UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

A REPORT ON INTERNET PROGRAMMING AND MOBILE APPLICATION

DERPARTMENT OF COMPUTER ENGINEERING

COURSE CODE: CEF 479

COURSE INSTUCTOR: Dr. NKEMENI VALERY

DEPARTMENT: COMPUTER ENGINEERING (SOFTWARE/NETWORK)

GROUP MEMBERS:

LEKEUGO DEMELIEU ROCHINEL	FE22A237
CHUYE PRINCELY TATA	FE22A184
DIONE CHANCELINE NZUO SONE	FE22A187
ACHUO ESEGNI	FE22A135
MISHAEL ZABUD	FE22A248

Table of Contents

Requirement Gathering

Introduction

Requirement gathering is the process of identifying and documenting what a system should do based on the needs of its users and stakeholders. It helps ensure the final product meets user expectations and solves the right problems. This step involves using techniques like interviews, surveys, and brainstorming to collect both functional and non-functional requirements.

I. Stakeholder Identification

In the development of an AI-powered car fault diagnosis mobile application, it is important to identify all relevant stakeholders. These are individuals or entities that either use the application, influence its development, or are affected by it directly or indirectly.

A **stakeholder** is anyone who has a vested interest in the project, regardless of whether they use the application.

While all users are stakeholders, not all stakeholders are users.

a. Identified Stakeholders and Their Roles

- **Car Owners / Drivers:**
  Primary end users of the app who scan dashboard lights and analyes engine sounds for faults.
- **Auto Mechanics:**
  Can use or verify the app's diagnosis; may recommend it to customers.
- **Software Developers:**
  Handle the frontend/backend app implementation and integration of models.
  Responsible for developing and training the computer vision and audio recognition models.
- **Car Manufacturers and Investors:**
  Provide information on official warning symbols and diagnostic codes.
  May have future interest in commercializing or investing in the product

b. Importance of Stakeholder Identificationn

- Helps ensure that the system meets real-world needs.
- Allows the developer to balance priorities between **technical accuracy**, **usability**, and **user experience**.
- Encourages collaboration between developers, experts, and users to make the app functional and useful

Stakeholder identification plays a critical role in the early phase of the software development life cycle. By properly classifying both users and non-user stakeholders, this project can meet both functional and non-functional requirements while delivering value to all relevant parties.

II. Requirement gathering techniques

Requirement gathering is a critical initial phase in any software development lifecycle, particularly for user-centered and intelligent systems like the ***Car Fault Diagnosis Mobile Application*.** This phase involves the systematic identification, analysis, and documentation of what users need from the system, as well as the operational, functional, and technical specifications necessary to guide development. The success of the application heavily depends on how well the requirements align with user needs and real-world automotive scenarios.

Given the complexity of diagnosing car faults—especially for non-technical users—and the integration of machine learning (ML) and artificial intelligence (AI), it is essential to use multiple complementary techniques to ensure that the requirements gathered are accurate, practical, and relevant.

❖ Objectives

- To identify and understand the specific needs, pain points, and expectations of car owners regarding vehicle diagnostics.
- To gather both qualitative and quantitative insights that inform feature development, UX/UI design, and ML model requirements.
- To validate the problem statement through direct interaction with stakeholders such as car owners and mechanics
- To note and analyze existing car diagnostic tools and mobile applications for inspiration and gap analysis.
- To uncover hidden or non-obvious requirements through collaborative brainstorming and expert consultations.
- To ensure that the system requirements are comprehensive, covering all technical, functional, and user experience aspects of the solution.

The following techniques were employed to ensure well-rounded and validated requirements:

a. Surveys

Surveys were deployed to gather quantitative data about user habits, confidence levels, and preferences when diagnosing car faults. This method was essential in helping us understand the real-world behavior and needs of car owners

- **Objective:**

To understand car owners' behavior, challenges, and preferences related to car maintenance and dashboard light interpretations.

