**Diagram, logo

Description automatically generated with medium confidence**

**DATABASE MANAGEMENT SYSTEMS**

**PROJECT REPORT**

**Project Name:**

**Car Rental Service**

**Project Members:**

**Hammad Ahmed (21CS004)**

**Nawal Ahmed (21CS008)**

**Paras Batool (21CS020)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Department of Computer Systems Engineering**  **Mehran University of Engineering and Technology, Jamshoro** | | | |
| **Course:  Database Management Systems (CS-353)** | | | |
| **Instructor** | Dr. Zartasha Baloch | **Assignment Type** | Complex Engineering Problem |
| **Semester** | 5th | **Year** | 3rd |
| **Submission Deadline** | 01-03-24 | **Assessment Score** | 10 |

# Problem Identification

Managing car rentals can be challenging for both rental companies and customers. Renters often face inconvenience when they have to visit the rental office in person, especially if they have busy schedules or transportation issues. Tasks such as booking a car, picking up or returning vehicles, and managing rental agreements involve time-consuming paperwork and may require multiple visits to the rental office. Additionally, without an accessible and user-friendly desktop application, keeping track of vehicle availability, payments, and rental records can be as challenging as generating reports from an outdated manual system.

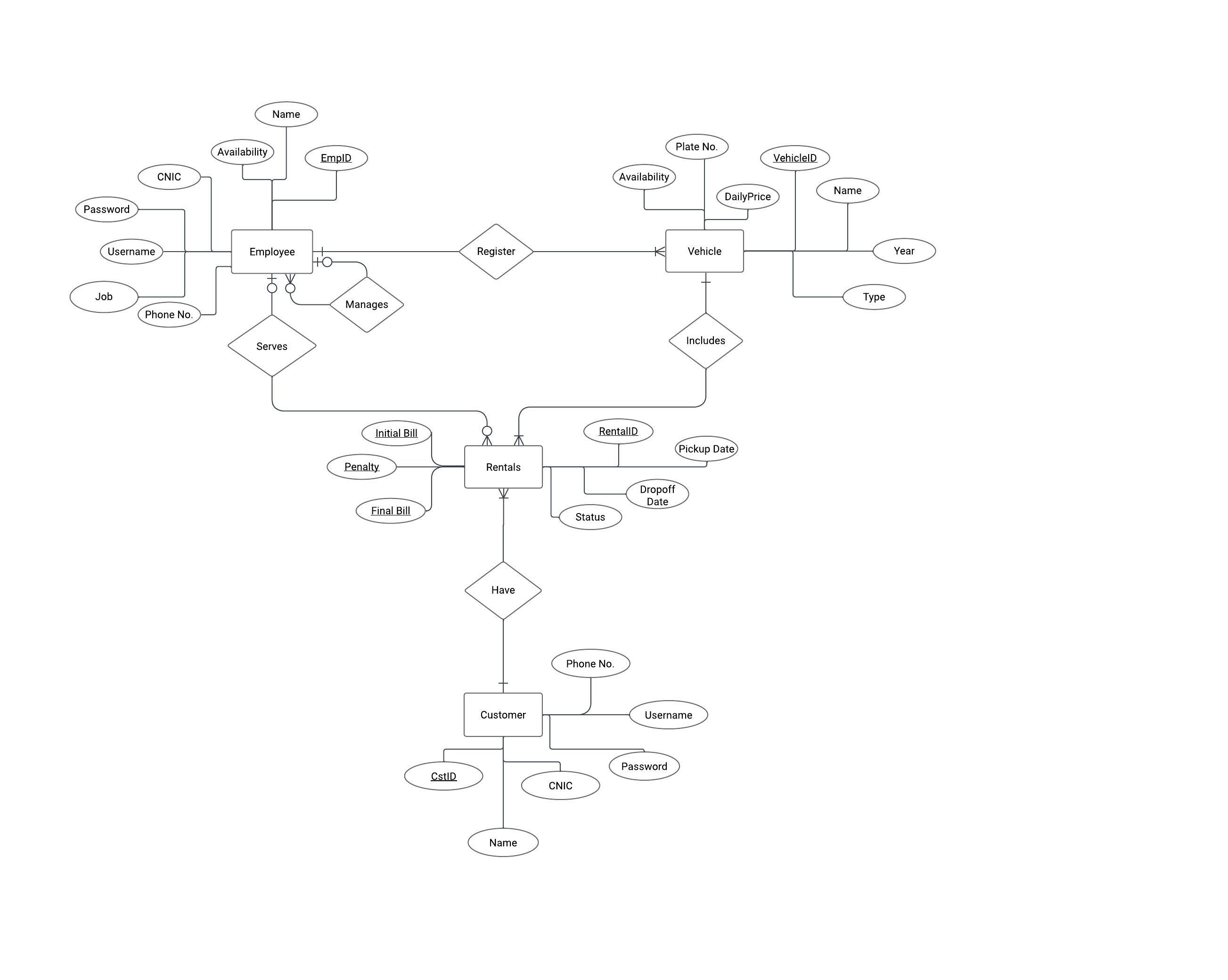
**Solution:**

**“RentMe"** is a comprehensive solution for car rental management, catering to both rental companies and customers. This user-friendly desktop application simplifies booking, reservation management, and rental agreement handling. Integrated payment processing ensures seamless transactions, while robust inventory management tools offer real-time tracking of vehicle availability and maintenance schedules.

