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

Task 4: System Modeling and Design

Project Title: **AI Powered Car Fault Diagnosis Mobile Application**

PREPARED BY: GROUP10

ABANDA SERGIO ABANDA	FE22A131
ABILATEZIE VIN-WILSON ANU	FE22132
AGBOR EMMANUEL NCHENGE	FE22A158
ASHLEY TAN WACHE	FE22A154
ASHU DESLEY	FE22A155

CEF440: **INTERNET PROGRAMMING AND MOBILE PROGRAMMING**

INSTRUCTOR: **Dr. NKEMENI VALERY**

Date: **May 26, 2025**

Table of Contents

Introduction

The increasing complexity of modern vehicles has made traditional car diagnosis both time-consuming and inaccessible for many drivers. As a response to this challenge, the CarDiagApp an AI-driven mobile application was developed to empower users to identify and resolve car issues more efficiently using dashboard imagery, engine sounds, and cloud-based machine learning. This report provides a comprehensive overview of the system's architecture, including its Context Diagram, Data Flow Diagram, Use Case Models, Sequence Diagrams, Class Diagrams, and Deployment Architecture. By leveraging intuitive visual tools and clear documentation, the report aims to guide stakeholders through the application's functionality and its underlying structure.

Aim of the Report

The primary aim of this report is to:

- Present a structured understanding of how the CarDiagApp interacts with external entities and manages internal processes.
- Visualize the flow of information through diagrams that depict system behavior, roles, and responsibilities.
- Provide a technical blueprint for developers and system architects to follow during implementation and scaling.
- Ensure alignment with stakeholder expectations and requirements by mapping out all core functionalities clearly and concisely.

1. CONTEXT DIAGRAM

1.1Definition of a Context Diagram

A Context Diagram is a high-level data flow diagram (Level 0 DFD) that provides a graphical overview of a system and its interaction with external entities. It visualizes the boundaries of the system, showing what the system interacts with but not how it works internally. It is particularly useful for both technical and non-technical stakeholders to understand the scope of the system.

1.2. Meaning and Components of a Context Diagram

A context diagram outlines the **external perspective** of a system. It depicts how external agents (e.g., users, APIs, services) **exchange data** with the system without diving into internal operations.

Component	**Description**
System (Process):	The central component representing the system being developed or studied.
External Entities:	People, systems, or devices outside the system that interact with it.
Data Flows (Arrows):	Arrows show how data moves to and from the system and external entities.

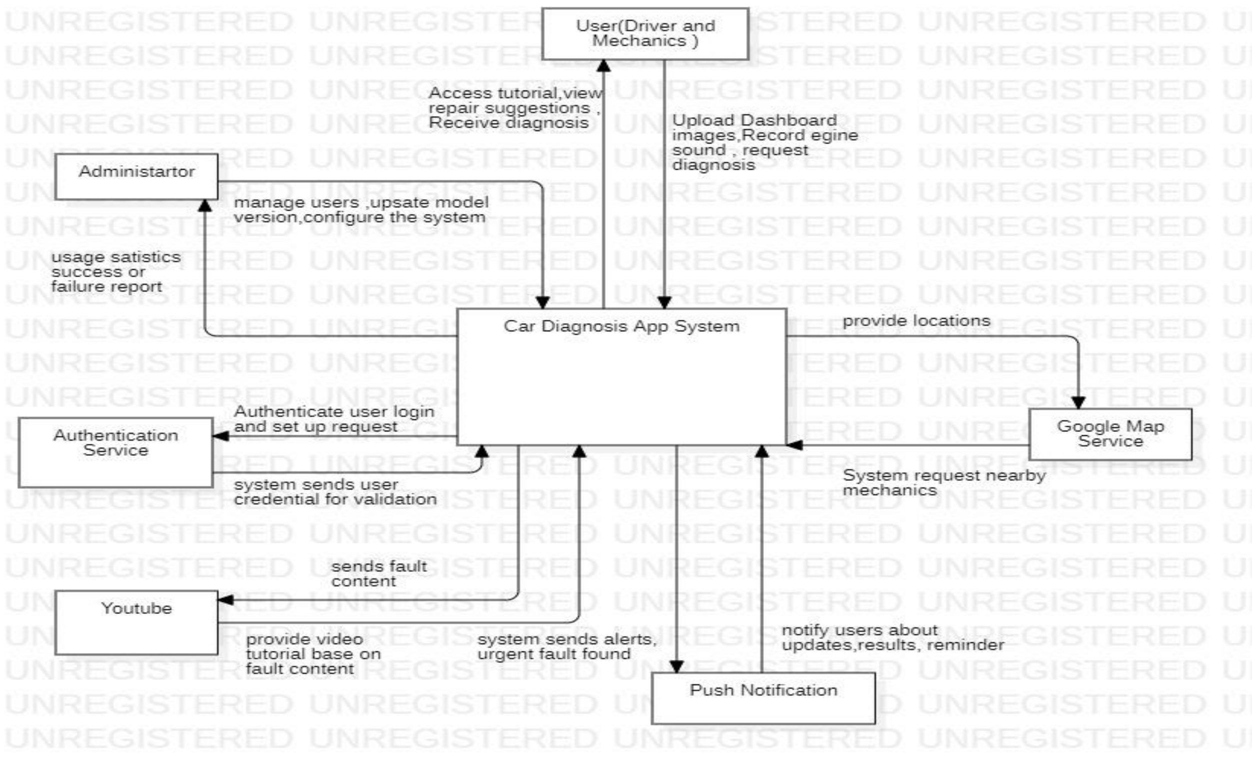

User(Driver and Mechanics)
Access tutorial,view repair suggestions , Receive diagnosis
Upload Dashboard images,Record egine sound , request diagnosis
Administartor
manage users ,upsate model version,configure the system
usage satistics success or failure report
Car Diagnosis App System
provide locations
Google Map Service
System request nearby mechanics
Authentication Service
Authenticate user login and set up request
system sends user credential for validation
sends fault content
Youtube
provide video tutorial base on fault content
system sends alerts, urgent fault found
notify users about updates,results, reminder
Push Notification

Figure 1: context Diagram

1.3 Description of Each Part of the Context Diagram (Car Diagnostic App)

System: Car Diagnostic Application

This is the main mobile and backend system where users:

- Upload images and audio
- Receive diagnostics
- View tutorials

It communicates with both users and multiple external services to function effectively

External Entities and Their Roles

1. **Administrator**
    - Manages user accounts and system configurations.
    - Monitors system health and performance.
2. **User (Driver/Mechanic)**
    - Provides input (dashboard images, engine sounds).
    - Receives diagnosis reports and repair suggestions.
3. **Google Maps Service**
    - Supplies location-based data for nearby repair centers or mechanic shops.
4. **Push Notification Service**
    - Sends alerts to users (e.g., diagnosis completed, urgent issues found).
5. **YouTube**
    - Provides relevant tutorial videos for detected car issues
6. **Authentication Service**: Manages user login and security validation.

2. DATA FLOW DIAGRAM COMPONENTS

2.1 Definition of a Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation used to visualize how data flows within a system. It illustrates how data enters a system, how it is processed through various functions or processes, where it is stored, and how it exits the system. DFDs are widely used in system analysis and design to map out the flow of information, helping developers and stakeholders understand how data moves and transforms within the system.

2.2. Meaning and Components of a Data Flow Diagram

A DFD provides a structured visualization of the movement and transformation of data through a system’s components. It focuses on:

- What happens to data (input, processing, output)
- Where data is stored
- Which entities interact with the system

Components of a Data Flow Diagram

Component	Symbol	Description
Process	rounded rectangle	Represents an operation or function that transforms input data into output.
Data Store	Open-ended rectangle	Represents where data is held for use by one or more processes.
External Entity	Rectangle	Represents people, systems, or organizations that interact with the system.
Data Flow	Arrow	Represents the movement of data between processes, data stores, and entities.

