REPUBLIC OF CAMEROON

Peace-Work-Fatherland

MINISTRY OF HIGHER EDUCATION

UNIVERSITY OF BUEA

REPUBLIQUE DU CAMEROUN

Paix-Travail-Patrie

MINISTRE DE L'ENSEGNEMENT SUPERIEUR

UNIVERSITE DE BUEA

UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

REQUIREMENT ANALYSIS OF CAR DIAGNOSIS MOBILE APPLICATION

PRESENTED BY GROUP 25

COURSE INSTRUCTOR: **Dr NKEMENI VALERY**

COURSE CODE/TITLE: **CEF 440/ INTERNET PROGRAMMING(J2EE) AND MOBILE PROGRAMMING.**

ACADEMIC YEAR 2024/2025

Table of Contents

1 Introduction

Requirement analysis is a systematic process used to identify, gather, examine, and document the needs and expectations of stakeholders for a proposed system or application. It involves transforming vague, high-level business ideas into detailed, structured, and actionable functional and non-functional requirements that guide the design, development, and validation phases of a project. This process ensures that the final product aligns with user goals, technical feasibility, legal constraints, and organizational objectives. It typically includes stakeholder interviews, use case modeling, feasibility studies, and prioritization of requirements.

2 Overview

This phase focuses on identifying and analyzing the needs, expectations, and challenges of target users in diagnosing vehicle faults. The goal is to convert real-world pain points into functional and non-functional requirements that guide system design.

3 Problem Statement

Car owners often struggle to interpret dashboard warning lights or identify unusual engine sounds. Many lack immediate access to a mechanic or do not understand whether the issue is urgent. Current solutions are limited in accuracy, accessibility, or offline usability.

4 Objectives of Requirement Analysis

- Understand user expectations from a mobile fault-diagnosis tool.
- Identify high-priority features such as dashboard light recognition, engine sound analysis, and offline usage.

- ❖ Document usability, performance, and integration requirements.
- ❖ Recognize system constraints and technical dependencies.

5 Importance of Requirements Analysis

Ensures Clarity: Identifies and resolves vague or conflicting requirements early to avoid downstream misinterpretation.

Promotes Feasibility: Helps determine if requirements can be implemented within technical, budgetary, and timeline constraints.

Facilitates Prioritization: Enables classification of critical vs optional features to guide development phases.

Reduces Risks: Minimizes the risk of scope creep, rework, and project failure by confirming requirements are valid and agreed upon.

Supports Traceability: Forms the basis for traceable links between requirements, design decisions, and testing activities.

6 Review and Analysis of Requirements Gathered

The review and analysis of gathered requirements is the process of examining all collected user needs and stakeholder expectations to ensure they are complete, consistent, feasible, clear, and aligned with the project's objectives. This step involves validating the relevance and accuracy of each requirement, resolving ambiguities, detecting contradictions, and classifying them into functional, non-functional, and technical categories.

6.1 Strengths of Requirements Gathered

The survey provides a comprehensive foundation for understanding user needs and preferences for a car fault diagnosis app. Key strengths include:

- **Clear user pain points**: The survey effectively confirms the need for the app, with most respondents indicating they have encountered dashboard warning lights they didn't understand
- **Well-defined core functionality**: The survey successfully identifies the primary features users want (dashboard light scanning, engine sound diagnosis, problem urgency alerts)
- **Strong user interest**: Responses show significant enthusiasm for AI-based car diagnostics, with most users willing to use such technology
- **Balanced approach to guidance**: The survey captures varied preferences for receiving help (text, video, voice), allowing for an inclusive design
- **Privacy considerations**: The survey appropriately addresses privacy concerns around microphone access
- **Feature prioritization indicators**: User responses clearly indicate which features are most valued (like real-time alerts and issue history)

6.2 Completeness Assessment

While the survey effectively covers key areas, some additional details would enhance completeness:

- Target devices and operating systems specifications
- Technical parameters for sound recognition accuracy
- Data storage requirements and security measures
- Performance requirements under different conditions

6.3 Clarity Assessment

The survey questions are generally well-structured and clear, providing actionable insights for development. Particularly clear aspects include:

- ❖ User behavior when encountering car problems (check themselves vs. go to mechanic)
- ❖ Preferences for assistance methods (text/video/voice)
- ❖ Comfort level with AI-based diagnostics
- ❖ Willingness to contribute to app improvement

6.4 Technical Feasibility

Based on current technology capabilities, the proposed features demonstrate strong feasibility:

Requirement	**Technical Feasibility**	**Supporting Technologies**
Dashboard light recognition	High	Existing image recognition APIs, machine learning frameworks
Engine sound diagnosis	Medium-High	Audio analysis algorithms, pattern matching techniques
Real-time alerts while driving	High	Background processing, notification systems
Mechanic/towing service locator	Very High	Maps API integration, location services

Works without internet	Medium-High	Local databases, efficient storage algorithms

6.5 Dependency Relationships

The survey effectively reveals important feature interdependencies:

Primary Dependencies:

- Engine sound diagnosis → Microphone access permissions → User comfort with privacy implications
- Real-time alerts → Issue severity assessment → User preference for notification style
- Offline functionality → Local data storage → App size management

Positive Reinforcement Relationships:

- History tracking enhances the value of diagnostic features
- Real-time monitoring complements dashboard warning detection
- Video tutorials enhance the usefulness of diagnostic results

7 Identification of Inconsistent, Ambiguity and Incomplete Data

- Inconsistent data is data that contradicts itself, data points or records that conflict with each other and also lack of uniformity.
- Ambiguous data is data that lacks clarity with unclear definitions. Such data leads to misinterpretations and inaccurate analysis.
- Incomplete data refers to information (survey forms) with missing values lacking specific entries.

Partial information that is formed providing only part of the required data

7.1 Identification

Some data in both the Google forms and survey forms were inconsistent, ambiguous and incomplete in this section, incomplete data, inconsistent and ambiguous data are identified and carefully analyzed

From such questions such as: have you ever seen a dashboard light and not know what it meant?

The response I usually ignored them is identified as ambiguous and made us to include a popup message in the app that reminds users not to ignore dashboard warning light

Q2, the question will you rather fix small issues or straight to a mechanic, The response Depends on the issue is ambiguous, it is analyzed that small faults are fixed handled without taking the car to the mechanic.

Incomplete data as seen in the response of the following two questions,

Do you think you can tell if something is wrong with your car just by listening to the engine sound most of the responses said yes.

The next question demands if you said Yes, then how do you do it , many of those who said yes above did not respond.

Inconsistent data is also observed from the above two questions where the second question only demands you to answer if your previous response is Yes, but some people with No as previous answer still responded and such were discarded

7.2 Considerations and features noted from the identification of inconsistency, ambiguity and in complete data

- The app should periodically notify users to check their dashboard light, this is drawn from the response some users ignore dashboard lights until car break downs. If car dashboard lights are checked and mitigation taken with respect to the dashboard lights it prevents car breakdowns.
- The app should request for permission for device feature such as microphone before making use of them.
- The app should notify users of nearby mechanics, since user take their car to mechanics for major car issues.

8 Requirement Prioritization

Requirement prioritization involves ranking the gathered requirements to determine which ones should be implemented first, based on:

- Business value
- Technical feasibility
- User needs
- Risk
- Time sensitivity

8.1 Prioritization Techniques

MoSCoW Method

Classifies requirements into:

- M – Must Have: Essential for MVP (Minimum Viable Product). The app cannot function without these.
- S – Should Have: Important but not vital. Can be scheduled for future updates.
- C – Could Have: Nice-to-haves. Only if time and resources permit.
- W – Won't Have (Now): Not needed for this phase; possible for future.

8.2 Factors to Consider When Prioritizing

- User Impact: How important is it to users?
- Technical Complexity: Can it be implemented with available resources?
- Dependencies: Does it rely on another feature?
- Cost vs Value: Is the value worth the development effort?
- Risk Level: What's the impact if it fails or is not implemented?

8.3 Prioritizing Key Functional Requirements

The main priority requirements are given below

Requirement	Priority	Reasoning
Scan dashboard indicators via camera	Must	Core functionality, quick MVP delivery
Audio-based fault detection	Must Have	Adds intelligence, highly valuable to users
Show repair suggestions	Must Have	Increases usefulness, not essential for first release
Save and export diagnostic history	Must Have	Useful, but not critical to basic functionality
Video tutorials integration	Must Have	Value-added feature; needs stable internet
Real-time update of fault database	Must Have	Adds maintenance overhead; defer to future versions

Other requirements and their priority levels are given below

Requirement	Type	Priority	Justification
Scan dashboard indicators using camera	Functional	High	Core feature for visual diagnostics
Detect multiple warning lights at once	Functional	High	Common real-world scenario; improves diagnostic accuracy
Provide explanation of warning symbols	Provide explanation of warning symbols	High	Enables users to understand warning indicators clearly
Record engine sound using microphone	Functional	High	Required input for audio diagnostics

