CAR RENTAL MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

(OOPS USING JAVA)

Submitted by

PRANAV TANIKAIVELAN 241901079

RAGHUNANDHAN.P.K 241901086

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI - 602105

An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project "CAR RENTAL MANAGEMENT SYSTEM" is the bonafide work of "PRANAV TANIKAIVELAN (241901079), RAGHUNANDHAN .P.K (241901086) who carried out the project work under my supervision.

SIGNATURE

Mrs. Diviya. R

ASSISTANT PROFESSOR

Department of Computer Science

and Engineering (Cyber Security)

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on

_____

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

We hereby declare that the mini project report Car Rental Management System, submitted as part of the curriculum requirements for the Bachelor of Engineering (B.E) degree affiliated to Anna University, is a bonafide work carried out by us under the supervision of Mrs. Diviya. R, Assistant Professor, Department of Computer Science Engineering and Cyber Security, Rajalakshmi Engineering College, Chennai.

This submission represents our ideas in our own words, and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea, fact, or source in our submission. We understand that any violation of the above will be grounds for disciplinary action by the institute and/or the University and may also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Rajalakshmi Engineering College, PRANAV TANIKAIVELAN

Chennai RAGHUNANDHAN .P.K

November 2025

ABSTRACT

The Car Rental Management System is a desktop application developed using Java Swing as the frontend and MySQL as the backend database. It is designed to manage and automate the daily operations of a car rental agency, providing an efficient and user-friendly platform to handle car details, availability status, and rental records.

The system allows administrators to perform essential database operations such as adding, updating, deleting, renting, and returning cars through an intuitive graphical user interface. The use of JDBC (Java Database Connectivity) ensures real-time synchronization between the user interface and the MySQL database.

This application minimizes manual effort, reduces errors, and improves data organization. It also provides a scalable foundation for future enhancements such as customer management, billing, and report generation.

The system's graphical user interface is logically structured into distinct modules for ease of navigation. A central dashboard often serves as the main entry point, offering quick access to 'Manage Cars', 'View Rental Records', and 'Process Transactions'. The 'Manage Cars' section utilizes Swing components like JTables to display the current vehicle fleet, allowing administrators to sort, search, and select specific cars. Data entry and modification are handled through clean, standardized forms (JFrames or JDialogs), which ensure that all necessary information, such as vehicle make, model, registration number, and rental rate, is captured accurately.

A critical aspect of the system is the implementation of robust business logic through JDBC. When an administrator initiates a "Rent" operation, the system performs a real-time check against the MySQL database to confirm the selected vehicle's availability. If available, the system atomically updates the car's status to 'Rented' and creates a new entry in the 'Rentals' table. Conversely, the "Return" process updates the status back to 'Available' (or perhaps 'Needs Maintenance'), ensuring the data integrity of the fleet's inventory and preventing common errors like double-booking. This backend validation is seamless to the user but essential for reliable operations.

Keywords: Java Swing, MySQL, JDBC, CRUD Operations, Database Connectivity, GUI.

ACKNOWLEDGEMENT

We like to convey our sincere appreciation to all who have supported and mentored us during the successful completion of this project work.

We express our profound gratitude to Mr. Benedict J.N., Associate Professor (SG) and Head of the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for furnishing us with essential resources, support, and an enabling environment to execute this project.

We express our profound gratitude to Mrs. Diviya. R, Assistant Professor in the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for her unwavering support, invaluable insights, and collaboration throughout our endeavor.

We would like to convey our gratitude to our faculty members and colleagues for their valuable feedback and encouragement.

We express our gratitude to our families and friends for their steadfast support, patience, and encouragement, which were instrumental in the effective execution of this seminar.

