UNIVERSITY OF BUEA

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

P.O. Box 63,
Buea, South West Region
CAMEROON
Tel : (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

DESIGN AND IMPLEMENTATION OF A MOBILE APPLICATION FOR CAR FAULT DIAGNOSIS

Project Title: Car Fault Diagnosis Mobile Application

Course Title: Internet Programming and Network Programming

Course Code: CEF 440

By Group 2:

BERINYUY CLETUS FE21A148
EFUETLATEH JOAN FE22A197
ETAPE NGABE FE22A210
ETIENDEM PEARL FE22A211
TATA GLEN FE22A309

2024/2025 Academic Year

Table of Contents

Table of Figures

CHAPTER ONE: GENERAL INTRODUCTION

1. Background and Context of the Study

Modern vehicles are becoming increasingly sophisticated, integrating advanced electronics, control systems, and smart sensors to enhance performance, comfort, and safety. However, this technological advancement has made fault diagnosis more complex for everyday car owners. Traditionally, vehicle diagnosis relied heavily on manual inspections and the expertise of mechanics, which often led to delays, higher repair costs, and reduced user confidence.

With the widespread adoption of smartphones equipped with powerful cameras and microphones, there is a unique opportunity to provide car owners with accessible and intelligent diagnostic tools. This context motivates the development of **CarCare**, a mobile application designed to empower car owners to identify, understand, and address car faults without the immediate need for specialized tools or deep mechanical knowledge.

2. Problem Statement

Car owners frequently face challenges in interpreting dashboard warning lights or unusual engine sounds. As a result, minor faults may escalate into serious and costly issues. Moreover, many current diagnostic solutions require external hardware such as OBD-II scanners and do not offer clear, user-friendly explanations. Mechanics also face challenges in efficiently reaching car owners who require reliable service.

There is a need for an integrated, user-friendly solution that allows car owners to perform preliminary diagnostics using their mobile devices and connect directly to qualified mechanics when necessary.

3. Objectives of the Study

3.1 General Objective

To design and implement an intelligent mobile application that enables car owners to diagnose vehicle faults through image and sound analysis and facilitates connection with local mechanics for further support.

3.2 Specific Objectives

- To develop a user-friendly interface tailored for both car owners and mechanics.

- To integrate camera-based dashboard light analysis and microphone-based engine sound diagnosis.
- To implement a secure backend and relational database for storing diagnostic and user data.
- To provide clear, step-by-step repair instructions and AI-recommended tutorial videos.
- To create a directory system for mechanics to register and showcase their services.

4. Proposed Methodology

The methodology combines user-centered design and iterative development. Steps include:

1. Requirement analysis through surveys and interviews with car owners and mechanics.
2. System design using diagrams (context, DFDs, UML) to plan architecture.
3. UI design with Figma mockups and wireframes.
4. Backend and database implementation using Node.js and MySQL.
5. AI module integration for image and sound analysis.
6. Testing the prototype with real users and analyzing feedback.
7. Deploying a pilot version for further evaluation.

5. Research Questions

- How can mobile devices be used effectively to diagnose vehicle faults?
- What design features make a diagnostic app more accessible to non-technical users?
- How can mechanics leverage a mobile platform to better reach and assist clients?

6. Research Hypotheses

- H1: Using mobile sensors (camera and mic) improves early fault detection for car owners.
- H2: Step-by-step instructions and video tutorials enhance user engagement and reduce repair delays.
- H3: Mechanics benefit from increased visibility and accessibility when integrated into a diagnostic app.

7. Significance of the Study

This study addresses a critical gap by providing car owners with an intuitive, accessible way to understand and respond to vehicle issues. It reduces maintenance delays, empowers users, and opens new service opportunities for mechanics. Technologically, it demonstrates the potential of AI and mobile sensing in automotive diagnostics.

8. Scope of the Study

The study focuses on:

- Developing a prototype mobile app for Android and iOS.
- Supporting common dashboard symbols and engine sound faults.
- Including mechanic profile management and connection features.
- Limited to urban areas for initial testing.

9. Delimitation of the Study

- Limited to a set of pre-defined dashboard symbols and common engine faults.
- AI models rely on existing data sets rather than extensive new data collection.
- Pilot mechanic network restricted to specific regions.

10. Definition of Keywords and Terms

- **Dashboard Light**: Visual indicators on the vehicle dashboard signaling specific issues.
- **Engine Sound Analysis**: Diagnosing engine health by analyzing audio signals.
- **AI (Artificial Intelligence)**: Machine learning algorithms used for image and sound classification.
- **Mechanic Profile**: Registered mechanic details including contact information and services.
- **Tutorial Video**: Instructional media guiding car owners through repair tasks.

11. Organization of the Dissertation

- **Chapter 1**: General Introduction
- **Chapter 2**: Literature Review
- **Chapter 3**: Analysis and Design
- **Chapter 4**: Implementation and Results
- **Chapter 5**: Conclusion and Further Works

CHAPTER 2: LITERATURE REVIEW

1. Introduction

With advances in automotive technology, vehicle systems have become increasingly complex, integrating electronic control units (ECUs), sensors, and advanced software. As a result, car diagnostics has evolved from manual inspections to sophisticated, computer-aided fault detection. This chapter reviews existing research, systems, and technologies that inform the development of the CarCare application.

2. Vehicle Diagnostics and Traditional Methods

Traditionally, mechanics relied on manual inspection and experience to detect faults. Early methods included observing physical symptoms such as smoke, noises, or fluid leaks. The emergence of the **On-Board Diagnostics (OBD)** system, particularly **OBD-II**, marked a significant advancement by providing standardized fault codes and sensor data. However, OBD-II scanners require specialized tools and knowledge, limiting accessibility for average car owners.

3. Existing Mobile Diagnostic Tools

Several mobile applications, such as Torque, FIXD, and Car Scanner, offer diagnostic capabilities through OBD-II connections. While these tools are effective for basic fault code reading, they rely on hardware dongles and do not typically analyze sounds or images. Furthermore, most apps provide technical data without simplified explanations, making them less user-friendly for non-experts.

4. Limitations of Current Solutions

Current solutions focus heavily on hardware-dependent diagnosis and often lack:

- Image-based dashboard light analysis.
- Sound-based engine fault detection.
- Integrated step-by-step repair guides or video recommendations.
- A centralized platform to connect car owners with local mechanics.

These gaps highlight the need for an all-in-one solution accessible through smartphones without additional hardware.

5. Role of AI in Vehicle Diagnostics

Artificial Intelligence (AI) has transformed many fields, including automotive diagnostics. Image recognition can be used to identify dashboard symbols, while sound analysis using machine learning can detect abnormal engine noises and classify them. Such capabilities have yet to be widely implemented in consumer-level mobile applications.

5.1 Mobile Application Design for Automotive Use

Effective mobile app design in the automotive context must consider user experience, simplicity, and clear communication. Studies show that apps designed with intuitive interfaces and minimal technical jargon improve user trust and engagement. Furthermore, incorporating repair guides and visuals reduces perceived complexity and supports decision-making.

5.2 Mechanic Integration in Mobile Platforms

Mechanics remain an essential part of the diagnostic and repair process. Some platforms, like RepairPal, focus on connecting users to trusted mechanics but do not integrate real-time diagnostics. Providing mechanics with a dedicated space within an app to list their services and contact details can enhance accessibility and business opportunities.

6. Summary

The literature reveals a gap in fully integrated, AI-supported mobile diagnostic applications that cater to both car owners and mechanics. CarCare aims to address this by combining image and sound diagnostics, user-friendly guidance, and mechanic connectivity in a single platform.

CHAPTER 3: ANALYSIS AND DESIGN

1. Introduction

This chapter describes the functional and non-functional requirements of the CarCare application, along with its design models and architectural diagrams that guided the implementation.

2. Requirement Analysis

2.1 Functional Requirements

- User registration and authentication.
- Scan dashboard lights using camera.
- Record and analyze engine sounds.
- Generate diagnostic results and suggestions.
- View diagnostic history.
- Connect to mechanic profiles.
- Mechanic login and profile management.

2.2 Non-Functional Requirements

- Usability: simple and intuitive interface.
- Scalability: capable of supporting many users.
- Security: user data protection and secure authentication.
- Performance: fast analysis and response times.

3. System Architecture

The CarCare system consists of a mobile client, backend server, AI engines, and a database. The mobile client captures user inputs, communicates with the backend, and displays results. The backend handles business logic, data storage, and AI model integration.