Analyze engine sounds using ML models	Functional	High	Essential for audio-based fault detection
Match recorded audio to known fault patterns	Functional	Functional	Required for generating meaningful diagnosis
Suggest possible fixes and maintenance tips	Functional	Medium	Valuable for self-service repair, but not required for core detection
Link to YouTube or embedded video tutorials	Functional	Low	Enhances experience; depends on internet and external APIs
Provide basic offline support for light and sound analysis	Functional	Medium	Important for accessibility; enables use without internet
Store fault history and allow users to view previous diagnoses	Functional	Low	Adds convenience; not essential for MVP
Sync with updated fault databases online	Functional	Low	Future enhancement; backend complexity involved
Allow export or sharing of diagnostic results	Functional	Low	Future enhancement; backend complexity involved
Simple and intuitive UI with visual feedback	Non-Functional	Medium	Improves UX but not mandatory for core function
Show urgency level of fault	Functional	Functional	Helps users prioritize actions but not critical for basic diagnosis

By categorizing requirements into high, medium, and low priority, the development team (we) can focus on delivering the most impactful features first. This approach ensures that the core functionalities are robust and user-centric, setting a solid foundation for future enhancements. Regular reviews with stakeholders will help adjust priorities based on user feedback and changing market conditions.

9 Software Requirement Specification Document (SRS)

A Software Requirements Specification (SRS) is a comprehensive, structured document that outlines the complete set of functional and non-functional requirements for a software system. It serves as a formal agreement between stakeholders including clients, users, and the development team defining what the software will do, how it is expected to perform, and the constraints under which it must operate.

An SRS typically includes the system's purpose, scope, intended audience, overall functionality, system interfaces, user interactions, and technical specifications. It acts as a foundation for system design, development, testing, and validation, ensuring alignment across all stages of the software development lifecycle.

9.1 Functional Requirements

Core Diagnostic Features

1. ***Dashboard Warning Light Recognition***

   - ❖ System shall identify dashboard warning lights from user-uploaded images
   - ❖ System shall provide detailed explanation of what each warning light means
   - ❖ System shall classify severity level (urgent, moderate, informational)

2. ***Engine Sound Analysis***

   - ❖ System shall analyze recorded engine/vehicle sounds to identify potential issues
   - ❖ System shall differentiate between normal operating sounds and problematic sounds
   - ❖ System shall categorize sounds by vehicle component (engine, brakes, transmission, etc.)

3. ***Problem Diagnosis***

   - ❖ System shall provide preliminary diagnosis based on combined input data
   - ❖ System shall estimate repair urgency (immediate attention, schedule service, monitor)

- System shall link symptoms to potential causes with probability ratings

User Management & Vehicle Information

1. ***User Profile Management***

   - Users shall be able to create and manage personal accounts
   - System shall store vehicle details (make, model, year, VIN)
   - System shall support multiple vehicles per user account

2. ***Vehicle History***

   - System shall maintain history of diagnosed issues for each vehicle
   - System shall track maintenance records and service intervals
   - System shall generate maintenance reminders based on mileage/time

Support Features

1. ***Mechanic Connectivity***

   - System shall provide directory of nearby mechanics/service centers
   - System shall enable sharing of diagnostic results with mechanics
   - System shall support optional direct mechanic consultation feature

2. ***Knowledge Base***

   - System shall include searchable database of common car problems
   - System shall provide DIY repair guides for simple issues
   - System shall offer preventative maintenance recommendations

Offline Functionality

1. ***Core Offline Features***

   - Essential diagnostic features shall work without internet connection
   - System shall store critical reference data for offline use
   - System shall synchronize offline activity when connection resumes

9.2 Non-Functional Requirements

Performance Requirements

1. ***Response Time***

   - ❖ Dashboard light recognition shall complete within 3 seconds
   - ❖ Sound analysis shall complete within 10 seconds
   - ❖ UI interactions shall respond within 0.5 seconds

2. ***Accuracy***

   - ❖ Dashboard light recognition shall have >95% accuracy
   - ❖ Sound analysis shall have >85% accuracy for common issues
   - ❖ System shall clearly indicate confidence levels for all diagnoses

3. ***Resource Utilization***

   - ❖ Application shall use <200MB of storage space (excluding user data)
   - ❖ Application shall consume <15% of battery per hour during active use
   - ❖ Application shall operate effectively on devices with 2GB RAM minimum

