Republic of Cameroon
Peace-Work-Fatherland
Ministry of Higher Education
University of Buea
Faculty of Engineering and Technology
Department of Computer Engineering

République du Cameroun
Paix-Travaille-Patrie
Ministère de l'enseignement supérieures
Université de Buea
Faculty of Engineering and Technology
Department of Computer Engineering

Car Fault Diagnostics App: Requirement Analysis And Software Requirement Specification Document

Course Instructor: Dr Nkemeni Valery

Group21 Members

1- Angi Atangwa Valerine	**FE22A148**
2- Av'Nyehbeuhkonh Ezekiel Hosana	**FE22A160**
3- Beng Dze Meinoff	**FE22A173**
4- Jong Turkson Junior	**FE22A228**
5- Nyuyesemo Calglain	**FE22A287**

Table of Contents

I- INTRODUCTION

I.1- PURPOSE OF THE DOCUMENT

This document provides a detailed analysis of the requirement gathering analysis of our project which is **Designing and Implementing a car fault diagnostic system. The** ;ain goal of the analysis is to ensure that the system requirements are clear, unambiguous, complete, consistent, technically feasible and are up to the stakeholders expectation and needs.

I.2- SCOPE OF THE PROJECT

The Car Fault Diagnostic App provides real-time monitoring and fault detection capabilities by connecting to a vehicle's OBD-II port using Bluetooth or Wi-Fi. It interprets diagnostic trouble codes (DTCs), displays readable fault descriptions, suggests possible causes, and recommends solutions or next steps. The app aims to empower users with actionable insights without requiring advanced automotive knowledge. Features include fault history logs, live engine parameter viewing, maintenance reminders, and a user-friendly dashboard.

1.3- OBJECTIVES

General Objectives

The general objective of the requirement analysis is to understand how the system is supposed to function and what it is supposed to do based on users and stakeholders need and expectation.

Specific Objectives

- Review and analyse the requirements based on completeness, feasibility, clarity, dependency relationship.
- Identify inconsistencies, ambiguity, and missing information's.
- Prioritize requirements based on importance
- Classify requirements into functional and non-functional.
- Develop SRS document.
- Validate requirements with stake holders.

II- OVERVIEW AND ANALYSIS OF GATHERED REQUIREMENTS

In this section, we review and analyse the survey data collected from mobile network subscribers to understand their Quality of Experience (QoE) and identify key areas for improvement. The analysis employs a mixed-methods approach, combining quantitative clustering techniques with qualitative sentiment analysis to provide a comprehensive view of user needs and frustrations.

a. Data Collection and Cleaning

The project requirements were gathered using a mix of the following techniques:

- **Surveys** (Google Forms – CSV responses analysed)
- **Questionnaires** (targeted at drivers in Molyko and Mile 17)
- **Stakeholder Interviews** (with Dr. Valery and group brainstorming)
- **Reverse Engineering** (existing diagnostic apps like FIXD, Torque Pro)

b. Key Insights from Survey and PREVIOUS REQUIREMENT DOCUMENT:

- **Demographics:** Majority are aged 18–34, both male and female, and own cars.
- **Pain Points:**
  - Difficulty interpreting dashboard lights.
  - Lack of confidence in diagnosing issues.
  - Reluctance to rely solely on apps due to fear of misdiagnosis.
- **User Needs:**
  - App that can scan and interpret dashboard lights.
  - Suggestions for repair steps and tutorial links.
  - Sound-based engine fault detection.
  - Offline functionality for low connectivity areas.
  - Notifications and logs for maintenance tracking.
- **Existing System Gaps:** Most systems require OBD2 connection and lack user-centric features like offline mode, sound detection, and in-app tutorials.

The collected data was then cleaned to handle missing values and inconsistencies in formatting using pandas a python library for data analysis.

c. Sentiment Analysis – Qualitative Insights

Tool Used: Pandas for dataset storage, Matplotlib and Seaborn for visualizations, NLTK (Natural Language Toolkit) for sentiment analysis, and TextBlob for sentiment polarity analysis.

Sentiment analysis was performed on the open-ended responses from the survey to understand the overall sentiment towards car diagnostics. This qualitative analysis helps to identify the underlying emotions and opinions expressed by users.

