Saudi sign language (SSL)

CPIT 499 Final Report

By

Hadeel Bakhshwen	2006299
Jude Kawther	2009599
Almaha Abdulwahab	2009018
Shahad Alharthi	2105261

Supervised By

Dr. Anod Alhazmi

Department of Information Technology Faculty of Computing and Information Technology King Abdulaziz University

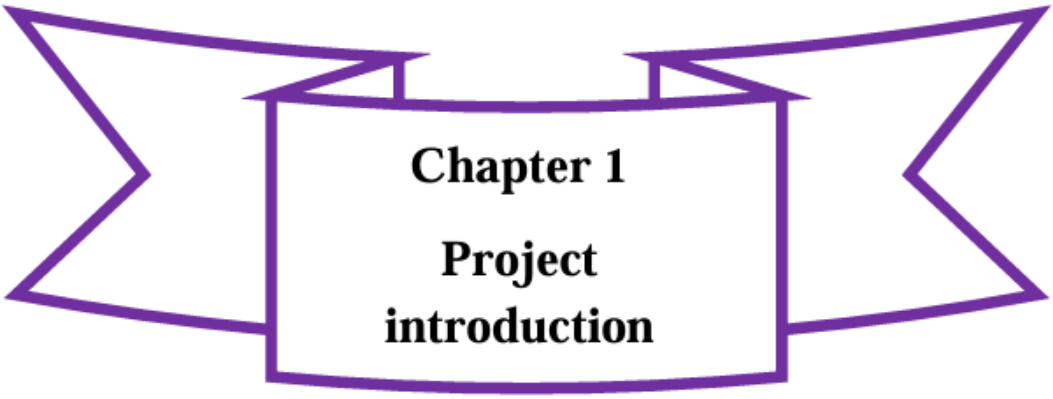
Jeddah – Saudi Arabia

[Fall 2025]

Contents

Table of Figures

Table of Tables



1.1 Introduction
1.2 Background
1.3 Problem Definition
1.4 Recommended Solution
1.5 Aims
1.6 Target Users
1.7 Objectives
1.8 Methodology
1.9 Project Plan
1.10 Report Organization
1.11 Conclusion

1.1 Introduction:

Communication challenges in the healthcare sector represent one of the most significant issues faced by individuals who are deaf or hard of hearing. These individuals often struggle to express their health needs or concerns about treatment due to the lack of effective communication methods. While medical disciplines offer various ways to provide adequate healthcare, there is a pressing need to expand these approaches to include technologies that bridge the communication gap between deaf patients and healthcare providers. This challenge highlights the intersection of different fields such as technology, public health, and digital media, creating a vast opportunity to develop innovative solutions that can address this gap.

Despite the fact that many modern technologies offer opportunities for digital interaction, the use of assistive technologies to enable communication between deaf individuals and healthcare providers in Saudi Arabia remains limited. In this context, the "Mueen" application stands as a significant step towards developing technological solutions that utilize Saudi Sign Language (SSL) to enhance communication experiences in medical clinics and reduce reliance on interpreters. This application leverages deep learning techniques and image processing to translate sign language gestures into real-time text and audio.

In this chapter, we discuss the problem faced by deaf individuals in medical clinics due to the lack of effective communication methods and present the solution offered by the "Mueen" application. We will also describe the methodologies that will be followed to develop this solution in line with the needs of both patients and healthcare providers, along with the benefits that the medical community will gain from this modern technology.

1.2 Background:

Sign language is considered one of the most important means of communication for individuals with hearing impairments to interact with others. In Saudi Arabia, Saudi Sign Language (SSL) is the primary method for interaction between deaf individuals and society in general. With the advancement of the digital world, new technologies have emerged that can significantly contribute to improving the interaction of this group with their surroundings, especially in areas that require effective communication, such as healthcare.

In medical clinics, doctors and healthcare practitioners face significant challenges in communicating with deaf patients, as specialized sign language interpreters are not always available. This leads to difficulties in conveying accurate information, especially in medical environments that require high precision. Despite the development of some tools and technologies aimed at improving this communication, many of the proposed solutions do not support Saudi Sign Language or are ineffective in healthcare settings.

At a time when Saudi Vision 2030 is witnessing a significant digital transformation in the healthcare sector, the need for innovative solutions to provide comprehensive and integrated healthcare for individuals with hearing disabilities has become essential. This requires the development of smart applications that help bridge the gap between deaf patients and doctors using assistive technologies, such as translating sign language gestures into text and voice.

The "Mu'een" app comes as a pivotal step in improving the healthcare experience for deaf individuals in medical clinics. Through this app, deaf patients can directly interact with healthcare providers using Saudi Sign Language, which is accurately converted into text and voice in real-time. This helps doctors understand the patients' needs easily and efficiently, without the need for specialized interpreters.

