

# **APP TITLE**

## **Auto-Care Connect**

### **Vehicle Services & Roadside Assistance**

**Mobile Application Development**

**2023-2024SEM10XCIS099-2**

**Assessment 2 – Group Mobile Application Development  
Report**

**Group Name: Group L**

**Student Name: Pravin Timalisina - University ID: 2133104**

**Student Name: SUBASH BASKOTA - University ID: 2148322**

**Student Name: RUPESH KUMAR CHAUDHARY- University ID: 2154880**



## Table of Contents

|  |           |
|--|-----------|
| <b>LIST OF GROUP MEMBERS.....</b>  | <b>3</b>  |
| <b>Introduction/Overview.....</b>  | <b>3</b>  |
| <b>Task Description.....</b>   | <b>4</b>  |
| <b>Project Plan/Schedule.....</b>  | <b>4</b>  |
| <b>Project Overview.....</b>   | <b>5</b>  |
| <b>Aims and Objectives.....</b>  | <b>6</b>  |
| <b>Project Scope.....</b>  | <b>6</b>  |
| <b>Selected Technologies.....</b>  | <b>7</b>  |
| <b>Resources.....</b>  | <b>8</b>  |
| <b>Methodology.....</b>  | <b>8</b>  |
| <b>Stakeholders.....</b>   | <b>10</b> |
| <b>Requirement Analysis.....</b>   | <b>10</b> |
| Market Research/Competitor Analysis.....                                 | 10        |
| SWOT Analysis.....   | 11        |
| Target User Group.....   | 13        |
| Primary Market Research.....   | 14        |
| Personas and Use Case.....   | 16        |
| <b>Overview of Functional, Technical and Usability Requirements.....</b> | <b>19</b> |
| Functional Requirements.....   | 19        |
| Non-Functional Requirements.....   | 20        |
| Usability Requirements.....  | 21        |
| <b>App Design.....</b>   | <b>22</b> |
| Use Case Diagram.....  | 22        |
| Activity Diagram.....  | 25        |
| Prototype.....   | 26        |
| <b>Database Design.....</b>  | <b>32</b> |
| <b>Mobile App Development/Implementation.....</b>                        | <b>34</b> |
| .....  | 38        |

|  |           |
|--|-----------|
| <b>Testing.....</b>  | <b>39</b> |
| Test Plan.....   | 39        |
| Test Log.....  | 40        |
| .....  | 43        |
| .....  | 44        |
| .....  | 45        |
| .....  | 45        |
| .....  | 46        |
| .....  | 48        |
| <b>Mobile App Evaluation.....</b>                                      | <b>49</b> |
| Plan.....  | 49        |
| Ethical issues.....  | 51        |
| Evaluation Results.....  | 52        |
| .....  | 56        |
| <b>Mobile App Store Optimization (ASO) and Marketing Strategy.....</b> | <b>60</b> |
| Marketing Strategy for Auto-Care Connect.....                          | 60        |
| Monetization Strategy.....   | 61        |
| App Store Optimization (ASO) Strategy for Auto-Care Connect.....       | 62        |
| <b>Group Member Contribution List.....</b>                             | <b>63</b> |
| <b>Discussion/Critical Analysis/Reflection.....</b>                    | <b>64</b> |
| <b>Conclusion.....</b>   | <b>70</b> |
| <b>References.....</b>   | <b>71</b> |
| <b>Appendix.....</b>   | <b>72</b> |

## LIST OF GROUP MEMBERS

### Group-L

| FULL NAME              | UNIVERSITY ID |
|------------------------|---------------|
| PRAVIN TIMALSINA       | 2133104       |
| SUBASH BASKOTA         | 2148322       |
| RUPESH KUMAR CHAUDHARY | 2154880       |

## Introduction/Overview

World is at the pinnacle of technology, as our lives are eased and depended on the convenience that technology has brought. In this fast-paced world, not only people seek and rely on personal mobility but also require easily accessible solutions in the automotive sector at their fingertips, where mobile applications stand as the cornerstone. To cater this growing demand, we would like to introduce “Auto-Care Connect”, a revolutionary application, with the aim to transform the landscape of Automotive sector in Nepal.

“Auto-Care Connect” is a one-of-a-kind mobile application developed primarily to facilitate the gap between consumers and garage businesses. The application aims to simplify the lives of consumers by equipping them with tools like location-based garage locator, appointment booking, on-demand mechanic services, along with emergency roadside assistance all the while empowering the garage businesses by providing them essential tools like checking scheduled appointments, see their monthly statistics, analyzing user reviews and ratings, staff management.

This report serves as a comprehensive documentation of the development and evaluation of the ‘Auto-Care Connect’. The report is carefully structured comprising of several sections, all of them collectively providing detailed overview of the app’s features, development process, testing results and overall performances.

Furthermore, the report also delves into the realm marketing strategy for the 'Auto-Care Connect'. The focus is primarily given to the creation of strategic plans and promotional tactics to reach and broaden app's target audience.

## Task Description

The assignment required students to develop a mobile application (or a game) using hybrid technologies like PhoneGap, Ionic, Flutter, or React Native. The completion of the project required practical implementation as well as the formulation of a business strategy.

The project could either be a standalone piece of work or an extension of the Assessment 1. Emphasis was on designing, programming, and evaluating the chosen project, alongside the detailed documentation for the final report.

Mandatory features for the mobile apps were well-designed user-interface, user input capability, result display along with multimedia integration. Further recommendation strongly emphasized on additional features such as interaction with other apps, GPS, Google API's as well as Network (i.e., downloading data capabilities).

## Project Plan/Schedule

| Week No. | Tasks   | Priority |
|----------|---|----------|
| 1        | Research Topic for Mobile App Proposal.<br>Conduct Online Research on Mobile Applications.<br>Identify stakeholders and target audience.<br>Discuss initial specifications with stakeholders. | Must     |
| 2        | Create the Mobile App Proposal.<br>Create Interactive Prototypes  | Must     |
| 3        | Requirement Analysis for the Mobile App<br>-User Interviews<br>-User Surveys<br>Analysis of the Results   | Must     |
| 4        | List Mobile App Requirements<br>-Functional Requirements  | Must     |

|   |  |      |
|---|--|------|
|   | -Technical Requirements<br>-Usability Requirements   |      |
| 5 | Source graphics required for the Mobile App.<br>Develop Mobile-App front-end using flutter.<br>Setup Firebase for Backend  | Must |
| 6 | Design and implement NoSQL database for the App.   | Must |
| 7 | Develop backend scripts using Firebase functions.<br>Implement user Login using Firebase Auth.<br>Integration of Google API (For Maps) from the GCP.<br>Conduct Testing.<br>-System Testing<br>-Validation Testing | Must |
| 8 | Evaluation of the Mobile App.<br>Recommendation for future endeavors of the Mobile App.<br>Write Mobile App Report.<br>Mobile Application Presentation.  | Must |

## Project Overview

The automotive industry in Nepal has seen a staggering growth in the recent years, with a whopping 3.1 million registered motor vehicles as of mid-2023. This figure represents a diverse range of vehicles, where two-wheelers and four-wheelers take the lead, highlighting the nation's increasing dependency on personal mobility.

As of the latest census data, conducted in 2021, the vehicle-to-population ratio stands at 1:7, demonstrating the pivotal role of automobiles in the daily lives of Nepali citizen. But as with every machinery, the automobiles also need proper maintenance and timely servicing for them to operate optimally and provide their services to the consumers. This is where garage and servicing stations comes into play, where people go to have their vehicles tuned and serviced, also the place they turn into as soon as their vehicles breakdown.

There is also the fact that, the modern world is fast-paced, and people are time-bounded, this reason has caused people to elect for services, primarily from mobile application due to their ease of access, saving them time and tackling hurdles.

This heavy reliance on the automobiles has brought up a pressing need: a user-friendly solution that satiates the demand of consumer and garage businesses in Nepal. “Auto-Care Connect” is the answer to this need. A comprehensive user-focused mobile application designed and developed completely based on the requirements gathered from the primary as well as the secondary market research.

The primary goal of this application is and always will be, to simplify the lives of people, with its user-centered features, for all their vehicle repairment services, all the while empowering the garage owners to manage and expand their services.

## Aims and Objectives

The major objectives this project aims to achieve are:

- Provide Location-Based Garage Locator: “Auto-care Connect” aims to empower users to find their nearby garage with the aid of an embedded map.
- Seamless Appointment Booking: The application intends to make the appointment scheduling process convenient, saving time and hassle for consumers.
- On-Demand Mechanic Services: The application strives to ensure assistance to the consumer by providing feature to find nearby mechanic mechanics as the need arises.
- Convenient Vehicle Pickup and Drop-Off: “Auto-Care Connect” aims to make consumer’s life convenient by offering doorstep vehicle pick and drop service.
- Emergency Roadside Assistance: “Auto-Care Connect” aspire to be a reliable platform that users count on for emergency roadside assistance (like towing services and mechanic).

## Project Scope

This project is an extension of the Assessment 1, proposed by our group member (Pravin Timalisina). The reason behind the selection of this application comes from various compelling findings based on the primary and secondary market research (more on this later) of the proposed application.

It was clear that the mobile app market for automotive in Nepal is in its early stages and rapidly growing, coupled with the fact that the existing applications in market offered services like vehicle search, price comparison and car reviews but none of them were developed to cater the needs of consumer and garage businesses.

With the help of SWOT analysis (detailed later), 'Auto-Care Connect', with its user-oriented design, would fill the void between the consumer and the garage services, transforming the landscape of the auto-motive sector of Nepal.

However, to make the application further user-friendly, few changes were made during the development phase of the application, notably the garage owner-faced services were modified resulting in minor changes in consumer-faced services to accommodate and integrate all the changes. Despite the changes, the application developed perfectly aligned with the main motto 'cater the needs of consumer and garage businesses' and successfully met the objectives of the application.

## Selected Technologies

After a careful analysis of key trends in the software industry and rising demands of hybrid applications, flutter proved to be the most promising choice for the development of the application. This choice perfectly aligned with the goal of developing a hybrid application, ensuring wider accessibility catering both android and iOS users.

For the backend services of the application, firebase was selected. With its seamless integration and ease of use comprising various features like real-time database, authentication, scalable storages, serverless functions and many more, firebase proved to be a quintessential part for the successful development of the application.



## Resources

This section represents the resources used for the developing, running and testing of the application.

### Software

- Visual studio code for code editing.
- Flutter framework for app development.
- Firebase for backend development and cloud services.
- Google cloud platform for google maps.

### Hardware

- Mac machine with 16 GB of Ram.
- Windows machine with 16 Gb of Ram.
- Virtual Mobiles: Pixel 3a, iOS 16 simulator.
- Mobiles: Huawei Nova 5T, various models of Samsung.

### Graphics

- Figma for prototyping

## Methodology

Agile methodology, influenced by the Agile Manifesto (Agile Alliance,2001), was employed for this mobile application project. Complemented by prototyping methodology principles, as outlined by the TryQA (2018), this approach allowed us to emphasize on iterative and collaborative development.

The project was organized into four parts, each spanning two weeks, which allowed us flexibility and continuous collaboration with the take stakeholders. Similarly, Figma was employed for rapid prototyping within the initial two weeks.

### Sprint Structure

| Sprint | Tasks  | Priority |
|--------|--|----------|
| 1      | Research Topic for mobile app proposal<br>-Conduct Online research on mobile applications<br>-Identify stakeholders and target audience<br>-Discuss initial specifications with stakeholders<br>-Write Mobile App Proposal | High     |
|        | Create interactive prototypes for the mobile app   | High     |
|        | Requirements Analysis for the mobile app<br>-User Interviews.<br>-User Surveys.<br>-Analyze results.   | High     |
| 2      | List Mobile App Requirements<br>-Functional Requirements<br>-Operational Requirements<br>-Usability Requirements   | High     |
|        | Source graphics required for the mobile app  | High     |
|        | Develop mobile app front-end using Flutter   | High     |
| 3      | Setup Firebase for backend functionalities   | High     |
|        | Design and implement a NoSQL database using Firebase.  | High     |
|        | Initiate the development of backend scripts using Firebase functions.  | High     |
|        | Implement login functionality using Firebase Authentication.   | High     |
| 4      | Conduct functionality testing and usability testing for the mobile app.  | High     |
|        | Evaluate the mobile app outcome and gather feedback.   | High     |
|        | Make recommendations for future enhancements.  | High     |
|        | Compile the Mobile App Report  |          |
|        | Prepare for the Mobile App Presentation.   |          |

## Stakeholders

### **Pravin Timalisina**

Role: Project Manager and Back-end Developer

Responsibilities: Overseeing the complete project management, coordinating tasks and responsible for decision-making.

### **RUPESH KUMAR CHAUDHARY**

Role: Lead Developer

Responsibilities: Leading the development team for technical guidance and making high-level decision's related to overall development.

### **SUBASH BASKOTA**

Role: Front-End Developer and Tester.

Responsibilities: Developing the mobile app's user interface as well as conducting complete test for the overall application.

## Requirement Analysis

### Market Research/Competitor Analysis

The secondary market research was conducted to gain valuable insights of the market and the consumer's behavior while mitigating expenses (vital for an independent student).

### **Objectives**

The major objectives of this secondary market research were:

- To understand the existing mobile application of automotive sector in Nepal.
- To assess the SWOT (Strength, Weaknesses, Opportunities and Threats) of the existing application to the proposed application.
- To get acquainted with the trends and challenges in the automotive mobile app market.

### **Relevant Sources**

Given below are the sources used to collect information and data for this research:

- Existing mobile applications serving the automotive sector of Nepal like Carmudi Nepal, Nepal Wheels.
- Online rating and reviews along with user-feedback provided to the existing app, from the App Store and Google Play.
- Surveys, studies, industry reports and blogs of Nepal's automobile sector like "Introduction of Automobile Industry of Nepal" published in Scribd.

### **Findings**

The key findings from the sources are listed below:

- The mobile app market for automotive in Nepal is in its early stages and rapidly growing.
- The existing application in market offers services like vehicle search, price comparison and car reviews but do not cater the needs of consumer and garage businesses.
- Rise in demand for digital automotive services and connected car services can be easily seen.

### SWOT Analysis

Based on the information collected from the secondary market research, the following SWOT analysis has been developed for the “Auto-Care Connect” application:

### **Strengths**

- Broad set of features to satisfy the needs of consumers and garage business in Nepal.
- Priority on ease of use and convenience.
- Harnessing the emerging trends such as connected car services.

### **Weaknesses**

- New to the market and narrow brand awareness.
- Capitalized brand can prove to be a potential competition.

### **Opportunities**

- The demand for digital automotive is all time high.
- Untapped market for a mobile app dedicated to connecting consumer and garage.

### **Threats**

- Unstable economic condition of Nepal.
- Changes in regulations of automotive industry by government.
- Disruptive technological advancement in the mobile app market.

Based on the secondary market research, it is safe to say that there is a growing demand for digital automobile services in Nepal. And the findings point to the fact that the existing mobile apps do not cater the needs of consumer and garage businesses. The proposed “Auto-Care Connect” application has the potential to fill this void by offering carefully selected set of features.

## Target User Group

The information collected from the secondary market research clearly indicates “Commuter and Vehicle Owners in Urban and Rural Nepal” as the primary target user group. This group’s needs and preferences serves as the basics for the design and functionality of the proposed application.

The compelling reasons behind why this user group is the focus for our app are explained below:

- **Strong Relevance:** Vehicle owners are directly vested in the automotive services. They use vehicles for day-to-day activities like work, colleges, personal transportation.
- **Wide-Ranging Needs:** Commuters have a diverse range of needs like maintenance, servicing, and occasional roadside assistance. The proposed application exhibits wide range of feature that directly serves to such needs of the consumers.
- **Market Potential:** Vehicle ownership patterns greatly varies in urban and rural areas of Nepal, with urban regions having the highest concentration. But as the rural market area is untapped, this gives a gateway to opportunities and by targeting both areas the application can maximize its reach.
- **Smartphone Users:** The smartphone penetration of Nepal is all time high and is ever increasing which shows the interest of population towards digital services.
- **Economic Importance:** This target group plays a major role on Nepal’s economics and their capabilities to maintain their vehicles provided with ease of access via the application make them imperative target base.

With this user group as primary target audience and following the user-centric design the app’s development can align perfectly with the requirement of their automotive demands.

## Primary Market Research

With the aim to uncover nuanced insights, by actively engaging consumers and garage owners, primary market research was conducted with the help of meticulously curated questionnaires and surveys.

### **Interviews Questions.**

To understand the challenges and expectations of individuals related to vehicle maintenance, carefully structured questions were used to conduct interviews.

1. Participants were asked to share the common problems they face in vehicle maintenance and repair.
2. Participants were asked about their past experiences with mobile apps that catered the automotive maintenance.
3. Participants were asked about the frequency of their visit to garage.
4. Participants were asked about their expectation from the applications that caters the service and maintenance of their vehicle.
5. Participants were asked about their current method of finding garage and using garage services.

### **Interview Outcomes.**

1. Participants expressed the common challenges they face such as difficulties in finding reliable garages, time-consuming appointment booking, and limited emergency services.
2. Most of the participants had grievance regarding the lack of features and efficiency from their experience with applications.
3. Virtually every participant admitted to the fact of multiple visits to garage for servicing several times a year.

4. The interview highlighted the pressing demands of the participants that they expected from a mobile app which included features like garage locator, appointment scheduling, on-demand mechanic services, and emergency assistance.
5. Most of the participants revealed that the current method for selecting garages was word-of-mouth recommendations.

### **Surveys/Questionnaires**

To obtain feedback from a larger group of target audience, online surveys and questionnaires were conducted via Google Forms.

#### **Survey Questions**

1. The participants were asked to rate their satisfaction level of finding nearby garages on a scale of 1 to 5.
2. To learn the features and services that participants regarded most valuable in an automotive mobile application.
3. The participants were asked the frequency of emergencies or breakdown of their vehicles within a timeframe.
4. The participants were asked about their willingness to pay for on-demand mechanic services and garage location services via mobile application.
5. An open-ended question was also kept in the survey as to get individual feedback and suggestion.

### **Survey/Questionnaires Outcomes**

1. Survey outcomes suggested that a noteworthy portion of participants were facing problems with the current methods for finding nearby garages.
2. Features like garage locator, on-demand mechanic, and emergency assistance in a mobile app highly piqued interest of participants.



3. Survey revealed a significant portion of participants willingness to pay for an on-demand mechanic service.
4. Survey outcomes suggested that the vehicle frequency varied within participants, yet a noticeable proportion admitted such issues.
5. Open-ended question revealed the strong desire for a vehicle maintenance and servicing solutions.

The outcomes from the primary research acted as the core frame basis for the implementation of design and functionalities of the mobile application.

## Personas and Use Case

### **Persona 1: Commuter Dipika**

Background:

Dipika's persona is derived from data collected during interviews and surveys. The qualitative data shed light on her daily commute and dependency on a motorbike. Survey results also provided the quantitative data which confirmed the need for efficient automotive services.

Persona Profile:

- Name: Dipika
- Age:32
- Location: Kathmandu, Nepal
- Occupation: Working Professional
- Vehicle: Motorbike
- Goal: Hassle-free vehicle maintenance

### **Persona 2: Garage Owner Raj**

Background:

Raj's persona is developed based on qualitative data collected from interviews with garage owners. The primary outcomes from these interviews were willingness to grow their customer reach.

Persona Profile:

- Name: Raj
- Age: 40
- Location: Kathmandu, Nepal
- Occupation: Garage Owner
- Business: Garage Services
- Goal: Expand Business and Improve efficiency.

**Scenario 1:** Regular Maintenance of motorbike (For Dipika)

Use Case:

1. Dipika opens the “Auto-Care Connect” app to get an appointment for her motorbike regular servicing.
2. She selects the highly curated garages based on her location preference.
3. Dipika chooses a garage and books an appointment.
4. On the appointment day, she drops off bike at the garage.
5. Dipika gets informed regarding service completion, payment options and reminders for next feasible servicing period.

**Scenario 2:** Dipika’s motorbike breaks down on her way to office.

Use Case:

1. Dipika opens the “Auto-Care Connect” app and selects the “On-Demand/Call Mechanic” feature.
2. Based on Dipika’s location, she selects the nearest verified mechanic based on the current Location.

3. The mechanic arrives and takes the bike to the garage and starts working on it.
4. Dipika gets notified after her bike gets fixed and choses drop off service and proceeds for payment.

### **Scenario 3:** Expanding Customer Base for Raj

#### Use Case:

1. Raj opens the “Auto-Care Connect” app and proceeds to request to register his garage and submit the needed information for approval.
2. After the approval, Raj creates a detailed profile for his garage which includes services offered, pricing, and working hours.
3. Raj uploads high quality images of his garage and services creating a compelling appeal to the potential customers.
4. The app then boosts Raj’s garage by showcasing it to the users looking for servicing and maintenance in his area.
5. This action causes Raj’s garage to gain visibility which ultimately leads to increased service requests from the customers.

### **Scenario 4:** Efficient Service Delivery for Raj

#### Use Case:

1. The application provides service request from the customers along with their needs and their location to Raj.
2. After reviewing the requests Raj assigns mechanic based on their availability and expertise.
3. Assigned mechanic then uses the application to navigate to the location of the customer.
4. Raj tracks the service status and mechanics whereabouts, ensuring timely delivery.

5. Upon completion, customer makes payment and Raj requests reviews and ratings.

These personas and use cases, highlights the role of “Auto-Care Connect” app in aiding the interaction between vehicles owners like Dipika and service providers like Raj.

