Car Repair and Maintenance App

Software Design Document

Prepared by:
Ahmad Mustafa, Muhammad Munawar Khan

Institution:
Namal University Mianwali

Date:
May 19, 2025

A comprehensive design document for a user-friendly desktop application to manage vehicle maintenance and tracking.

Prepared for Namal University Mianwali, Software Engineering Department

Contents

1 Introduction **2**
- 1.1 Purpose 2
- 1.2 Scope 2
- 1.3 Overview 2
- 1.4 Reference Material 2
- 1.5 Definitions and Acronyms 2

2 System Overview **2**

3 System Architecture **3**
- 3.1 Architectural Design 3
- 3.2 Decomposition Description 4

4 Data Design **5**
- 4.1 Data Description 5
- 4.2 Data Dictionary 6

5 Component Design **6**
- Vehicle Management 6
- Reminder Manager 6

6 Human Interface Design **7**
- 6.1 Overview of User Interface 7
- 6.2 Dashboard Layout 7
- 6.3 Screen Images 8
- 6.4 Screen Objects and Actions 8
- 6.5 User Experience Notes 8

7 Requirements Matrix **8**

Appendix A: Sequence Diagrams **9**
- A.1 Admin – Manage Users and View Logs 9
- A.2 User – Add Vehicle, Log Health, Set Reminder 10

Appendix B: Core Use Case Sequence Diagrams **12**
- B.1 Add Vehicle Use Case 12
- B.2 Log Health Status Use Case 13
- B.3 Set Maintenance Reminder Use Case 14
- B.4 Track Cost Use Case 14
- B.5 Generate Maintenance Alert Use Case 15

Appendix C: Entity-Relationship Diagram **15**

1 Introduction

1.1 Purpose

This Software Design Document (SDD) describes the architecture and system design of the Car Repair and Maintenance App. It serves as a reference for developers, testers, and stakeholders to understand how the software is structured to meet the requirements specified in the SRS.

1.2 Scope

The Car Repair and Maintenance App is a standalone, user-friendly desktop application that allows users to manage information related to their vehicles. Features include adding and tracking vehicles, logging odometer readings, setting reminders for maintenance, monitoring health status, and calculating usage costs. The application stores all data locally and supports offline usage.

1.3 Overview

This document includes system overview, architecture, data structures, user interface design, and a traceability matrix mapping the system components to functional requirements.

1.4 Reference Material

- Software Requirements Specification – Car Repair and Maintenance App
- IEEE Std 1016-2009 – Recommended Practice for Software Design Descriptions

1.5 Definitions and Acronyms

- UI – User Interface
- VIN – Vehicle Identification Number
- CRUD – Create, Read, Update, Delete
- DFD – Data Flow Diagram

2 System Overview

The system is designed as a user-friendly desktop application. It allows users to manage multiple cars and their maintenance schedules. The app helps in tracking odometer readings, health status, fuel and service costs, and generating alerts when maintenance is due. All records are stored locally using lightweight storage solutions (such as embedded databases or structured files).

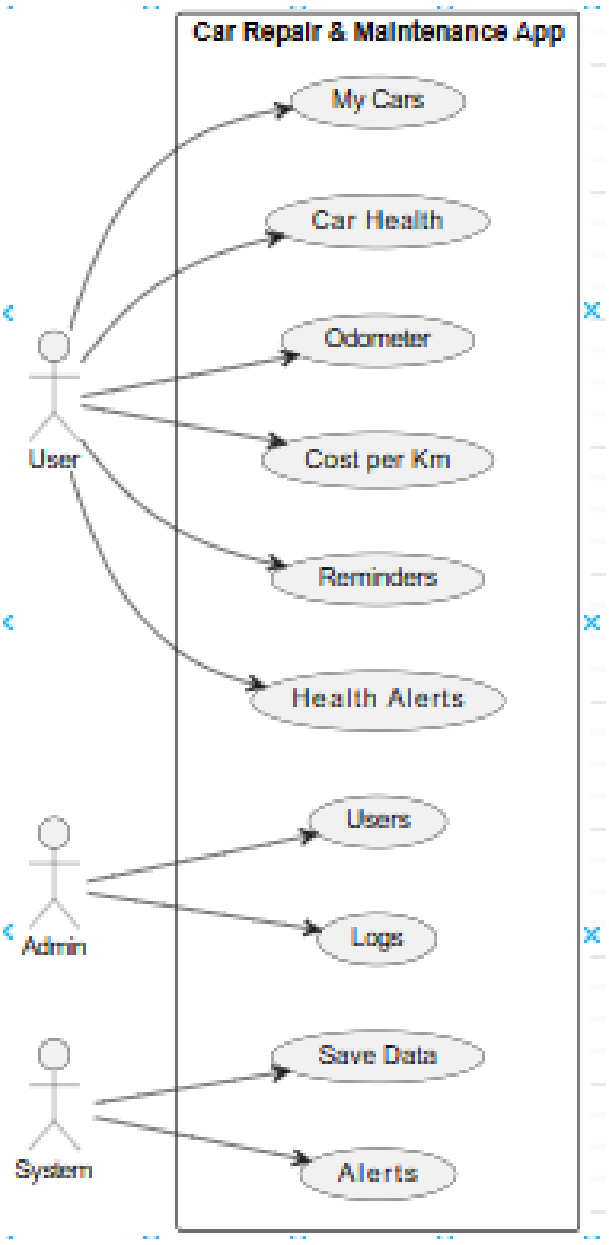

Figure 1: Use Case Diagram (Placeholder)

3 System Architecture

3.1 Architectural Design

The software architecture follows a modular pattern, separating the core logic, data handling, and interface presentation. The main components include:

- **Main Application Module**
- **Vehicle Management Module**
- **Maintenance Reminder Module**
- **Health Tracking Module**
- **Cost Tracking Module**