2.3 Description of the data flow diagram for Car Diagnosis APP

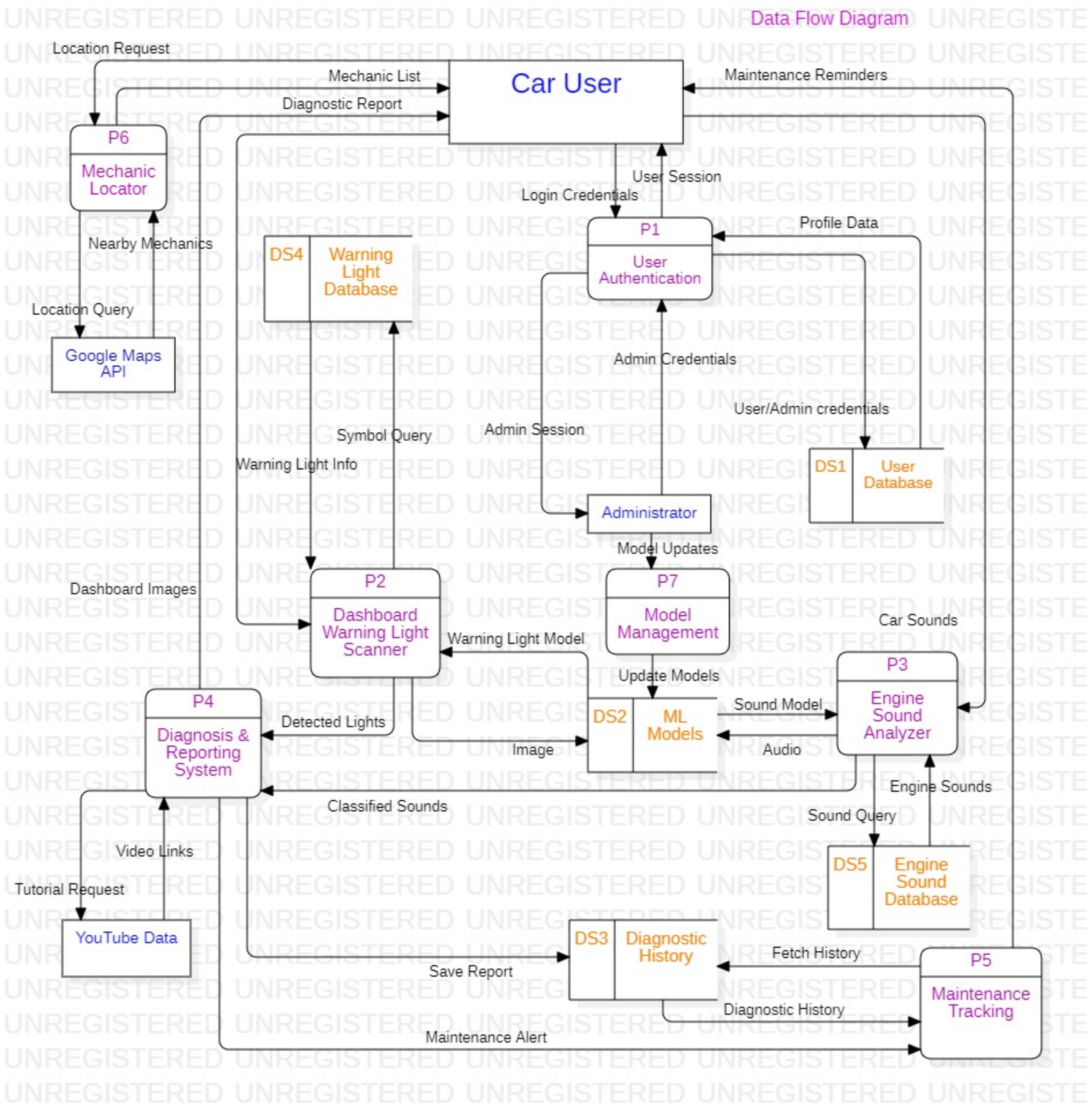


Figure 2: data flow diagram

2.3.1 External Entities

Car User

- Provides login credentials, dashboard images, and engine sounds
- Receives diagnostic reports, maintenance reminders, and mechanic locations

Administrator

- Provides admin credentials and system commands
- Receives admin reports and updates ML models

External APIs (Google Maps, YouTube, Authentication Service)

2.2 Processes

P1: User Authentication

- Handles registration, login, and session management
- Interfaces with the Authentication Service and User Profiles data store

P2: Dashboard Warning Light Scanner

- Processes dashboard images to identify warning lights
- Uses ML models and warning light database for interpretation

P3: Engine Sound Analyzer

- Analyzes recorded engine sounds to identify potential issues
- Uses ML models and sound classification database

P4: Diagnosis & Reporting System

- Combines results from scanners and analyzers to generate comprehensive reports
- Interfaces with YouTube API for tutorial content
- Stores reports in diagnostic history

P5: Maintenance Tracking

- Manages diagnostic history and generates maintenance reminders
- Accesses diagnostic history data store

P6: Mechanic Locator

- Processes location requests to find nearby mechanics
- Interfaces with Google Maps API

P7: Model Management

- Handles updates to machine learning models
- Updates the ML Models data store

2.3 Data Stores

DS1: User Profiles

- Stores user authentication data and preferences
- Accessed by authentication and admin processes

DS2: ML Models

- Stores machine learning models for image and sound analysis
- Used by the scanner and analyzer processes

DS3: Diagnostic History

- Stores past diagnostic reports and maintenance records
- Accessed by reporting and maintenance tracking processes

DS4: Warning Light Database

- Contains information about dashboard warning lights and their meanings
- Referenced by the dashboard scanner process

DS5: Engine Sound Database

- Contains reference patterns for engine sounds and their associated issues
- Referenced by the engine sound analyzer process

2.4 Key Data Flows

- **Authentication flows:** credentials, tokens, session.
- **Diagnostic flows:** images, sounds, classifications, reports
- **Maintenance flows:** history, reminders, alerts
- **External API interactions**: location queries, tutorial requests
- **Admin flows:** model updates, system management

The data flow diagram illustrates how information moves through the system to fulfill the functional requirements specified in the SRS, particularly focusing on the core diagnostic features and supporting functionality.

3. USE CASE DIAGRAM

3.1 Introduction to Use Case Diagrams

A use case diagram is a behavioral UML diagram that visually represents the interactions between users (actors) and a system to achieve specific goals. It depicts the functional requirements of a system from an external perspective, focusing on what the system does rather than how it does it. Use case diagrams consist of actors (users or external systems), use cases (functions or services), and relationships between them.

3.2 Importance of Use Case Diagrams for CarDiagApp

For the AI-based Car Fault Diagnosis Mobile Application (CarDiagApp), use case diagrams are particularly important for several reasons:

1. **Stakeholder Communication**: The diagram provides a clear, visual representation of system functionality that can be understood by both technical and non-technical stakeholders, including developers, car users, and mechanics.
2. **Scope Definition**: It helps define the boundaries of the application, clarifying what features are included (e.g., dashboard light scanning, engine sound analysis) and what is excluded (e.g., direct OBD-II connection).
3. **Role Clarification**: With multiple user types specified in the SRS (car users, mechanics, content creators, administrators), the use case diagram clearly delineates which functions are available to each user type.
4. **Requirements Validation**: It serves as a visual validation tool to ensure all functional requirements from the SRS are properly captured and assigned to appropriate actors.
5. **Development Planning**: The development team can use the diagram to plan implementation phases, prioritize features, and organize testing efforts.

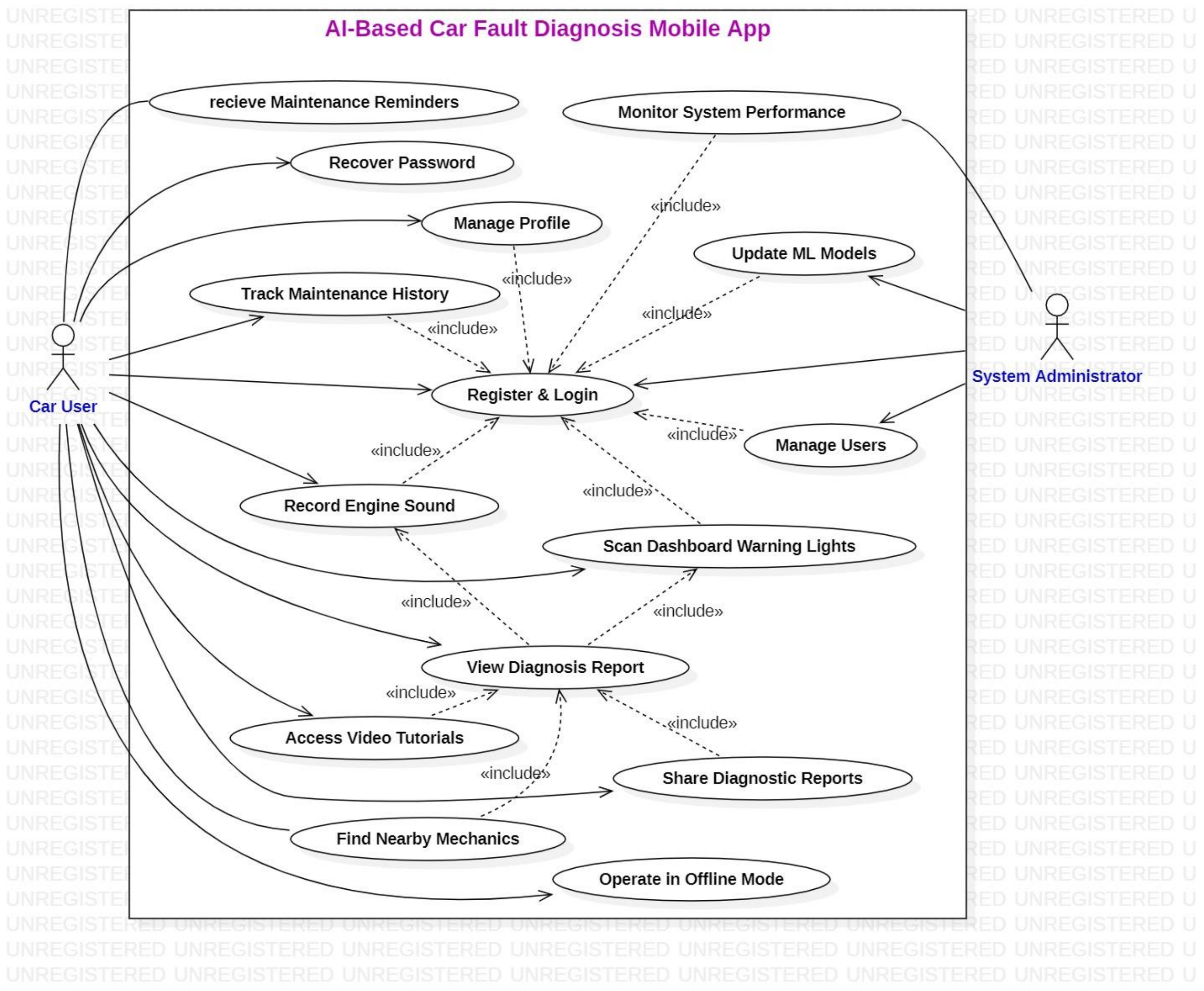


Figure 3: Use case diagram

3.3 Detailed Use Cases Descriptions

UC1: Register & Login

Attribute	**Description**
Use Case ID	UC1
Use Case Name	Register & Login
Actors	Car User, Administrator

