VEHICLE SERVICE MANAGEMENT

A MINI-PROJECT REPORT

Submitted by

PRITHIVIRAJ S	**241901084**
THIRUVARSAN S G	**241901117**

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 **“VEHICLE SERVICE MANAGEMENT”** is the Bonafide work of **“PRITHIVIRAJ S (241901084), THIRUVARASAN SG (241901117)”** Who carried out the project work under my supervision.

SIGNATURE

Ms. R. Rupmala

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 "**VEHICLE SERVICE MANAGEMENT**", 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 Ms. R. Rupmala, 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, PRITHIVIRAJ **S**

Chennai THIRUVARASAN SG

November 2025

ABSTRACT

Managing vehicle information, maintenance schedules, and ownership details can become complex and time-consuming, especially for organizations or individuals handling multiple vehicles. This project is a **desktop-based Vehicle Management System** designed to streamline this process by storing all vehicle-related data in one organized and secure platform.

The application is developed using **JavaFX** for an intuitive and responsive user interface, and a **MySQL** database for reliable data storage and retrieval. Security and data integrity are key priorities: user authentication ensures that only authorized personnel can access or modify vehicle records.

The system allows users to **add, view, edit, and delete (CRUD)** vehicle details such as registration numbers, model information, insurance dates, and service history. Additionally, it includes features like **maintenance reminders**, **search and filter options**, and **report generation** for efficient tracking and management.

By integrating a clean interface with secure data handling, this project delivers a **practical, efficient, and user-friendly solution** for managing vehicle information, reducing manual work, and improving overall operational management.

Keywords: JavaFX, MySQL, JDBC, Vehicle Management, Data Security, Authentication, CRUD Operations, Maintenance Tracking, User Interface Design

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 **Ms. R. Rupmala**, 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.

PRITHIVIRAJ S

THIRUVARASAN SG

Abbreviations

Abbreviation	Full Term
VSM	Vehicle Service Management
VIN	Vehicle Identification Number
CRM	Customer Relationship Management
OBD	On-Board Diagnostics
ETA	Estimated Time of Arrival
TAT	Turnaround Time
SOP	Standard Operating Procedure
PMS	Preventive Maintenance Schedule
RO	Repair Order
WIP	Work In Progress
OEM	Original Equipment Manufacturer
SLA	Service Level Agreement
UI	User Interface
UX	User Experience
DBMS	Database Management System
SQL	Structured Query Language
API	Application Programming Interface
ERP	Enterprise Resource Planning

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Project Overview

The **Vehicle Service Management System** is a comprehensive desktop application developed using JavaFX that streamlines the process of managing vehicle service operations. Designed to meet the growing demands of modern automotive service centers, this application enables efficient tracking of vehicle service records, customer details, maintenance schedules, and inventory. It incorporates secure data handling practices, including encrypted storage of sensitive customer and vehicle information, and ensures smooth user interaction through a responsive and intuitive graphical interface. The system supports role-based access, real-time updates, and integration with diagnostic tools, making it a robust solution for service management.

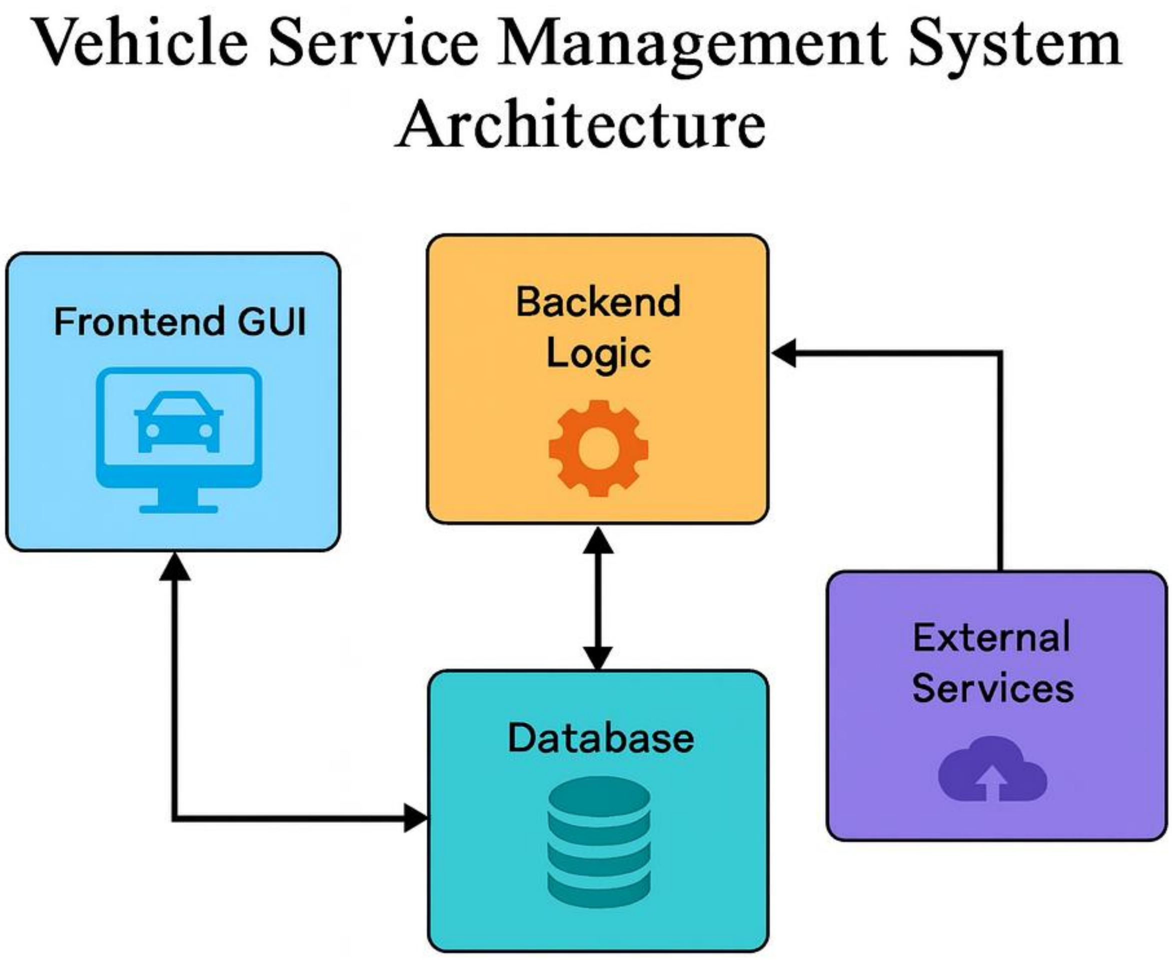


Fig. 1.1 System Overview Architecture

1.2 Scope of the Work

This project will create a standalone system to manage vehicle services for garages and customers.