- **Admin Module**
- **Data Storage Module**

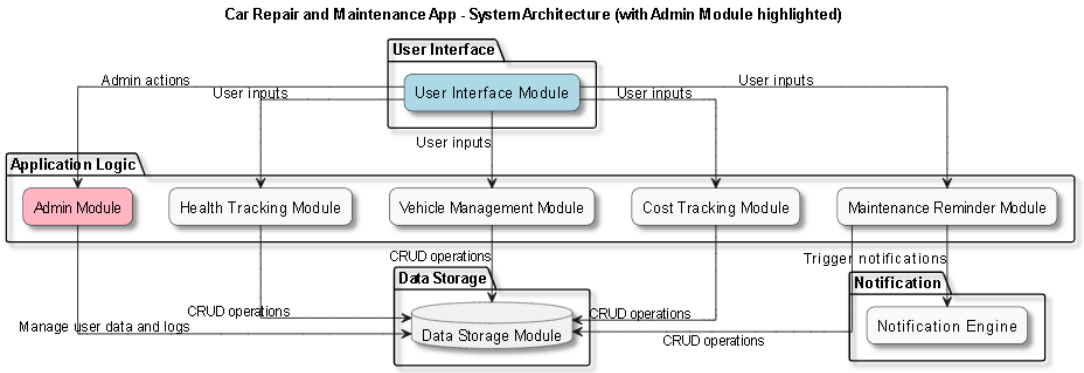

Figure 2: System Architecture Diagram (Placeholder)

3.2 Decomposition Description

Each module is decomposed further into functionalities:

- Vehicle Manager: Add, view, edit, and delete car information.
- Reminder Manager: Add new maintenance reminders.
- Health Logger: Input and validate vehicle health.
- Odometer Logger: Record mileage.
- Cost Logger: Track expenses and usage.

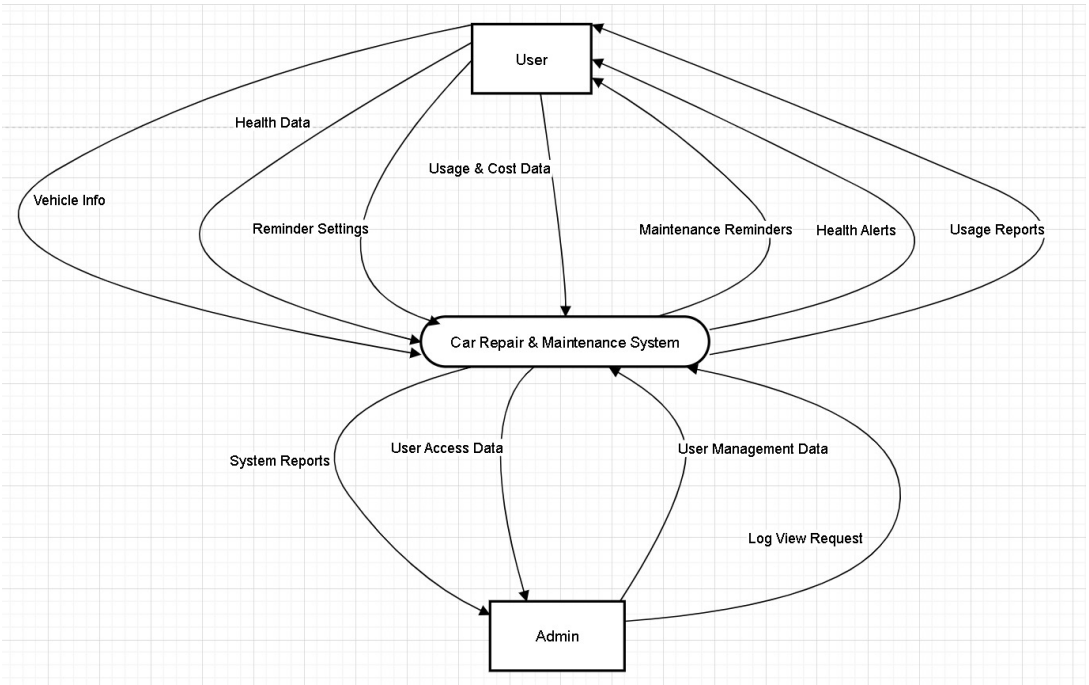

Figure 3: Data Flow Diagram Level 0 (Placeholder)

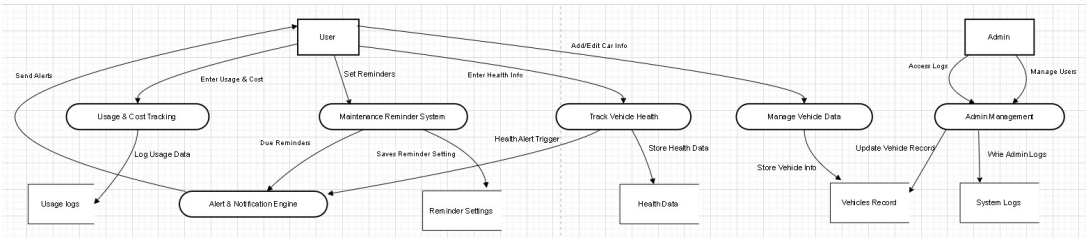

Figure 4: Data Flow Diagram Level 1 (Placeholder)

Class Diagram

The following UML class diagram presents the structural organization of the Car Repair and Maintenance App. It includes classes like User, Admin, Vehicle, Reminder, HealthStatus, and others that represent core modules and responsibilities of the system.

