

UNIVERSITY OF BUEA

**FACULTY OF ENGINEERING
AND TECHNOLOGY**



REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

**COURSE TITLE: INTERNRT PROGRAMMING AND MOBILE
PROGRAMMING**

COURSE CODE: CEF440

COURSE INSTRUCTOR: Dr. VALERY NKEMENI

GROUP: 18

TASK 4: UI DESIGN AND IMPLEMENTATION

GROUP 18 MEMBERS

S/N	NAME	MATRICULE
1	SINDZE DJAM ALAN WILFRIED	FE20A105
2	ENJEMA NDIVE PEACE	FE22A203
3	NGWINKEM KETTY NERITA	FE22A269
4	NJAMUTOH SHALOM BRENDA TASA	FE22A271
5	TABI ATEM ROGAN ESSEMBION	FE22A300

02/06/2025

Table of contents

COURSE TITLE: INTERNRT PROGRAMMING AND MOBILE PROGRAMMING	1
1. App Identity	3
1.1 App Name:	3
1.2 Purpose	3
1.3 Target Audience	4
1.4 Key Features	4
1.5 Uniqueness	6
2. Visual Design	7
2.1 Design Tools Used	7
2.2 Color Scheme	7
2.3 Typography	7
2.4 Screen Designs	8
2.5 UX Principles	14
3. Frontend Implementation Document	15
3.1 Framework Used	15
3.2 Structure of Your App	15
3.3 Important Widgets	32
3.4 State Management	32
3.5 Responsiveness	32
3.5 Integration Points	33
3.6 Challenges Encountered and Solutions	33
4.0 Conclusion	34

1. App Identity

1.1 App Name:

➤ Car Fault Finder

✧ Name Justification:

The name “Car Fault Finder” was chosen to reflect the app’s role as a trusted assistant for diagnosing automobile issues. “CAFF” is short for “Car Fault Finder,” making it immediately clear that the app deals with car fault detection. The term “Fault Finder” emphasizes the app’s supportive and helpful nature, acting like a co-pilot or assistant for drivers and vehicle technicians.

1.2 Purpose

Car Fault Finder addresses a common and often frustrating problem for car users: interpreting and resolving dashboard warning lights and vehicle faults. Modern vehicles come with complex dashboards featuring dozens of indicators, many of which are not immediately understandable to average drivers.

The app helps users:

- Quickly identify unknown warning lights using their smartphone camera.
- Understand the meaning of dashboard alerts.
- Watch Repair tutorials on fixing issues with your car.
- Automatically Diagnose your car to know the fault Using fault codes.
- Record Engine unusual sound and propose a cause and solution to it.
- Provide Understanding about OBD-2 fault codes.

- Learn about the possible causes and severity of issues.
- Receive suggestions for immediate actions or repairs.
- Avoid unnecessary trips to mechanics for minor issues or know when a visit is urgent.

By making car diagnostics more accessible, Car Fault Finder empowers users with knowledge, helping them save time, reduce stress, and maintain their vehicle more effectively.

1.3 Target Audience

The app is designed for a broad spectrum of vehicle users, including:

- Private car owners who want to maintain their vehicles and reduce repair costs.
- Taxi and ride-share drivers who rely on their vehicles for income and need fast diagnostics.
- Mechanics and auto technicians looking for a quick reference or second opinion.
- Fleet managers responsible for monitoring multiple vehicles.
- Learner drivers and automotive students who want to understand vehicle warning systems.

1.4 Key Features

1. Scan and Recognize Dashboard Lights

- Uses smartphone camera and AI image recognition to detect dashboard warning lights.
- Supports a wide range of vehicle models and light designs.
- Option to manually browse or search for symbols if camera scanning is not possible.

2. Display Fault Descriptions

- Provides clear explanations of what each dashboard symbol means.
- Includes potential causes of the warning, categorized by severity: low (informational), medium (service soon), high (urgent attention required).
- Offers additional technical information for advanced users.

3. Suggest Possible Actions or Fixes

- Lists recommended immediate actions (e.g., “Check oil level”, “Stop driving immediately”).
- Provides troubleshooting steps and estimated repair urgency.
- Links to nearby service centers, spare part stores, or professional mechanic services

4. Record Unusual Engine Sound

- Use Audio classification to identify potential mechanical issues.
- Propose a cause and solution for the unusual engine sound.
- Use machine learning or deep learning models trained on sound data-set to identify abnormalities in unusual engine noise.

5. Provide Youtube Videos from verified car experts

- Show step by step repair or maintenance processes related to the detected issue.

6. Provide Basic Offline Functionality

7. Provides Online features for accessing updated fault database and video content

8. Diagnose Car problems using OBD-2 Data

9. Provide a car maintenance History

1.5 Uniqueness

What Sets Car Fault Finder Apart:

- **Image-Based Recognition:** Unlike many apps that rely solely on manual lookup, this app uses real-time image recognition, making it faster and easier for users to identify unfamiliar symbols.
- **Localized Recommendations:** Provides suggestions and support tailored to regional contexts, including common vehicle brands and local service providers.
- **User-Friendly Interface:** Designed with simplicity in mind—minimal steps required for diagnosis and a clear, intuitive layout.
- **Knowledge Base Updates:** Regularly updated database to cover new car models and evolving dashboard symbol standards.
- **Offline Mode:** Essential features (e.g., symbol lookup and fixes) available without an internet connection for emergencies.
- **Learning Mode:** Educational resources for users who want to learn about car maintenance and warning lights.