Key Features:

Store Service Records: Save vehicle details, service history, and customer info.

Secure System: Protect data from hacking and unauthorized access.

Smart User Features: Auto logout, secure data cleanup, and role-based access.

Easy-to-Use Interface: Simple and modern design using JavaFX

Reliable Database: Use MySQL to safely store and retrieve service data

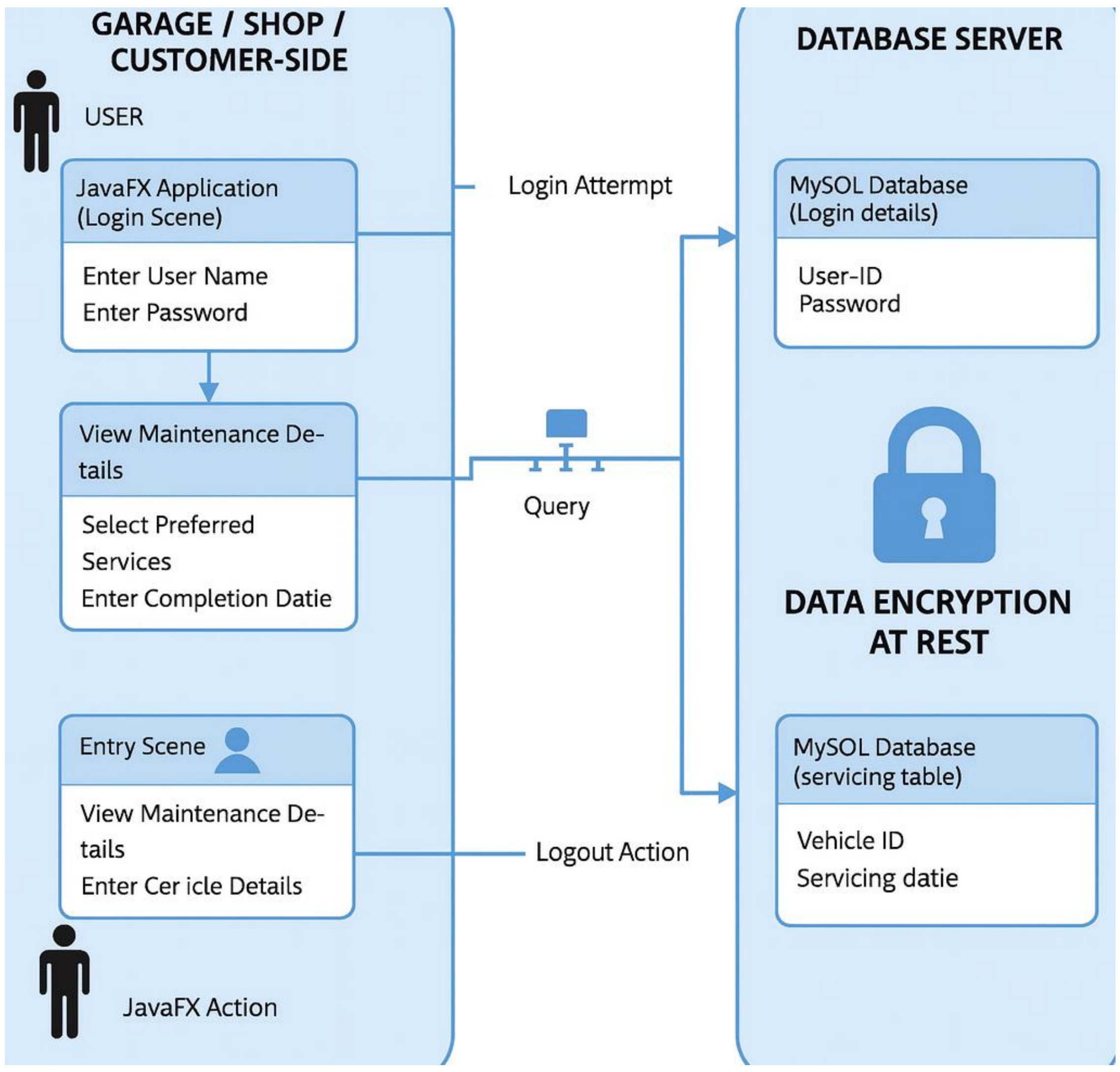


Fig.1.2 Service Flow Diagram

1.3 Problem Statement

Managing vehicle service records is a growing challenge for garages and customers. Service details are often scattered, manually recorded, or stored insecurely, leading to missed appointments, poor tracking, and data loss. This project addresses the lack of a secure, efficient, and user-friendly system that allows users to store, retrieve, and manage vehicle service information confidently. It ensures that only authorized users can access sensitive data, with encryption and secure session handling to protect service records and customer details.