Attribute	**Description**
Description	Allows users to create accounts, verify email, login with credentials, and recover forgotten passwords
Preconditions	User has installed the application
Main Flow	1. User opens application 2. User selects "Register" option 3. User provides email, password, and selects role 4. System sends verification email 5. User verifies email 6. User completes profile setup 7. User can subsequently login with credentials
Alternative Flow	- User selects "Login" and enters credentials - User selects "Forgot Password" and follows recovery process
Postconditions	User is authenticated and granted role-based access
Requirements	FR1, FR2, FR3, FR4

UC2: Scan Dashboard Lights

Attribute	**Description**
Use Case ID	UC2
Use Case Name	Scan Dashboard Lights
Actors	Car User
Description	Captures and interprets dashboard warning light icons using the device camera
Preconditions	-User is authenticated -Camera permission is granted -Adequate lighting conditions
Main Flow	1. User selects "Scan Dashboard" from home screen 2. Camera interface appears with framing guide

Attribute	Description
	3. User captures dashboard image 4. System processes image and identifies warning lights 5. System highlights identified symbols on image 6. User confirms or retakes image 7. System proceeds to diagnosis report
Alternative Flow	- No lights detected: System suggests improving lighting or angle - Camera permission denied: System explains requirement and requests again - Processing error: System suggests retry with better conditions
Postconditions	Dashboard warning lights are identified and ready for diagnosis report
Requirements	FR5, FR6, FR15

UC3: Record Engine Sound

Attribute	Description
Use Case ID	UC3
Use Case Name	Record Engine Sound
Actors	Car User
Description	Records and analyzes engine sounds to identify potential mechanical issues
Preconditions	-User is authenticated -Microphone permission is granted -Relatively quiet environment
Main Flow	1. User selects "Record Engine Sound" from home screen 2. System provides recording instructions 3. User positions device appropriately 4. User starts recording (5-10 seconds) 5. System processes audio 6. System classifies engine sounds

Attribute	Description
	7. System proceeds to diagnosis report
Alternative Flow	- Microphone permission denied: System explains requirement and requests again - Recording too noisy: System suggests quieter environment - Recording too short: System requests longer recording
Postconditions	Engine sound issues are identified and ready for diagnosis report
Requirements	FR7, FR8

UC4: View Diagnosis Report

Attribute	Description
Use Case ID	UC4
Use Case Name	View Diagnosis Report
Actors	Car User
Description	Displays comprehensive diagnostic information based on warning lights and/or engine sounds
Preconditions	User has completed either dashboard scan or engine sound recording or both
Main Flow	1. System combines results from warnings and/or sounds 2. System determines issues and urgency levels 3. System displays detailed report with explanations 4. System suggests potential solutions and repair tips 5. System displays relevant video tutorials 6. System saves report to history automatically
Alternative Flow	- No issues detected: System shows all-clear message - Indeterminate diagnosis: System suggests professional inspection - Critical issue detected: System highlights urgency with alert
Postconditions	User is informed about car issues with explanation and suggestions

Attribute	Description
	Report is saved to history
Requirements	FR9, FR16, FR20

UC5: Access Video Tutorials

Attribute	Description
Use Case ID	UC5
Use Case Name	Access Video Tutorials
Actors	Car User
Description	Provides relevant video content about diagnosed issues
Preconditions	-User has viewed a diagnosis report, -Internet connection available
Main Flow	1. User views diagnosis report 2. System displays relevant tutorial links 3. User selects video tutorial 4. System loads and plays selected tutorial
Alternative Flow	- No internet connection: System displays cached tutorials if available - No relevant tutorials found: System shows general maintenance videos
Postconditions	User gains visual understanding of the issue and repair process
Requirements	FR10

UC6: Track Maintenance History

Attribute	Description
Use Case ID	UC6
Use Case Name	Track Maintenance History
Actors	Car User

Attribute	Description
Description	Shows chronological list of past diagnostics with details
Preconditions	-User is authenticated, -User has previous diagnostic reports
Main Flow	1. User selects "History" from home screen 2. System displays chronological list of past diagnostics 3. User selects specific report 4. System displays detailed view of selected report
Alternative Flow	- No history available: System shows empty state with explanation - Filter by date: User applies date range filter - Filter by issue type: User filters by specific problems
Postconditions	User can view comprehensive maintenance history
Requirements	FR11

UC7: Receive Maintenance Reminders

Attribute	Description
Use Case ID	UC7
Use Case Name	Receive Maintenance Reminders
Actors	Car User
Description	Gets notifications based on fault urgency levels
Preconditions	-User has notification permissions enabled, -User has diagnosed issues with urgency ratings
Main Flow	1. System analyzes fault urgency 2. System schedules reminders based on urgency: - Red (immediate): Immediate alert - Yellow (soon): 1-hour alert - Monitoring: 2-hour alert

Attribute	**Description**
	3. System delivers notifications at scheduled times
Alternative Flow	- User dismisses notification: System reschedules as appropriate - User acts on notification: System updates reminder status
Postconditions	User is reminded of maintenance needs based on urgency
Requirements	FR12

UC8: Share Diagnostic Reports

Attribute	**Description**
Use Case ID	UC8
Use Case Name	Share Diagnostic Reports
Actors	Car User
Description	Enables sharing of diagnostic results with others
Preconditions	-User has completed diagnosis, -Share permission granted
Main Flow	1. User views diagnosis report 2. User selects "Share" option 3. System prepares shareable report 4. User selects sharing method (email, messaging, etc.) 5. System shares report via selected method
Alternative Flow	- User exports as PDF: System generates downloadable report - Share directly with others: System prepares specialized format
Postconditions	Diagnostic report is shared with selected recipient
Requirements	FR17

UC9: Find Nearby Mechanics

Attribute	**Description**
Use Case ID	UC9
Use Case Name	Find Nearby Mechanics
Actors	Car User
Description	Locates mechanics in the vicinity using map integration
Preconditions	-User has completed diagnosis -Internet connection available -Location permission granted (optional)
Main Flow	1. User selects "Find Nearby Mechanics" option 2. System requests location permission if not already granted 3. System displays map with mechanics in vicinity 4. User can filter results by distance, services, or ratings 5. User selects a mechanic to view details 6. System shows contact information and directions
Alternative Flow	- Location permission denied: System allows manual location entry - No mechanics found: System expands search radius - User shares report with selected mechanic
Postconditions	User can contact or navigate to appropriate mechanic
Requirements	FR21, FR22

UC10: Manage Users

Attribute	**Description**
Use Case ID	UC10
Use Case Name	Manage Users
Actors	Administrator
Description	Allows administrators to view, modify, and manage user accounts
Preconditions	-User is authenticated as Administrator,

Attribute	Description
	-Internet connection available
Main Flow	1. Administrator accesses admin dashboard 2. Administrator selects user management section 3. System displays list of registered users 4. Administrator can view, edit, or disable user accounts 5. System applies and saves changes
Alternative Flow	- Search for specific user - Filter users by role or status - Reset user password - Configure role permissions
Postconditions	User accounts are managed according to administrative actions
Requirements	FR18

UC11: Update ML Models

Attribute	Description
Use Case ID	UC11
Use Case Name	Update ML Models
Actors	Administrator
Description	Enables pushing updates to the machine learning models
Preconditions	-User is authenticated as Administrator, -Internet connection available, -New model versions are ready
Main Flow	1. Administrator accesses model management section 2. System displays current model versions and statistics 3. Administrator selects new model version to deploy 4. System validates and stages the update 5. Administrator confirms rollout schedule

Attribute	Description
	6. System deploys updates to users
Alternative Flow	- Rollback to previous version - Staged rollout to percentage of users - A/B testing of model versions
Postconditions	ML models are updated to improve diagnostic accuracy
Requirements	FR14

UC12: Monitor System Performance

Attribute	Description
Use Case ID	UC12
Use Case Name	Monitor System Performance
Actors	Administrator
Description	Provides analytics and metrics on system usage and performance
Preconditions	-User is authenticated as Administrator, -Internet connection available
Main Flow	1. Administrator accesses system dashboard 2. System displays key performance metrics 3. Administrator can view detailed reports on: - Model accuracy statistics, - User engagement metrics, - Error rates and types, - Usage patterns 4. Administrator can export reports
Alternative Flow	- Filter data by date range - Configure alert thresholds - Investigate specific issues

Attribute	**Description**
Postconditions	Administrator has insights into system performance and usage
Requirements	Inferred from admin role

UC13: Recover Password

Attribute	**Description**
Use Case ID	UC13
Use Case Name	Recover Password
Actors	Car User
Description	Enables users to reset their password when forgotten
Preconditions	- Registered email address, - Internet connection
Main Flow	1. User selects "Forgot Password" and provides registered email 2. System validates User input and checks for existing User 3. System generates password reset token and sends it to User’s registered email 4. User receives an email with password reset link or token 5. User reset password: The user clicks on the password reset link and provide a new password 6. System validate and update new password
Alternative Flow	None
Postconditions	User Can Log In: The user can log in with their new password.
Requirements	FR3

UC14: Operate in Offline Mode

Attribute	Description
Use Case ID	UC14
Use Case Name	Operate in Offline Mode
Actors	Car User
Description	Enables core diagnostic functionality without internet connection
Preconditions	-User has previously authenticated, -Required ML models are downloaded
Main Flow	1. System detects offline status 2. User can access core features: - Scan dashboard lights - Record engine sounds - View basic diagnostics - Access cached tutorials 3. System stores results locally
Alternative Flow	- Reconnection: System syncs local data when connection restored, - Limited functionality notification if models not pre-downloaded
Postconditions	User can diagnose issues without internet connectivity
Requirements	FR13

UC15: Manage Profile

Attribute	Description
Use Case ID	UC15
Use Case Name	Manage Profile
Actors	Car User, Administrator
Description	Allows users to update their profile information and preferences
Preconditions	-User is authenticated
Main Flow	1. User accesses profile section,

Attribute	Description
	2. System displays current profile information, 3. User modifies personal information, preferences, or settings, 4. System validates changes, 5. System saves updated profile
Alternative Flow	- Change password: System requires current password verification, - Update notification preferences, - Configure data usage settings
Postconditions	User profile is updated with new information
Requirements	FR19

3.4 Relationships between Use Cases

1. **Include Relationships**:
    - "View Diagnosis Report" includes "Scan Dashboard Warning Lights" and/or "Record Engine Sound"
    - "Share Diagnostic Reports" includes "View Diagnosis Report"
    - "Find Nearby Mechanics" includes "View Diagnosis Report"

These include relationships show that certain functionality depends on other functionality being executed first.