4. Context Diagram

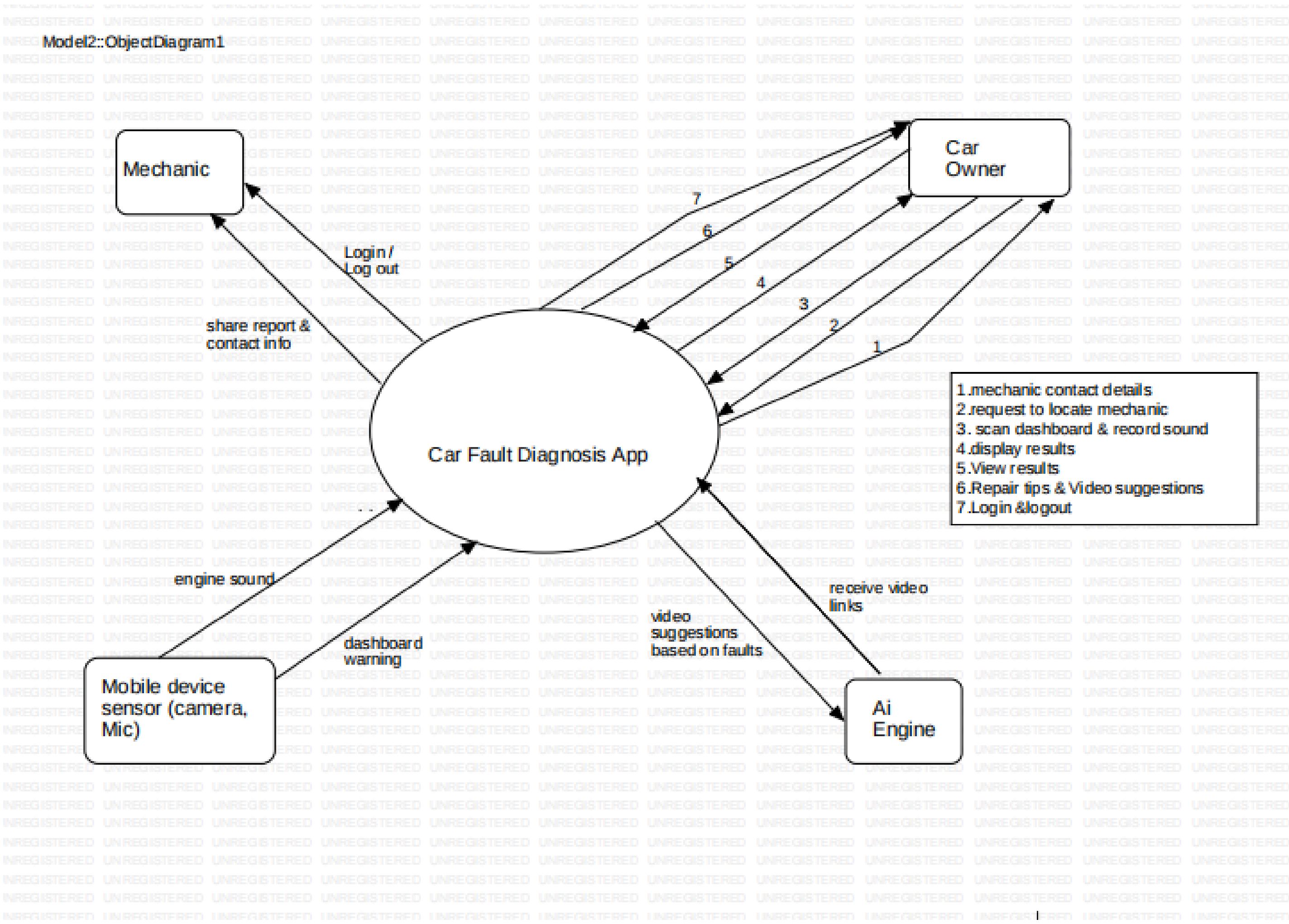


Figure 1: Context Diagram

The context diagram provides a high-level overview of external interactions, showing how car owners and mechanics interact with the system, and how external services like the AI engine and YouTube API support it.

5. Data Flow Diagrams

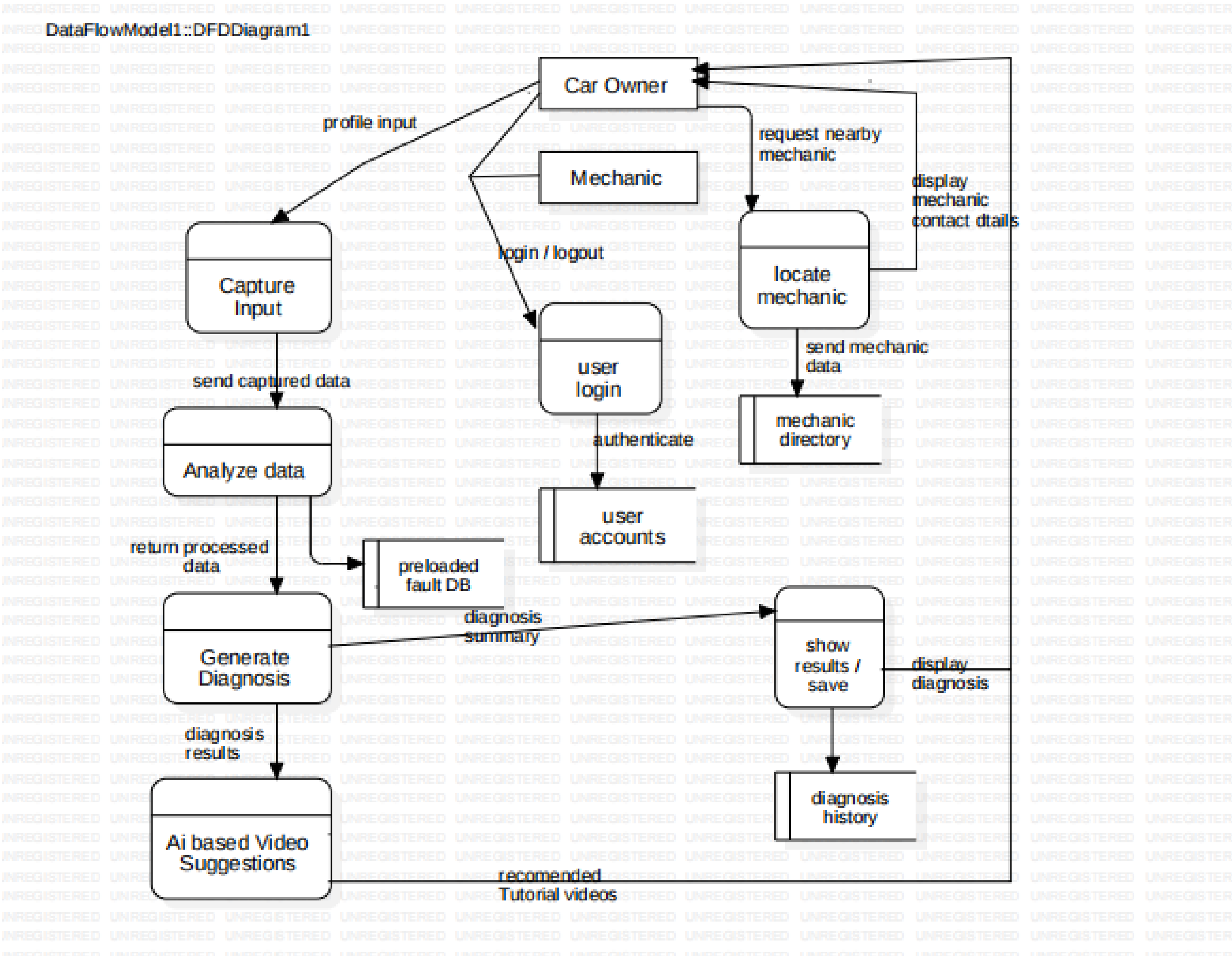


Figure 2: Data Flow Diagram

The data flow diagrams detail how data moves from user inputs to the system processes and outputs, including fault analysis and report generation.