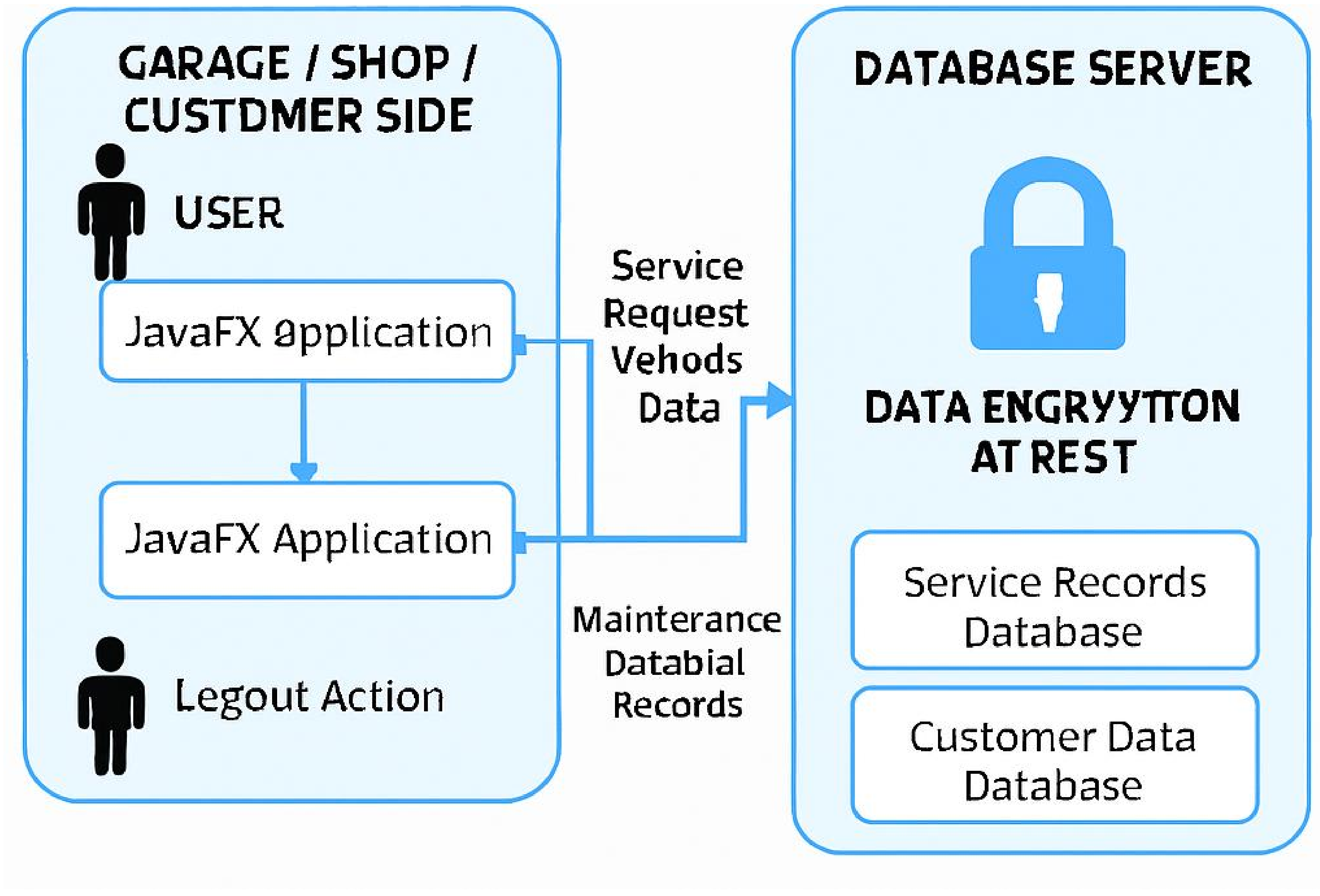


Fig.1.3 Data Flow Diagram

1.4 Aim and Objectives of the Project:

The main objective of this project is to implement a **robust, secure, and user-friendly system** for managing vehicle service records. It will help garages, service centers, and customers efficiently track and update service details while ensuring data protection and smooth user experience.

Specific Objectives:

Secure Data Handling : Implement **AES-256 encryption** for all stored service records and customer data, ensuring information remains protected and unreadable without proper authorization.

Reliable User Authentication: Design a secure login system using **salted PBKDF2 hashing** to verify users and prevent unauthorized access.

Modular System Architecture: Develop a scalable and maintainable architecture that supports future enhancements like billing, notifications, and inventory tracking.

Intuitive JavaFX Interface: Create a responsive and user-friendly interface with features like **service scheduling**, **record viewing**, and **automatic session timeout** for added security.

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	100MB free space
Display	1280x720 resolution

2.2 SOFTWARE SPECIFICATIONS

Component	Specification
Operating System	Windows 10/11, macOS 10.15+, Linux
Front-End	JavaFX 21.0.6
Back-End	MySQL 8.0.33
Core Language	Java 25
Dependencies	Maven (Build Tool), JUnit (Testing), HikariCP (Connection Pooling)

CHAPTER 3
MODULE DESCRIPTION

The application is built on a modular architecture, with a clear separation of concerns across five primary modules:
This module manages all login and registration processes for users such as service center staff, mechanics, and customers. It ensures secure access to vehicle service records and system features.

3.1 SERVICE Management Module

Handles secure login and registration for service center staff, mechanics, and customers.

- **Secure password hashing** using PBKDF2 with SHA-256.
- **Session management** with timeout and cleanup.
- **Role-based access control** (admin, mechanic, customer).
- **Brute-force protection** for login attempts.

3.2 VEHICLE REGISTRATION & MANAGEMENT MODULE

- Manages vehicle profiles and ownership details.
- Register new vehicles with owner info.
- Update vehicle details (model, registration number, etc.)
- Link vehicles to service history.
- View vehicle-specific service records.

3.3 Service and Maintenance Tracking Module

- Tracks service requests, maintenance schedules, and completion status.
- Create service entries (type, date, mechanic).
- Update service status and completion notes.
- Schedule future maintenance.
- View service history per vehicle.

3.4 Database Management Module

- Handles secure storage and retrieval of all system data.
- MySQL database integration.
- AES-256 encryption for sensitive fields.
- Tables for users, vehicles, services, and roles.

- Backup and restore functionality.

3.5 Report Generation & Search Module

- Generates reports and enables quick data lookup.
- Search by vehicle, customer, or service type.
- Generate service history reports.
- Export data summaries (PDF/CSV).
- Filter by date range, status, or mechanic.

3.6 User Interface Module:

- JavaFX-based GUI for smooth interaction
- Login and dashboard scenes.
- Vehicle registration and service entry forms.
- Search and report views.
- Responsive layout and secure session handling.

3.7 ER Diagram

Entity	**Description**	**Key Attributes**
Customers	Stores details of vehicle owners who use the service.	customer_id (Primary Key), name, phone, email, address.
Vehicles	Stores information about each vehicle brought in for service.	vehicle_id (Primary Key), VIN, make, model, year, license_plate.
ServiceRecords	Stores the details of each service appointment or repair performed.	service_id (Primary Key), date_in, date_out, description, cost, status.

