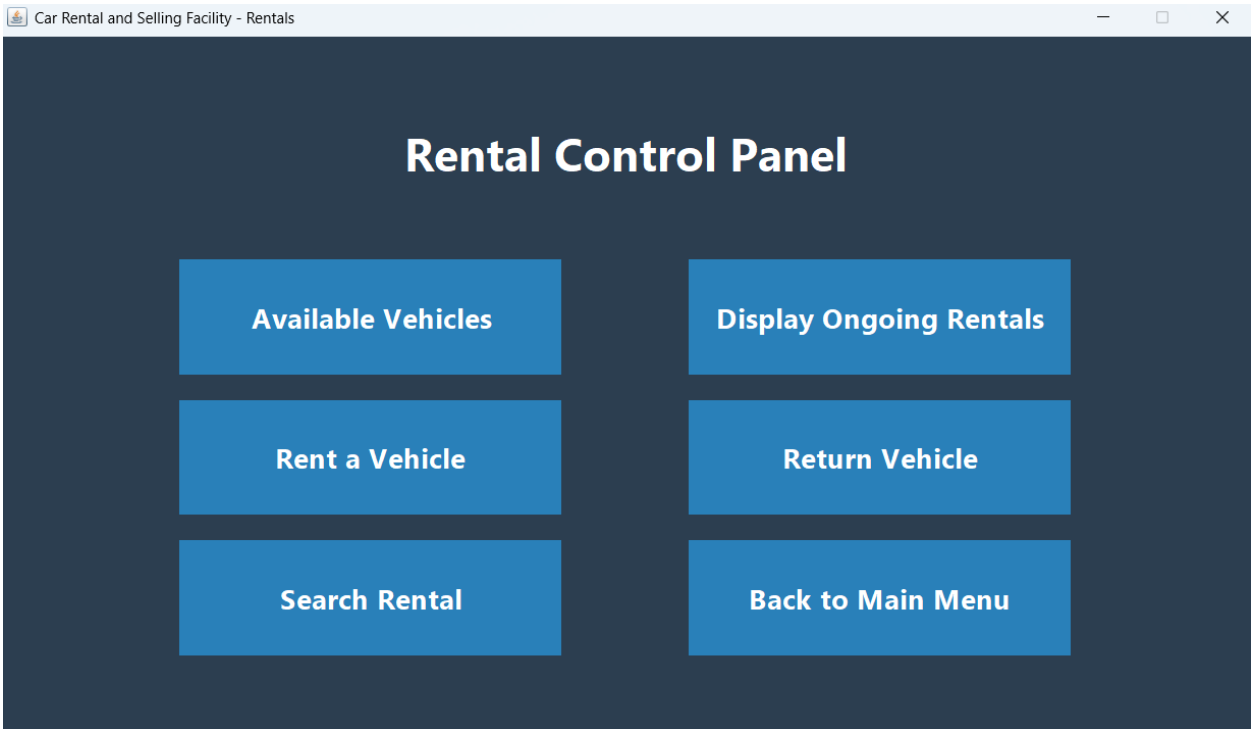
Vehicle Menu :



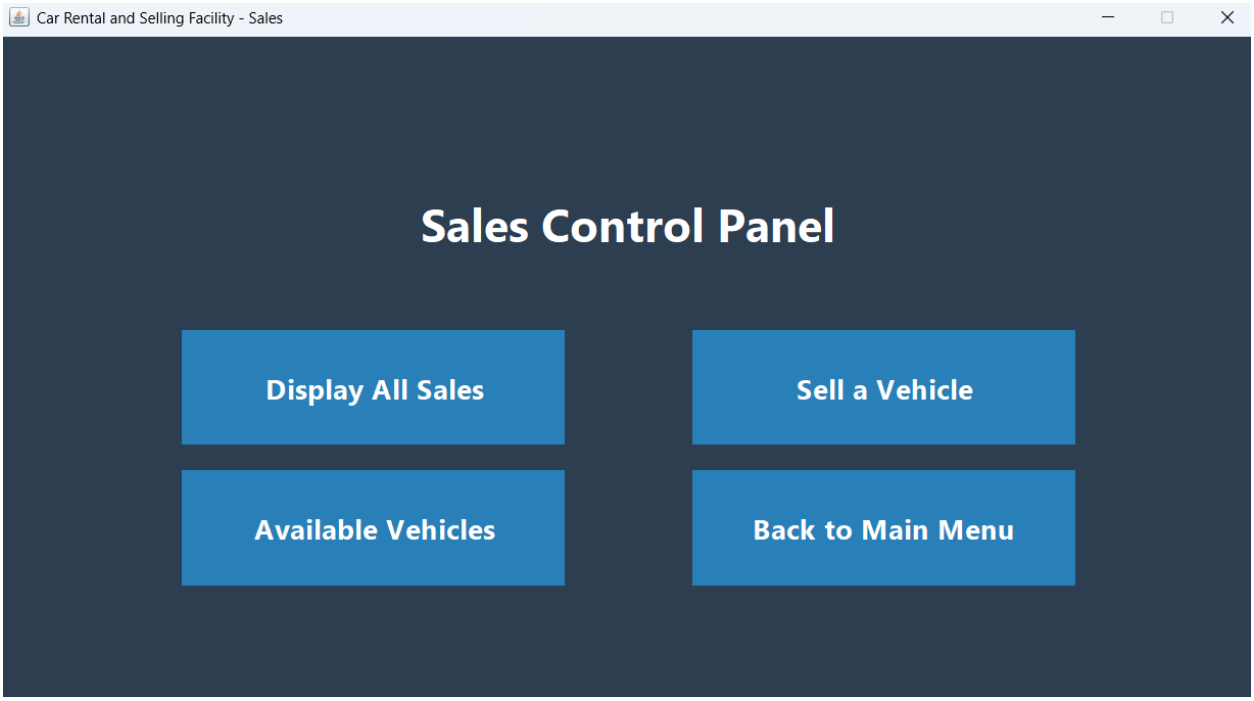
Customer Menu :