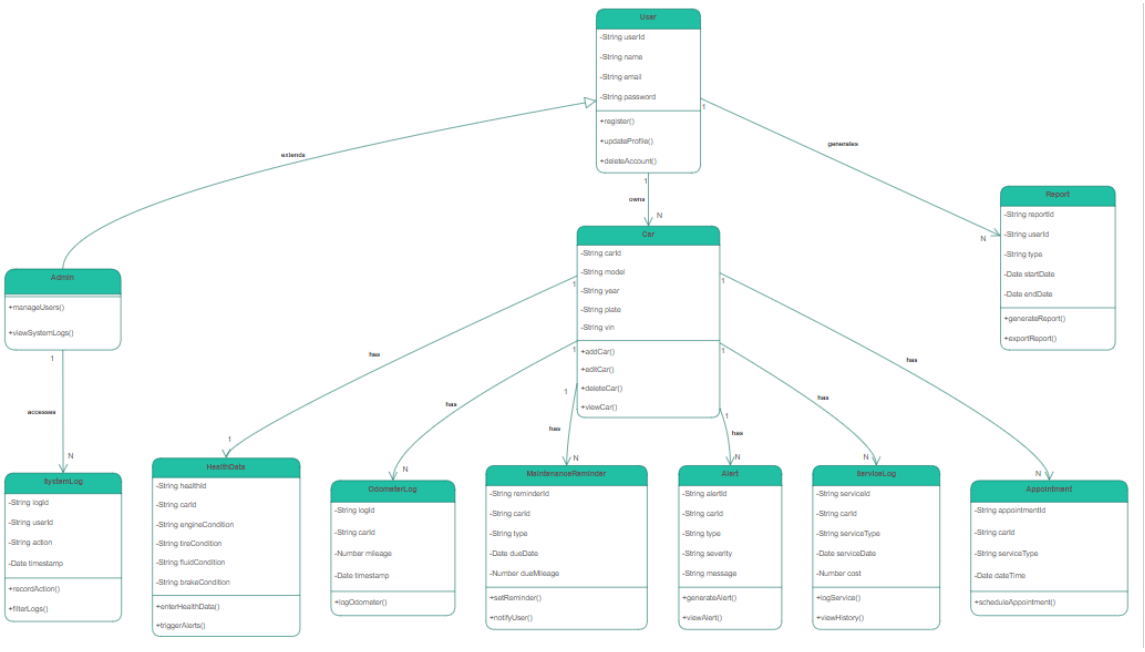

Figure 5: UML Class Diagram – Car Repair and Maintenance App

4 Data Design

4.1 Data Description

The Car Repair and Maintenance App handles a variety of structured data to support its core functionalities. Data is primarily stored locally using an embedded relational database (e.g., SQLite), which offers lightweight, reliable storage without requiring an internet connection. All data is organized in normalized tables to ensure consistency and reduce redundancy.

The system stores the following key categories of data:

- **User Data:** Includes credentials, roles (admin or regular user), and login history. User authentication ensures data security and role-based access.
- **Vehicle Information:** Each vehicle is associated with a user and contains essential

attributes such as VIN, make, model, year, engine capacity, and registration date. Vehicles can have multiple logs associated with them.

- **Odometer Readings:** Periodic entries of mileage with timestamps are stored. This data is used to calculate usage trends, detect service intervals, and generate reminders.
- **Health Status Logs:** This includes condition ratings or statuses for key vehicle components like engine, tires, fluids, and brakes. Each entry is time-stamped and linked to a specific vehicle.
- **Maintenance Reminders:** Stored with parameters such as reminder type, due date, and mileage threshold. The system checks these regularly to trigger visual alerts.
- **Cost Logs:** Records of all vehicle-related expenses such as fuel, repairs, and services. Each record includes date, type of cost, amount, and associated vehicle.
- **System Logs:** Used to audit actions performed by users, such as login, data entry, and updates. Logs include timestamp, user ID, and action description.
- **Alert History:** Archive of past reminders and alerts shown to users, aiding in historical analysis and report generation.

4.2 Data Dictionary

Entity	Type	Description
User	Object	Username, password, role
Admin	Inherits User	Elevated privileges to manage accounts
Vehicle	Object	VIN, model, year, owner ID
HealthStatus	Object	Engine, tires, fluids, brakes
Reminder	Object	Type, due date, mileage
OdometerLog	Object	Mileage with timestamp
CostLog	Object	Fuel/service costs

Table 1: Data Dictionary for Car Repair and Maintenance App

5 Component Design

Vehicle Management

Pseudocode:

```
Function add_vehicle():
    if validate_input():
        save_vehicle_to_database()
```

Reminder Manager

```
Function check_reminders():
    for each reminder in database:
```

```
if due_date_passed() or mileage_threshold_reached():
    notify_user()
```

6 Human Interface Design

6.1 Overview of User Interface

The dashboard of the Car Repair and Maintenance App is designed for clarity and efficiency. It provides an integrated view of essential car data, ensuring users can quickly monitor their vehicle's health, odometer status, and receive critical alerts.

6.2 Dashboard Layout

The main dashboard includes the following components:

- **Sidebar Navigation:** A vertical menu on the left provides access to the following sections:
  - My Cars
  - Health Tracking
  - Odometer
  - Cost Tracking
  - Alert System
- **Car Panel:** Displays the selected vehicle with image, model, and a "View Details" button.
- **Health Tracking Indicators:** Horizontal progress bars show the current status of:
  - Engine
  - Tires
  - Fluids
  - Brakes
- **Odometer Display:** Large digital-style display showing current mileage (e.g., `52,360 mi`) and a "Log Entry" button for updating.
- **Alert System:** A visual warning area indicating upcoming maintenance needs (e.g., "Maintenance due soon") with buttons to "View Alerts" or "View Report".