## Overview of Functional, Technical and Usability Requirements.

### Functional Requirements

| Req No. | Requirements   | Priority |
|---------|--|----------|
| 1       | User must be able to register on the app.                                    | Must     |
| 2       | User must be able to log in to the app.                                      | Must     |
| 3       | User must be able to log out from the app.                                   | Must     |
| 4       | Garage owner must be able to apply for registering to the app.               | Must     |
| 5       | Garage owner must be able to log in to the app.                              | Must     |
| 6       | Garage owner must be able to log out from the app.                           | Must     |
| 7       | User must be able to surf and select the listed garages.                     | Must     |
| 8       | Mechanics must be able to apply for registering to the app.                  | Must     |
| 9       | User must be able search mechanics and garages based on their location.      | Must     |
| 10      | User must be able to access maps for accessing garages over the google maps. | Must     |
| 11      | User must be able to make appointments from the app.                         | Must     |
| 12      | User should be able to provide reviews regarding services from the app.      | Should   |

|    |   |        |
|----|---|--------|
| 13 | User should be able to upload multimedia (images) from the app. | Should |
| 14 | Garage owner should be able to see their scheduled bookings.    | Should |

### Non-Functional Requirements

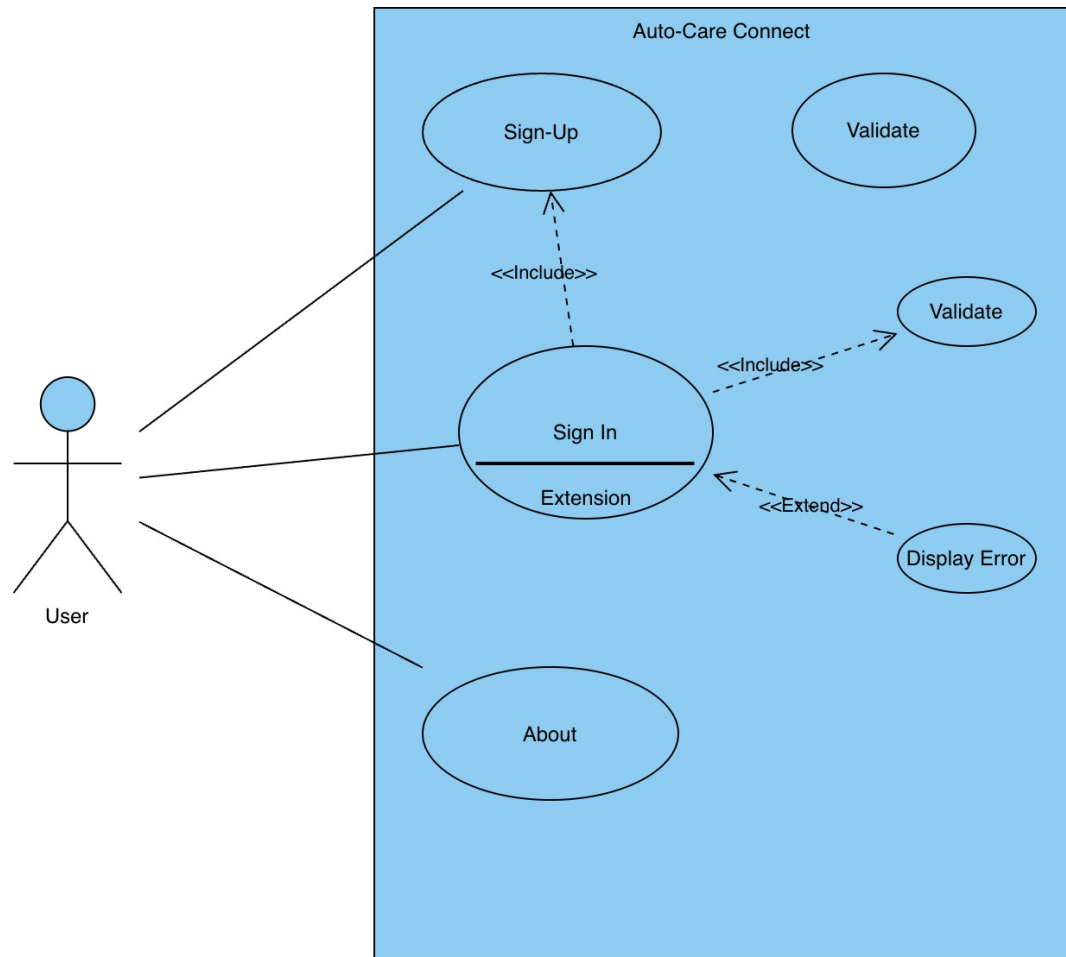
| Req No. | Requirement  | Priority |
|---------|--|----------|
| 1       | The application should process and return results within 10 seconds  | Must     |
| 2       | The app should run on a range of devices.  | Must     |
| 3       | The design should be sufficiently scalable and flexible for future enhancements.                                   | Should   |
| 4       | User should not experience critical system failures; 99.99% uptime should be achieved.                             | Must     |
| 5       | The app should provide a responsive and user-friendly interface across various screen size.                        | Must     |
| 6       | The application should comply with mobile platform guidelines and standards.                                       | Should   |
| 7       | The app should be optimized for performance and minimal battery consumption.                                       | Should   |
| 8       | The app should support offline mode, allowing users to access certain features without active internet connection. | Could    |
| 9       | The app should have clear and concise error message for better user understanding.                                 | Should   |
| 10      | Regular updates and bug fixes should be provided to ensure app stability and security                              | Must     |

### Usability Requirements

| Req No. | Requirements   | Priority |
|---------|--|----------|
| 1       | The mobile app should feature a user-centric design.   | Must     |
| 2       | The design should demonstrate a keen understanding of mobile interface design principles, ensuring consistency in form layouts, content arrangements, color schemes, and navigational methods across various screens resolutions and dimensions. | Must     |
| 3       | All data entry forms in the mobile app should be concise, easy to complete, and include entry validation.  | Must     |
| 4       | Content addition and updates should be a straightforward and swift process in the mobile app.  | Must     |
| 5       | The mobile app should have a clear and intuitive navigation.   | Must     |
| 6       | The app's design should comply with mobile accessibility standards.  | Should   |
| 7       | Text within the app should be easy to read, and language styles should be appropriate with no grammar or spelling errors.  | Should   |
| 8       | Text should be resizable for better readability on various mobile devices.   | Could    |
| 9       | The mobile app should maintain a clear and consistent layout throughout.   | Must     |

## App Design

## Use Case Diagram



**Figure No.1: Sea-Level Use Case**

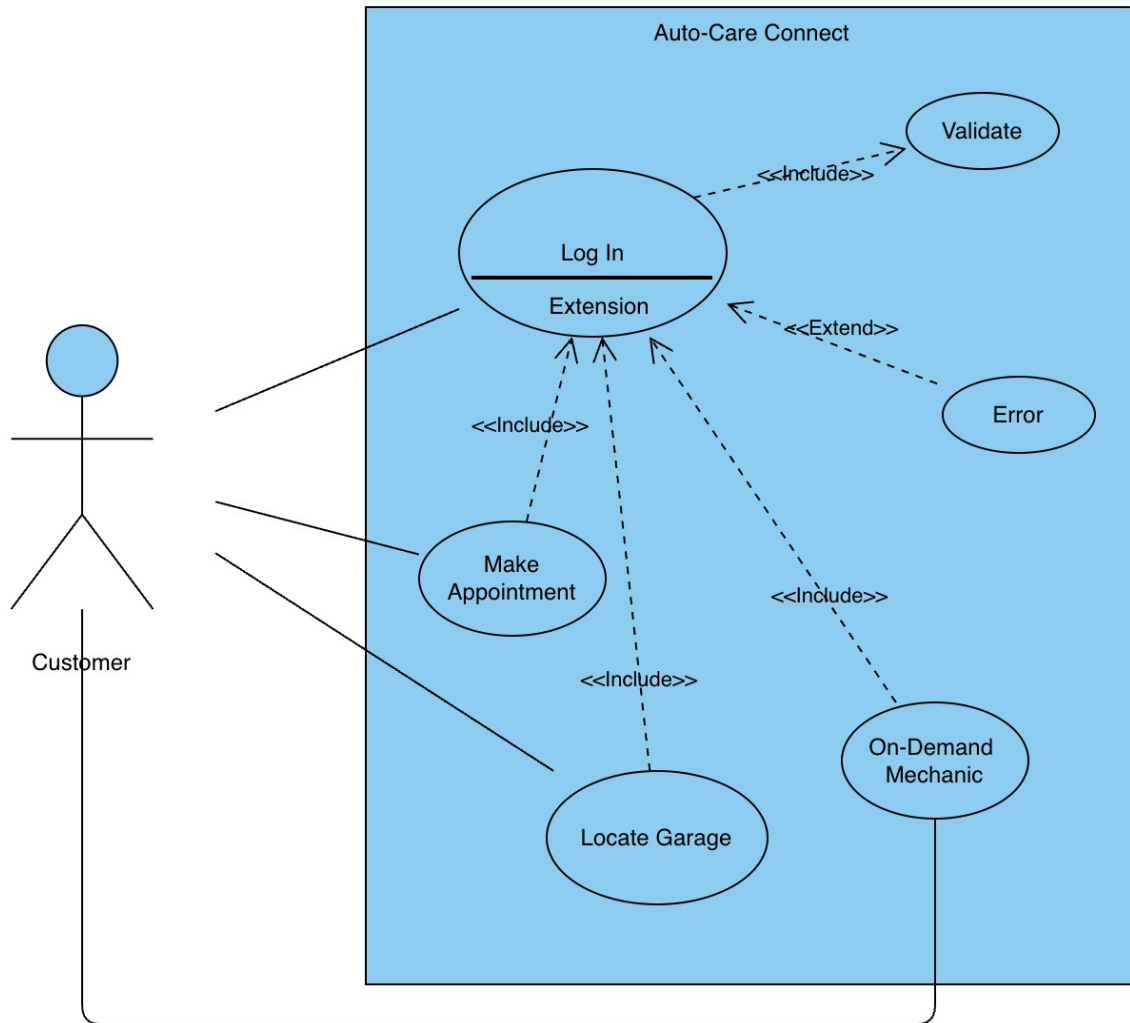
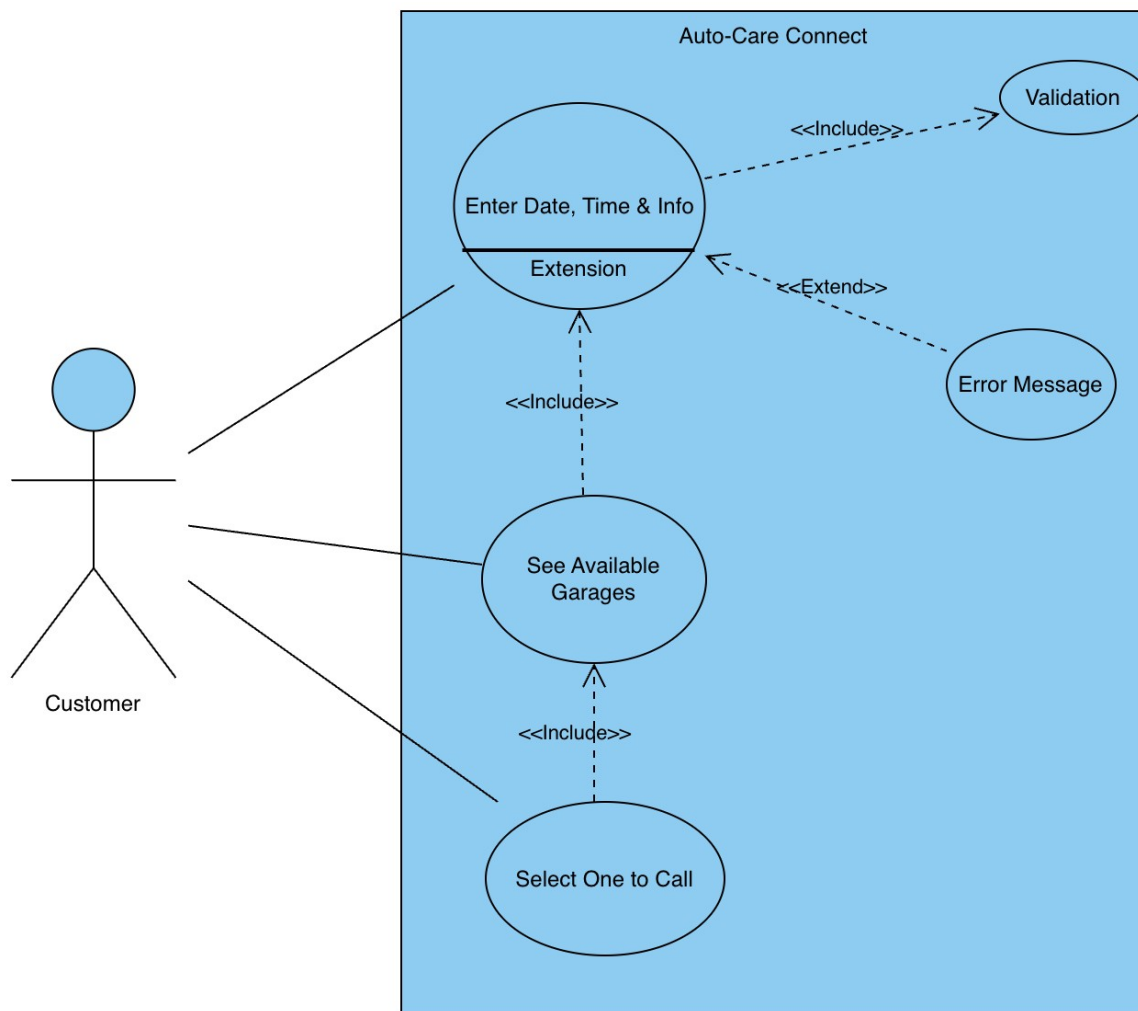


Figure No.2: Fish-Level Use Case





**Figure No.3: Clam-Level Use Case**

In the diagrams above, we can see Use Case Diagram for the “Auto-Care Connect” app, depicting user interactions at three distinct levels: Sea-Level (Fig.1), Fish-Level (Fig.2), and Clam-Level (Fig.3) are illustrated. Each level showcases users engaging with the app’s functionalities to schedule appointments seamlessly.

## Activity Diagram

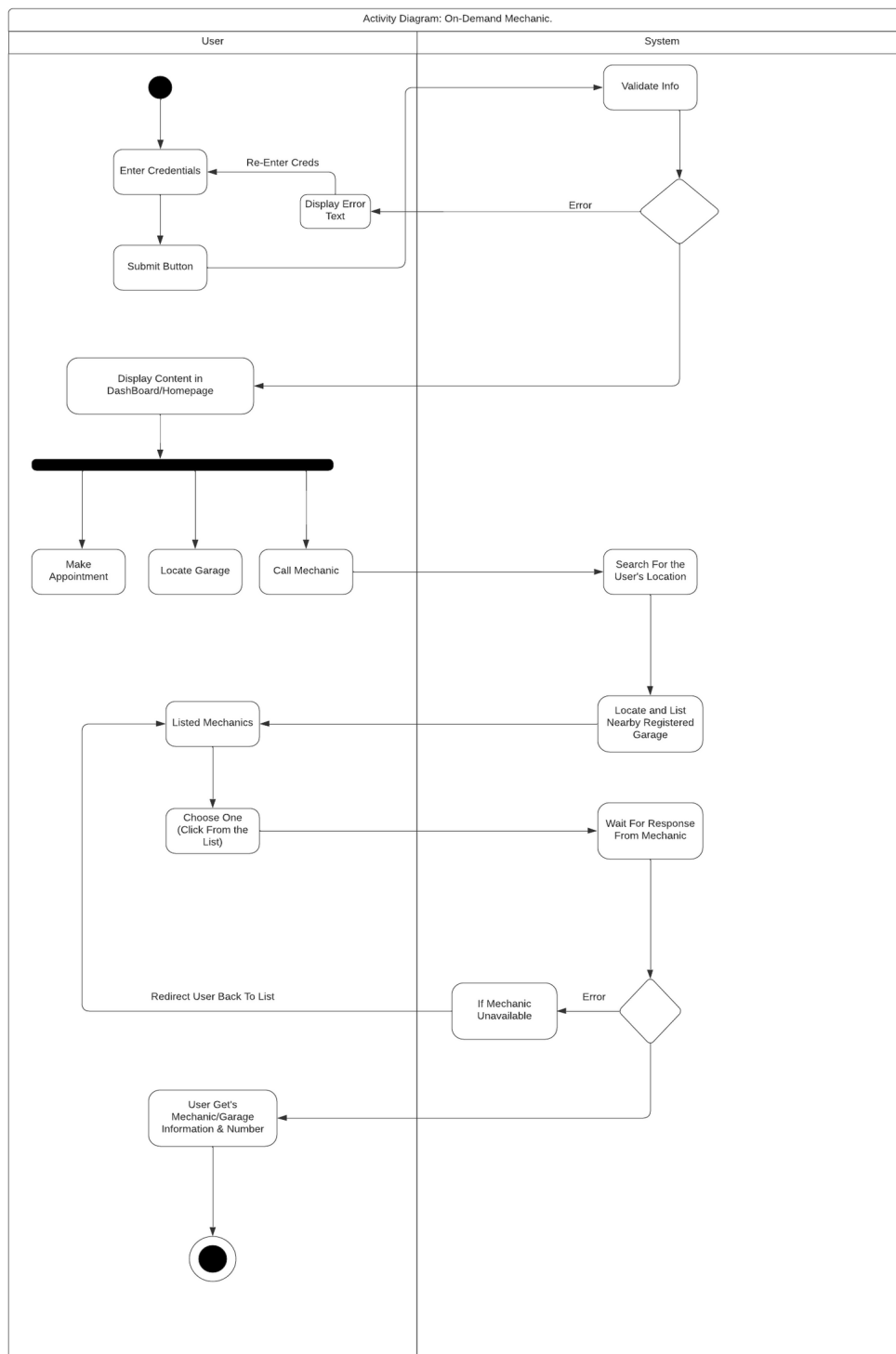


Figure No.4: Using On-Demand Mechanic

In the figure no. 4 above, the “On-Demand Mechanic” process of the “Auto-Care Connect” app is illustrated. This process includes several key steps, such as user login, validation of credentials, redirection to the dashboard and the user’s selection for the On-Demand Mechanic options. Multiple system activities are also clearly illustrated throughout the diagram like searching user’s location to provide nearby garages, and verification of user selected mechanics/garages are also pictured in the diagram.

This diagram provides a visual overview of the workflow, which helps in understanding how users interact with the app during the “On-Demand Mechanic” process.



### Prototype

Based on the identified needs and preferences discovered from the primary and secondary market research, a prototype for the “Auto-Care Connect” was developed with the help of Figma.

## Login

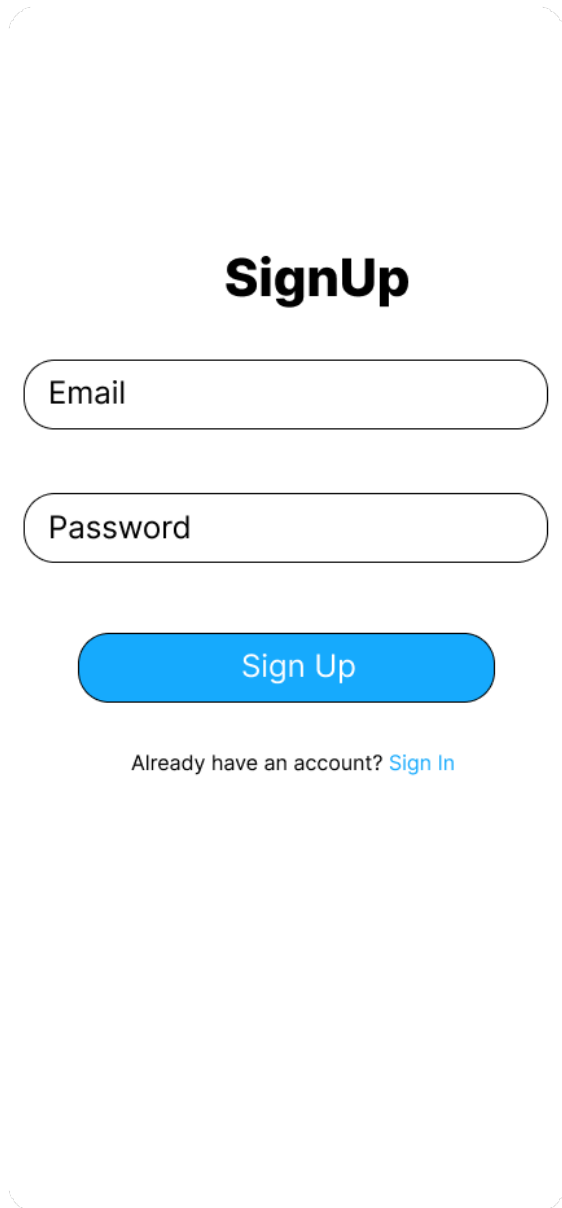
Login

Don't have an account? [Sign Up](#)

Garage Owners

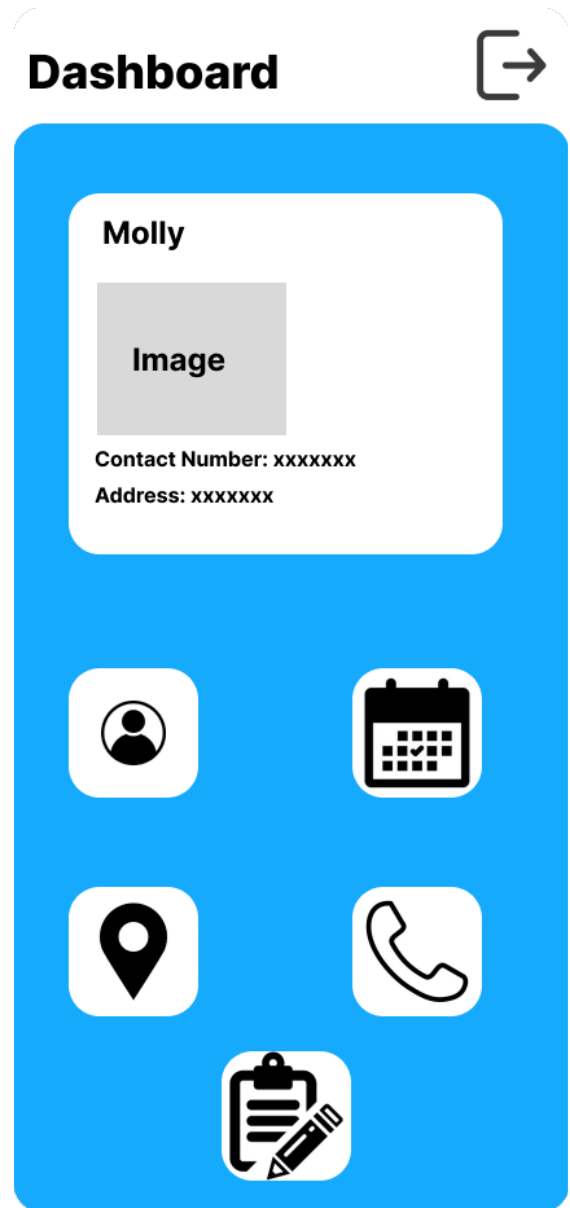
**Figure No.5: Splash Screen**

**Figure No.6: Login Screen**



The image shows a mobile application screen for signing up. It features a white background with a large, bold, black title "SignUp" at the top. Below the title are two rounded rectangular input fields, one labeled "Email" and the other "Password". Under these fields is a blue rounded rectangular button with the text "Sign Up" in white. At the bottom, there is a link that says "Already have an account? Sign In" in a smaller, blue font.

Figure No.7: SignUp Screen



The image shows a mobile application dashboard screen. It has a blue background. At the top, the word "Dashboard" is written in bold black text, followed by a white square icon with a black arrow pointing right. Below this is a white rounded rectangular card. Inside the card, the name "Molly" is at the top, followed by a grey square placeholder labeled "Image". Below the image placeholder, the text "Contact Number: xxxxxxxx" and "Address: xxxxxxxx" are displayed. Below the white card, there are five white rounded square icons on a blue background: a person icon, a calendar icon, a location pin icon, a telephone icon, and a clipboard with a pencil icon.