6. Use Case Diagram

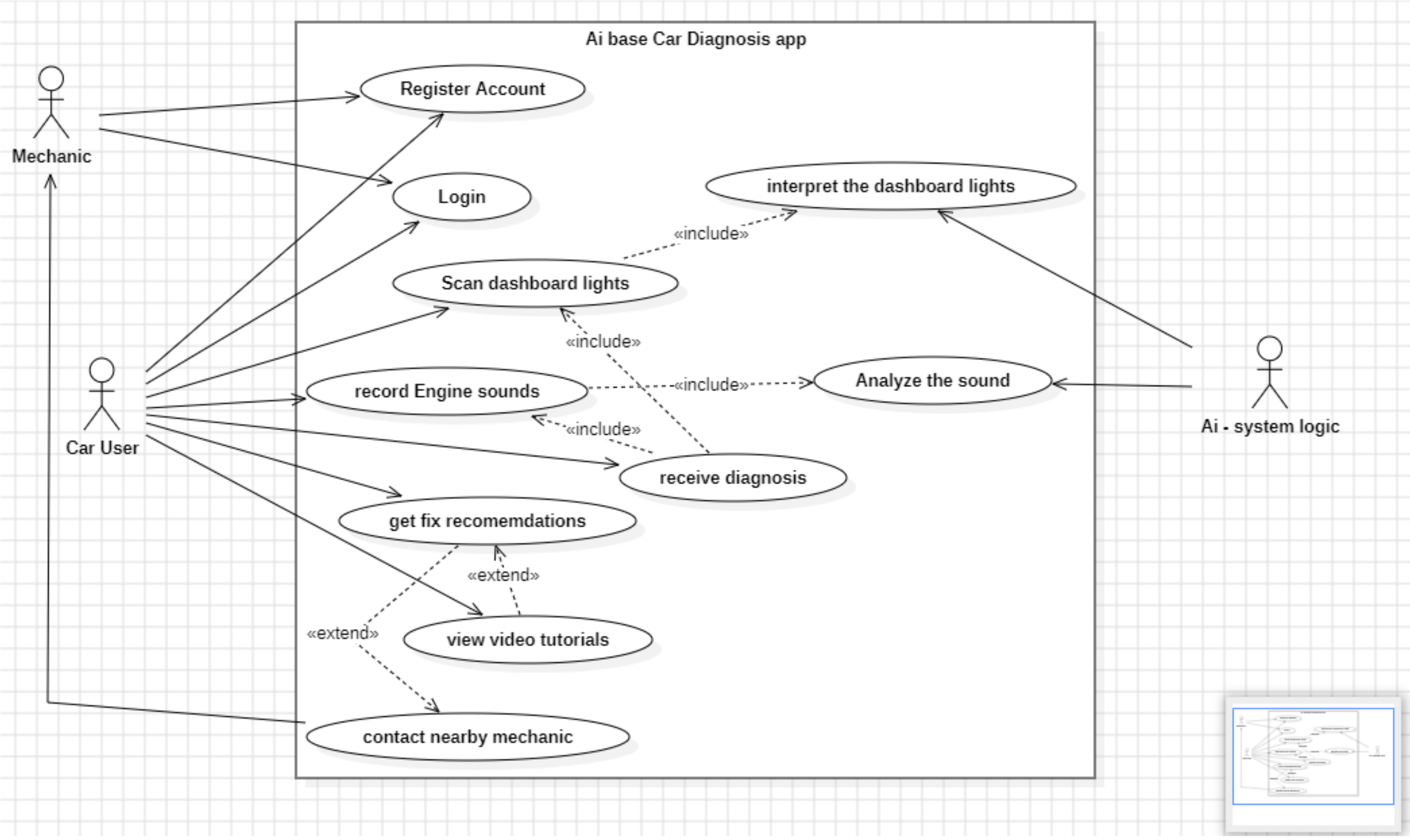


Figure 3: Use Case Diagram

The use case diagram depicts user interactions such as scanning, recording sounds, viewing history, and managing mechanic profiles.