PRANAV TANIKAIVELAN

RAGHUNANDHAN.P.K

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	ACKNOWLEDGEMENT	ii
	LIST OF ABBREVIATION	v
	LIST OF FIGURES	vi
1	INTRODUCTION	1
1.1	Project Overview	1
1.2	Scope of the Work	2
1.3	Problem Statement	3
1.4	Aim and Objectives	4
2	SYSTEM SPECIFICATIONS	5
2.1	Hardware Specifications	5
2.2	Software Specifications	5
3	MODULE DESCRIPTION	6
3.1	Car Management Module	6
3.2	Rental Management Module	6
3.3	Database Module Module	7

3.4	Database Connectivity Module (JDBC Layer)	8
3.5	User Interface Module	8
3.6	ER Diagram	9
3.7	Database Schema	10
4	CODING	11
4.1	Introduction	11
4.2	Database Connection	11
4.3	Car Retrieval Module	12
4.4	CarDAO Module	13
4.5	Rental Logic Module	14
4.6	User Interface	15
4.7	Summary	16
5	SCREENSHOTS	17
6	CONCLUSION AND FUTURE ENHANCEMENT	19
6.1	Conclusion	19
6.2	Future Enhancements	20
	REFERENCES	21

LIST OF ABBREVIATION

Abbreviation	Full Term
CRUD	Create, Read, Update, Delete
SQL	Structured Query Language
JDBC	Java Database Connectivity
Java Swing	Java Framework (for GUI)
GUI	Graphical User Interface
UI	User Interface
ER	Entity-Relationship

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO
1.1	High-Level System Architecture	1
3.1	ER Diagram	9
3.2	Database Schema of Cars	10
5.1	Car Rental Management UI	17

CHAPTER 1

INTRODUCTION

1.1 Project Overview

The **Car Rental Management System** is a desktop application built using **Java Swing** for the user interface and **MySQL** as the backend database. Its purpose is to help car rental agencies manage their vehicles in an organized and efficient way. Instead of using paper records or spreadsheets, the system provides a simple and reliable digital platform to store and update car details.

The application allows the administrator to add new cars, update existing car information, delete cars, and manage their rental status. All these operations are performed through a clean and user-friendly GUI, and the data is stored safely in the MySQL database. The system uses **JDBC** (Java Database Connectivity) to ensure smooth communication between the application and the database. Overall, the project demonstrates how GUI programming and database management can be combined to create a practical tool for real-world use.

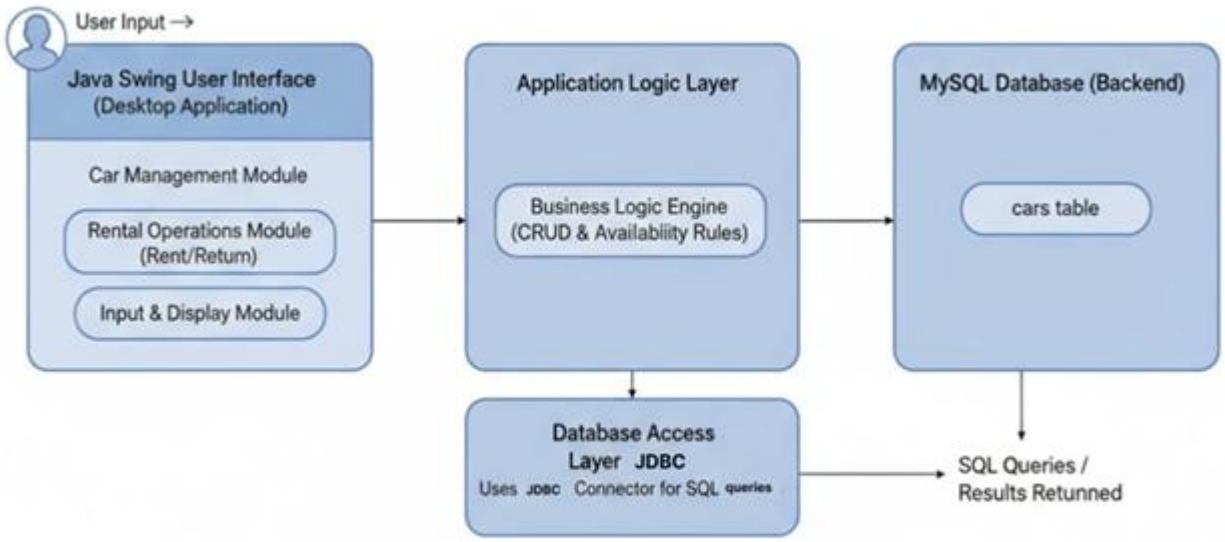

Fig.1.1 High-Level System Architecture

1.2 Scope of the Work

The scope of the Car Rental Management System is centered around building a foundational tool for rental operations. While the current version focuses on car management, it serves as a strong base for more advanced features.