Figure No.8: Dashboard Screen

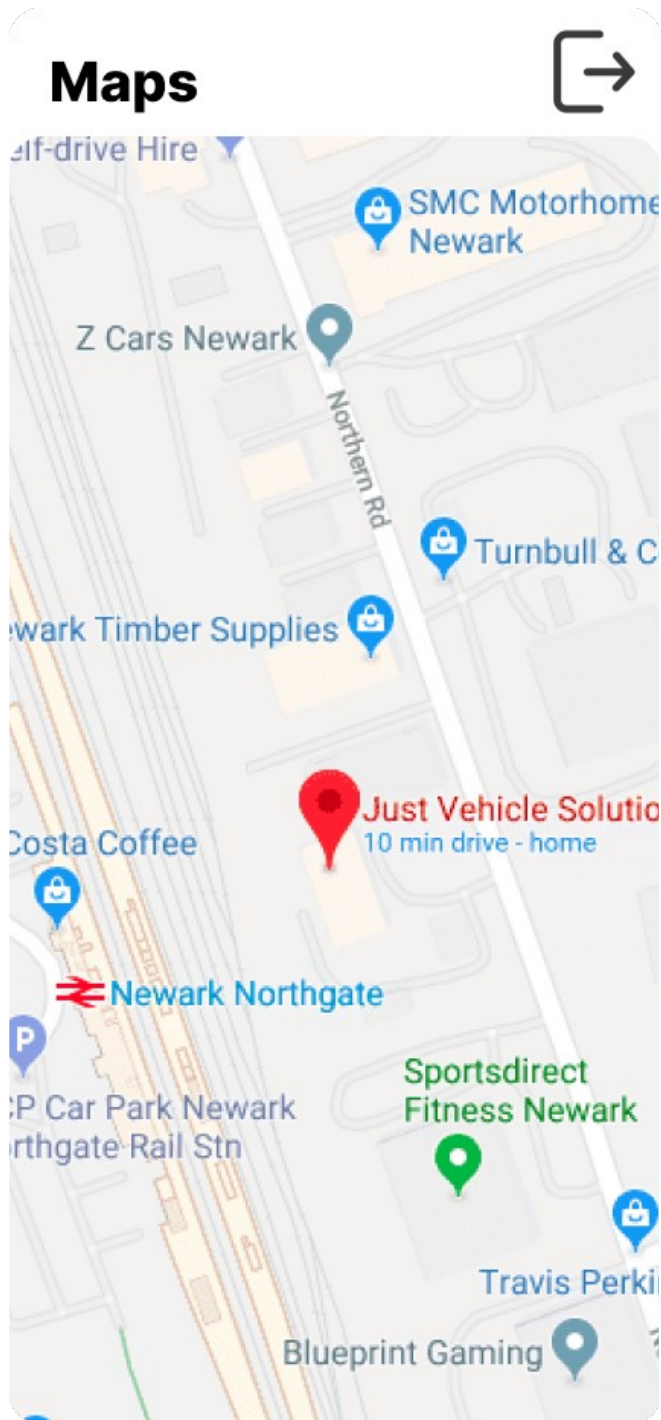


Figure No.9: Garage Locator Screen

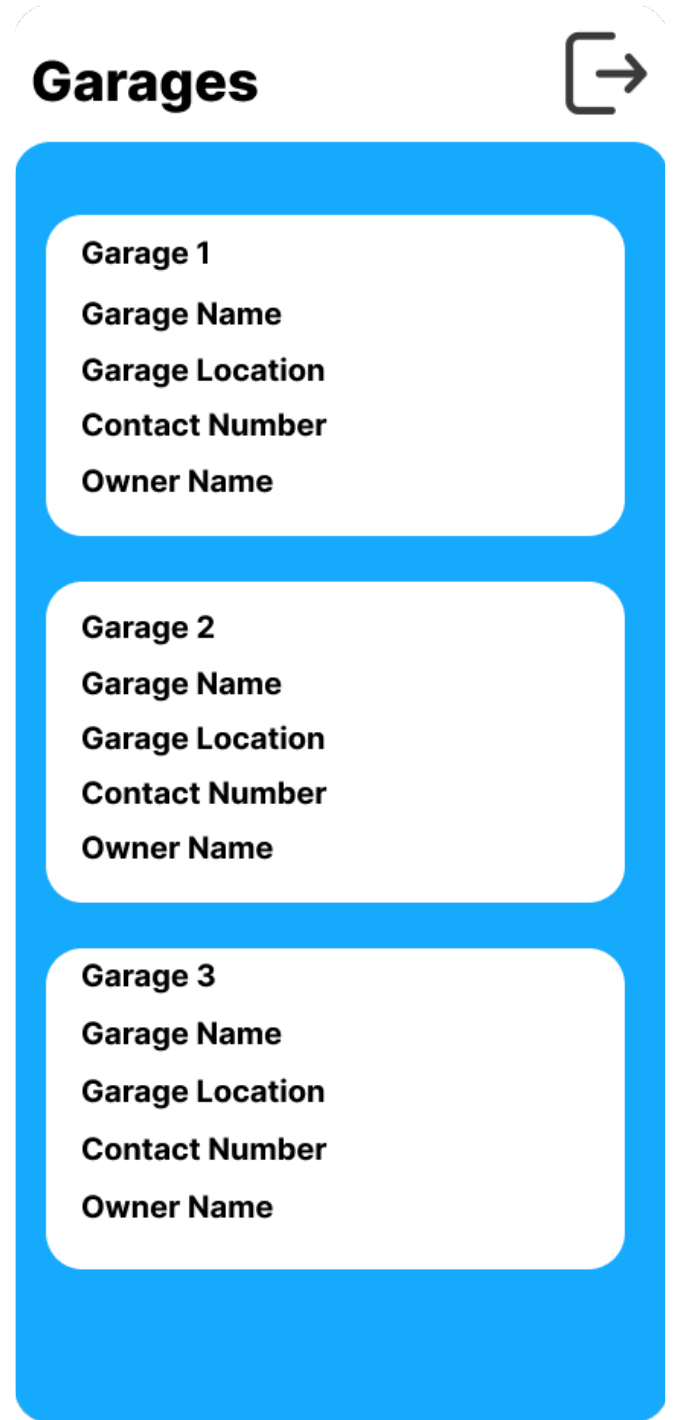



Figure No.10: Listed Garages Screen

**Contact Details** 

**Mechanic Name**

**Image**

**Contact Number**

**Address**

**Mechanic Name**

**Image**

**Contact Number**

**Address**


**Garage 3**

**Image**

**Contact Number**

**Address**

Figure No.11: Mechanic Details

**File a Complaint** 

**Upload a Picture**

**Image**

**Garage/Mechanic Name**


**Date:2023-12-27**

**Description**

**Submit**

Figure No.12: Filing Complaint Screen


## Your Profile



You may set up your phone number and username during the initial setup. For updating information second time and later, please click the button below

Figure No.13: Profile Update Screen

## Dashboard



### Upcoming Requests

#### Request Details

Name


Number

Service Type

Date

### Available Mechanics

#### Mechanic Name



Image

Contact Number

Address

Figure No.14: Garage Owner Dashboard Screen



## Annotation

Figma proved itself to be an invaluable asset for the creation of prototype for “Auto-Care Connect” application. The ease of access and versatility of Figma, helped creating a user-friendly, user-centered design for the application.

## Database Design

During the development phase, Firestore, a NoSQL database provided by Firebase was employed to store and manage data efficiently. Below, exemplary document structures for the prominent key collections on the Firestore are briefed:

### User Collection:

```
{
  "email": "user@example.com",
  "password": "hashed_password",
  "phoneNumber": "+977984567890",
  "username": "huerta_max"
}
```

### Garage Collection:

```
{
  "location": "Lokanthali Chowk, Kathmandu",
  "name": "Best Auto Garage",
  "number": "+977986543210",
  "owner": "Sailesh Chaurasia"
}
```

Figure No.15: JSON Format Document-oriented Schema-I.

The 'user' collection here represents the collection of email, hashed password, phone number, and username of the registered users. Whereas, the 'garage' collection is responsible for storing details about registered garages, such as their name, location, contact number, and owner information.

Contacts/Mechanics Collection:

```
{
  "address": "Banesor, KTM",
  "name": "SaileshChapagain",
  "number": "+1122334455",
  "image_url": "gs://your-firebase-storage/mechanic_images/mechanic123.jpg"
}
```

Complaint Collection:

```
{
  "date": "2023-01-01",
  "description": "My car is making a strange noise.",
  "image_url": "gs://your-firebase-storage/complaint_images/user123/complaint456.jpg",
  "name": "Raghu ",
  "uid": "user123"
}
```

Request Collection:

```
{
  "date": "2023-01-02",
  "name": "Seth Rogen",
  "number": ["+97798706566"],
  "service_type": "Oil Change"
}
```

**Figure No.16: JSON Format Document-oriented Schema-II.**

The 'contacts' collection, contains information about the registered mechanics including their name, address, contact number, and a link to their profile image stored in the Firebase Storage. Similarly, the 'Complaint' collection being responsible to store the complaints logged by the

users which includes capturing the date, description, user uploaded image link, user's name and user ID associated with each complaint. Finally, the 'request' collection stores the details like the date, user's name, contact numbers, and the requested service type.

## Mobile App Development/Implementation

The 'Auto-Care Connect' application is designed to act as a bridge between consumers and garage businesses of Nepal. The prominent feature of the app includes location-based garage locator, hassle free appointment bookings, on call mechanic services, which prominently comes handy in emergency roadside problems. The app uses flutter (for cross-platform) but is initially tailored for android devices as the android market of Nepal being dominant at a staggering 87.24% compared to that of 12.73% of iOS.

The development team comprised of three individuals, Pravin Timalisina, who took on the role of Project Manager and Back-end Developer, Rupesh Kumar Chaudhary, as the Project Lead Developer, and Subash Baskota as the Front-End Developer and Tester.

But, as the team had adopted the Scrum Methodology, all the team members actively engaged throughout the development and helped when any of the member needed help (coding included).

By utilizing various means of communications, such as online meeting, in-person meeting, and sharing of progress coupled with the culture of mutual support, the app was developed successfully.

The development of the application was carried out in a well-structured approach, under the instructions provided by the Agile Framework. All the sprints were carefully designed with each of them addressing multiple tasks based on the priorities. Elaboration of the development stages are mentioned below:

### Inception and Proposal

- Conducted thorough online research on mobile applications.
- Identified key stakeholders and engaged in discussions to understand target audience requirements.
- Initiated specifications by collaborating with stakeholders.

- Formulated and documented the Mobile App Proposal.

### **Requirements and Prototyping**

- Developed interactive prototypes to visualize the mobile app's interface.
- Conducted user interviews and surveys for comprehensive requirements analysis.
- Analyzed results, identifying functional, technical and usability requirements.
- Acquired necessary graphics for the app.
- Implemented the front-end using Flutter.

### **Backend Setup and Firebase Implementation**

- Established Firebase for backend functionalities.
- Designed and implemented a NoSQL database using Firebase.
- Initiated the development of backend scripts using Firebase functions.
- Implemented secure login functionality using Firebase Authentication.

### **Testing and Evaluation**

- Conducted rigorous functionality and usability testing to ensure a robust application.
- Evaluated the app's performance, gathering user feedback for improvement.
- Made recommendations for future enhancements based on testing outcomes.
- Compiled a comprehensive Mobile App Report.
- Prepared for the Mobile App Presentation.

The cross-platform compatibility, coupled with availability of learning resources, support and versatile UI toolkit of 'Flutter' compelled us to adopting it for the "Auto-Care Connect" development. However, the team decided to put a focus on the development for android devices as to align with the market dynamics in Nepal, where Android devices have a more significant presence.

Similarly, Firebase was utilized to handle the backend aspects of the application. Firestore, a NoSQL database from the Firebase service suite, was leveraged for its efficient data storage, whereas, Storage (another product from Firebase) has been utilized for managing images and media for the application.

Similarly, authentication for the application was also implemented using the Firebase (Authentication). Furthermore, Google Cloud Platform (GCP) played a pivotal role in enhancing the app's feature and functionality. GCP was specifically employed to embed Google Maps into

the application, allowing the users to leverage one of the most prominent features “Location-based Garage Locator” of the “Auto-Care Connect” application.

Visual Studio Code (VS Code) was leveraged as the primary IDE, whereas for testing various virtual devices (iOS 15, Pixel 3a) as well as physical smartphone (Huawei, Various Samsung models) were used, enabling the development team to complete the project with ease.

```
Future<void> displayPrediction(  
    Prediction p, ScaffoldState? currentState) async {  
    GoogleMapsPlaces places = GoogleMapsPlaces(  
        apiKey: kGoogleApiKey,  
        apiHeaders: await const GoogleApiHeaders().getHeaders(),  
    ); // GoogleMapsPlaces  
  
    PlacesDetailsResponse detail = await places.getDetailsByPlaceId(p.placeId!);  
  
    final lat = detail.result.geometry!.location.lat;  
    final lng = detail.result.geometry!.location.lng;  
  
    markersList.clear();  
    markersList.add(  
        Marker(  
            markerId: const MarkerId("0"),  
            position: LatLng(lat, lng),  
            infoWindow: InfoWindow(title: detail.result.name),  
        ), // Marker  
    );  
};
```

**Figure No.17: Snippets from the page Handling Maps.**

In the fig.no.15, the code is a portion of the ‘CurrentLocationScreen’ (complete commented code in the Appendix), that displays a Google Map centered on the user’s current location by accessing the device GPS.

```
child: StreamBuilder<QuerySnapshot>(
  stream:
    FirebaseFirestore.instance.collection('contacts').snapshots(),
  builder: (context, snapshot) {
    try {
      if (!snapshot.hasData) {
        return CircularProgressIndicator();
      }

      var documents = snapshot.data!.docs;

      return Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          for (int i = 0; i < documents.length; i++)
            ContactCard(
              contactNumber: documents[i]['number'],
              contactName: documents[i]['name'],
              contactAddress: documents[i]['address'],
              imageStoragePath: documents[i]['image_url'],
            ), // ContactCard
        ],
      ); // Column
    } catch (e) {
      print('Error fetching data from firestore:$e');
      return Text('Error Fetching data. Please try again!');
    }
  },
);
```

**Figure No.18: Snippets demonstrating some of the Firebase & FireStore Implementation.**

Similarly, in fig.no.16, the code is a portion of the 'ContactsPage', which displays contact details of registered mechanics to the user. The image is updated from the Firebase Storage, while other details like name, address, number are updated from the Firestore of each mechanics and displayed to the user via Contact Cards.

```
void _pickImage() async {  
  final picker = ImagePicker();  
  final pickedFile = await picker.pickImage(source: ImageSource.gallery);  
  
  setState(() {  
    if (pickedFile != null) {  
      _image = File(pickedFile.path);  
    }  
  });  
}  
  
void _submitComplaint(BuildContext context) async {  
  // Set _isSubmitting to true when starting the submission  
  setState(() {  
    _isSubmitting = true;  
  });  
  String name = nameController.text;  
  String description = descriptionController.text;  
  String currentDate =  
    DateFormat('yyyy-MM-dd').format(DateTime.now().toLocal());  
  
  if (_image != null) {  
    // Uploading the image to Firebase Storage  
    String fileName = DateTime.now().millisecondsSinceEpoch.toString();  
    Reference storageReference =  
      FirebaseStorage.instance.ref().child('complaint_images/$fileName');
```

**Figure No.19: Snippets demonstrating image picking and uploading to the Firebase.**

Likewise, fig.no.19, the code represents a portion of the 'ComplaintPage' which lets the users to pick image from their gallery and store the chosen picture to the Firebase Storage, and coupled with various text fields enabling users to provide a descriptive problem that they might face.

With the choices made like incorporating hybrid technology, Firebase and Google Cloud Platforms, the team's emphasis on learning from diverse sources, modern and efficient approach for a dynamic app development is clear.

## Testing

Testing of the “Auto-Care Connect” involved a meticulous planning and examination of various critical aspects to ensure its reliability in the real world. This report includes some of the major tests performed amongst others. Test like functionality testing to verify user interactions, unit testing, performance testing to ensure the application’s responsiveness under varying conditions, integration testing to ensure seamless collaboration between front-end, back-end (Firebase) and Google Cloud Platform (API for Maps), as well as compatibility testing to ensure smooth functionality across multiple devices alongside with evidence are included in this report.

Notably, the report here represents a focus on manual testing procedures and does not include code-based tests that were conducted. This deliberate choice was influenced by the factor that, manual testing, especially with multiple devices would provide a brief usage of the application in the real-world scenario all the while encompassing tests of major components of the application.

Below are the tables comprising of test plan and test logs from the testing of the “Auto-Care Connect”.

### Test Plan

| Test ID | Test Case             | Description                                | Preconditions  | Test Steps  | Expected Result.                                      |
|---------|-----------------------|--|----------------|---|---|
| 1       | Signup Functionality. | Verify that users can sign up for the app. | App installed. | 1.Open the app.<br>2.Navigate to the signup screen.<br>3.Enter valid signup details.<br>4.Tap “Sign Up” button. | Successful signup: user directed to the login screen. |



|   |                      |   |   |  |  |
|---|----------------------|---|---|--|--|
| 2 | Login Functionality. | Verify that users can log in using Firebase authentication.                 | User account exists, app installed.               | 1.Open the app.<br>2.Enter valid credentials.<br>3.Tap “Login” button.   | Successful login: user directed to the dashboard.              |
| 3 | Image Functionality. | Check if stored images can be loaded from Firebase Storage under 5 seconds. | User logged in; image stored in Firebase Storage. | 1.Navigate to the screen where the image is loaded.  | Image is displayed on the screen within 5 seconds.             |
| 4 | Image Upload         | Test the capability to upload Firebase Storage.                             | User logged in.                                   | 1.Navigate to the complaint page: from where the images can be uploaded.<br>2.Select an image and initiate the upload. | Image is successfully uploaded to Firebase Storage.            |
| 5 | User Data Display    | Display user data based on the logged-in user’s ID from the firestore.      | User logged in; data available in Firestore       | 1.Navigate to the profile page where users’ data is displayed and updated.   | User data is correctly displayed (based on uid) on the screen. |
| 6 | Multi-Device Testing | Test the app on multiple devices for ensuring compatibility.                | App installed on different devices                | 1.Install app on different devices.<br>2.Verify consistent functionality across devices.                               | App works seamlessly across multiple devices.                  |

### Test Log

| Date | Test | Input | Expected Output | Actual Output | Screenshots |
|------|------|-------|-----------------|---------------|-------------|
|------|------|-------|-----------------|---------------|-------------|

|            | ID |   |  |  | References.         |
|------------|----|---|--|--|---------------------|
| 2023/12/17 | 1  | Valid Signup Details.                     | Successful signup (validated entries): user directed to login page.          | Successful signup: redirected to the login screen. | Figure No. 18,19,20 |
| 2023/12/17 | 2  | Valid login credentials.                  | Successful login: user directed to the dashboard.                            | Successful login: dashboard displayed.             | Figure No. 21,22,23 |
| 2023/12/22 | 3  | Navigate to image loading screens.        | Image is displayed on the screens.   | Image displayed on the screens.                    | Figure No.23        |
| 2023/12/24 | 4  | Upload an image.                          | Image is successfully uploaded to Firebase Storage.                          | Image successfully uploaded.                       | Figure No. 24,25    |
| 2023/12/27 | 5  | Navigate to user data screen.             | User data is correctly displayed on the screen (based on the logged user ID) | User data displayed as expected.                   | Figure No. 26,27    |
| 2023/12/28 | 6  | Install and use app on different devices. | App works consistently on various devices.                                   | Consistent functionality across devices            | Figure No. 28,29,30 |

Below are the images/screenshot from the simulator as well some of the physical devices, captioned according to their purpose respectively.

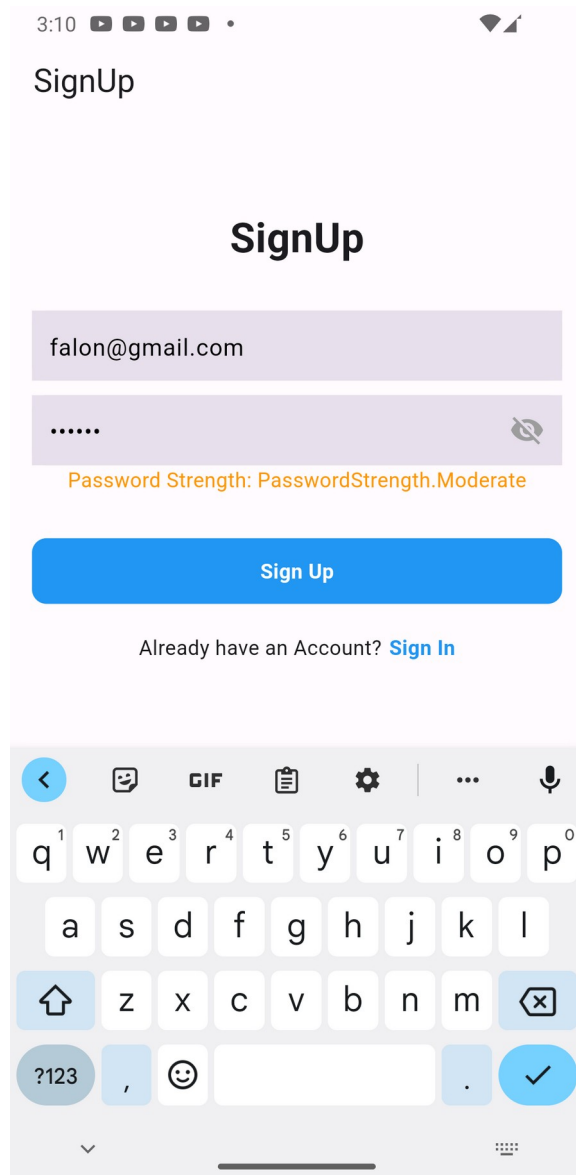


Figure No.20: SignUp With Valid Credentials.

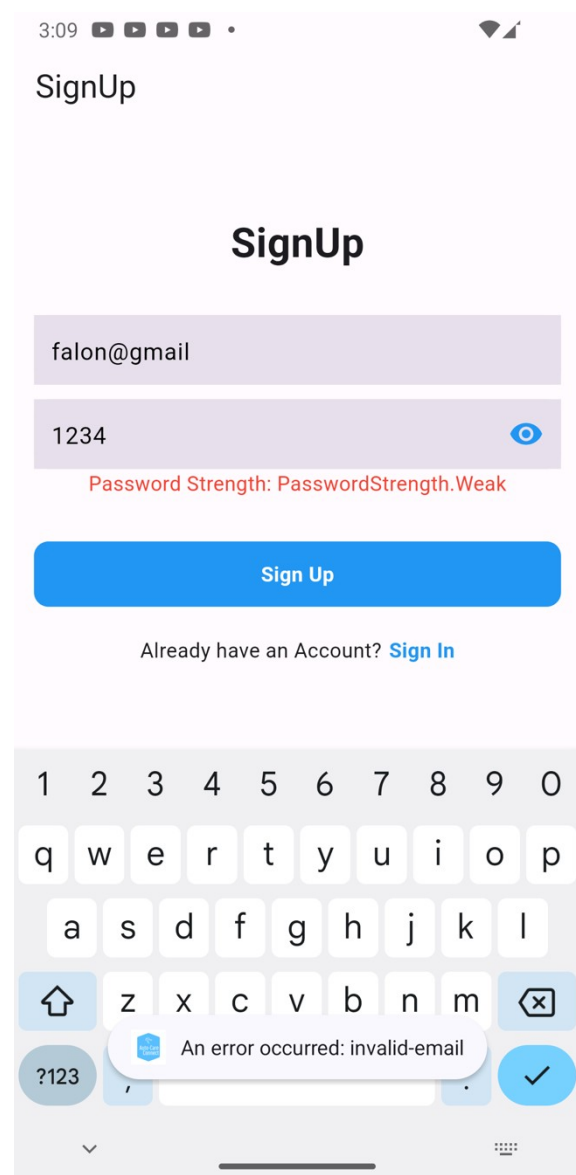


Figure No.21: SignUp With Invalid Credentials

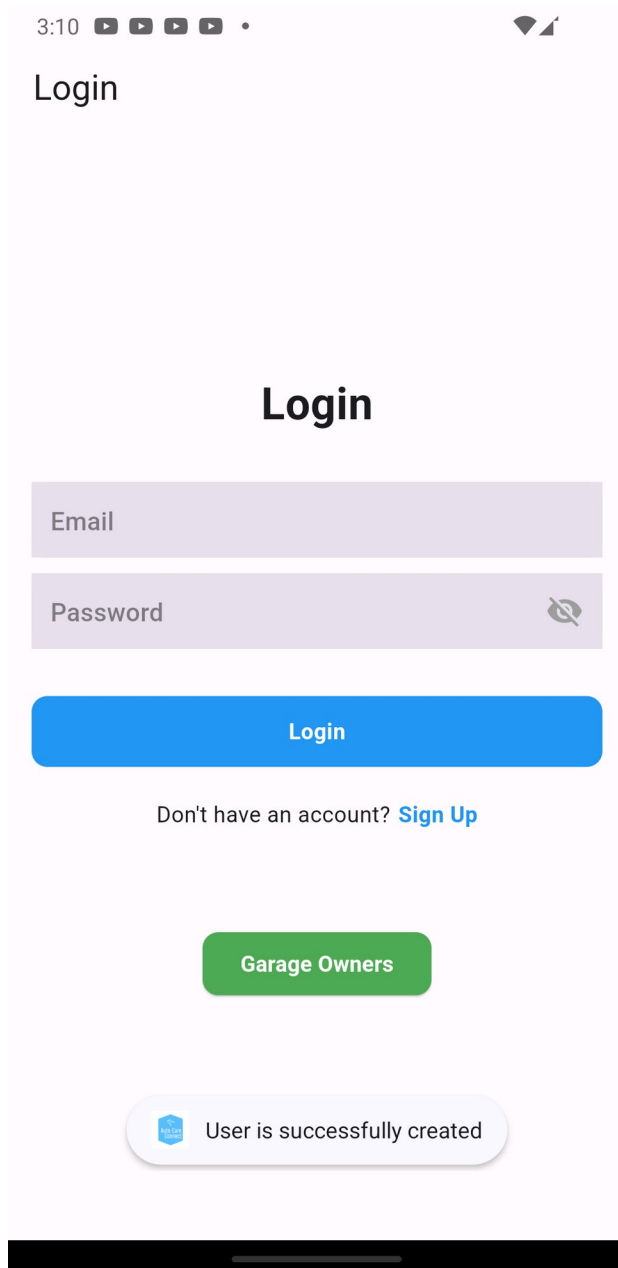


Figure No.22: Signup Validated and Directed to Login

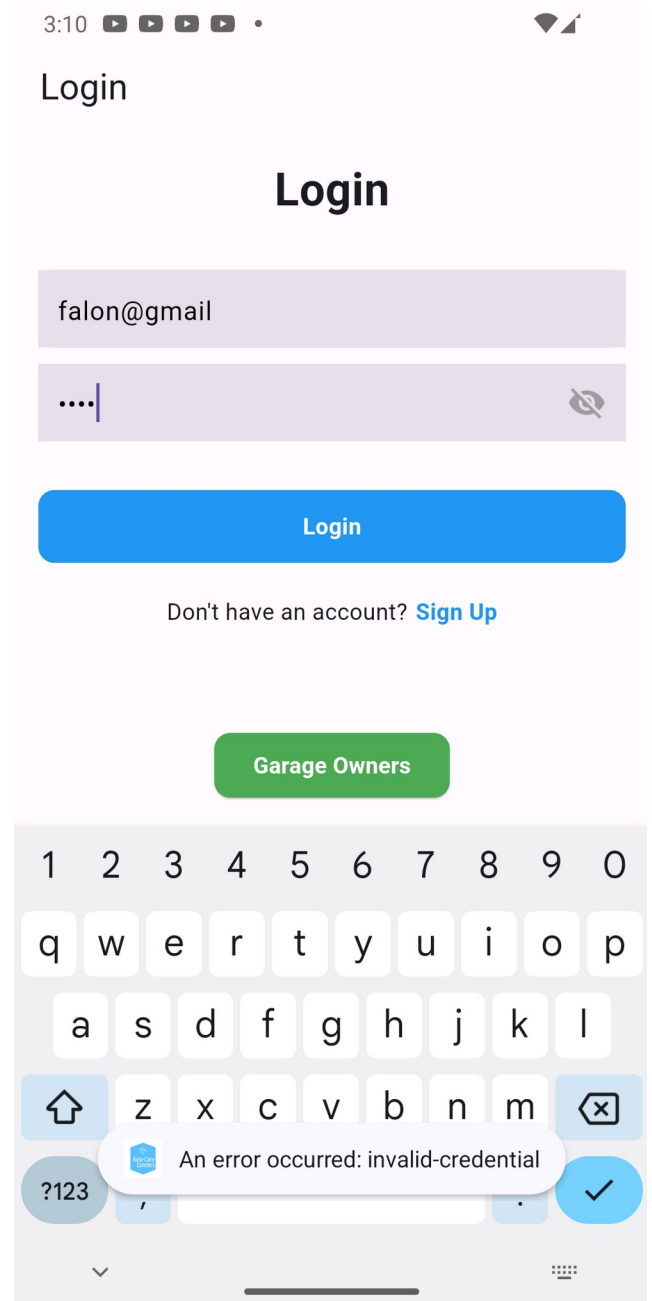


Figure No.23: Login with invalid credentials


3:11

## Login

**Login**

falon@gmail.com

.....