Use Case Access by Actor

The diagram illustrates which actors have access to which use cases:

1. **Car User**: Has access to all basic functions, including registration, login, diagnostics, maintenance tracking, and location services.
2. **System Administrator**: Focuses on backend system management, user administration, model updates and monitors performance with IT/ML expertise.

3.5 Technical Implementation Considerations

The use case diagram provides a foundation for architectural decisions:

1. **Component Structure**: The diagram suggests natural component boundaries (authentication system, image processing system, audio processing system, reporting system).
2. **Data Flow**: The include relationships indicate data flow between components, which influences the application architecture.
3. **Testability**: Each use case can be treated as a testable unit, facilitating the testing strategy outlined in Section 5.3.

The use case diagram for CarDiagApp clearly visualizes the functional scope of the application and the interactions between different user types and system features. It aligns with the SRS requirements and provides a solid foundation for subsequent development activities.
This diagram will serve as a reference for stakeholders throughout the project lifecycle, ensuring that development efforts remain focused on the intended functionality, and that testing verifies all required use cases. As development progresses, the diagram can be refined to reflect any changes in requirements or additional functionality identified during implementation.

4. SEQUENCE DIAGRAM

4.1 Definition of Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that shows how objects or components interact in a particular sequence to achieve a specific functionality. It illustrates the flow of messages between different entities over time, making it particularly useful for visualizing the order of operations in a system. Sequence diagrams are crucial for:

- Clarifying complex system interactions
- Identifying required components and their relationships
- Spotting potential issues in the workflow early
- Serving as a communication tool between developers and stakeholders
- Documenting system behavior for future reference

4.2 Sequence Diagrams in Car Diagnostic App Development

By providing these sequence diagrams before development begins, the team can:

1. Identify all necessary system components upfront

2. Spot potential integration challenges early
3. Estimate development effort more accurately
4. Maintain consistent understanding across the team
5. Reduce rework by clarifying requirements visually

The diagrams serve as both planning tools during development and reference documentation for future maintenance and updates.

4.3 Explanation of Sequence Diagrams and their overflow

4.3.1. User Registration & Login Flow

Objective: Allow users to create an account, verify their email, log in, and reset passwords.

Importance: This diagram clarifies the authentication workflow, showing exactly when system validation occurs and how errors are handled. It helps developers:

- Implement proper security measures at each step
- Design appropriate error messages
- Structure the verification email system
- Plan the database schema for user data storage
- Coordinate frontend and backend interactions

Registration Process

1. User Action: Selects "Register".
2. System Response: Displays a registration form.
3. User Input: Submits email, password, and role (e.g., "Car User").
4. System Action: Requests account creation and sends a verification email.
5. Email Verification: User clicks the verification link in the email, and the system confirms
6. Verification and prompts for profile setup.
7. Profile Completion: User submits profile details (e.g., car model, contact info), and the system saves data and confirms successful registration.

Login Process

1. User Action: Selects "Login".
2. System Response: Displays login form.

3. User Input: Enters credentials (email & password).
4. System Action: Verifies credentials.
5. If correct → Displays home screen.
6. If incorrect → Shows error message.

Forgot Password Flow

- User Action: Selects "Forgot Password".
- System Response: Requests email.
- User Input: Submits email.
- System Action: Sends a password reset link via email.
- User clicks the link and submits a new password.
- System updates the password and confirms the change.
- Redirects to the login screen.

Error Cases

- Invalid credentials → Login error.
- Email not found → Password reset failure.

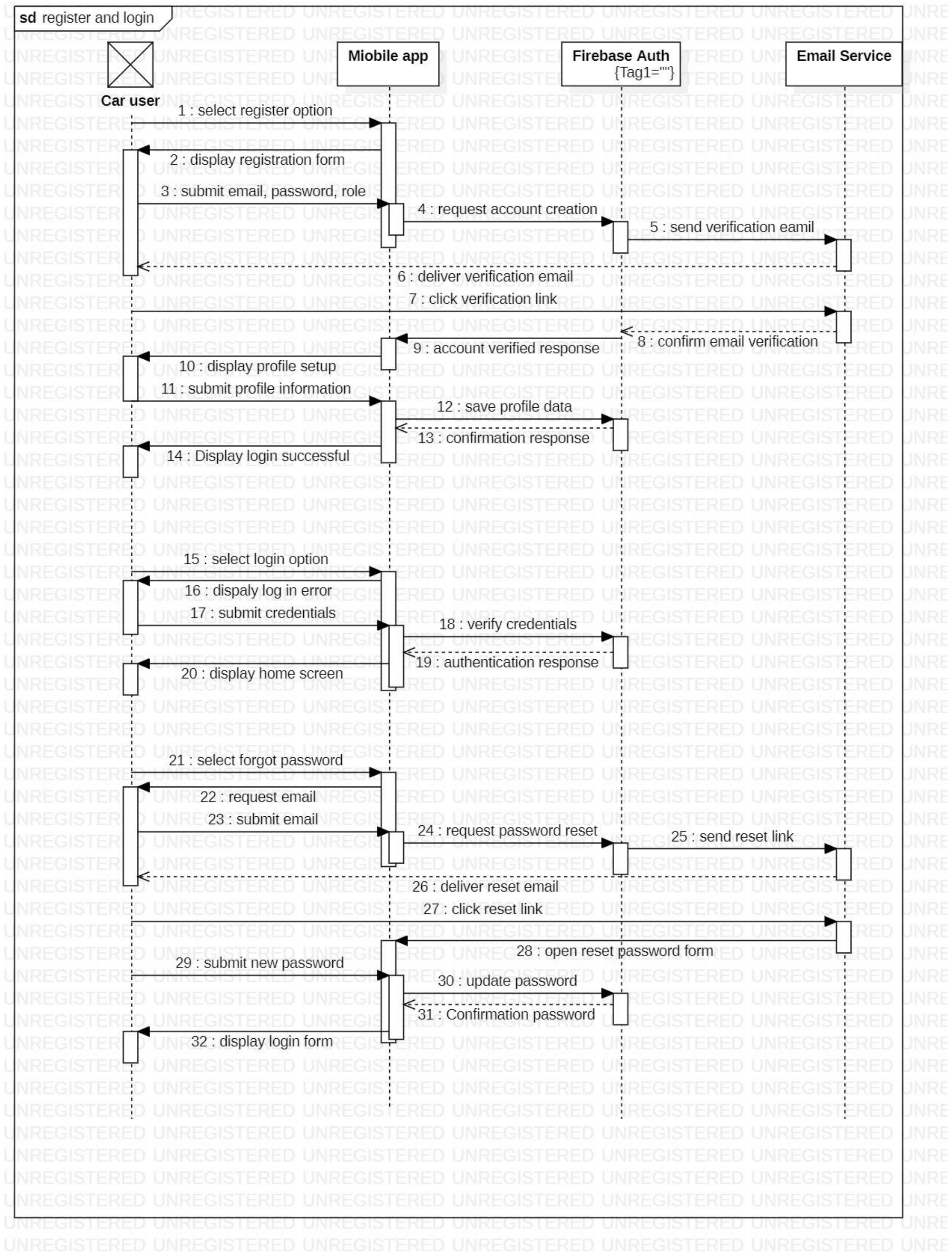

sd register and login
Car user
Miobile app
Firebase Auth {Tag1=""}
Email Service
1 : select register option
2 : display registration form
3 : submit email, password, role
4 : request account creation
5 : send verification eamil
6 : deliver verification email
7 : click verification link
8 : confirm email verification
9 : account verified response
10 : display profile setup
11 : submit profile information
12 : save profile data
13 : confirmation response
14 : Display login successful
15 : select login option
16 : dispaly log in error
17 : submit credentials
18 : verify credentials
19 : authentication response
20 : display home screen
21 : select forgot password
22 : request email
23 : submit email
24 : request password reset
25 : send reset link
26 : deliver reset email
27 : click reset link
28 : open reset password form
29 : submit new password
30 : update password
31 : Confirmation password
32 : display login form

Figure 4.1 sequence diagram for registration and login

4.3.2 Dashboard Lights Scanning Flow

Objective: Scan the car dashboard for warning lights and provide diagnostics.