Overall, Car Fault Finder fills an essential gap in the automotive ecosystem by simplifying car fault detection and enabling users to act quickly and confidently. Whether you're a seasoned driver or a car enthusiast, this app is your personal assistant in ensuring your vehicle stays roadworthy and reliable.

2. Visual Design

2.1 Design Tools Used

The UI/UX design of the Car Fault Finder app was prototyped using **Figma**. Figma was selected for its collaborative features, ease of use, and real-time design iteration capabilities. In addition, the actual layout was refined within Flutter using its flexible widget tree system.

2.2 Color Scheme

The color palette was chosen to enhance clarity:

- **Orange (#FFA000):** Represents repair icons.
- **Green (#388E3C):** Used for normal status or successful scans.
- **Blue (#1976D2):** For interactive elements like buttons, icons and links.
- **White and Light Gray backgrounds:** Provide contrast and maintain readability.

2.3 Typography

The app uses **Roboto** font family, consistent with Android's material design guidelines.

Font sizes vary by context:

- **24–28 pt** for headers (e.g., screen titles).
- **16–18 pt** for body text and labels.
- **12–14 pt** for captions and descriptions.

This hierarchy ensures easy reading and scannability on different screen sizes.

2.4 Screen Designs

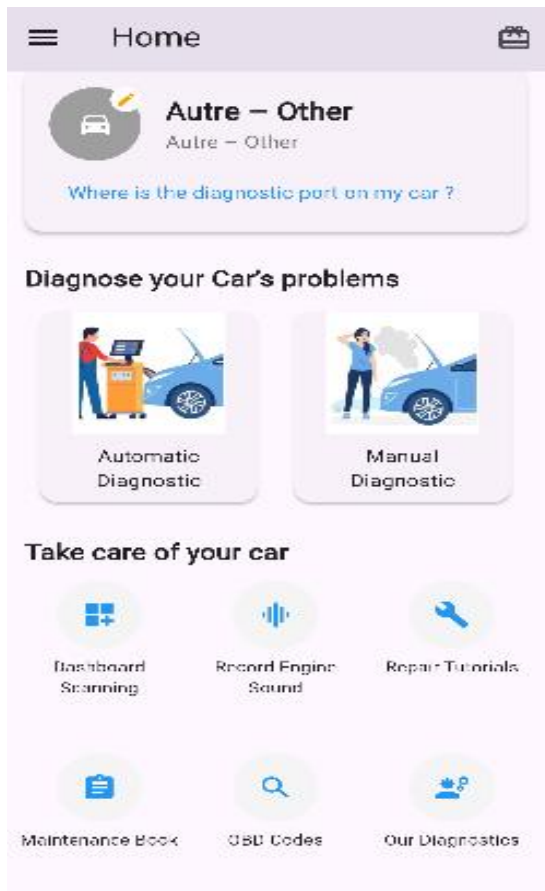
Splash Screen:

- Displays app logo with animation.
- Background color aligns with the theme (deep blue or black).



Home Screen:

- Large scan button in the center.
- Recent scans listed below in a scrollable ListView.
- Navigation bar at the bottom.



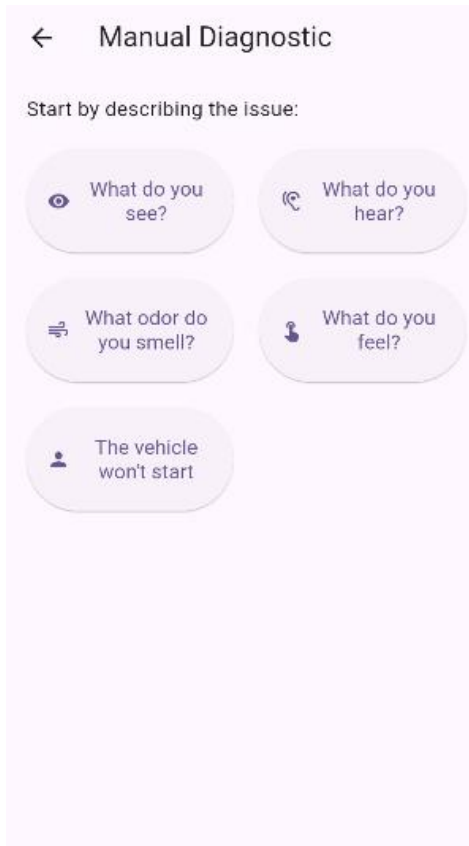
Automatic Diagnostic Screen:

- Shows image of the ELM 327.
- Displays guide on how to connect the ELM 327.
- Displays a run button



Manual Diagnostic Screen:

- Displays “what do you see, what do you hear, what odor do you smell, what do you feel, and the vehicle won’t start.



The image shows a mobile application screen titled "Manual Diagnostic". At the top, there is a back arrow icon and the title. Below the title, a prompt reads "Start by describing the issue:". There are five interactive buttons arranged in a grid. The first four buttons are in a 2x2 grid, and the fifth is centered below them. Each button contains an icon and a text label: an eye icon for "What do you see?", an ear icon for "What do you hear?", a nose icon for "What odor do you smell?", a hand icon for "What do you feel?", and a person icon for "The vehicle won't start".