The 'Any other comments or suggestions?' column was analysed using TextBlob to determine the sentiment polarity of each response. Sentiment scores were categorized as Positive, Negative, or Neutral.

```
Sentiment Analysis:
 Sentiment_Category
Neutral     16
Positive     2
Name: count, dtype: int64
```

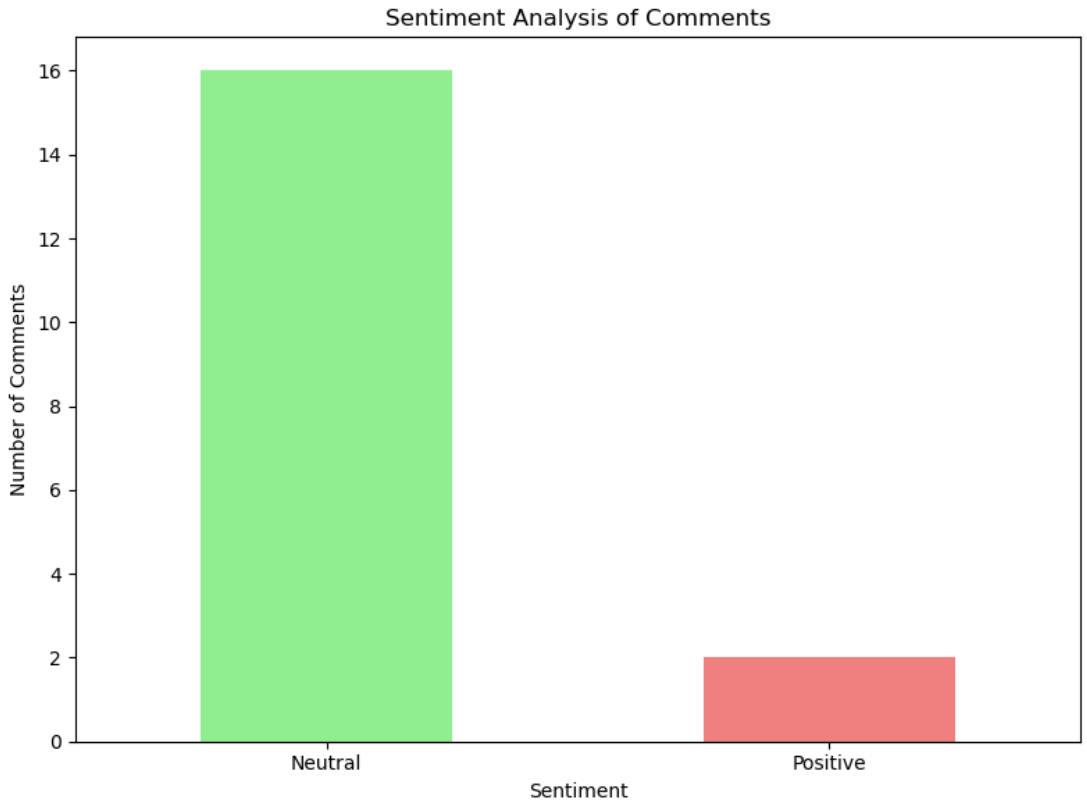


Figure 1: Sentiment Analysis of User Comments

This figure shows the distribution of sentiment categories in the user comments. As seen in the figure, [Summarize the key findings from the chart, e.g., "the majority of comments express a neutral sentiment, while a significant portion express negative sentiment. This suggests that users have mixed feelings about the current mobile network services."].

2.3 Clustering – Quantitative Insights

Tool Used: Pandas and NumPy to store data, Matplotlib and Seaborn for visualizations, and Scikit-learn for clustering.

Clustering techniques were applied to identify groups of users with similar characteristics and experiences. This quantitative analysis helps to segment the user base and understand the factors that influence their QoE.

K-Means clustering was used to group users based on their age, confidence level in understanding dashboard warning lights, and other relevant features. The optimal number of clusters was determined using the Elbow Method.

