REPUBLIQUE DU CAMEROON
PAIX-Travail-Patrie
MINISTRE DE
L'ENSEIGNEMENT SUPERIEUR
FACULTE D'INGINERIE
ET TECHGNOLOGIE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
MINISTER OF HIGHER EDUCATION

FACULTY OF ENGINEERING AND TECHNOLOGY

Requirements Analysis

Project: AI-based Car Fault Diagnosis Mobile Application (CarDiagApp)

Prepared by: Group 10

ABANDA SERGIO	**FE22A131**
ABILATEZIE VIN-WILSON ANU	**FE22A132**
AGBOR EMMANUEL NCHENGE	**FE22A138**
ASHLEY TAN WACHE	**FE22A154**
ASHU DESLEY	**FE22A155**

CEF440: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

INSTRUCTOR: Dr. VALERY NKEMENI

Date: May 5, 2025

Table of Contents

Requirements Analysis 1
Project: AI-based Car Fault Diagnosis Mobile Application (CarDiagApp) 1
Prepared by: Group 10 1
Requirements Analysis 4
General Introduction 4
1.1 Introduction 4
1.2 Review of raw user responses 4
1.2.1 Raw Functional Requirements from User Responses 4
1.2.2 Raw Non-Functional Requirements from User Responses 5
1.3 Analysis of Requirements 5
1.3.1 Finalized Functional Requirements with Explanations 6
1.3.2. Finalized Non-Functional Requirements with Explanations 6
2.0 Identify Issues in Requirements 7
2.1 Introduction to Issues Identification 7
2.2 Summary of Identified Issues 7
2.3 Conclusion to Issues Identification 8
3.0 Prioritization of Requirements Based on Importance and Feasibility 8
3.2 Summary of prioritization 9
4.0 Requirements classification and specifications 10
4.1 Purpose of Classifying Requirements 10
4.2 Functional Requirements Specifications 10
4.3 Non-Functional Requirements 11
5. Software Requirements Specification (SRS) 13
Version: 1.1 13
Status: Approved 13
5.1 Introduction to SRS 13
5.1.1 Purpose 13
5.1.2 Scope 13
5.1.3 Definitions, Acronyms, and Abbreviations 14
5.1.4 References 14

5.1.5 Overview 15
5.2. Overall Description 15
5.2.1 Product Perspective 15
5.2.2 Product Functions 16
5.2.3 User Classes and Characteristics 16
5.2.4 Operating Environment 17
5.2.5 Design and Implementation Constraints 18
5.2.6 User Documentation 19
5.2.7 Assumptions and Dependencies 19
5.3. Specific Requirements 20
5.3.1 Functional Requirements 20
5.3.2 External Interface Requirements 22
5.3.3 Non-Functional Requirements 26
5.4. System Features 32
5.4.1 User Authentication System 32
5.4.2 Dashboard Warning Light Scanner 34
5.4.3 Engine Sound Analyzer 35
5.4.4 Diagnosis and Reporting 36
5.4.5 Maintenance Tracking 37
5.4.6 Online/Offline Functionality 38
5.5.1 Testing Strategy 39
5.6. Appendices 39
A. Glossary 39
B. Analysis Models 40
C. Issue Tracking 40
6. Requirements Validation with Stakeholders 41
6.1. Validation and Risk Mitigation 41
6.1.1 Stakeholder Engagement 41
6.1.2 Identified Risks & Mitigation 42
General Conclusion 43

Requirements Analysis

General Introduction

Requirements analysis is the critical bridge between stakeholder aspirations and a successful software solution. In the context of our AI-powered car fault diagnosis app, this phase has gone far beyond capturing a wish list, it has rigorously validated and refined every feature through stakeholder workshops, surveys, expert interviews, brainstorming sessions, and reverse-engineering of existing tools. By dissecting user stories, mapping them to technical capabilities, and stress-testing each requirement for feasibility, clarity, and consistency, we have ensured that every line in our specification is both actionable and aligned with real-world needs. This disciplined approach minimizes downstream surprises, tightly focuses our development effort on high-impact functionality, and lays a rock-solid foundation for design and implementation.

1.0 Review and Analysis of Functional Requirements

1.1 Introduction

This report outlines the requirements for a car diagnostic mobile application based on user responses gathered through a structured survey. The report categorizes the requirements into functional and non-functional aspects. It follows a two-part structure for each:

(i) Review of raw user responses,

(ii) Analysis (based on completeness, clarity, technical feasibility, and dependencies relationships) of requirements.