7. Sequence Diagrams

7.1 Scan Dashboard Light and Receive Diagnosis

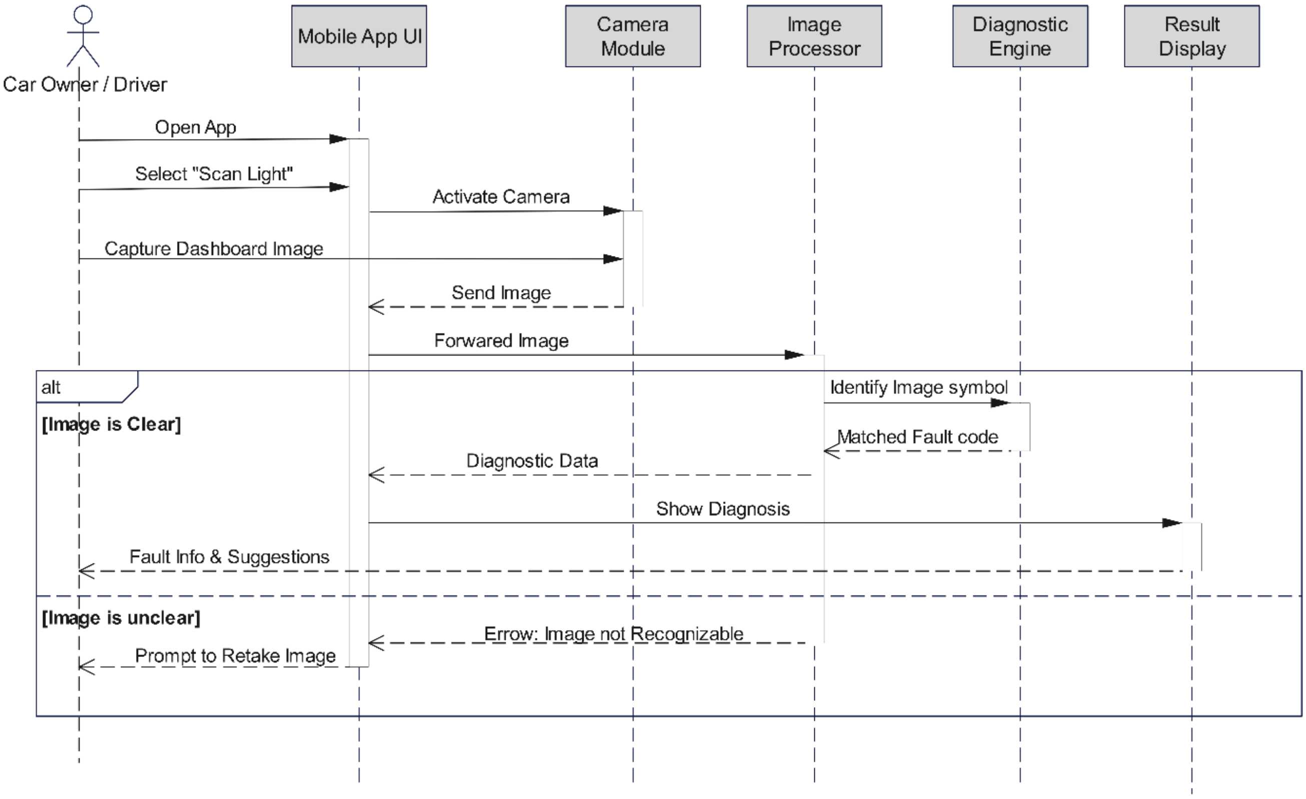


Figure 4: Scan Dashboard Light Sequence Diagram

7.2 Record Engine Sound and Receive Diagnosis

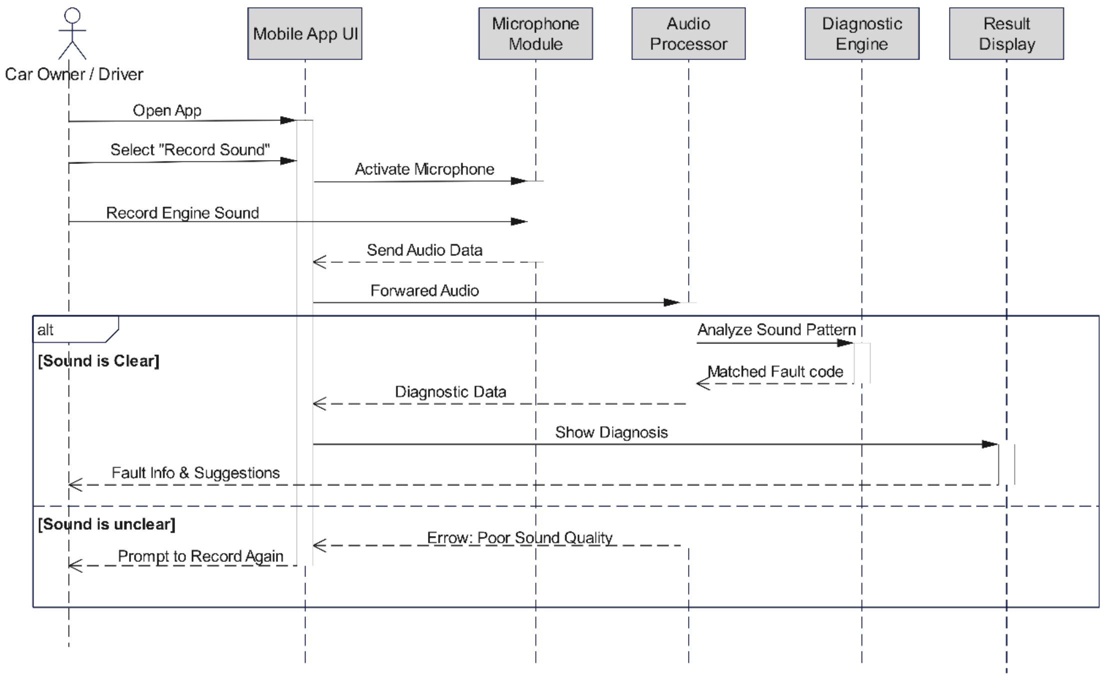


Figure 5: Record Engine Sound Sequence Diagram

7.3 Login, Logout, and Sign Up

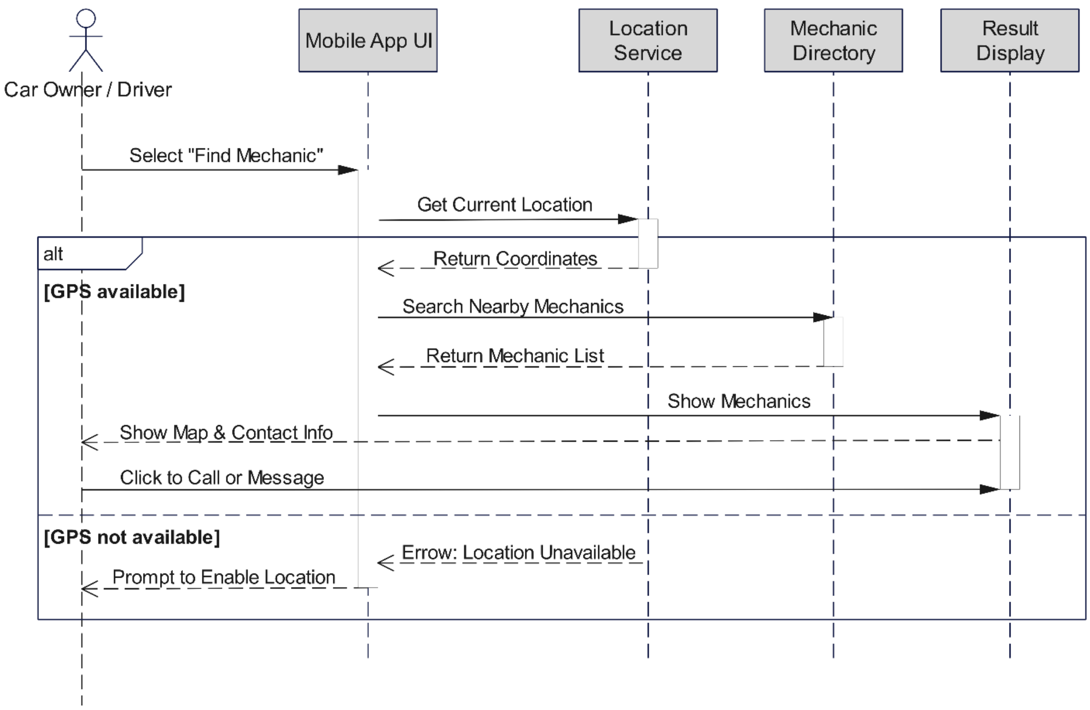


Figure 6: Login, Logout and Signup Sequence Diagram

7.4 Locate and Contact Mechanic

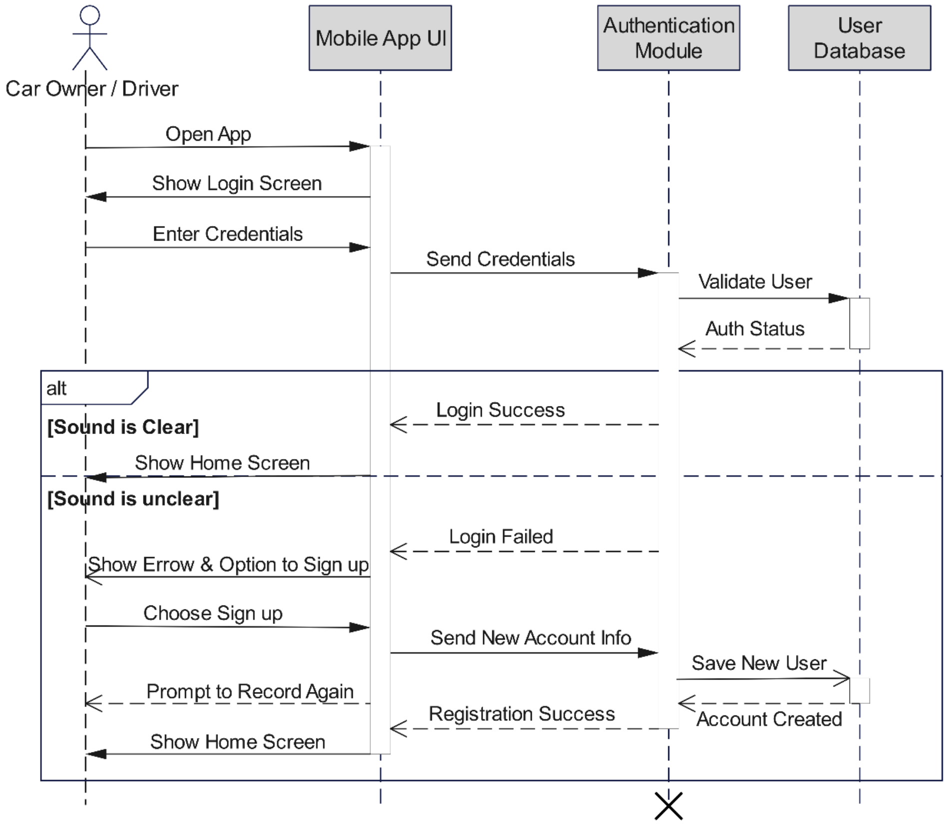


Figure 7: Locate Mechanic Sequence Diagram

Sequence diagrams illustrate the step-by-step interactions during scanning, sound recording, and tutorial video recommendation processes.

8. Class Diagram

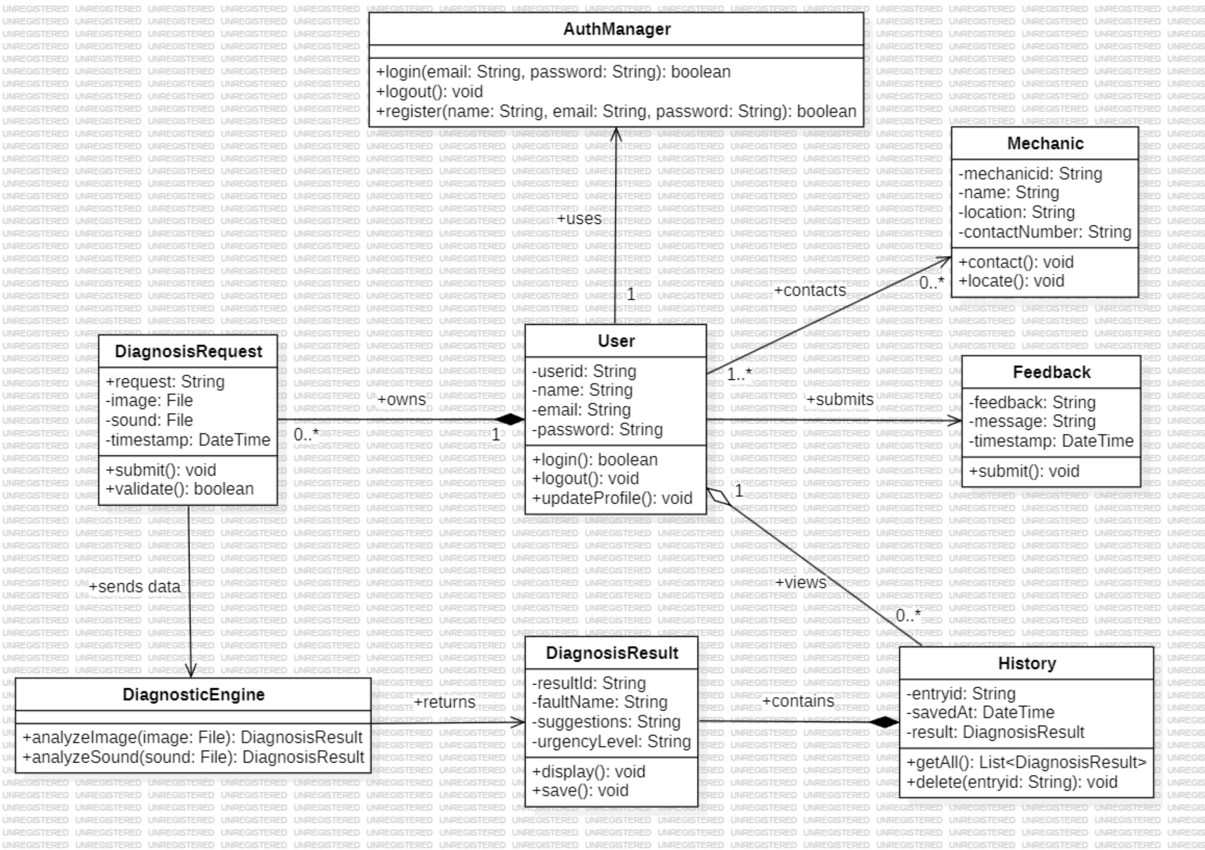


Figure 8: Class Diagram

The class diagram shows the static structure of the application, including classes like User, Vehicle, Diagnostic, and MechanicProfile, and their relationships.

