UNIVERSITY OF BUEA
FACULTY OF
ENGINEERING AND
TECHNOLOGY

REPUBLIC OF CAMEROON
PEACE-WORK-FATHRTLAND

COURSE TITLE: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

COURSE CODE: CEF440

COURSE INSTRUCTOR: Dr. VALERY NKEMENI

GROUP NUMBER: 18

TASK 3: REQUIREMENT ANALYSIS

Group 18 Members

S/N	NAME	MATRICULE
1	Sindze Djam Alan Wilfried	FE20A105
2	Enjema Ndive Peace	FE22A203
3	Njamutoh Shalom Brenda Tasah	FE22A271
4	Ngwinkem Ketty Nerita	FE22A269
5	Tabi Atem Rogan Essembion	FE22A300

04/05/2025

A) Review and analysis of the requirements gathered (Completeness, Clarity, Technical Feasibility, Dependency Relationships)

1) Completeness:

- The report demonstrates a good level of completeness in gathering requirements.
- It covers requirements from various stakeholders, including car owners/drivers, mechanics and lecturers, using multiple gathering techniques like surveys, interviews, brainstorming and reverse engineering.
- The report identifies both functional (e.g., app features) and non-functional requirements (e.g., usability, performance).
- It also addresses data requirements and cleaning, which is crucial for the AI-driven aspects of the app.
- However, the report could benefit from even more detailed specifications of certain requirements. For example, the specific types of car faults to be diagnosed could be listed exhaustively.

2) Clarity:

- Most of the requirements are stated clearly and are easy to understand, especially those gathered through surveys and presented with visual aids.
- The use of direct quotes from interviews also enhances clarity.
- However, some requirements could be more specific. For instance, "The app should generate revenue" is a bit vague. Specifying how (e.g., ads, premium subscriptions) would add clarity.

3) Technical Feasibility:

- The report demonstrates a good consideration of technical feasibility.
- It acknowledges the use of OBD-II ports, AI-driven diagnostics, and mobile app development, all of which are technically feasible.
- The report also considers the feasibility of data collection and cleaning.

- The choice of technologies like CNNs and LSTM for image and audio analysis suggests an awareness of current technical capabilities.
- Potential challenges, such as user reluctance are also acknowledged and mitigation strategies are proposed.
- A deeper dive into the feasibility of accurately diagnosing a wide range of car faults with AI, especially within the constraints of a mobile app, would strengthen this analysis.

4) Dependency Relationships:

- The report implicitly suggests some dependency relationships. For example, the accuracy of the fault diagnosis (a functional requirement) is dependent on the quality and quantity of the data collected and the effectiveness of the AI models.
- The ability to provide cost estimates for repairs is dependent on having data about typical repair costs for various faults.
- The user interface design (UI/UX) is dependent on the functional requirements of the app, such as the need to display diagnostic information and guide users through troubleshooting steps.
- However, the report does not explicitly map out these dependencies. A dependency matrix or diagram would make these relationships clearer and aid in project planning and management.

B) Identification of inconsistencies, ambiguities and missing information

1) Inconsistencies:

- **Conflicting Survey Results:**
  - There are two different survey questions about ignoring warning lights/strange noises, with conflicting results. One shows 65% of respondents ignore warnings, while another shows 51.3% ignore them. The sample size also changes from 40 to 39.
- **Data Cleaning vs. User Reluctance:**
  - The report has sections for "Data Cleaning" and "User Reluctance Assessment." It's unclear if "User Reluctance" issues were addressed during the data cleaning process.
- **Sample Population:**
  - The survey was distributed through social media groups and to car owners. The survey was mainly carried out on taxi drivers and private car owners. It's not clear if there is a difference between "car owners" and "taxi drivers and private car owners".

2) Ambiguities:

- **"Major feedback" Clarity:**
  - The "Major feedback" sections after survey results are vague. It's unclear how "major" is defined (most frequent, most important, etc.). Also, the format is inconsistent (sometimes bullet points, sometimes not).
- **App Feature Prioritization:**
  - The brainstorming session categorized features as core, important, and future. The report doesn't specify which features fall into which category.
- **"Observation" Detail:**
  - The description of the "Observation" data gathering method is brief. It doesn't specify the number of observations, the duration, or specific behaviors observed.

- **"Manual Review" Explanation:**
  - The report mentions "manual review" for data cleaning but doesn't detail the process. It's unclear what criteria were used.