Don't have an account? [Sign Up](#)

**Garage Owners**

1 2 3 4 5 6 7 8 9 0  
q w e r t y u i o p  
a s d f g h j k l  
↑ z x c v b n m ↵  
?123 , . ✓

3:11


## Dashboard





**Random**


**Contact Number: 9806674385**


Address: Bhaktapur

 **Profile Update**

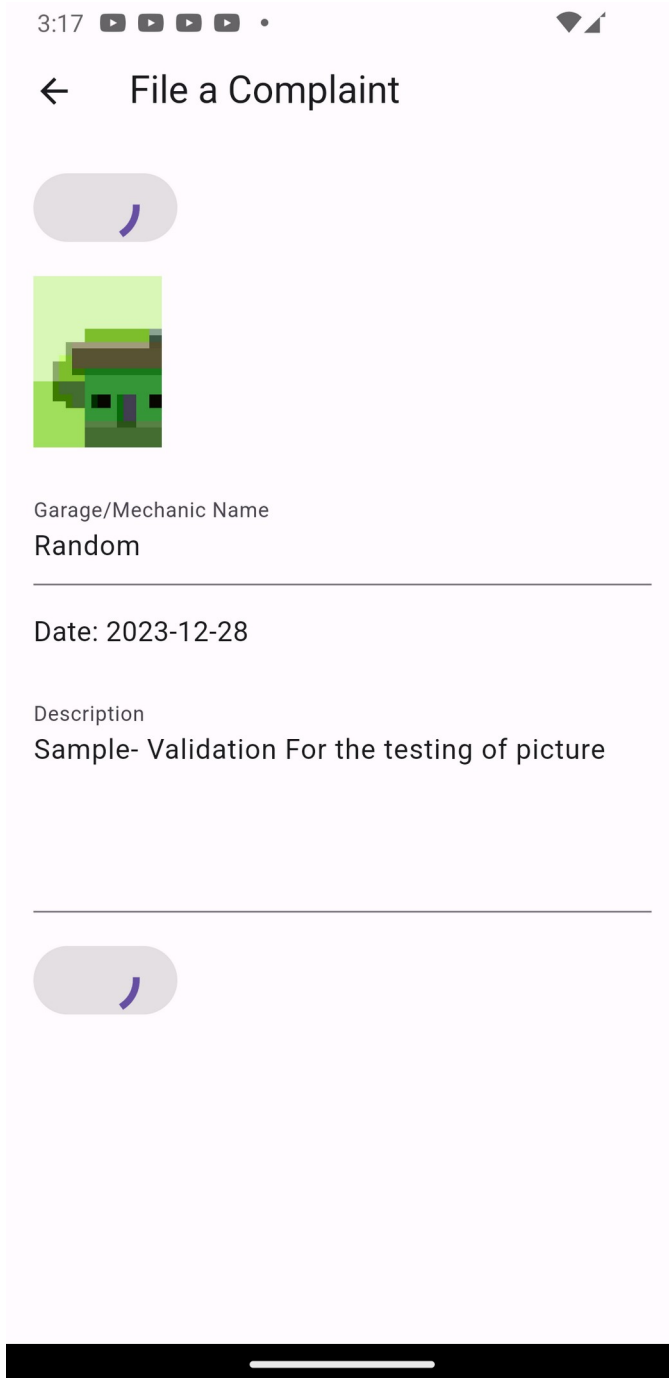
 **Make Appointments**

 **Find Garages**

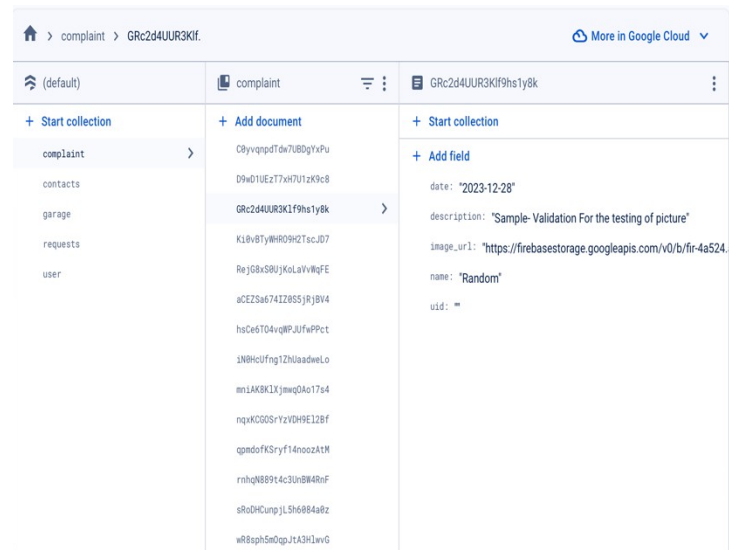
 **Call Mechanic**

 **Complaint**

**Figure No.24: Login authenticating credentials.  
 Database.**



**Figure No.25: Verified user directed to  
 Dashboard/Images from the**



**Figure No.27: Image  
 stored in storage  
 and referenced on  
 Firestore.**

Figure No.26: Image Upload initiated from

device

The screenshot shows a mobile application interface with a status bar at the top displaying the time 3:22, signal strength, and battery level. The app's header is titled "Your Profile" with a share icon on the right. The main content area has a blue background and is titled "Update Your Profile!". It contains four input fields: "Email" with the value "falon@gmail.com", "Password" with masked characters "\*\*\*\*\*", "Phone Number" with the value "12332", and "Username" with the value "falon Jimmy". Below these fields is a white "Update" button. At the bottom, there is a white button labeled "Request Profile Update" and a paragraph of text: "You may set up your phone number and username during the initial setup. For updating your profile info from the second time and onwards, please click the button below".

3:22

Your Profile

### Update Your Profile!

Email  
falon@gmail.com

Password  
\*\*\*\*\*

Phone Number  
12332

Username  
falon Jimmy

Update

You may set up your phone number and username during the initial setup. For updating your profile info from the second time and onwards, please click the button below

Request Profile Update

Figure No.29: Firestore Data of the user.

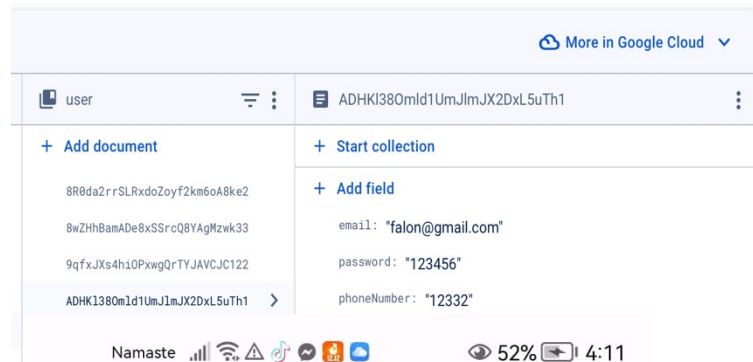


Figure No.28: Profile based on UID.

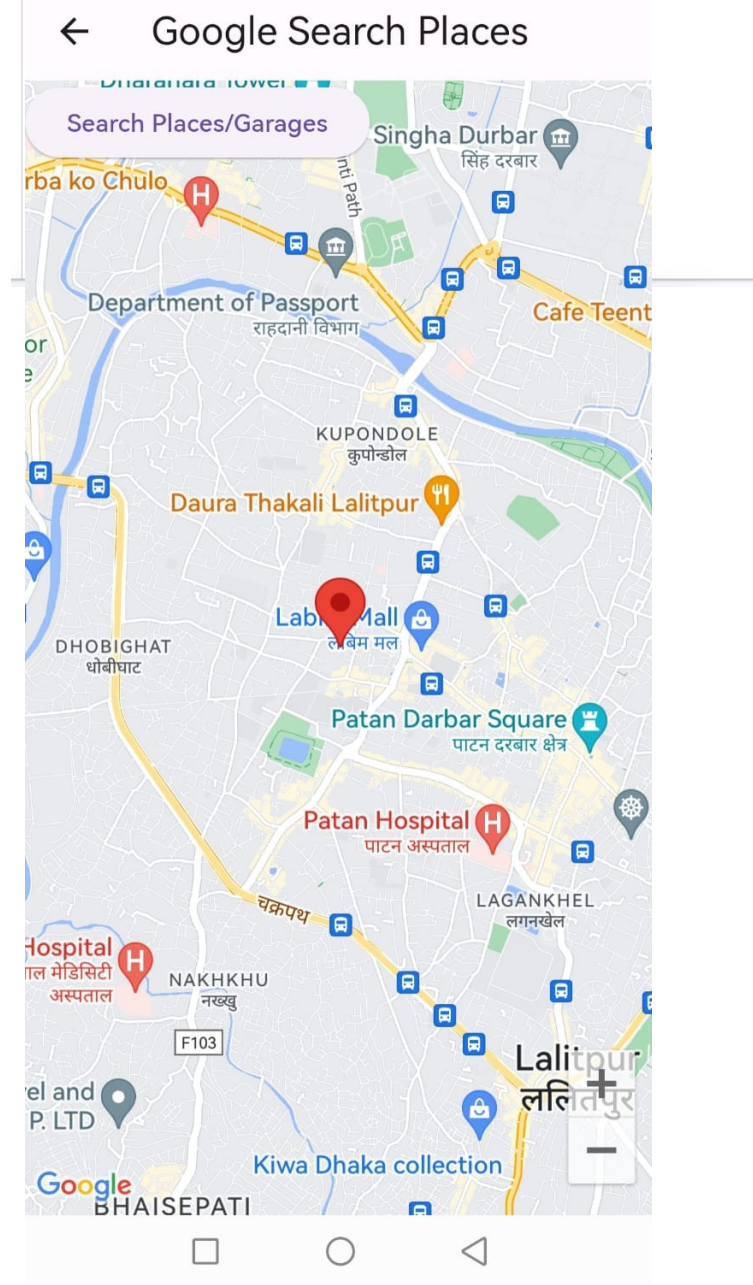
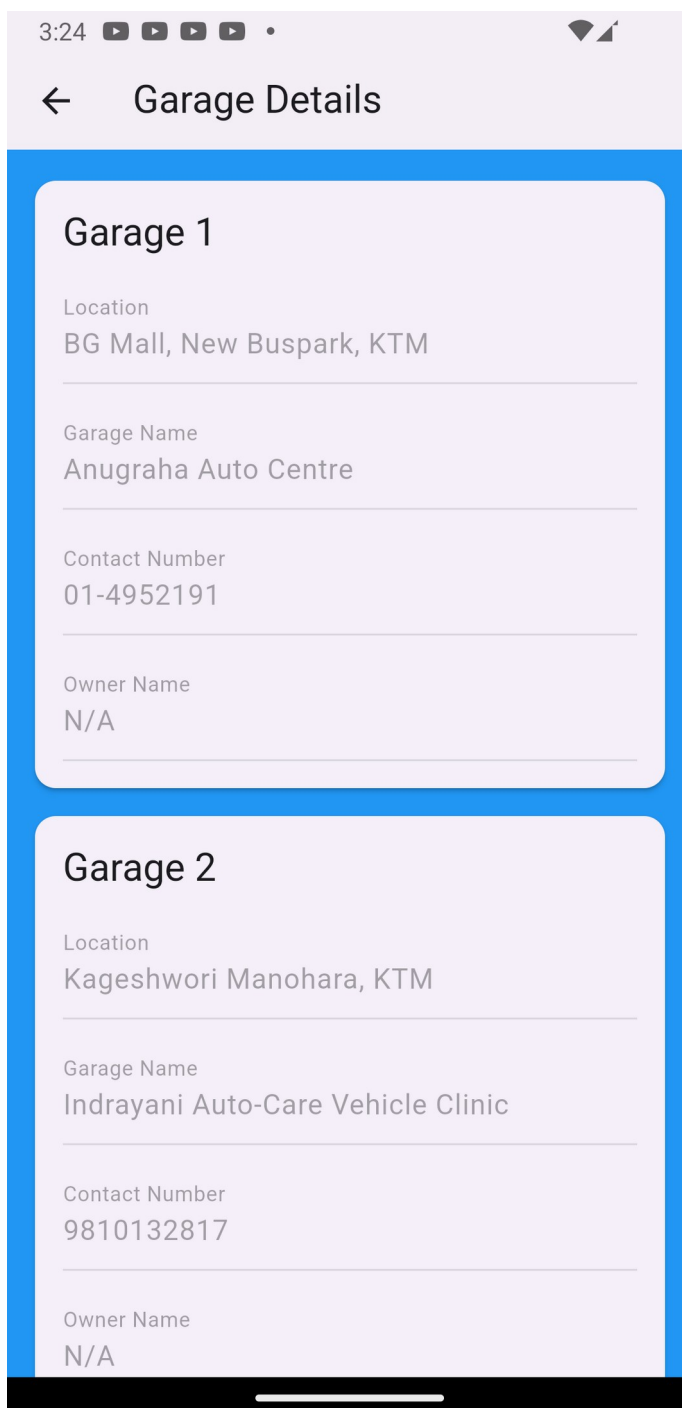




Figure No.30: App running on a simulator.

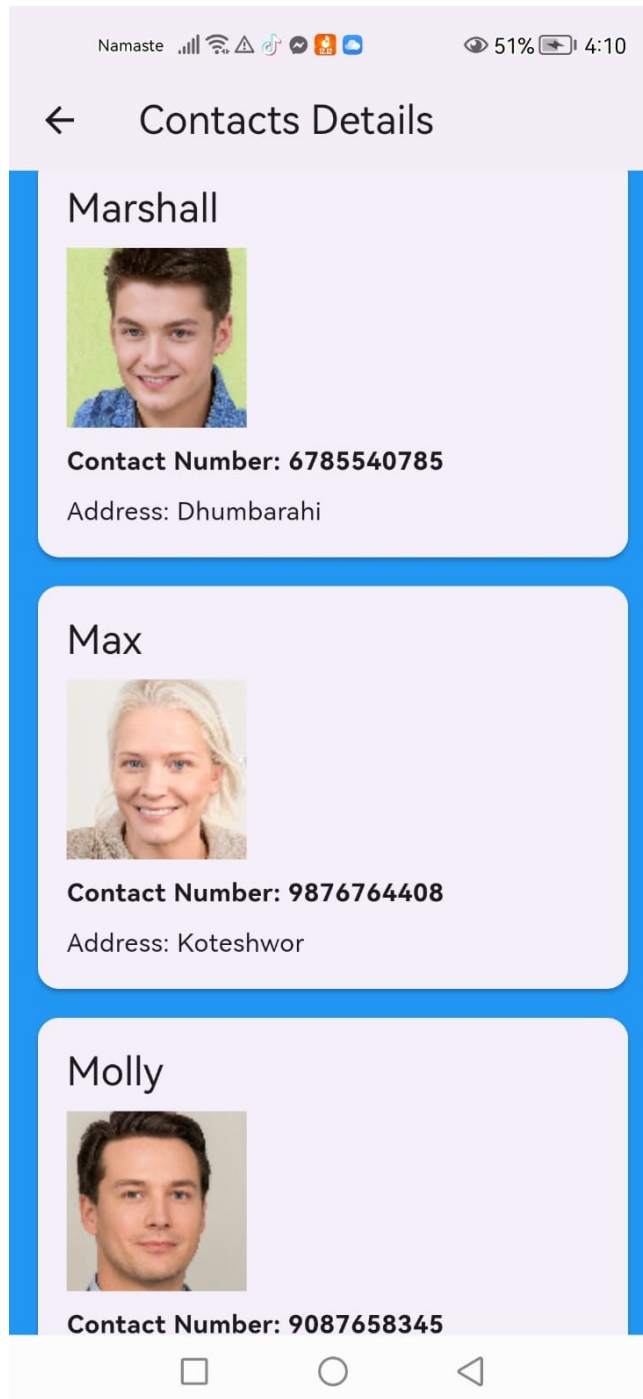


Figure No.31: App running on a physical device.

**Figure No.32: App running on a physical device.**

## Mobile App Evaluation

As with any application, evaluation of the 'Auto-Care Connect' was done to ensure its usability and user interface design align with the expectations and needs of its target audience. The 'DECIDE' framework was employed to create the plan and to analyze the findings (outcome) for the mobile app evaluation.

### Plan

#### Define

To ensure a detailed assessment of the app's development, the evaluation plan comprised a multi-stage process focused on key aspects like usability, user satisfaction, and functionality. These aspects were quintessential for creating a user-centric application.

#### Explore

To exhaustively evaluate the app, multiple methods were employed as given below:

- Usability Testing: The participants were asked to perform task within the app while closely observing their performance and feedback.
- Surveys: To gather the information regarding their experience with the app, the participants were requested to complete surveys.
- Heuristic Evaluation: To make a comparison of the app with established design of the existing application in the market.

## **Choose**

The evaluation involved two primary target groups:

- Potential Users: Individual who own vehicles and are interested in using a mobile app for garage services.
- Stakeholders: Team members who were directly involved in the development of the application and entitled to ownership and all the further marketing strategies.

## **Identify**

The evaluation plan comprised specific success factors and key performance indicators (KPI) for each stage to evaluate the effectiveness of the app and its compliance to user-centered principles.

### Usability Testing

- Success Factor: Participants could complete the tasks within the app with ease.
- KPIs: User satisfaction, task completion rate, task completion time and error occurrences.

### Surveys

- Success Factor: Users were satisfied with the overall app experience.
- KPIs: User satisfaction rating, Net Promoter Score (NPS) and the willingness to recommend the app to others.

### Heuristic Evaluation

- Success Factor: The app aligned perfectly with the established usability heuristics.

- KPIs: Quantity of usability violation identified along with the severity of usability violations.

## **Deliver**

A detailed report to be created comprising the key findings of this evaluation. This report will act as a medium to communicate the findings while providing recommendation for refining and enhancing the application.

## **Evaluate**

The overall evaluation plan is to be inspected based on:

- Completeness: To ensure that the plans cover all the key aspects of the app.
- Appropriateness: To get assured whether the selected methods aligned with the specific areas to be evaluated.
- Feasibility: To make sure that the evaluation plan could be executed given the available resources and time constraints.

## **Ethical issues**

To carry out the evaluation process smoothly, it is quintessential to address the ethical and practical concerns to maintain the integrity of the evaluations. The concerns primarily involve:

- Obtaining Consent: All the participants were required to provide informed consent before participating in the evaluations.
- Protecting User Data: All the data collected from the participants were treated in accordance with privacy laws and regulations, guaranteeing data security and privacy

protection.

## Evaluation Results

After conducting a thorough evaluation using the plan formulated using the “DECIDE” framework for the “Auto-Care Connect” application, with the specific user groups, we obtained more insightful results that helped us to make qualitative user experience.

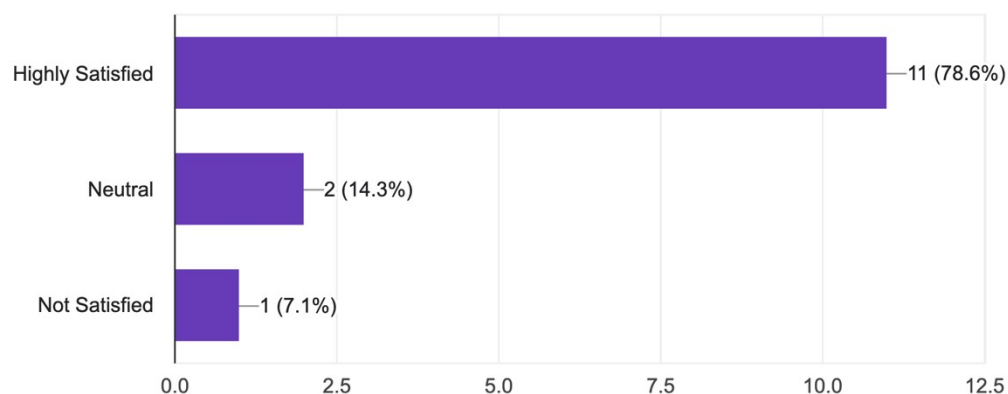
### Survey Questionnaires asked during Evaluation Phase.

1. How satisfied are you with the overall experience of using the app?
2. On a scale of 1 to 10, how likely are you to recommend this app to others?
3. Did you find it easy to navigate and perform tasks within the app?
4. How would you rate the speed and responsiveness of the app?
5. Were you able to successfully complete the tasks you attempted within the app?
6. Did you encounter any difficulties or errors while using the app?
7. What features of the app do you find most useful?
8. How would you rate the visual design and layout of the app?
9. Do you feel that the app meets your expectations for a user-friendly experience?
10. In what ways do you think the app could be improved to better meet your needs?