9. Entity-Relationship Diagram

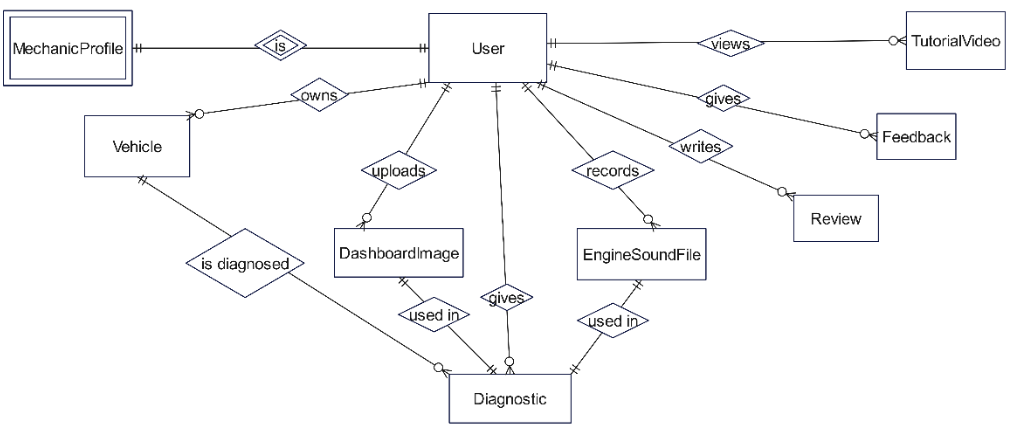


Figure 9: Entity Relationship Diagram

The ER diagram defines the database structure and relationships, forming the basis for backend implementation.

10. UI Design and Mockups

The application design includes ten main screens with a clean, teal and white theme. Wireframes and mockups were created to ensure usability and consistency.

11. Summary

The design phase established a clear roadmap for implementation, ensuring the system is robust, user-friendly, and technically feasible.

CHAPTER FOUR: IMPLEMENTATION AND RESULTS

1. Introduction

The Car Fault Diagnosis App is a comprehensive mobile solution built with React Native and Expo, designed to assist car owners and mechanics in diagnosing vehicle faults using dashboard light images and engine sound recordings. The app leverages modern mobile device capabilities to capture, upload, and analyze media, providing users with actionable diagnostic results and maintenance recommendations. Its intuitive interface, robust error handling, and modular architecture make it suitable for both technical and non-technical users.

The motivation for this project stems from the need for accessible, rapid, and accurate vehicle diagnostics, especially for users without deep automotive knowledge. By leveraging AI-powered backend services and the ubiquity of smartphones, the app bridges the gap between car owners and professional diagnostics, reducing maintenance costs and improving road safety.

2. Tools and Materials Used

- **React Native**: Framework for building native mobile apps using JavaScript and React.
- **Expo:** Platform and set of tools for rapid React Native development, including build, deployment, and device APIs.
- **TypeScript**: Superset of JavaScript for static typing and improved code reliability.
- **Axios**: HTTP client for making API requests to backend services.
- **react-native-fs:** Library for advanced file system access and file uploads.
- **expo-av:** Expo module for audio recording and playback.
- **expo-image-picker**: Expo module for capturing and selecting images from the device.
- **VS Code:** Code editor used for development.
- **Custom Backend API**: Receives and processes uploaded images and audio, returning diagnostic results (not included in this repository).
- **Node.js & npm:** For managing dependencies and running scripts.
- **Figma:** For UI/UX prototyping and design.
- **Git:** For version control and collaboration.
- **Android Studio:** For device emulation and testing.

3. Description of Implementation Process

3.1 Project Initialization

The project was bootstrapped using Expo CLI with a TypeScript template. The folder structure was organized into `components`, `screens`, `utils`, `context`, and `assets` for maintainability. Initial dependencies were installed, and the project was configured for both Android and iOS platforms.

3.2 UI/UX Design

Modern, responsive layouts were created for all screens, including splash, login, signup, home, dashboard scan, sound recording, results, and history. Custom components (headers, banners, metrics, quick actions) were developed for consistency and reusability. Figma was used for prototyping, ensuring a user-centric design with accessibility in mind.

3.3 Media Capture

Integrated `expo-image-picker` for dashboard light image capture and selection, and `expo-av` for high-quality engine sound recording. Permission handling and user feedback were implemented for a seamless experience. The app provides clear prompts and error messages if permissions are denied or if the device does not support certain features.

3.4 Data Upload

Developed robust upload functions using Axios for FormData-based uploads and `react-native-fs` for direct file uploads, ensuring compatibility across Android and iOS. Special care was taken to handle file URIs, MIME types, and backend requirements. The upload logic was abstracted for easy maintenance and future extension.

3.5 Backend Integration

The app communicates with a custom backend API for both image and audio diagnosis. API endpoints are configured in a central location, and all requests include error handling to provide clear feedback to users. The backend is responsible for running machine learning models that analyze the uploaded media and return structured diagnostic results.

3.6 Result Handling

Diagnostic results are parsed and presented in visually distinct cards, with icons, color coding, and severity indicators. Users can view detailed explanations and recommended actions. The app also supports history tracking, allowing users to review past diagnoses and monitor trends over time.

3.7 History and Notifications

Implemented history tracking for past diagnoses and a notification system with badges and modals for important updates. The notification system is designed to be extensible, supporting both local and push notifications in future versions.

3.8 Testing and Refinement

The app was tested on both Android and iOS devices. User feedback was incorporated to improve navigation, error handling, and overall usability. Automated and manual testing ensured that all features work as expected across different devices and OS versions.

4. Deployment Notes

- Ensure your backend API is deployed and accessible from mobile devices (consider CORS, HTTPS, and authentication).
- For production, configure environment variables securely and use Expo's EAS for managed builds.
- Test the app on real devices for best results, as emulators may not fully replicate hardware features.

5. Presentation and Interpretation of Results

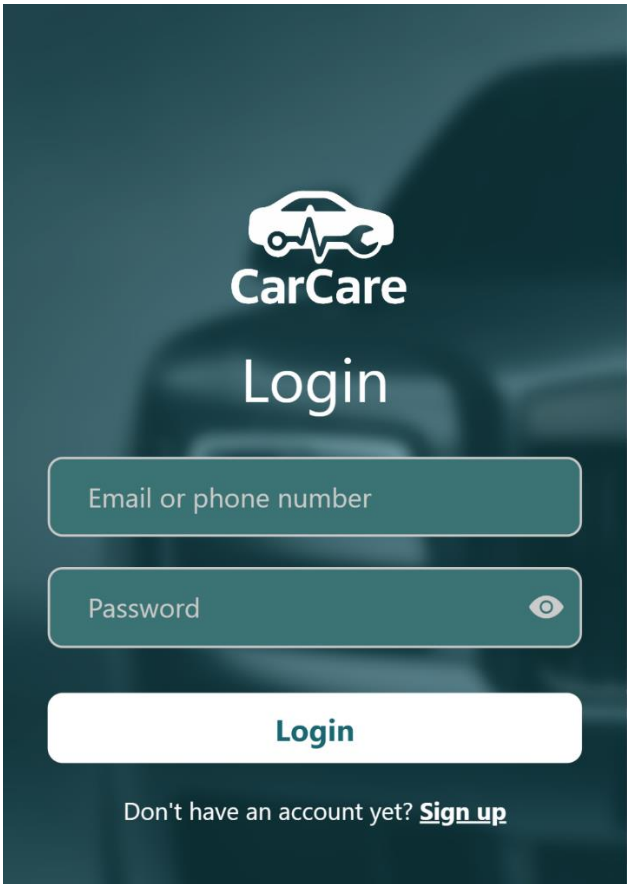
This section is designed to include annotated screenshots of the app's major screens. For each screen, a description is provided, and space is reserved for inserting screenshots in the final document.