❖ **Tools Used:**

Google Forms (for online surveys)

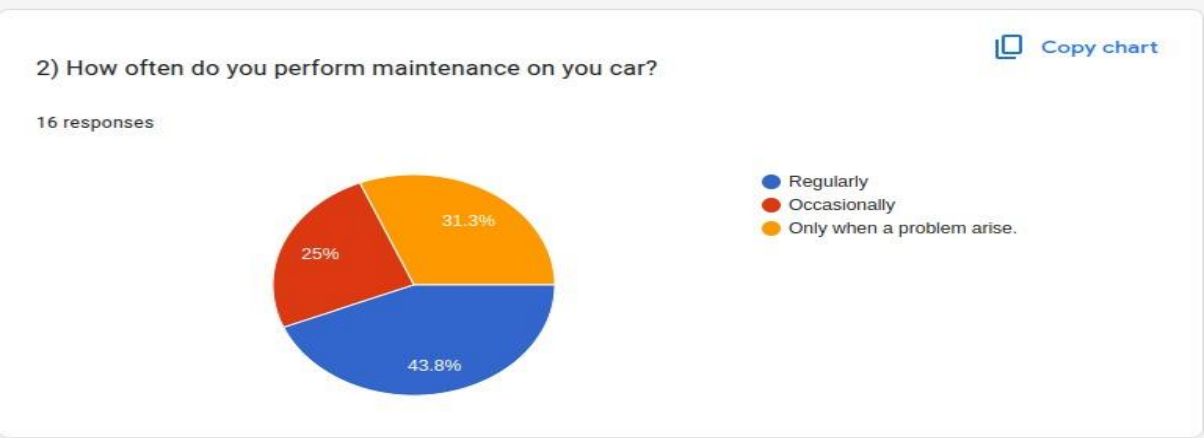
❖ **<u>Key Findings</u>**

Here are some of the important responses we got from the different stakeholders based on their experiences with cars

1. <u>From Car owners and Drivers</u>

Q2: Car Maintenance Frequency

- **43.8%** of the people perform maintenance **regularly**
- **31.3%** only perform maintenance **when a problem arises**
- **25%** do maintenance **occasionally**

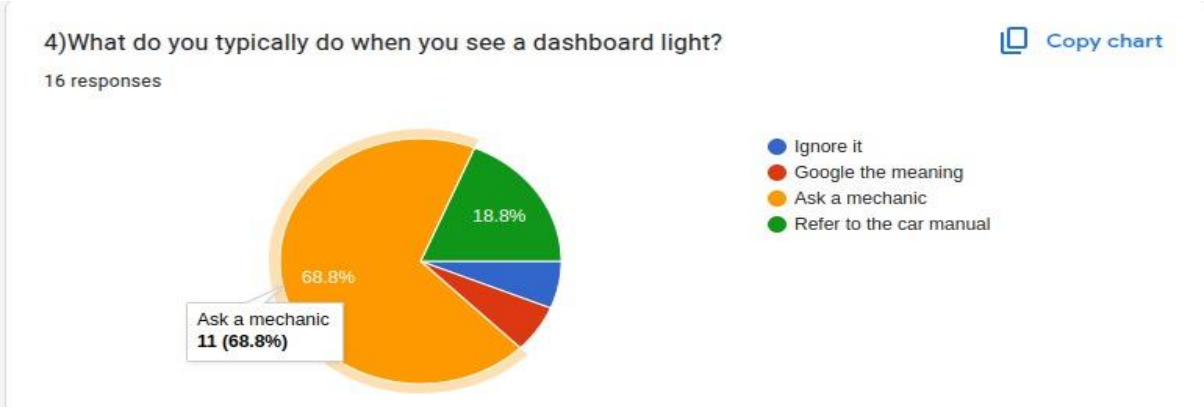


Insight:
Most users are proactive, but a significant number delay maintenance until issues appear. This supports the need for a smart diagnostic tool that alerts and guides users.