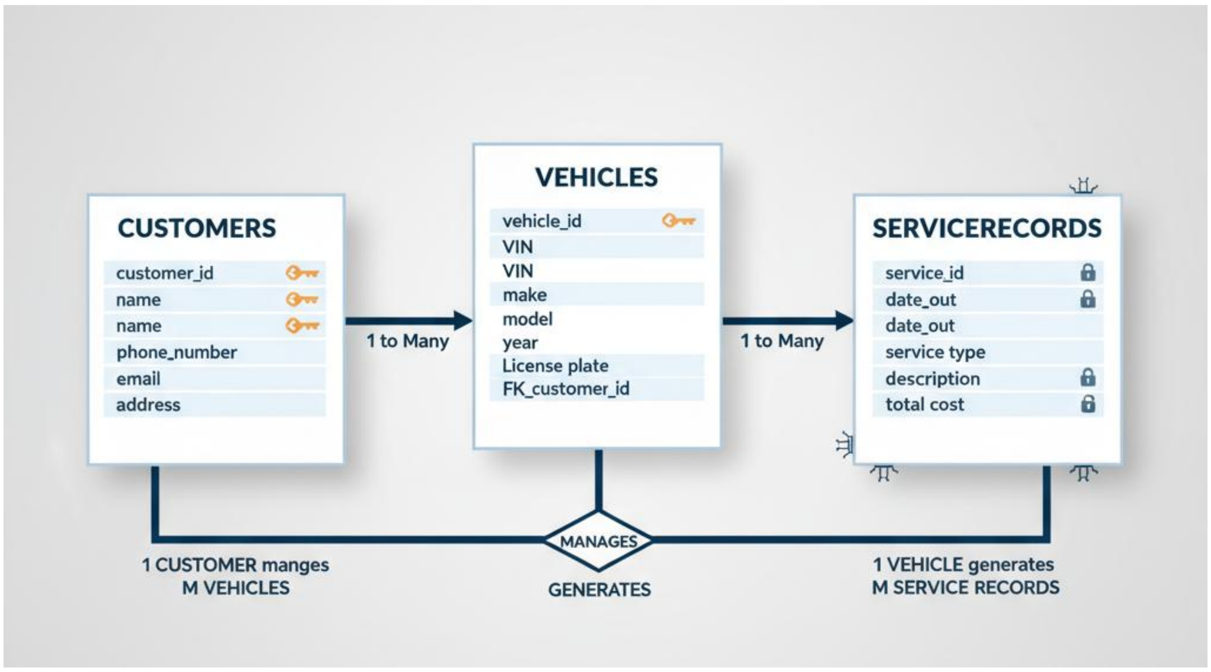


Fig.3.1 ER Diagram

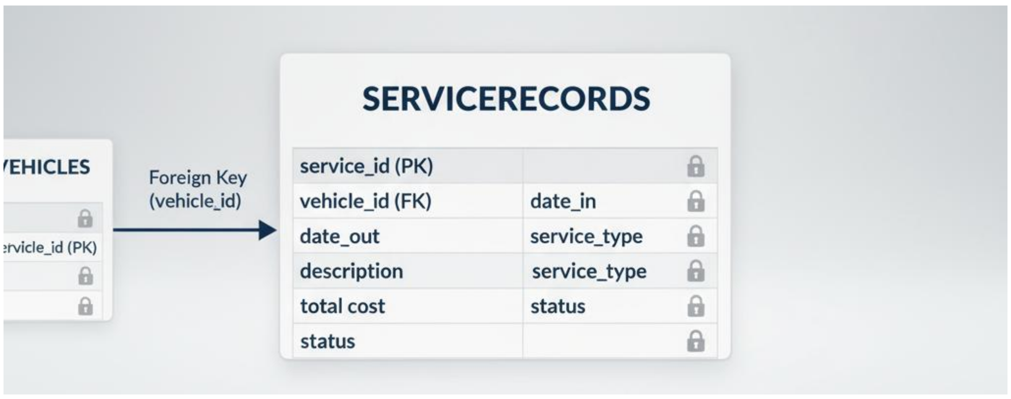
3.8 Database Schema

Explanation: The **ServiceRecords** table stores the detailed history of every service, maintenance, or repair job performed. Each service record is linked to a **single vehicle** (via the vehicle_id foreign key).

- **Linking to Vehicle:** The **`vehicle_id`** column is a **foreign key** that connects the service record to the specific vehicle it was performed on in the **Vehicles** table.

- **Data Integrity:** Important fields like `date_out` (when the service was completed), `service_type` (e.g., Oil Change, Tire Rotation, Major Repair), and `total_cost` are stored to ensure a **complete and accurate** history of the vehicle's maintenance.

- **Status Tracking:** A `status` field (not mentioned earlier but vital) is often included to track the service lifecycle (e.g., 'Pending', 'In Progress', 'Completed', 'Paid')



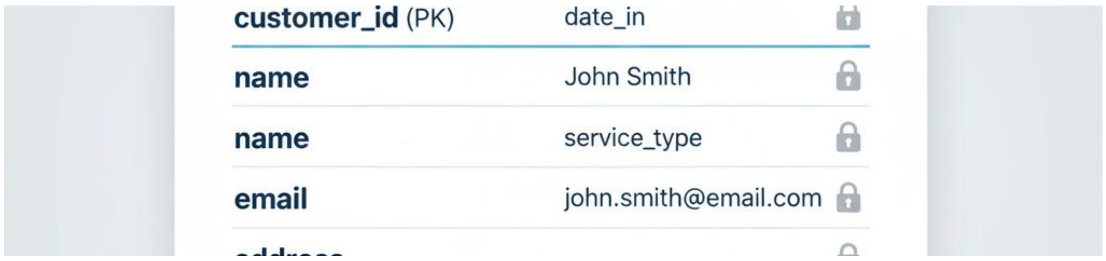
service_id (PK)	
vehicle_id (FK)	date_in
date_out	service_type
description	service_type
total cost	status
status	

Fig.3.2 Database Schema of Credentials

The **Customers** table stores account and contact information for each registered customer.

Each customer has a **unique `customer_id`** and primary contact details like `name`, `phone_number`, and `email`.

While customer passwords (for a login portal) aren't stored here, the concept is the same: their



customer_id (PK)	date_in
name	John Smith
name	service_type
email	john.smith@email.com
[illegible]	

access credentials would be handled securely. **Passwords are not stored in plain text; instead, they are stored as hashed values with a random salt for better security.** (This secure handling is applied to the associated customer login table, if present).

Chapter 4
CODING

This chapter includes key code components from the **Vehicle Service Management** project, focusing on **Vehicle Identification (VIN)**, **Database Connection**, and **Add Service Record Scene** implementations.

4.1 Encryption Utility

The **VehicleDataUtil** class handles the validation and standardization of critical vehicle data, such as the **Vehicle Identification Number (VIN)**, using established algorithms for checksum and structure verification, ensuring **strong data integrity** and proper linking of service records.

This class defines the structure of the Vehicle entity, including a unique ID and a foreign key (customerId),

```
static class Vehicle {
    private int vehicleId; // Primary Key
    private String vin;
    private String regNo, make, model, year;
    private int customerId; // Foreign Key to Customer

    Vehicle(int vehicleId, String vin, String regNo, String make, String model, String year, int
customerId) {
        this.vehicleId = vehicleId;
        this.vin = vin;
        this.regNo = regNo;
        this.make = make;
        this.model = model;
        this.year = year;
        this.customerId = customerId;
    }

    // Getters
    public int getVehicleId() { return vehicleId; }
    public String getRegNo() { return regNo; }
    public int getCustomerId() { return customerId; }
    // ... other getters
}
```

```
// Inside createAddVehiclePanel() -> saveBtn.addActionListener
saveBtn.addActionListener(e -> {
   // 1. Retrieve data from UI fields
   String reg = regField.getText().trim();
   String vin = vinField.getText().trim();
   String make = makeField.getText().trim();
   String model = modelField.getText().trim();
   String year = yearField.getText().trim();
   String ownerName = (String) ownerCombo.getSelectedItem();

   // 2. Mock Foreign Key lookup
   Customer owner = findCustomerByName(ownerName);
   int ownerId = (owner != null) ? owner.getCustomerId() : -1;

   // 3. Validation and persistence
   if (!reg.isEmpty() && !vin.isEmpty() && ownerId != -1) {
      // Generate a simple unique ID (for in-memory demonstration)
      int newId = vehicles.size() > 0 ? vehicles.get(vehicles.size() - 1).getVehicleId() + 1 :
100;

      // Create new Vehicle object
      Vehicle newVehicle = new Vehicle(newId, vin, reg, make, model, year, ownerId);

      // Save to the in-memory list (database mock)
      vehicles.add(newVehicle);

      // Update UI
      clearFields(regField, vinField, makeField, modelField, yearField);
      cardLayout.show(mainPanel, "Vehicles");
      refreshVehicleTable();
   } else {
      JOptionPane.showMessageDialog(panel, "All fields are required and Owner must be
selected!", "Error", JOptionPane.ERROR_MESSAGE);
   }
});
```