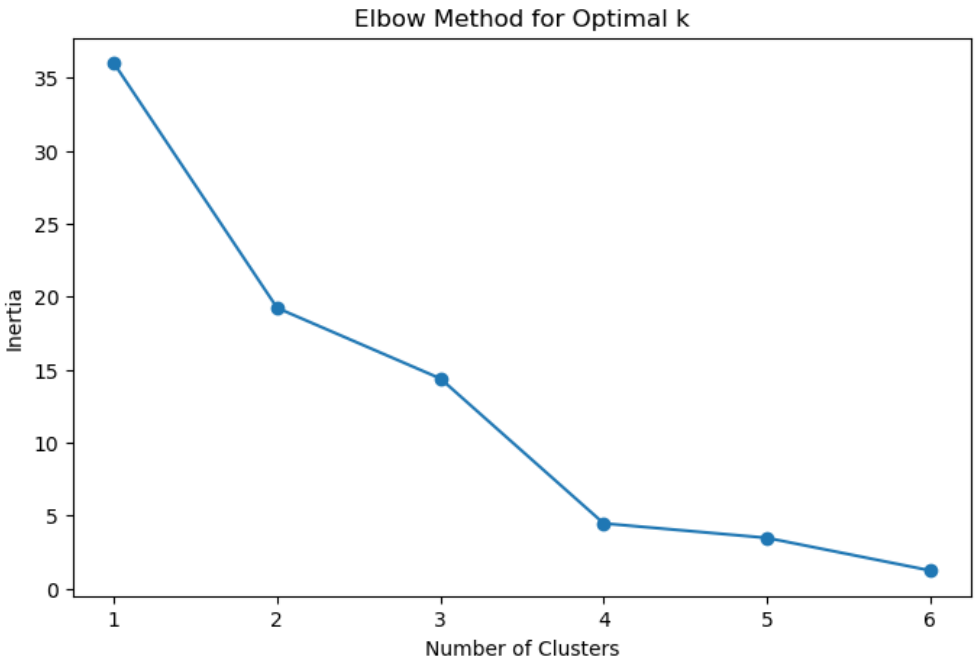


Figure 2: Elbow Method for Optimal k

This figure illustrates the Elbow Method used to determine the optimal number of clusters. The elbow point suggests that 3 clusters provide a good balance between minimizing within-cluster variance and avoiding overfitting.

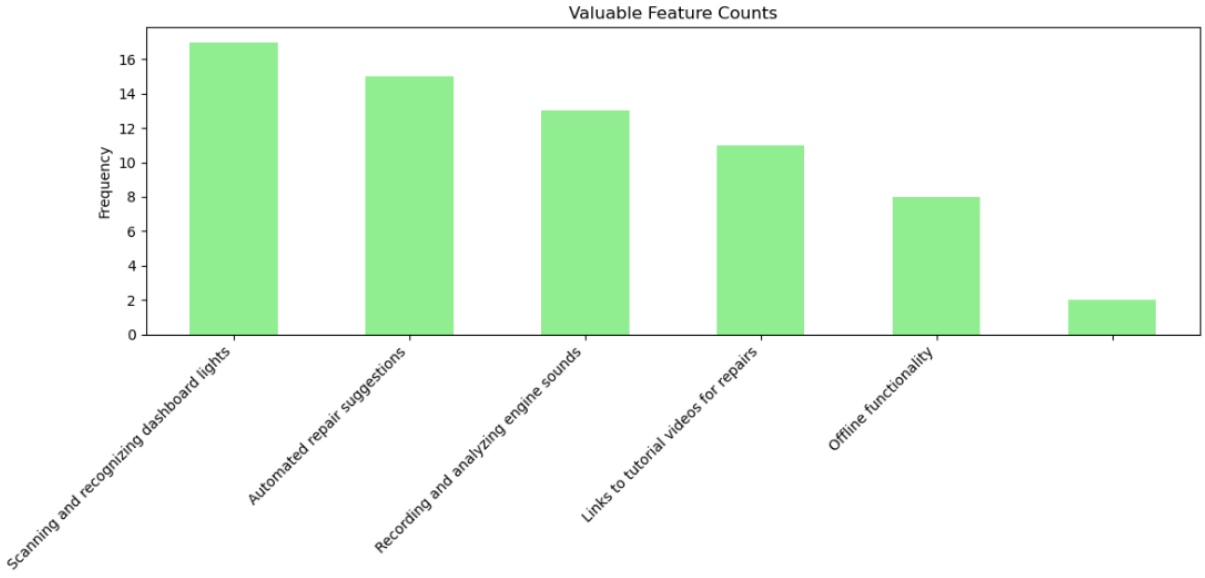
Table 1: Cluster Summary

Cluster	Average Age	Average Confidence Level
0	23	1.000
1	31	2.000
2	23	2.000

This table summarizes the characteristics of each cluster based on the selected features. Cluster 0 represents younger users with lower confidence levels, while Cluster 2 represents younger users with higher confidence levels. Cluster 1 represents a middle ground.