1.3 Problem Definition:

The problem in medical clinics is that deaf individuals face significant difficulty in communicating with healthcare providers due to the lack of specialized interpreters for Saudi Sign Language (SSL). In clinics and hospitals, deaf patients struggle to express their health needs or the symptoms they are experiencing, leading to challenges in accurate diagnosis and appropriate treatment. Doctors and healthcare practitioners often do not master sign language, making communication between the two parties ineffective and negatively impacting the patient's experience. Additionally, relying on specialized interpreters at times can result in delays in the healthcare process, potentially worsening the health condition in emergency situations.

The lack of technological tools that support Saudi Sign Language in the medical environment hinders effective communication between doctors and deaf patients, complicating the provision of comprehensive and integrated healthcare.

1.4 Recommended Solution:

The recommended solution is to develop the "Mu'een" app, which aims to provide a technological tool that enables deaf patients to interact easily with healthcare providers. The app relies on Saudi Sign Language recognition technologies to convert sign language into text and voice in real-time, allowing doctors to quickly and accurately understand the patients' needs.

Through this app, deaf patients will be able to communicate directly with doctors without the need for specialized interpreters. The app will also provide a simple and user-friendly interface, enabling patients to effectively express their emotions or describe their symptoms. This will enhance the healthcare experience and contribute to improving the accuracy of diagnosis and treatment.

1.5 Aims:

The main objectives of the "Mu'een" app are to facilitate communication and collaboration between deaf patients and healthcare providers, and to achieve a better healthcare experience by improving communication using Saudi Sign Language. The objectives include:

- Improving effective communication between deaf patients and doctors by converting sign language gestures into text and voice in real-time.
- Reducing reliance on specialized interpreters in medical clinics, thus helping to lower costs and improve the speed of healthcare delivery.
- Enabling deaf patients to express their health problems easily and accurately, enhancing doctors' understanding of their health needs.
- Strengthening collaboration between patients and doctors by facilitating communication and ensuring active participation in treatment and care.
- Improving the healthcare experience for individuals with hearing disabilities in medical clinics by providing a technological tool that contributes to their access to comprehensive healthcare services.

1.6 Target users:

Deaf Patients:

The primary target of the "Mu'een" app is deaf patients who need effective means to communicate with healthcare providers. The app allows them to use Saudi Sign Language (SSL) to express their health needs and the symptoms they are experiencing in an accurate and efficient way, improving their experience in medical clinics.

Healthcare Providers:

This includes doctors and healthcare practitioners who interact with deaf patients. The app aims to facilitate communication by translating Saudi Sign Language gestures into text and voice in real-time, helping doctors to understand the patients' needs more quickly and accurately.

Healthcare Institutions:

This includes clinics and hospitals that require a technological tool to enable communication between deaf patients and medical staff, thus improving the quality of healthcare and facilitating access to comprehensive medical services.

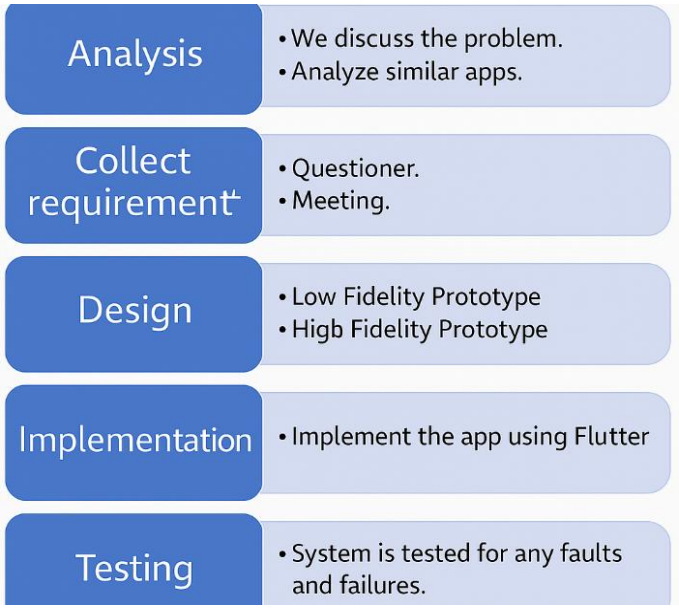
1.7 Objectives:

We decided to develop the "Mu'een" app, which aims to facilitate communication between deaf patients and healthcare providers, and to improve their experience in medical clinics by translating Saudi Sign Language (SSL) gestures into text and voice in real-time. The app's objectives can be summarized as follows:

- Facilitating effective communication between deaf patients and doctors by converting Saudi Sign Language gestures into text and voice.
- Enabling deaf patients to express their health needs easily and accurately through the app, making it easier for doctors to understand the symptoms and health issues the patient is facing.
- Reducing reliance on specialized interpreters in medical clinics, which helps lower costs and improve service speed.
- Improving the healthcare experience for individuals with hearing disabilities by providing an effective and simple communication tool that helps patients interact with doctors comfortably.
- Providing continuous support to doctors and patients by offering reports on the patient's health status, making it easier to monitor health conditions and track progress in treatment.