[The application necessitates a database to efficiently handle reservations, customer data, vehicle tracking, and financial transactions, ensuring streamlined operations and enhanced user experience](https://www.inettutor.com/source-code/car-rental-system-database-design/).

ER Diagram:

This ER diagram for the Car Rental Management System project outlines the structure of our database in terms of entities and their relationships.



Here’s an explanation based on the description:

Entities and Attributes:

* + **Employee**: This entity stores employee details such as ID(a unique identification number), Name, CNIC, Username, Password, Job title, Phone Number and Availability Status (as “yes” or “no”).
  + **Vehicle**: This entity holds information about the vehicles available for rent, including its ID, Plate Number, Availability status, Daily Price, Name, Year, and Type (as “Sedans”, “Compacts” and “Minivans”).
  + **Customer**: This entity contains customer information, including ID, Name, Phone Number, CNIC, Username & Password for login.
  + **Rentals**: This is the associative entity that captures rental transaction details like Initial Bill, Penalty, Final Bill, Rental ID, Status, Pickup Date, and Dropoff Date.

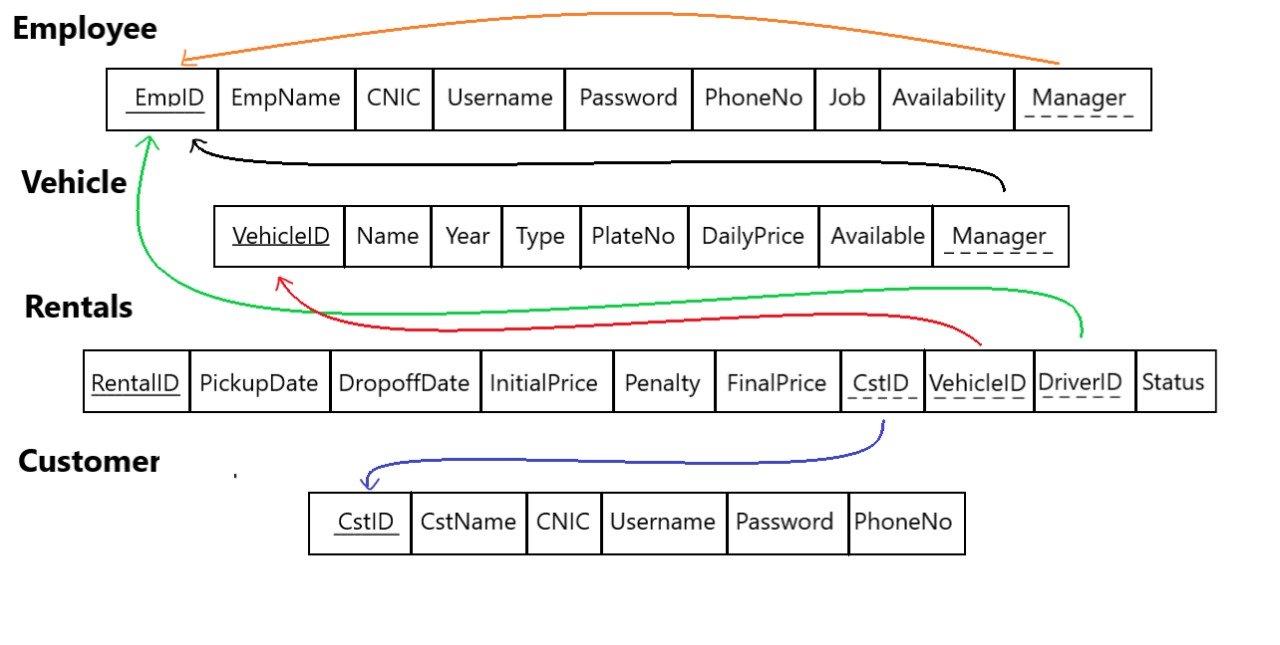
Relationships:

* + An **Employee** (must be HR) can manage one or more employees as Inventory managers or drivers, as indicated by the “Manages” relationship. Also, an **Employee** (must be Inventory Manager) can register one or more **Vehicles**, as indicated by the “Register” relationship.
  + A **Vehicle** can be included in one or more **Rentals**, shown by the “Includes” relationship.
  + A **Customer** can have one or more **Rentals**, as depicted by the “Have” relationship.