In the usability testing, participants displayed a high success rate while completing tasks within the app. Key Performance Indicator (KPI) comprising of user satisfaction, task completion rate contributed to a positive assessment.

How satisfied are you with the overall experience of using the app?

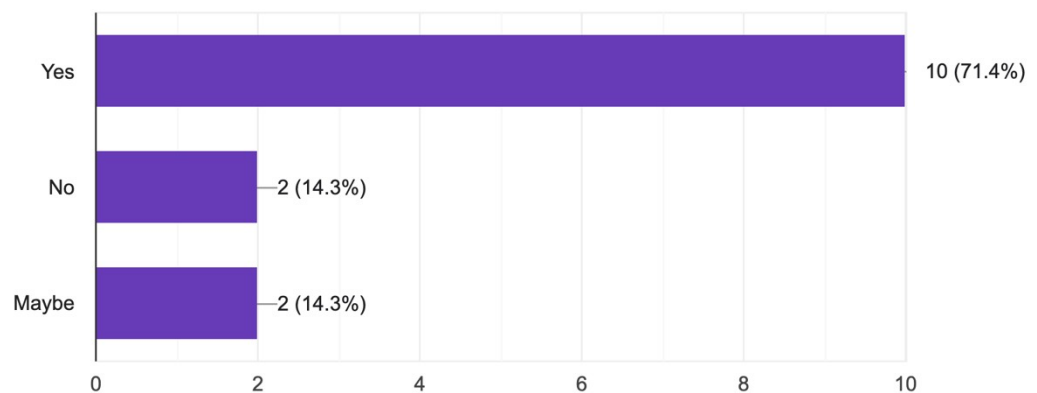
14 responses



**Figure No.33: Survey results demonstrating High satisfaction.**

Do you feel that the app meets your expectations for a user-friendly experience?

14 responses

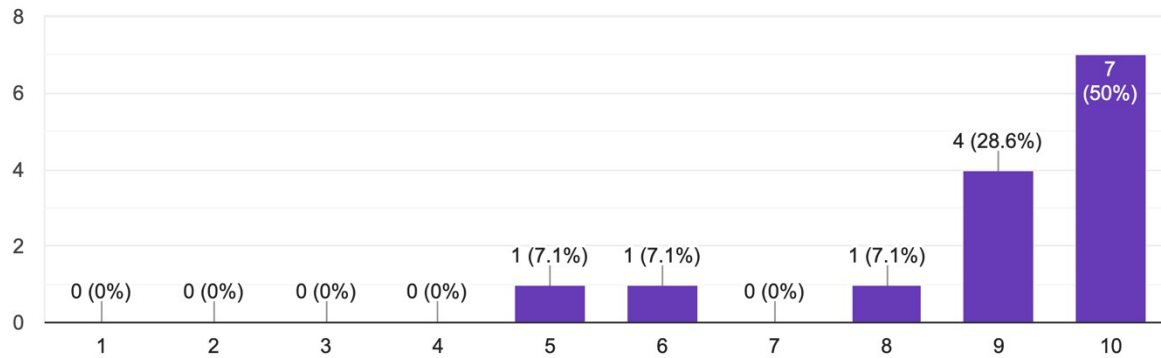


**Figure No.34: Survey results demonstrating majority users finding the app user-friendly.**

Analyzing the surveys result, the overall app satisfaction is clearly high. During the interviews, users expressed contentment with their experience as well, while expressing a strong desire to recommend the “Auto-Care Connect” app to others.

On a scale of 1 to 10, how likely are you to recommend this app to others

14 responses

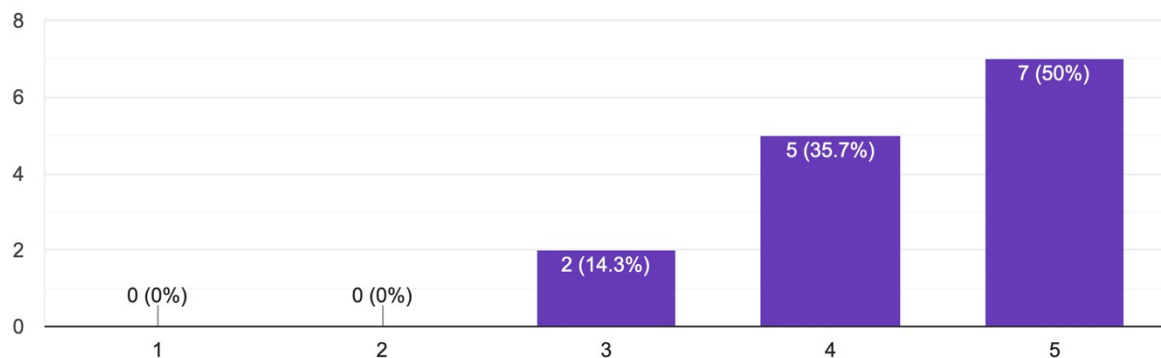


**Figure No.35: Survey results demonstrating High number of participants willingness to recommend app to others.**

The evaluation further revealed that the app perfectly aligned with established usability heuristics, indicating a compelling design. The survey results, as evident in the Figure No. 34 makes the statement more evident.

How would you rate the visual design and layout of the app?

14 responses



**Figure No.36: Survey results demonstrating satisfaction of user regarding the visual layout.**

Complying to the user feedback, a noteworthy adjustment was made to the app's user interface for making appointments. While the proposal design of the app had the interface, where the users were able to enter the related information directly, majority of the participants preferred communication through phone calls for making bookings (also preferred by virtually all the garage owners). So, to enhance the user experience and align with the market preference, the initial releases of the app, will feature, clicking on a listed garage to direct users to initiate phone call with selected garage.

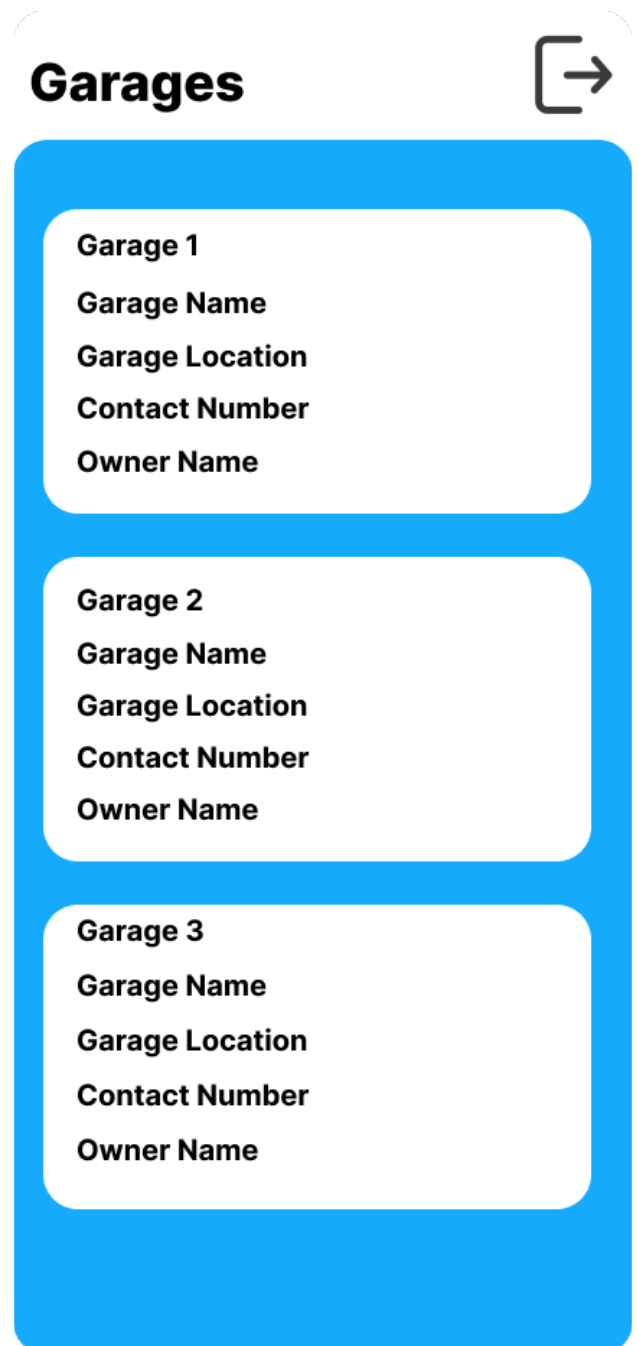




The 'Make An Appointment' form is a purple rounded rectangle. It features a back arrow icon at the top left. The form contains the following fields and controls:

- Appointment Date:** A white rounded input field with a calendar icon on the right.
- Appointment Time:** A white rounded input field with a clock icon on the right.
- Select a Garage:** A white rounded input field.
- Your Name:** A white rounded input field.
- Vehicle Type:** A white rounded input field.
- Checkboxes:** A checked checkbox followed by the text 'Select For Pick Up & Drop-Off'.
- Address:** A white rounded input field.
- Make Booking:** A white rounded button with a shadow at the bottom.

Figure No.37: Proposal UI design for appointment booking.



The 'Garages' screen is a blue rounded rectangle with a share icon at the top right. It displays a list of three garage entries, each in a white rounded box:

- Garage 1:** Garage Name, Garage Location, Contact Number, Owner Name
- Garage 2:** Garage Name, Garage Location, Contact Number, Owner Name
- Garage 3:** Garage Name, Garage Location, Contact Number, Owner Name

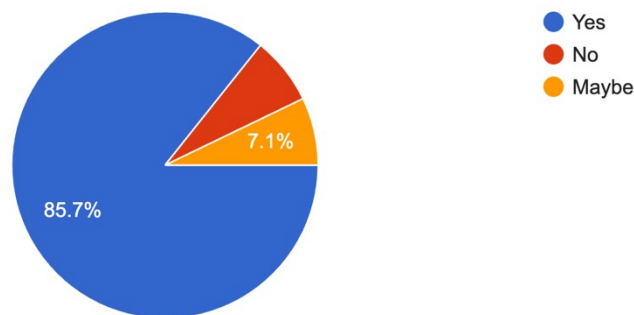
Figure No.38: App UI design for Appointment booking

(based on evaluation analysis)

However, the strategic UI modification reflects a commitment of the development team to meet the current user needs. It's worth to acknowledge that this UI modification does not disregard the original UI design for the appointment booking completely, rather it serves as an extension, that provides identical functionality while satiating the predominant preference for phone calls of the market.

Did you find it easy to navigate and perform tasks within the app?

14 responses

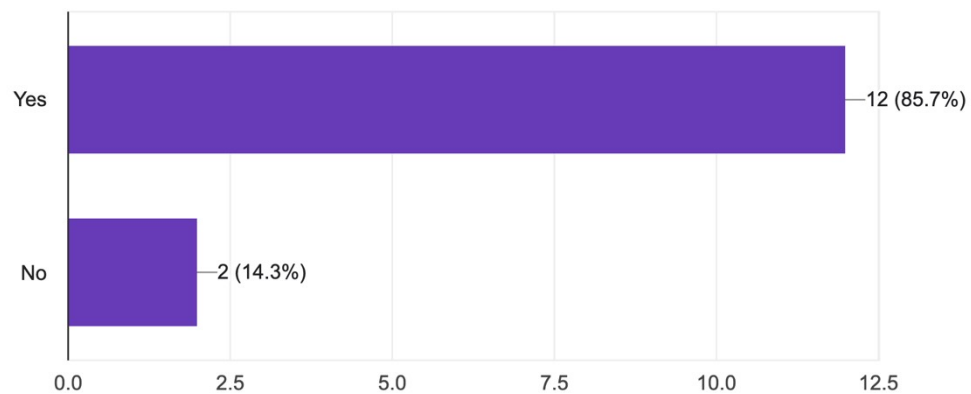


**Figure No.39: Survey Results Indicating Easy navigation on the app.**

Further, the evaluation revealed, that most users found the app remarkably easy to navigate and perform tasks, highlighting, the design to be intuitive and user-friendly experience.

Were you able to successfully complete the tasks you attempted within the app?

14 responses

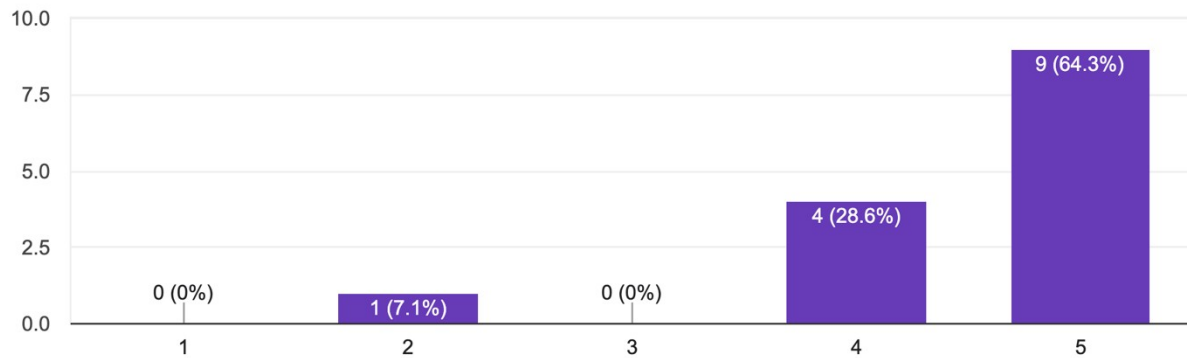


**Figure No.40: Survey results demonstrating high rates of task completion.**

Similarly, the app was highly rated in the context of speed and responsiveness. This positive response clearly indicates the app's efficient backend infrastructure and well-optimized front-end design, further supported by the outcomes of the in-person interviews revealing the same response from the participants.

On a scale of 1-5, how would you rate the speed and responsiveness of the app?

14 responses

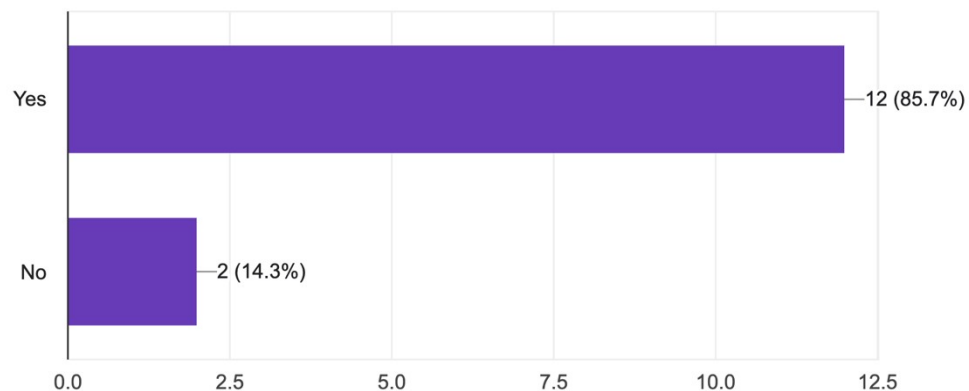


**Figure No.41: Survey results showing high ratings for the app's speed and response.**

Notably, the evaluation of the user's ability to successfully complete tasks within the app was also included in the app evaluation. Like most of the aspects, majority of users reported positive feedback for the ability to perform tasks successfully with ease, expressing satisfaction in achieving their goal.

Were you able to successfully complete the tasks you attempted within the app?

14 responses



**Figure No.42: Survey results showing majority participants completing their task with ease.**

It is safe to say that the app's user centered approach, coupled by the strategic adaptation based on the thorough evaluation, resulted a highly satisfactory user-experience. The planned integration of a detail-enterable UI for making appointments in the future iterations demonstrates a commitment to continuous improvement.

## Mobile App Store Optimization (ASO) and Marketing Strategy

### Marketing Strategy for Auto-Care Connect

#### **Social Media Presence**

- Establish a solid presence on popular social media platforms.
- Build a community around the app by continuously sharing posts, blogs, and updates.
- Use comments, posts, and contests to regularly keep up with the followers.

#### **Content Marketing**

Make blogs, articles for not only Auto-Care but vehicle maintenance, road safety and automotive industries as well. Establish the name "Auto-Care Connect" as the go-to resource for any information regarding the automotive industries.

#### **Influencer Marketing**

Promote the app by partnering with influencers to create content, reviews, and tutorial all the while building trust and credibility amongst the audience.

#### **Partnership with Garages and Mechanics.**

Emphasize the benefits of increased visibility and customer reach by using the app and develop partnership with local garages and mechanics to promote the app.

#### **Promotional Campaigns.**

- Offer discount for first-time users, referral bonuses, as well as seasonal promotions.
- Highlight the prominent problem-solving nature of the app like saving time, convenience, and guaranteed auto-care services.

### Monetization Strategy.

The following serve as a tentative monetization strategy for the Auto-Care when it hits the market.

1. **Garage Subscription Model:** Garages would pay a monthly fee to get listed on the platform, resulting them to get benefits like gaining visibility to a wider audience.
2. **Mechanic Subscription Model:** Individual mechanics, will also pay a minimal monthly fee to be listed on the platform, helping them to establish their name gradually in a bigger market.
3. **Admin-Based Subscription Model:** Garages would pay for an admin-based subscription, which would let the garages to reap the benefits like monthly statistics, boosted visibility, user review and rating management, and regular updates.
4. **Consumer (Maps) Subscription Model:** Consumers would pay to use the Location-based garage locator if they want to access nearby garages instantly. The maps would be filtered with the registered garages to the application and displayed to the users, based on the location that the user is searching, hence, mitigating the loss of the garage businesses to get overlooked as well.

### App Store Optimization (ASO) Strategy for Auto-Care Connect

#### Keyword Research

Identify relevant keywords related to garage locator, appointment booking, mechanic services, vehicle pickup by using tools like Google Keyword Planner, Sensor Tower to find high-ranking keywords with low competition.

Some of the relevant keywords are listed below:

Auto care, Garage Locator, Appointment Booking, Mechanic Services, Vehicle pickup, Roadside Assistance, Car Repair Services, Automotive industry, Mobile mechanic, Emergency Bike

services, On-demand auto assistance, Local garage finder, Auto service scheduler, Vehicle breakdown assistance, Bike maintenance app, Nearby mechanic locator

### **Optimized App Title and Subtitle**

Including main keyword in the app title for increasing visibility as well as, utilize the subtitle to provide additional relevant keywords and key features.

Viable App Title: “Auto-Care Connect: Vehicle Services & Roadside Assistance”

Subtitle: “Find Nearby Garages, Book Appointments, On-demand Mechanics and 24/7 Roadside Help”

### **Compelling App Description**

Concise yet descriptive app brief, describing key features and benefits.

Most suitable (Room for improvement): “Welcome to Auto-Care Connect, your all-in-one solution for seamless auto care and assistance. Designed to simplify your vehicle maintenance journey, offering a range of features like Location-based Garage Locator, Appointment Booking to Garages, On-Demand Mechanic Services, Roadside Assistance for Emergencies and many more.”

### **Beautiful Screenshots and Video**

Visually appealing screenshots that boast app’s user interface and functionalities. At the same time, incorporate demonstration video to provide a walkthrough of the app’s core functionalities.

### **Localized Metadata**

Translate the app metadata including title, subtitle, and description, into Nepali language to broaden app’s reach.

### **User Reviews**

Encourage users to leave positive reviews within the app. Prompt response to user feedback to improve overall app ratings.

## Regular Updates

Updating the app regularly with new features (including user suggestions) and improvement.

With its robust ASO strategy coupled with diversified monetization model and targeted marketing approach, the “Auto-Care Connect” is confident for its success in the Nepal’s market. The primary goal is the expansion of its user-base alongside establishing a sustainable revenue, backed up by continuous monitoring, adaptation of strategies based on user-feedback as well as market trends with the aspiration to be the go-to solution for every auto-care needs of Nepal.

## Group Member Contribution List

| Student ID             | Student Name | Role                                    |
|------------------------|--------------|---|
| PRAVIN TIMALSINA       | 2133104      | Project Manager and Back-End Developer. |
| SUBASH BASKOTA         | 2148322      | Front-End Developer and Tester.         |
| RUPESH KUMAR CHAUDHARY | 2154880      | Lead Developer.                         |

NOTE: Contribution of each member are further elaborated on the Reflection section the report.

## Discussion/Critical Analysis/Reflection

The successful development of the project involved a combination of effective group work, commitment to learning and adopting along with strategic time management. The decision to employ and adhere to the guidelines of the Agile Scrum Methodology helped directly with time management for the project.

By breaking down the project into manageable sprints, adaptation to evolving requirements, the development team was able to maintain a focused and aligned timeline backed up further by regular planning session and finally produce a well-rounded application.

Through a well-organized approach to learning, team collaboration, and guidance from the unit supervisor, the development team was successful to implement all the requirements as well as



additional requirements like interaction with other apps, GPS, Google API, advance solution for efficient saving and loading of data specified in the assignment brief.

The choice for this specific application development was based on the clear potential of the app to disrupt the auto-motive digital market in Nepal. However, with a meticulous calculation, given the timeframe and team size, the development team adopted a practical approach by prioritizing achievable goals. This deliberate choice helped us to strike a perfect balance between delivering a product that met the outlined requirements as well as showcase the ambition of the application.

As the project had a time-constraint, the team optimized the available time for learning, preparation and developing. The development team ensured that each stage of the project received required attention as the adoption of the Agile Scrum Methodology guided our decision making. Despite the challenges faced during the development of the application (mostly regarding the knowledge gathering), the team successfully met the challenge.

Working on the project the team has cultivated a deeper understanding of the multiple aspects of an effective teamwork and contemporary technologies. Using the emerging technologies like Firebase and Google Cloud Platform, has widened our horizon in the technical aspects, but at the same time, it showcases our team members commitment to stay relevant with tools integral for modern app development.

Similarly, the hand-on experience with Flutter gave us intricate details of cross-platform frameworks. At the same time creating marketing strategies, optimizing App Store and the evaluating the app with multiple techniques widened our knowledge far more than just the technical realm.

We realized the importance of not only creating a functional application but also the importance of user experience and enhancing the discoverability and relevance of the application within the target audience.

The development team showcased unity by utilizing each member's distinct strength. Specific roles were assigned to each member like one member was assigned with role of the project management and backend development, while the second member was responsible as team's lead developer and the third member of the team was assigned with the front-end development and testing responsibilities. But with the culture fostered by mutual respect, each member helped and worked, when needed beyond their designated roles.

Management of the project was facilitated by the blend of both, Agile and Scrum methodologies. Work distribution was structured meticulously based on the unique strengths and preferences of each team member, ensuring every team member contribution.

Regular meetings, sharing of insights and progress throughout the development phase, played a vital role in providing transparent work environment. This approach of continuous communication ensured the cultivation of unity amongst the team member and allowed us for quick and informed adjustments as needed. In short, the project management framework, finely attuned to the project's changing needs and the shared goal of team, evolved onto a flexible system.

A noteworthy mention, which showcases the development team's dedication to the users' needs and preferences can be highlighted by the fact that, after the evaluation phase of the application, a pivotal refinement of the product was done. During the evaluation process, surveys and interviews revealed a noticeable trend in the auto-motive service market of Nepal.

The original method designed for the appointment booking, where users could enter details into the app was well received and appreciated, yet the market trends indicated a strong preference for traditional phone call to make appointment, among both consumers and garage businesses.

Being new to the market, it's important for any product to align with the existing preferences. So, we decided to modify the app's appointment booking process (for initial releases), with the aim to capture a wider market. This showcases the commitment of the application towards its audience as well as a strategic intervention to enhance its appeal and relevance in the local context.

Nonetheless, complying to the market research and the app's evaluation analysis, we achieved our primary goals. However, there is always room for improvement, so when the app successfully captures the target audience and shows positive hold on the auto-motive sector of Nepal, we would add more capabilities as deemed necessary by the trends and users, and to name some of the planned features are payment options, a web version for desktop accessibility and the original functionality of letting users to enter details for appointment booking among others.