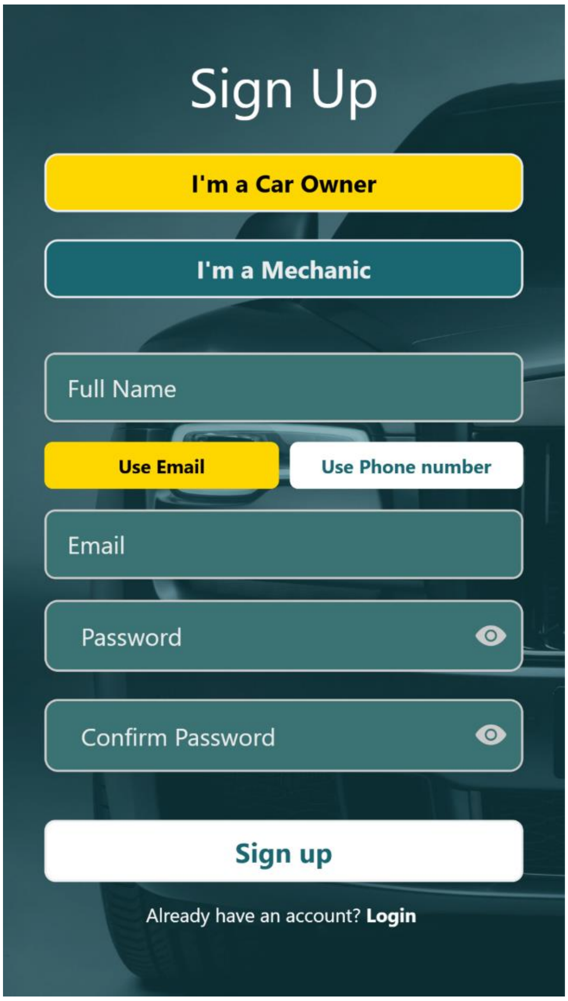
5.1 Splash Screen

Description: The splash screen welcomes users with the app logo and a modern background, setting the tone for a professional experience.

5.2 Login and Signup Screens

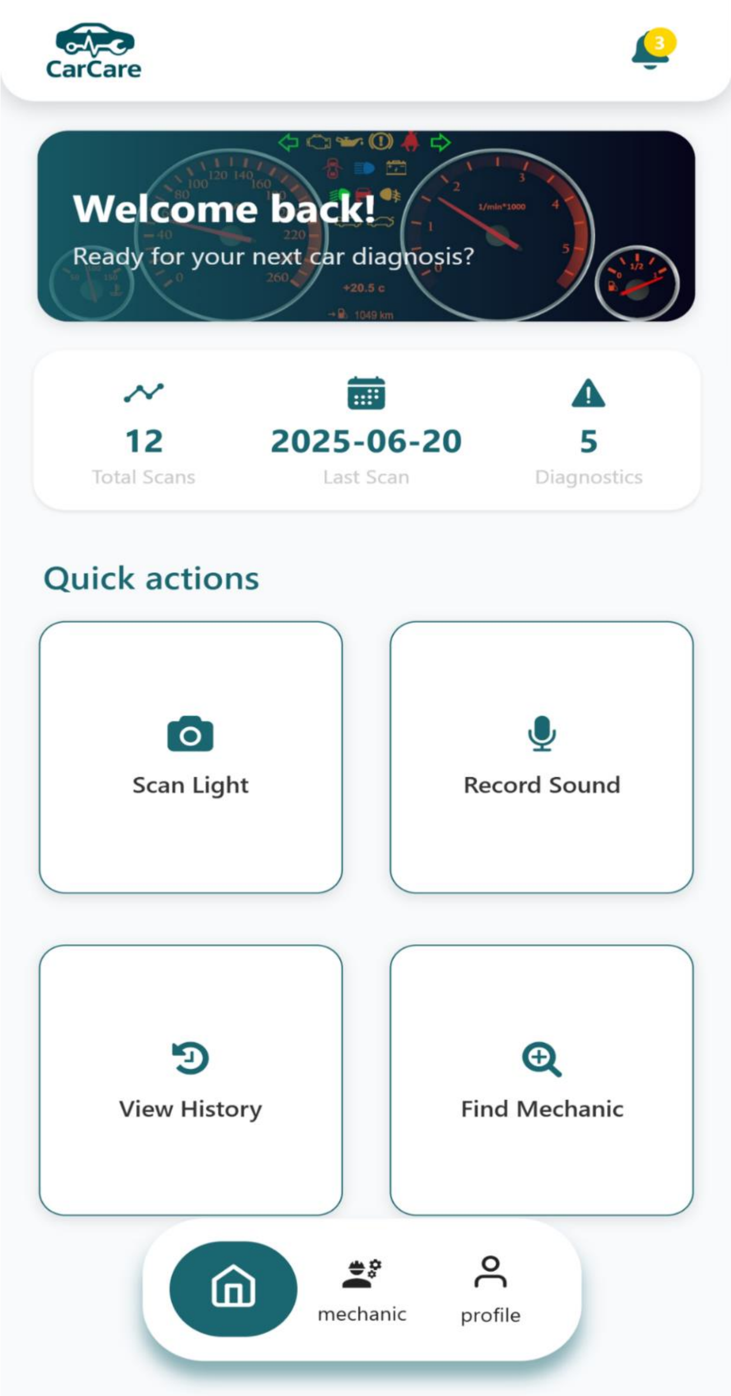
Description: These screens allow users to securely log in or create a new account. The forms are scrollable, visually appealing, and provide clear feedback for errors or missing information.





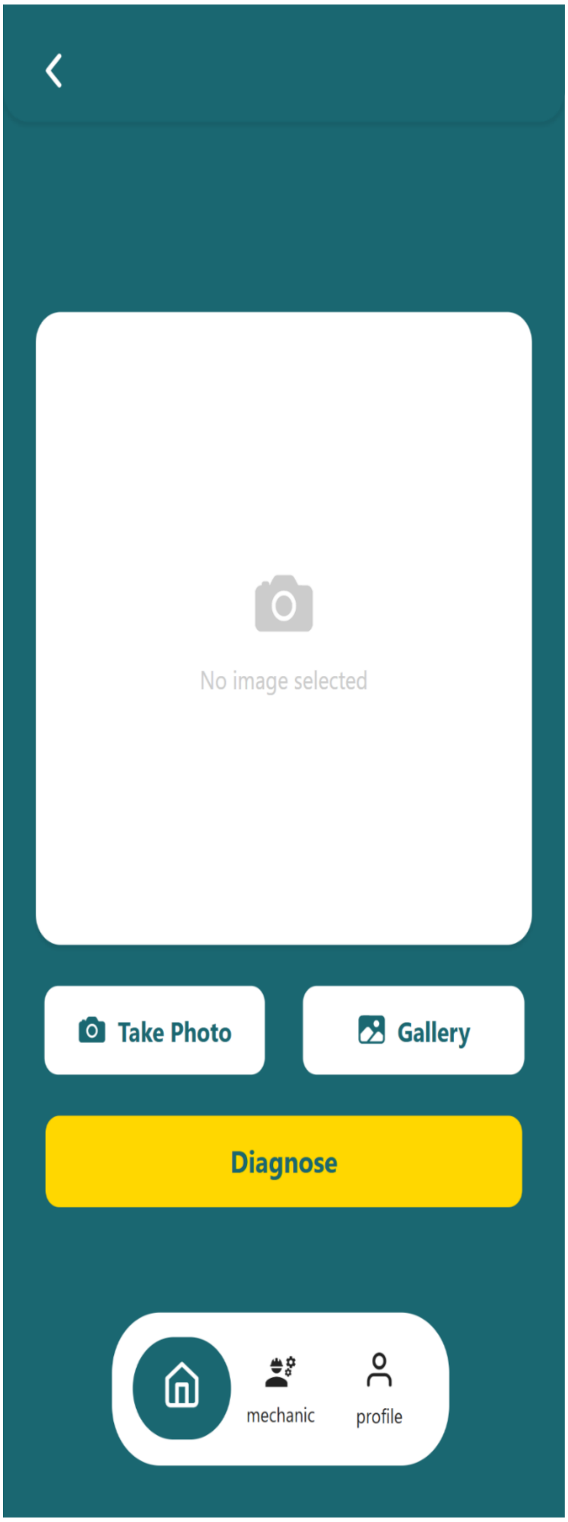
5.3 Home Screen

Description: The home screen features a modern header, user metrics, quick action buttons, and a notification badge. It serves as the central hub for navigation.