Current Scope

- Managing car details (make, model, year, rate).
- Displaying all records through a graphical table.
- Adding, editing, or deleting cars from the database.
- Updating availability status (rent or return).
- Ensuring data security and integrity through database constraints.

Functional Boundaries

The existing system is designed primarily for administrators or staff, not customers. It is intended for offline desktop usage inside the business office.

Possible Extensions

1. Customer Information Module: Storing customer details, driving license information, and rental history.
2. Billing and Payment System: Automatically calculating rental charges based on duration.
3. Online Booking Portal: Allowing customers to book vehicles through a website or mobile app.
4. Vehicle Maintenance Tracking: Monitoring service schedules and damage reports.
5. Advanced Reporting System: Generating monthly/yearly rental statistics, revenue breakdown, and car usage reports.

The system is useful for:

- Small and medium-scale rental agencies.
- Local businesses seeking digital transformation.
- Academic demonstration of DBMS concepts such as ER diagrams, SQL, and GUI integration.

1.3 Problem Statement

Managing the operations of a car rental agency involves tracking vehicle availability, maintaining accurate records, updating rental status, and ensuring that information remains up-to-date at all times. In many small and medium-sized agencies, these tasks are still performed manually using notebooks, spreadsheets, or simple offline tools. Such methods create a number of challenges that affect efficiency and accuracy.

Manual systems often suffer from **data inconsistency**, where different employees may maintain their own copies of car information, leading to conflicting or outdated entries. Tracking which vehicle is available, which is rented, and when it will return becomes difficult without a central, real-time system. This frequently results in miscommunication, double-bookings, or cars being marked incorrectly as available or unavailable.

Furthermore, searching for or updating information manually is a **time-consuming and error-prone process**. Employees may forget to update records after a rental or return, which leads to incorrect booking decisions. Essential details such as the car's model, year, and rental rate may also be recorded incorrectly, affecting customer service quality and pricing transparency.

Another issue with manual processes is the lack of **structured data storage**. There is no standardized way to store car information, which means that retrieving past records, monitoring trends, or performing audits becomes difficult. Without a digital system, analyzing business performance or generating reports is virtually impossible.

There is also a lack of **security and reliability** in manual systems. Paper records can be lost, damaged, or misplaced, while spreadsheets can be accidentally deleted or corrupted. There is no automatic backup or recovery mechanism.

Considering these problems, there is a clear need for a **centralized, automated, and reliable management system** that can streamline rental operations, maintain consistent and accurate records, and provide real-time updates.

The Car Rental Management System aims to solve these issues by providing a digital platform where car information is stored in a structured database, and rental activities can be managed easily through an interactive user interface. By integrating Java Swing and MySQL, the system ensures accuracy, reduces human error, improves operational efficiency, and offers a scalable foundation for future expansion.

1.4 Aim and Objectives of the Project

The primary aim of the Car Rental Management System is to develop a database-driven rental management application that allows administrators to handle vehicle data and rental activities smoothly, accurately, and efficiently.

Objectives

The objectives of the project are as follows:

Database Integration: To design and implement a MySQL database that stores essential car information in a structured format.

- GUI-Based Interaction: To build a user-friendly desktop interface using Java Swing for performing car management operations without using SQL commands manually.
- CRUD Operations: To enable Create, Read, Update, and Delete operations for all cars in the system with secure and efficient SQL queries.
- Rental Status Management: To manage car availability by allowing administrators to rent or return cars, updating the database instantaneously.
- Real-Time Synchronization: To ensure that the GUI is always in sync with the latest data from the database through JDBC.
- Ease of Use and Reliability: To reduce manual workload and provide a stable and intuitive digital solution for car rental agencies.

These objectives collectively help in improving productivity and minimizing human errors in car rental management.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Minimum Specification
Processor	Dual-core 2.0 GHz or higher
Memory (RAM)	4GB (Minimum), 8GB (Recommended)
Storage	200MB free space
Display	1366x768 resolution

2.2 SOFTWARE SPECIFICATIONS

Component	Specification
Operating System	Windows 10/11, macOS 10.15+, Linux
Front-End	Java Swing
Back-End	MySQL 8.0 or above

Core Language	Java SE 17
Dependencies	MySQL JDBC Driver (connect JAVA to MYSQL), JDK 17 (Core Java Runtime)

CHAPTER 3

MODULE DESCRIPTION

The Car Rental Management System is designed using a modular architecture, ensuring that each part of the application has a clear purpose and can function independently. This structure improves maintainability, scalability, and clarity of the overall system. The project is divided into several modules, each responsible for a core functionality of the system.

