Car Maintenance CLI - Requirements Document

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements and specifications for the Car Maintenance CLI (Command Line Interface) project. It outlines the features, functionalities, constraints, and constraints of the application.

1.2 Scope

The Car Maintenance CLI is designed to assist users in managing and tracking their vehicle's maintenance schedule and expenses. It provides features such as user registration, car information management, maintenance record management, service reminders, and expense tracking.

1.3 Document Conventions

- **Requirement ID:** A unique identifier for each requirement (e.g., FR01 for Functional Requirement 1).
- **Dependencies:** Any dependencies or related requirements.
- **Description:** A detailed description of the requirement.
- **Acceptance Criteria:** The conditions that must be met for the requirement to be considered fulfilled.

1.4 Definitions

- **User:** A person using the Car Maintenance CLI application.
- **Maintenance Record:** A record of a maintenance activity for a specific vehicle, including details such as date, type, cost, and notes.
- **Service Reminder:** A notification to the user regarding upcoming maintenance tasks based on mileage or time intervals.

2. System Overview

2.1 System Description

The Car Maintenance CLI is a command-line tool that allows users to manage their vehicle's maintenance. It includes features like user registration, car information management, maintenance record management, service reminders, and expense tracking.

2.2 Goals and Objectives

The primary goals and objectives of the Car Maintenance CLI project are as follows:

- Provide users with an efficient and user-friendly way to manage their vehicle maintenance.
- Ensure data accuracy and security.

- Enhance the user experience by offering reminders and expense tracking.

3. Functional Requirements

3.1 User Registration

- **FR01:** Users can create a file with their personal information about their vehicles.

3.2 Car Information Management

- **FR02:** Users can add multiple cars, including make, model, year, and VIN.
- **FR03:** Users can view and update car information.

3.3 Maintenance Record Management

- **FR04:** Users can add maintenance records, including date, type, cost, and notes.
- **FR05:** Users can view maintenance records.
- **FR06:** Users can set maintenance reminders based on mileage or time intervals.
- **FR07:** Users receive notifications for upcoming maintenance.

3.4 Expense Tracking

- **FR08:** Users can track maintenance expenses.
- **FR09:** Users can view expense reports and total expenses.

3.5 Backup and Restore

- **FR12:** Users can back up their data.
- **FR13:** Users can restore data from a backup file.

3.6 Data Export

- **FR14:** Users can export maintenance data to a CSV file.

4. Non-Functional Requirements

4.1 Performance

- **NFR01:** The application should respond to user inputs promptly.
- **NFR02:** It should handle a large number of maintenance records efficiently.

4.2 Security

- **NFR03:** Access to sensitive features and data should be restricted to users in possession of the file.

4.3 Usability

- **NFR04:** The CLI should have an intuitive and user-friendly interface.
- **NFR05:** Documentation should be clear and accessible.

4.4 Compatibility

- **NFR06:** The application should be compatible with Python 3.6+.
- **NFR07:** It should work on major operating systems (Windows, macOS, Linux).

4.5 Reliability

- **NFR08:** The application should be stable and error-tolerant.

4.6 Data Storage

- **NFR09:** Data should be stored in a reliable and maintainable database.

5. User Interface

5.1 CLI Design

The CLI should have a clear and intuitive command-line interface with well-defined commands and options.

5.2 User Flow

The user flow should be designed to guide users through the process of adding cars, recording maintenance, setting reminders, and viewing reports seamlessly.

6. Constraints

6.1 Technology Stack

- The project must use Python for development.

6.2 Timeframe

The project should be completed within the specified timeframe.

7. Risks

7.1 Identification

- Potential risks include data loss, security vulnerabilities, and scope creep.
- Compatibility issues with different Python versions and operating systems.
- User adoption and acceptance may vary.