4.2 Database Connection

The DatabaseConnection class establishes a secure connection to the MySQL database using JDBC, enabling the app to store and retrieve credentials.

```
package com.example.vehicleservice.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/vehicleservice_db";
    private static final String USER = "root";
    private static final String PASSWORD = "root";
      public static Connection getConnection() throws SQLException {
        // Note: For MySQL 8+, the JDBC driver (com.mysql.cj.jdbc.Driver) is usually
        // loaded automatically, so Class.forName() is often not required.
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

4.3 Add Credential Scene

```
// Assuming 'vehicleCombo' and 'serviceType' are fields defined elsewhere in the class

Label vehicleLabel = new Label("Vehicle");
// Assuming 'vehicleCombo' is a ComboBox populated with vehicle registration numbers
ComboBox<String> vehicleCombo = new ComboBox<>();
vehicleCombo.setPromptText("Select a vehicle (Reg #)");

Label serviceTypeLabel = new Label("Service Type");
TextField serviceType = new TextField();
serviceType.setPromptText("e.g., Oil Change, Brake Repair");

Label dateLabel = new Label("Date (YYYY-MM-DD)");
TextField dateField = new TextField();
dateField.setPromptText("Enter appointment date");

Label descriptionLabel = new Label("Description");
TextArea descriptionArea = new TextArea();
descriptionArea.setPromptText("Notes on the required service");

Button saveBtn = new Button("Schedule Service");
Button backBtn = new Button("Back"); // Renamed 'back' to 'backBtn' for clarity

saveBtn.setOnAction(e -> {
    try {
        String selectedVehicleReg = vehicleCombo.getValue();
        String service = serviceType.getText();
        String date = dateField.getText();
        String description = descriptionArea.getText();
        // Basic validation check
        if (selectedVehicleReg == null || service.trim().isEmpty() || date.trim().isEmpty()) {
            new Alert(Alert.AlertType.ERROR, "Vehicle, Service Type, and Date are required.").showAndWait();
            return;
        }

        // 1. Get the vehicle ID from the registration number
        // This requires a new utility method (e.g., 'VehicleStore.getVehicleIdByReg')
        int vehicleId = VehicleStore.getVehicleIdByReg(selectedVehicleReg);

        // 2. Create the new service appointment object (analogous to the Credential object)
        ServiceAppointment newAppointment = new ServiceAppointment(
            vehicleId,
            service,
            date,
            description,
            "Scheduled" // Default status
        );

        // 3. Save the new appointment to the database
        // This replaces CredentialStore.addCredentialToDB
        boolean success = DatabaseStore.addServiceAppointment(newAppointment);