This ER diagram is a visual representation of how data is organized within our rental system database, showing the interconnections between different data points which are crucial for managing the car rental process.

# ER Mapping

In this ER mapping we’ve converted the conceptual design of an ER diagram into a logical structure that can be implemented in a relational database.



Here’s an explanation based on the description:

Entities to Tables:

Each entity in the ER diagram, such as Employee, Vehicle, Rentals, and Customer, is transformed into a table in the relational schema.

Attributes to Columns:

**1. Employee Table:**

* **EmpID** **(INT):**

Its constraints include: Primary Key, Not Null, Unique and Auto Increment.

* **EmpName (VARCHAR (20)):**

Its constraints include: Not Null.

* **CNIC (VARCHAR (15)):**

Its constraints include: Not Null and Unique.

* **Username (VARCHAR (20))**:

Its constraints include: Unique.

* **Password (VARCHAR(15))**:

No such constraint.

* **PhoneNo (VARCHAR (12))**:

No such constraint.

* **Job (ENUM (“HR”, “Manager”, “Driver”))**:

Its constraints include: Not Null.

* **Availability (ENUM (“Yes”, “No”))**:

Its constraints include: Not Null.

* **Manager (INT)**:

EmpID of the manager responsible for the recruitment.

Its constraints include: Recursive Foreign Key (reference employee table).

**2. Vehicle Table:**

* **VehicleID** **(INT)**:

Its constraints include: Primary Key, Not Null, Unique and Auto Increment.

* **Name (VARCHAR (20))**:

No such constraint.

* **Year (INT)**:

No such constraint.

* **Type (ENUM(“Sedan”, “Compact”, “Minivan”))**:

Its constraints include: Not Null.

* **PlateNo (VARCHAR (7))**:

Its constraints include: Not Null and Unique.

* **DailyPrice (INT)**:

No such constraint.

* **Available (ENUM(“Yes”, “No”))**:

Its constraints include: Not Null.

* **Manager (INT)**:

EmpID of the manager responsible for the vehicle.

Its constraints include: Foreign Key (reference employee table) and Not Null.

**3. Rentals Table:**

* **RentalID (INT)**:

Its constraints include: Primary Key, Not Null, Unique and Auto Increment.

* **Pickup Date (TIMESTAMP)**:

Its constraints include: Not Null.

* **Dropoff Date (TIMESTAMP)**:

Its constraints include: Not Null.

* **Initial Price (INT)**:

Its constraints include: Not Null.

* **Penalty (INT)**:

No such constraint.

* **Final Price (INT)**:

No such constraint.

* **CstID (INT)**:

CstID of the customer who rent the vehicle.

Its constraints include: Foreign Key (reference customer table) and Not Null.

* **VehicleID (INT)**:

VehicleID of the vehicle that is being rented.

Its constraints include: Foreign Key (reference vehicle table) and Not Null.

* **DriverID (INT)**: Identifier for the driver, if different from the customer.

EmpID of the employee if user requests for a driver.

Its constraints include: Foreign Key (reference employee table).

* **Status (ENUM(“Rented”, “Completed”))**:

Its constraints include: Not Null.

**4. Customer Table:**

* **CstID** **(INT):**

Its constraints include: Primary Key, Not Null, Unique and Auto Increment.

* **CstName (VARCHAR (20))**:

Its constraints include: Not Null.

* **CNIC (VARCHAR (15))**:

Its constraints include: Not Null and Unique.

* **Username (VARCHAR (20))**:

Its constraints include: Not Null and Unique.

* **Password (VARCHAR (15))**:

No such constraint.

* **PhoneNo(VARCHAR (12))**:

No such constraint.

In this mapping:

* The **Employee** table manages the **Employee** table through the Manager attribute also it manages the **Vehicle** table through another Manager attribute.
* The **Rentals** table includes vehicles from the **Vehicle** table indicated by VehicleID column and is associated with drivers from the **Employee** table by DriverID attribute.
* The **Customer** table has a relationship with the **Rentals** table, indicating the rentals made by each customer through CstID column.

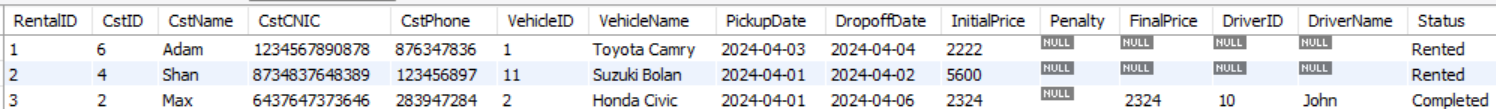
This ER mapping ensures that our database is well-structured, with clear relationships between different entities, which is essential for managing the car rental process efficiently.