← Manual Diagnostic

Start by describing the issue:

What do you see?

What do you hear?

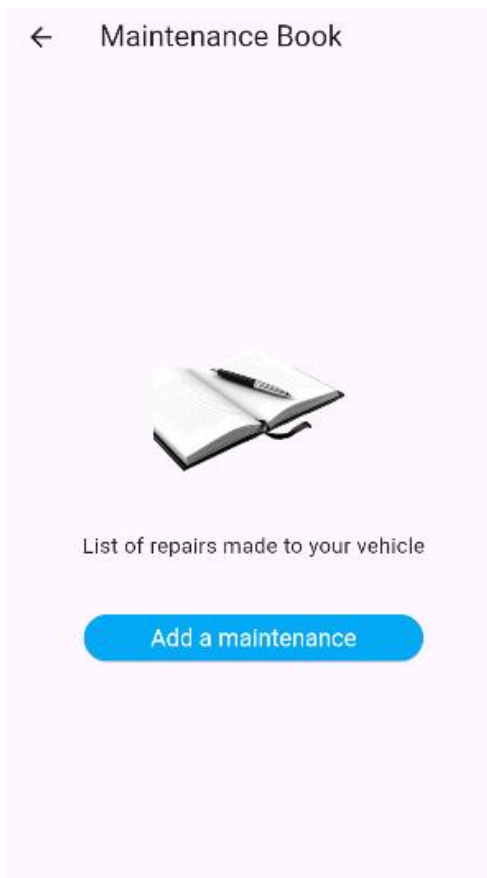
What odor do you smell?

What do you feel?

The vehicle won't start

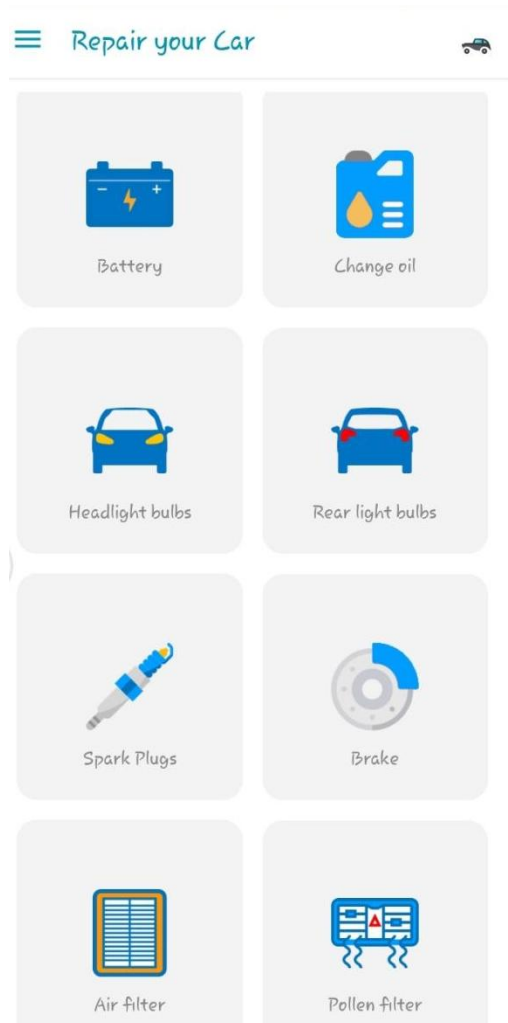
Maintenance Book Screen

- Allows the user to add a maintenance



Repair Tutorials Screen:

- Allows the user to choose what he/she wants to watch a tutorial on, be it battery, change oil, head light bulbs, rear light bulbs, spark plugs, brake, air filter or pollen filter.



2.5 UX Principles

- **Simplicity:** Clean layout with minimal clutter, focusing on core functionality.
- **Easy Navigation:** Bottom navigation bar for quick access to main sections (Home, History, Help).
- **Accessibility:** High-contrast colors, readable font sizes, and large touch targets for all elements.
- **Visual Feedback:** Loading indicators, animations on scan success/failure, and color-coded fault urgency help users understand system status.

3. Frontend Implementation Document

3.1 Framework Used

The frontend of the Car Fault Finder mobile application is built using **Flutter**, an open-source UI software development kit created by Google. Flutter was chosen for its ability to create natively compiled applications for mobile from a single codebase, its wide community support, and its rich set of pre-designed widgets.

3.2 Structure of Your App

Widget and Component Organization:

The app is structured using the principle of separation of concerns. Widgets are divided into reusable components like `'car_care_grid.dart'`, `'car_info_card.dart'`, and `'diagnostic_option_card.dart'`.