6.3 Screen Images

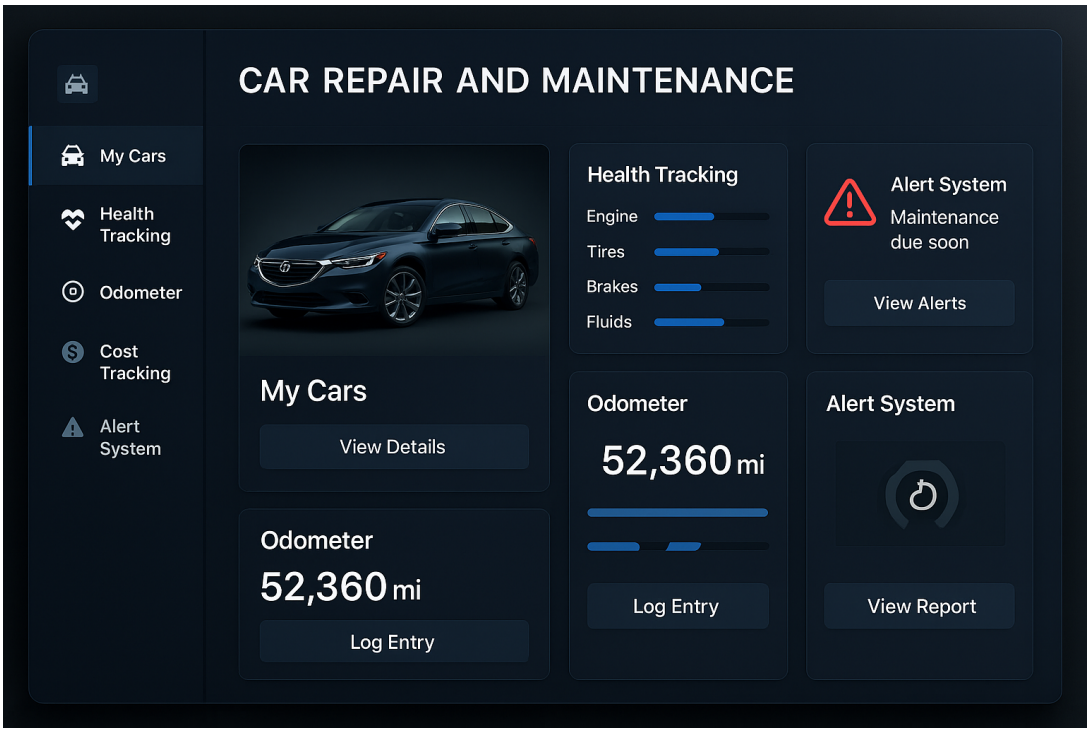

Figure 6: Dashboard Screen

6.4 Screen Objects and Actions

- **Navigation Panel Items:** Direct access to key modules (e.g., My Cars, Alerts).
- **Car Display Panel:** Shows current car image and stats.
- **Health Bars:** Visual indicators of component condition.
- **Mileage Display:** Shows mileage and logs new readings.
- **Alert Notification Panel:** Displays maintenance warnings and links to detailed reports.

6.5 User Experience Notes

- Intuitive layout with clearly defined panels and labels.
- A dark theme improves readability and reduces eye strain.
- Button-based actions are supported with minimal steps.
- Icons enhance quick recognition of module functions.

7 Requirements Matrix

Requirement ID	Short Description	Component(s)

REQ-1	Add new vehicle with details like make, model, year, and VIN.	Vehicle Management
REQ-2	Edit or delete existing vehicle information.	Vehicle Management
REQ-3	Input and update engine, tires, fluids, and brakes condition.	Health Tracking Module
REQ-4	Display vehicle health with visual indicators (progress bars).	Health Tracking Module, UI
REQ-5	Log odometer readings with timestamps.	Odometer Logger
REQ-6	Display the latest mileage on the dashboard.	Odometer Logger, Dashboard UI
REQ-7	Record fuel and service expenses.	Cost Logger
REQ-8	Calculate total costs and monthly summaries.	Cost Logger, Dashboard UI
REQ-9	Set maintenance reminders by date or mileage.	Reminder Manager
REQ-10	Show visual alerts for due or missed maintenance.	Alert System, Dashboard UI
REQ-11	View alerts and generate maintenance reports.	Alert System
REQ-12	Admin can manage user accounts and credentials.	Admin Module
REQ-13	Admin can view activity logs for audit purposes.	Admin Module, Logging System
REQ-14	Record all user activities in logs.	Activity Logging System
REQ-15	Log entries must include timestamp, action, and user ID.	Activity Logging System
REQ-16	Store all data locally in structured format.	Data Storage Module
REQ-17	Ensure data persistence after application is closed.	Data Storage Module

Table 2: Requirements Traceability Matrix

Appendix A: Sequence Diagrams

A.1 Admin – Manage Users and View Logs

The following sequence diagram illustrates how the Admin interacts with the system to manage user accounts and view activity logs.