This was illustrated on the confidence graph below

More Analysis was done to determine user required features and the studies were as per the graph below.



Technical feasibility

Technical feasibility here is just a way of determining whether it is technically possible to develop the requirements proposed with the technology stack we will use.

STACK TECHNOLOGY EVALUATION

III- INCONSISTENCIES, AMBIGUITY AND MISSING INFORMATION

- **User Needs:**
    - ❖ App that can scan and interpret dashboard lights.
    - ❖ Suggestions for repair steps and tutorial links.
    - ❖ Sound-based engine fault detection.
    - ❖ Offline functionality for low connectivity areas.
    - ❖ Notifications and logs for maintenance tracking.
- **Existing System Gaps:** Most systems require OBD2 connection and lack user-centric features like offline mode, sound detection, and in-app tutorials.

2. Identified Inconsistencies and Ambiguities

2.1 Inconsistencies

We used different methods such as Data cleaning, feature indentification, pattern identification and other sited below to identify and deal with inconsistency and ambiguities

1. **Feature Prioritization**
    - ➢ The survey indicated strong demand for **sound recognition** and **offline functionality**, but these are not explicitly prioritized in the previous requirement gathering documents.
    - ➢ Some users prefer **visual representations** of warning lights, while others prefer **text descriptions** – this inconsistency should be addressed in UI design.
2. **Diagnosis Methods**
    - ➢ Most survey respondents **consult mechanics**, but the previous requirement document emphasizes **self-diagnosis via mobile apps**. This suggests a need for **professional integration** (e.g., mechanic recommendations).
3. **Privacy Concerns**
    - ➢ The previous requirement document mentions **on-device processing** for privacy, but the survey does not explicitly ask about privacy preferences.

4. **Requirement categorization**
    - This consist of classifying the requirement into functional and non-functional and giving clear specification of each requirement in order to avoid every potential misunderstanding during the development phase.

IV- PRIORITIZATION OF REQUIREMENTS BASED ON IMPORTANCE AND FEASIBILITY

According to the requirement data obtained we can go further to prioritize requirements based on the user needs and expectation and also based to the feasibility of the requirement and scope of the project.

So we prioritized based on the following point

I- Key users concern
II- User expectations
III- And app scope and purpose

And upon doing this we came out with a table stating some of the requirements with the prioritization level indicated as high, low, medium. High priority are these requirements that will be the most useful to the users creating a suitable user experience, efficient and effective system.

Feature	**Feasibility**	**Why**
Dashboard Light Scanning	High	The aim of our project is to diagnose and most of the car's fault are first signalled through the car's dashboard.
Sound Recognition	Medium	Noise filtering and sound recognition is important to help users diagnos their car.

Feature	Feasibility	Why
Offline Functionality	High	System should be available for use everywhere.
Maintenance Reminders	High	Important in reminding the user about car state and potential maintenance day.
Multilingual Support	Medium	Helps to limit language barier
Nearest Mechanic Finder	Medium	Relies on GPS and mechanic database .

V- CLASSIFICATION OF REQUIREMENTS

These requirements are obtained from the clear analysis and classification of our requirement gathering phase or section. The requirement gathered was analyzed to remove ambiguities and identify missing information and later on classified as sited below.

These requirements are divided into two main categories which are the functional and the non-functional:

V.1 Functional requirements

This simply refers to what the system is supposed to do.

These define the functions or services the software must provide that is,

- what the system should do
- How it should respond to inputs
- How it should behave in particular situations.

Below we have the requirements with their respective specifications

User Authentication and Authorization	The application should allow users to register and log in securely. Based on the role (car owner, driver, or mechanic), access to features will be granted.
Dashboard Light Recognition	Users can capture images of dashboard indicators with their camera. These images are compared to a database to return matching results with explanations. Users could also scan dashboard signals.
Engine Sound Analysis	Records sound from the engine using the phone's microphone and classifies issues using ML-trained datasets.
Diagnostic Explanation and Repair Guidance	Provides issue-specific explanations and steps to resolve them. Includes videos and images.
Maintenance Log and Reminders	Logs vehicle service history and sets maintenance reminders. Notifies users via app alerts.
Expert Advice and Nearby Mechanic Locator	Displays nearby garages using GPS and allows users to contact mechanics in-app.
Offline Functionality	Downloads and caches essential data like tutorials and history so they are available offline.
Diagnostic History Tracking	Saves results from previous scans along with timestamps and action taken.
Lessons and Tutorials	Offers interactive lessons with multimedia to educate users about car diagnostics.