Importance: This visual interaction diagram is essential for:

- Planning the camera integration and permission handling
- Designing the image processing pipeline
- Determining how to communicate with the computer vision component
- Structuring the diagnostic results display
- Handling edge cases like poor image quality

Successful Scan Process

1. User Action: Selects "Scan Dashboard".
2. System Action: Requests camera access and displays a live camera feed with framing guidance.
3. User Action: Captures an image of the dashboard.
4. System Action: Processes the image and identifies warning symbols, highlighting detected symbols.
5. User Action: Confirms the image.
6. System Action: Generates a diagnostic report and suggests next steps (e.g., repairs).

Alternative Scenarios

- No Lights Detected: System displays "No issues found".
- Permission Denied (Camera Access Blocked): System explains why camera access is needed.
- Retry Suggestion: If image is unclear, prompts user to retake the photo.

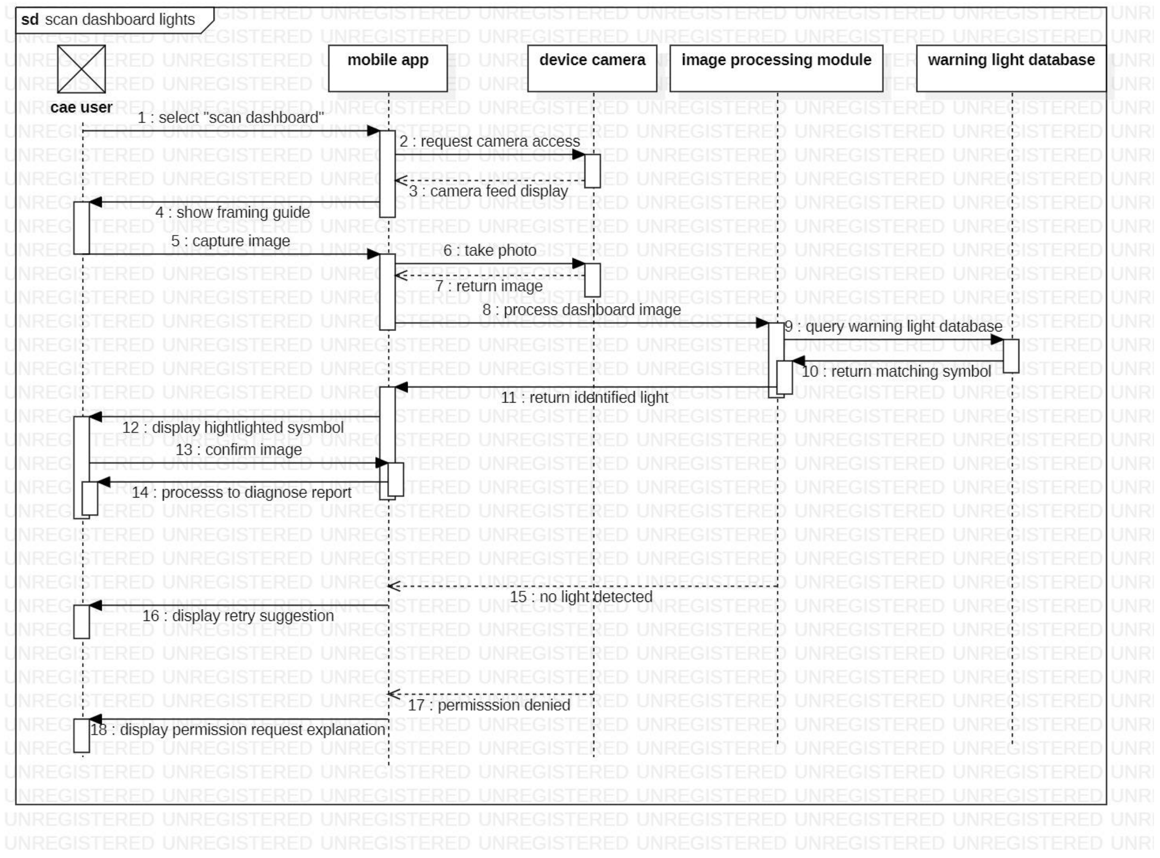

sd scan dashboard lights
cae user
mobile app
device camera
image processing module
warning light database
1 : select "scan dashboard"
2 : request camera access
3 : camera feed display
4 : show framing guide
5 : capture image
6 : take photo
7 : return image
8 : process dashboard image
9 : query warning light database
10 : return matching symbol
11 : return identified light
12 : display hightlighted sysmbol
13 : confirm image
14 : processs to diagnose report
15 : no light detected
16 : display retry suggestion
17 : permisssion denied
18 : display permission request explanation

Figure 4.2 Sequence diagrams for Scan Dashboard Light

4.3.3. Engine Sound Recording Flow

Objective: Record engine sounds and analyze them for potential issues.

Importance: The sequence diagram for audio analysis helps developers:

- Understand microphone permission requirements
- Design the audio capture and processing workflow
- Structure communication with audio analysis algorithms
- Plan how to present complex diagnostic results simply
- Handle environmental factors that affect recording quality

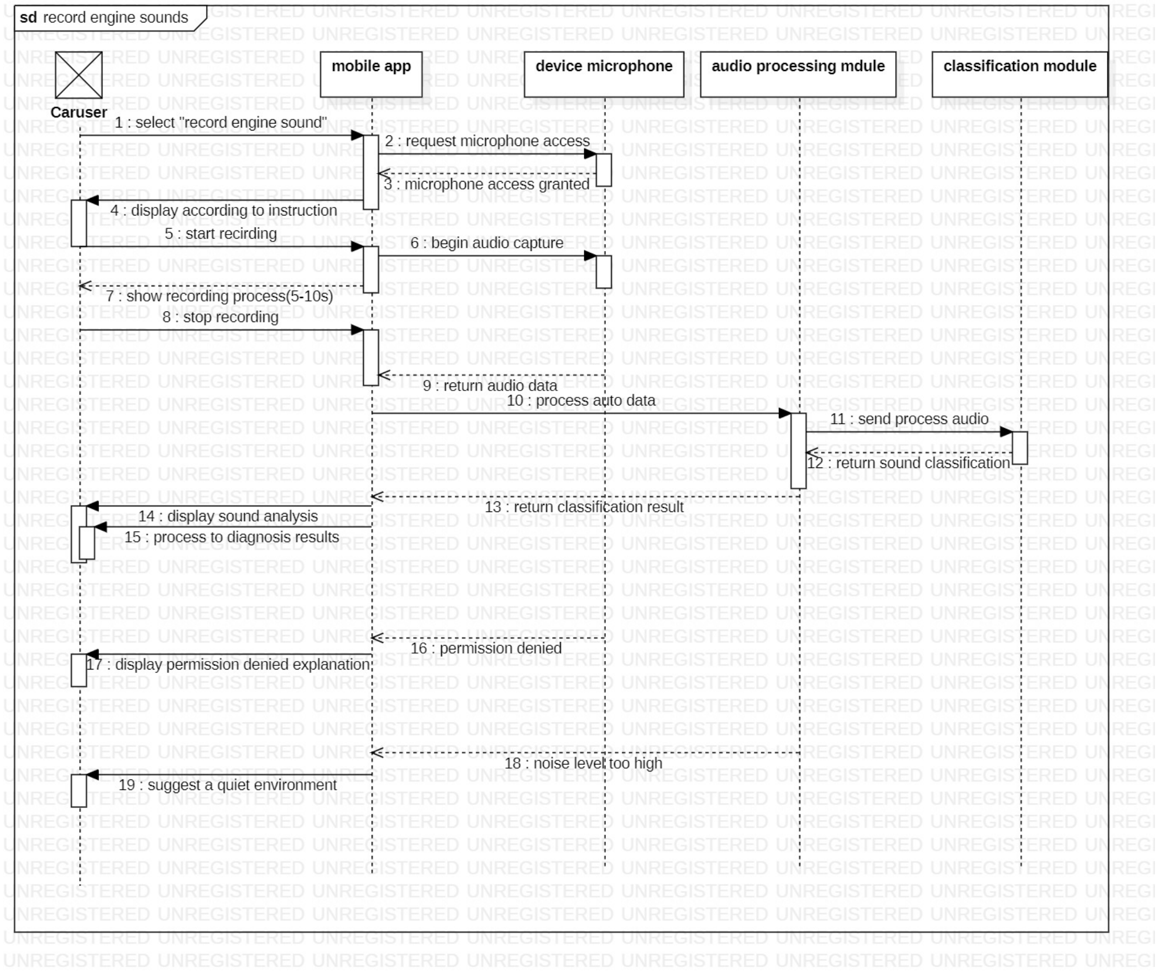
Successful Recording Process

1. User Action: Selects "Record Engine Sound".
2. System Action: Requests microphone access and displays recording instructions.
3. User Action: Starts recording (5-10 seconds).

4. System Action: Captures audio, processes the sound data, and classifies engine noise (e.g., knocking, hissing).
5. Result Display: Shows analysis (e.g., "Possible misfire detected") and recommends diagnostics or repairs.

Error Cases

- Permission Denied: Explains why microphone access is required.
- Noise Level Too High: Suggests recording in a quieter environment.



sd record engine sounds
Caruser
mobile app
device microphone
audio processing mdule
classification module
1 : select "record engine sound"
2 : request microphone access
3 : microphone access granted
4 : display according to instruction
5 : start recirding
6 : begin audio capture
7 : show recording process(5-10s)
8 : stop recording
9 : return audio data
10 : process auto data
11 : send process audio
12 : return sound classification
13 : return classification result
14 : display sound analysis
15 : process to diagnosis results
16 : permission denied
17 : display permission denied explanation
18 : noise level too high
19 : suggest a quiet environment

Figure 4.3 Sequence Diagram for Record Engine Sound

4.3.4. Diagnosis Report Viewing Flow

Objective: Display a detailed report of detected car issues.