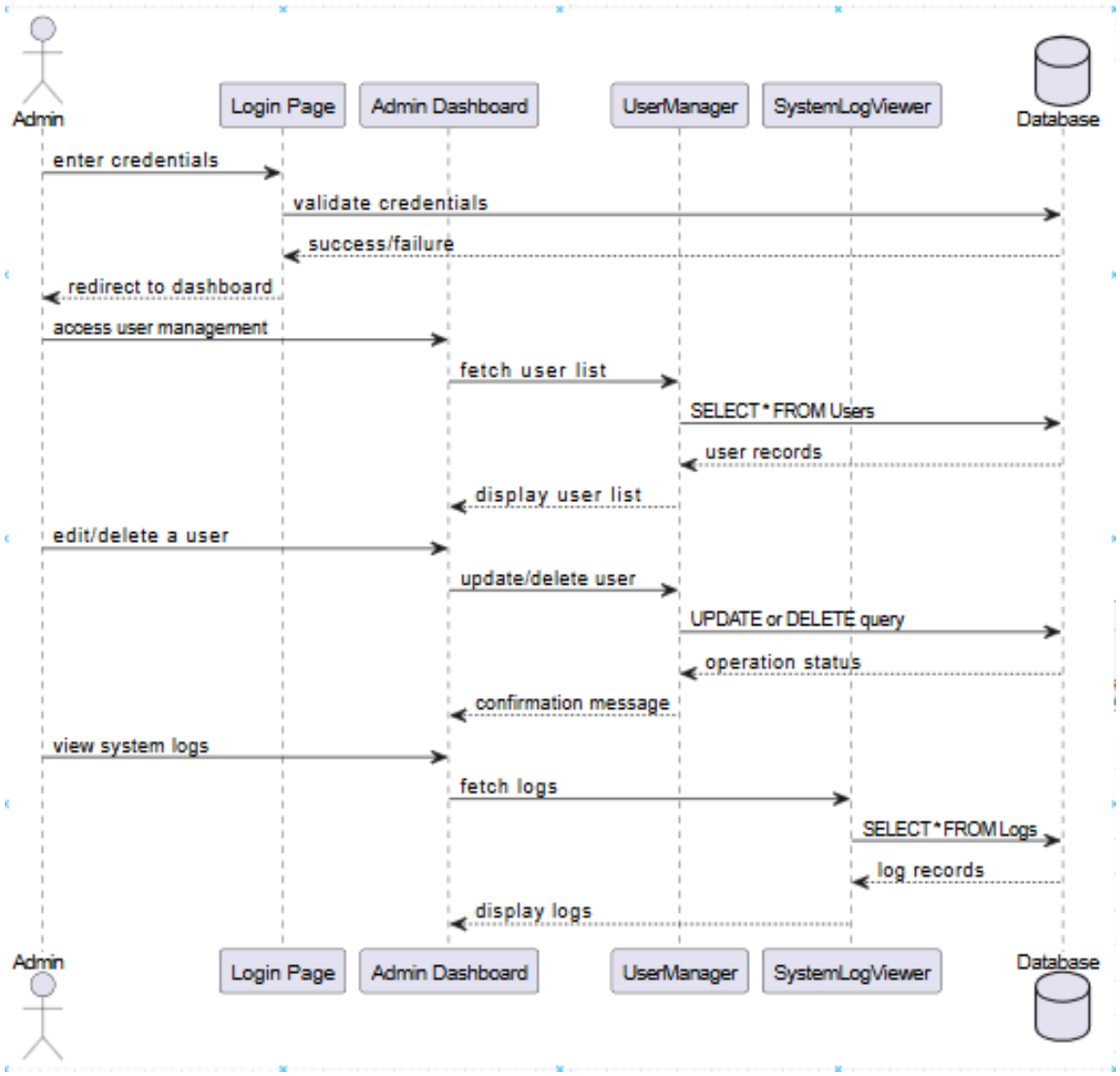

Figure 7: Admin User Management and Log Viewing

A.2 User – Add Vehicle, Log Health, Set Reminder

The following sequence diagram illustrates how a user interacts with the system to manage vehicle health and maintenance tracking from login to dashboard interaction.