Working on this project, greatly increased our knowledge regarding the mobile app development. As we used flutter for the development of our app, we got the practical experience in cross-platform development. Not only that, as we worked with the emerging

technologies like Firebase and Google Cloud Platform, but most of the concepts were also new to us, which prompted us to enhance our skills and stay updated on industry trends.

The project made it clear that, for a successful app development and its launching in the market involves multiple factors besides just technical proficiency and united teamwork, rather market research, thorough evaluation, and strategic deployment, all play an equal role.

Technically speaking, the application provided an in-depth insight to an array of software engineering concepts. We were able to learn, hone our skills and practically use concepts like Agile, Scrum, API usage, Cloud Computing (with Google Cloud Platform) as well as had the privilege to get hands-on experience in design, testing, evaluating, and marketing a mobile application, working as a team successfully.

Our project journey can be marked by effective group collaboration, meticulously planned time management, and a serious commitment to continuous learning. We successfully met the specified requirements, gained invaluable insights into mobile app development, project management, and emerging technologies.

At the same time, we developed a valuable skill of problem-solving nature by successfully navigating through the challenges that we faced during the development of the “Auto-Care Connect” application.

This experience has motivated us to not only adopt a methodical approach for all the future projects, but also enhanced our technical skills and at the same time added fuel to our desire of contributing to the software engineering landscape.

## **Team Member’s Individual Roles/Responsibilities/Reflection.**

### **Project Manager and Back-End Developer Pravin Timalisina. University ID-2133104**

I took on the role of project manager and back-end developer for our mobile application development. I got the opportunity to not only guide our team through technical difficulties but also witnessed the evolution of the mobile app proposal, that I had created for my assignment 1 and this project being an extension of the same.

As the project manager, my decision of combining Agile and Scrum methodologies was heavily influenced by the foundational work done in the initial proposal as well as in agreement with all

my team members. The market (primary and secondary) research and the initial requirement analysis from my proposal served as the foundation for the development of our application.

Utilizing the proposal, I was able to focus on developing a collaborative environment. I was able to specify roles to each member of our team by assessing their individual strengths and preferences. Using regular check-ins and open communications, and a supportive team, I was able to ensure the successful development of our application.

Not only that, with the foundational layout built from the help of the proposal, I was able to further contribute to my team and take on the role of back-end developer and conduct the app evaluations as well as create a comprehensive marketing strategy for the application.

In the ever-evolving field of software engineering, I was inclined to explore the realm of Cloud Computing. With a mutual agreement from my fellow members, I was confidently able to employ Firebase for the back-end development of “Auto-Care Connect” application.

This choice, coupled with the decision to use Flutter for front-end, enabled our team to successfully complete the project that met all the specified requirements (including the additional requirements) in the assignment brief and at the same time gave me an opportunity to get intricate details on technical aspects of Firebase, Google Cloud Platform, API usage and NoSQL databases.

As the development progressed, I had the privilege to conduct the app evaluation process guided from the plan that I had created during the proposal assignment using DECIDE framework, which allowed us to make further refinements to our product, aligning with the user’s preferences and market trends. Not only that, I also got the opportunity to learn the process of marketing a software application and from the knowledge that I could gather in the restricted time-frame I created brief marketing strategies for our application.

Handling the overall responsibilities of decision making and developing a mutual respect amongst the team member as the project manager, this experience has boosted my self-esteem as well as technical knowledge for a successful project development. Similarly, working on the back end of the app, I now have a detailed knowledge on the realm of Cloud Computing, API usage and NoSQL database, which all were new to me.

Working with an enthusiastic team, and performing my roles and responsibilities for the project, I now have a deeper understanding of project’s lifecycle, from its conceptualization in the initial proposal to its execution in the development phase. The experience from this project has provided me a boost to continue my exploration on the various field of software engineering and development.

**Lead Developer.**

**RUPESH KUMAR CHAUDHARY. University ID-2154880**

As the lead developer of our team, I took the in handling the technical side of our project. As, we followed Agile-Scrum methodology, I also covered a range of tasks in our project while focusing mainly on creating the visual representations of our software architecture.

I took on the responsibility of designing use case diagrams, which provided a clear blueprint of how our system's component interacted with external entities, which smoothened the process of development. Not only that, I applied my skills to develop detailed activity diagrams, which guided our team to understand the flow of activities within the system.

Beyond system designing, I took the responsibility of creating a strategic App Store Optimization (ASO) plan. By investigating and learning the realm of ASO, I was able to create a strategic plan that is poised to provide a boost for the app and increase its visibility among targeted Nepali market.

As our team adopted Agile method, all the members helped each other, so, I also participated in creating two important pages of our app, "Dashboard" and "Profile Update". With this effort, I made sure to provide my contribution on the front-end part of our app to add appealing visuals, guided by our front-end developer.

Working on this project, I got to know the dynamics of a teamwork and realized the multi-facets of my role. This experience not only increased my technical skills, but also deepened my knowledge of need for collaboration and development of a successful software solution.

In summary, I was able to shape the technical foundation of our project by designing the system, strategic planning in ASO and hands-on coding contribution. This experience has shaped my approaches for all my upcoming projects and gave me a confidence to be a contributing part of a larger team.

## **Front-End Developer**

**Subash Baskota. University ID-2148322**

In this project, I, took the role of Front-end developer and Tester for our application. I was responsible to design the overview user-interface of the application. From the early development stage, we conducted online meetings with other team members, such as team manager and lead developer to first select the technologies.

After mutual agreement amongst team members, I chose figma to develop a user-centric user interface for our application (various changes made than that of the proposal as decided with the timeframe in mind). I often consulted and informed our team members, regarding the design of the application at every stage and consulting about the features to be implemented and changing the design accordingly.

After the results were satisfactory to the group members, I started to implement the design into code using Flutter. Implementing all features and designing the application was a journey. As informed by our project manager and decided among all team members, firebase was to be used for our database as well as cloud storage as it was one of the emerging technologies based on cloud computing.

It also offered various features such as cloud storage, authentication, NoSQL as well as a complementary platform to easily integrate Google Maps (from the Google Cloud Platform) into our application. Working with firebase provided me a learning experience and usage of API and the designing the front end according to the back-end function provided to me, was a learning experience and relatively simple to integrate to the front-end that I was responsible for creating. We often conducted group meetings and online meetings during the process so that the team was updated with the development process.

Another role that I took on was the role of testing and for it I planned and performed various tests on the application such as cross-platform testing, functional testing, UI testing etc. As we opted for a hybrid technology (Flutter), the application functioned seamlessly across various devices. After a comprehensive, meticulously curated planned testing, everything turned out to be normal and successful.

In simple words, the experience from this project was very insightful and positive. By learning and implementing our software successfully, I got to contribute and witness the successful completion of a software application. Challenges occurred during the implementation phase of the project but with firm determination and working in collaboration with our team, I was able to tackle every hurdle that came into the way of project completion. The project not only gave me a hands-on experience with the emerging technologies like Firebase but also gave me the

intricacies regarding the front-end aspect of a mobile application, which I am confident to say that it has opened a vast array of opportunity in the realm of software engineering as I got the opportunity to successfully learn and test a piece of software.

## Conclusion

In conclusion, the journey of creating “Auto-Care Connect” stands as a witness for the power of effective collaboration, commitment to continuous improvement, and meticulous planning. The development process was fueled by a thorough understanding of user’s needs, deduced from the primary and secondary market research, to provide a user-oriented application, aiming to facilitate the digital auto-motive services of Nepal with the help of the app’s features like Location-based Garage Locator, appointment bookings, On-Demand Mechanic services and Emergency Roadside assistance.

The team was able to complete the project with a systematic and strategic approach, structured with sprint model, we initialized the project by performing extensive online research and stakeholder engagement, resulting in a comprehensive Mobile App Proposal.

To get a clear understanding of requirements, we utilized user interviews, surveys and interactive prototypes aligning within the user-centric design principles. The decision to use Flutter for front-end development and Firebase alongside Google Cloud Platform for back-end functionalities was solely based on the intention of leveraging contemporary technologies.

Our approach was focused on creating a user-friendly product, which was not only technically efficient but also in compliance with the market trends in the auto-motive sector of Nepal. To achieve this, we prioritized user feedback, conducted rigorous testing throughout the project, while strategically employing firebase functions and authentication mechanism to ensure a secure experience.

The success of the project was directly influenced by the effective project management comprising both Agile and Scrum methodologies, further coupled with transparent work and timely adjustments. Our work met, all the specified requirements (in the assignment brief including the additional requirement) and at the same time showcases our commitment to excellence.

Working on the project, our understanding of mobile app development, emerging technologies and the intricacies of project management is significantly boosted. The working experience from the project has motivated us for continuous learning and adopting holistic approach in future ventures. The skills and knowledge gained from this project will shape our approach to future projects and challenges, helping us to excel and contribute to the field of software engineering.

## References

Scribd. (n.d.). *Introduction of Automobile Industry of Nepal | PDF | Car | Nepal*. [online] Available at: <https://www.scribd.com/document/585872860/Introduction-of-Automobile-Industry-of-Nepal> [Accessed 16 Oct. 2023].

Krug, S. (2009). *Don't Make Me Think*. Pearson Education.

Patton, J. (n.d.). *User story mapping: discover the whole story, build the right product*. O'Reilly.

Smashing Magazine. (2019). *Smashing Magazine — For Web Designers and Developers — Smashing Magazine*. [online] Available at: <https://www.smashingmagazine.com/>.

Interaction Design Foundation (2010). *UX Design Courses & Global UX Community*. [online] The Interaction Design Foundation. Available at: <https://www.interaction-design.org/>.

Martin, R.C. (2010). *Clean code a handbook of agile software craftsmanship*. Upper Saddle River [Etc.] Prentice Hall.



Ries, E. (2011). *The lean startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 1st ed. New York: Crown Business.

Li, K. and Wu, M. (2006). *Effective GUI Testing Automation*. John Wiley & Sons.

Eyal, N. and Hoover, R. (2014). *Hooked: how to build habit-forming products*. New York, New York: Portfolio/Penguin.

Singh, S. (2021). *Beginner's Guide To The Digital Marketing*. Sukhdeep Singh.

## Appendix

### **Scripts of Interview**

#### Introduction:

The interviews were introduced to the interviewer, and the purpose of the interview was explained. Informed consent was obtained from the participants to ensure their willingness to participate.

#### Understanding Challenges:

The interviewees were asked about the common challenges they encounter in vehicle maintenance and repair. They were also questioned regarding their usual methods for locating a suitable garage for vehicle servicing.

#### Experience with Mobile Apps:

Participants were inquired about their history of using mobile apps related to automotive assistance. If they had previous experiences, they were asked to describe their experiences and the specific features and functionalities they expected from such apps.

#### Frequency of Garage Visits:

The frequency of the interviewees' visits to garages for vehicle servicing or repairs within a year was explored.

#### Expectations from a Mobile App:

The interviewees were prompted to share their expectations from a mobile app that connects them to automotive assistance. They were specifically asked about the features and services they deemed most valuable and whether they had a need for on- demand mechanic services.

#### Smartphone Usage:

The participants were asked about their regular use of smartphones for various tasks, including those related to automotive activities.

#### Economic Impact:

The economic impact of vehicle maintenance and repairs on the interviewees' daily lives and economic well-being was discussed.

#### Closing:

The interviews concluded by expressing gratitude to the participants for sharing their valuable insights. Any additional information about the "Auto-Care Connect" app or the next steps, as needed, were provided.

#### **Script for Garage Owner Interviews:**

## Introduction

The interviewees were introduced to the interviewer, and the purpose of the interview was explained. Informed consent was obtained from the participants to ensure their willingness to participate.

## Understanding Garage Operations:

Garage owners were questioned about the typical operations and services offered at their garages. They were asked about how customers usually find their garage for servicing.

## Challenges and Customer Needs:

Common challenges and issues faced by garage owners in their operations were discussed. Garage owners were encouraged to share insights into the specific needs and preferences of their customers.

## Experience with Mobile Apps:

Participants were inquired about their consideration of using mobile apps to enhance their garage's services or customer engagement. They were also asked about the features they found valuable in such an app.

## Technological Adaptation:

Garage owners were asked about their garage's tech-savviness in terms of using digital tools for operations and customer interactions.

## Economic Impact:

The direct impact of the garage's performance on the local community and economy was explored.

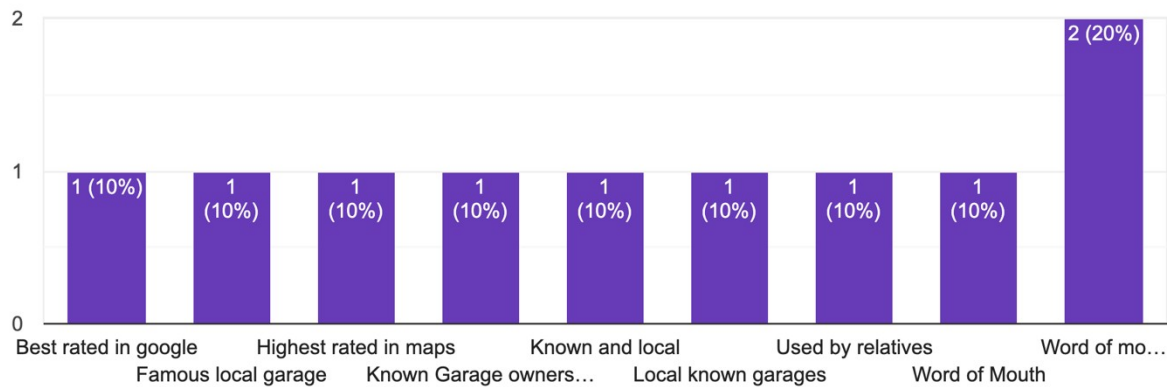
Closing:

The interviews concluded by expressing gratitude to the participants for sharing their valuable insights. Any additional information about the “Auto-Care Connect” app or the next steps, as needed, were provided.

**Survey Questionnaires (Result Stat) from Primary Market Research.**

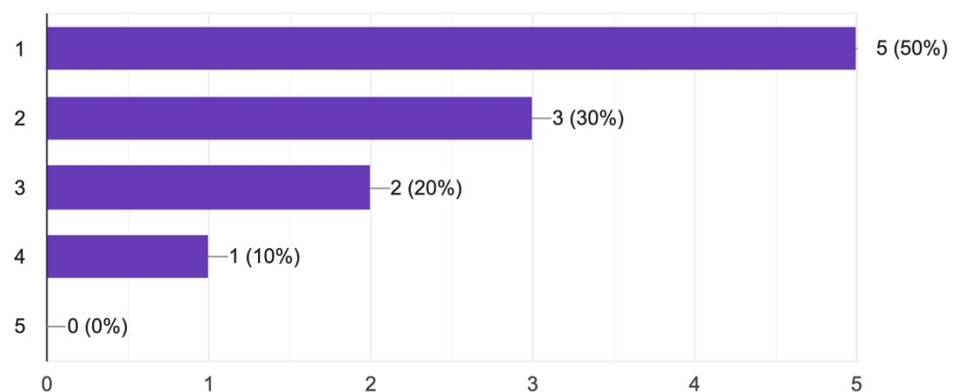
### How do you currently find and select a garage for vehicle servicing?

10 responses



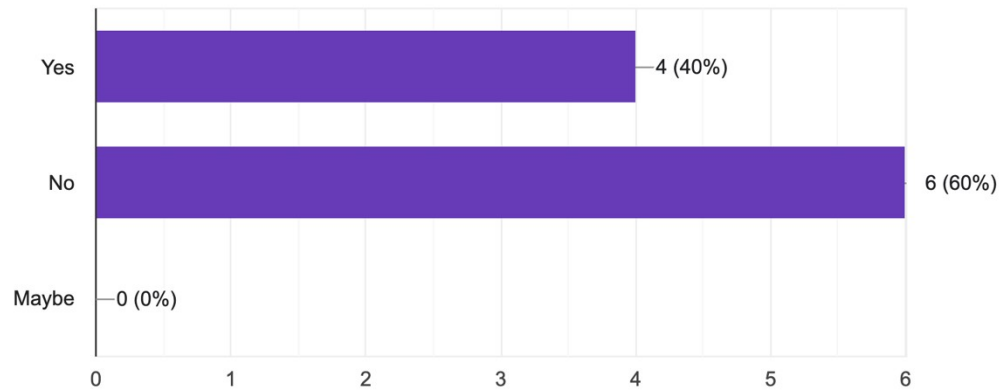
### How satisfied are you with your current methods for finding nearby garages for vehicles servicing? (Scale: 1-5, with 1 being very dissatisfied and 5 being very satisfied.)

10 responses



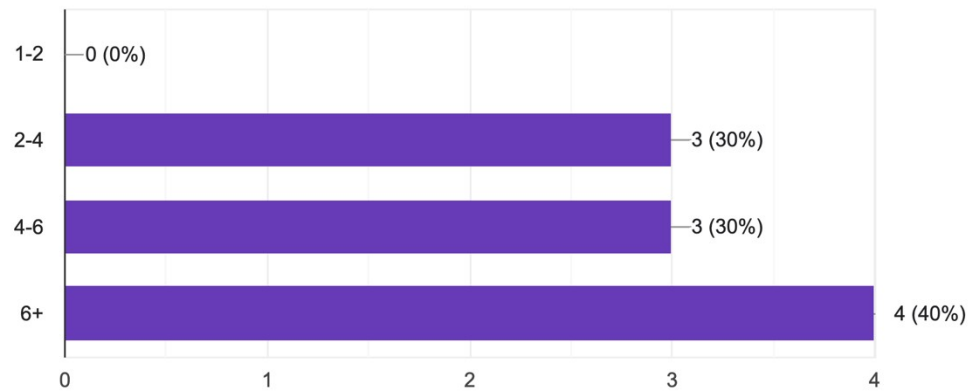
Have you ever used a mobile app related to automotive assistance?

10 responses



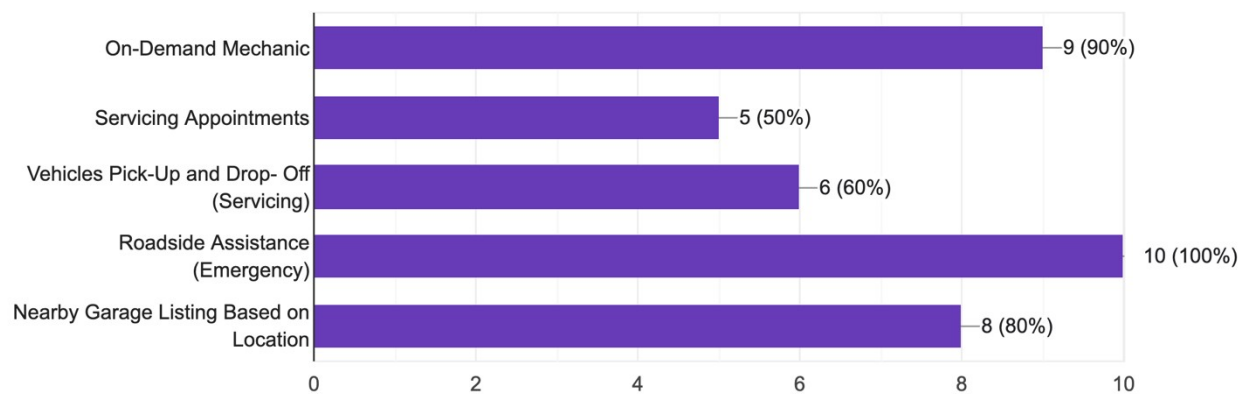
How often do you visit garages for vehicles servicing or repairs in a year?

10 responses



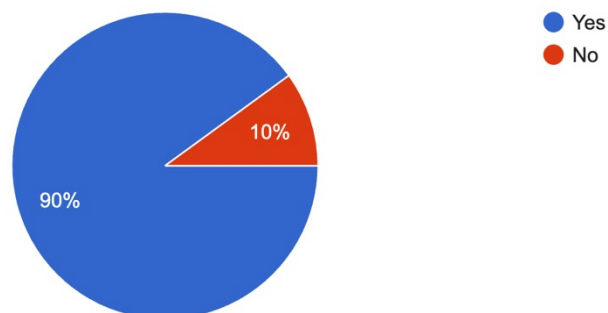
### Which features and services do you consider most valuable in a mobile app dedicated to automotive assistance?

10 responses



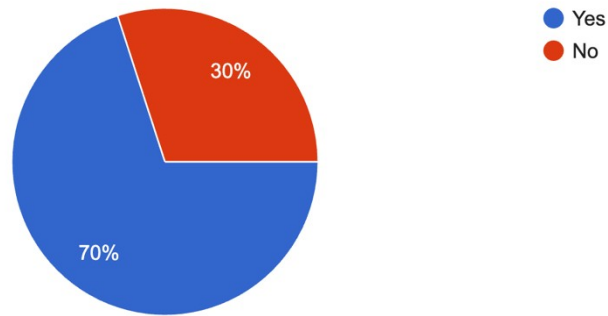
### Have you experienced vehicle emergencies or breakdowns in the past year?

10 responses



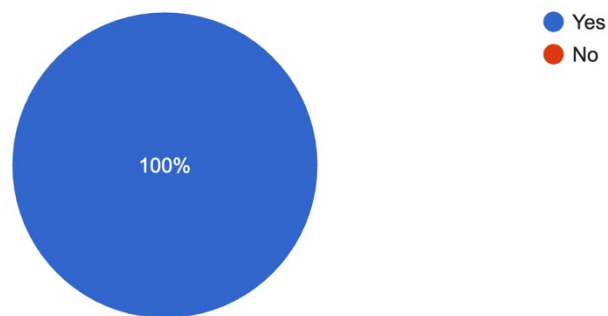
Would you be willing to pay for on-demand mechanic services through a mobile app?

10 responses



Do you use a smartphone for various tasks, including those related to your vehicle or automotive needs?