- **Based on the Dashboard Lights, We got the following responses from car owners**

Q4: Actions Taken When Dashboard Lights Appear

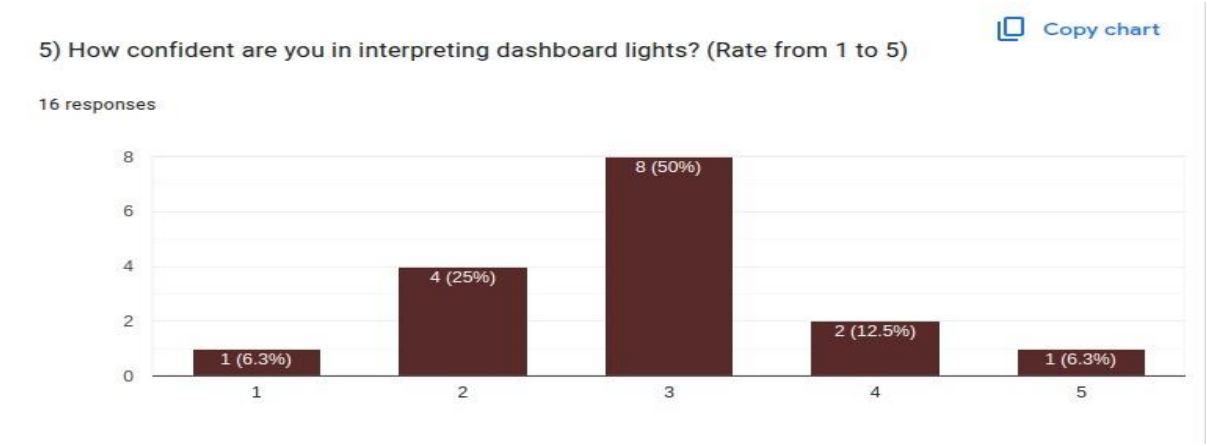
- **68.8%** of the people ask a mechanic for help
- **18.8%** refer to the car manual
- **6.3%** Google search the meaning
- **6.3%** ignore it



Insight:
Users overwhelmingly rely on mechanics, indicating a lack of confidence or awareness about dashboard lights. Your app can empower users by making dashboard warning light interpretation simple and accessible.

Q5: Confidence in Interpreting Dashboard Lights (Scale of 1-5)

- Majority of the people rated **3 (50%)**, meaning average confidence
- **25%** rated **2**
- Only **6.3%** rated **5 (very confident)**

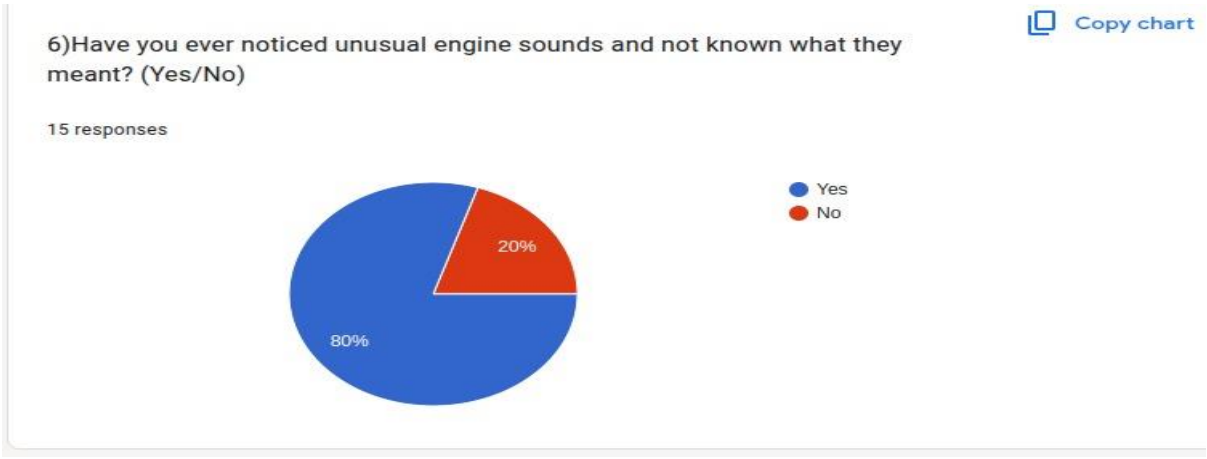


Insight:
There is a clear gap in knowledge or understanding. Most users aren't confident interpreting these lights, justifying the app's educational and advisory features.

