Maintenance Buddy

Ashton Hull, Jordan Glatz, Daniel Vincent

- 1: Introduction
  - **1.1 Purpose**

    The purpose of this document is to define an outline for Maintenance Buddy. It will focus on defining the vision of this project as well as constraints, such as time and resources. This outline will be a reference for all involved such as stakeholders, developers, and users because a clear problem will be acknowledged and solved.

    Maintenance Buddy is a tool in the form of a mobile app that helps keep a complete maintenance log for your vehicle in one easy to access place. Users have the ability to input a multitude of information such as services performed, mileage, dates, receipts, photos, and notes. By doing this a log is created where information on your car can be easily organized. Furthermore, manufacturers provide recommended maintenance schedules based on vehicle make, model, and year, which can be used to keep users accountable. Maintenance Buddy will be a tool that will consistently help in caring for and maintaining your vehicle.

  - **1.2 Scope**

    Maintenance Buddy will be a mobile application that:

    - Focuses on ease of use and organization for individuals
    - Creates a maintenance log for users to view
    - Helps users to know when maintenance is required
    - Allows users to record changes with their vehicle
- 2: Positioning
  - **2.1 Business opportunity**

    Today, most people own a car but maintenance tracking is often a forgotten part about ownership. Without proper documentation it can make it hard to properly know what needs work or when something was changed. This creates a perfect opportunity for an easy to use app utilizing a user-friendly interface to track and store this data. This app will replace relying on one's memory or trying to utilize basic note taking.

  - **2.2 Problem statement**

    Keeping track of personal vehicle records can be hard without proper organization. Receipts get misplaced, invoices scattered, and it can be hard to remember everything that has been done to the car. This can make problems arise when trying to sell the car, negotiating repairs, handling warranty issues, and disputing questionable work.

- 3: Stakeholder and user descriptions
  - **3.2 Stakeholder Summary**

Name	Skills	Role
Ashton Hull	Java, SQL, Javascript	Backend development, Database management, GUI
Daniel Vincent	Java, SQL	GUI & Frontend Development
Jordan Glatz	Java	Frontend development & Backend development

  - **3.3 User Summary**

    Users of Maintenance Buddy include:

    - Individual car owners
    - Used car sellers
    - Owners of multiple cars
- 4: Product overview
  - **4.1 Product perspective**

    Common Car maintenance logs have tools included like CARFAX, FIXD, Google Sheets/Microsoft Excel, physical notebooks, and templates used by subscription based fees. Maintenance Buddy is an app where you can store your car's data, maintenance history, and future vehicle services and issues which need attention. The product is built around consistency and reliability. Users are assisted in scheduling maintenance and are reminded to upload information at the point of service and care. The software automatically creates a clear maintenance schedule that demonstrates responsible car ownership by asking users to enter accurate information each time and safely logging notes, images, and receipts.

  - **4.2 Summary of capabilities**

    Maintenance logging, Reminder Scheduling, Vehicle profile management, Document Storage, History visualization, Search and Filtering, Exportable Reports, Multi-Vehicle Support.

  - **4.3 Assumptions and dependencies**
  - **App Dependencies**
    - Users have internet access
    - Users Manually enter accurate Data

        - Manufacturer maintenance data is available
        - Users have Up-to-Date Software to use application
        - Users have correct documentation
        - Users have Access to device features such as camera and file storage for uploading
    - **Backend and Infrastructure Dependencies**
        - A backend server or cloud service to store user data securely
        - Database system to manage vehicle profiles, records, and documents
        - Cloud storage service for handling receipt uploads
    - **Data Dependencies**
        - Manufacturer maintenance schedules based on vehicle make, model, and year
        - Accurate user input for mileage, dates, and services performed on the vehicle.
        - Maintenance categories to ensure consistent logging and reporting.
    - **Notification Dependencies**
        - Push notification services for maintenance reminders
        - Background task scheduling supported by the M.O.S (Mobile Operating System)
        - User permissions for Notifications and reminders.
- 5: Product features
    1. Users can enter their year, make, and model to search for an OEM maintenance schedule. Schedules for many of the most common models will be created manually by us.
    2. Users can create a maintenance schedule for a specific year, make, and model if one does not exist already. Schedules for all other models will be community-driven. For community members to create a schedule, they must have access to the manual and be able to list it as a reference. This is so that other users can ensure the schedule's validity. Note: manuals are copyrighted content, and direct access to the manual within the app is not legal.
    3. The app can send reminders to schedule maintenance appointments by comparing the OEM schedule and the time/mileage since last service.
    4. Upon receiving a scheduling reminder, users may create appointment entries. Appointment entries do not allow you to schedule appointments within the app. Instead, they are meant to be done after the user has already scheduled the appointment externally. Appointment entries must contain a date and time for a follow-up. They may optionally contain information like "Service being performed", "Company Title", etc.
    5. When creating an appointment entry, users may quickly add information by having the app parse a screenshot of the "appointment confirmation details".
    6. When the time for a follow-up comes, the app will send a reminder letting the user know that it is time to document the service.
    7. Users may view their vehicle's maintenance history on a timeline and expand a "service event" to view the full details.
    8. While viewing the maintenance history, appointment entries will show up on the timeline. Upon completion of the follow-up they will become a service event.

9. If users do not want to use the appointment entry and appointment reminder features, they may opt to manually create the service event instead. Users will select the date the service occurred.
10. When creating a service event manually, users may select options including but not limited to: 1) "I did this service myself" 2)"Cosmetic modification", 3) "Performance modification"
11. Users may export complete maintenance history. This may come in two forms: 1) "Maintenance Summary" (PDF/DOCX) : Condensed version of the vehicle history, includes mostly text, maybe 1 photo per service.
    2) "Full Report" (ZIP Archive) : Package containing full, unmodified vehicle history. Inherently a file structure like:
    "2010 Honda Accord History" → "01-01-26 Oil Change" → before.jpg, after.jpg, invoice.jpg, MechanicComments.txt

- 6: Constraints
  - **Time Constraints**: Limited development time within the semester can lead to problems with not having enough time to fully design and test the software and may lead to restricted features.
  - **Scope Risk:** Too many planned features can make a project large and completed within the time limit will become a bigger issue.
  - **Maintenance Schedule availability:** scheduling for common models may be readily available but because maintenance schedules are community-driven, owners of less common models may have to wait before a maintenance schedule is available.
- 9: Other product requirements
  - **Availability:** software must be able to run on all platforms including android and IOS operating systems.
  - **Scalability:** software must be able to handle multiple cars per user.
  - **Accessibility:** User interface must use fonts that are easy to read and appeal to a wide range of users.
  - **Notifications:** Users must be able to receive updates and reminders based on vehicle recommendations.
  - **Offline Availability:** Users will be able to use the app and upload without the use of an internet connection.
  - **Maintainability:** System should be structured that allows for future updates and features. Code and DS should support debugging and maintenance overtime.
  - **Security:** Sensitive data should be protected, and User accounts should require authentication to access personal and vehicle data.
  - **Performance:** Records and Vehicle data should be loaded quickly and easy to use, and uploading photos or receipts should be completed in a good amount of time.