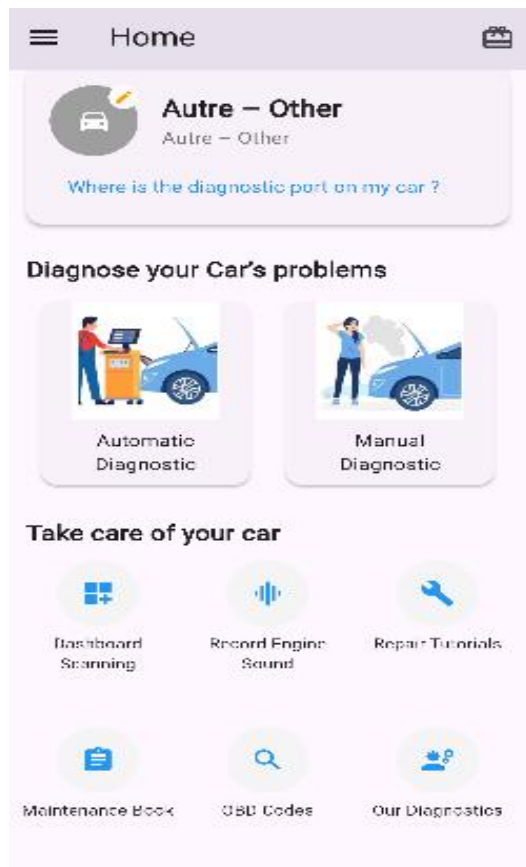
- Logic-heavy components are separated into service classes or providers.

Screens and Navigation:

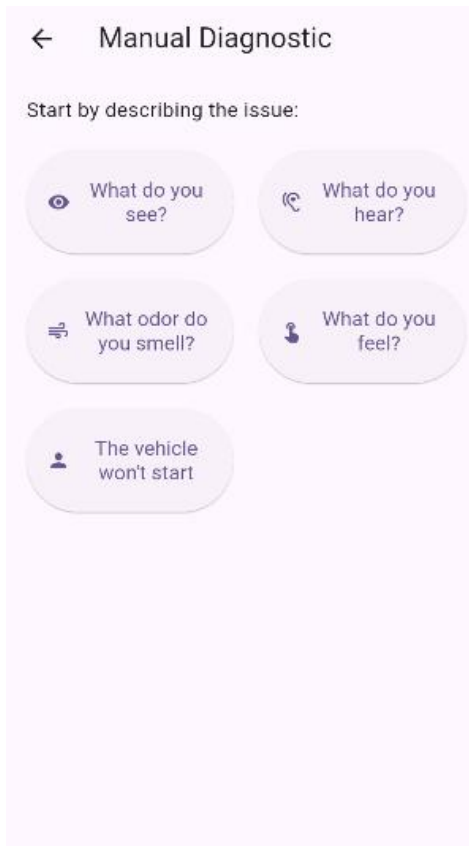
- **SplashScreen:** Initial loading and branding.



- **HomeScreen:** Main interface for scanning and accessing recent faults.



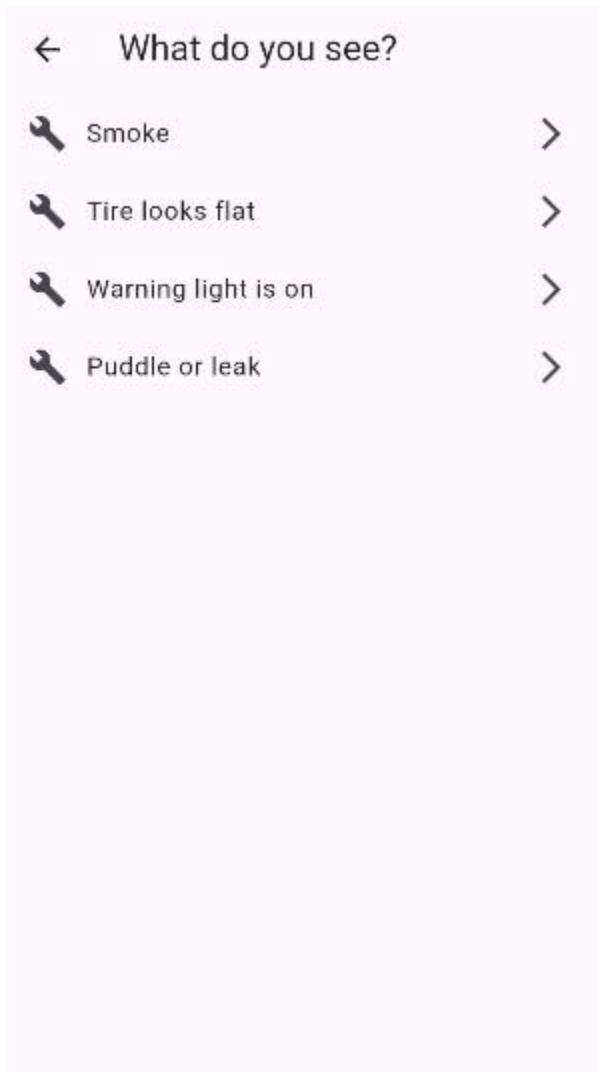
- **ManualDiagnosticScreen:** Displays options which the user can choose from.