- **Based on Engine Sounds, Users gave us the following Feedbacks**

Q6 Have you ever noticed unusual engine sounds and not known what they meant?

- **80%** of the people were unable to get the meaning of strange engine sounds
- While **20%** always knew what it meant

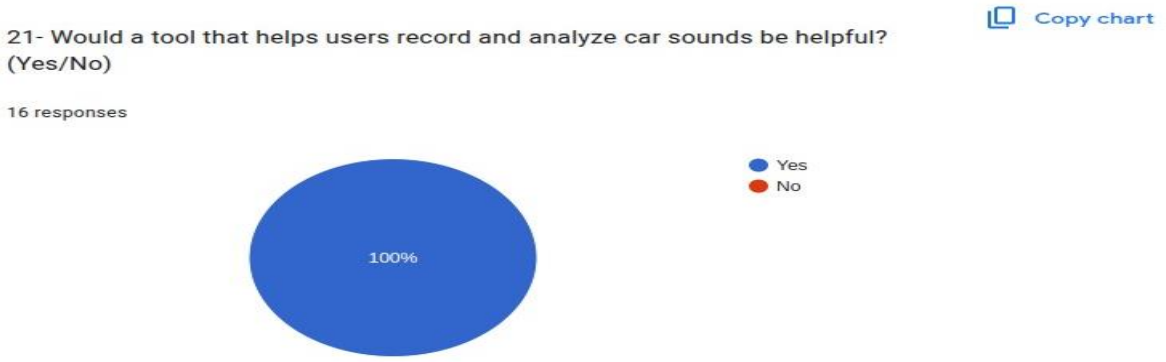


Insight
A great majority of the people can't identify car problems from the engine sounds.
This shows that the application will serve a lot in terms of providing users with valuable interpretation of engine sounds and recommendation

---

Q21. Would a tool that helps users record and analyze car sounds be helpful

- **100%** of the people would like to use an AI tool for their car fault diagnoses



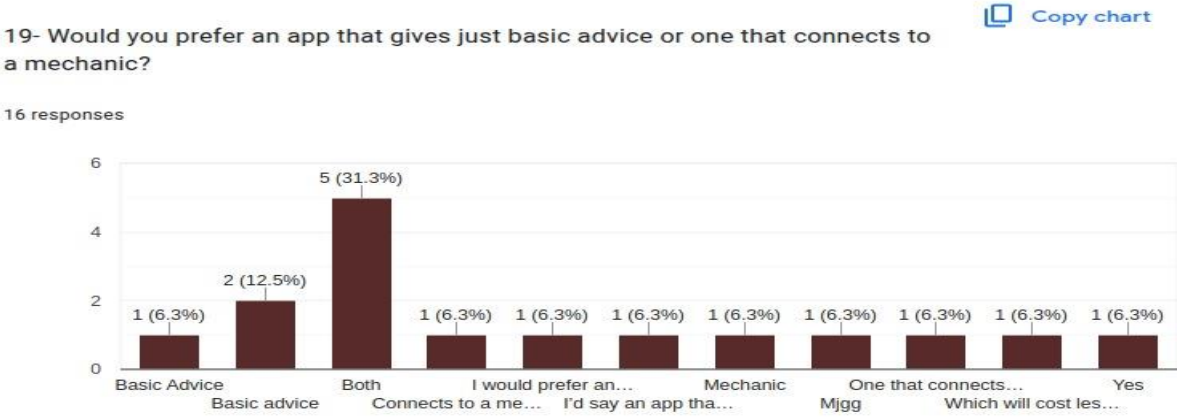
Insight

Everyone would like a tool like this, so its really going to be a good innovation according to the responses gotten from the people

---

Q19. Would you prefer an app that gives just basic advice or one that connects to a mechanic

- As seen below, **31%** of the people would like an app that will connect them to a mechanic