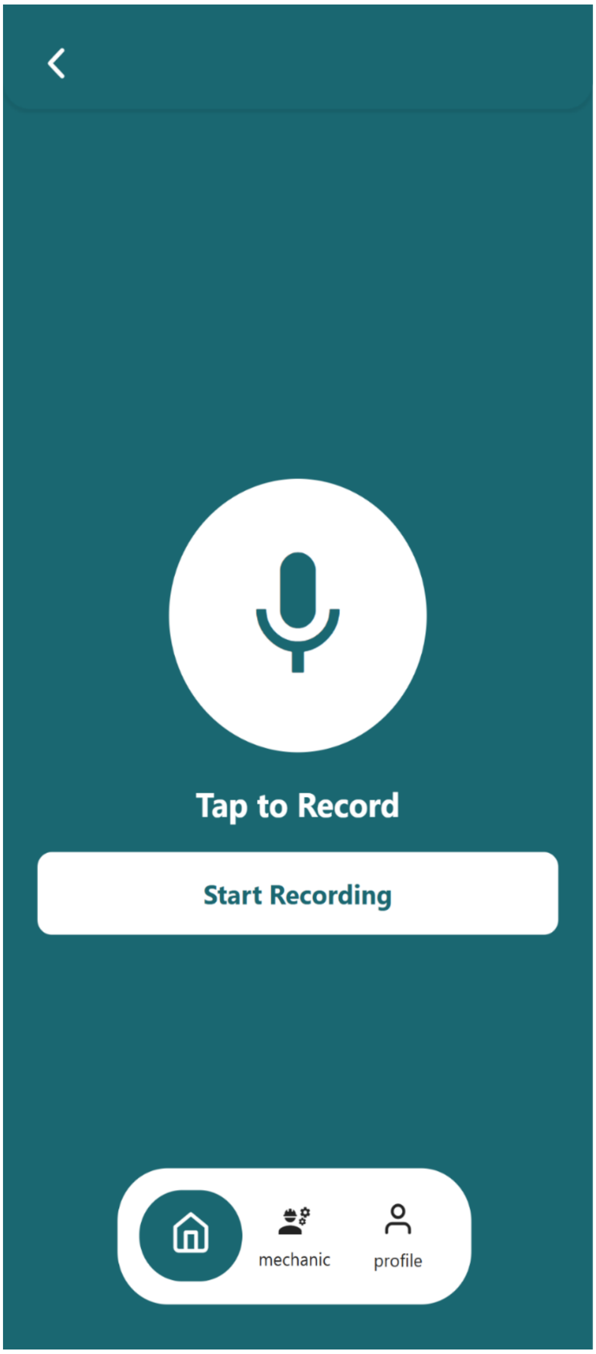
5.4 Dashboard Scan Screen

Description: Users can capture or select a dashboard light image, preview it, and submit it for diagnosis. The layout is clean, with prominent action buttons and a clear call to action.



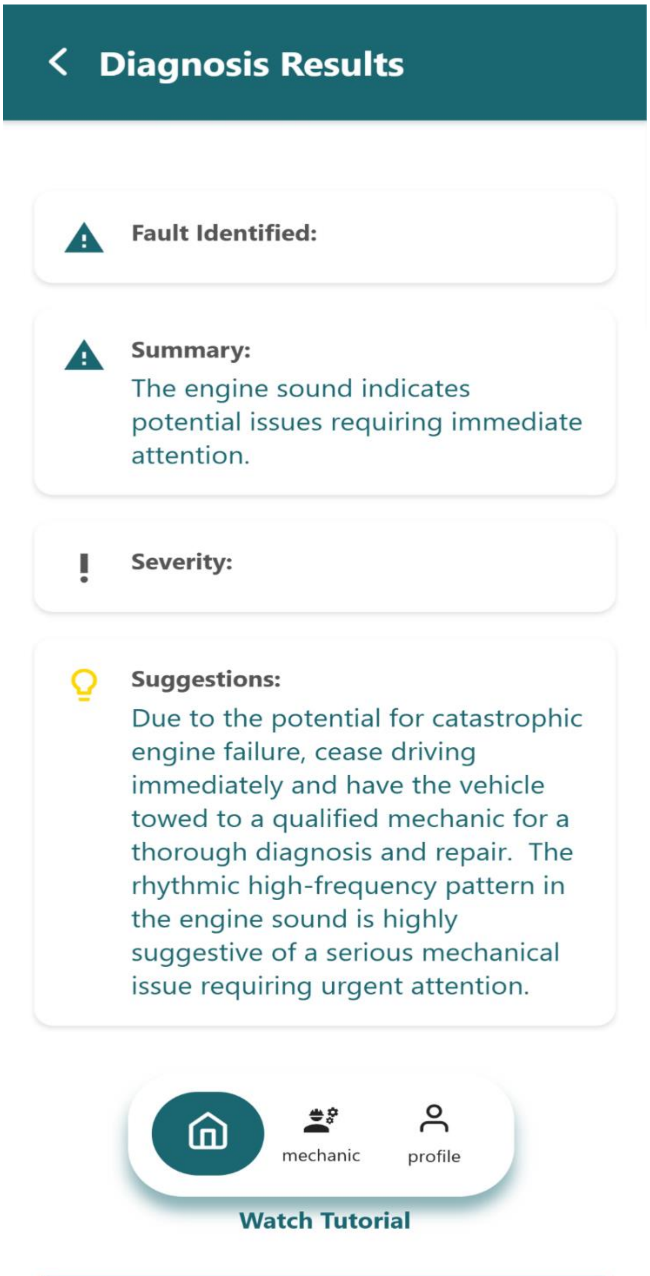
5.5 Record Sound Screen

Description: This screen enables users to record engine sounds, play them back, and submit for diagnosis. The UI features a circular mic card, glowing effects, and intuitive controls.



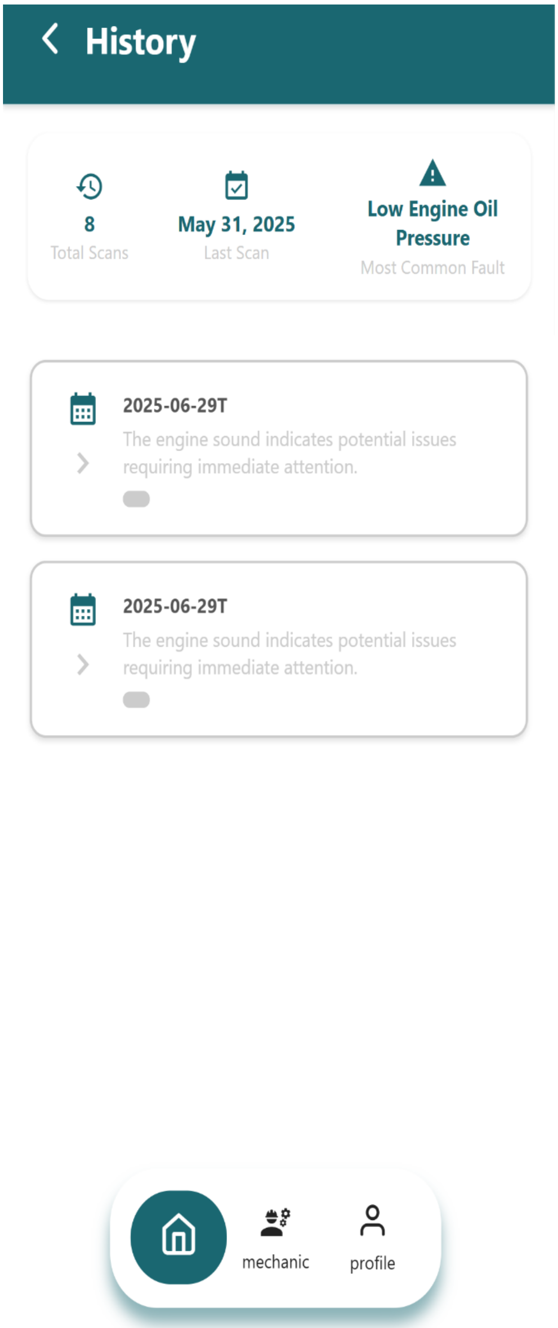
5.6 Diagnostic Result Screen

Description: Diagnostic results are presented in a card layout, with icons, color-coded severity, and detailed explanations. Users receive actionable recommendations based on the diagnosis.