1.8 Methodology:

For this project, the Agile software development process was chosen because of its flexibility, user-centered design, and iterative nature. This approach fits in nicely with the project's goal of improving user response and involvement. Through frequent stakeholder feedback, agile allows the team to continue continuous improvement while effectively addressing real-time translation difficulties. Flexibility and adaptability are essential for real-time communication solutions in specialized situations since agile allows for adjustments depending on user feedback and new requirements. User-Centered Approach: Agile places a high value on user input, which closely relates to project goals and guaran



Figuer1.1 Waterfall Model Methodology Phases

1.9 Project Plan:

Our project focuses on creating a mobile application because it offers better accessibility and portability for users compared to a website. Mobile devices allow users to easily interact with the application on-the-go, making it more convenient, especially for medical staff and patients in various healthcare environments.

1.9.1 Software

The software we will use for our mobile application is **Flutter** for app development due to its cross-platform capabilities, allowing us to deploy the app on both Android and iOS. Additionally, we will use **Firebase** for database management and user authentication, as it is a powerful and scalable backend service. **Dart** programming language will be used in Flutter, and while the team is familiar with the basics, it will still present some challenges in terms of advanced features

1.9.2 Hardware

The hardware we will use to develop and test the mobile application includes high-performance **PCs/Laptops** for coding and debugging. We will also use **smartphones** (Android and iOS devices) to test the application's functionality and ensure compatibility across different platforms.

1.10 Report Organization:

Our project is divided into eight chapters.

Chapter 1: Introduction
It discusses the problem and how to solve it, and also summarizes the aim and objectives of the application.

Chapter 2: Literature Review
This chapter includes a literature review, examining related systems, their focus, and how they work. It also compares these systems to our project to highlight the differences and improvements made in **"Mueen"**.

Chapter 3: Data Analysis
This chapter includes the data we collected through questionnaires and interviews. It also presents two diagrams to describe our system: the **Use Case Diagram** and the **Data Flow Diagram (DFD)**.

Chapter 4: Design
This chapter presents the design of our application, including two levels of prototypes: **Low-Fidelity Prototype** and **High-Fidelity Prototype**. It also shows the database design and includes two diagrams: the **Entity Relationship Diagram (ERD)** and the **Relational Schema**.

Chapter 5: Implementation
This chapter describes the implementation of the **"Mueen"** app and the functionality of the main interfaces, along with the final database design.

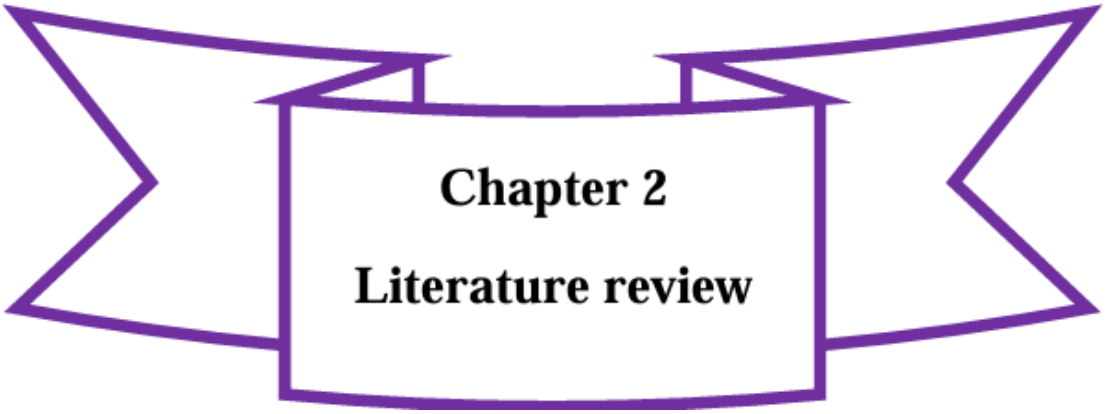
Chapter 6: Testing
This chapter covers the testing of the application, describing its different types, including functional and non-functional tests.

Chapter 7: Interface Description
This chapter provides a detailed description of all the app's interfaces and the functionality of each interface.

Chapter 8: Challenges and Future Work
This chapter discusses the challenges we faced during the implementation of the app, along with some suggestions for future improvements and enhancements to the application.

1.11 Conclusion:

patients and healthcare providers, aiming to make interactions in medical settings easier and more effective. **"Mueen"** empowers patients by providing a real-time translation of **Saudi Sign Language (SSL)** into text and audio, bridging the communication gap and enhancing the overall healthcare experience. The app allows for smoother communication without the need for an interpreter, helping to ensure better diagnosis and treatment. We hope to make the **"Mueen"** app a useful tool that serves the needs of the deaf community, providing them with a more inclusive healthcare experience and improving their access to medical services.



2.1 Introduction
2.2 Overview of Related System
2.3 Conclusion

2.1 Introduction:

This chapter presents a comparison between our project, **"Mueen"**, and other similar works to get the most benefit from existing experiences. The idea behind **"Mueen"** is to create a mobile application that connects deaf patients with healthcare providers, enabling real-time communication through **Saudi Sign Language (SSL)**. We will highlight some of the features as well as compare them with our project.