Usability Requirements

1. ***User Interface***

   - ❖ Interface shall be navigable by users with minimal technical knowledge
   - ❖ Application shall use clear, non-technical language with optional detailed information
   - ❖ System shall provide guided workflows for diagnosis processes

2. ***Accessibility***

   - ❖ Application shall comply with WCAG 2.1 AA standards
   - ❖ Interface shall support screen readers and voice commands
   - ❖ System shall provide alternative text identification methods for color-blind users

3. ***User Experience***

- First-time user shall complete first diagnosis within 5 minutes
- System shall require no more than 3 steps to initiate any primary function
- Application shall provide clear progress indicators for all processes

Reliability Requirements

1. ***Availability***

- System shall be available 99.5% of the time
- Planned maintenance shall not exceed 4 hours per month
- System shall provide degraded service mode during server outages

2. ***Fault Tolerance***

- Application shall recover from crashes without data loss
- System shall validate all user inputs to prevent errors
- Application shall maintain local backup of critical user data

Security Requirements

1. ***Data Protection***

- All user data shall be encrypted at rest and in transit
- Personal user information shall not be shared without explicit consent
- System shall comply with relevant data protection regulations

2. ***Authentication***

- System shall enforce strong password policies
- Application shall support biometric authentication where available
- System shall implement secure session management

Compatibility Requirements

1. ***Platform Support***

- Application shall function on iOS 14+ and Android 10+
- System shall adapt UI for different screen sizes (phone and tablet)
- Application shall support landscape and portrait orientations

2. ***Integration***

- System shall support integration with popular vehicle maintenance apps
- System shall enable export of diagnostic data in standard formats
- System shall provide API for authorized third-party integrations

Scalability Requirements

1. ***User Load***

- System shall support concurrent usage by up to 100,000 users
- Database shall scale to accommodate 1M+ vehicle profiles
- System performance shall degrade gracefully under peak loads

Legal and Compliance Requirements

1. ***Disclaimer***

- System shall display clear disclaimer about diagnostic limitations
- System shall require acknowledgment that app does not replace professional service
- Terms of service shall limit liability for misdiagnosis

2. ***Regulatory Compliance***

- System shall comply with automotive data standards
- Application shall adhere to app store policies for both iOS and Android
- System shall implement appropriate accessibility features per local regulations

This requirements analysis provides a foundation for developing a comprehensive automotive diagnostic mobile application that addresses the identified problem while meeting the outlined objectives.

10 Validation of Requirements with Stakeholders

Validation of requirements with stakeholders is the process of ensuring that all identified requirements accurately reflect stakeholder needs, expectations, and project goals. This involves verifying that the documented requirements are complete, feasible, and agreed upon by all parties, thereby reducing the likelihood of misunderstandings and costly revisions later in the development process.

Key Activities:

- **Stakeholder Review Meetings:**
  Formal review sessions were conducted with key stakeholders (e.g., car owners, mechanics, development team, project sponsors) to present gathered requirements.
- **Requirement Walkthroughs:**
  Requirements were discussed in detail to confirm its relevance, clarify ambiguities, and ensure alignment with user needs and technical feasibility.
- **Feedback Collection:**
  Inputs from stakeholders were collected regarding missing features, potential conflicts, or unrealistic expectations.
- **Requirement Prioritization:**
  Collaborative work was done with stakeholders to prioritize requirements based on value, urgency, and feasibility.
- **Documentation Updates:**
  Requirement specifications were refined based on stakeholder feedback and finalize the validated set of requirements.

Importance:

- Ensures that the system meets the actual needs of end-users and project objectives.
- Minimizes the risk of misunderstandings or misinterpretations during development.
- Provides an agreed-upon baseline for design, implementation, and testing activities.
- Enhances stakeholder confidence and reduces the need for extensive revisions later in the project lifecycle.

Conclusion

This Requirement Analysis Report captures the key functional and non-functional requirements for the development of the Car Fault Diagnosis Mobile Application. Through systematic requirement gathering, validation with stakeholders, and careful analysis, the report provides a clear foundation for the design and development phases.

By aligning user needs with technical feasibility and stakeholder expectations, this report aims to ensure that the final product effectively addresses core problems faced by car owners, such as interpreting dashboard warning lights and diagnosing engine issues using AI. The validated requirements will guide the development team toward delivering a reliable, user-friendly, and secure solution that meets project goals and enhances the overall user experience.