1.2 Review of raw user responses

1.2.1 Raw Functional Requirements from User Responses

- Scan and interpret dashboard warning lights
- Record and analyze engine sounds
- Provide automated explanations and repair suggestions
- Provide access to video tutorials for repairs
- Display instant diagnosis results after analysis

- Offer detailed repair recommendations based on diagnosis
- Send maintenance reminders based on vehicle condition or schedule
- Log maintenance records and maintenance cost
- Support live chat with a mechanic or support representative
- Provide step-by-step guides for vehicle troubleshooting and repair
- Show maintenance alerts when critical thresholds are detected
- Diagnose all common and critical faults in a car
- Perform predictive maintenance using machine learning to anticipate potential issues
- Enable community forums and support interaction among users
- Provide early detection and prediction of faults before they escalate

1.2.2 Raw Non-Functional Requirements from User Responses

- Accuracy of the diagnosis
- Complexity of the technology
- High data usage or cost
- Zero ads display
- Efficiency

1.3 Analysis of Requirements

The requirements were assessed for:

- **Completeness:** whether enough information is provided--**Do the requirements cover all aspects of what the system needs to do?**
- **Clarity:** whether the requirement is easy to understand--**Is each requirement clearly expressed without ambiguity?**
- **Technical feasibility:** whether it is realistically implementable--Can these requirements be implemented with available technology?
- **Dependency relationships:** whether one requirement depends on others--How do requirements relate to and impact each other?

The results are summarized and refined below.

1.3.1 Finalized Functional Requirements with Explanations

1) **Scan and interpret dashboard warning lights**

   Use image recognition to detect and explain dashboard indicators.

2) **Record and analyze engine sounds**

   Capture engine noises via microphone and identify issues like misfires using ML algorithms.

3) **Provide automated explanations and repair suggestions**

   Generate readable reports with likely causes and potential solutions based on detected issues.

4) **Provide access to video tutorials for repairs**

   Offer repair videos for common problems to guide users through fixing them.

5) **Display instant diagnosis results after analysis**

   Show a quick summary of diagnostics as soon as analysis completes.

6) **Send maintenance reminders based on vehicle condition or schedule**

   Notify users of upcoming maintenance based on mileage or time intervals.

7) **Log maintenance records and maintenance cost**

   Maintain a history of repairs and cost estimates for future reference and analytics.

8) **Support live chat with a mechanic or support representative**

   Provide real-time chat with experts or AI assistant for consultation.

9) **Show maintenance alerts when critical thresholds are detected**

   Send warnings when certain engine or system values exceed safe limits.

10) **Perform predictive maintenance using machine learning to anticipate potential issues**

    Use historical data trends to forecast problems before they occur.

11) **Enable community forums and support interaction among users**

    Allow users to post, answer questions, and interact about vehicle issues and repairs.

12) **Provide early detection and prediction of faults before they escalate**

    Continuously monitor and alert users when early signs of potential problems are identified.

1.3.2. Finalized Non-Functional Requirements with Explanations

1) **Accuracy of the diagnosis**

   The app should provide diagnostic results with at least 90% accuracy under typical conditions.

2) **Complexity of the technology**

   The UI/UX design must simplify complex diagnostic operations for non-technical users.

3) **Data usage and cost optimization**

   The app should minimize bandwidth consumption by using lightweight data formats and caching.

4) **Zero advertisement policy**

   No in-app advertisements shall be displayed to maintain focus and professionalism.

5) **Efficiency**

   App functions should complete within acceptable timeframes

2.0 Identify Issues in Requirements

2.1 Introduction to Issues Identification

High-quality, unambiguous requirements are the foundation of any successful software project. To ensure our car-fault diagnostic app meets user and technical needs, we conducted structured peer reviews and walkthroughs with the development team, systematically flagging and clarifying problematic statements before design and implementation began.

2.2 Summary of Identified Issues

- **Inconsistencies:**

  We discovered overlapping repair-guidance requirements. For example, "Offer detailed repair recommendations based on diagnosis" and "Provide step-by-step guides for vehicle troubleshooting and repair" both duplicate "**Provide automated explanations and repair suggestions,**" leading to redundant functionality that must be consolidated.

- **Ambiguities:**

  The requirement "**Diagnose all common and critical faults in a car**" is overly broad—its scope and the definition of "common" versus "critical" faults vary by make, model, and region. We must specify which fault categories are supported to avoid misinterpretation.

- **Missing Information:**

  Our review found no gaps in the documented requirements. All stakeholder needs appeared to be captured; instead, the focus is on refining existing statements for clarity and consistency.

2.3 Conclusion to Issues Identification

By identifying and resolving these inconsistencies and ambiguities early, through collaborative reviews and targeted stakeholder queries, we strengthen the SRS, reduce downstream rework, and pave the way for a development phase that is both efficient and aligned with user expectations.

3.0 Prioritization of Requirements Based on Importance and Feasibility

To guide an efficient development road map, the requirements were prioritized using the MoSCoW method:

- **Must-Have:** Critical for core functionality and user value.
- **Should-Have:** Important but not critical; adds value.
- **Could-Have:** Useful enhancements; implementable later.
- **Won't-Have** (for now): Out of scope or not feasible in the current phase.