3) Missing Information:

- **Survey Questions:**
  - While the report presents results from the surveys, it doesn't include the exact wording of all the survey questions. This makes it difficult to fully interpret the responses.
- **Interview Questions:**
  - Similar to the surveys, the interview questions are not included. This limits understanding of the context of the feedback from mechanics, car owners, and lecturers.
- **Brainstorming Process:**
  - The report mentions brainstorming sessions but provides limited detail about how the sessions were structured, the number of participants, or the methods used to generate ideas.
- **Reverse Engineering Analysis:**
  - While it mentions reverse engineering existing apps, it lacks specifics on *how* these apps were analyzed (features, UI, etc.) and what specific gaps were identified.
- **Data Storage and Management:**
  - The report doesn't mention how the collected data will be stored, managed, and secured.
- **Ethical Considerations:**
  - There's no discussion of ethical considerations related to data collection and user privacy.

C) Prioritization of requirements based on importance and feasibility

Below is a simple High, Medium, and Low scale for both importance and feasibility.

1) Prioritization of Requirements:

S/N	Requirements	Importance	Feasibility	Overall Priority	Justification
1	Scan dashboard indicators using the phone's camera and use computer vision to detect and interpret warning lights.	High	Medium	High	This is a core feature directly addressing the problem statement of interpreting warning lights. While computer vision can be complex, advancements in mobile ML make it moderately feasible.
2	Record engine sound and use audio classification algorithms to identify potential mechanical issues	High	Medium	High	This directly addresses the problem of diagnosing issues from engine sounds. Audio classification is a well-researched area, but real world engine noise can be challenging, impacting feasibility.
3	Provide explanation of each warning symbol, its possible cause, and urgency level.	High	High	High	This is crucial for user understanding and actionable advice. A database of warning lights and their meanings is relatively feasible to compile and integrate.

4	Present users with a list of likely issues, possible fixes, and maintenance tips for each identified fault.	High	Medium	High	This provides direct value to the user. Compiling a Comprehensive database of issues and fixes will require effort but is achievable
5	Use of machine learning or deep learning models trained on sound datasets to identify abnormalities in engine noise.	High	Medium	High	This is key to the audio analysis functionality. Training robust models requires significant data and computational resources but is fundamental to accurate sound-based diagnosis
6	Match recorded audio to known fault patterns (e.g., knocking, squealing, hissing) and provide a diagnosis	High	Medium	High	This is the practical application of the audio analysis. Creating and matching against a library of sound patterns is crucial.
7	Provide YouTube or embedded video links from verified car experts showing step-by-step repair or maintenance processes related to the detected issue.	Medium	High	Medium	Adds significant user value by providing visual guidance. Integrating with YouTube or embedding videos is technically feasible, but curating and verifying the videos is important.
8	Provides basic offline functionality (e.g., light recognition and	Medium	Medium	Medium	Enhances usability in areas with limited connectivity. Preloading sufficient data for

	sound analysis using preloaded data)				accurate offline analysis might be challenging in terms of app size and model complexity
9	Has online features for accessing updated fault databases and video content.	Medium	High	Medium	Allows for continuous improvement and access to a wider range of information. Implementing online data fetching and video streaming is technically straightforward.
10	Diagnose car problems using OBD-II data	High	Medium	High	Directly addresses the need for accurate fault diagnosis, as highlighted by mechanics and car owners. Feasible with readily available OBD-II Bluetooth adapters
11	User-friendly UI with icons, one-click scan, and optional voice narration	High	Medium	High	Ensures accessibility and ease of use for non-technical users, which is a key concern.
12	Provide cost estimates for repairs	High	Medium	High	Helps users avoid being overcharged by mechanics, a significant pain point.
13	Provide a car maintenance history	Medium	Medium	Medium	Helps users track past repairs and maintenance, but requires data storage and management
14	Provide a possibility	Medium	High	Medium	Helps connect users

	for on-site car fault detection				to trustworthy mechanics using their phone numbers to providing guides for on-site repairs for users.
15	Generate revenue (ads or premium version)	Medium	Medium	Medium	Important for sustainability but is secondary to core functionality.
16	Predict the fault of the car before the user can reach the mechanic	Medium	Medium	Medium	Helps users know the fault of their car before reaching the mechanic
17	Send notifications concerning car	Medium	Medium	Medium	___
18	Making the app to be in French and English	Medium	Low	Low	___

2) Explanation of Prioritization:

● **High Priority (High Importance & Medium/High Feasibility):** These are the core features that directly address the problem statement and are reasonably achievable within the scope of a project. Focusing on these first will deliver the most value to the user.

