Software Requirements and Design Document

For

Group 9

Version 1.0

Authors:
Jack Hyland
Gabriel Rigdon
Joab Temotio

Overview

The Car Maintenance Tracker is a web application built for DIY mechanics to keep a detailed record of their vehicle maintenance. Users can add multiple cars to their profiles and log important details like service dates, mileage, and descriptions of repairs. The home page provides an overview of all added vehicles, while each one has a dedicated profile page where users can open and submit a form to add maintenance items. The User Profile page currently stores basic user information, with plans for more personalized features in future updates.

Data related to vehicles and maintenance is temporarily stored to allow users to track their vehicle's history. The app features a simple, intuitive interface with forms for entering vehicle and maintenance information, along with interactive elements like buttons and hover effects to enhance usability. While there is no backend storage or advanced features like service reminders or reports yet, this version focuses on the essential functions for tracking maintenance for multiple vehicles, laying the groundwork for future upgrades.

Functional Requirements

1. Navigation and Page Structure

1.1 The system shall provide a Home Page, User Profile Page, and Car Profile Page, accessible through a navigation bar present on all pages. (High Priority)

2. Car Management

2.1 The system shall provide a button on the Homepage labeled "Add Car", which, when clicked, shall open a form allowing users to enter car details, including:

- Nickname
- Car Make
- Car Model
- Year
- Trim
- Engine
- Transmission
  (High Priority)

2.2 The system shall allow users to submit the car form, after which the entered details shall be displayed in a dedicated box on the Home Page. (High Priority)

2.3 The system shall allow users to click on a car’s box on the Home Page, which shall navigate them to the Car Profile Page. (High Priority)

3. Maintenance Tracking

3.1 The system shall provide an "Add Maintenance Item" button on the Car Profile Page, which, when clicked, shall open a form for users to enter maintenance details, including:

- Date
- Cost
- Description
  (High Priority)

3.2 The system shall allow users to submit the maintenance form, displaying a confirmation message that the form was submitted successfully. (Low Priority)

- Rationale: Currently, maintenance records do not persist, but submission confirmation reassures users that their input was received.

4. User Profile

4.1 The system shall provide a User Profile Page that displays a sample user’s information, including:

- Name
- Email
- Phone Number
  (Low Priority)
- Rationale: The current version does not support user-specific accounts, so this page is informational only.

Non-functional Requirements

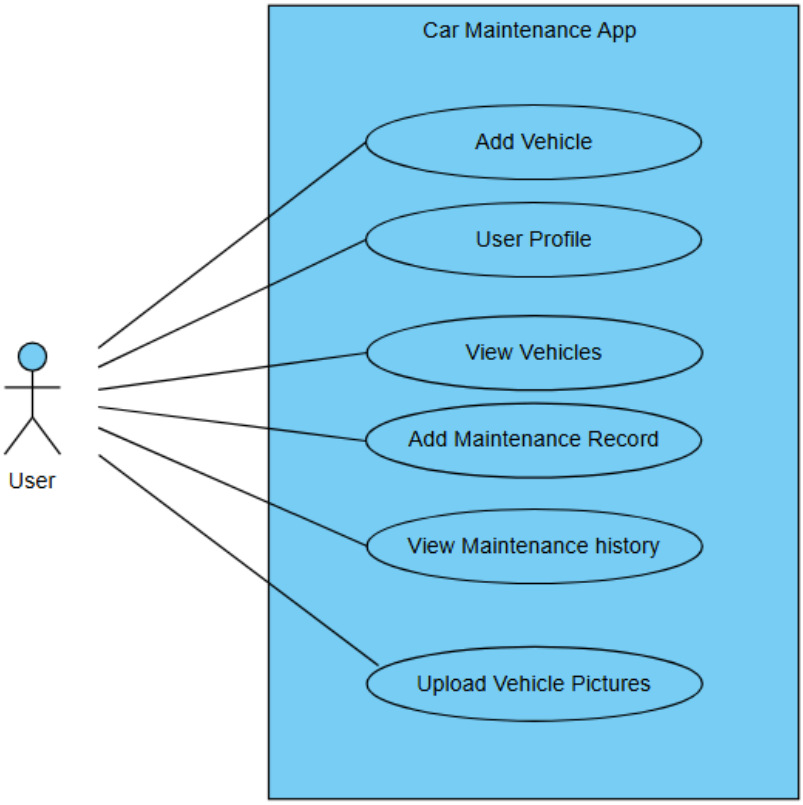
Performance
1.1 The system shall load each page within two seconds under normal network conditions.
1.2 The system shall allow users to navigate between pages without requiring a full page reload.