Insight

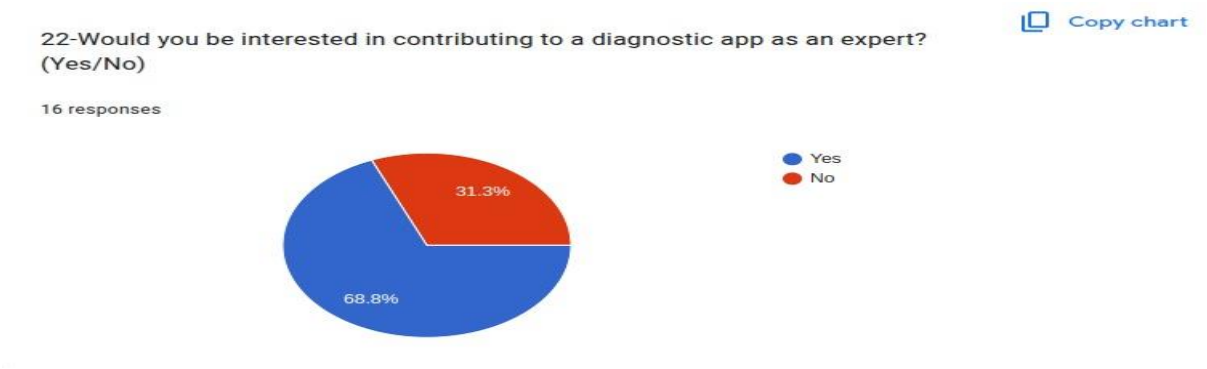
This will be a great implementation in the future, implementing such a function in our future upgrades

---

2. From Mechanics

Q22. Would you be interested in contributing to a diagnostic app as an expert?

- 68.8% of the mechanics we met were ready to contribute to this app as experts
- 31.1% were not really in to the idea



Insight:

A great deal of the people would love to contribute as experts meaning the intergration of the feature to link car owners to mechanics will be a great one

b. Interview Feedback Highlights

(From informal sessions)

- **Objective:**

- To gain deeper insights into the issues users face and understand expert perspectives.

We were able to talk to 3 car users during this session

During this sessions, stakeholders were asked some relavant questions on the different features and functionalities of the app and what they would like to see in the app and the following insights were gotten from their responses

- Users want a **simple, non-technical Application**
- Prefer **visual cues or icons** over text
- Would trust the app **more if it links to verified experts/videos**
- **Offline functionality** would be useful when strandedd
- Users also showed interest in sound-based diagnosis.

---

c. Brainstorming Sessions

- **Objective:**

To explore features and design decisions for the app.

- **Participants:**

- Team Members

- **Key Ideas:**

In addition to the insights gotten from the user, we also had some brainstorming to come up with some helpful features like:

- Integrate sound recognition for early detection

- Provide YouTube video suggestions through a YouTube API
- Display confidence scores for predictions to increase the trust of users
- Integrate a feature that links car owners to expert mechanics, this will also be part of the business plan to generate income for the team and from the app

---

d. Reverse Engineering

- **Objective:**

To examine existing apps and learn from their features, technologies, and limitations.

- **Apps Reviewed:**

App Name	Description	Features	Tech Stack	Technologies Used
OBD Auto Doctor	Diagnostic app using OBD-II	Real-time engine data, fault code reading	Java, Android SDK	OBD-II, Bluetooth, Diagnostic Trouble Codes (DTCs)
Car Scanner ELM OBD2	Comprehensive scanner tool	Real-time diagnostics, emission tests, dashboard customization	Java/Kotlin	OBD-II, ELM327 Adapter
Sound Analyzer Apps	Engine noise recognition tools	Frequency analysis, waveform visualization, real-time detection	Python, Flutter, TensorFlow	Microphone Input, Fast Fourier Transform (FFT), ML classification
AutoMate Diagnostic AI	AI-based diagnosis app	Symptom checker, fault prediction, suggestion engine	React Native, Python	TensorFlow, Firebase, REST APIs

CarMD	Cloud-based diagnostics and repair guide	Reports based on fault codes, repair cost estimates	Web-based	AI-driven diagnosis, repair cost APIs
My Car	Vehicle management and maintenance logs	Reminders, dashboard light tracking, fault code logs	Java, SQLite	Notification API, SQLite, Cloud Backup

❖ **Insights Gained:**

- Most apps rely on external OBD hardware and do not offer sound-based diagnostics.
- Few integrate educational tools to help users understand faults.
- AI/ML is underutilized in many popular apps, leaving room for innovation.
- Existing solutions lack personalized repair recommendations or trust-building indicators like confidence scores.