Rental Menu :

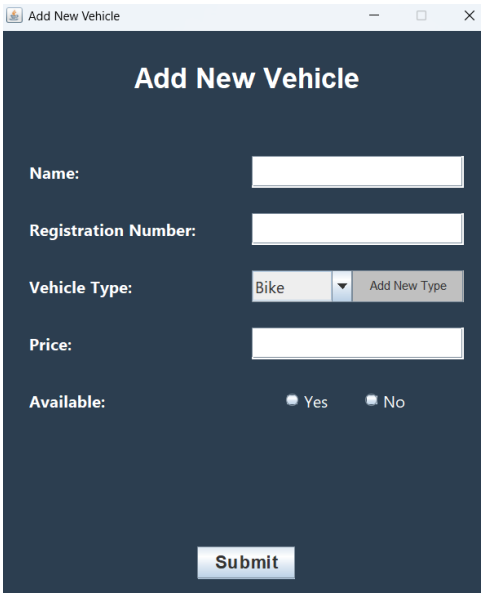


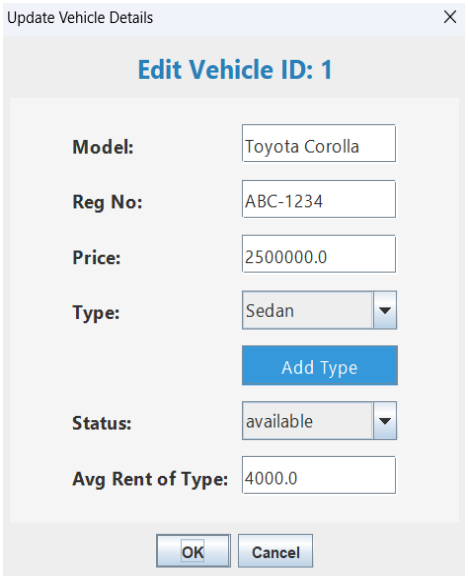
Sales Menu :