Importance:This diagram provides crucial guidance for:

- Designing the report generation system
- Determining data sources for diagnostics
- Structuring the severity classification system
- Planning the database queries for known issues
- Implementing the history storage functionality

Report Generation Process

1. System Action: Scans diagnostic data (from scans/sensors) and queries known issues database.
2. Report Display: Shows detected problems (e.g., "Low oil pressure") and indicates urgency (e.g., "Critical – Immediate repair needed").
3. Additional Support: Provides repair tips and links to video tutorials.
4. Storage: Saves the report in history.

Possible Outcomes

- Critical Issue (e.g., engine failure): Displays urgent alert.
- No Issues Found: Shows "All clear" message.

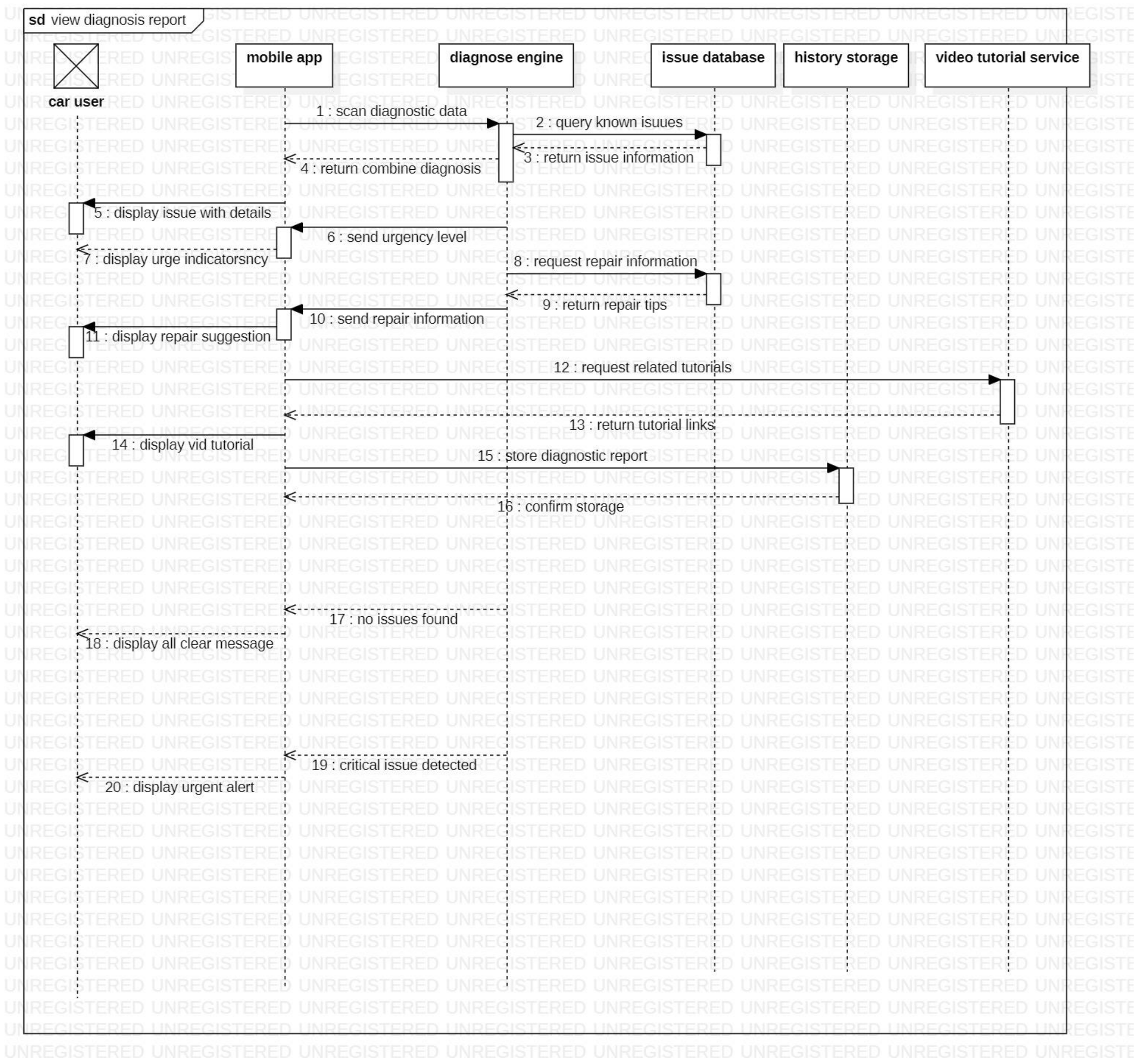

sd view diagnosis report
car user
mobile app
diagnose engine
issue database
history storage
video tutorial service
1 : scan diagnostic data
2 : query known isuues
3 : return issue information
4 : return combine diagnosis
5 : display issue with details
6 : send urgency level
7 : display urge indicatorsncy
8 : request repair information
9 : return repair tips
10 : send repair information
11 : display repair suggestion
12 : request related tutorials
13 : return tutorial links
14 : display vid tutorial
15 : store diagnostic report
16 : confirm storage
17 : no issues found
18 : display all clear message
19 : critical issue detected
20 : display urgent alert

Figure 4.4 Sequence Diagram for View diagnose Report

4.3.5. Video Tutorial Access Flow

Objective: Provide instructional videos for car repairs/maintenance.

Importance: The tutorial access sequence helps with:

- Designing the content delivery system
- Implementing offline/online mode handling
- Structuring the video recommendation system
- Planning caching mechanisms
- Handling network connectivity issues gracefully

Tutorial Retrieval Process

1. User Action: Views diagnosis report and selects a tutorial.
2. System Action: Requests related tutorial links and displays available videos.
3. Video Playback: Streams the video if online, retrieves cached version if offline.

Alternative Cases:

- No Specific Tutorial Found: Suggests general maintenance videos.
- Connection Error: Prompts to retry or use cached content.

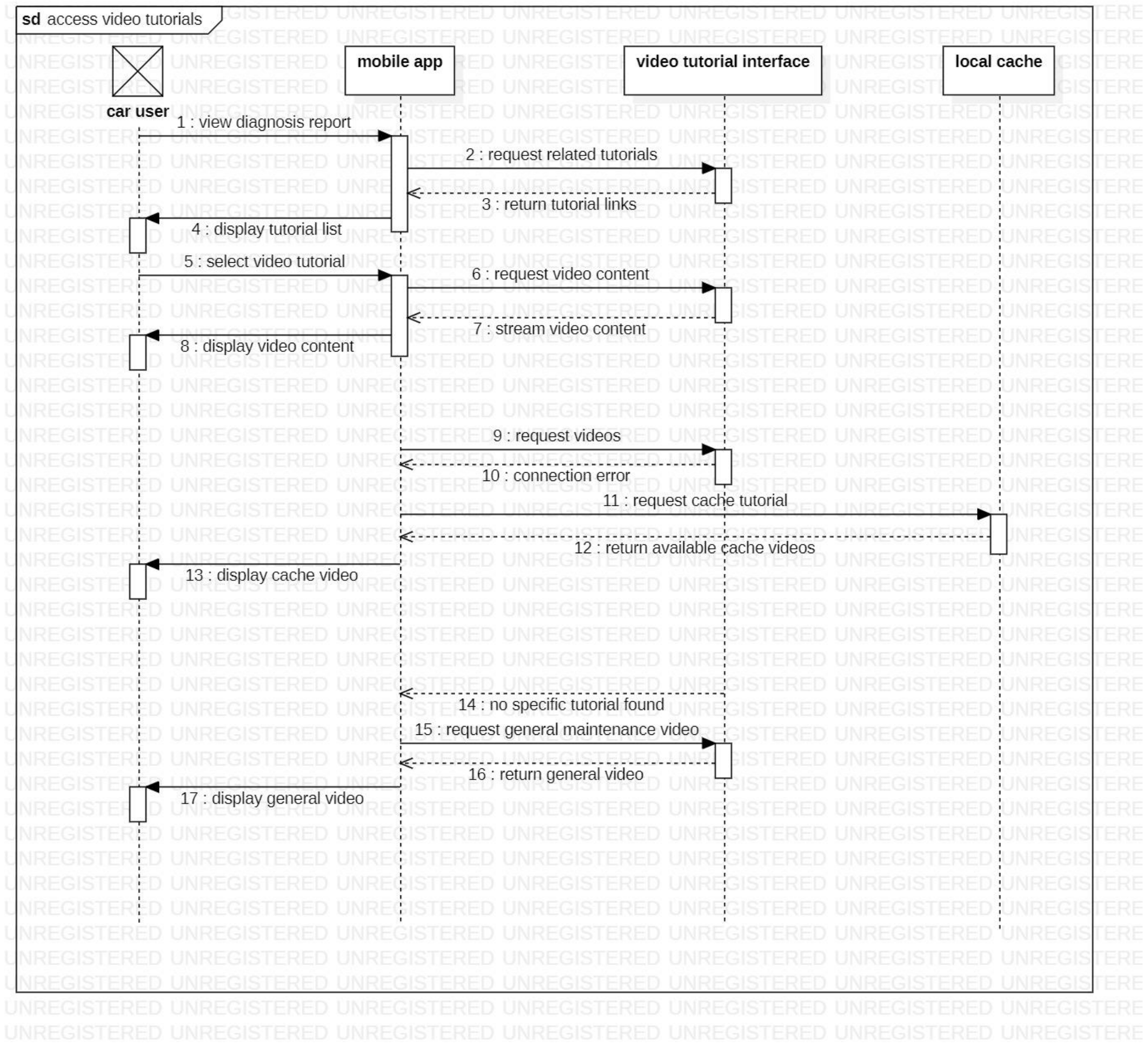

sd access video tutorials
car user
mobile app
video tutorial interface
local cache
1 : view diagnosis report
2 : request related tutorials
3 : return tutorial links
4 : display tutorial list
5 : select video tutorial
6 : request video content
7 : stream video content
8 : display video content
9 : request videos
10 : connection error
11 : request cache tutorial
12 : return available cache videos
13 : display cache video
14 : no specific tutorial found
15 : request general maintenance video
16 : return general video
17 : display general video
UNREGISTERED