Requirement	**Priority**	**Justification**
Scan and interpret dashboard warning lights	Must-Have	Foundational functionality; feasible via image recognition
Record and analyze engine sounds	Must-Have	Core ML-based feature for fault detection
Provide automated explanations and repair suggestions	Must-Have	Essential for user understanding and self-action
Display instant diagnosis results after analysis	Must-Have	Enhances user experience and immediacy
Show maintenance alerts when thresholds are detected	Must-Have	Preventive maintenance function with direct impact on safety
Perform predictive maintenance using machine learning	Should-Have	High value but requires historical data and advanced modeling

Requirement	Priority	Justification
Provide access to video tutorials for repairs	Could-Have	Valuable for user guidance, but depends on content integration
Send maintenance reminders based on condition/schedule	Should-Have	Enhances utility; implementable with scheduled triggers
Log maintenance records and maintenance cost	Should-Have	Improves record keeping, adds analytical capabilities
Support live chat with mechanic/support representative	Could-Have	Useful for user assistance but costly and complex for initial version
Enable community forums and user interaction	Could-Have	Promotes engagement but not essential at MVP stage
Provide early detection and prediction of faults	Must-Have	Aligns with preventive diagnostics goal

3.2 Summary of prioritization

This extended analysis confirms the technical and user-aligned prioritization of functional and non-functional requirements for the car diagnostic mobile application. Stakeholder feedback has been effectively integrated to ensure that development efforts focus first on high-impact, technically feasible functionalities, with enhancement features planned for subsequent releases.

4.0 Requirements classification and specifications

4.1 Purpose of Classifying Requirements

Classifying requirements into functional and non-functional categories ensures clear understanding of what the system must do (functional) versus how well it must do it (non-functional). This helps:

- Guide the design and implementation process.
- Ensure all user needs and technical constraints are addressed.
- Facilitate testing, validation, and system maintenance.
- Aid in resource allocation and prioritization.

4.2 Functional Requirements Specifications

ID	Functional Requirement	Specification
FR1	User Registration	User registration with email verification step to confirm identity.
FR2	Role-based Authentication	Role-based login access (Car Owner, Mechanic, Administrator).
FR3	Password Recovery	Secure password recovery via email or security questions.
FR4	Session Management	User sessions with inactivity timeout to ensure security.
FR5	Image Capture	Users can take dashboard photos; app detects warning icons.
FR6	Icon Interpretation	App explains detected icons with urgency labels (e.g., critical).
FR7	Audio Recording	Record engine sounds for up to 10 seconds using device microphone.
FR8	Sound Classification	Audio analysis classifies sound into predefined fault categories.
FR9	Text Explanation	Faults include explanations and text-based repair suggestions.

FR10	Tutorial Links	Diagnostic results include links to relevant tutorial videos.
FR11	History Storage	Diagnostic history saved locally with timestamps.
FR12	Maintenance Reminders	Sends reminders for maintenance based on faults or schedule.
FR13	Offline Diagnostics	Diagnostics (image/audio) work offline using preloaded models.
FR14	Online Updates	Updates fault database/tutorials when internet is available.
FR15	Vehicle Info Input	Vehicle details (make/model/year) influence diagnostic accuracy.
FR16	Multiple Icon Detection	Detects and interprets multiple warning icons in one image.
FR17	Urgency Estimation	Assigns urgency levels (Immediate, Soon, Monitor) to issues.
FR18	Report Sharing	Users can save/share diagnostics reports.
FR19	Maintenance Tips	Provides preventive tips based on identified problems.
FR20	Mechanic Notes	Mechanics can add notes to diagnostic reports (restricted to role).
FR21	User Management	Admins manage users (add, remove, edit profiles).
FR22	Profile Management	Users can edit profile and set preferences.

4.3 Non-Functional Requirements

Performance

- Fast diagnostics: ≤5s (image), ≤7s (audio)
- Fast UI and authentication: ≤3s

- High throughput: 50 sessions/day/user, 100+ concurrent auth requests
- Resource limits: <200MB RAM, <30% CPU, <1% battery during standby

Reliability

- Uptime: 99% online, 100% offline core features
- MTBF: <5 crashes per 1000 sessions
- Strong error handling and recovery for session loss, denied permissions, network failure

Usability

- Simple UI for non-technical users; max 3 taps to any function
- Fast onboarding: <2 mins registration, <3 mins first diagnosis
- Accessibility: screen readers, voice commands, high contrast, WCAG 2.1 AA compliance

Security

- Multi-factor auth, secure storage, role-based access
- Local processing of sensitive data, opt-in for analytics
- Explicit permission requests with clear justification
- Integrity checks, audit trails, anti-tampering

Maintainability & Portability

- Modular, documented Flutter codebase with layered architecture
- Support for Android/iOS, including tablets
- OTA updates, versioned migrations, isolated auth module

Quality Attributes

- Accuracy: ≥90% (visual), ≥85% (audio), <5% false positives
- Efficient: ≤5s total diagnosis, low battery and data use
- Privacy: no tracking, data anonymized, compliant with regulations
- Lightweight: optimized for low bandwidth, compressed data transfers

5. Software Requirements Specification (SRS)

Version: 1.1
Status: Approved

5.1 Introduction to SRS

5.1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for an AI-based mobile application that enables car owners to diagnose faults using dashboard warning light recognition and engine sound analysis. It serves as the agreement between stakeholders and the development team and provides a basis for estimating costs, schedules, and validation criteria.