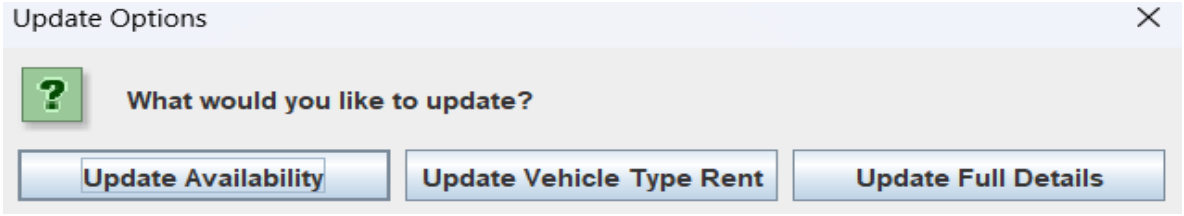


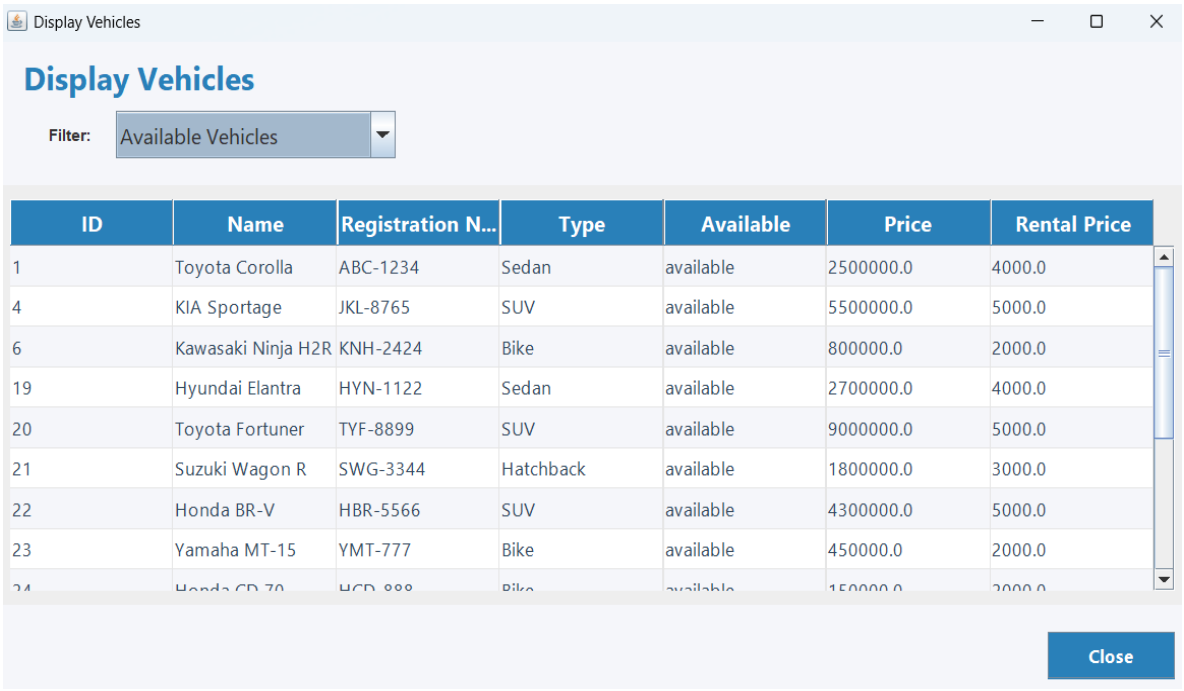
4. Project Features

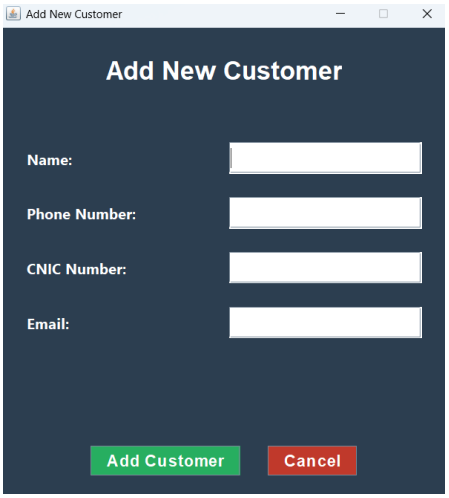
- Add, Search, Update, Delete and Display vehicles
- Add, Search, Update, Delete and Display Customers
- Rent and Return Vehicles with database tracking
- Display records (rented, sold, ongoing rentals) with Customer and Vehicle details in formatted GUI tables
- Login system with Admin (username “admin” and Password “admin”)
- Database integration with MySQL

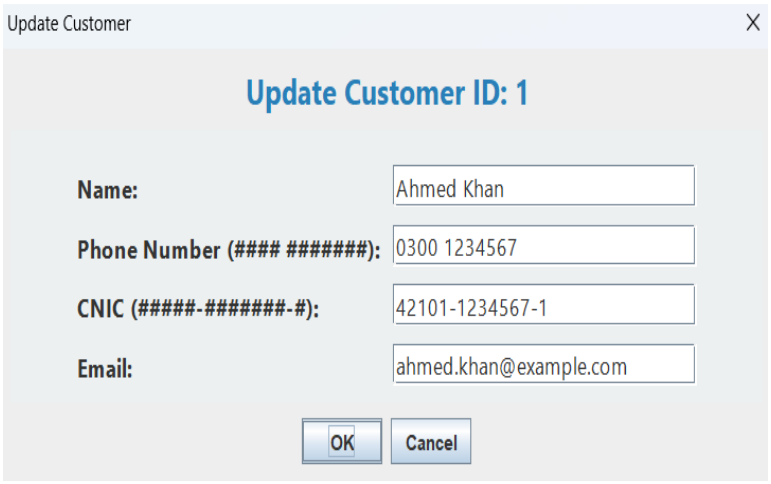

Add New Vehicle
Add New Vehicle
Name:
Registration Number:
Vehicle Type:
Bike
Add New Type
Price:
Available:
Yes
No
Submit