10 responses



## Complete Source Code

Main.dart



```
import 'package:crud_based/features/app/splash_screen/splash_screen.dart';
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: const FirebaseOptions(
      apiKey: "",
      authDomain: "",
      projectId: "",
      storageBucket: "",
      messagingSenderId: "",
      appId: "",
    ),
  );

  // Initializing Firestore
  FirebaseFirestore.instance.settings = Settings(persistenceEnabled: true);

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Firebase',
      home: SplashScreen(child: LoginPage()),
    );
  }
}
```

### Search\_places\_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_google_places/flutter_google_places.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:google_maps_webservice/places.dart';
import 'package:google_api_headers/google_api_headers.dart';

class SearchPlacesScreen extends StatefulWidget {
  const SearchPlacesScreen({super.key});

  @override
  State<SearchPlacesScreen> createState() => _SearchPlacesScreenState();
}

const kGoogleApiKey = '';
final homeScaffoldKey = GlobalKey<ScaffoldState>();

class _SearchPlacesScreenState extends State<SearchPlacesScreen> {
  static const CameraPosition initialCameraPosition =
    CameraPosition(target: LatLng(27.717245, 85.323959), zoom: 14.0);

  Set<Marker> markersList = {};

  late GoogleMapController googleMapController;

  final Mode _mode = Mode.overlay;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: homeScaffoldKey,
      appBar: AppBar(
        title: const Text("Google Search Places"),
      ),
      body: Stack(
        children: [
          GoogleMap(
            initialCameraPosition: initialCameraPosition,
            markers: markersList,
            mapType: MapType.normal,
            onMapCreated: (GoogleMapController controller) {
              googleMapController = controller;
            },
          ),
        ],
      ),
    );
  }
}
```

```
    ),  
    ElevatedButton(  
      onPressed: () => _handlePressButton(context),  
      child: const Text("Search Places/Garages"),  
    ),  
  ],  
),  
);  
}  
  
Future<void> _handlePressButton(BuildContext context) async {  
  Prediction? p = await PlacesAutocomplete.show(  
    context: context,  
    apiKey: kGoogleApiKey,  
    onError: (response) => onError(response, context),  
    mode: _mode,  
    language: 'en',  
    strictbounds: false,  
    types: ["car_repair"], //garages  
    decoration: InputDecoration(  
      hintText: 'Search',  
      focusedBorder: OutlineInputBorder(  
        borderRadius: BorderRadius.circular(20),  
        borderSide: BorderSide(color: Colors.white),  
      ),  
    ),  
    components: [Component(Component.country, "np")],  
  );  
  
  if (p != null) {  
    displayPrediction(p, homeScaffoldKey.currentState);  
  }  
}  
  
void onError(PlacesAutocompleteResponse response, BuildContext context) {  
  ScaffoldMessenger.of(context).showSnackBar(  
    SnackBar(  
      content: Text(response.errorMessage ?? 'Error occurred'),  
    ),  
  );  
}  
  
Future<void> displayPrediction(  
  Prediction p, ScaffoldState? currentState) async {
```

```
GoogleMapsPlaces places = GoogleMapsPlaces(
  apiKey: kGoogleApiKey,
  apiHeaders: await const GoogleApiHeaders().getHeaders(),
);

PlacesDetailsResponse detail = await places.getDetailsByPlaceId(p.placeId!);

final lat = detail.result.geometry!.location.lat;
final lng = detail.result.geometry!.location.lng;

markersList.clear();
markersList.add(
  Marker(
    markerId: const MarkerId("0"),
    position: LatLng(lat, lng),
    infoWindow: InfoWindow(title: detail.result.name),
  ),
);

setState(() {});

googleMapController
  .animateCamera(CameraUpdate.newLatLngZoom(LatLng(lat, lng), 14.0));
}
```

### Current\_location\_screen.dart

```
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:crud_based/googlemaps/search_places_screen.dart';
import 'package:crud_based/features/user_auth/presentation/pages/dashboard.dart';

class CurrentLocationScreen extends StatefulWidget {
  const CurrentLocationScreen({Key? key}) : super(key: key);

  @override
  State<CurrentLocationScreen> createState() => _CurrentLocationScreenState();
}

class _CurrentLocationScreenState extends State<CurrentLocationScreen> {
```

```
late GoogleMapController googleMapController;

static const CameraPosition initialCameraPosition =
    CameraPosition(target: LatLng(27.717245, 85.323959), zoom: 14);

Set<Marker> markers = {};

@override
void initState() {
    super.initState();
    _checkLocationPermission();
}

@override
void dispose() {
    super.dispose();
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text("Maps"),
            centerTitle: true,
            automaticallyImplyLeading: false,
        ),
        body: GoogleMap(
            initialCameraPosition: initialCameraPosition,
            markers: markers,
            zoomControlsEnabled: false,
            mapType: MapType.normal,
            onMapCreated: (GoogleMapController controller) {
                googleMapController = controller;
            },
        ),
        floatingActionButton: FloatingActionButton.extended(
            onPressed: () {
                // Navigating to the SearchPlacesScreen
                Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) => SearchPlacesScreen()),
                );
            },
            label: const Text("Search Places"),
        ),
    );
}
```

```
        icon: Icon(Icons.search),
      ),
    );
  }

Future<void> _checkLocationPermission() async {
  bool serviceEnabled;
  LocationPermission permission;

  try {
    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {
      _showLocationServiceDialog();
    } else {
      _getCurrentLocation();
    }
  } catch (e) {
    print(e.toString());
  }
}

Future<void> _showLocationServiceDialog() async {
  try {
    bool enableLocationService = await showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text("Enable Location Services"),
          content:
            Text("Please enable location services to use this feature."),
          actions: <Widget>[
            TextButton(
              child: Text("CANCEL"),
              onPressed: () {
                Navigator.of(context).pop(false);
              },
            ),
            TextButton(
              child: Text("ENABLE"),
              onPressed: () {
                Navigator.of(context).pop(true);
              },
            ),
          ],
        );
      },
    );
  }
}
```

```
1,
);
},
);

if (enableLocationService == true) {
  _getCurrentLocation();
} else {}
} catch (e) {
  print(e.toString());
}
}

Future<void> _getCurrentLocation() async {
  try {
    Position position = await _determinePosition();

    googleMapController.animateCamera(CameraUpdate.newCameraPosition(
      CameraPosition(
        target: LatLng(position.latitude, position.longitude),
        zoom: 14)));

    markers.clear();

    markers.add(Marker(
      markerId: const MarkerId('currentLocation'),
      position: LatLng(position.latitude, position.longitude)));

    if (mounted) {
      setState(() {});
    }
  } catch (e) {
    print(e.toString());
  }
}

Future<Position> _determinePosition() async {
  bool serviceEnabled;
  LocationPermission permission;

  try {
    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {
```

```
        throw Exception('Location services are disabled');
    }

    permission = await Geolocator.checkPermission();

    if (permission == LocationPermission.denied) {
        permission = await Geolocator.requestPermission();

        if (permission == LocationPermission.denied) {
            throw Exception("Location permission denied");
        }
    }

    if (permission == LocationPermission.deniedForever) {
        throw Exception(
            "Location permissions are permanently denied. Allow it from the settings.");
    }

    Position position = await Geolocator.getCurrentPosition();

    return position;
} catch (e) {
    throw e;
}
}
```

## Toast.dart

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

void showToast({required String message}) {
    Fluttertoast.showToast(
        msg: message,
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        timeInSecForIosWeb: 1,
        backgroundColor: Colors.blue,
        textColor: Colors.white,
        fontSize: 16.0);
}
```



## Mechanic.dart

```
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:url_launcher/url_launcher.dart';
import 'package:cached_network_image/cached_network_image.dart';

class ContactsPage extends StatefulWidget {
  const ContactsPage({Key? key}) : super(key: key);

  @override
  State<ContactsPage> createState() => _ContactsPageState();
}

class _ContactsPageState extends State<ContactsPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Contacts Details'),
      ),
      backgroundColor: Colors.blue,
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: StreamBuilder<QuerySnapshot>(
            stream:
              FirebaseFirestore.instance.collection('contacts').snapshots(),
            builder: (context, snapshot) {
              try {
                if (!snapshot.hasData) {
                  return CircularProgressIndicator();
                }
              }
            }
          )
        )
      )
    );
  }
}
```

```
var documents = snapshot.data!.docs;

return Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    for (int i = 0; i < documents.length; i++)
      ContactCard(
        contactNumber: documents[i]['number'],
        contactName: documents[i]['name'],
        contactAddress: documents[i]['address'],
        imageStoragePath: documents[i]['image_url'],
      ),
  ],
);
} catch (e) {
  print('Error fetching data from firestore:$e');
  return Text('Error Fetching data. Please try again');
}
},
),
),
),
);
}
}

class ContactCard extends StatelessWidget {
  final String contactNumber;
  final String contactName;
  final String contactAddress;
  final String imageStoragePath;

  const ContactCard({
    Key? key,
    required this.contactNumber,
    required this.contactName,
    required this.contactAddress,
    required this.imageStoragePath,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 2.0,
```

```
margin: const EdgeInsets.symmetric(vertical: 8.0),
child: InkWell(
  onTap: () {
    _launchPhoneApp(contactNumber);
  },
  child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          contactName,
          style: Theme.of(context).textTheme.headline6,
        ),
        SizedBox(height: 8.0),
        FutureBuilder<Uint8List?>(
          future: _downloadImage(imageStoragePath),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return CircularProgressIndicator();
            } else if (snapshot.hasError) {
              return Icon(Icons.error);
            } else if (!snapshot.hasData) {
              return Container();
            } else {
              return Image.memory(
                snapshot.data!,
                height: 100.0,
                width: 100.0,
                fit: BoxFit.cover,
              );
            }
          },
        ),
        SizedBox(height: 8.0),
        Text(
          'Contact Number: $contactNumber',
          style: TextStyle(fontWeight: FontWeight.bold),
        ),
        SizedBox(height: 8.0),
        Text('Address: $contactAddress'),
      ],
    ),
  ),
),
```

```
    ),  
    );  
}  
  
Future<Uint8List?> _downloadImage(String imageStoragePath) async {  
  try {  
    Reference storageReference =  
      FirebaseStorage.instance.ref().child(imageStoragePath);  
    final Uint8List? data = await storageReference.getData();  
    return data;  
  } catch (e) {  
    print('Error downloading image: $e');  
    return null;  
  }  
}  
  
void _launchPhoneApp(String phoneNumber) async {  
  try {  
    String url = 'tel:$phoneNumber';  
    if (await canLaunch(url)) {  
      await launch(url);  
    } else {  
      print('Could not Launch $url');  
    }  
  } catch (e) {  
    print('Error launching phone app: $e');  
  }  
}  
}
```

## Garage.dart

```
import 'package:flutter/material.dart';  
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:url_launcher/url_launcher.dart';  
  
class Garage extends StatefulWidget {  
  const Garage({Key? key}) : super(key: key);  
  
  @override  
  State<Garage> createState() => _GarageState();  
}
```

```
class _GarageState extends State<Garage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Garage Details'),
      ),
      backgroundColor: Colors.blue,
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: StreamBuilder<QuerySnapshot>(
            stream: FirebaseFirestore.instance.collection('garage').snapshots(),
            builder: (context, snapshot) {
              if (!snapshot.hasData) {
                return CircularProgressIndicator();
              }

              var documents = snapshot.data!.docs;

              return Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  for (int i = 0; i < documents.length; i++)
                    ContactCard(
                      garageNumber: i + 1,
                      documentId: documents[i].id,
                      contactNumber: documents[i]['number'],
                    ),
                ],
              );
            },
          ),
        ),
      );
  }
}

class ContactCard extends StatefulWidget {
  final int garageNumber;
  final String documentId;
  final String contactNumber;
```

```
const ContactCard({  
  Key? key,  
  required this.garageNumber,  
  required this.documentId,  
  required this.contactNumber,  
}) : super(key: key);  
  
@override  
_ContactCardState createState() => _ContactCardState();  
}  
  
class _ContactCardState extends State<ContactCard> {  
  TextEditingController _garageNameController = TextEditingController();  
  TextEditingController _ownerNameController = TextEditingController();  
  TextEditingController _locationController = TextEditingController();  
  TextEditingController _contactNumberController = TextEditingController();  
  
  @override  
  void initState() {  
    super.initState();  
  
    _fetchAndPopulateGarageData();  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Card(  
      elevation: 2.0,  
      margin: const EdgeInsets.symmetric(vertical: 8.0),  
      child: InkWell(  
        onTap: () {  
          _launchPhoneApp(widget.contactNumber);  
        },  
        child: Padding(  
          padding: const EdgeInsets.all(16.0),  
          child: Column(  
            crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
              Text(  
                'Garage ${widget.garageNumber}',  
                style: Theme.of(context).textTheme.headline6,  
              ),  
              SizedBox(height: 8.0),  
            ],  
          ),  
        ),  
      ),  
    );  
  }  
}
```

```
    TextField(
      controller: _locationController,
      decoration: InputDecoration(labelText: 'Location'),
      enabled: false,
    ),
    SizedBox(height: 8.0),
    TextField(
      controller: _garageNameController,
      decoration: InputDecoration(labelText: 'Garage Name'),
      enabled: false,
    ),
    SizedBox(height: 8.0),
    TextField(
      controller: _contactNumberController,
      decoration: InputDecoration(labelText: 'Contact Number'),
      keyboardType: TextInputType.phone,
      enabled: false,
    ),
    SizedBox(height: 8.0),
    TextField(
      controller: _ownerNameController,
      decoration: InputDecoration(labelText: 'Owner Name'),
      enabled: false,
    ),
  ],
),
),
),
);
}

void _fetchAndPopulateGarageData() async {
  try {
    DocumentSnapshot garageDoc = await FirebaseFirestore.instance
      .collection('garage')
      .doc(widget.documentId)
      .get();

    if (garageDoc.exists) {
      setState(() {
        _locationController.text = garageDoc['location'];
        _garageNameController.text = garageDoc['name'];
        _contactNumberController.text = garageDoc['number'];
        _ownerNameController.text = garageDoc['owner'];
      });
    }
  } catch (e) {
    // Handle error
  }
}
```

```
});  
}  
} catch (e) {  
  print("Error fetching garage data: $e");  
}  
}  
  
void _launchPhoneApp(String phoneNumber) async {  
  String url = 'tel:$phoneNumber';  
  if (await canLaunch(url)) {  
    await launch(url);  
  } else {  
    print('Could not Launch $url');  
  }  
}  
}
```

## Manager.dart

```
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';  
import 'package:flutter/material.dart';  
import 'package:carousel_slider/carousel_slider.dart';  
import 'dart:typed_data';  
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:firebase_storage/firebase_storage.dart';  
import 'package:url_launcher/url_launcher.dart';  
  
class ManagerPage extends StatelessWidget {  
  const ManagerPage({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Manager Dashboard'),  
        actions: [  
          IconButton(  
            icon: Icon(Icons.arrow_back),  
            onPressed: () {  
              Navigator.pushReplacement(  
                context,  
                MaterialPageRoute(builder: (context) => LoginPage()),  
              );  
            }  
          )  
        ],  
      ),  
    );  
  }  
}
```



```
    );  
  },  
),  
],  
),  
body: SingleChildScrollView(  
  child: Padding(  
    padding: const EdgeInsets.all(20.0),  
    child: Column(  
      crossAxisAlignment: CrossAxisAlignment.center,  
      children: [  
        Text(  
          'Upcoming Requests',  
          style: TextStyle(  
            fontSize: 28,  
            fontWeight: FontWeight.bold,  
          ),  
        ),  
        SizedBox(height: 20),  
  
        // Carousel Slider Section for Requests  
        RequestCarousel(),  
        SizedBox(height: 20),  
        Text(  
          'Available Mechanics',  
          style: TextStyle(  
            fontSize: 28,  
            fontWeight: FontWeight.bold,  
          ),  
        ),  
      ],  
    ),  
  ),  
),  
),  
);  
}  
}  
  
class ContactCarousel extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {
```

```
return StreamBuilder<QuerySnapshot>(  
  stream: FirebaseFirestore.instance.collection('contacts').snapshots(),  
  builder: (context, snapshot) {  
    try {  
      if (!snapshot.hasData) {  
        return CircularProgressIndicator();  
      }  
  
      var documents = snapshot.data!.docs;  
  
      return CarouselSlider(  
        options: CarouselOptions(  
          height: 300.0,  
          enlargeCenterPage: true,  
          autoPlay: true,  
          aspectRatio: 16 / 10,  
        ),  
        items: documents.map((document) {  
          return Builder(  
            builder: (BuildContext context) {  
              return ContactCard(  
                contactNumber: document['number'],  
                contactName: document['name'],  
                contactAddress: document['address'],  
                imageStoragePath: document['image_url'],  
              );  
            },  
          );  
        }).toList(),  
      );  
    } catch (e) {  
      print('Error fetching data from firestore:$e');  
      return Text('Error Fetching data. Please try again');  
    }  
  },  
);  
}  
}  
  
class ContactCard extends StatelessWidget {  
  final String contactNumber;  
  final String contactName;  
  final String contactAddress;  
  final String imageStoragePath;
```

```
const ContactCard({  
  Key? key,  
  required this.contactNumber,  
  required this.contactName,  
  required this.contactAddress,  
  required this.imageStoragePath,  
}) : super(key: key);  
  
@override  
Widget build(BuildContext context) {  
  return Card(  
    elevation: 2.0,  
    margin: const EdgeInsets.symmetric(vertical: 8.0),  
    child: InkWell(  
      onTap: () {  
        _launchPhoneApp(contactNumber);  
      },  
      child: Padding(  
        padding: const EdgeInsets.all(16.0),  
        child: Column(  
          crossAxisAlignment: CrossAxisAlignment.start,  
          children: [  
            Text(  
              contactName,  
              style: Theme.of(context).textTheme.headline6,  
            ),  
            SizedBox(height: 8.0),  
            FutureBuilder<Uint8List?>(  
              future: _downloadImage(imageStoragePath),  
              builder: (context, snapshot) {  
                if (snapshot.connectionState == ConnectionState.waiting) {  
                  return CircularProgressIndicator();  
                } else if (snapshot.hasError) {  
                  return Icon(Icons.error);  
                } else if (!snapshot.hasData) {  
                  return Container(); //  
                } else {  
                  return Image.memory(  
                    snapshot.data!,  
                    height: 100.0,  
                    width: 100.0,  
                    fit: BoxFit.cover,  
                  );  
                }  
              },  
            ),  
          ],  
        ),  
      ),  
    ),  
  );  
}
```

```
    }  
  },  
),  
  SizedBox(height: 8.0),  
  Text(  
    'Contact Number: $contactNumber',  
    style: TextStyle(fontWeight: FontWeight.bold),  
  ),  
  SizedBox(height: 8.0),  
  Text('Address: $contactAddress'),  
],  
),  
),  
),  
);  
}  
  
Future<Uint8List?> _downloadImage(String imageStoragePath) async {  
  try {  
    Reference storageReference =  
      FirebaseStorage.instance.ref().child(imageStoragePath);  
    final Uint8List? data = await storageReference.getData();  
    return data;  
  } catch (e) {  
    print('Error downloading image: $e');  
    return null;  
  }  
}  
  
class RequestCarousel extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return StreamBuilder<QuerySnapshot>(  
      stream: FirebaseFirestore.instance.collection('requests').snapshots(),  
      builder: (context, snapshot) {  
        try {  
          if (!snapshot.hasData) {  
            return CircularProgressIndicator();  
          }  
  
          var documents = snapshot.data!.docs;  
  
          return CarouselSlider(  

```

```
options: CarouselOptions(
  height: 300.0,
  enlargeCenterPage: true,
  autoPlay: true,
  aspectRatio: 16 / 10,
),
items: documents.map((document) {
  return Builder(
    builder: (BuildContext context) {
      return RequestCard(
        name: document['name'],
        number: document['number'],
        serviceType: document['service_type'],
        date: document['date'],
      );
    },
  );
}).toList(),
);
} catch (e) {
  print('Error fetching data from firestore:$e');
  return Text('Error Fetching data. Please try again!');
}
},
);
}
}

class RequestCard extends StatelessWidget {
  final String name;
  final String number;
  final String serviceType;
  final String date;

  const RequestCard({
    Key? key,
    required this.name,
    required this.number,
    required this.serviceType,
    required this.date,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
```

```
return Card(  
  elevation: 2.0,  
  margin: const EdgeInsets.symmetric(vertical: 8.0),  
  child: Padding(  
    padding: const EdgeInsets.all(16.0),  
    child: Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: [  
        Text(  
          'Request Details',  
          style: Theme.of(context).textTheme.headline6,  
        ),  
        SizedBox(height: 8.0),  
        Text(  
          'Name: $name',  
          style: TextStyle(fontWeight: FontWeight.bold),  
        ),  
        SizedBox(height: 8.0),  
        Text('Number: $number'),  
        SizedBox(height: 8.0),  
        Text('Service Type: $serviceType'),  
        SizedBox(height: 8.0),  
        Text('Date: $date'),  
      ],  
    ),  
  ),  
);  
}  
}  
  
void _launchPhoneApp(String phoneNumber) async {  
  try {  
    String url = 'tel:$phoneNumber';  
    if (await canLaunch(url)) {  
      await launch(url);  
    } else {  
      print('Could not Launch $url');  
    }  
  } catch (e) {  
    print('Error launching phone app: $e');  
  }  
}
```

### Form\_container\_widgets.dart

```
import 'package:flutter/material.dart';

class FormContainerWidget extends StatefulWidget {
  final TextEditingController? controller;
  final Key? fieldKey;
  final bool? isPasswordField;
  final String? hintText;
  final String? labelText;
  final String? helperText;
  final FormFieldSetter<String>? onSaved;
  final FormFieldValidator<String>? validator;
  final ValueChanged<String>? onFieldSubmitted;
  final TextInputType? inputType;
  final Null Function(dynamic password) onChanged;

  const FormContainerWidget({
    this.controller,
    this.isPasswordField,
    this.fieldKey,
    this.hintText,
    this.labelText,
    this.helperText,
    this.onSaved,
    this.validator,
    this.onFieldSubmitted,
    this.inputType,
    required this.onChanged,
  });

  @override
  _FormContainerWidgetState createState() => new _FormContainerWidgetState();
}

class _FormContainerWidgetState extends State<FormContainerWidget> {
  bool _obscureText = true;

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      decoration: BoxDecoration(
```

```
color: Colors.grey.withOpacity(.35),
borderRadius: BorderRadius.circular(10),
),
child: new TextFormField(
  style: TextStyle(color: Colors.black),
  controller: widget.controller,
  keyboardType: widget.inputType,
  key: widget.fieldKey,
  obscureText: widget.isPasswordField == true ? _obscureText : false,
  onSaved: widget.onSaved,
  validator: widget.validator,
  onFieldSubmitted: widget.onFieldSubmitted,
  onChanged: widget.onChanged, // Add this line
  decoration: new InputDecoration(
    border: InputBorder.none,
    filled: true,
    hintText: widget.hintText,
    hintStyle: TextStyle(color: Colors.black45),
    suffixIcon: new GestureDetector(
      onTap: () {
        setState(() {
          _obscureText = !_obscureText;
        });
      },
      child: widget.isPasswordField == true
        ? Icon(
            _obscureText ? Icons.visibility_off : Icons.visibility,
            color: _obscureText == false ? Colors.blue : Colors.grey,
          )
        : Text(""),
    ),
  ),
),
);
}
```

### Sign\_up\_page.dart

```
import 'package:crud_based/features/user_auth/firebase_auth_implementation/
firebase_auth_services.dart';
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';
```



```
import 'package:crud_based/features/user_auth/presentation/widgets/form_container_widget.dart';
import 'package:crud_based/global/common/toast.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class SignUpPage extends StatefulWidget {
  const SignUpPage({Key? key}) : super(key: key);

  @override
  _SignUpPageState createState() => _SignUpPageState();
}

class _SignUpPageState extends State<SignUpPage> {
  bool isSigningUp = false;
  final FirebaseAuthService _auth = FirebaseAuthService();
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();
  PasswordStrength _passwordStrength = PasswordStrength.Weak;

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("SignUp"),
      ),
      body: Center(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 15),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                "SignUp",
                style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold),
              ),
              SizedBox(
                height: 30,
```

```
),  
FormContainerWidget(  
  controller: _emailController,  
  hintText: "Email",  
  isPasswordField: false,  
  onChanged: (password) {},  
),  
SizedBox(  
  height: 10,  
),  
FormContainerWidget(  
  controller: _passwordController,  
  hintText: "Password",  
  isPasswordField: true,  
  onChanged: (password) {  
    // Updating the password strength dynamically.  
    setState(() {  
      _passwordStrength = calculatePasswordStrength(password);  
    });  
  },  
),  
// Displaying the dynamic password strength indicator.  
PasswordStrengthIndicator(strength: _passwordStrength),  
SizedBox(  
  height: 30,  
),  
GestureDetector(  
  onTap: () {  
    _signUp();  
  },  
  child: Container(  
    width: double.infinity,  
    height: 45,  
    decoration: BoxDecoration(  
      color: Colors.blue,  
      borderRadius: BorderRadius.circular(10),  
    ),  
    child: Center(  
      child: isSigningUp  
        ? CircularProgressIndicator(  
          color: Colors.white,  
        )  
        : Text(  
          "Sign Up",
```

```

        style: TextStyle(
          color: Colors.white,
          fontWeight: FontWeight.bold,
        ),
      ),
    ),
  ),
  SizedBox(
    height: 20,
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text("Already have an Account?"),
      SizedBox(width: 5),
      GestureDetector(
        onTap: () {
          Navigator.pushAndRemoveUntil(
            context,
            MaterialPageRoute(builder: (context) => LoginPage()),
            (route) => false,
          );
        },
      ),
      child: Text(
        "Sign In",
        style: TextStyle(
          color: Colors.blue,
          fontWeight: FontWeight.bold,
        ),
      ),
    ],
  ),
],
),
],
),
),
);
}

void _signUp() async {
  setState(() {
    isSigningUp = true;
  });
}