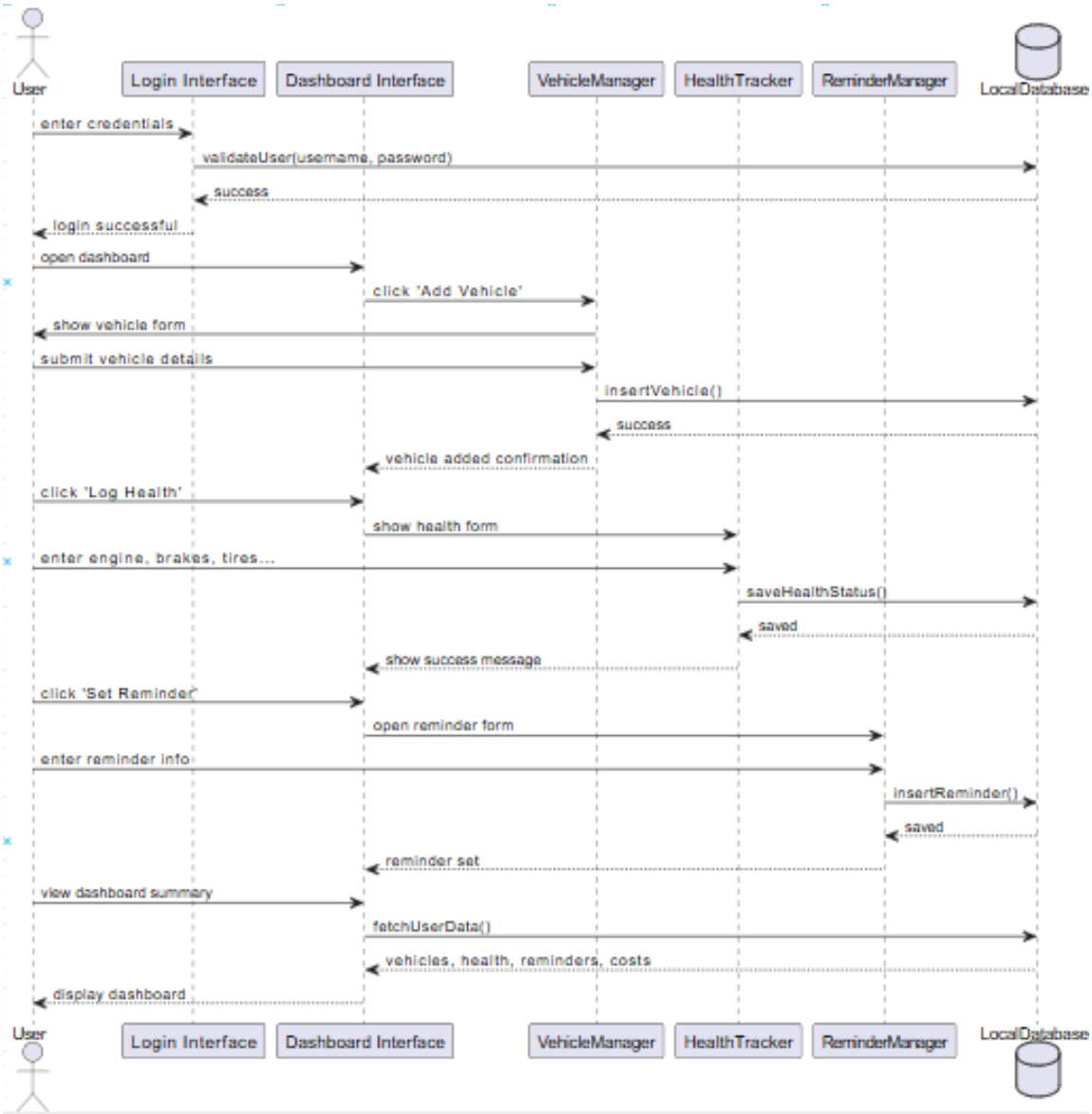

Figure 8: User Workflow

Appendix B: Core Use Case Sequence Diagrams

B.1 Add Vehicle Use Case

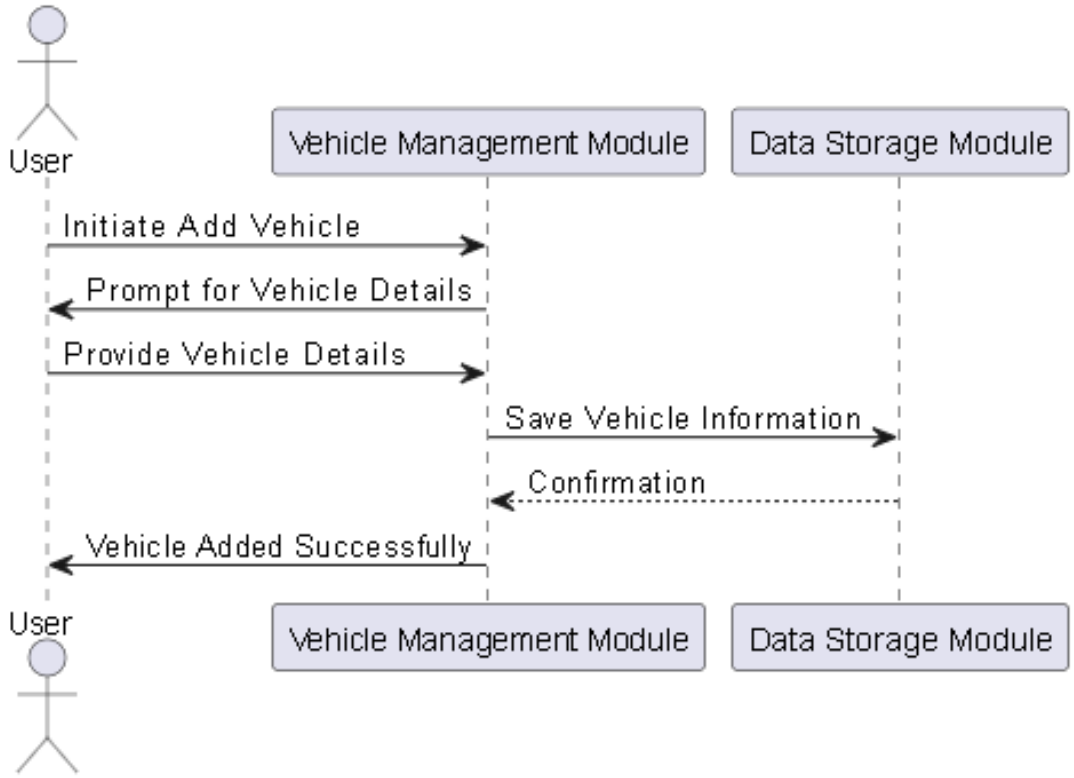

Figure 9: Add Vehicle

B.2 Log Health Status Use Case

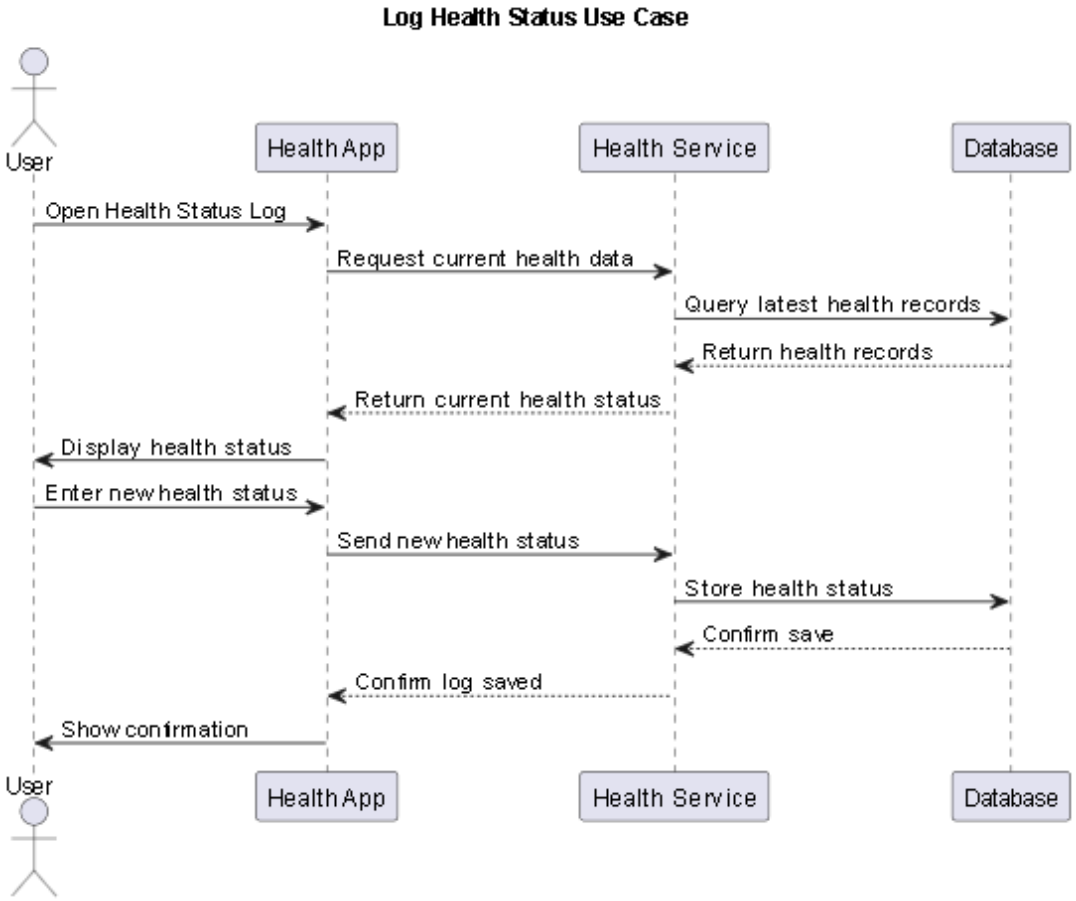

Figure 10: Log Health Status