3.1 Car Management Module

The Car Management Module is the core component of the system. It handles all operations related to storing and managing car information. This module directly interacts with the database through the JDBC layer to ensure accurate record maintenance.

Responsibilities

- Add new car records to the system.
- View all existing cars stored in the database.
- Update details of any car (make, model, year, rental rate).
- Delete car records that are no longer needed.
- Validate input data before updating the database.

Key Features

- Prevents incomplete or invalid entries.
- Maintains unique IDs for each car.
- Keeps a consistent and updated list of vehicles.

This module forms the foundation of the application, as all operations revolve around car data.

3.2 Rental Management Module

The Rental Management Module deals with the availability and rental status of each car. It ensures accurate tracking of whether a car is currently available for rent or has been rented out.

Responsibilities

- Mark a car as rented when it is issued to a customer.

- Mark a car as returned when it is received back.
- Prevent renting a car that is already unavailable.
- Maintain accurate real-time availability status in the database.

Key Features

- One-click Rent and Return operations.
- Automatic synchronization of rental status with the database.
- Checks to avoid inconsistent rental states (e.g., renting an already rented car).

This module ensures smooth rental operations and eliminates conflicts in car availability.

3.3 Database Module (MySQL)

The Database Module stores all the system data in structured tables. It guarantees persistent storage and ensures that the system retains all information even after shutdown.

Responsibilities

- Store car data in the cars table.
- Maintain consistent and valid data using constraints.
- Support CRUD operations through SQL queries.
- Provide fast access to large sets of data.

3.4 Database Connectivity Module (JDBC Layer)

The JDBC module acts as the communication bridge between the Java program and MySQL database.

Responsibilities:

- Establish secure connection to MySQL.
- Execute SQL commands (Insert, Update, Delete, Select).
- Convert SQL results into Java objects for display.
- Handle database errors and maintain stable transactions.
- Use PreparedStatement for security against SQL injection.
- This module ensures efficient and error-free data flow throughout the application.

3.5 User Interface Module

The UI module is the interaction layer between the user and the system. Built using Java Swing, it provides a simple and clean interface for managing car rental operations.

Responsibilities

- Display car records in a table format.
- Provide input fields for entering car details.
- Include action buttons (Add, Update, Delete, Rent, Return, Refresh).
- Show confirmation dialogs, alerts, and error messages.
- Update the display based on database changes.

This module enables the user to efficiently operate the system without needing any technical knowledge or SQL commands.

3.6 ER Diagram

The Entity–Relationship (ER) Diagram shown above represents the logical structure of the database used in the **Health Tracker** application. The system primarily operates on the **Food_Nutrition** entity, which stores essential nutritional information for various food items, such as calories, carbohydrates, proteins, and fats per 100 grams. This entity forms the foundation of the application, enabling accurate nutritional calculations based on user input. Each food item is stored as a unique record in the Food_Nutrition table, identified by a Primary Key (PK). The attributes stored in this entity ensure that the system can efficiently retrieve nutritional values whenever the user enters a food name and quantity.

Although the current version of the Health Tracker uses a single primary entity, the design is **scalable and future-ready**. Additional entities such as **User_Details** and **Daily_Intake_Log** can be easily introduced to extend the system's functionality. In such an extended design, the Daily_Intake_Log entity acts as a bridge between the User and Food_Nutrition entities, storing details of each food item consumed by a user, along with calculated totals such as calories or macronutrient values.