Figure 5 Sequence Diagram for Access Video Tutorial

Summary of Key Features

Feature	**Description**	**Possible Errors**
Registration/Login	Secure account setup & access	Invalid credentials, email issues
Dashboard Scan	Camera-base warning light detection	Blurry image, no lights found
Engine Sound Analysis	Microphone-based diagnostics	Background noise, permission issues
Diagnosis Report	AI-powered issue detection & repair tips	False positives, no issues found
Video Tutorials	Guided repair instructions	Network issues, missing tutorials

The sequence diagrams illustrate a user-friendly, multi-functional car diagnostic app with:

- Secure authentication
- Multiple diagnostic methods (visual & audio)
- Smart error handling (permissions, retries, fallbacks)
- Helpful resources (repair tips, video tutorials)

5. CLASS DIAGRAM

5.1 Introduction to class diagram

The class diagram represents the object-oriented structure of the CarDiagApp system, showing classes, their attributes, methods, and relationships. It's designed based on the functional requirements, user classes, and system features described in the SRS.

5.2. Class Descriptions and Interactions

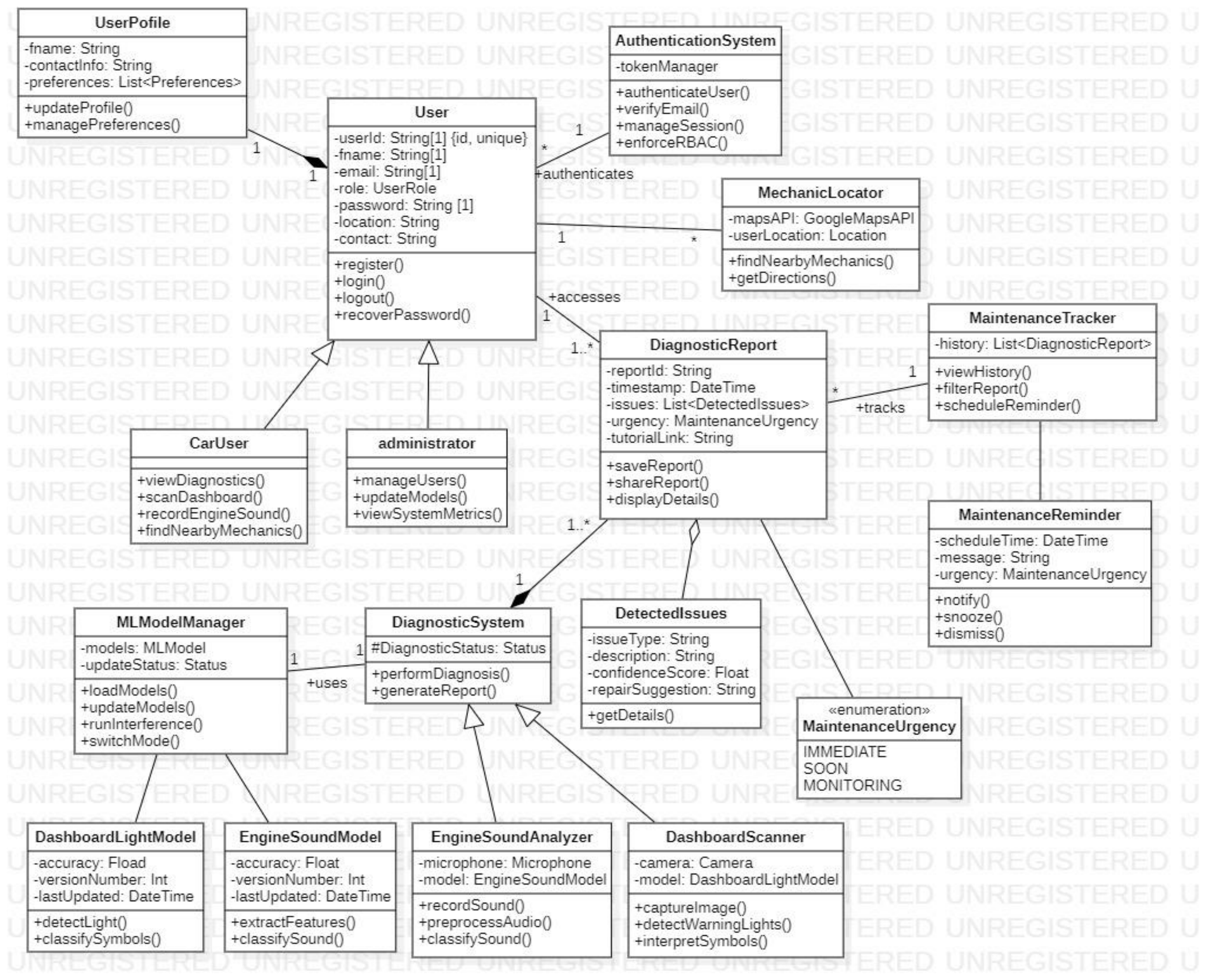


FIGURE 5: Class diagram

Below is a breakdown of each class and its role within the system, along with thei primary interactions:

1. **UserProfile**

   - **Use:** Stores personal information and preferences for a user.
   - **Attributes:** Name, contact info, preferences (e.g., list of preferences).
   - **Operations:** updateProfile(), updatePreferences().
   - **Interactions:** Associated with User (a User has a UserProfile).

2. User

- **Use:** Represents a generic user of the system, managing core user functionalities like registration, login, and role assignment.
- **Interactions:**
  - Authenticates with AuthenticationSystem.
  - Associated with UserProfile.
  - Parent class for CarUser and administrator (inheritance).

3. AuthenticationSystem

- **Use:** Handles user authentication and authorization (RBAC - Role-Based Access Control).
- Interactions: Authenticates user instances.

4. CarUser

- **Use:** Represents a specific type of user who owns or operates a car, with functionalities related to car diagnostics and maintenance.
- **Interactions:**
  - Inherits from User.
  - Interacts with DashboardScanner and EngineSoundModel for diagnostics.
  - Interacts with MechanicLocator to find mechanics.
  - Interacts with DiagnosticReport (likely views reports).

5. Administrator

- **Use:** Represents a system administrator with elevated privileges for managing users and system settings.
- **Interactions:** Inherits from User. May interact with MLModelManager to update models.

6. MechanicLocator

- **Use:** Provides functionalities to locate mechanics and get directions.
- **Interactions:** Accessed by CarUser (and potentially MaintenanceTracker if it tracks mechanic assignments).

7. **DiagnosticReport**

- **Use:** Stores and manages diagnostic reports generated by the system.
- **Interactions:**
  - Generated by DiagnosticSystem.
  - MaintenanceTracker tracks diagnostic reports.
  - DetectedIssues are part of a diagnostic report.

8. **MaintenanceTracker**

- **Use:** Tracks maintenance history and schedules reminders.
- **Interactions:** Tracks DiagnosticReport instances.

9. **MaintenanceReminder**

- **Use:** Manages and sends maintenance reminders to users.
- **Interactions:** Likely triggered by MaintenanceTracker.

10. **DiagnosticSystem**

- **Use:** Centralizes the diagnostic process, performing diagnosis and generating reports.
- **Interactions:**
  - Uses MLModelManager for diagnosis.
  - Generates DiagnosticReport instances.
  - Detects DetectedIssues.

11. **MLModelManager**

- **Use:** Manages machine learning models used for diagnostics.

- **Interactions:** Used by DiagnosticSystem for diagnosis.

12. DetectedIssues

- **Use:** Represents a specific issue identified during diagnostics.
- **Interactions:** Part of a DiagnosticReport. Associated with IMMEDIATE_URGENCY_SOON_MONITORING enumeration.

13. IMMEDIATE_URGENCY_SOON_MONITORING

- **Use:** An enumeration representing different levels of maintenance urgency.
- **Values:** IMMEDIATE, URGENCY, SOON, MONITORING.
- **Interactions:** Used by DetectedIssues and potentially DiagnosticReport and MaintenanceReminder.

14. DashboardLightModel

- **Use:** Represents a model for interpreting dashboard warning lights.
- **Interactions:** Used by DashboardScanner (implicitly, as DashboardScanner uses a "model" attribute).

15. EngineSoundModel

- **Use:** Represents a model for analyzing engine sounds.
- **Attributes:** Accuracy, Float, Version number, Last update date/time.
- **Operations:** extractFeatures(), classifySound().
- **Interactions:** Used by EngineSoundAnalyzer (implicitly, as EngineSoundAnalyzer uses a "model" attribute).

16. Microphone

- **Use:** Represents a microphone input device.
- **Interactions:** Used by EngineSoundAnalyzer to record sound.