Usability
2.1 The system shall use a consistent font and color scheme across all pages.
2.2 Buttons and interactive elements shall have hover effects to provide visual feedback.
2.3 Forms shall provide clear input validation messages if a required field is missing.

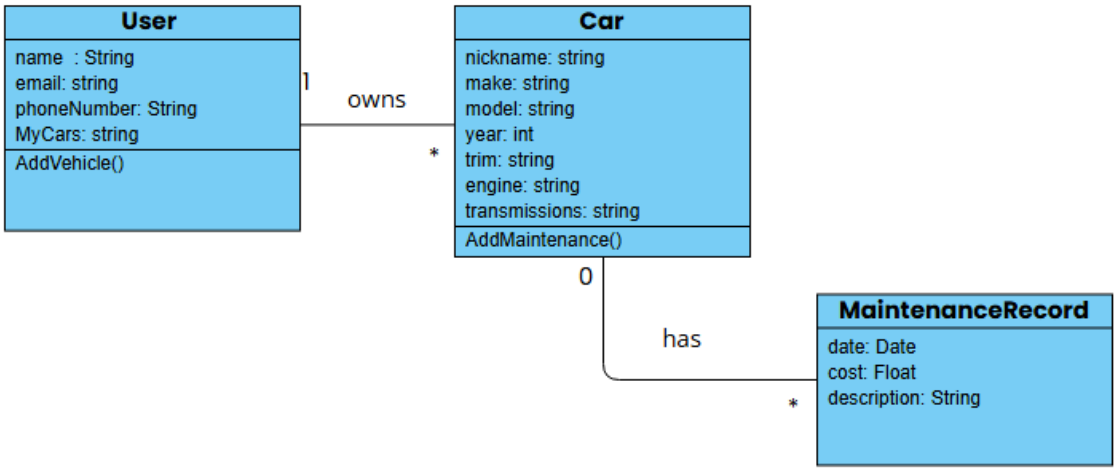
Reliability & Availability
3.1 The system shall not crash due to invalid user inputs in forms.

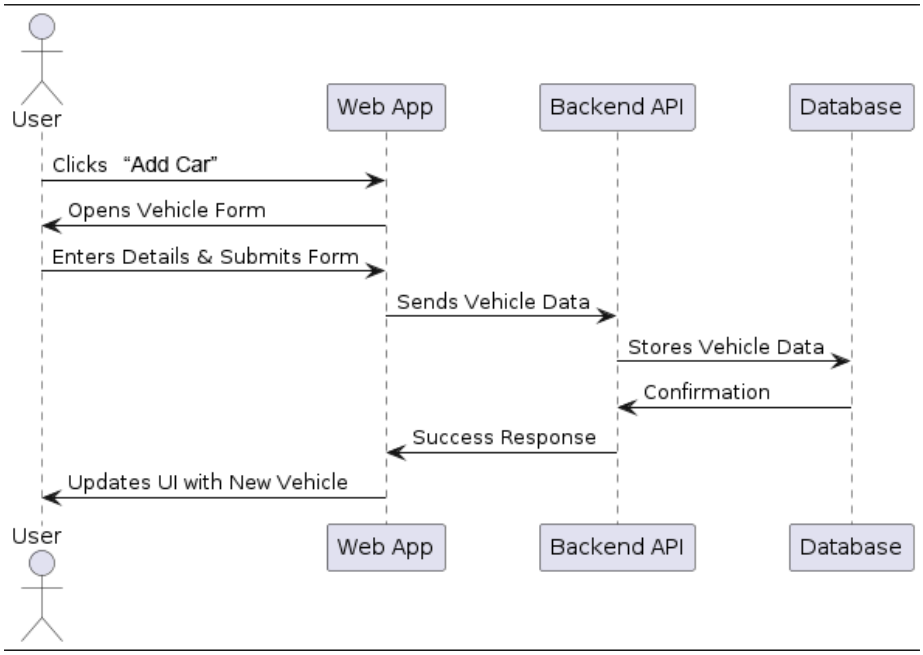
Maintainability & Scalability
4.1 The system shall be structured in a modular way to allow easy updates to individual components.