B.3 Set Maintenance Reminder Use Case

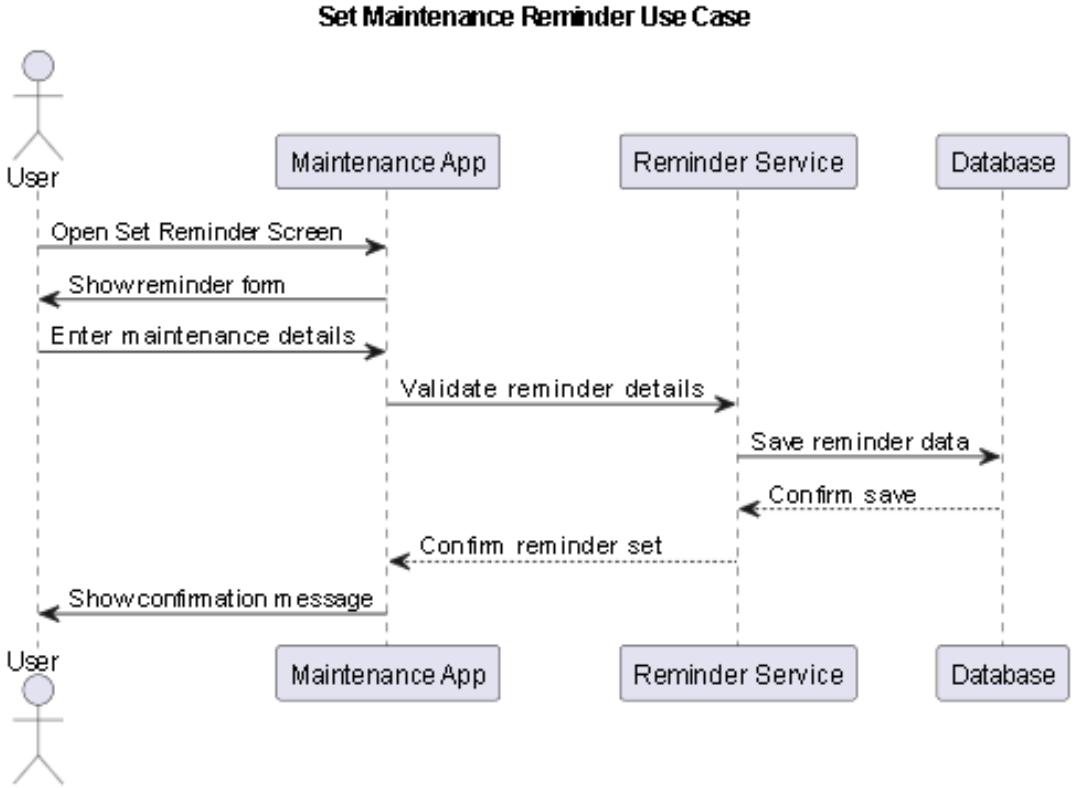

Figure 11: Set Maintenance Reminder

B.4 Track Cost Use Case

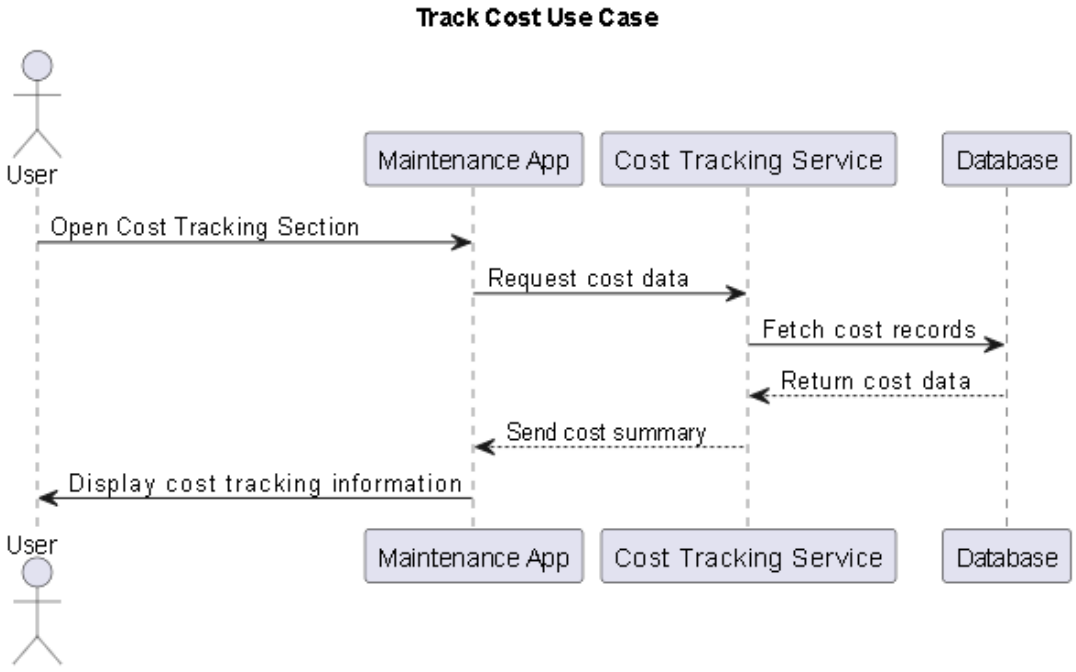

Figure 12: Track Cost

B.5 Generate Maintenance Alert Use Case

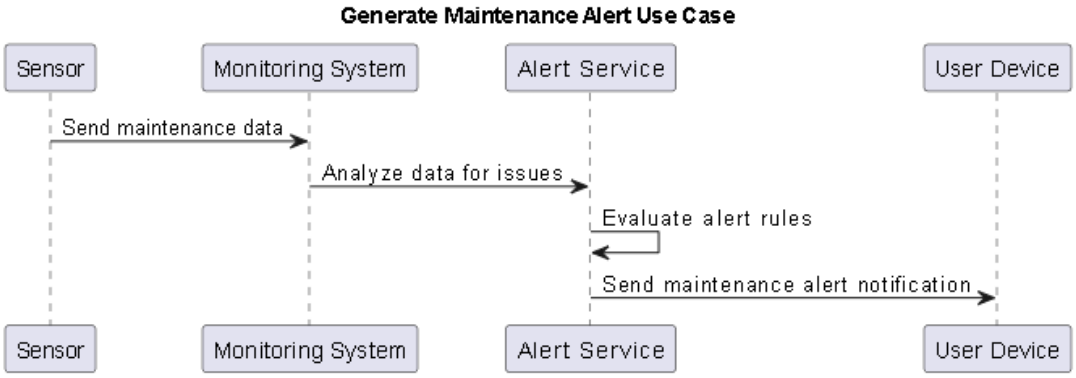

Figure 13: Generate Maintenance Alert