# Normalization

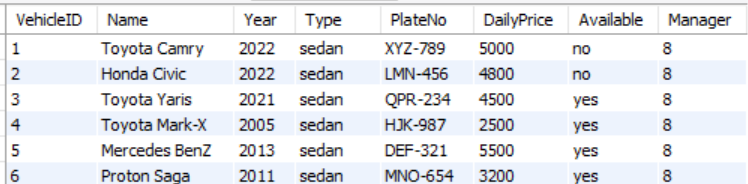
Our approach to normalization follows the principles of First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).

**Initial State of the database Tables before Normalization:**

**Rentals Table:**

****

**Vehicles Table:**

****

**Employee Table:**

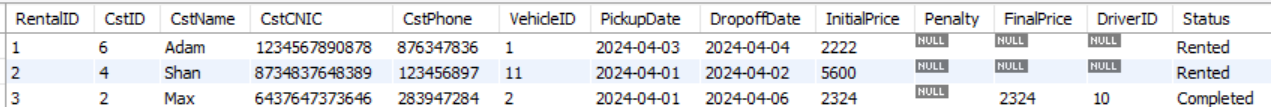
**A screenshot of a computer

Description automatically generated**

**First Normal Form (1NF):**

In the first stage of normalization, we ensured that each table in our database contains only atomic attributes, meaning that each column holds a single value. All the tables in our tables contain atomic attributes hence no changes required. But there is unnecessary repetition of the columns **VehicleName** and **DriverName** in Rentals table. For this reason we have created **VehicleID** and **DriverID** as foreign keys that links to the Vehicle and Employee Table.

**Results:**

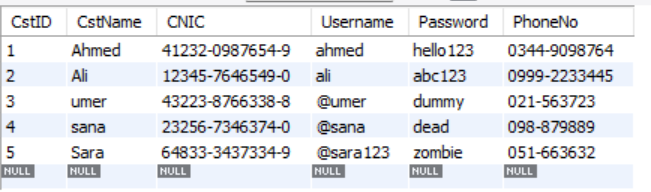


**Second Normal Form (2NF):**

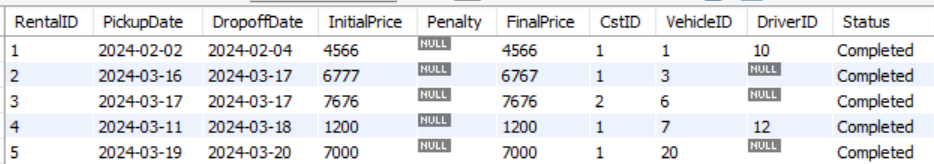
Building upon 1NF, we addressed partial dependencies by ensuring that every non-key attribute is fully functionally dependent on the primary key. This involved breaking down composite key **(RentalID + CstID)** and creating separate table **Customers** for related attributes to remove any partial dependencies.