2.2 Overview of Related Systems

In this section, we review several related systems that have contributed to the development of sign language recognition technologies. Each system offers unique features and limitations that help guide and shape the *Mueen* application.

1. Efhamni: A Deep Learning-Based Saudi Sign Language Recognition Application

"Efhamni" is a mobile application developed to support communication for the deaf and hard-of-hearing community in Saudi Arabia. It uses Convolutional Neural Networks (CNNs) to recognize isolated Saudi Sign Language (SSL) gestures and translate them into text or speech. The system is known for its high accuracy and simple user interface, allowing users to perform gestures in front of a camera for instant translation. Despite its strengths, the application is limited to isolated gestures and may be affected by changes in lighting or background conditions.

2. Saudi Sign Language Recognition System Based on CNNs

This system focuses on recognizing Saudi Sign Language gestures using CNNs trained on a dataset of 40 signs and over 27,000 images. It was developed with consideration for real-world challenges, such as different skin tones, lighting conditions, and backgrounds. As a result, the model is robust and highly accurate. However, it is limited to static gestures and does not support dynamic or continuous signing, which makes it less suitable for real-time conversations.

3. SingAll: A Universal Sign Language Recognition Application

"SingAll" is a deep learning-based application designed for the global deaf community. It recognizes multiple sign languages and translates them into text and speech in real time. Using CNNs, the app delivers high accuracy for isolated gestures and features a user-friendly interface. However, the system still lacks support for continuous gestures and may face accuracy issues in uncontrolled environments.

4. SpeakSign: AI-Powered Sign Language Translation

"SpeakSign" is a more advanced application that integrates both CNNs and Recurrent Neural Networks (RNNs) to recognize static and continuous gestures. It supports several sign languages, including SSL, ASL, and BSL, making it accessible to users around the world. The support for continuous signing makes conversations more natural and fluid. Nevertheless, the system requires significant processing power and further development to cover specialized vocabulary, especially in medical contexts.

5. King Saud University – Saudi Sign Language (KSU-SSL) Dataset

The KSU-SSL dataset is the largest known dataset for Saudi Sign Language, containing 293 signs and more than 145,000 samples. It utilizes Convolutional Graph Neural Networks (CGCNs) to enhance recognition accuracy across different environments. The dataset is highly valuable for clinical applications like *Mueen*, offering reliable input data for training and testing. Its main challenge lies in the complexity of CGCNs, which may not run efficiently on all mobile devices.

6. Mueen: Real-Time SSL Translation in Dental Clinics

"Mueen" is developed specifically for clinical environments, focusing on improving communication between deaf patients and healthcare providers. It provides real-time translation of SSL gestures, addressing unique challenges in dental clinics such as mask-wearing, limited hand visibility, and medical-specific gestures. The system builds on the strengths of previous solutions while aiming to overcome their limitations, especially in dynamic and real-time communication scenarios.

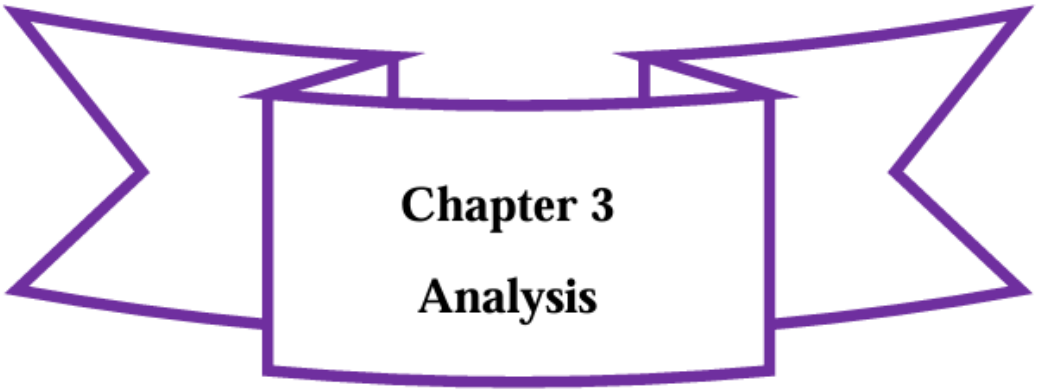
2.2.3: Comparison with related system:

Project Name	About	Key Features	Target Users	Focus	Advantages	Disadvantages
Efhamni	Deep learning-based SSL recognition for real-time translation.	High accuracy, real-time translation, user-friendly interface.	Deaf and hard-of-hearing individuals in Saudi Arabia.	Real-time communication for the deaf community.	High accuracy, scalable for other languages.	Limited to isolated gestures, sensitive to environment.
Saudi SSL Recognition System (CNN)	SSL gesture recognition using CNN for static gestures.	High accuracy, robust dataset with diverse conditions.	Deaf individuals in Saudi Arabia.	Static gesture recognition.	High accuracy, effective in diverse conditions.	Limited to static gestures, general application.
SingAll	Universal sign language translation using CNNs.	Multi-language support, real-time translation, user-friendly.	Global deaf community.	Cross-language sign language translation.	Multi-language support, real-time recognition.	Limited to isolated gestures, sensitive to environment.
SpeakSign	AI-powered translation using CNNs and RNNs for both static and dynamic gestures.	High accuracy for both gesture types, multilingual support.	Deaf and hard-of-hearing individuals globally.	Real-time and continuous gesture translation.	Supports continuous gestures, multilingual, real-time.	High computational needs, sensitive to environment.
Mueen	Real-time SSL translation in dental clinics for deaf patients.	Real-time translation, healthcare-specific signs, discussion forum.	Deaf patients and healthcare providers.	Real-time communication in dental clinics.	Focused on healthcare, real-time, practical use in clinics.	Limited to dental use, lacks general SSL coverage.
KSU-SSL Dataset	Largest Saudi SSL dataset, supporting clinical SSL development.	Covers 293 signs, high accuracy, CGCN-based recognition.	Developers working on SSL applications.	Dataset for clinical and general SSL use.	Comprehensive dataset, high recognition accuracy.	Focused only on SSL, not usable for other languages.

Table2.1 Comparative Summary of Related SSL Recognition Projects and Datasets

2.3 Conclusion

This chapter focused on comparing our project "Mueen" with other related systems and identifying their advantages and disadvantages. Through this comparison, we showed how our application stands out by focusing on real-time translation in dental clinics using Saudi Sign Language. In addition, most of the limitations found in the reviewed systems, such as lack of continuous gesture support or limited medical focus, were taken into consideration and avoided as much as possible in our project.



3.1 Introduction

In this chapter we discuss the analysis phase of this project. It reviews the requirements that revealed from analyzing the current website through the questionnaires that we published among the target users, administrator and student/researcher, to collect information that helps to meet their desires. The results of the questionnaires have been analyzed to extract the requirement speciation of the website. Based on questionnaires analysis and results which contain user opinions, we determined all the tools and techniques that are used during project. For more clarity, we create diagrams that explain the basic structure of the system requirement.

3.2 Data collection

We conducted a questionnaire for a group of students and faculty from the Faculty of Computing and Information Technology because we have huge number of target users and need to take their responses and suggesting, which is includes 13 questions to collect the requirements and we collected 50 responses from students and 10 responses from faculty. We conducted an interview with deanship of scientific research to ask them about administrator features and with IT support team to ask them technical questions.

3.2.1.1 target users questionnaire:

We published the questionnaire to a group of and deaf prsons and then monitored the results as follows:

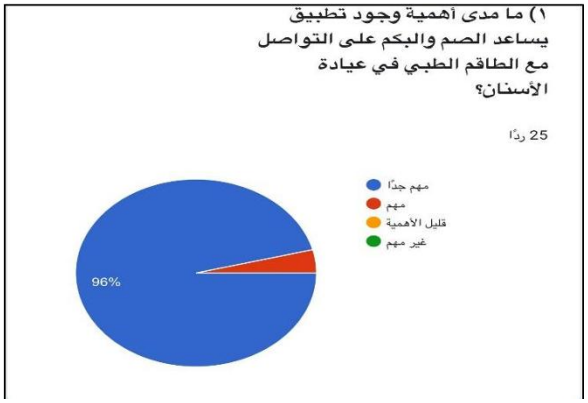


Figure3.1 Question about participation in a project

1. Question about Participation in a Multidisciplinary Project

As shown in Figure 3.1, we asked the users if they have ever participated in a project with people different from their specialty. 56% said "Yes," while 44% said "No." This shows that more than half of the participants have experience working with individuals from other disciplines, indicating a high level of openness toward multidisciplinary collaboration

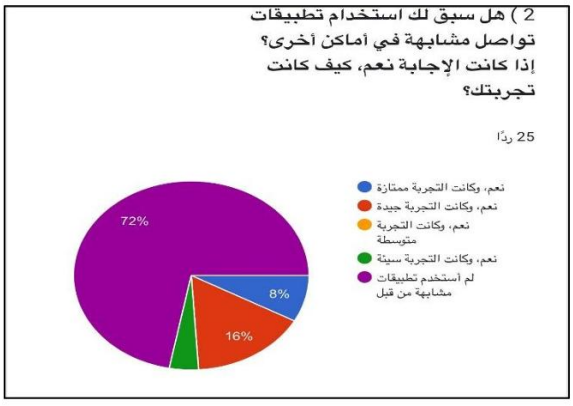


Figure3.2 ***Use Question about the Use of Similar Applications***

2. **Question about the Use of Similar Applications in Other Places**

As shown in Figure 3.2, we asked the users if they have used similar applications in other places. 72% said "Yes," and 8% said "No." This result indicates that a large proportion of users have prior experience with such applications, which suggests that the target users are familiar with and open to the idea of using similar applications in other contexts.