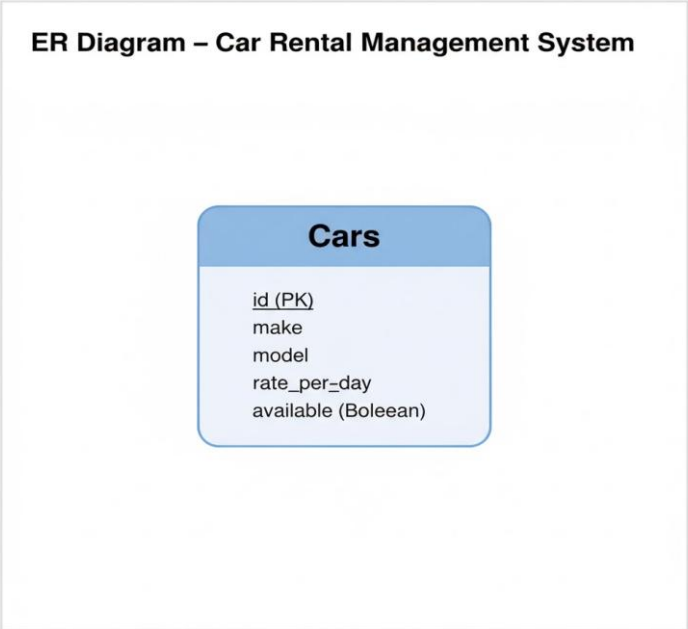

Fig.3.1 ER Diagram

3.7 Database Schema

Explanation: Database Schema of Cars

The database schema defines the structure of the database used in the Car Rental Management System. The system uses a single table in the current version, as the focus is on managing car information and rental availability. The database is designed to store car details in a structured and normalized form, ensuring efficient storage, retrieval, and modification of data.

The database is created in **MySQL**, and the schema is shown below.

Column	Type	Nullable	Indexes
id	int	NO	PRIMARY
make	varchar(100)	NO	
model	varchar(100)	NO	
year	int	NO	
rate_per_day	decimal(8,2)	NO	
available	tinyint(1)	NO	

Fig.3.2 Database Schema of Cars

CHAPTER 4

CODING

4.1 Introduction

This chapter provides an overview of the key coding components used in the development of the Car Rental Management System. The system is implemented using **Java Swing** for the user interface, **JDBC** for database connectivity, and **MySQL** for backend storage. The code is organized into logical sections, including database connectivity, data access objects, core logic, and user interface implementation.

The following sections highlight the important portions of the source code essential for the functioning of the system.

4.2 Database Connection Code

The system uses **Java Database Connectivity (JDBC)** to establish a connection with the MySQL database. The DriverManager class is used to open the connection, and PreparedStatement is used for executing secure SQL queries.

```
private static final String DB_URL =
"jdbc:mysql://localhost:3306/carrental?serverTimezone=UTC";

private static final String DB_USER = "root";

private static final String DB_PASS = "yourpassword";

private Connection connect() throws SQLException {

   return DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);

}
```

4.3 Car Retrieval Module

The Car class represents the structure of a car record. It maps directly to the cars table in the database.

```
public class Car {

   public int id;

   public String make;
```

```
    public String model;
    public int year;
    public double ratePerDay;
    public boolean available;
    public Car() {}
    public Car(int id, String make, String model, int year, double ratePerDay, boolean available)
{
        this.id = id;
        this.make = make;
        this.model = model;
        this.year = year;
        this.ratePerDay = ratePerDay;
        this.available = available;
    }
}
```

4.4 CarDAO Module

The **CarDAO** class contains the CRUD operations used to interact with the database. It provides methods to add, update, delete, and retrieve cars.

```
public List<Car> listCars() {
    List<Car> cars = new ArrayList<>();
    String query = "SELECT * FROM cars ORDER BY id";
    try (Connection con = connect();
        PreparedStatement ps = con.prepareStatement(query);
        ResultSet rs = ps.executeQuery()) {
```

```
        while (rs.next()) {
            cars.add(new Car(
                rs.getInt("id"),
                rs.getString("make"),
                rs.getString("model"),
                rs.getInt("year"),
                rs.getDouble("rate_per_day"),
                rs.getBoolean("available")
            ));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return cars;
}
```

4.5 Add Food Module

When a food item is not found in the database, the system provides an option to add it. This module inserts the new food item into the food_nutrition table.

```
private void addFoodToDatabase(String name, double cal, double carb, double prot, double fat)
{
    try {
        Connection conn = getConnection();
        PreparedStatement ps = conn.prepareStatement(
```

```
            "INSERT INTO
food_nutrition(food_name,calories_per_100g,carbs_per_100g,protein_per_100g,fat_per_100g)
VALUES (?,?,?,?,?)"

        );

        ps.setString(1, name);

        ps.setDouble(2, cal);

        ps.setDouble(3, carb);

        ps.setDouble(4, prot);

        ps.setDouble(5, fat);

        ps.executeUpdate();



        JOptionPane.showMessageDialog(this, "Food Added Successfully!");



    } catch (Exception e) {

        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());

    }

}
```