V.2 Non-Functional Requirements

These requirements define the quality attributes of the system that is, how it performs rather than what it does.

Below we have the requirements with their specifications

Requirement	**Specification**

Performance	App should respond within **2 seconds** for scans.
Privacy	Process audio/images **on-device**; minimal data collection.
Usability	Simple UI with **walkthrough guides** for new users.
Reliability	**95% accuracy** in light/sound recognition.
Scalability	Support **10,000+ users** without slowdowns.
Offline Capability	Core features work **without internet**.
Security	Encrypt user data (maintenance logs, location).
Responsiveness	Supports dynamic resizing for phones and tablets in both portrait and landscape orientations.
Accessibility	Meets accessibility standards; supports screen readers and high contrast modes.

VI- SOFTWARE REQUIREMENT AND SPECIFICATION DOCUMENT (SRS)

This section contains the SRS document. The SRS document provides a clearer overview of the system plus the requirements and its specification aimed at allowing the general public understand what the system is and what it can do.

Software Requirement and Specification Document

Project title: Design and Implementation of a Mobile App for Car Fault Diagnosis.

I – INTRODUCTION

I.1 Purpose

The purpose of this document is to define the software requirements for the Car Fault Diagnostic App. This mobile application is designed to assist vehicle owners and mechanics in identifying and diagnosing faults in vehicles by interfacing with the car's onboard diagnostics (OBD-II) system. The SRS will serve as a guideline for developers, testers, and stakeholders involved in the development and deployment of the application.

I.2 Scope

The Car Fault Diagnostic App provides real-time monitoring and fault detection capabilities by connecting to a vehicle's OBD-II port using Bluetooth or Wi-Fi. It interprets diagnostic trouble codes (DTCs), displays readable fault descriptions, suggests possible causes, and recommends solutions or next steps. The app aims to empower users with actionable insights without requiring advanced automotive knowledge. Features include fault history logs, live engine parameter viewing, maintenance reminders, and a user-friendly dashboard.

I.3 Definitions, Acronyms, and Abbreviations

OBD-II: On-Board Diagnostics version 2

DTC: Diagnostic Trouble Code

ECU: Engine Control Unit

UI: User Interface

App: Mobile application

II– SYSTEM OVERVIEW

Problem statement and proposed solutions

a) Problem Statement

Car owners often struggle to interpret dashboard warning lights and engine sounds, leading to delayed maintenance, unnecessary mechanic visits, and inflated repair costs. Many rely on mechanics even for minor issues due to a lack of accessible, user-friendly diagnostic tools. Existing solutions either require technical expertise, lack multilingual support, or fail to integrate critical features like sound analysis and offline functionality.

b) Proposed Solution

A mobile application that empowers non-technical users to:

- **Diagnose Issues**: Scan dashboard lights/signs and analyze engine sounds via smartphone sensors.
- **Receive Guidance**: Get explanations of warnings, repair tutorials, and maintenance reminders.
- **Connect with Experts**: Locate nearby mechanics/garages and request professional advice.
- **Work Offline**: Access core features (tutorials, diagnostic history) without internet.

System core Features and Purpose

a) Core Features:

- Dashboard light/sign recognition via camera.
- Engine sound analysis using ML models.
- Multilingual support (English, French, local languages).
- Maintenance logs and reminders.
- Explanation of warnings, repair tutorials, and maintenance reminders.
- Geolocation-based mechanic/garage suggestions.
- Offline functionality for critical tasks.

Users: Car owners, drivers, mechanics (secondary stakeholders).

b) Purpose:

- Reduce dependency on mechanics for minor issues.
- Lower maintenance costs and extend vehicle lifespan.
- Educate users about car diagnostics through tutorials and expert advice.
- Provide a scalable, intuitive tool compatible with Android and iOS devices.

II – TOOLS AND TECHNOLOGY TO BE USED

The technology stack is selected based on cross-platform compatibility, scalability, and alignment with the project's functional requirements (like offline support, machine learning, geolocation).