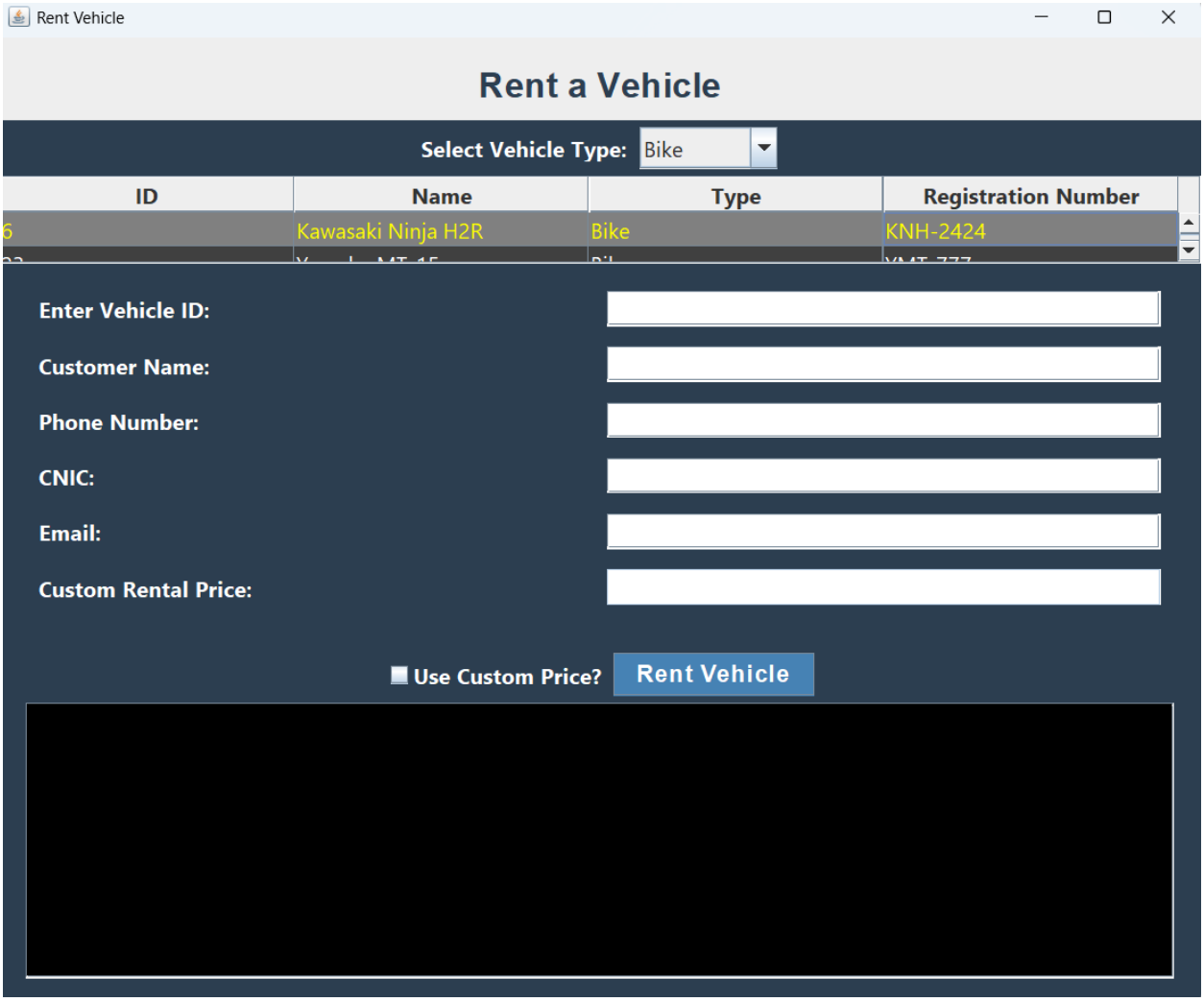

Update Vehicle Details
Edit Vehicle ID: 1
Model:
Toyota Corolla
Reg No:
ABC-1234
Price:
2500000.0
Type:
Sedan
Add Type
Status:
available
Avg Rent of Type:
4000.0
OK
Cancel