4.6 User Interface (UI) Module

The User Interface (UI) is designed using **Java Swing**, with a modern card-based layout for better readability and user experience.

```
private void refreshTable() {

    List<Car> cars = dao.listCars();

    tableModel.setRowCount(0);
```

```
    for (Car c : cars) {
        tableModel.addRow(new Object[]{
            c.id, c.make, c.model, c.year, c.ratePerDay, c.available
        });
    }
}
```

4.6 Input Validation Logic

To ensure reliable data entry, the system validates inputs before updating the database.

```
private Car readInput() {
    String make = txtMake.getText().trim();
    String model = txtModel.getText().trim();
    int year = Integer.parseInt(txtYear.getText().trim());
    double rate = Double.parseDouble(txtRate.getText().trim());

    if (make.isEmpty() || model.isEmpty()) {
        throw new IllegalArgumentException("Fields cannot be empty.");
    }
    Car c = new Car();
    c.make = make;
    c.model = model;
    c.year = year;
    c.ratePerDay = rate;
    c.available = chkAvailable.isSelected();
```

```
    return c;

}
```

4.7 Summary

This chapter presented the key coding components of the Car Rental Management System. The implementation integrates multiple technologies such as **Java Swing**, **JDBC**, and **MySQL**, each contributing to the system's overall functionality. The chapter began by establishing the database connection using JDBC, followed by defining the Car data model, which represents the structure of each car record.

The **CarDAO** class was highlighted as the core component for performing CRUD operations on the database, ensuring secure and efficient data handling. The rental logic was then explained, demonstrating how the system manages availability by marking cars as rented or returned. The user interface implementation using Java Swing showcased how car data is displayed in a table and how user actions trigger backend processes. Lastly, input validation methods were described to ensure that only accurate and consistent data enters the system.

Overall, the coding structure emphasizes modularity, readability, and maintainability. Each component works together cohesively to deliver a functional, reliable, and user-friendly car rental management application.