5.1.2 Scope

The application ("CarDiagApp") will allow users to:

- Register and login with role-based access (car owners, mechanics, administrators)
- Capture and interpret dashboard warning lights via camera
- Record and classify engine sounds via microphone
- Receive automated explanations, repair suggestions, maintenance alerts, and video tutorials
- Track maintenance history and receive reminders
- Operate offline for core diagnostics and online for updated data

The application will not:

- Replace professional diagnostic tools or mechanic expertise

- Interface directly with vehicle onboard computers (no OBD-II connection)
- Provide real-time monitoring while driving
- Guarantee 100% accuracy in all diagnostic scenarios

5.1.3 Definitions, Acronyms, and Abbreviations

- **App:** The CarDiagApp mobile application
- **ML:** Machine Learning
- **UI/UX:** User Interface / User Experience
- **MVP:** Minimum Viable Product
- **OBD-II:** On-Board Diagnostics II (vehicle self-diagnostic standard)
- **CV:** Computer Vision
- **PII:** Personally Identifiable Information
- **API:** Application Programming Interface
- **2FA:** Two-Factor Authentication
- **RBAC:** Role-Based Access Control

5.1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for SRS
- Identify_Issues and Priority.docx: Requirements analysis and prioritization
- GROUP_10_FINAL_REPORT_TASK2.pdf: Report on requirement gathering
- ISO/IEC 25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE)

- ISO 26262: Road vehicles - Functional safety standard
- OWASP Mobile Application Security Verification Standard (MASVS)

5.1.5 Overview

The remainder of this document covers:

- Section 2: Overall description of the product, including perspective, functions, user characteristics, constraints, and assumptions
- Section 3: Specific requirements, including functional, interface, and non-functional requirements
- Section 4: System features with detailed descriptions of core functionality
- Section 5: Validation methods and risk mitigation strategies
- Section 6: Appendices with supplementary information

5.2. Overall Description

5.2.1 Product Perspective

CarDiagApp is a standalone mobile application designed for iOS and Android platforms using the Flutter cross-platform framework. It leverages on-device ML models (TensorFlow Lite) for image and audio classification, with optional cloud connectivity for data updates and enhanced features.

The application relates to existing products in the following ways:

- Unlike professional OBD-II scanners, it doesn't require physical connection to the vehicle
- Compared to existing automotive apps, it uniquely combines visual and audio analysis
- Integrates with third-party content platforms (YouTube) for educational materials

5.2.2 Product Functions

- **User Authentication:** Registration, login, and profile management with role-based access
- **Dashboard Light Scanner:** Detect and classify multiple warning lights in a single image
- **Engine Sound Analyzer:** Record, preprocess, and classify engine noises (knocking, squealing, hissing, etc.)
- **Diagnosis Report:** Provide cause, urgency level, repair tips, and video links
- **Maintenance Tracker:** Log past diagnostics and schedule reminders
- **Knowledge Base:** Access to common car problems and solutions
- **Offline Mode:** Core features without internet; **Online Mode:** Fetch latest models and tutorials

5.2.3 User Classes and Characteristics

- **Primary: Car Owners (Non-Technical)**
    - Demographics: Ages 18-65, varying technical expertise
    - Goals: Quick diagnosis, basic understanding of car issues
    - Technical expertise: Limited automotive knowledge
    - Usage frequency: Occasional, when warning lights appear
    - Access level: Standard diagnostic and maintenance features
- **Secondary: Mechanics/Technicians**
    - Goals: Quick pre-checks, client education tool
    - Technical expertise: Professional automotive knowledge
    - Usage frequency: Regular, multiple vehicles

- Access level: Enhanced diagnostic features, ability to add professional notes

- **Tertiary: Content Creators**
  - Goals: Provide and manage tutorial content
  - Technical expertise: Automotive expertise, content creation skills
  - Usage frequency: Content submission and updates
  - Access level: Content management interface

- **System Administrators**
  - Goals: Manage backend, update models, monitor performance
  - Technical expertise: IT/ML expertise
  - Usage frequency: Regular maintenance and monitoring
  - Access level: Full system access, user management, analytics dashboard

5.2.4 Operating Environment

- **Mobile Devices:**
  - Android 8.0 (API level 26) or higher
  - iOS 12.0 or higher
  - Minimum 2GB RAM, 100MB free storage
  - Camera with minimum 5MP resolution
  - Microphone with noise cancellation capabilities (preferred)

- **Network:**
  - Internet connectivity for online features (Wi-Fi or cellular)

- Offline capability for core features

- **Backend Services:**
  - Cloud-based user authentication system
  - Cloud-based model training infrastructure
  - Content delivery network for tutorials and model updates
  - Secure database for user profiles and preferences