The reverse engineering process highlighted major gaps in current solutions, including the lack of AI-powered audio diagnostics and simplified user guidance. Our project is designed to fill these gaps by focusing on mobile-based, hardware-independent, intelligent diagnostics with accessible, user-friendly recommendations.

The survey and interview results strongly validate the need for this app. There's a clear demand for a **mobile-first, AI-powered tool** that enables users to:

- Diagnose dashboard alerts and engine sounds
- Gain confidence in car maintenance
- Reduce dependency on physical mechanics for basic issues

III. Data Gathering

- **Objectives**

- Understand the common behaviors and challenges car users face with vehicle faults.
- Identify the level of trust and acceptance of AI-based diagnosis tools.
- Determine the most desired features for an AI car diagnostic mobile application.
- Evaluate existing applications for inspiration and differentiation.

- **Techniques and Data Sources**

❖ **Data From Online Survey**

Tool Used: Google Forms
Respondents: Car owners, drivers, and mechanics
Purpose: Quantitative insight into users' behaviors and preferences

Key Findings:

Question	Key Result
Q2: Maintenance Frequency	43.8% regularly, 31.3% when a problem occurs
Q4: Action on Dashboard Lights	68.8% consult a mechanic
Q5: Confidence in Dashboard Light Interpretation	50% rated 3/5; 25% rated 2/5
Q6: Ability to Detect Engine Fault via Sound	80% cannot detect
Q19: Feature to Contact Mechanic	31.3% find it helpful
Q21: Helpfulness of Sound-based App	100% said yes
Q22: Mechanics willing to help train AI	68.8% said yes

❖ **Data From Interviews**

Participants: Selected car users and mechanics
Method: Informal one-on-one conversations
Purpose: Gather qualitative context to explain survey responses

Findings:

- Most users **lack technical knowledge** but want to understand basic faults.
- There is strong demand for **visual guidance**, like icons or videos.
- Users fear being misled by mechanics and **prefer AI assistance** as a second opinion.
- Offline use is essential due to poor connectivity in some regions.

❖ Data From Brainstorming Sessions

Participants: Project development team
Objective: Translate user feedback into actionable feature ideas

Outcomes:

- Use of **YouTube API** to recommend videos for faults
- Integration of **mechanic-matching service**
- Show **confidence percentage** of the AI prediction to increase users confidence in the App
- Develop a **simple UI** with icons for sounds, symptoms, and error lights to make the App interface less complex
- Enable **offline mode for easy access even in remote areas**

❖ Data From Reverse Engineering

Apps Analyzed:

App	Notable Features	Tech Used	Gaps Identified
OBD Auto Doctor	Real-time diagnostics via Bluetooth OBD	Java	Requires hardware
Car Scanner ELM OBD2	Diagnostic reports, emission tests	Java/Kotlin	Not user-friendly for laypersons
Sound Analyzer App	Detect frequencies and classify sounds	Python, TensorFlow	Not tailored to car issues
AutoMate AI	Predict faults via symptoms	React Native	Lacks mechanic interaction
My Car	Maintenance tracking	SQLite	No fault diagnosis
CarMD	Repair cost estimation	Web-based	No sound/audio feature

---

IV. Data Cleaning

After gathering data from various sources, a structured cleaning process was performed to ensure data quality, remove redundancy, and extract usable insights.