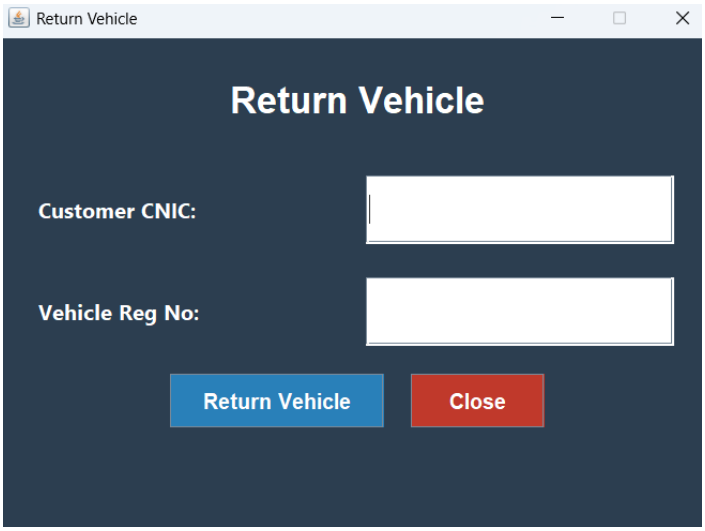

Update Options
What would you like to update?
Update Availability
Update Vehicle Type Rent
Update Full Details