5.2.5 Design and Implementation Constraints

- **Technical Constraints:**
  - On-device ML model size must be <10MB
  - Use TensorFlow Lite for ML inference
  - Flutter framework for cross-platform development
  - Latency: Diagnostic feedback within 5 seconds
  - Battery usage: <5% per diagnostic session
- **Business Constraints:**
  - Comply with privacy regulations: no personal data storage beyond app scope
  - No advertisements in core diagnostic functions
  - App size not to exceed 50MB for initial download
- **Regulatory Constraints:**
  - Must include disclaimer about diagnostic accuracy
  - Must comply with data protection regulations (GDPR, CCPA)

- Cannot claim to replace professional diagnosis
- Authentication must comply with industry security standards

5.2.6 User Documentation

The following user documentation will be provided:

- In-app tutorial for first-time users
- FAQ section covering common use cases
- Online help center with troubleshooting guides
- Video demonstrations for key features
- Tooltips for interface elements
- Role-specific guidance for different user types

5.2.7 Assumptions and Dependencies

- Users grant camera and microphone permissions
- Access to YouTube API for tutorial links
- Availability of labeled audio and image datasets for model training
- Users have basic familiarity with smartphone operation
- Adequate lighting conditions for dashboard photo capture
- Vehicle engine accessible for sound recording
- Secure authentication service availability

5.3. Specific Requirements

5.3.1 Functional Requirements

ID	Requirement	Priority	Source
FR1	The app shall provide user registration functionality with email verification.	Must-Have	Security Standards
FR2	The app shall support role-based user authentication (car owner, mechanic, and administrator).	Must-Have	Project Revision
FR3	The app shall allow users to recover forgotten passwords securely.	Must-Have	Security Standards
FR4	The app shall maintain user sessions with appropriate timeout controls.	Must-Have	Security Standards
FR5	The app shall allow users to capture dashboard images and detect warning light icons.	Must-Have	Survey, Interviews
FR6	The app shall interpret each detected icon and display its meaning and urgency level.	Must-Have	Survey, Expert Interviews
FR7	The app shall record engine sound clips up to 10 seconds in length.	Must-Have	Survey, Interviews
FR8	The app shall classify recorded engine sounds into predefined categories (e.g., knocking, squealing).	Must-Have	Requirements Analysis
FR9	The app shall provide text-based explanations and repair suggestions for detected faults.	Must-Have	Survey, Interviews

ID	Requirement	Priority	Source
FR10	The app shall include links to video tutorials relevant to the diagnosed issue.	Should-Have	Survey, User Feedback
FR11	The app shall store past diagnostics locally with timestamps and results.	Should-Have	Survey, Interviews
FR12	The app shall send maintenance reminders based on user-defined schedules or fault urgency.	Should-Have	Requirements Analysis
FR13	The app shall operate offline for image and audio diagnostics using preloaded models.	Must-Have	Survey, Interviews
FR14	The app shall fetch updated fault databases and tutorial metadata when online.	Should-Have	Survey, Interviews
FR15	The app shall allow users to input vehicle make, model, and year for more specific diagnostics.	Should-Have	Expert Interviews
FR16	The app shall detect multiple warning lights simultaneously in a single image.	Must-Have	Project Description
FR17	The app shall estimate maintenance urgency on a scale (immediate, soon, monitoring).	Must-Have	Project Description
FR18	The app shall allow users to save and share diagnostic reports.	Could-Have	User Feedback
FR19	The app shall provide maintenance tips specific to identified issues.	Should-Have	Expert Interviews

ID	Requirement	Priority	Source
FR20	The app shall enable mechanics to add professional notes to diagnoses (mechanic role only).	Should-Have	Project Revision
FR21	The app shall provide administrators with user management capabilities.	Must-Have	Project Revision
FR22	The app shall allow users to manage their profile information and preferences.	Should-Have	User Feedback

5.3.2 External Interface Requirements

5.3.2.1 User Interfaces

- **Authentication Screens:**
    - Registration form with email verification
    - Login screen with forgot password option
    - Role selection during registration
    - Profile management screen
- **Home Screen:**
    - Buttons for "Scan Lights" and "Record Sound"
    - Quick access to history and settings
    - Vehicle profile selection
    - User-specific features based on role

- **Dashboard Scanner Screen:**
    - Camera viewfinder with guide overlay
    - Capture button and flash control
    - Preview and retake options
- **Sound Recorder Screen:**
    - Recording button with timer
    - Sound level visualization
    - Instructions for optimal recording
- **Results Screen:**
    - Displays identified warning icons with labels
    - Sound classification results
    - Explanations, urgency rating, and tutorial links
    - Save/share options
    - Professional notes section (visible to mechanics)
- **History Screen:**
    - List of past diagnostics with date/time and summary
    - Filtering and search functionality
    - Option to compare multiple diagnoses
- **Admin Dashboard:**
    - User management interface

- System performance metrics
- Model update controls

- **Settings Screen:**
  - Notification preferences
  - Data usage controls
  - Vehicle profile management
  - Authentication settings