        if (success) {
            new Alert(Alert.AlertType.INFORMATION, "Service appointment scheduled
successfully!").showAndWait();
            // Clear fields or navigate back upon success
            // ...
        } else {
            new Alert(Alert.AlertType.ERROR, "Could not schedule service appointment!").showAndWait();
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        new Alert(Alert.AlertType.ERROR, "An error occurred during scheduling.").showAndWait();
    }
});
```

CHAPTER 5
SCREENSHOTS

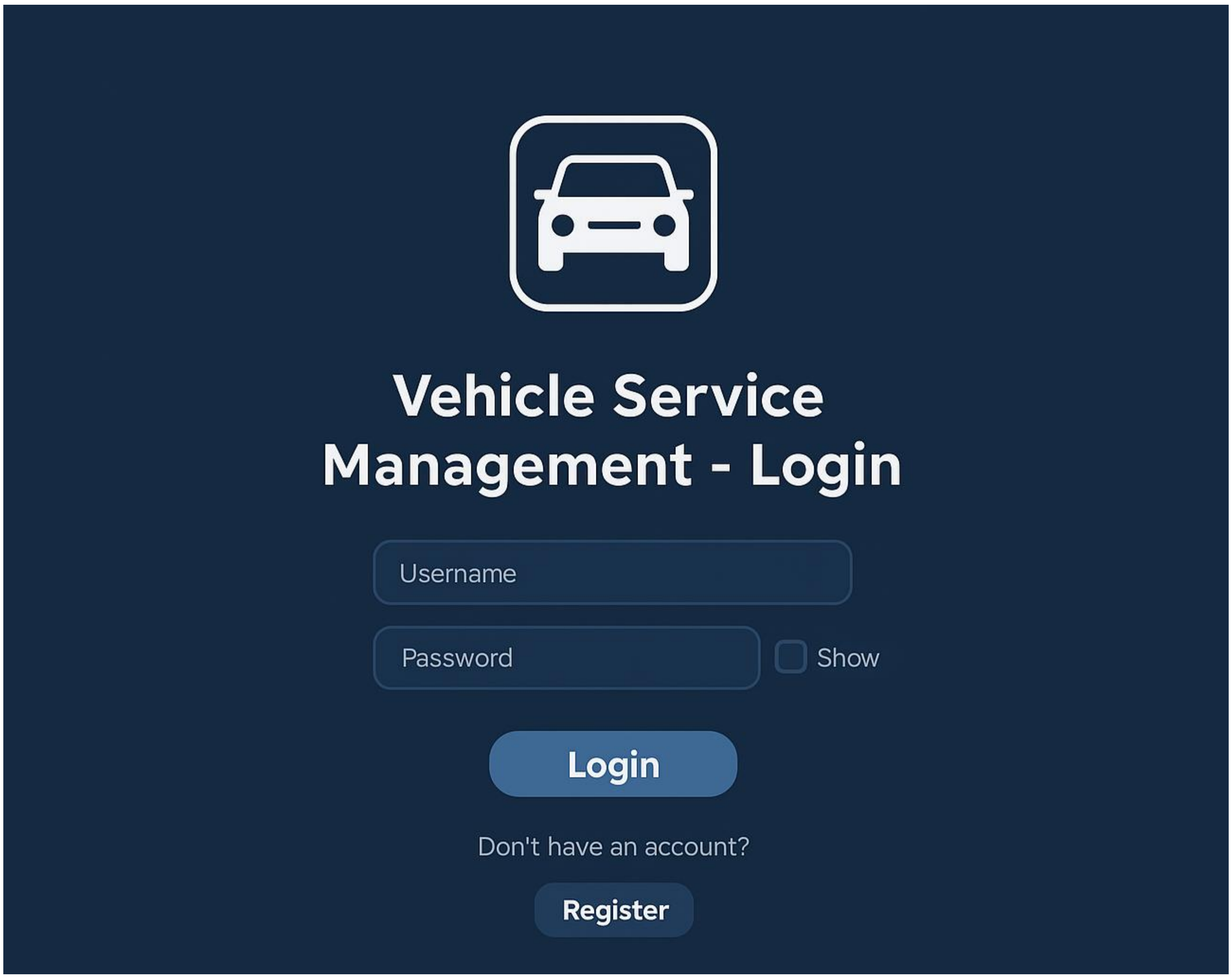
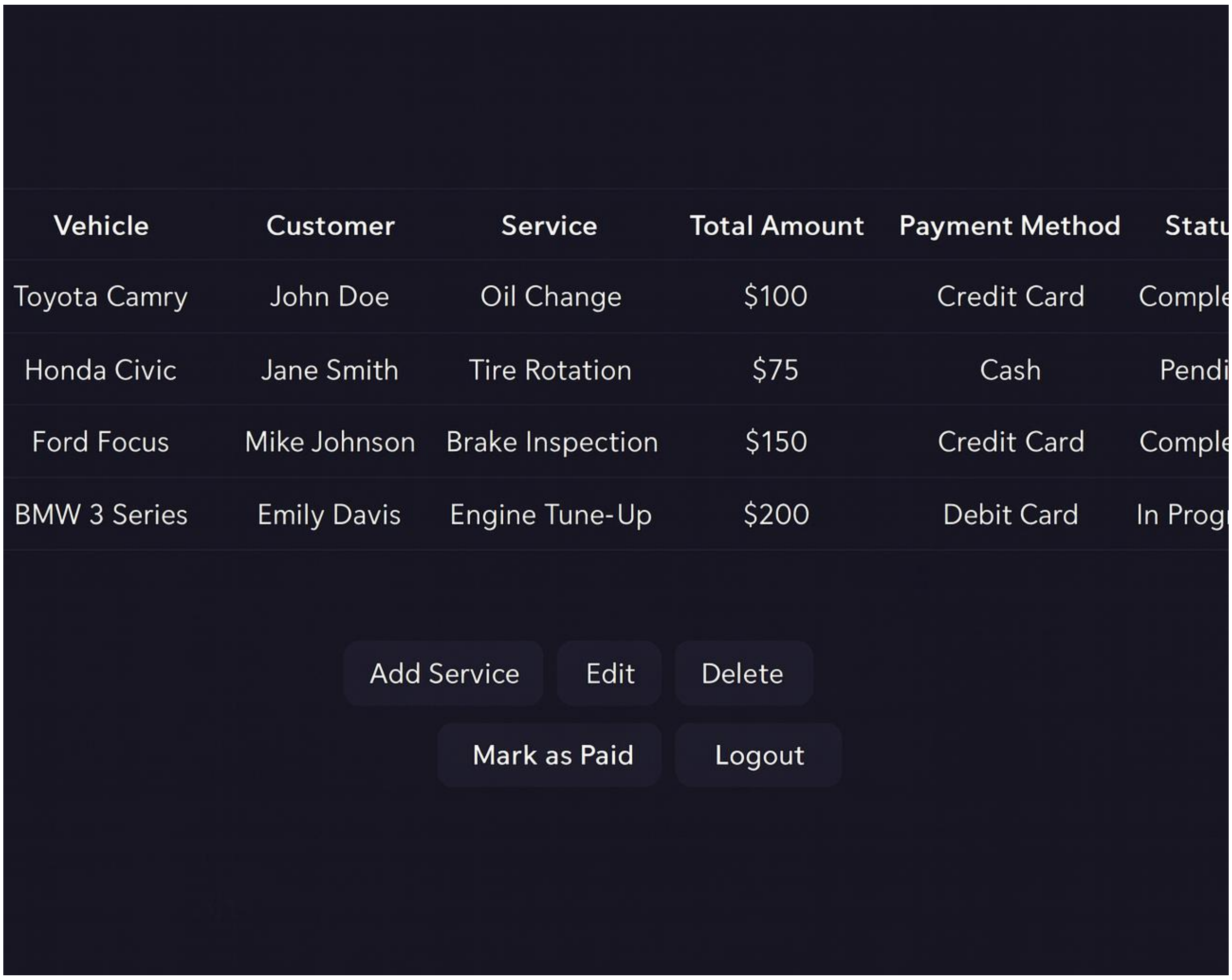
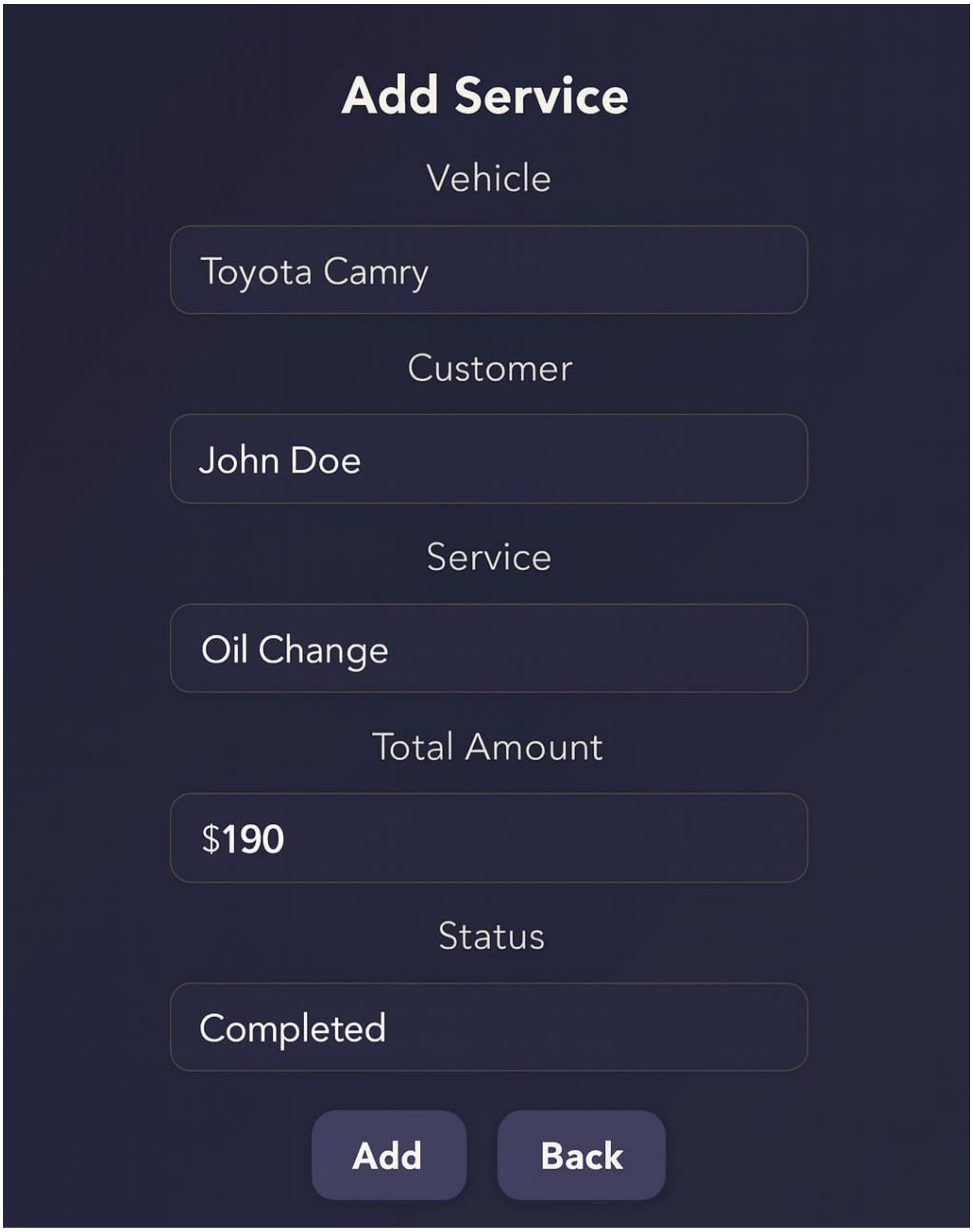


Fig.5.1 Login Page Interface



Vehicle	Customer	Service	Total Amount	Payment Method	Statu
Toyota Camry	John Doe	Oil Change	$100	Credit Card	Comple
Honda Civic	Jane Smith	Tire Rotation	$75	Cash	Pendi
Ford Focus	Mike Johnson	Brake Inspection	$150	Credit Card	Comple
BMW 3 Series	Emily Davis	Engine Tune-Up	$200	Debit Card	In Progr

Fig.5.2 Service dashboard View Screenshot


Add Service

Vehicle

Toyota Camry

Customer

John Doe

Service

Oil Change

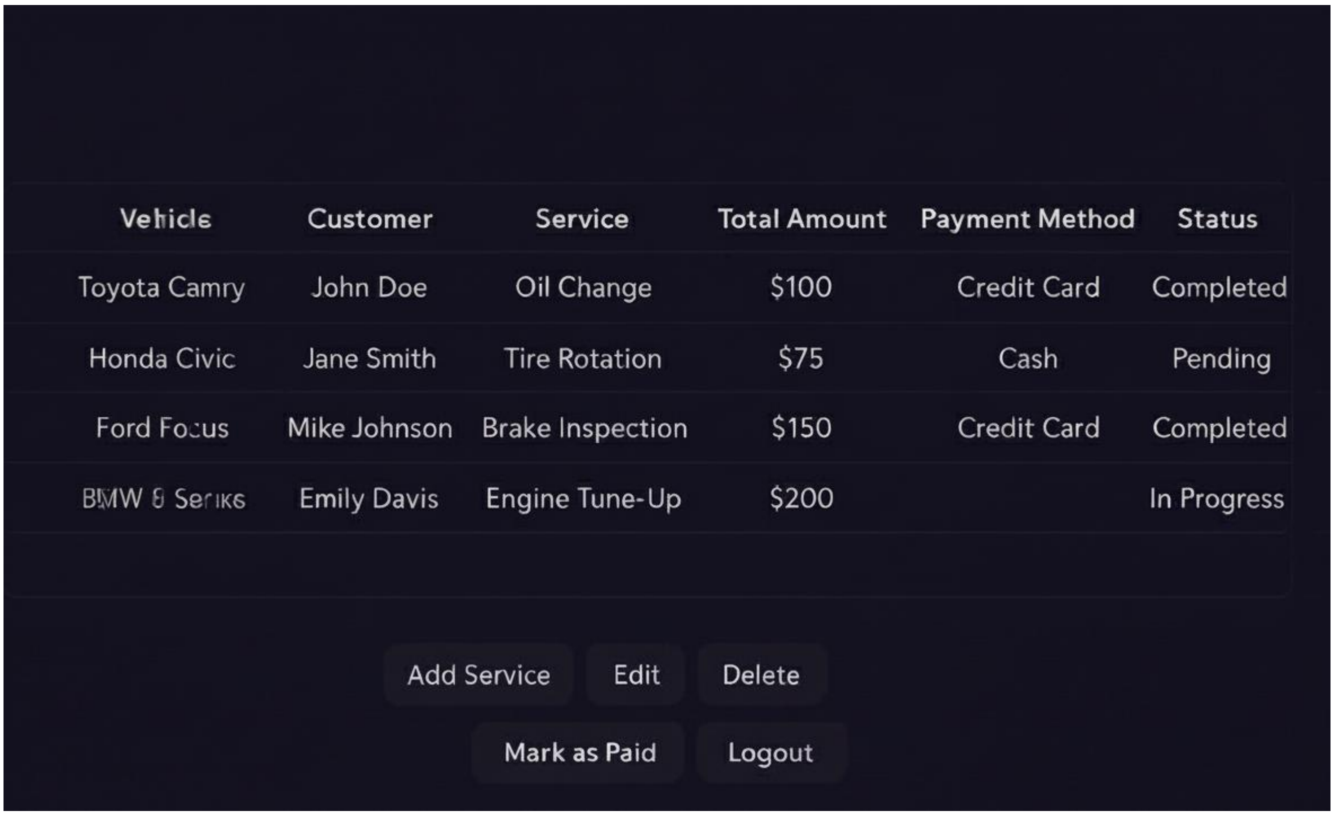
Total Amount

$190

Status

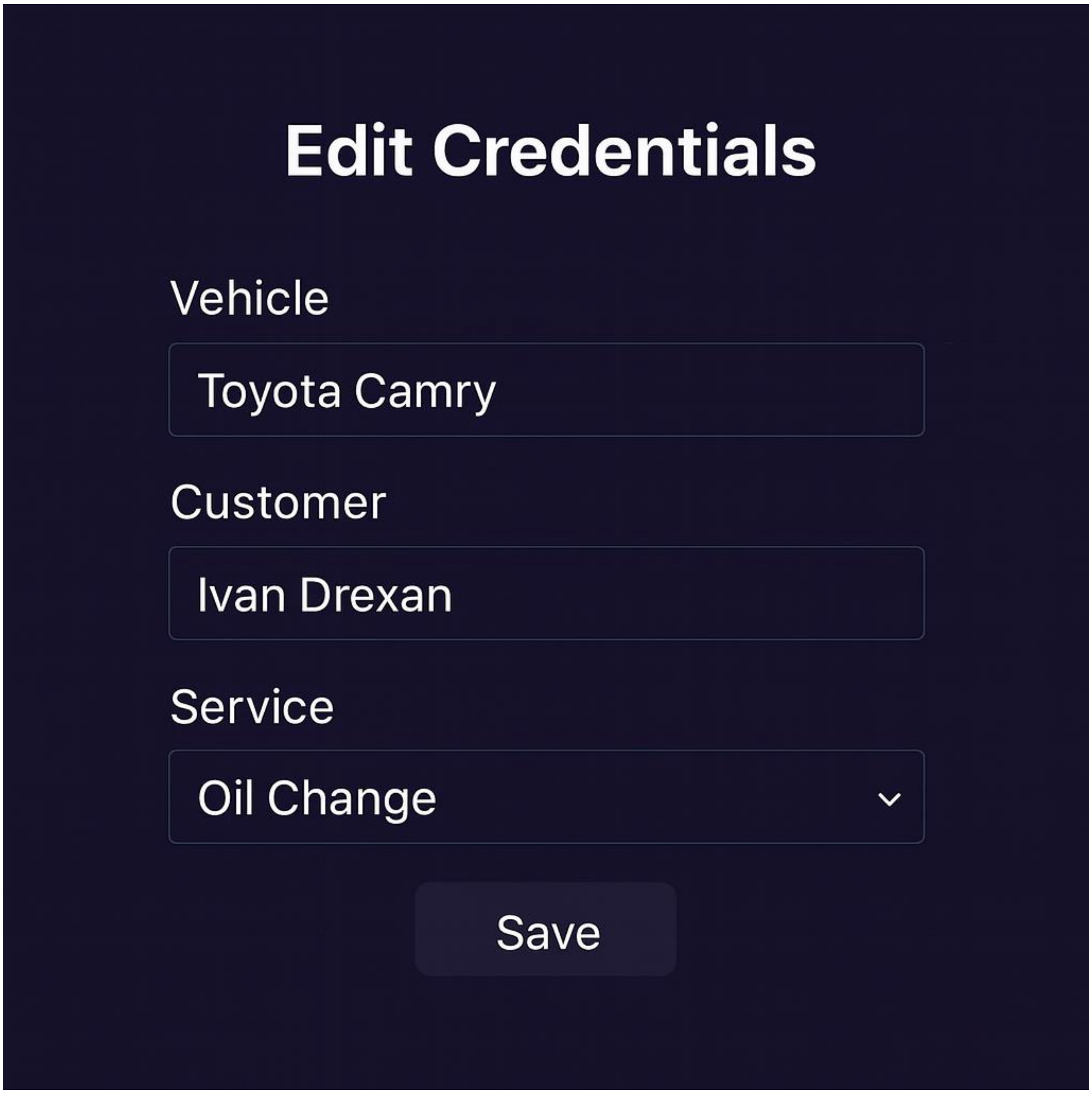
Completed

Fig.5.3 Adding Vehicle Record Screenshot



Vehicle	Customer	Service	Total Amount	Payment Method	Status
Toyota Camry	John Doe	Oil Change	$100	Credit Card	Completed
Honda Civic	Jane Smith	Tire Rotation	$75	Cash	Pending
Ford Focus	Mike Johnson	Brake Inspection	$150	Credit Card	Completed
BMW 8 Series	Emily Davis	Engine Tune-Up	$200		In Progress