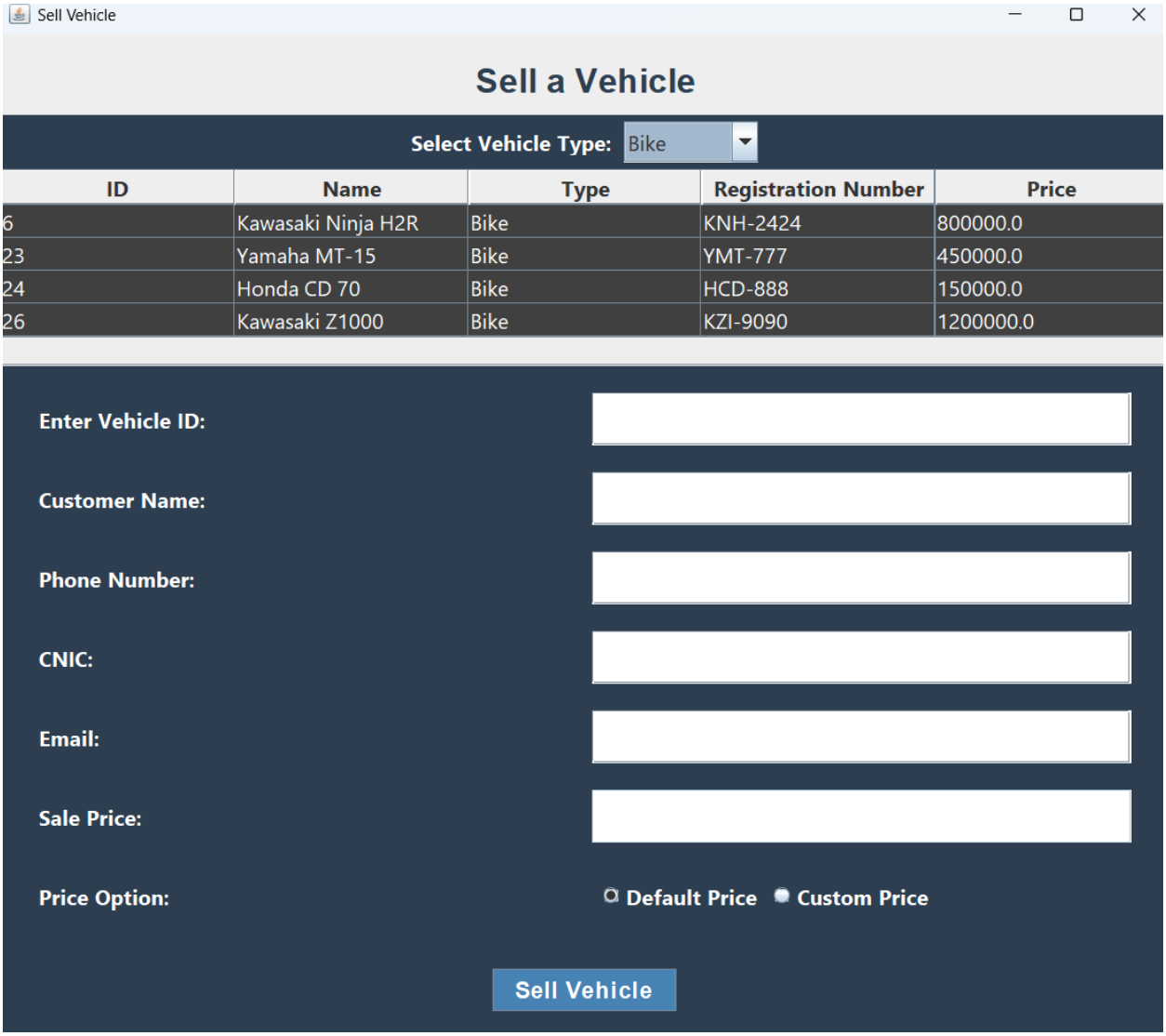

Display Vehicles
Display Vehicles
Filter:
Available Vehicles
ID
Name
Registration N...
Type
Available
Price
Rental Price
1 Toyota Corolla ABC-1234 Sedan available 2500000.0 4000.0
4 KIA Sportage JKL-8765 SUV available 5500000.0 5000.0
6 Kawasaki Ninja H2R KNH-2424 Bike available 800000.0 2000.0
19 Hyundai 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
Close


Add New Customer
Add New Customer
Name:
Phone Number:
CNIC Number:
Email:
Add Customer
Cancel


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


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?
Rent Vehicle


Return Vehicle
Return Vehicle
Customer CNIC:
Vehicle Reg No:
Return Vehicle
Close


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
Sell Vehicle

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 Customer**

- **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.

10. SQL Queries used in the app

Vehicle Manager Queries in Java Code:

Add Task :

- SELECT Vehicle_type FROM VehicleType
- SELECT * FROM VehicleType WHERE Vehicle_type = ?
- INSERT INTO VehicleType (Vehicle_type, Average_rent) VALUES (?, ?)
- SELECT * FROM Vehicles WHERE Registration_no = ?
- SELECT TypeId FROM VehicleType WHERE Vehicle_type = ?
- INSERT INTO VehicleType (Vehicle_type) VALUES (?)
- INSERT INTO Vehicles (Vehicle_name, Registration_no, TypeId, VehicleStatus, Price) VALUES (?, ?, ?, ?, ?)

Search Task :

- SELECT Vehicle_type FROM VehicleType
- SELECT v.VehicleID, v.Vehicle_name, v.Registration_no, vt.Vehicle_type, v.VehicleStatus , v.Price, vt.Average_rent FROM Vehicles v JOIN VehicleType vt ON v.TypeId = vt.TypeId WHERE v.VehicleID = ? OR v.Registration_no = ?
- SELECT v.VehicleID, v.Vehicle_name, v.Registration_no, vt.Vehicle_type, v.VehicleStatus, v.Price , vt.Average_rent FROM Vehicles v JOIN VehicleType vt ON v.TypeId = vt.TypeId WHERE vt.Vehicle_type = ?"

Display Task :

- SELECT v.VehicleID, v.Vehicle_name, v.Registration_no, vt.Vehicle_type, v.VehicleStatus, v.Price, vt.Average_rent FROM Vehicles v JOIN VehicleType vt ON v.TypeId = vt.TypeId WHERE v.VehicleStatus like 'available'
- WHERE v.VehicleStatus != 'available'

Delete Task :

- SELECT * FROM Vehicles WHERE VehicleID = ?

- SELECT * FROM Vehicles WHERE Registration_no = ?
- SELECT * FROM Rentals WHERE VehicleID = ?
- DELETE FROM Vehicles WHERE VehicleID = ?

Update Task :

- UPDATE VehicleType SET Average_rent = ? WHERE TypeId = ?
- UPDATE Vehicles SET Vehicle_name=?, Registration_no=?, TypeId=?, VehicleStatus=?, Price=? WHERE VehicleID=?
- SELECT TypeId FROM VehicleType WHERE Vehicle_type = ?
- INSERT INTO VehicleType (Vehicle_type, Average_rent) VALUES (?, ?)
- SELECT Vehicle_type FROM VehicleType
- UPDATE VehicleType SET Average_rent = ? WHERE TypeId = ?
- UPDATE Vehicles SET VehicleStatus = ? WHERE VehicleID = ?
- SELECT V.*, VT.Vehicle_type, VT.Average_rent, VT.TypeId FROM Vehicles V JOIN VehicleType VT ON V.TypeId = VT.TypeId WHERE V.VehicleID = ? OR V.Registration_no = ?

Customer Manager Queries in Java Code:

Add Customer :

- INSERT INTO Customers (CustomerName, CustomerCNIC, Phone_no, Email) VALUES (?, ?, ?, ?)
- SELECT * FROM Customers WHERE CustomerCNIC = ? OR Phone_no = ? OR Email = ?

Search Customer:

- SELECT * FROM Customers WHERE CustomerName = ?
- SELECT * FROM Customers WHERE CustomerCNIC = ?
- SELECT * FROM Customers WHERE CustomerID = ?