5.3.2.2 Hardware Interfaces

- **Camera:**
  - Access to rear-facing camera with autofocus
  - Flash control for low-light conditions
  - Minimum 5MP resolution
- **Microphone:**
  - Access to device microphone
  - Support for external microphones
  - Audio sampling at 44.1kHz, 16-bit depth
- **Storage:**
  - Access to local storage for saving diagnostic history
  - Cache management for model data
  - Secure storage for authentication tokens

- **Display:**
    - Support for various screen sizes and resolutions
    - Landscape and portrait orientation support
    - Accessibility considerations for text size and contrast

5.3.2.3 Software Interfaces

- **Authentication Service:**
    - Firebase Authentication
    - OAuth 2.0 integration for social logins (optional)
    - Token-based authentication system
- **TensorFlow Lite:**
    - On-device ML inference using .tflite models
    - Model version tracking and updates
- **YouTube Data API:**
    - Tutorial video retrieval and embedding
    - Search functionality based on diagnostic results
- **Local Database:**
    - SQLite for diagnostic history storage
    - Secure storage for user credentials
    - JSON files for vehicle specifications
- **Push Notification Service:**

- Firebase Cloud Messaging for maintenance reminders
- Authentication alerts

5.3.2.4 Communications Interfaces

- **Network:**
  - HTTPS/TLS for secure data fetch and API calls
  - REST API for backend services
  - Secure communication for authentication
- **File Transfer:**
  - Secure download of model updates
  - Upload of anonymized feedback data (with user consent)
  - Encrypted data transmission

5.3.3 Non-Functional Requirements

5.3.3.1 Performance Requirements

- **Response Time:**
  - Maximum 5 seconds for image classification
  - Maximum 7 seconds for audio classification
  - Maximum 2 second for UI transitions
  - Authentication processes complete within 3 seconds
- **Startup Time:**
  - App launches within 5 seconds on supported devices

- Models load within 5 seconds after launch
- Authentication state verification within 2 second

- **Throughput:**
  - Support for up to 50 diagnostic sessions per day per user
  - Database queries complete in <100ms
  - Handle 100+ concurrent authentication requests

- **Resource Utilization:**
  - Memory usage <200MB during operation
  - CPU usage <30% during analysis
  - Battery consumption <1% during standby

5.3.3.2 Reliability Requirements

- **Availability:**
  - 99% uptime for online features
  - 100% availability for offline core features
  - Authentication services available 99.9% of time

- **MTBF (Mean Time Between Failures):**
  - Less than 5 crash per 1000 user sessions
  - Authentication system failure rate <0.5%

- **Error Handling:**
  - Graceful message if camera/mic access is denied

- Appropriate fallbacks for network failures
- User-friendly error messages with recovery options
- Authentication failure handling with clear user guidance

- **Recovery:**
  - Auto-save partial diagnostic data during system interruptions
  - Automatic retry for failed network operations
  - Session recovery after unexpected termination

5.3.3.3 Usability Requirements

- **UI Complexity:**
  - Suitable for users with no technical background
  - Maximum 3 taps to reach any main function
  - Consistent design language and navigation patterns
  - Clear role-specific interfaces
- **Learnability:**
  - First-time users can complete registration in <2 minutes
  - First-time users can complete a diagnosis in <3 minutes
  - Interactive onboarding tutorial for new users
  - Role-specific guidance for different user types
- **Accessibility:**
  - Support for screen readers and large fonts

- Color schemes with adequate contrast ratios
- Voice commands for key functions
- Compliance with WCAG 2.1 AA standards

5.3.3.4 Security Requirements

- **Authentication:**
  - Multi-factor authentication option
  - Password complexity requirements
  - Account lockout after multiple failed attempts
  - Secure credential storage
- **Data Privacy:**
  - Audio/image data processed locally; no PII stored or transmitted
  - Anonymized usage data only with explicit opt-in
  - Secure storage of diagnostic history
  - User data segregation and access controls
- **Permissions:**
  - Explicit runtime requests for camera and microphone
  - Granular permission management
  - Clear explanation of permission usage
  - Role-based access control for features
- **Data Integrity:**

- Checksum verification for model updates
- Prevention of diagnostic history tampering
- Audit trails for administrative actions

5.3.3.5 Maintainability and Portability

- **Codebase Structure:**
  - Modular design in Flutter
  - Separation of UI, business logic, and data layers
  - Comprehensive code documentation
  - Authentication module isolation
- **Platform Support:**
  - iOS and Android with identical feature sets
  - Tablet optimization for larger screens
  - Consistent authentication experience across platforms
- **Upgradability:**
  - Support for over-the-air model updates
  - Database migration strategy for version updates
  - Authentication system upgradable without user disruption

5.3.3.6 Quality Attributes

- **Accuracy:**
  - ≥90% diagnostic accuracy for common warning lights

- ≥85% accuracy for engine sound classification
- False positive rate <5%
- Zero authentication false positives

- **Efficiency:**
  - ≤5s total response time for complete diagnosis
  - Battery consumption <1% per diagnostic session
  - Data usage <1MB per online session (excluding video playback)
  - Authentication tokens with appropriate expiration periods
- **Privacy:**
  - User data anonymized and secure
  - No tracking of user location or behavior
  - Compliance with global privacy regulations
  - Separation of authentication and diagnostic data
- **Data Usage:**
  - Optimized for low bandwidth environments
  - Data compression for model updates
  - Wi-Fi only option for large downloads
  - Minimal data usage for authentication processes

5.4. System Features

5.4.1 User Authentication System

5.4.1.1 Description

The user authentication system provides secure registration, login, and role-based access control for different user types (car owners, mechanics, and administrators), ensuring appropriate feature access and data security.