- **Survey Data Cleaning**

Step	Action
Incomplete Responses Removed	Survey entries with skipped questions were excluded
Consistent Formatting Applied	Normalized confidence rating scales (1 to 5)
Text Answers Grouped	Open-ended responses were categorized into keywords (e.g., "trust issues," "prefers videos")
Duplicate Entries Checked	Similar responses (e.g., repeated from same user) were filtered

---

- **Interview Data Cleaning**

Step	Action
Noise and Off-topic Remarks Removed	Unrelated chat excluded (e.g., price comparisons, casual jokes)
Merged Repeated Comments	Reworded and combined repetitive suggestions to avoid duplication

---

- **Brainstorming Idea Cleaning**

Step	Action
Grouped Similar Ideas	Suggestions like "video explanation" and "YouTube integration" were combined
Classified Features	Organized into categories: UI, AI Logic, External APIs, Connectivity
Filtered Out Impractical Concepts	Ideas requiring very expensive APIs or hardware were removed like the use of Bluetooth and ChatGPT API

❖ **Reverse Engineering Data Cleaning**

Step	Action
Extracted Feature Sets	Created a matrix of app features for comparison
Identified Gaps	Listed features missing in all compared apps like the use of ML for sound analysis in a mobile application and the recommendation of YouTube videos
Mapped to User Needs	Aligned missing features with survey/interview desires (e.g., AI sound detection + video explanations)

❖ **Summary of Cleaned Insights**

Theme	Validated Insight
User Knowledge	Most users can't interpret sounds or dashboard lights
Feature Demand	High demand for AI-based sound diagnosis and expert explanation
Trust Factor	Users more likely to use app if real mechanics back it up
Offline Use	Critical for rural areas or emergency use
UI Expectations	Icons, simplicity, and video-based education preferred
Competitor Gaps	No app currently integrates sound-based AI with expert-backed guidance in an offline-capable mobile solution

V. User Reluctance Assessment

User reluctance assessment is about understanding **why users might hesitate or refuse to use your mobile application**, even if it's helpful. It helps you discover barriers to adoption so you can fix or reduce them early in your design.

❖ Why It's Important:

You can build the best app in the world, but if users find it:

- Too complicated
- Untrustworthy
- Not relevant to them
  ...they may not use it.

In our **car fault diagnosis app**, for example, from the responses we got from most car users, we thought about why car owners might **not** want to use the app — and how to **remove those blockers**.

❖ Common Reasons for User Reluctance (with examples):

Reluctance Reason	Example in our Project	How to Reduce It
Lack of Technical Skills	Older car owners might not be comfortable using mobile apps.	Use a very simple interface with guidance.
Privacy Concerns	Users may fear their car data is being shared or tracked.	Add clear privacy policies and permissions.
Distrust in Accuracy	They may not trust the app's fault diagnosis.	Use AI with a good accuracy rate and show confidence levels.
Preference for Mechanics	Some users may prefer going to a human expert.	Position the app as a "first check" tool.
App Overload	Users may feel they already have too many apps.	Make the app light, useful, and unique.
Battery/Data Usage	If it uses too much battery or internet, users may avoid it.	Optimize performance and allow offline mode.

❖ How we Assessed It in our Project:

We use the following methods:

1. **Survey Questions**
2. **Interviews**
   We spoke directly to potential users and ask open-ended questions:

❖ <u>Some responses we got from the survey in regards to Reluctance Assessment:</u>

Survey Question 1: ***Would you feel confident using an app to check car faults?***

- **Yes, I would feel confident.**
  "I think it would be very helpful, especially when I can't reach a mechanic quickly."
- **Not sure.**
  "It depends on how accurate the app is and if it's easy to use."
- **No, I wouldn't feel confident.**
  "I prefer a mechanic to check my car; I'm not sure an app can do it properly."

Survey Question 2: ***What concerns would stop you from using such an app?***

- **Accuracy of the diagnosis.**
  "I'm worried the app might give wrong information about my car's problems."
- **Complexity of use.**
  "If the app is too complicated, I might not be able to use it effectively."

Survey Question 3: ***Do you prefer using a mechanic over an app? Why?***

- **Yes, I prefer a mechanic.**
  "Mechanics have experience and can physically inspect the car, which an app can't do."
- **No, I prefer using an app.**
  "An app would be quicker and could help me understand issues before going to a mechanic. It helps me save money"
- **I would use both.**
  "I would use the app for initial checks and then consult a mechanic if needed."

Survey Question 4: What features would help you trust this app more?

"If the app is accurate in its diagnoses and recommendations, I'd be more likely to use it."

"Providing precise and reliable information about car issues is crucial."