● **Medium Priority (Medium Importance or Lower Feasibility):** These features add significant value but might be more complex to implement initially or are not as fundamental as the high-priority items. They can be considered for later iterations or if resources allow.

● **Low Priority (Low Importance or Very Low Feasibility):** There are no explicitly low-priority items in the given requirements, as they all contribute to the application's functionality. However, if we had requirements like advanced sensor data integration(requiring OBD-II adapters) or social sharing features, those might fall into a lower priority due to increased complexity or less direct impact on the core problem.

D) Classification requirements (Functional and Non-Functional)

1) Functional Requirements (FRs):

These are the features and services the system must provide:

- **Dashboard Light Recognition:**
  - ➢ The app shall allow users to scan dashboard indicators using their phone’s camera.
  - ➢ The app shall use computer vision to detect and interpret warning lights.
  - ➢ The app shall detect multiple active warning lights simultaneously.
- **Engine Sound Analysis:**
  - ➢ The app shall allow users to record engine sounds.
  - ➢ The app shall use audio classification to identify abnormal sounds.
  - ➢ The app shall match recorded audio to known fault patterns (e.g., knocking, squealing, hissing).
- **Diagnostics and Feedback:**
  - ➢ The app shall provide explanations of detected warning lights.
  - ➢ The app shall list possible causes and urgency levels for detected faults.
  - ➢ The app shall estimate the urgency of required maintenance.
  - ➢ The app shall provide suggestions for possible fixes and maintenance tips.
  - ➢ The app shall recommend actions based on detected faults.

- **Multimedia Integration:**
  - The app shall provide links to YouTube or embedded video tutorials from verified car maintenance experts.
- **Offline and Online Functionality:**
  - The app shall support basic offline functionality for light recognition and sound analysis using preloaded data.
  - The app shall support online updates for fault databases and video content.
- **Connect to cars OBD-II Port:**
  - The app shall connect to cars OBD-II port via bluetooth adapter so it can read diagnostic trouble codes
- Provide numbers for trustworthy mechanics and electricians in emergencies
- Provide a car maintenance history
- Provide cost estimates for fixing faults
- Diagnose car problems using OBD-II data
- The app should generate revenue by ads or premium version

2) Non-Functional Requirements (NFRs):

These are the system's quality attributes and constraints:

- **Usability:**
  - The app should be user-friendly and easy to navigate for non-technical car owners.
- **Performance:**
  - The app should process and analyze inputs (images and sounds) within a few seconds.

- The app should work efficiently on typical Android/iOS smartphones without requiring external hardware.

- **Accuracy:**
  - The computer vision and audio analysis models should have high accuracy in detecting faults.

- **Reliability:**
  - The app should reliably function in various lighting and noise conditions.
  - The app should handle errors gracefully (e.g., when camera or microphone input is unclear).

- **Maintainability:**
  - The system should allow easy updates to the fault recognition database and model retraining as needed.

- **Security and Privacy:**
  - The app should protect user data, especially any personal or diagnostic history.
  - If cloud services are used, the data transmission must be secure (e.g., encrypted).

- **Portability:**
  - The app should be compatible across major mobile platforms (Android and iOS).

- **Scalability:**
  - The architecture should support future additions, such as integration with OBD-II devices or remote mechanic consultations.

E) Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to outline the functional and non-functional requirements for a mobile
application that assists car owners in diagnosing vehicle issues using dashboard light recognition and engine
sound analysis through artificial intelligence (AI) and mobile technologies.

1.2 Scope

This application will:

- Scan and interpret dashboard warning lights using the smartphone camera and computer vision.
- Record and analyze engine sounds to detect mechanical faults using audio recognition.
- Provide explanations, urgency levels, likely causes, and repair suggestions for identified issues.
- Offer links to instructional videos from car maintenance experts.
- Function offline for basic diagnostics and online for accessing updated databases and multimedia content.

1.3 Definitions, Acronyms, and Abbreviations

- AI - Artificial Intelligence
- ML - Machine Learning
- CV - Computer Vision
- OBD - On-Board Diagnostics
- UI - User Interface

1.4 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, testers, and stakeholders interested in understanding the application's functionality and requirements. It provides a basis for design, implementation,
and testing.

1.5 Overview

This document is organized into sections covering overall description, functional requirements, non-functional
requirements, and system models.

2. Overall Description

2.1 Product Perspective

The application is standalone and designed for Android and iOS platforms. It utilizes the device's camera and microphone for input and AI models for processing.