5.4.1.2 Functional Requirements

- FR1: User registration with email verification
- FR2: Role-based authentication
- FR3: Password recovery
- FR4: Session management
- FR21: Administrator user management
- FR22: Profile management

5.4.1.3 Technical Implementation

- Firebase Authentication or equivalent backend service
- Token-based authentication
- Secure credential storage using industry-standard encryption
- Role-based access control (RBAC) system
- Email verification workflow
- Password policy enforcement

5.4.1.4 User Workflow

1. **Registration:**
    - User selects "Register" from login screen
    - User provides email, password, and selects role (car owner/mechanic)
    - Email verification sent to confirm address
    - User confirms email and completes profile setup
2. **Login:**
    - User enters credentials on login screen
    - System authenticates and determines appropriate role
    - User directed to role-specific home screen
3. **Password Recovery:**
    - User selects "Forgot Password" from login screen
    - User provides registered email
    - System sends password reset link
    - User creates new password
4. **Profile Management:**
    - User accesses profile from settings
    - Can update personal information, vehicle details, and preferences
    - Changes saved securely to user profile

5.4.2 Dashboard Warning Light Scanner

5.4.2.1 Description

The dashboard warning light scanner allows users to capture images of their vehicle's dashboard and automatically identifies and interprets warning lights that are active.

5.4.2.2 Functional Requirements

- FR5: Image capture functionality
- FR6: Warning light interpretation
- FR16: Multi-light detection capability

5.4.2.3 Technical Implementation

- Computer vision model using CNN architecture
- Preprocessing steps for image enhancement in various lighting conditions
- Database of 100+ common warning light symbols across major manufacturers
- Real-time highlighting of detected symbols

5.4.2.4 User Workflow

1. User selects "Scan Dashboard" from home screen
2. Camera interface appears with framing guide
3. User captures dashboard image
4. App processes image and highlights detected warning lights
5. User confirms or retakes image
6. App displays detailed information about each detected light

5.4.3 Engine Sound Analyzer

5.4.3.1 Description

The engine sound analyzer records audio of engine operation and uses machine learning to classify sounds associated with common mechanical issues.

5.4.3.2 Functional Requirements

- FR7: Audio recording capability
- FR8: Sound classification functionality

5.4.3.3 Technical Implementation

- Audio preprocessing pipeline including noise reduction
- Feature extraction using MFCC (Mel-frequency cepstral coefficients)
- Classification model using CNN or LSTM architecture
- Detection of at least 10 common problematic engine sounds

5.4.3.4 User Workflow

1. User selects "Record Engine Sound" from home screen
2. App provides instructions for optimal recording
3. User positions device and starts recording (5-10 seconds)
4. App processes audio and performs classification
5. Results display with confidence scores for detected issues

5.4.4 Diagnosis and Reporting

5.4.4.1 Description

The diagnosis system combines results from warning light and sound analysis to provide comprehensive fault information, repair suggestions, and educational content.

5.4.4.2 Functional Requirements

- FR9: Explanation and repair suggestions
- FR10: Video tutorial links
- FR17: Urgency estimation
- FR19: Maintenance tips
- FR20: Professional notes (mechanic role)

5.4.4.3 Technical Implementation

- Rule-based system for combining visual and audio diagnostic results
- Urgency algorithm based on safety implications and potential damage
- Curated database of repair suggestions and maintenance tips
- Integration with YouTube API for relevant tutorial retrieval
- Notes system with role-based visibility

5.4.4.4 User Workflow

1. After completing scan/recording, results page displays
2. Each issue shows description, potential causes, and urgency level
3. User can expand each issue for detailed information

4. Video tutorials appear as embedded previews or external links
5. Mechanics can add professional notes to diagnosis
6. User can save report or share with mechanic

5.4.5 Maintenance Tracking

5.4.5.1 Description

The maintenance tracking system stores diagnostic history and provides reminders for regular maintenance and follow-ups on detected issues.

5.4.5.2 Functional Requirements

- FR11: Diagnostic history storage
- FR12: Maintenance reminders
- FR18: Report saving and sharing

5.4.5.3 Technical Implementation

- Local SQLite database for diagnostic history
- Cloud synchronization for registered users
- Notification system for scheduled maintenance
- Export functionality for reports (PDF format)
- Secure sharing via standard OS mechanisms

5.4.5.4 User Workflow

1. User accesses history from home screen
2. Chronological list of past diagnostics displayed

3. User can filter by date, issue type, or urgency
4. Detailed view available for each past diagnosis
5. Options to set reminders or share with mechanic

5.4.6 Online/Offline Functionality

5.4.6.1 Description

The application provides core functionality without internet connection while enabling enhanced features and updates when online.