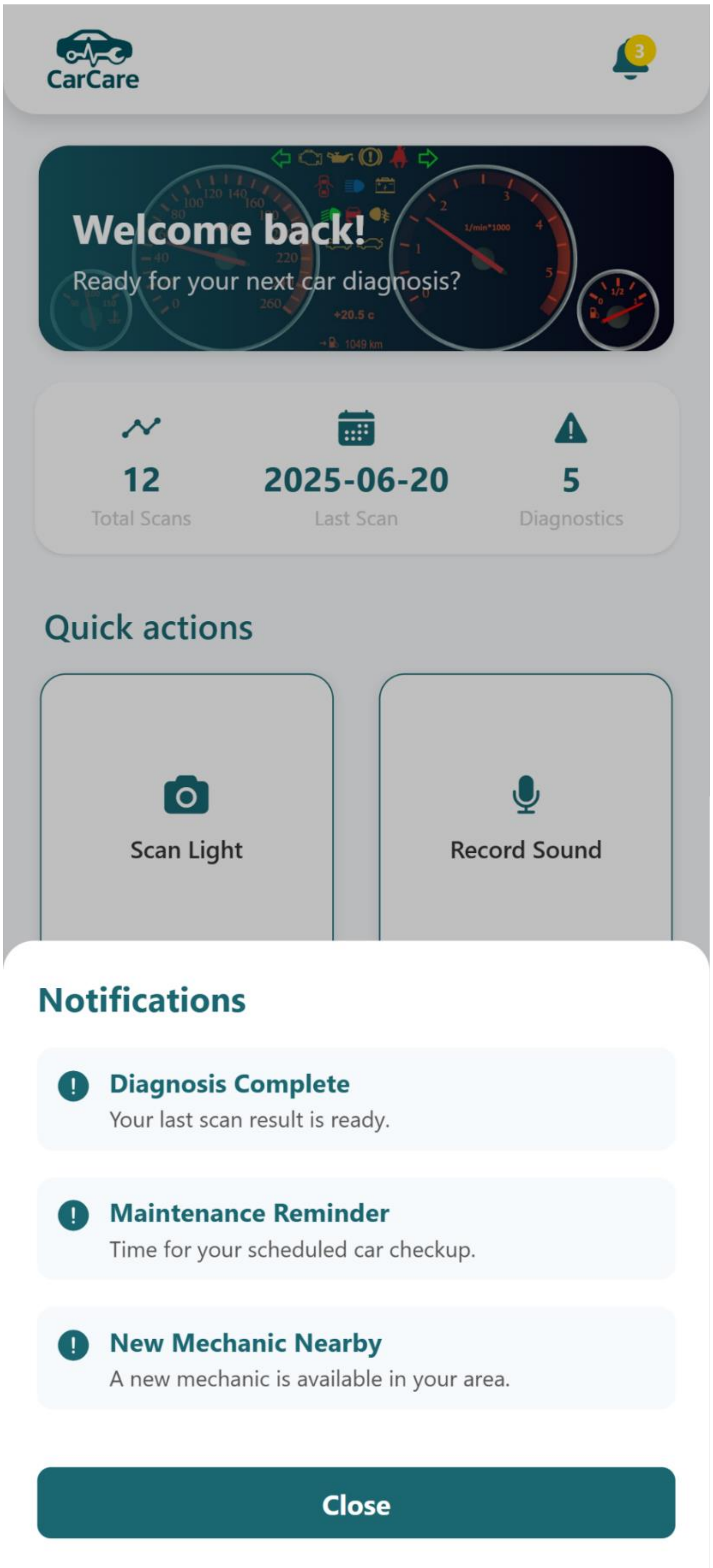
5.7 History Screen

Description: Users can view a chronological list of past diagnoses, including images, audio, and results. This helps track recurring issues and monitor vehicle health over time.



5.8 Notifications

Description: The app provides notification badges and modals for important updates, reminders, and new results. Notifications are designed to be non-intrusive but informative.



6. Evaluation of Solution (Project)

- **Strengths:**
  - Modern, intuitive, and visually appealing UI/UX.
  - Reliable media capture and upload, with comprehensive error handling.
  - Clear, actionable diagnostic results and history tracking.
  - Modular, maintainable codebase with reusable components.
  - Flexible upload logic (supports both Axios and react-native-fs).
  - Designed for extensibility and future feature additions.

- **Limitations:**

- Dependent on backend API quality, uptime, and model accuracy.
- Diagnosis is only as good as the backend's machine learning models.
- Some advanced features (e.g., real-time push notifications, offline support) are not fully implemented.
- Requires device permissions for camera, microphone, and storage.
- UI/UX may require further localization for global deployment.

7. Summary

The Car Fault Diagnosis App demonstrates how modern mobile technology can empower users to quickly and accurately identify vehicle issues. Its robust implementation, user-friendly design, and flexible architecture make it a valuable tool for both car owners and mechanics. Future improvements could include deeper integration with vehicle telematics, enhanced AI models, real-time notifications, and expanded support for additional vehicle systems. The current solution lays a strong foundation for ongoing development and real-world impact.

CHAPTER FIVE: CONCLUSION AND FURTHER WORKS

1. Summary of Findings

The goal of this project was to design and implement a user-friendly mobile application — **CarCare** — that assists car owners in diagnosing vehicle faults using image and sound analysis, while providing a platform for mechanics to share their services.

The application successfully integrates camera-based dashboard light detection and microphone-based engine sound analysis, providing immediate and clear feedback to users. Additional features such as step-by-step repair instructions and AI-recommended tutorial videos were incorporated to empower users in resolving minor faults themselves.

Furthermore, the inclusion of mechanic profiles allows car owners to connect with professional services when self-repair is not feasible, bridging a major gap present in many existing solutions.

2. Contribution to Engineering and Technology

The CarCare app represents a significant advancement in the integration of mobile sensing technologies and AI for automotive diagnostics. While existing solutions primarily rely on OBD-II scanners and specialized hardware, this project demonstrates that smartphones alone can offer meaningful, accessible diagnostic support to non-technical users.

Key contributions include:

- **Image-based dashboard fault recognition**: Using smartphone cameras to scan and interpret warning lights, making fault detection more approachable.
- **Sound analysis for engine diagnostics**: Leveraging microphones to detect abnormal sounds, an area often ignored in existing tools.
- **Unified user experience**: Combining diagnostics, repair guidance, and mechanic connectivity in a single mobile platform.
- **Mechanic service integration**: Providing mechanics with visibility and facilitating easier service booking, enhancing their reach.

Through these contributions, the project showcases how AI and mobile technology can simplify car maintenance, support preventive repairs, and ultimately improve road safety.

3. Recommendations

To further improve the CarCare application, the following recommendations are proposed:

- Expand the fault code database to cover more vehicle makes and models, including localized or region-specific issues.

- Improve AI models by collecting and incorporating more diverse training data from real-world environments, especially under varied noise conditions.
- Develop multilingual support to increase accessibility for non-English-speaking users.
- Introduce a live chat or video consultation feature to allow real-time interactions with mechanics.
- Strengthen data security measures to ensure user privacy and build trust.

4. Difficulties Encountered

Several challenges were faced during the development and implementation of CarCare:

- **Data availability**: Limited access to real engine sound datasets and comprehensive dashboard light images restricted the scope of AI model training.
- **Variability in phone hardware**: Differences in camera and microphone quality across devices affected consistency in diagnostics.
- **User privacy concerns**: Ensuring secure storage and management of sensitive user and vehicle data required careful design and additional security measures.
- **Integration complexities**: Combining AI modules, backend services, and frontend interfaces required extensive testing and fine-tuning.

Despite these challenges, the project met its core objectives and successfully delivered a functional prototype.

5. Further Works

Future work on CarCare can focus on the following directions:

- **Enhanced AI capabilities**: Incorporate deep learning approaches for more accurate and robust fault detection, including fine-grained sound analysis and advanced symbol recognition.
- **IoT integration**: Connect the app directly with in-vehicle systems and sensors for real-time monitoring without user input.
- **Predictive maintenance features**: Use collected data to predict potential failures before they occur, supporting proactive servicing.
- **Expanded mechanic network**: Develop partnerships with certified workshops and official service centers to provide verified repair options.
- **Augmented reality (AR) guidance**: Use AR to visually guide users through repairs, improving usability and engagement.
- **Scalability improvements**: Optimize infrastructure to support large-scale deployments and a growing user base globally.

Conclusion

The CarCare project offers a promising, innovative solution to common vehicle diagnostic challenges faced by car owners, while also supporting mechanics in expanding their services. With further development and research, it has the potential to significantly impact automotive maintenance practices and contribute to safer, more efficient vehicle ownership.