```

```
});

String email = _emailController.text.trim();
String password = _passwordController.text.trim();

User? user = await _auth.signUpWithEmailAndPassword(email, password);

setState(() {
  isSigningUp = false;
});

if (user != null) {
  showToast(message: "User is successfully created");
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(
      builder: (context) => LoginPage(),
    ),
  );
} else {
  showToast(message: "Some error occurred");
}
}

PasswordStrength calculatePasswordStrength(String password) {
  if (password.length < 6) {
    return PasswordStrength.Weak;
  } else if (password.length < 10) {
    return PasswordStrength.Moderate;
  } else {
    return PasswordStrength.Strong;
  }
}

enum PasswordStrength { Weak, Moderate, Strong }

class PasswordStrengthIndicator extends StatelessWidget {
  final PasswordStrength strength;

  const PasswordStrengthIndicator({Key? key, required this.strength})
    : super(key: key);

  @override
```

```
Widget build(BuildContext context) {  
  return Text(  
    'Password Strength: ${strength.toString()}',  
    style: TextStyle(  
      color: getColorForStrength(strength),  
    ),  
  );  
}  
  
Color getColorForStrength>PasswordStrength strength) {  
  switch (strength) {  
    case PasswordStrength.Weak:  
      return Colors.red;  
    case PasswordStrength.Moderate:  
      return Colors.orange;  
    case PasswordStrength.Strong:  
      return Colors.green;  
  }  
}
```

### Login\_page.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:crud_based/features/user_auth/firebase_auth_implementation/  
firebase_auth_services.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/dashboard.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/home_page.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/sign_up_page.dart';  
import 'package:crud_based/features/user_auth/presentation/widgets/form_container_widget.dart';  
import 'package:crud_based/future_scope/manager.dart';  
import 'package:crud_based/global/common/toast.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:provider/provider.dart';  
  
class LoginPage extends StatefulWidget {  
  const LoginPage({Key? key});  
  
  @override  
  _LoginPageState createState() => _LoginPageState();  
}
```

```
class _LoginPageState extends State<LoginPage> {  
  bool _isSigning = false;  
  final FirebaseAuthService _auth = FirebaseAuthService();  
  
  TextEditingController _emailController = TextEditingController();  
  TextEditingController _passwordController = TextEditingController();  
  
  String _userId = ""; // Storing the UID in the LoginPage  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Login"),  
      ),  
      body: Center(  
        child: Padding(  
          padding: const EdgeInsets.symmetric(horizontal: 15),  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.center,  
            children: [  
              Text(  
                "Login",  
                style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold),  
              ),  
              SizedBox(height: 30),  
              FormContainerWidget(  
                controller: _emailController,  
                hintText: "Email",  
                isPasswordField: false,  
                onChanged: (password) {},  
              ),  
              SizedBox(height: 10),  
              FormContainerWidget(  
                controller: _passwordController,  
                hintText: "Password",  
                isPasswordField: true,  
                onChanged: (password) {},  
              ),  
              SizedBox(height: 30),  
              GestureDetector(  
                onTap: () => _signIn(context),  
                child: Container(  

```

[illegible]

```
    ),  
    ],  
  ),  
  SizedBox(height: 20),  
  Text(""),  
  SizedBox(height: 20),  
  ElevatedButton(  
    onPressed: () {  
      // Navigating to the 'Manager' page when the button is clicked  
      Navigator.pushReplacement(  
        context,  
        MaterialPageRoute(  
          builder: (context) => ManagerPage(),  
        ),  
      );  
    },  
    style: ElevatedButton.styleFrom(  
      primary: Colors.green, // Setting the button color  
      shape: RoundedRectangleBorder(  
        borderRadius: BorderRadius.circular(10),  
      ),  
    ),  
    child: Text(  
      'Garage Owners',  
      style: TextStyle(  
        color: Colors.white,  
        fontWeight: FontWeight.bold,  
      ),  
    ),  
  ),  
  ],  
),  
),  
),  
);  
}  
  
void _signIn(BuildContext context) async {  
  setState(() {  
    _isSigning = true;  
  });  
  String email = _emailController.text.trim();  
  String password = _passwordController.text.trim();
```



```
User? user = await _auth.signInWithEmailAndPassword(email, password);

setState(() {
  _isSigning = false;
});

if (user != null) {
  print("User is successfully Signed In. UID: ${user.uid}");
  showToast(message: "User is successfully Signed In");

  //Checking if user data exists in firestore user collection
  bool isUserDataExists = await _checkUserDataExists(user.uid);

  if (!isUserDataExists) {
    // If user data doesn't exist, adding a new document to the 'user' collection
    await _addUserDataToFirestore(user.uid, email, password);
  }

  // Storing the UID in the LoginPage
  setState(() {
    _userUid = user.uid;
  });

  Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => DashboardPage()),
  );
} else {
  showToast(message: "Some error occurred");
}
}

Future<bool> _checkUserDataExists(String uid) async {
  try {
    // Checking if user data exists in Firestore 'user' collection
    DocumentSnapshot userDoc =
      await FirebaseFirestore.instance.collection('user').doc(uid).get();
    return userDoc.exists;
  } catch (e) {
    print("Error checking user data: $e");
    return false;
  }
}
```

```
Future<void> _addUserDataToFirestore(  
  String uid, String email, String password) async {  
  try {  
    // Adding a new document to Firestore 'user' collection  
    await FirebaseFirestore.instance.collection('user').doc(uid).set({  
      'email': email,  
      'password': password,  
    });  
  } catch (e) {  
    print("Error adding user data to Firestore: $e");  
  }  
}
```

## Home\_page.dart

```
import 'dart:ffi';  
import 'dart:io';  
  
import 'package:crud_based/features/user_auth/presentation/pages/dashboard.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:firebase_storage/firebase_storage.dart';  
import 'package:flutter/material.dart';  
import 'package:image_picker/image_picker.dart';  
import 'package:url_launcher/url_launcher.dart';  
  
class HomePage extends StatefulWidget {  
  const HomePage({Key? key}) : super(key: key);  
  
  @override  
  State<HomePage> createState() => _HomePageState();  
}  
  
class _HomePageState extends State<HomePage> {  
  final TextEditingController _emailController = TextEditingController();  
  final TextEditingController _passwordController = TextEditingController();  
  final TextEditingController _phoneNumberController = TextEditingController();  
  final TextEditingController _userNameController = TextEditingController();  
  
  @override
```

```
void initState() {  
  super.initState();  
  // Fetching and populating user data when the HomePage is created  
  _fetchAndPopulateUserData();  
}  
  
@override  
Widget build(BuildContext context) {  
  _fetchAndPopulateUserData();  
  return Scaffold(  
    backgroundColor: Colors.blue,  
    appBar: AppBar(  
      automaticallyImplyLeading: false,  
      title: Text(" Your Profile"),  
      actions: [  
        IconButton(  
          onPressed: () {  
            _signOut(context);  
          },  
          icon: Icon(Icons.logout),  
        ),  
      ],  
    ),  
    body: SingleChildScrollView(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
          //Circle Avatar to display the picture  
  
          Center(  
            child: Text(  
              "Update Your Profile!",  
              style: TextStyle(fontWeight: FontWeight.bold, fontSize: 19),  
            ),  
          ),  
          SizedBox(height: 20),  
          Padding(  
            padding: const EdgeInsets.all(8.0),  
            child: TextField(  
              controller: _emailController,  
              decoration: InputDecoration(  
                labelText: 'Email',  
                border: OutlineInputBorder(),  
              ),  
            ),  
          ),  
        ],  
      ),  
    ),  
  );  
}
```

```
    ),  
  ),  
  Padding(  
    padding: const EdgeInsets.all(8.0),  
    child: TextField(  
      controller: _passwordController,  
      obscureText: true,  
      decoration: InputDecoration(  
        labelText: 'Password',  
        border: OutlineInputBorder(),  
      ),  
    ),  
  ),  
  ),  
  Padding(  
    padding: const EdgeInsets.all(8.0),  
    child: TextField(  
      controller: _phoneNumberController,  
      decoration: InputDecoration(  
        labelText: 'Phone Number',  
        border: OutlineInputBorder(),  
      ),  
    ),  
  ),  
  ),  
  Padding(  
    padding: const EdgeInsets.all(8.0),  
    child: TextField(  
      controller: _usernameController,  
      decoration: InputDecoration(  
        labelText: 'Username',  
        border: OutlineInputBorder(),  
      ),  
    ),  
  ),  
  ),  
  ElevatedButton(  
    onPressed: () {  
      _updateUserData();  
    },  
    child: Text('Update'),  
  ),  
  SizedBox(height: 20),  
  Text(  
    'You may set up your phone number and username during the initial setup. '  
    'For updating your profile info from the second time and onwards, please click the button  
below',
```

```
        textAlign: TextAlign.center,
        style: TextStyle(fontSize: 16),
      ),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: () {
          _launchEmailApp();
        },
        child: Text('Request Profile Update'),
      ),
    ],
  ),
);
}

void _fetchAndPopulateUserData() async {
  try {
    // Getting the current user
    User? user = FirebaseAuth.instance.currentUser;

    if (user != null) {
      // Retrieving user data from Firestore
      DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection('user')
        .doc(user.uid)
        .get();

      // Updating the controllers with the retrieved data
      if (userDoc.exists) {
        setState(() {
          _emailController.text = userDoc['email'];
          _passwordController.text = userDoc['password'];
          _phoneNumberController.text = userDoc['phoneNumber'];
          _userNameController.text = userDoc['userName'];
        });
      }
    }
  } catch (e) {
    print("Error fetching user data: $e");
  }
}

void _updateUserData() async {
```

```
try {
  User? user = FirebaseAuth.instance.currentUser;

  if (user != null) {
    String uid = user.uid;

    // Creating a map with the updated data
    Map<String, dynamic> updatedData = {
      'email': _emailController.text,
      'password': _passwordController.text,
      'phoneNumber': _phoneNumberController.text,
      'userName': _userNameController.text,
    };

    // Updating the user data in Firestore
    await FirebaseFirestore.instance
      .collection('user')
      .doc(uid)
      .update(updatedData);

    _showSnackBar('Data Updated Successfully!');

    // Redirecting back to the predecessor page
    Navigator.pop(context);
  }
} catch (e) {
  print('Error updating user data: $e');
  _showSnackBar('Error updating user data');
}

void _signOut(BuildContext context) {
  Navigator.pushAndRemoveUntil(
    context,
    MaterialPageRoute(builder: (context) => DashboardPage()),
    (route) => false,
  );
}

void _showSnackBar(String message) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(message),
      duration: Duration(seconds: 2),
    ),
  );
}
```

```
    ),  
    );  
}  
  
void _launchEmailApp() async {  
  const email = 'prabin.timilsina14@gmail.com';  
  final Uri params = Uri(  
    scheme: 'mailto',  
    path: email,  
  );  
  
  final Uri emailLaunchUri = Uri(  
    scheme: params.scheme,  
    path: params.path,  
  );  
  
  if (await canLaunch(emailLaunchUri.toString())) {  
    await launch(emailLaunchUri.toString());  
  } else {  
    _showSnackBar('Could not launch email app');  
  }  
}  
}
```

## Dashboard.dart

```
import 'package:crud_based/features/user_auth/presentation/pages/complaint.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/home_page.dart';  
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';  
import 'package:crud_based/garages/garage.dart';  
import 'package:crud_based/garages/mechanic.dart';  
import 'package:crud_based/googlemaps/current_location_screen.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:carousel_slider/carousel_slider.dart';  
import 'dart:typed_data';  
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:firebase_storage/firebase_storage.dart';  
import 'package:url_launcher/url_launcher.dart';  
  
class DashboardPage extends StatelessWidget {  
  @override
```

```
Widget build(BuildContext context) {  
  double screenHeight = MediaQuery.of(context).size.height;  
  double screenWidth = MediaQuery.of(context).size.width;  
  
  return Scaffold(  
    appBar: AppBar(  
      title: Text("Dashboard"),  
      actions: [  
        IconButton(  
          icon: Icon(Icons.logout),  
          onPressed: () {  
            _signOut(context);  
          },  
        ),  
      ],  
    ),  
    body: Container(  
      color: Colors.blue,  
      child: Stack(  
        children: [  
          // Icon Buttons Section  
          Positioned(  
            top: screenHeight * 0.45,  
            left: screenWidth * 0.04,  
            child: _buildIconWithText(  
              context,  
              Icons.account_circle,  
              "Profile Update",  
              () {  
                Navigator.push(  
                  context,  
                  MaterialPageRoute(builder: (context) => HomePage()),  
                );  
              },  
            ),  
          ),  
          Positioned(  
            top: screenHeight * 0.62,  
            left: screenWidth * 0.06,  
            child: _buildIconWithText(  
              context,  
              Icons.location_on,  
              "Find Garages",  
              () {
```



```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => CurrentLocationScreen(),
  ),
);
},
),
),
Positioned(
  top: screenHeight * 0.45,
  left: screenWidth * 0.48,
  child: _buildIconWithText(
    context,
    Icons.calendar_today,
    "Make Appointments",
  ) {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => Garage(),
      ),
    );
  },
),
),
Positioned(
  top: screenHeight * 0.62,
  left: screenWidth * 0.56,
  child: _buildIconWithText(
    context,
    Icons.phone,
    "Call Mechanic",
  ) {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => ContactsPage()),
    );
  },
),
),
Positioned(
  top: screenHeight * 0.74,
  left: screenWidth * 0.5 - 55,
```

```
child: _buildIconWithText(
  context,
  Icons.book,
  "Complaint",
) {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ComplaintPage(
        uid: "",
      )),
  );
},
),
),

// Carousel Slider Section
Positioned(
  top: screenHeight * 0.01,
  left: 0,
  right: 0,
  child: ContactCarousel(),
),
],
),
),
);
}

Widget _buildIconWithText(
  BuildContext context,
  IconData iconData,
  String title,
  VoidCallback onTap,
) {
  return Column(
    children: [
      GestureDetector(
        onTap: onTap,
        child: Icon(
          iconData,
          size: 50,
          color: Colors.white,
        ),
      ),
    ],
  );
}
```

```
    ),  
    SizedBox(height: 8),  
    Text(  
      title,  
      style: TextStyle(  
        fontSize: 20, color: Colors.white, fontWeight: FontWeight.bold),  
      ),  
    ],  
  );  
}  
  
void _signOut(BuildContext context) async {  
  await FirebaseAuth.instance.signOut();  
  Navigator.pushAndRemoveUntil(  
    context,  
    MaterialPageRoute(builder: (context) => LoginPage()),  
    (route) => false,  
  );  
  _showSnackBar(context, "Sign out successful");  
}  
  
void _showSnackBar(BuildContext context, String message) {  
  ScaffoldMessenger.of(context).showSnackBar(  
    SnackBar(  
      content: Text(message),  
      duration: Duration(seconds: 2),  
    ),  
  );  
}  
}  
  
class ContactCarousel extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return StreamBuilder<QuerySnapshot>(  
      stream: FirebaseFirestore.instance.collection('contacts').snapshots(),  
      builder: (context, snapshot) {  
        try {  
          if (!snapshot.hasData) {  
            return CircularProgressIndicator();  
          }  
  
          var documents = snapshot.data!.docs;
```

```
return CarouselSlider(  
  options: CarouselOptions(  
    height: 300.0,  
    enlargeCenterPage: true,  
    autoPlay: true,  
    aspectRatio: 16 / 10,  
  ),  
  items: documents.map((document) {  
    return Builder(  
      builder: (BuildContext context) {  
        return Container(  
          width: MediaQuery.of(context).size.width,  
          margin: EdgeInsets.symmetric(horizontal: 5.0),  
          decoration: BoxDecoration(  
            color: Colors.blue,  
          ),  
          child: ContactCard(  
            contactNumber: document['number'],  
            contactName: document['name'],  
            contactAddress: document['address'],  
            imageStoragePath: document['image_url'],  
          ),  
        );  
      },  
    );  
  }).toList(),  
);  
} catch (e) {  
  print('Error fetching data from firestore:$e');  
  return Text('Error Fetching data. Please try again!');  
}  
},  
);  
}  
}  
  
class ContactCard extends StatelessWidget {  
  final String contactNumber;  
  final String contactName;  
  final String contactAddress;  
  final String imageStoragePath;  
  
  const ContactCard({  
    Key? key,
```

```
required this.contactNumber,  
required this.contactName,  
required this.contactAddress,  
required this.imageStoragePath,  
}) : super(key: key);  
  
@override  
Widget build(BuildContext context) {  
  return Card(  
    elevation: 2.0,  
    margin: const EdgeInsets.symmetric(vertical: 8.0),  
    child: InkWell(  
      onTap: () {  
        _launchPhoneApp(contactNumber);  
      },  
      child: Padding(  
        padding: const EdgeInsets.all(16.0),  
        child: Column(  
          crossAxisAlignment: CrossAxisAlignment.start,  
          children: [  
            Text(  
              contactName,  
              style: Theme.of(context).textTheme.headline6,  
            ),  
            SizedBox(height: 8.0),  
            FutureBuilder<Uint8List?>(  
              future: _downloadImage(imageStoragePath),  
              builder: (context, snapshot) {  
                if (snapshot.connectionState == ConnectionState.waiting) {  
                  return CircularProgressIndicator();  
                } else if (snapshot.hasError) {  
                  return Icon(Icons.error);  
                } else if (!snapshot.hasData) {  
                  return Container();  
                } else {  
                  return Image.memory(  
                    snapshot.data!,  
                    height: 100.0,  
                    width: 100.0,  
                    fit: BoxFit.cover,  
                  );  
                }  
              },  
            ),  
          ],  
        ),  
      ),  
    ),  
  );  
}
```

```
        SizedBox(height: 8.0),
        Text(
          'Contact Number: $contactNumber',
          style: TextStyle(fontWeight: FontWeight.bold),
        ),
        SizedBox(height: 8.0),
        Text('Address: $contactAddress'),
      ],
    ),
  ),
);
}

Future<Uint8List?> _downloadImage(String imageStoragePath) async {
  try {
    Reference storageReference =
      FirebaseStorage.instance.ref().child(imageStoragePath);
    final Uint8List? data = await storageReference.getData();
    return data;
  } catch (e) {
    print('Error downloading image: $e');
    return null;
  }
}

void _launchPhoneApp(String phoneNumber) async {
  try {
    String url = 'tel:$phoneNumber';
    if (await canLaunch(url)) {
      await launch(url);
    } else {
      print('Could not Launch $url');
    }
  } catch (e) {
    print('Error launching phone app: $e');
  }
}
}
```

```
import 'package:flutter/material.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:image_picker/image_picker.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:intl/intl.dart';
import 'dart:io';

class ComplaintPage extends StatefulWidget {
  final String uid; // Passing the UID from the login page

  ComplaintPage({required this.uid});

  @override
  _ComplaintPageState createState() => _ComplaintPageState();
}

class _ComplaintPageState extends State<ComplaintPage> {
  final TextEditingController nameController = TextEditingController();
  final TextEditingController descriptionController = TextEditingController();
  File? _image;
  bool _isSubmitting = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("File a Complaint"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              ElevatedButton(
                onPressed: _isSubmitting ? null : () => _pickImage(),
                // Disabling the button if submitting is in progress
                child: _isSubmitting
                  ? CircularProgressIndicator()
                  : Text("Upload a Picture"),
              ),
              SizedBox(height: 16.0),
              _image != null

```

```
        ? Image.file(
            _image!,
            height: 100.0,
        )
        : Container(),
    SizedBox(height: 16.0),
    TextField(
        controller: nameController,
        decoration: InputDecoration(labelText: 'Garage/Mechanic Name'),
    ),
    SizedBox(height: 16.0),
    Text(
        'Date: ${DateFormat('yyyy-MM-dd').format(DateTime.now().toLocal())}',
        style: TextStyle(fontSize: 16.0),
    ),
    SizedBox(height: 16.0),
    TextField(
        controller: descriptionController,
        maxLines: 4,
        decoration: InputDecoration(labelText: 'Description'),
    ),
    SizedBox(height: 16.0),
    ElevatedButton(
        onPressed:
            _isSubmitting ? null : () => _submitComplaint(context),
        // Disabling the button if submitting is in progress
        child: _isSubmitting
            ? CircularProgressIndicator()
            : Text("Submit"),
    ),
  ],
),
),
);
}

void _pickImage() async {
  final picker = ImagePicker();
  final pickedFile = await picker.pickImage(source: ImageSource.gallery);

  setState(() {
    if (pickedFile != null) {
      _image = File(pickedFile.path);
    }
  });
}
```



```
    }  
  });  
}  
  
void _submitComplaint(BuildContext context) async {  
  // Set _isSubmitting to true when starting the submission  
  setState(() {  
    _isSubmitting = true;  
  });  
  String name = nameController.text;  
  String description = descriptionController.text;  
  String currentDate =  
    DateFormat('yyyy-MM-dd').format(DateTime.now().toLocal());  
  
  if (_image != null) {  
    // Uploading the image to Firebase Storage  
    String fileName = DateTime.now().millisecondsSinceEpoch.toString();  
    Reference storageReference =  
      FirebaseStorage.instance.ref().child('complaint_images/$fileName');  
    UploadTask uploadTask = storageReference.putFile(_image!);  
    await uploadTask.whenComplete(() async {  
      String imageUrl = await storageReference.getDownloadURL();  
  
      // Save data to Firestore  
      await FirebaseFirestore.instance.collection('complaint').add({  
        'uid': widget.uid,  
        'name': name,  
        'date': currentDate,  
        'description': description,  
        'image_url': imageUrl,  
      });  
  
      // using imageUrl as the URL to the uploaded image  
      print('Image URL: $imageUrl');  
    });  
  }  
  
  print('Garage/Mechanic Name: $name');  
  print('Date: $currentDate');  
  print('Description: $description');  
  
  // Set _isSubmitting back to false once submission is complete  
  setState(() {  
    _isSubmitting = false;  
  });  
}
```

```
});  
Navigator.pop(context);  
}  
}
```

### Firestore\_auth\_services.dart

```
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
  
import '../global/common/toast.dart';  
  
class FirebaseAuthService {  
  FirebaseAuth _auth = FirebaseAuth.instance;  
  
  Future<User?> signUpWithEmailAndPassword(  
    String email, String password) async {  
    try {  
      UserCredential credential = await _auth.createUserWithEmailAndPassword(  
        email: email, password: password);  
      return credential.user;  
    } on FirebaseAuthException catch (e) {  
      if (e.code == 'email-already-in-use') {  
        showToast(message: 'The email address is already in use.');      } else {  
        showToast(message: 'An error occurred: ${e.code}');      }  
    }  
    return null;  
  }  
  
  Future<User?> signInWithEmailAndPassword(  
    String email, String password) async {  
    try {  
      UserCredential credential = await _auth.signInWithEmailAndPassword(  
        email: email, password: password);  
      return credential.user;  
    } on FirebaseAuthException catch (e) {
```

```
    if (e.code == 'user-not-found' || e.code == 'wrong-password') {  
      showToast(message: 'Invalid email or password.');
```

### Splash\_screen.dart

```
import 'package:crud_based/features/user_auth/presentation/pages/login_page.dart';  
import 'package:flutter/material.dart';  
  
class SplashScreen extends StatefulWidget {  
  final Widget? child;  
  const SplashScreen({Key? key, this.child});  
  
  @override  
  State<SplashScreen> createState() => _SplashScreenState();  
}  
  
class _SplashScreenState extends State<SplashScreen> {  
  @override  
  void initState() {  
    Future.delayed(Duration(seconds: 3), () {  
      Navigator.pushAndRemoveUntil(  
        context,  
        MaterialPageRoute(builder: (context) => widget.child!),  
        (route) => false);  
    });  
  
    super.initState();  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: Center(  

```

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: [  
    Image.asset(  
      'assets/logo.png',  
      width: 200,  
      height: 200,  
    ),  
    SizedBox(height: 20),  
    Text(  
      'Auto-Care Connect',  
      style: TextStyle(color: Colors.blue, fontWeight: FontWeight.bold),  
    ),  
  ],  
,  
,  
,  
);  
}
```

### APP Evaluation Surveys (ALL of them being in the report -below is the sample)

How satisfied are you with the overall experience of using the app?

14 responses