5.4.6.2 Functional Requirements

- FR13: Offline operation for core diagnostics
- FR14: Online updates for databases and models
- FR15: Vehicle-specific diagnostics

5.4.6.3 Technical Implementation

- Preloaded ML models for offline operation
- Incremental model updates to minimize data usage
- Vehicle database with common specifications and known issues
- Version management system for models and databases
- Offline authentication caching mechanism

5.4.6.4 User Workflow

1. App functions with core capabilities in offline mode for authenticated users
2. Background checks for updates when connectivity restored

3. Notification of available updates with size information
4. Option to schedule updates during Wi-Fi connection

5.5.1 Testing Strategy

- **Security Testing:** Authentication system penetration testing and vulnerability assessment
- **Unit Testing:** Individual components (authentication, camera, audio recording, classification)
- **Integration Testing:** Combined diagnostic workflows
- **Performance Testing:** Response times, resource usage
- **Usability Testing:** With representative user groups across different roles
- **Beta Testing:** Limited release to gather real-world feedback
- **Model Validation:** Using labeled test datasets for accuracy metrics

5.6. Appendices

A. Glossary

- **Authentication:** The process of verifying a user's identity
- **RBAC:** Role-Based Access Control - restricting system access based on user roles
- **Diagnosis:** Process of identifying mechanical or electrical faults
- **Model Inference:** Running ML model on input to get predictions
- **Warning Light:** Dashboard indicator that signals vehicle system status
- **OBD System:** On-Board Diagnostics system in modern vehicles
- **Engine Knocking:** Abnormal combustion sound indicating potential issues

- **False Positive:** Incorrect identification of a non-existent issue
- **Model Quantization:** Technique to reduce ML model size for mobile devices
- **Token:** A piece of data used in authentication processes to represent authorization

B. Analysis Models

- **Use Case Diagram:** Captures major use cases: Register, Login, ScanLights, RecordSound, ViewResults, ViewHistory
- **Data Flow Diagram:** Illustrates flow: Authentication → Input → Preprocessing → Inference → Output
- **Class Diagram:** Shows relationships between major software components
- **Sequence Diagram:** Details interactions during authentication and diagnostic process
- **State Diagram:** Represents application states during user session

C. Issue Tracking

ID	Issue	Status	Resolution
ISSUE-001	Authentication token expiration handling	In Progress	Implementing refresh token mechanism
ISSUE-002	Model accuracy in low light	Open	Investigating preprocessing enhancements
ISSUE-003	Audio classification with background noise	In Progress	Adding noise isolation algorithms
ISSUE-004	Battery consumption during scanning	Resolved	Optimized camera operations

ID	Issue	Status	Resolution
ISSUE-005	Support for international warning symbols	Open	Expanding training dataset
ISSUE-006	Role transition process for users	Open	Designing approval workflow
ISSUE-007	Tutorial relevance for specific issues	In Progress	Refining metadata matching algorithm

6. Requirements Validation with Stakeholders

6.1. Validation and Risk Mitigation

6.1.1 Stakeholder Engagement

The requirements were gathered and validated through:

- User surveys targeting car owners of various expertise levels
- Interviews with automotive technicians and mechanics
- Review sessions with project stakeholders
- Competitive analysis of existing diagnostic tools

6.1.2 Identified Risks & Mitigation

Risk	Impact	Likelihood	Mitigation Strategy
Authentication system breach	Critical	Low	Implement industry-standard security protocols and regular security audits
Inaccurate fault classification	High	Medium	Iterative model retraining and user feedback loop
Limited datasets availability	Medium	High	Use public datasets and crowd-source anonymized data
Audio interference in real-world recordings	Medium	Medium	Include noise-reduction preprocessing pipeline
Model size exceeds mobile limits	High	Low	Optimize using model quantization and pruning
Low adoption due to technical complexity	High	Medium	Focus on UI/UX simplicity and education tools
Battery drain during extended use	Medium	Medium	Optimize algorithms and implement power-saving modes
False sense of security from diagnosis	High	Medium	Clear disclaimers and urgency indicators
Privacy concerns with vehicle data	Medium	Medium	Local processing and explicit consent for any data sharing

Risk	Impact	Likelihood	Mitigation Strategy
User resistance to registration	Medium	High	Offer limited guest mode functionality

General Conclusion

Our structured requirements analysis has delivered a prioritized, unambiguous, and complete set of functional and non-functional requirements. We've reconciled overlapping features, resolved ambiguous scopes, and verified that no critical gaps remain. Armed with this refined specification, the project team can now move confidently into system architecture and development, knowing that every diagnostic capability, user interaction, and performance target has been meticulously vetted and is poised to deliver maximum value to car owners and technicians alike.