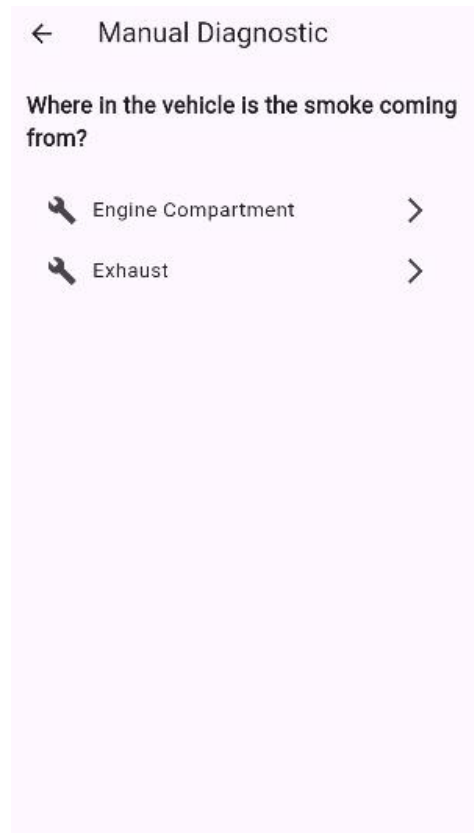
The image shows a mobile application screen titled "Manual Diagnostic". At the top left is a back arrow icon. Below the title, the instruction "Start by describing the issue:" is displayed. The screen contains five selectable options, each in a rounded rectangular button with an icon and text:

- What do you see? (Eye icon)
- What do you hear? (Ear icon)
- What odor do you smell? (Nose icon)
- What do you feel? (Hand icon)
- The vehicle won't start (Person icon)

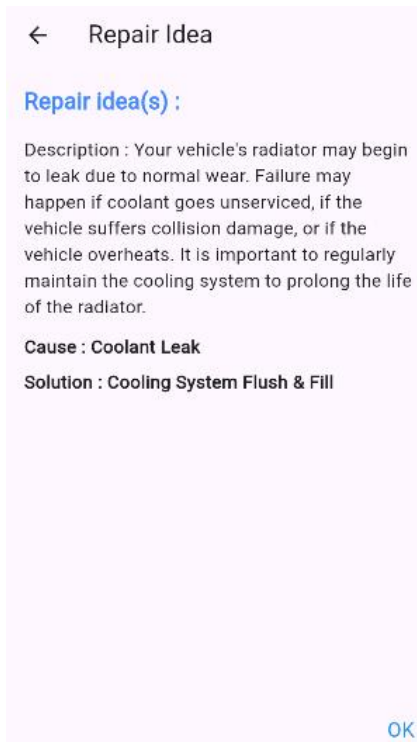
- **WhatDoYouSeeScreen:** Displays suggestions for you to choose from.



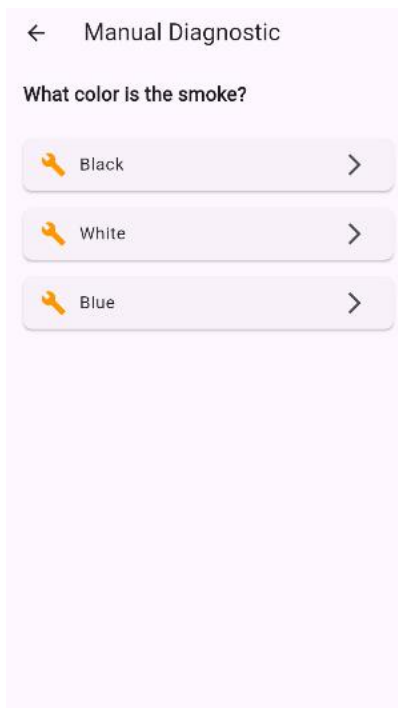
When you click on the smoke button above, it takes you to this page




When you click on the engine compartment above, it takes you to this page



And if you click on the Exhaust button, it takes you to this page



If you click on black, it takes you to this page

 **Black Smoke Repair Ideas**

Repair idea(s) :

Description : Black smoke may mean your engine is burning too much fuel or that your fuel return line is clogged. It is important to check your sensors, fuel injectors, and fuel-pressure regulator. To fix the issue and gain better fuel economy, be sure to have your vehicle repaired by an expert.

Cause : Engine Is Burning Too Much Fuel

Solution : Engine Diagnostics

OK

If you click on white, it takes you to this page

← White Smoke Repair Ideas

Repair Idea(s):

Could be because of the following reasons:

1. Normal Condensation (Cold Start)

Description:

White smoke appears briefly when you first start the car, especially on cold mornings, and disappears once the engine warms up.

Cause:

Cause:

Water vapor (condensation) evaporating in the exhaust system.

Solution:

Solution:

No action needed; this is normal and harmless.

2. Coolant Leak Into Combustion Chamber

Description:

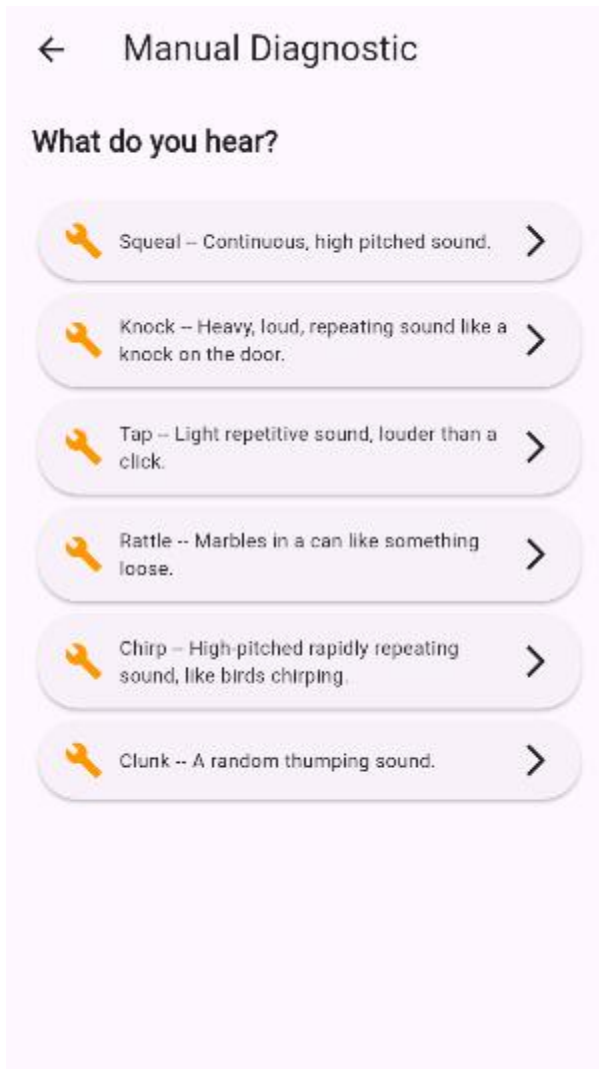
Continuous white smoke that has a sweet smell and does not go away during driving.

Cause:

Cause:

Coolant leaking into the engine's combustion chamber, usually due to a blown head gasket, cracked cylinder head, or damaged engine block.

- **WhatDoYouHearScreen:** Allows users to choose an option depending on what they hear.





From the above, when you click any button, it takes you to a page, proposing the repair idea, possible cause and solution


- WhatDoYouSmellScreen:


← Manual Diagnostic


What odor do you smell?


 Gasoline >

 Rotten eggs >

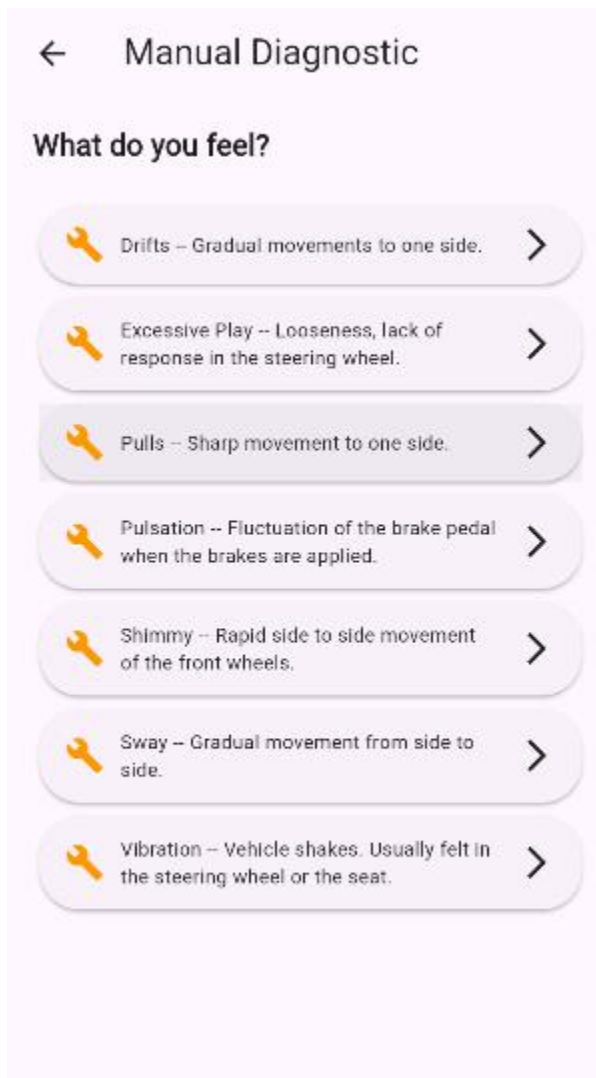
 Mildew >

 Antifreeze (sweet smell) >


 Burning oil >

 Exhaust >


- WhatDoYouFeelScreen:





- TheVehicleWontStartScreen:


 Manual Diagnostic


What happens when you try to start the vehicle?


 The engine cranks normally but does not start >


 The engine cranks over slowly >

 The vehicle is backfiring when trying to start >

 Nothing >

 One strong click or knock >

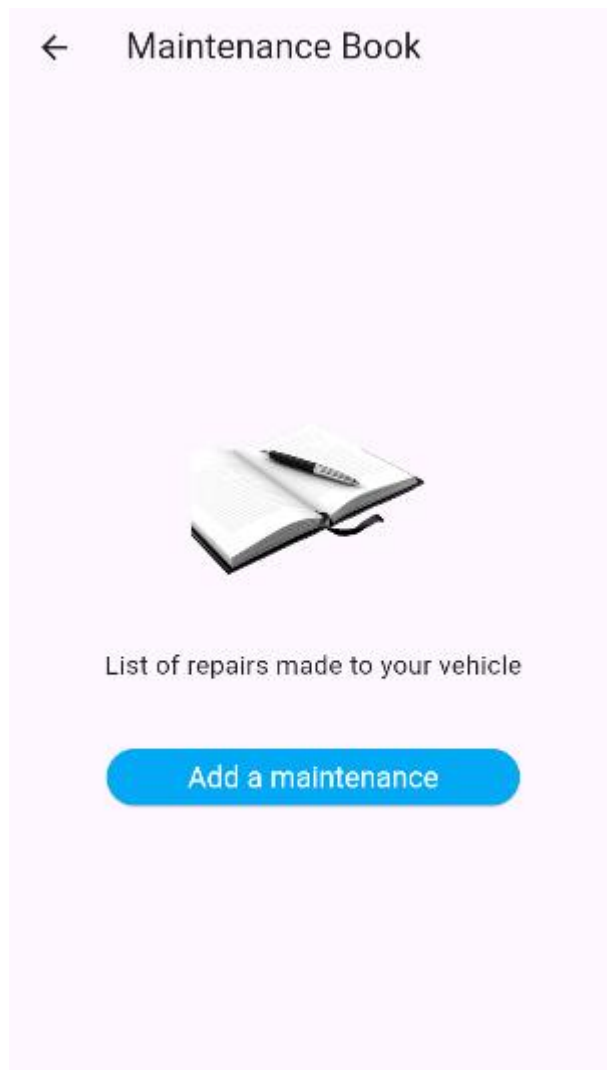
 A spinning, whirling, or gear grinding sound >