Appendix C: Entity-Relationship Diagram

The following diagram represents the Entity-Relationship model of the Car Repair and Maintenance App, detailing entities like User, Vehicle, Reminder, CostLog, and their relationships.

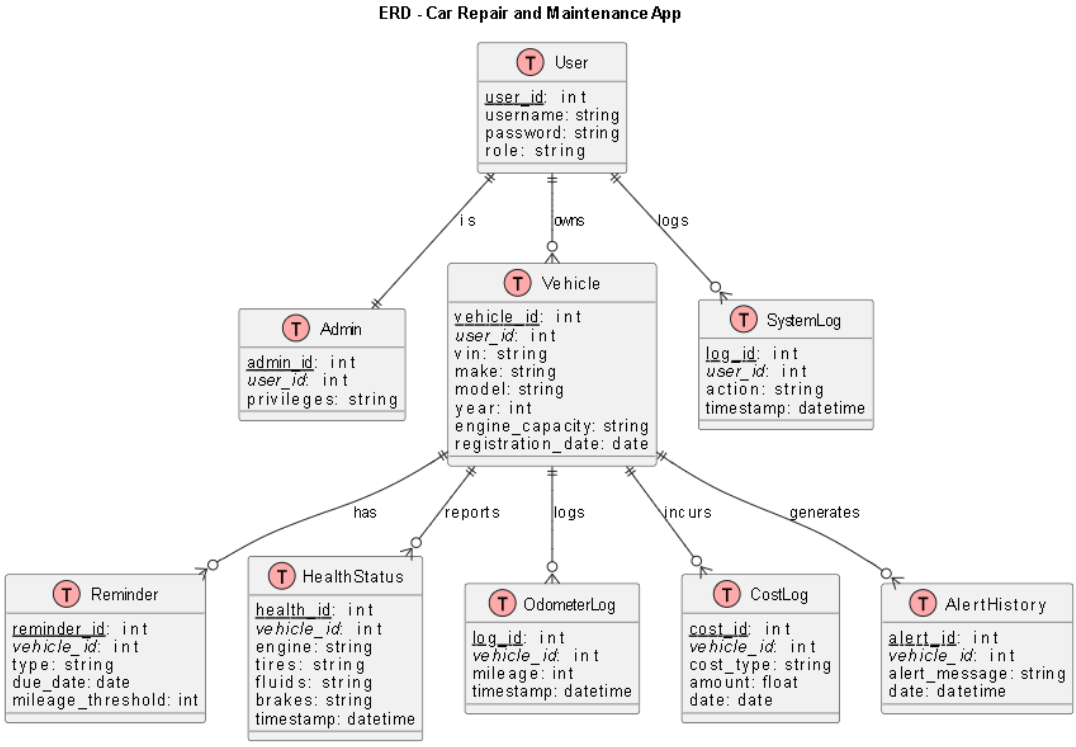

Figure 14: Entity-Relationship Diagram