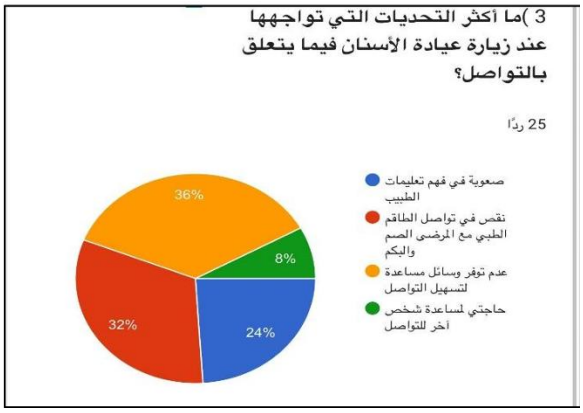


Figure3.3 Challenges Faced When Visiting a Dental Clinic

3. **Challenges Faced When Visiting a Dental Clinic**

As shown in Figure 3.3, we asked the users what challenges they face when visiting a dental clinic related to communication. 36% mentioned "Difficulty in understanding instructions," 24% mentioned "Difficulty in communicating with the doctor," and 32% mentioned "Lack of effective communication tools."

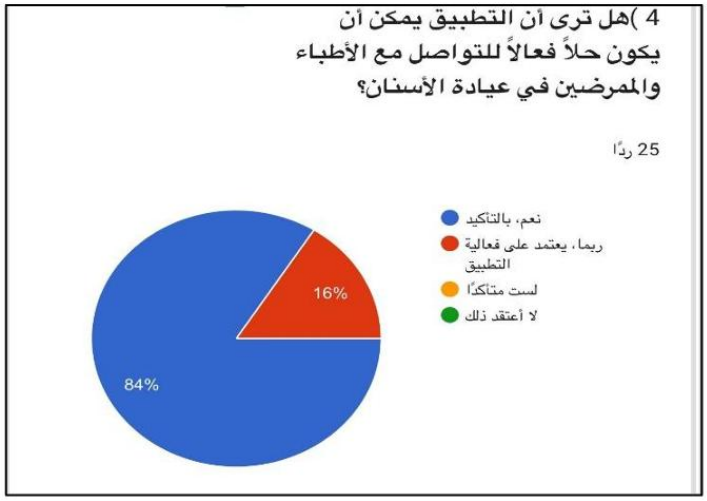


Figure3.4 Can the Application Be Effective for Communication with Doctors

4. Can the Application Be Effective for Communication with Doctors?

As shown in Figure 3.4, we asked the users if they think the application can be effective in communicating with doctors. 84% said "Yes," and 16% said "No." This result shows a high level of confidence in the application's potential to improve communication in medical settings, particularly dental clinics.

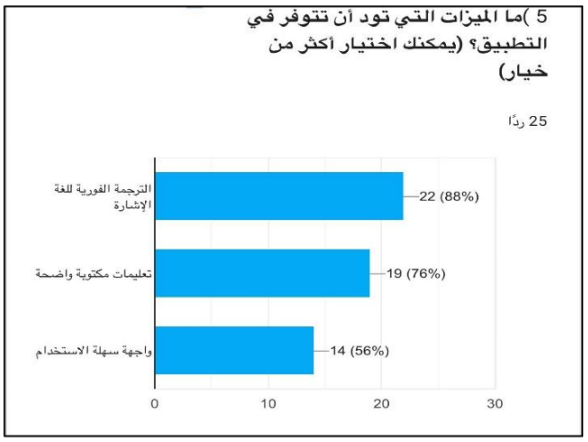


Figure3.5 What Features Should the Application Provide1

5. What Features Should the Application Provide?

As shown in Figure 3.5, we asked the users what features they want to see in the application. 88% chose "Support for visual and audio communication," 76% selected "Ability to control audio," and 56% selected "Control over visual elements." This suggests that users prioritize effective visual and audio communication, with an emphasis on flexibility and control.

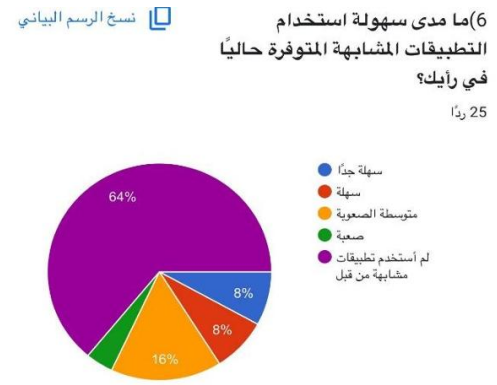


Figure3.6 Ease of Use of Assistive Technologies in the Application

6. Ease of Use of Assistive Technologies in the Application

As shown in Figure 3.6, we asked the users how easy it is to use assistive technologies in the application. 64% said "Very easy," and 16% said "Moderately easy." This indicates that most participants find assistive technologies easy to use, which enhances the likelihood of user adoption