Display Customer:

- SELECT CustomerID, CustomerName, Phone_no, CustomerCNIC, Email FROM Customers

Delete Customer:

- SELECT * FROM Customers WHERE CustomerID = ? OR CustomerCNIC = ? OR Phone_no = ?
- SELECT * FROM Rentals WHERE CustomerID = ?
- DELETE FROM Customers WHERE CustomerID = ?

Update Customer:

- UPDATE Customers SET CustomerName = ?, Phone_no = ?, CustomerCNIC = ?, Email = ? WHERE CustomerID = ?
- UPDATE Customers SET Phone_no = ?, Email = ? WHERE CustomerID = ?

- SELECT * FROM Customers WHERE (Phone_no = ? OR CustomerCNIC = ? OR Email = ?) AND CustomerID != ?
- SELECT * FROM Customers WHERE CustomerID = ? OR CustomerCNIC = ? OR Phone_no = ?
- SELECT * FROM Customers WHERE (Phone_no = ? OR Email = ?) AND CustomerID != ?

Dealing Manager Queries in Java Code:

Rent Vehicle :

- SELECT Vehicle_type FROM VehicleType
- SELECT v.VehicleID, v.Vehicle_name, t.Vehicle_type, v.Registration_no FROM Vehicles v JOIN VehicleType t ON v.TypeId = t.TypeId WHERE t.Vehicle_type = ? AND v.VehicleStatus = 'available'
- SELECT Vehicle_name, VehicleStatus, TypeId FROM Vehicles WHERE VehicleID = ?
- SELECT Average_rent FROM VehicleType WHERE TypeId = ?
- SELECT CustomerID FROM Customers WHERE CustomerCNIC = ?
- INSERT INTO Customers (CustomerName, CustomerCNIC, Phone_no, Email) VALUES (?, ?, ?, ?)
- UPDATE Vehicles SET VehicleStatus = 'rented' WHERE VehicleID = ?
- INSERT INTO Rentals (VehicleID, CustomerID, rental_price) VALUES (?, ?, ?)

Display Ongoing Rentals:

- SELECT c.CustomerName, c.CustomerCNIC, v.VehicleID, v.Vehicle_name, v.Registration_no , r.rental_date, r.rental_price FROM Rentals r JOIN Customers c ON r.CustomerID = c.CustomerID JOIN Vehicles v ON r.VehicleID = v.VehicleID";

Display Available vehicle:

- SELECT v.VehicleID, v.Vehicle_name, v.Registration_no, vt.Vehicle_type, v.Price, vt.Average_rent FROM Vehicles v JOIN VehicleType vt ON v.TypeId = vt.TypeId WHERE v.VehicleStatus = 'available'

Search Rentals:

- SELECT DISTINCT C.CustomerName, C.CustomerCNIC, V.VehicleID, V.Vehicle_name, V.Registration_no FROM Customers C JOIN Vehicles V ON V.VehicleID IN (
  SELECT VehicleID FROM Rentals WHERE CustomerID = C.CustomerID
  UNION
  SELECT VehicleID FROM Records WHERE CustomerID = C.CustomerID
) WHERE C.CustomerCNIC = ?
- SELECT DISTINCT C.CustomerName, C.CustomerCNIC, V.VehicleID, V.Vehicle_name, V.Registration_no FROM Vehicles V JOIN Customers C ON C.CustomerID IN (
  SELECT CustomerID FROM Rentals WHERE VehicleID = V.VehicleID
  UNION

SELECT CustomerID FROM Records WHERE VehicleID = V.VehicleID
) WHERE V.Registration_no = ?

Return Vehicle:

- SELECT r.Rental_id, r.VehicleID, r.CustomerID, r.rental_date, r.rental_price, c.CustomerNsame, v.Vehicle_model FROM Rentals r JOIN Customers c ON r.CustomerID = c.CustomerID JOIN Vehicles v ON r.VehicleID = v.VehicleID WHERE c.CustomerCNIC = ? AND v.Registration_no = ?
- INSERT INTO Records (VehicleID, CustomerID, record_date, return_date, price, Status) VALUES (?, ?, ?, CURRENT_DATE, ?, 'Rented')
- INSERT INTO History (CustomerName, CustomerCNIC, Vehicle_name, Registration_no, record_date, return_date, price, Status) VALUES (?, ?, ?, ?, ?, CURRENT_DATE, ?, 'Rented')
- DELETE FROM Rentals WHERE Rental_id = ?
- UPDATE Vehicles SET VehicleStatus = 'available' WHERE VehicleID = ?