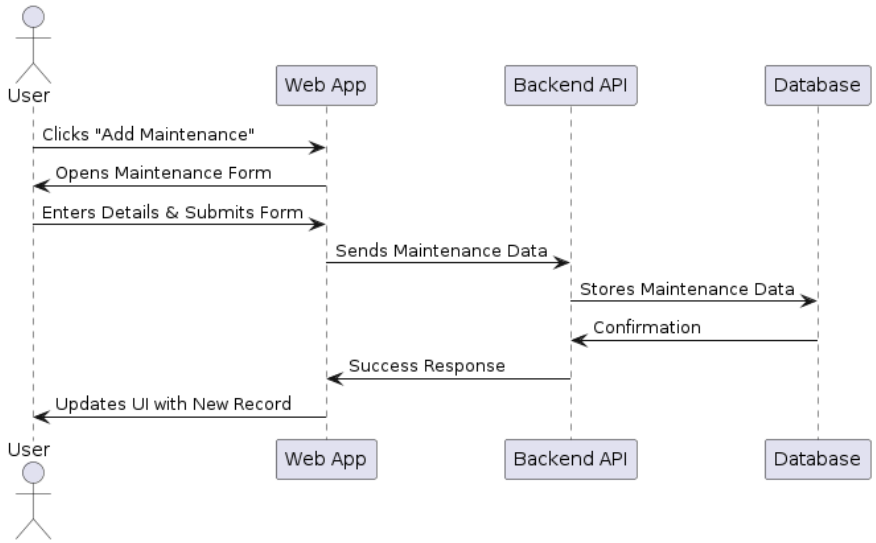
Use Case Diagram

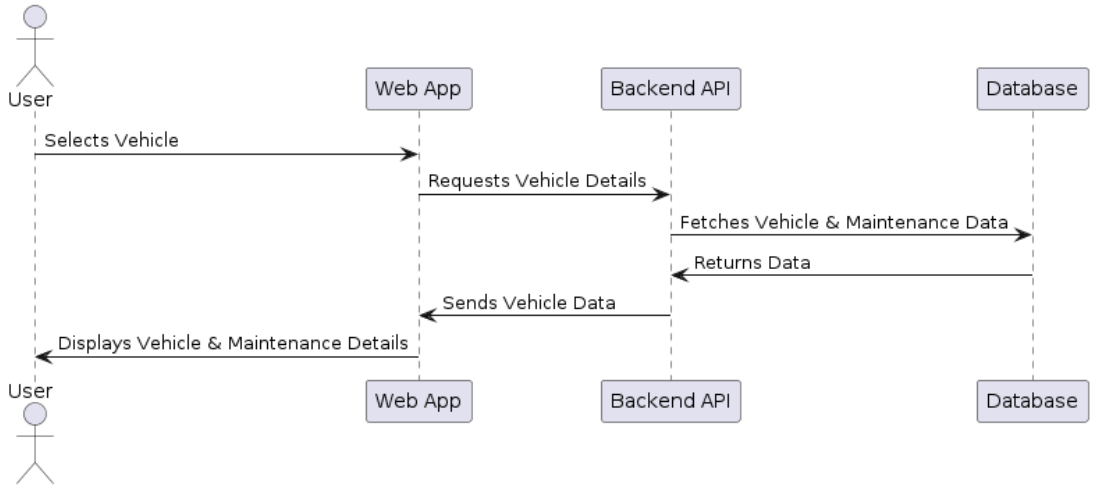


Class Diagram and/or Sequence Diagram









Operating Environment

The Car Maintenance Tracker is a web-based application designed to run on desktops, laptops, tablets, and mobile devices through modern web browsers like Google Chrome, Firefox, Safari, and more. It's usable on macOS, Windows, and Linux. The front end is built using react and styled in CSS. In the future, the back end will be developed in C# and the database will be built in PostgreSQL.

Assumptions and Dependencies

- Assumptions:
    - Frontend-Backend Integration will proceed smoothly without major architectural conflicts.

    - Scalability & Performance are achievable within the current infrastructure.
    - Security & Compliance will meet standard requirements without unexpected changes.
- Dependencies:
    - Third-Party APIs & Services (e.g., authentication, payments) must remain stable.
    - Database System (e.g., PostgreSQL) should handle expected performance loads.
    - Hosting & Deployment Infrastructure must be reliable and cost-effective.