 Repeating clicking sound: "click, click, click" >

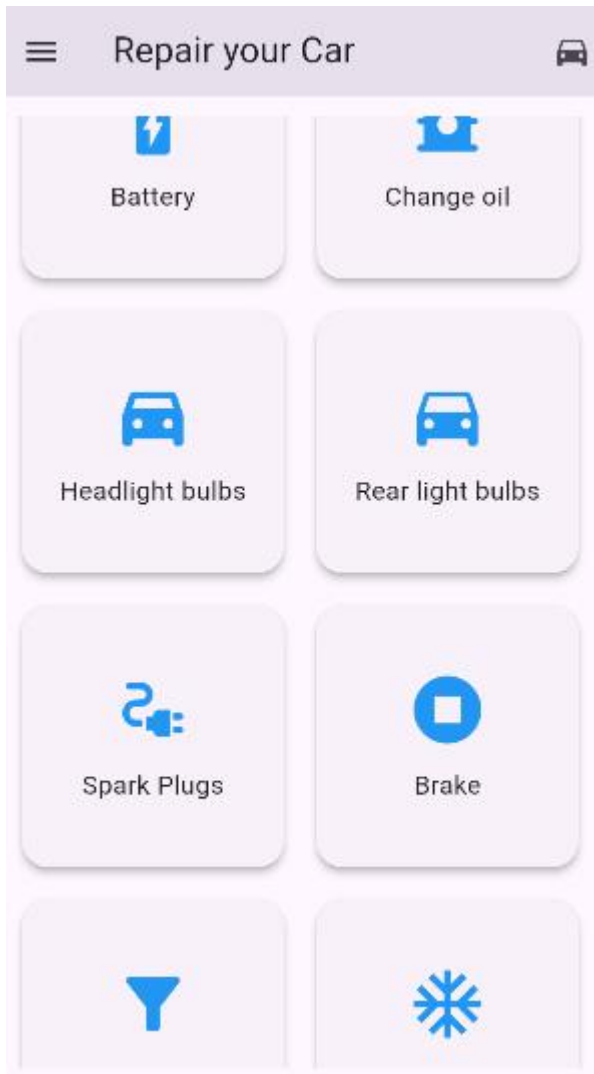
- AutomaticDiagnosticScreen:



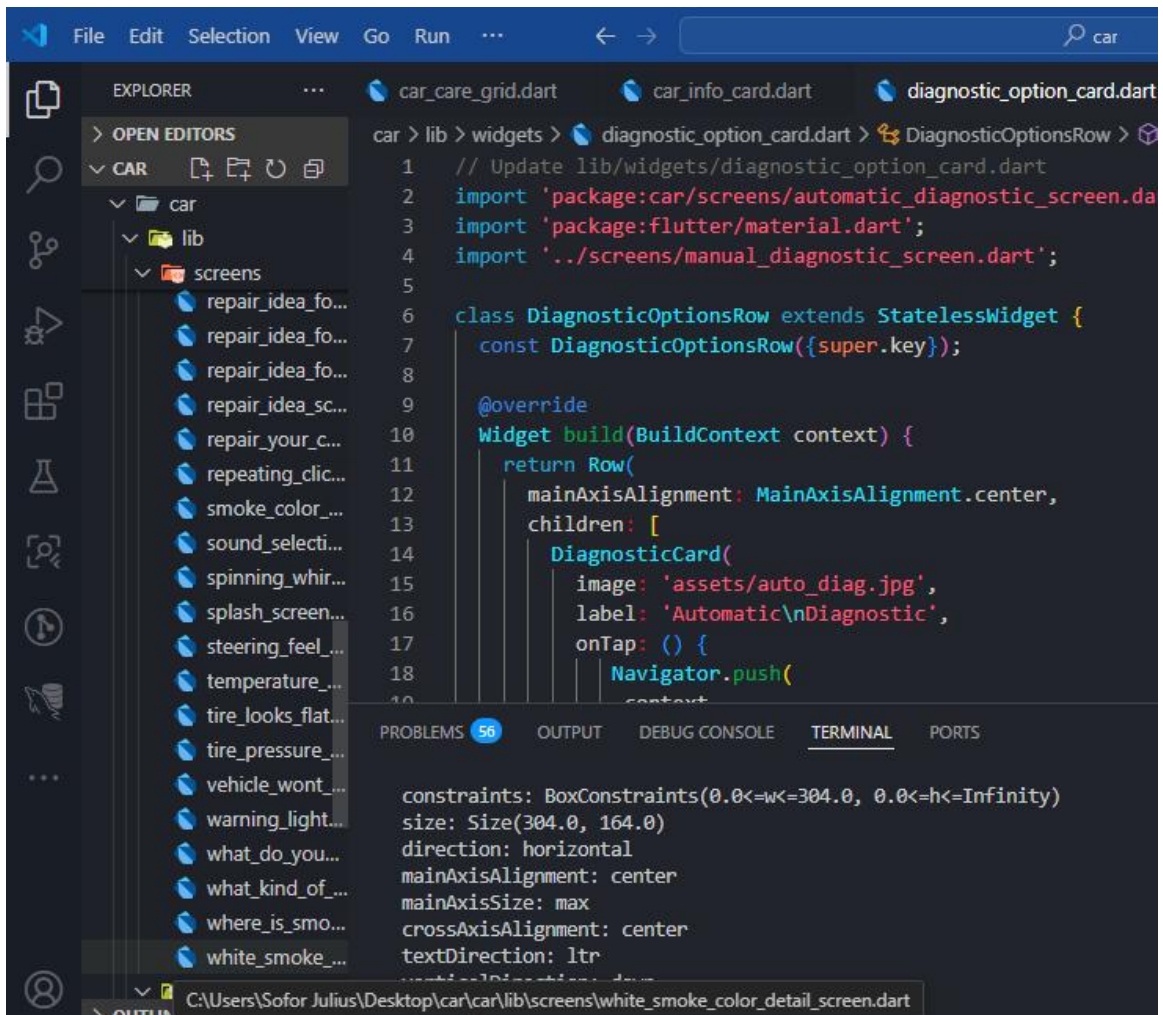
- MaintenanceBookScreen:



- RepairTutorialsScreen: Here, the application allows you to choose which specific component you want to watch a repair tutorial on.



Overall, we implemented 40 screens and 3 widgets



3.3 Important Widgets

- **Buttons:** Used extensively for actions like 'Scan Now', 'Rescan', 'Get Fixes'. Implemented using `ElevatedButton` and `IconButton`.
- **ListViews:** To display historical scan results, suggested fixes, and fault descriptions.
- **Image Recognition Result Display:** Custom widget displays the recognized dashboard symbol, its meaning, and recommended actions in a card layout using `Card`, `Image`, and `Text` widgets.

3.4 State Management

State management is handled using **Provider**, a lightweight and efficient solution for sharing state between widgets. Provider allows the app to:

- Update UI reactively when scan results are received.
- Maintain a shared history of scans across different screens.
- Handle UI loading states and user preferences efficiently.

3.5 Responsiveness

To ensure the app works across various screen sizes and orientations:

- Layouts use `MediaQuery` and `LayoutBuilder` for dynamic sizing.
- Flexible widgets like `Expanded`, `Flexible`, and `Wrap` were used to adapt to different resolutions.
- Widgets and text sizes scale based on screen dimensions using `flutter_screenutil` package.

3.5 Integration Points

The frontend interacts with a local machine learning (ML) model for image recognition:

- Camera input is passed to a TensorFlow Lite model bundled in the app.
- Results from the ML model are parsed and displayed through the UI.
- Optionally, the frontend can send scan results to a backend server for storage or advanced diagnostics (future feature).

3.6 Challenges Encountered and Solutions

- **Challenge:** Integrating the ML model in a lightweight and performant way.

Solution: Used TensorFlow Lite optimized for mobile, and deferred heavy processing using Isolates.

- **Challenge:** Managing complex state across multiple UI screens.

Solution: Adopted the Provider package to simplify state distribution and UI updates.

- **Challenge:** Making the app responsive across devices.

Solution: Extensively tested layouts using emulators and applied dynamic scaling strategies.

4.0 Conclusion

The car fault diagnosis mobile application successfully delivers on its identity, visual appeal, and technical implementation. With a clear purpose of helping users—ranging from car owners to mechanics—understand and resolve car faults, the app integrates unique features such as automatic OBD2-based diagnostics, manual symptom-based input, camera-based dashboard light recognition, and audio fault detection. These features, combined with access to repair tutorials and OBD2 code interpretation, distinguish the app from traditional diagnostic tools.

Visually, the app adopts a clean and intuitive interface, guided by clear UX principles like simplicity, accessibility, and effective visual feedback. The thoughtful use of color (e.g. blue for normal operation), readable typography, and structured screen layouts ensures a smooth user experience across all app sections—from the splash screen to detailed scan results.

On the frontend, the app leverages Flutter to build a responsive and dynamic interface. Key widgets like buttons, ListViews, and multimedia components (camera, audio, image display) are efficiently organized across multiple screens. Navigation is seamless, and state management ensures that user interactions, results, and feedback are handled smoothly. Despite technical challenges like integrating Bluetooth and multimedia inputs, the app achieves a stable and scalable frontend architecture, laying the foundation for future improvements.

Overall, the strong alignment between the app's identity, design, and implementation contributes to a cohesive and user-centered mobile diagnostic solution.