**Results:**

**Customers Table:**



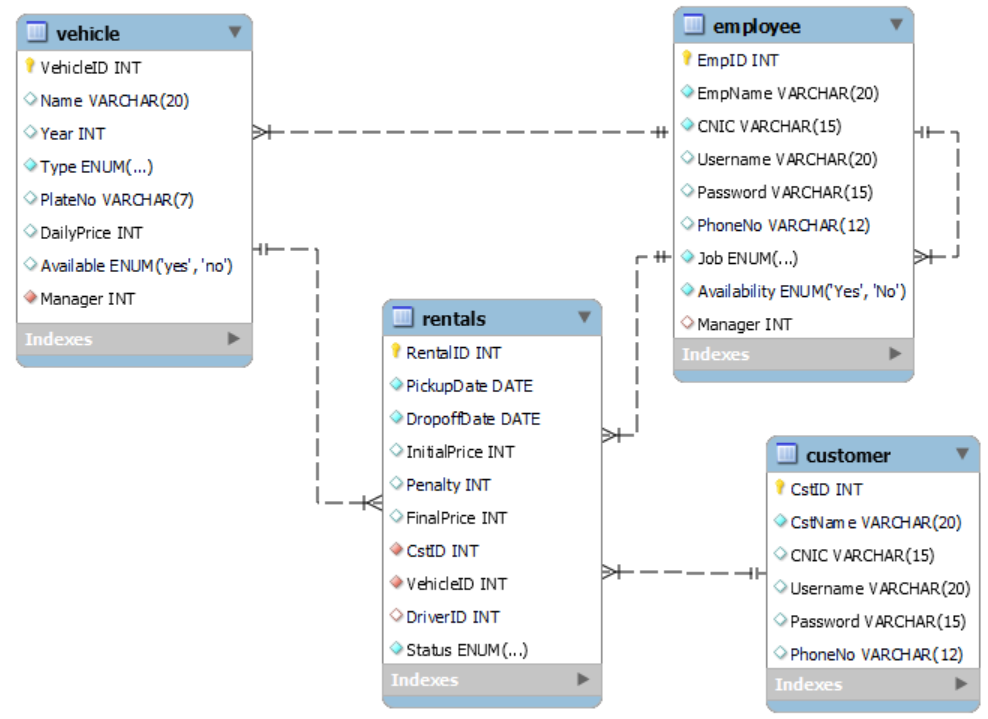
**Rentals Table:**

****

**Third Normal Form (3NF):**

In the final stage of normalization, we focused on eliminating transitive dependencies by ensuring that non-key attributes are not dependent on other non-key attributes. In or database there’s no transitive dependencies left hence it achieve 3NF.

**The Final Schema:**

****

# Implementation

During the implementation phase of the Car Rental Management System, we transformed the design specifications into tangible software components. This stage involved the creation of both the frontend user interface and the backend logic. Additionally, we integrated the MySQL database to handle data storage and retrieval, ensuring a comprehensive solution for managing car rental operations.

During the implementation phase of the Car Rental Management System, we transformed the logical designs into physical design by implementing the defined business logic. This transformation also involved developing both the front-end user interface and backend functionality to ensure smooth interaction between users and the system.

Business Logic

1. **Car Rental Process**

* User selects a car and provides pickup and drop-off dates.
* Total rental duration is calculated based on the pickup and drop-off dates.

1. **Driver Hiring**

* Option for users to hire a driver for an additional daily fee (₹1000 per day).

1. **Initial Billing**

* Initial bill is calculated based on the total rental duration and the daily rental price of the car, along with any additional driver charges.

1. **Penalty Calculation**

* If the car is returned after the drop-off date, a penalty of ₹2000 per day is applied.
* Penalty charges are added to the initial bill to calculate the final bill.

1. **Final Billing**

* Final bill is generated by adding penalty charges (if any) to the initial bill.
* The final amount payable is determined.

Frontend Development

We used NetBeans that uses JavaFX for building the frontend as it offers a wide range of drag and drop UI elements and works well with Java Swing. The interface was designed to be easy to understand and use, with different screens for tasks like booking and managing vehicles. JavaFX controllers helped handle user actions and update the interface as needed.

Backend Development

The backend logic was implemented using Java programming language to handle logic and system functionalities. Java classes were created to manage and perform necessary operation for each task, encapsulating methods for data manipulation and processing. JDBC (Java Database Connectivity) was employed to establish connections with the MySQL database and execute SQL queries for data retrieval and manipulation.

Database Integration

Integration with the MySQL database was essential for data storage and retrieval in the Car Rental Management System. The database schema, designed based on normalized principles, included tables for vehicles, customers, employees, and rentals. JDBC was utilized to establish connections with the database, execute SQL queries, and handle transactions to ensure data integrity.

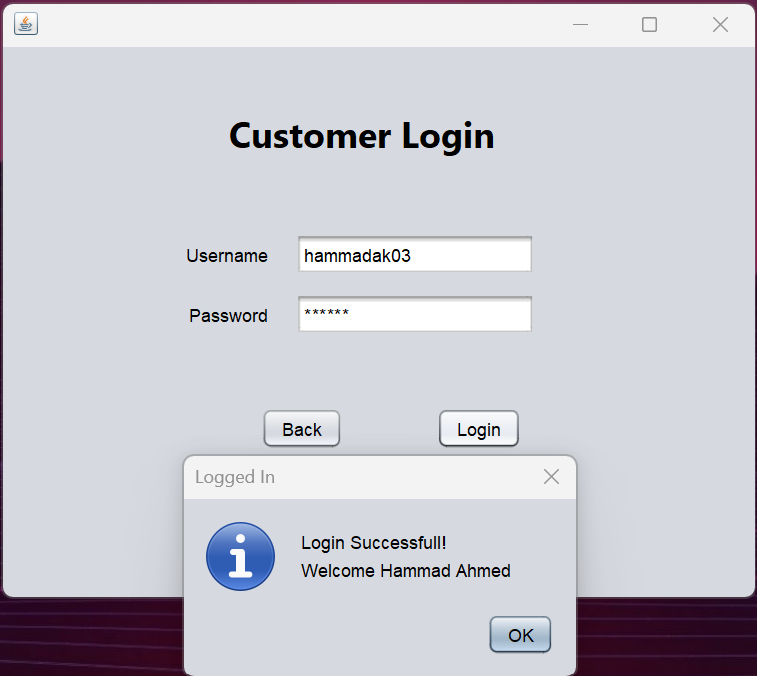
SQL Connection:

A computer screen shot of a program

Description automatically generated

SQL Queries used

* **Customer Login**



* **Employee Login**

A screenshot of a computer

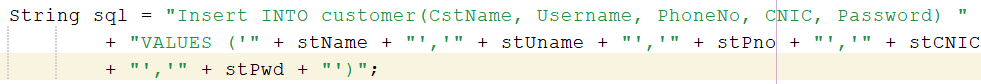
Description automatically generated



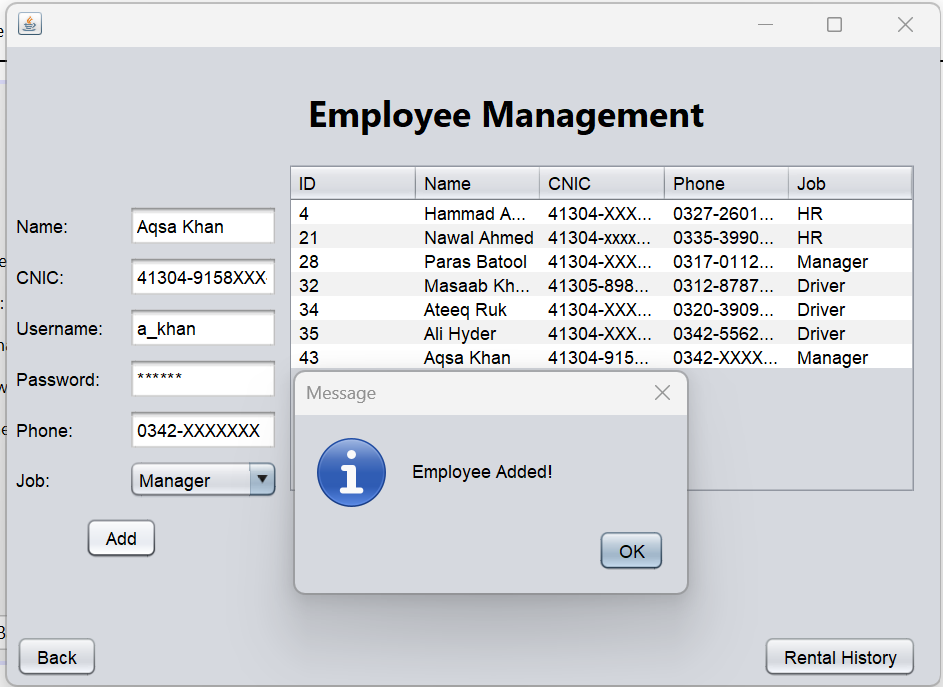
* **Customer Sign-up**

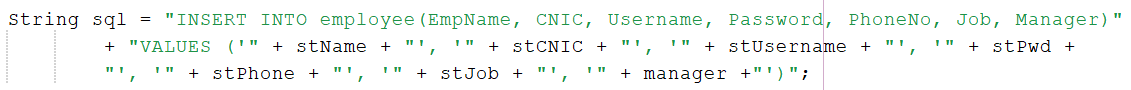
A screenshot of a computer

Description automatically generated

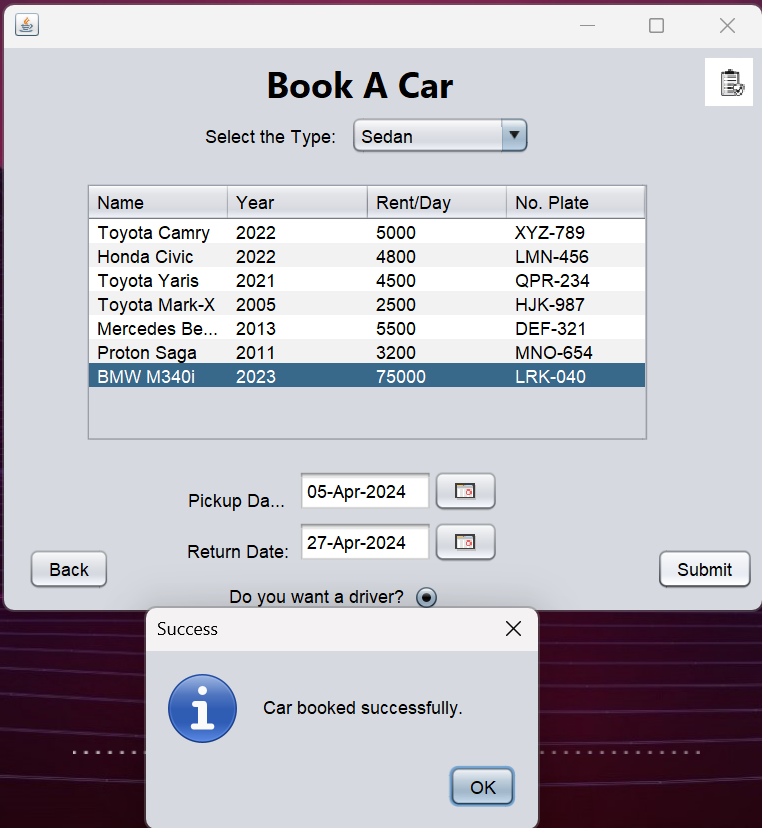


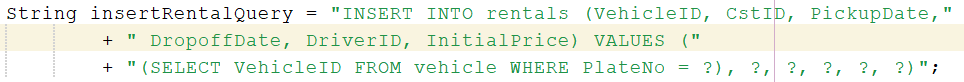
* **Employee Register**



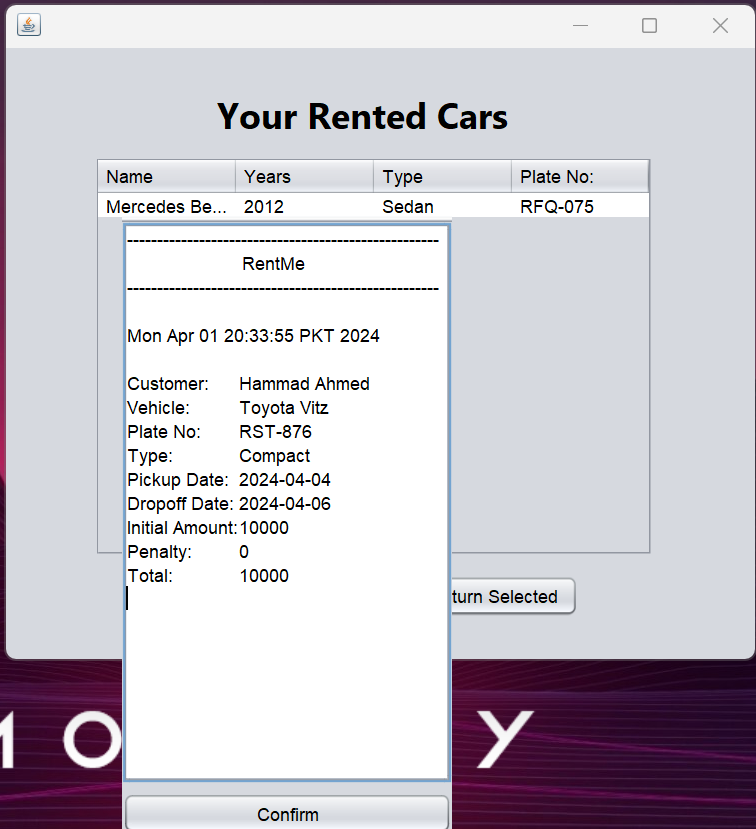


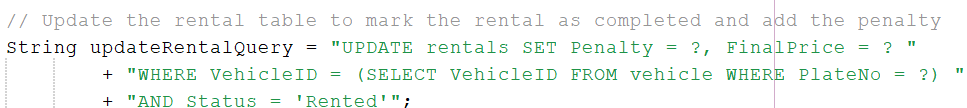
* **Car Booking**



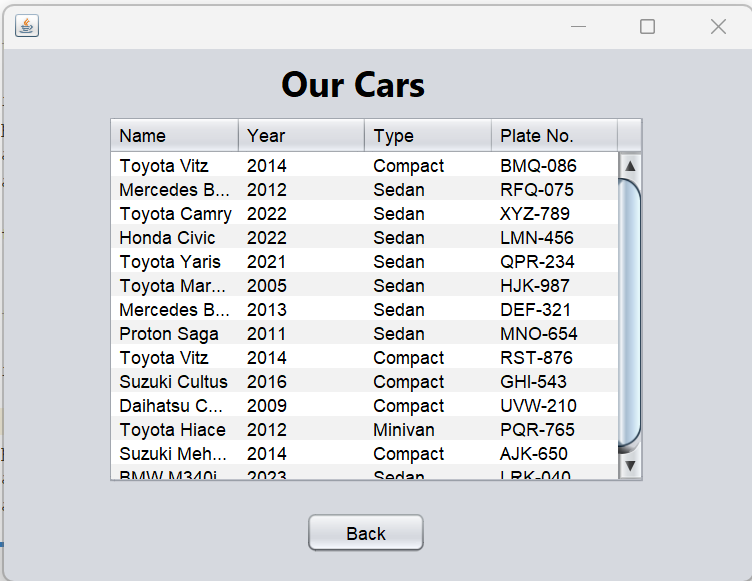


* **Car Return**





* **Show Inventory**





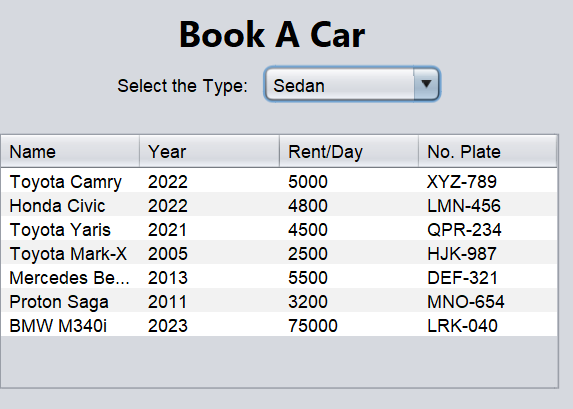
* **Remove Employee**

A screenshot of a computer

Description automatically generated



* **Display Available Cars**





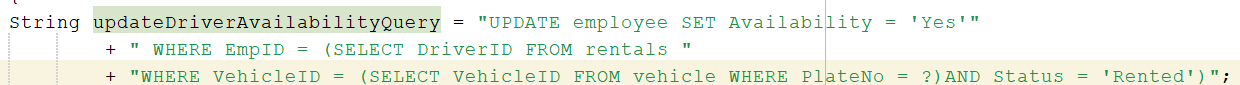
* **Check Driver Availability**

A screenshot of a computer

Description automatically generated



* **Select Driver**
* **Update Driver Status (at the time of rent)**
* **Update Driver Status (at the time of return)**



# Drawbacks and Future Improvements

1. **Limited User Feedback Mechanism**: There's currently no structured mechanism for collecting user feedback and reviews. Incorporating a feedback system would enable users to share their experiences and suggestions for improvement.
2. **Inadequate Vehicle Tracking:** The system does not provide real-time tracking of vehicle locations. Introducing a vehicle tracking system would improve operational efficiency, enhance security, and provide users with accurate pickup and drop-off information.