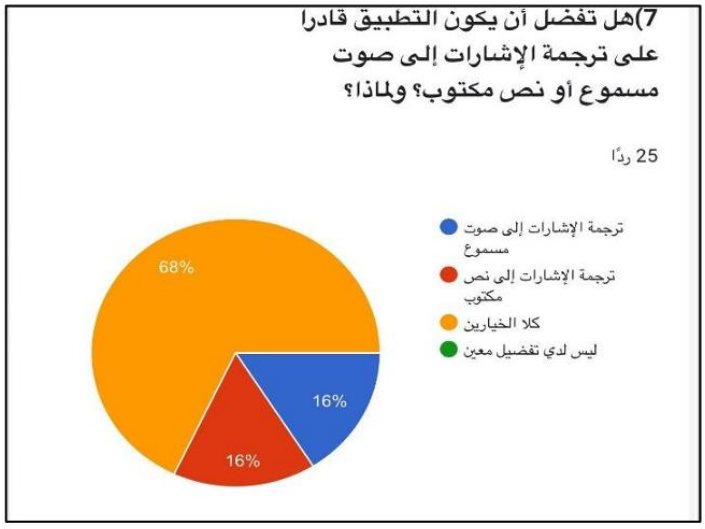


Figure3.7 Preference for Translating Gestures into Audio or Text

7. Preference for Translating Gestures into Audio or Text

As shown in Figure 3.7, we asked the users if they prefer the application to translate gestures into audio or text. 68% preferred "Audio translation," while 16% preferred "Text translation." This suggests that users favor audio translation for better real-time interaction, and this should be a priority feature in the application

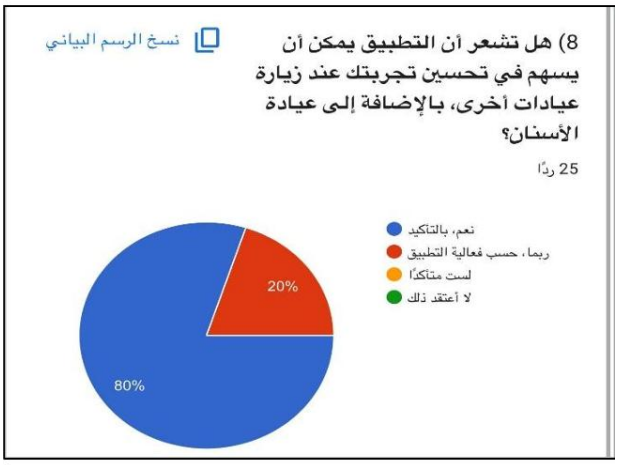


Figure3.8 Will the Application Improve Your Experience at Other Dental Clinics

8.Will the Application Improve Your Experience at Other Dental Clinics?

As shown in Figure 3.8, we asked the users if they think the application will improve their experience when visiting other dental clinics. 80% said "Yes," and 20% said "Maybe." This shows a strong belief that the application can positively impact the user experience in dental clinics

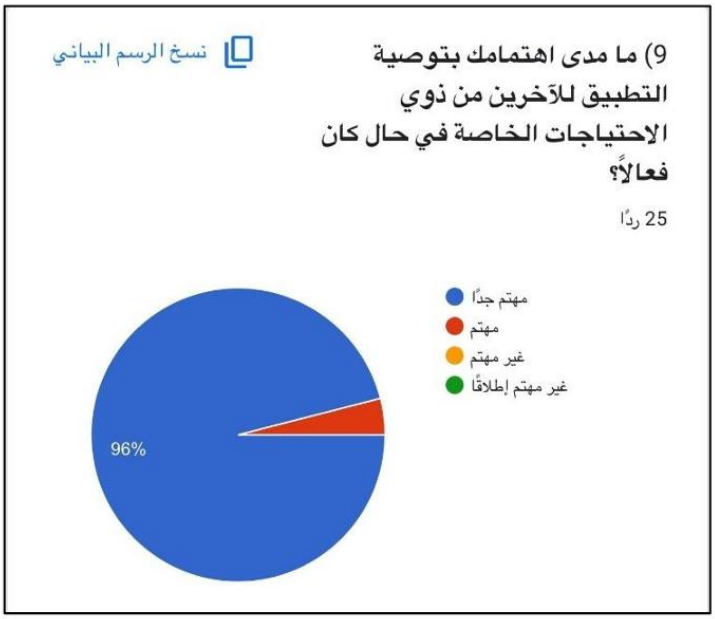


Figure3.9 How Important is the Application for People with Special Needs

9. How Important is the Application for People with Special Needs?

As shown in Figure 3.9, we asked the users how important the application is for individuals with special needs. 96% considered it "Very important," indicating a strong demand for inclusive features that cater to those with hearing impairments