17. EngineSoundAnalyzer

- **Use:** Analyzes engine sounds to detect issues.
- **Interactions:**
    - Uses EngineSoundModel.
    - Uses Microphone to record sound.

18. DashboardScanner

- **Use:** Scans the car dashboard, captures images, and interprets warnings.
- **Interactions:** Uses DashboardLightModel (implicitly, through its "model" attribute).

5.3 Workflow Summary

The system seems to support the following high-level workflows:

1. **User Management:** Users can register, log in, and manage their profiles. Administrators have privileges to manage users and system models.
2. **Diagnostic Process:** CarUser initiates diagnostics (e.g., scanning dashboard or recording engine sound). This data is processed by DashboardScanner (using DashboardLightModel) and EngineSoundAnalyzer (using EngineSoundModel and Microphone). The DiagnosticSystem utilizes MLModelManager to perform the diagnosis, identify DetectedIssues, and generate a DiagnosticReport.
3. **Maintenance and Tracking:** DiagnosticReports are tracked by MaintenanceTracker, which can view history and schedule MaintenanceReminder instances. CarUser can also use MechanicLocator to find nearby mechanics for repairs based on the diagnostic results.

The class diagram outlines a comprehensive system for automotive diagnostics and maintenance. It leverages machine learning models for analyzing car data (dashboard lights, engine sounds) and provides functionalities for user management, diagnostic reporting, maintenance tracking, and locating mechanics. The modular design, with clear separation of concerns among classes, suggests a well-structured system capable of handling various aspects of vehicle health monitoring. The relationships indicate a strong focus on automated diagnostics and user-centric features for managing vehicle maintenance.

6. DEPLOYMENT DIAGRAM

6.1. Introduction to deployment diagram

The deployment architecture of CarDiagApp illustrates how different components of the system interact across the mobile device, cloud backend, and external content resources. The architecture supports key functions including diagnostic data analysis using AI models, storage and retrieval of user data, online learning through tutorial links, and model training updates.

This system is structured around three major domains:

- **Client Mobile Device**: where the CarDiagApp is installed and interacts with the user
- **Cloud Services**: backend infrastructure responsible for authentication, data storage, and model training
- **YouTube Link**: an external service used to deliver learning content through tutorial videos

The architecture is designed to provide real-time performance for diagnostics while relying on cloud services for heavier operations like model training and user management.

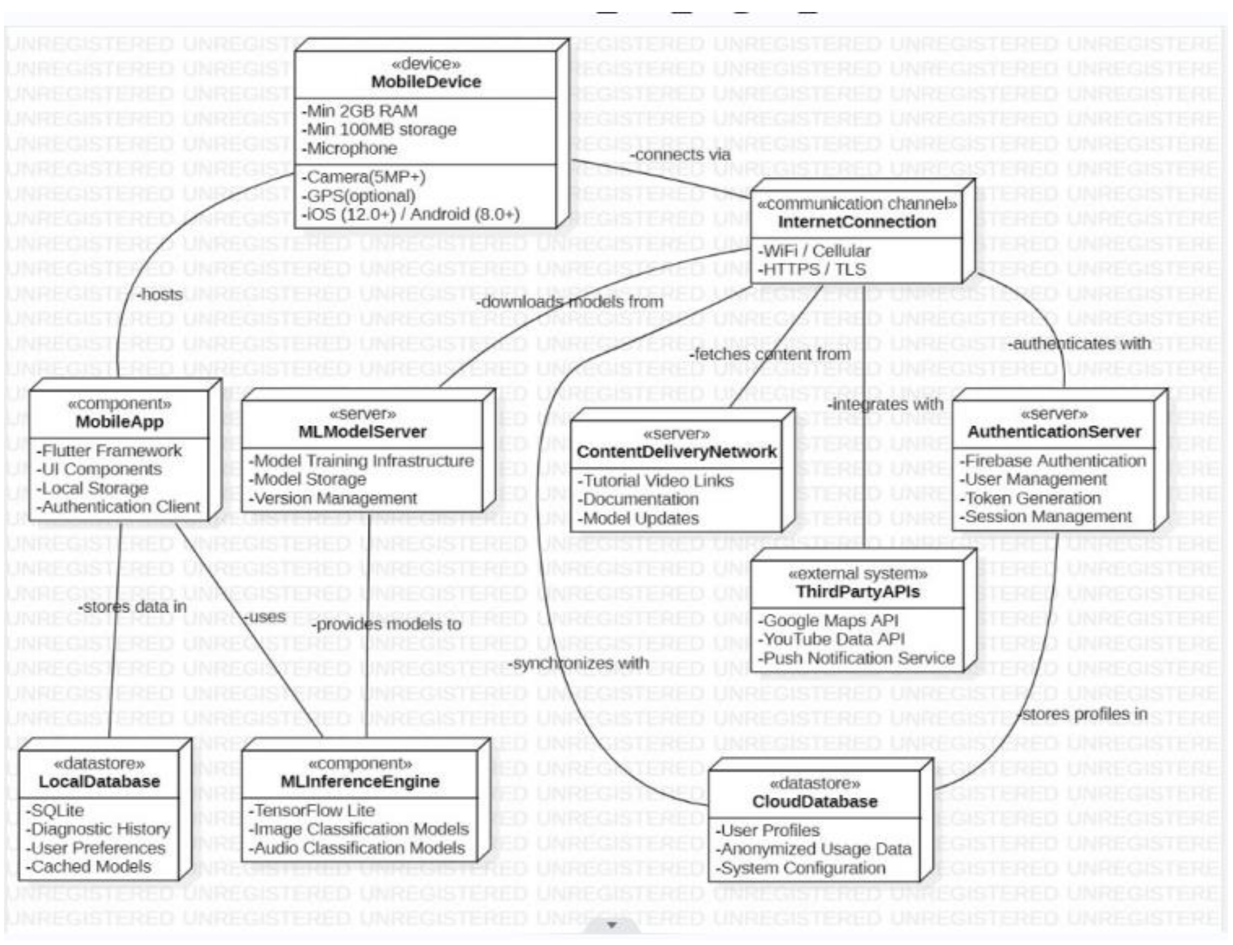


Figure 6: Deployment Diagram

6.2. **Component Explanations**

6.2.1. Client Mobile Devise

This is the end-user's smartphone or tablet running the CarDiagApp. It contains the following internal components:

- **CarDiagApp (Flutter Application):** The main user interface and control layer. Built using Flutter, this application allows users to collect vehicle data, interact with diagnostic models, and receive feedback or tutorial recommendations.
- **Local SQLite Database:** A lightweight database embedded within the app to store user history, diagnostic logs, and cached data for offline access. It improves responsiveness by reducing the need for constant cloud queries.

- **TensorFlow Lite Models:** These are pre-trained machine learning models optimized for mobile execution. They run locally on the device to provide real-time diagnostics without needing to contact the server for every analysis.

6.2.2. Cloud Services

These services provide backend infrastructure to support operations that are too resource-intensive or sensitive for local processing. The cloud architecture is divided into:

1. **Firebase Platform:**

Authentication Session: Handles secure user login and session management using Firebase

2. **Authentication.**

Model Storage: Stores diagnostic models and provides updated versions to client devices when necessary.

3. **ML Model Server:**

Training Pipeline: A system that processes user-generated or expert-tagged data to retrain and fine-tune diagnostic models.

4. **Tutorial Metadata:** Metadata generated during training that maps vehicle issues to recommended learning resources.

These cloud components interact closely with the mobile client to:

- Authenticate users
- Synchronize data
- Push updated models and recommendations

6.2.3. YouTube Link

It represents: Tutorial Video Access via Hyperlinks: Instead of directly integrating with the YouTube API for content retrieval, the application fetches YouTube video links from cloud metadata and opens them using the device’s browser or video app.

This simplification makes the architecture more lightweight and reduces the need for complex API integration, especially since viewing tutorials doesn't require dynamic content manipulation.

6.3. System Interactions

- User Authentication: Initiated from the app to Firebase, which returns session tokens.
- Local Diagnostics: TensorFlow Lite models evaluate input locally, and results may be stored in SQLite or sent to the cloud.
- Model Updates: The app periodically checks Firebase for new model versions.
- Training and Metadata: The ML Model Server processes data, generates new models, and maps problem categories to educational YouTube video links.
- Tutorial Delivery: Links to relevant YouTube tutorials are shown in the app and opened externally when needed.

The CarDiagApp deployment model achieves a balance between on-device performance and cloud-based intelligence. By executing diagnostics locally and relying on cloud services for storage and learning, it ensures a fast, responsive, and scalable system.

Conclusion

The CarDiagApp represents a significant advancement in automotive diagnostics, bridging the gap between traditional mechanical troubleshooting and intelligent digital solutions. Through well-structured diagrams and system modeling, this report illustrates how user-friendly interfaces, machine learning models, and cloud integration converge to deliver accurate, real-time fault detection and educational support. The system not only empowers users to diagnose issues independently but also contributes to safer, more cost-effective vehicle maintenance. Overall, the architecture and design presented here ensure scalability, security, and accessibility setting the foundation for future development and innovatio

Appendix

Table 1 – Acronyms Used

Acronym	**Meaning**

Acronym	**Meaning**
DFD	Data Flow Diagram
ML	Machine Learning
API	Application Programming Interface
UI	User Interface
SRS	Software Requirements Specification

Figure References

- **Figure 1**: Context Diagram
- **Figure 2**: Data Flow Diagram
- **Figure 3**: Use Case Diagram
- **Figures 4.1 – 4.5**: Sequence Diagrams
- **Figure 5**: Class Diagram
- **Figure 6**: Deployment Diagram