Layer	Technology/Tools	Purpose

Frontend	Flutter (Dart)	Cross-platform UI development for Android and iOS users.
Backend	Firebase (Authentication, Firestore, Cloud Functions)	User authentication, real-time data storage, and serverless backend logic.
Machine Learning	TensorFlow Lite, PyTorch (for model training)	On-device sound recognition (engine noises) and dashboard light detection.
Database	Firestore (NoSQL), Hive/SQFlite (local storage)	Cloud-based and offline storage for diagnostic history and user profiles.
APIs	Node.js + Express (REST API), API's for image and sound recognition for dashboard and engine sound analysis.	Integration with third-party services (e.g, mechanics' databases, maps).
Geolocation	Google Maps API, Flutter Geolocator Plugin	Locate nearby garages and mechanics based on user's location.
Image Processing	OpenCV, TensorFlow (for image recognition)	Analyze dashboard lights/signs via camera input.
Offline Functionality	Hive, SQFlite	Enable app usage without internet (e.g., cached tutorials, basic features).

State Management	Provider/Riverpod	Manage app state efficiently across screens.
CI/CD	GitHub Actions, Codemagic	Automated testing and deployment for Android and iOS.

IV – REQUIREMENTS AND SPECIFICATIONS

This refers to the detailed, precise, and actionable statements that describe how a software system must behave or perform to meet its goals. They translate broad system objectives exact instructions that developers, testers and stakeholders can follow and very.

These requirements are obtained from the clear analysis and classification of our requirement gathering phase or section. The requirement gathered was analyzed to remove ambiguities and identify missing information and later on classified as sited below.

These requirements are divided into two main categories:

Functional Requirements

These define the functions or services the software must provide that is,

- what the system should do
- How it should respond to inputs
- How it should behave in particular situations.

Below we have the requirements with their respective specifications

User Authentication and Authorization	The application should allow users to register and log in securely. Based on the role (car owner, driver, or mechanic), access to features will be granted.
Dashboard Light Recognition	Users can capture images of dashboard indicators with their camera. These images are compared to a database to return matching results with explanations. Users could also scan dashboard signals.

Engine Sound Analysis	Records sound from the engine using the phone's microphone and classifies issues using ML-trained datasets.
Diagnostic Explanation and Repair Guidance	Provides issue-specific explanations and steps to resolve them. Includes videos and images.
Maintenance Log and Reminders	Logs vehicle service history and sets maintenance reminders. Notifies users via app alerts.
Expert Advice and Nearby Mechanic Locator	Displays nearby garages using GPS and allows users to contact mechanics in-app.
Offline Functionality	Downloads and caches essential data like tutorials and history so they are available offline.
Diagnostic History Tracking	Saves results from previous scans along with timestamps and action taken.
Lessons and Tutorials	Offers interactive lessons with multimedia to educate users about car diagnostics.

Non-Functional Requirements

These requirements define the quality attributes of the system that is, how it performs rather than what it does.

Below we have the requirements with their specifications

Requirement	**Specification**
Performance	App should respond within **2 seconds** for scans.
Privacy	Process audio/images **on-device**; minimal data collection.
Usability	Simple UI with **walkthrough guides** for new users.

Reliability	**95% accuracy** in light/sound recognition.
Scalability	Support **10,000+ users** without slowdowns.
Offline Capability	Core features work **without internet**.
Security	Encrypt user data (maintenance logs, location).
Responsiveness	Supports dynamic resizing for phones and tablets in both portrait and landscape orientations.
Accessibility	Meets accessibility standards; supports screen readers and high contrast modes.

VII – KEY USE CASE SCENARIO FOR THE APPLICATION

Use Case 1: Dashboard Warning Light Scanning

Actor: Car Owner

Scenario:

- A driver notices a warning light and opens the app to scan it using the camera.
- App detects issue.

Flow:

1. App identifies the light using offline computer vision (pre-trained model).
2. Displays explanation: “Problem with engine – check the spark plugs or the air filter.”
3. Suggests urgency level (“High – Visit mechanic within 48 hours”).
4. Links to a video tutorial by a mechanic on cleaning air filters.

Use Case 2: Engine Sound Analysis

Actor: Car Owner

Scenario:

- User records engine noise while idling.
- App detects a knocking sound (indicative of engine misfire).

Flow:

1. App filters background noise (street vendors, traffic) using spectral analysis.
2. Matches audio to preloaded fault patterns (trained on common engine sounds).
3. Recommends e.g.: "Problem with spark plugs – Buy spare parts from [local vendor] at mile 17."
4. Provides offline access to step-by-step replacement guides.