CHAPTER 5

SCREENSHOTS

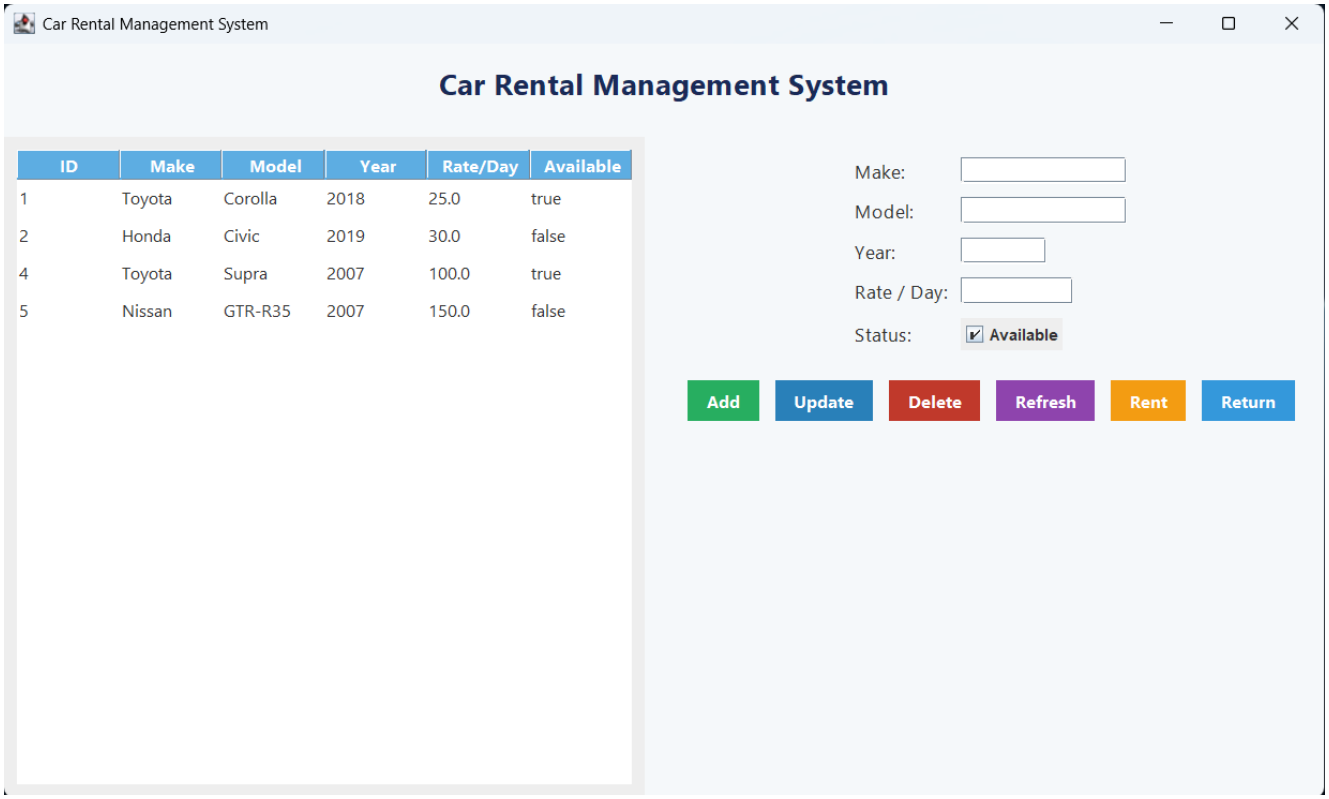

Fig.5.1 Car Rental Management UI

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The Car Rental Management System successfully demonstrates the integration of **Java Swing**, **JDBC**, and **MySQL** to build a functional, reliable, and user-friendly desktop application for managing car rental operations. The system provides an efficient solution for maintaining vehicle records, tracking availability, and performing rental transactions with accuracy and ease.

Through the implementation of structured data storage, CRUD operations, and real-time updates, the system eliminates the challenges associated with manual record-keeping such as data inconsistency, delays, and human errors. The application's modular design, consisting of separate components for database handling, core logic, and user interface, ensures maintainability and scalability for future development.

Overall, the system meets the primary objectives of simplifying rental operations, ensuring data reliability, and providing an intuitive interface for the administrator. It serves as a strong foundation for further expansion into a more advanced rental management platform.

6.2 Future Enhancements

Although the current system fulfills the basic requirements of a car rental agency, several improvements can enhance its functionality and usability. The proposed future enhancements are listed below:

1. Customer Management

Introduce a dedicated module to store customer details, driving license information, and rental history.

2. Rental Billing System

Automatically calculate rental duration, generate invoices, and support payment tracking.

3. Multi-User Authentication

Implement login systems with roles (Admin, Staff) for secure access and usage tracking.

4. Report Generation

Generate daily, weekly, or monthly reports summarizing:

- Car usage
- Revenue
- Rental trends
- Available vs. rented vehicles

5. Online / Cloud Integration

Allow customers to view availability and make bookings online using a web or mobile interface, powered by cloud database synchronization.

6. Vehicle Maintenance Tracking

Track service schedules, fuel logs, and damage reports for each vehicle.

7. Enhanced User Interface

Upgrade from Java Swing to **JavaFX** or a web-based UI for improved design, responsiveness, and user experience.

8. Notification Features

Send email or SMS notifications for:

- Due returns
- Booking confirmations
- Maintenance reminders

REFERENCES

1. Oracle Corporation, “Java Swing Tutorial”, https://docs.oracle.com/javase/tutorial/uiswing/
2. MySQL AB, “MySQL 8.0 Reference Manual”, https://dev.mysql.com/doc/refman/8.0/en/
3. FormDev Software, “FlatLaf Look and Feel for Swing”, https://www.formdev.com/flatlaf/
4. GeeksforGeeks, “JDBC – Java Database Connectivity”, https://www.geeksforgeeks.org/jdbc-java-database-connectivity/
5. TutorialsPoint, “Java Swing Components and Layouts”, https://www.tutorialspoint.com/java_swing/
6. W3Schools, “MySQL Tutorial”, https://www.w3schools.com/mysql/
7. Oracle University, “Java Database Programming Using JDBC”, https://www.oracle.com/java/technologies/javase/jdbc.html
8. Stack Overflow Discussions, “Common JDBC and Swing Implementation Queries”, https://stackoverflow.com/
9. Official MySQL Connector/J Documentation, https://dev.mysql.com/doc/connector-j/8.0/en/