Sell Vehicle:

- SELECT Vehicle_type FROM VehicleType
- SELECT V.VehicleID, V.Vehicle_name, T.Vehicle_type, V.Registration_no, V.Price FROM Vehicles V JOIN VehicleType T ON V.TypeId = T.TypeId WHERE T.Vehicle_type = ? AND V.VehicleStatus = 'available'
- SELECT V.*, T.Vehicle_type FROM Vehicles V JOIN VehicleType T ON V.TypeId = T.TypeId WHERE V.VehicleID = ? AND V.VehicleStatus = 'available'
- SELECT CustomerID FROM Customers WHERE CustomerCNIC = ?
- INSERT INTO Customers (CustomerName, CustomerCNIC, Phone_no, Email) VALUES (?, ?, ?, ?)
- SELECT * FROM Records WHERE VehicleID = ? AND CustomerID = ? AND record_date = CURRENT_DATE AND Status = 'Sold'
- INSERT INTO Records (VehicleID, CustomerID, record_date, price, Status) VALUES (?, ?, CURRENT_DATE, ?, 'Sold'
- INSERT INTO History (CustomerName, CustomerCNIC, Vehicle_name, Registration_no, record_date, return_date, price, Status) VALUES (?, ?, ?, ?, CURRENT_DATE, NULL, ?, 'Sold'
- UPDATE Vehicles SET VehicleStatus = 'sold' WHERE VehicleID = ?

Display Sold Vehicles:

- SELECT c.CustomerName, c.Phone_no, c.CustomerCNIC, c.Email, v.VehicleID, v.Vehicle_name, v.Registration_no, v.Price, vt.Vehicle_type FROM Records r JOIN Customers c ON r.CustomerID = c.CustomerID JOIN Vehicles v ON r.VehicleID = v.VehicleID JOIN VehicleType vt ON v.TypeId = vt.TypeId WHERE r.Status = 'Sold'

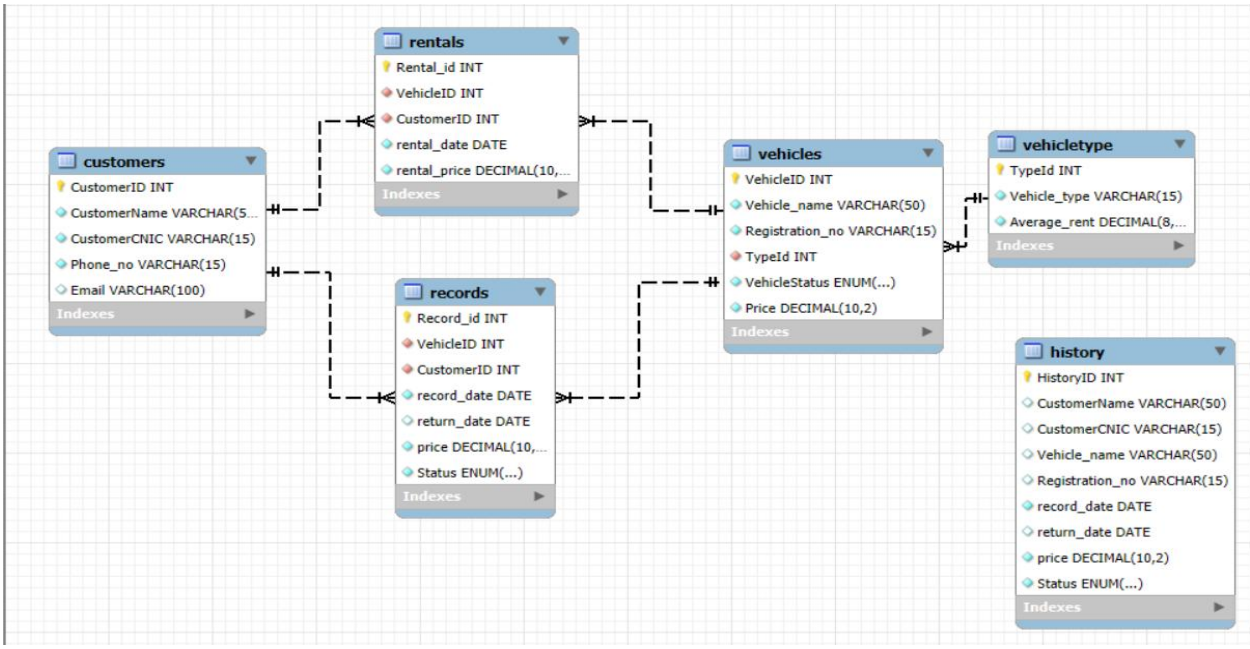
11. Database Design (Project):

The database consists of multiple tables:

Table	Description
Customers	Stores customer information
Vehicles	Stores vehicle details
Vehicle Type	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.

12. ER Diagram of Database linked with this project:



THE END