3. **Multilingual Support**: Provide multilingual support for diverse customer base, offering services and support in multiple languages to enhance accessibility and user experience.
4. **Mobile App Development:** Develop a mobile application for the car rental service, providing users with greater flexibility and convenience to book, manage, and track their rentals on the go.

# Conclusion

The Car Rental Management System is a significant step streamlining car rental operations and improving user experience. Through the development and implementation of desktop application, we have successfully addressed key challenges faced by both rental companies and customers in the car rental process.

While the system has been successfully implemented with essential functionalities such as booking, returns, billing, and database integration, there are areas for improvement and future enhancements. These include implementing a user feedback mechanism, introducing vehicle tracking capabilities, providing multilingual support, and developing a mobile application.

Moving forward, these improvements will be essential for ensuring the continued effectiveness of the system in meeting the needs of users and the industry.

Overall, the project has demonstrated the effective application of engineering knowledge and technical skills in developing a robust solution to address real-world challenges in the car rental industry. With continued focus on improvement and innovation, the Car Rental Management System is dedicated to deliver significant value to both rental companies and customers alike.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rubrics | Assessment | | | | | Marks |
| Unacceptable (0) | Poor (2) | Acceptable (5) | Adequate (8) | Proficient (10) |
| R1 Identification of constraints/requirements/demands/ research gap or challenges well defined | 🞏 | 🞏 | 🞏 | 🞏 | 🞏 |  |
| R2 Engineering knowledge (standards) | 🞏 | 🞏 | 🞏 | 🞏 | 🞏 |  |
| R3 Efficiency of the solution  R4 Technical Writing | 🞏  🞏 | 🞏  🞏 | 🞏  🞏 | 🞏  🞏 | 🞏  🞏 |  |
| Total Marks | | | | | |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rubrics | Unacceptable | Poor | Acceptable | Adequate | Proficient | Score |
| R1 Problem/Requirement Identification | Problem not identified | Problem poorly identified | Problem is identified | Problem is defined adequately. | Problem is identified and analyzed in a well-defined manner. |  |
| R2 Engineering knowledge (standards) | Can not apply engineering knowledge to the solution. | Has difficulty  applying  mathematics to  the solution  of complex  engineering  problems | Correctly applies basic  sciences to the solution  of complex engineering  problems | Correctly applies engineering  fundamentals to the solution of complex  engineering problems | Correctly applies  engineering  specialization to the  solution of complex  engineering problems. |  |
| R3 Efficiency of the solution | Solution does not meet requirements. | A difficult and inefficient solution. | A logical solution that is easy to follow but it is not the most efficient. | Solution is adequately efficient. | Solution is efficient, easy to understand, and maintain. |  |
| R4 Technical Writing | The report is submitted but lacks solutions to major requirements. | The report submitted but not according to the requirements. | The requirements of report writing are not properly addressed. | Reports meets all prescribed requirements. | Reports meets all requirements, and it is prepared in original and corrective way to engage readers. |  |
|  |  |  |  |  |  |  |