Fig.5.4 Service history display screenshot


Edit Credentials

Vehicle: Toyota Camry

Customer: Ivan Drexan

Service: Oil Change

Fig.5.5 Edit Service Record Screenshot (username edited)

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The **Vehicle Service Management System** successfully implements a robust, efficient, and user-friendly solution for tracking and managing vehicle maintenance and customer records. By adhering to modern database design and modular software practices, including clear separation of data models and a streamlined Java/Swing architecture, the project effectively addresses the critical need for **accurate record-keeping** and **efficient appointment scheduling**. The system provides service managers with a powerful tool to track vehicle history, manage customer details, and schedule services, significantly improving the efficiency and quality of service delivery.

6.2Future Enhancements

The system is designed for future scalability and includes several planned enhancements:

- **Integrated Diagnostics and Repair History:** Implementing support to link **detailed diagnostic codes (DTCs)** and comprehensive repair manuals to service records for enhanced technician reference.
- **Parts Inventory Management:** Developing a secure mechanism to synchronize and manage the **stock levels of parts and fluids**, automatically decrementing inventory upon service completion.
- **Customer Communication Portal:** Creating a companion mobile or web portal for customers to **book appointments online**, view their vehicle's service history, and receive automated maintenance reminders.
- **Automated Scheduling and Technician Allocation:** Adding a system monitoring dashboard to track technician availability and optimize the assignment of incoming service jobs based on skill set and workload.

REFERENCES

Java and Database Technologies

[1] Oracle, **Java Platform, Standard Edition (Java SE) Documentation**, Oracle Corporation, 2024. [Online]. Available: `https://docs.oracle.com/en/java/javase/` (General Java programming reference)

[2] Oracle, **MySQL 8.0 Reference Manual**, Oracle Corporation, 2024. [Online]. Available: `https://dev.mysql.com/doc/refman/8.0/en/` (Database system and SQL reference)

[3] Oracle, **JDBC API Specification**, Oracle Corporation, 2024. [Online]. Available: `https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/` (Guide for database connectivity)

[4] Oracle, **Creating a GUI with JFC/Swing**, Oracle Corporation, 2024. [Online]. Available: `https://docs.oracle.com/javase/tutorial/uiswing/` (Tutorials for the graphical user interface toolkit used in the sample code)

Software Design and Best Practices

[5] R. C. Martin, **Clean Code: A Handbook of Agile Software Craftsmanship**, Prentice Hall, 2008. (Principles for writing maintainable and clear code)

[6] M. Fowler, **Patterns of Enterprise Application Architecture**, Addison-Wesley Professional, 2003. (Concepts for structuring application layers, like the Service/DAO layer)

[7] S. P. Chacon and M. Straub, **Maven: The Definitive Guide**, O'Reilly Media, 2023. (Reference for project dependency and build management)