2.2 Product Functions

- Image processing for dashboard warning light recognition
- Audio processing for engine sound fault detection
- Display of diagnostic results and recommendations
- Integration of video-based repair tutorials
- Offline and online support features

2.3 User Classes and Characteristics

- Regular car owners: Non-technical users seeking easy-to-use diagnostic tools.
- Mechanics (optional): May use the app for quick diagnostics.
- Developers/Admins: Responsible for maintaining model updates and database content.

2.4 Operating Environment

- Android (8.0+) and iOS (12+)
- Uses phone camera and microphone
- Internet connection for updates and video content

2.5 Design and Implementation Constraints

- Must work without external diagnostic tools.
- Must support real-time processing with minimal latency.

2.6 User Documentation

- In-app tutorials and help section
- FAQs and user guide (PDF/online)

3. Functional Requirements

FR1: Dashboard Light Detection

- The app shall capture dashboard images using the phone camera.
- The app shall detect and recognize multiple dashboard warning lights simultaneously.
- The app shall identify the meaning and urgency of each warning light.

FR2: Engine Sound Analysis

- The app shall record engine sound via microphone.
- The app shall analyze the audio for abnormal patterns using trained ML models.
- The app shall match sounds to common mechanical faults.

FR3: Fault Diagnosis and Recommendations

- The app shall display possible causes and urgency levels for detected faults.
- The app shall recommend repair actions and maintenance tips.

FR4: Multimedia Support

- The app shall provide embedded video links from verified car experts.
- The app shall suggest videos based on the specific identified issue.

FR5: Offline Functionality

- The app shall offer offline support for basic diagnostics using preloaded data.

FR6: Online Features

- The app shall fetch updated fault databases when connected to the internet.
- The app shall access and stream video tutorials online.

FR7: Diagnose car problems using OBD-II data

- The app shall be able to interpret the OBD-II code to the user telling him/her the fault with their car

FR8: Send notifications concerning car

- The app shall be able to send notifications to users concerning their car like when to change the engine oil, or when to go for maintenance

FR9: Predict the fault of the car before the user can reach the mechanic

-The app should be able to Predict the fault of the car before the user can reach the mechanic

FR10: Generate revenue

- This will be done by Ads or premium version

FR11: Provide cost estimates for repairs

- This is to minimize the level of extortion

FR12: User-friendly UI with icons, one-click scan, and optional voice narration

FR13: Recommend reliable and trustworthy mechanics

- The app should recommend reliable mechanics by giving their contact information

4. Non-Functional Requirements

NFR1: Usability

- The UI shall be intuitive and require minimal technical knowledge.
- Visual and audio feedback should guide user interaction.

NFR2: Performance

- The app shall process images and audio within 5 seconds.
- ML models shall be optimized for mobile performance.

NFR3: Accuracy

- Dashboard and audio recognition must achieve at least 85% accuracy in standard test environments.

NFR4: Reliability

- The app shall function under various lighting and sound conditions.
- It shall handle poor input gracefully (e.g., unclear audio/image).

NFR5: Portability

- The app shall run on Android and iOS platforms.
- Codebase should be modular for future upgrades.

NFR6: Maintainability

- The system shall support updates to models and databases without requiring app re-installation.

NFR7: Security

- The app shall secure all user data using local encryption.
- Internet communication shall use HTTPS.

5. Appendices

A. Datasets:

- Audio and image datasets used for model training (e.g., UrbanSound8K, custom car dashboard icons).

B. Tools and Libraries:

- TensorFlow Lite, OpenCV, Librosa, Flutter/React Native.

C. Glossary:

- Knocking: Engine sound caused by premature combustion.
- Squealing: Commonly due to worn belts.
- Hissing: Usually indicates a vacuum leak.

F) Verification of Requirements with Stakeholders.

Here, we meet some specific car owners who were very much interested in out mobile application. We were opportune to share our requirements with them one by one asking them if each requirement is:

- Clear and unambiguous
- "Is it complete?"
- "Is it feasible?"

And as for if it aligns with the stakeholders goals, we as a group were able to verify this using the pre-document that was given to us by our lecturer.

G) Conclusion

The document provides a comprehensive requirements analysis and specification for a mobile app that helps car owners diagnose vehicle issues using AI. It enables users to scan dashboard warning lights and analyze engine sounds with their smartphone. The analysis ensures completeness, feasibility, and clarity while identifying gaps and refining initial data. Requirements are prioritized and categorized into functional (like fault diagnosis and media support) and non-functional (such as usability and performance). The Software Requirements Specification (SRS) formalizes the app's scope, audience, functions, and constraints. Appendices include relevant datasets, tools, and a glossary. Overall, it serves as a detailed blueprint for app development.