Use Case 3: Video Tutorial Access

Actor: Car Owner

Scenario: User gets a diagnostic result and wants help understanding the issue.

Outcome: The app provides YouTube links or offline embedded videos.

.

VIII – REFERENCES

- ISO15031: Communication between vehicle and external diagnostic tools
- SAEJ1979: Standard for OBD-II diagnostic services
- IEEE830-1998: Recommended Practice for Software Requirements Specifications

VII – VERICATION AND VALIDATION OF REQUIREMENTS

1) Functional Requirement Analysis

i. User Authentication and Authorization

- **Technique Used**: Inspection, Peer Review
- **Result**: Requirement is complete, necessary, feasible, and verifiable. Login and role-based access control are standard implementations. No ambiguity found.

ii. Dashboard Light Recognition

- **Technique Used**: Walkthrough, Checklist
- **Result**: Verified as necessary and correct. Completeness was flagged to ensure the database of symbols includes local variants. Improvement suggested in database coverage.

iii. Engine Sound Analysis

- **Technique Used**: Peer Review, Feasibility Check
- **Result**: Found feasible using pre-trained ML models. Completeness confirmed, effectiveness dependent on training data quality.

iv. Diagnostic Explanation and Repair Guidance

- **Technique Used**: Walkthrough, Inspection
- **Result**: Complete and correct. Stakeholders validated usefulness and clarity. Verifiable via content tests.

v. Maintenance Log and Reminders

- **Technique Used**: Checklist, Peer Review

- **Result**: Verified as consistent and complete. Efficiency verified through time-based reminders and test cases.

vi. Expert Advice and Nearby Mechanic Locator

- **Technique Used**: Feasibility Analysis, Walkthrough
- **Result**: Location-based functions confirmed feasible. Stakeholder walkthrough validated necessity.

vii. Offline Functionality

- **Technique Used**: Checklist, Peer Review
- **Result**: Offline accessibility validated as effective and necessary for rural users. Verifiable through simulation tests.

viii. Multilingual Support

- **Technique Used**: Inspection, Checklist
- **Result**: Requirement is necessary and effective. Completeness depends on language translation coverage.

ix. Diagnostic History Tracking

- **Technique Used**: Walkthrough, Peer Review
- **Result**: Confirmed as verifiable and feasible. Completeness validated for user scenarios.

x. Lessons and Tutorials

- **Technique Used**: Walkthrough, Inspection
- **Result**: Tutorials validated as effective and complete. Verification possible through usability testing.

SUMMARY POINTS

Completeness	Most requirements are complete. A few (e.g., dashboard light database, multilingual coverage) require expansion to ensure comprehensive coverage across models and regions.

Feasibility	All requirements were confirmed feasible using current mobile technologies (e.g., ML models, GPS, local caching). No critical implementation barriers were identified.
Necessity	Every requirement was validated as necessary based on stakeholder input. They address real user pain points like lack of knowledge, high repair costs, and limited offline tools.
Verifiability	All requirements can be verified through functional testing, simulated use cases, or by measuring performance criteria. Each has clear test paths.
Effectiveness	Requirements are effective in solving the identified problem. Particularly strong are sound analysis and offline functionality. Some effectiveness (e.g., sound detection accuracy) will depend on data quality during implementation.

2)Non-Functional Requirement verification and validation

Quality Attribute	**Result Summary**
Completeness	All non-functional requirements are well-defined. They address performance, usability, security, and accessibility. No major omissions were found.
Feasibility	Every non-functional requirement is feasible with current mobile development frameworks (e.g., Flutter, React Native, Android SDK). Especially achievable are responsiveness and encryption standards.
Necessity	Each requirement serves a critical role in ensuring the app is usable, scalable, and secure. Stakeholders emphasized usability and offline reliability as essential for adoption.
Verifiability	All are testable using system benchmarks, usability tests, accessibility audits, and security checks. Each can be validated during system testing phases.

Effectiveness	Non-functional requirements significantly contribute to user satisfaction, trust, and engagement. Effective UI, fast response times, and offline access were seen as major usability enhancers.

VIII – CONCLUSION

This section of the report outlines the overall system design, specific functional and non-functional requirements, user interface guidelines, external interface definitions, and assumptions for the Car Fault Diagnostic App. It is intended to ensure all stakeholders share a common understanding of the application's objectives and operational requirements