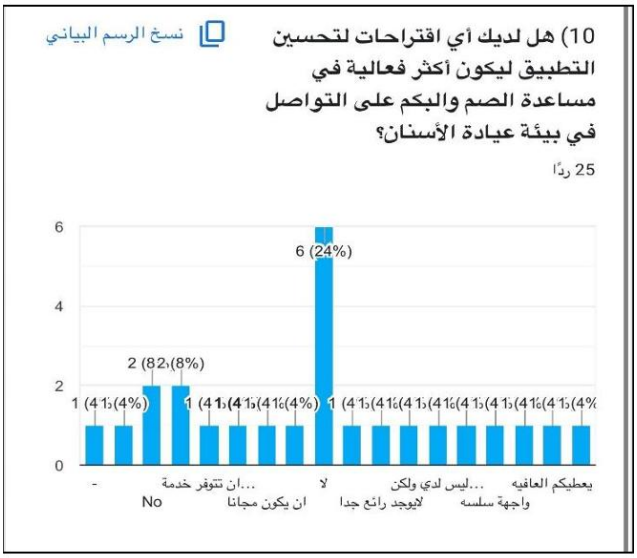


figure 3.10 Suggestions for Improving the Application

10. Suggestions for Improving the Application

As shown in Figure 3.10, we asked the users if they had any suggestions for improving the application. 24% suggested improvements, such as adding more features for communication. This reflects an interest in making the application more comprehensive, and the feedback can be used for future development.

Data Analysis:

The information gathered from our target consumers' interviews and surveys was examined in this part. The findings validated the necessity for a solution such as "Mueen" and offered insightful information about the difficulties deaf people have at dental clinics. Features like audio support, convenience of use, and real-time translation were highly desired by users. These results led the creation of features that meet user expectations and assisted us in defining precise system requirements.

3.3 Requirement specification:

3.3.1 Functional requirements:

Perform SSL Gestures:
• The system must enable users (both guests and registered users) to perform Saudi Sign Language gestures, which are captured through the camera for processing.

Translate SSL Gestures:
• The system must process SSL gestures using the recognition engine and translate them into text or audio outputs in real-time.

Display Text:
• The system must display the translated gestures as text on the screen to provide a clear, readable format for communication.

Generate Audio:
• The system must convert translated gestures into audio format using text-to-speech technology for verbal communication.

Save in Library:
• The system must allow registered users to save translated outputs in their personal library for future reference.

History of Saved Translations:
• The system must provide registered users access to their history of saved translations, enabling them to retrieve and review past translations.

Edit Info:
• The system must allow registered users to edit their personal account information, such as username, email, and preferences.

Camera Integration:
• The system must support the integration of a camera to capture SSL gestures accurately for processing.
• The system will require a custom dataset of Saudi Sign Language (SSL) gestures, which will be collected as part of this project. This dataset will include medical-specific gestures commonly used in dental clinics and general SSL gestures for broader application.

User Management:
• The system must differentiate between guests and registered users, restricting certain functionalities (e.g., saving data and accessing history) to registered users only.

3.2 Non-Functional Requirements

Performance and Responsiveness:
The app should perform translations with minimal latency (under two seconds per gesture) to ensure efficient and real-time communication.

Accuracy of Recognition:
The gesture recognition engine should maintain an accuracy rate above 90% for dental-specific gestures and above 85% for general SSL gestures.

Security and Data Privacy:
The application should adhere to Saudi data protection regulations, especially for healthcare settings. Sensitive information, such as patient records, should be encrypted.

User-Friendly Interface:
Design the interface to be accessible and easy to use for all users, including those with limited technical experience. Clear instructions and an intuitive design should enable easy navigation and operation.

Scalability:
The app should be designed to support future expansions, such as additional sign languages or broader medical vocabulary, without major reconfigurations.

3.4 User Profile:

In this section, we define each user type of our system and what actions they can perform.

3.4.1 User Categories:

There are three types of users who interact with the **Mueen** application: registered users (deaf individuals), guest users, and administrators. Each of these users has different roles and functionalities within the application.

- **Registered Users (Deaf Individuals):**
  Registered users are those who have signed up for the application and are seeking to communicate effectively in dental clinics using Saudi Sign Language (SSL). They can log in to the application, perform SSL gestures using the camera, and view the translated text or audio output. They also have the ability to save their translations in a personal library for future reference. Furthermore, they can access their translation history and edit their account information, such as username and preferences.
- **Guest Users:**
  Guest users are those who access the application without registering or logging in. They can use the "Open Camera" feature to perform SSL gestures and receive translations in real time, but they cannot save data or access the translation history. Guest users can only interact with the application in a limited manner.

3.5 System Structure:

- We decided to use a **use case diagram** because it holds functional requirements in an easy-to-read and trackable format, and it represents the goal of interaction between an actor (user) and the system. The use case diagram will help illustrate the primary functionalities of the **Mueen** app, such as capturing gestures, translating them into text/audio, and saving them in a user's library.
- Additionally, we decided to use a **data flow diagram** because it illustrates how data is processed by the system in terms of inputs and outputs. The data flow diagram will demonstrate how SSL gestures are captured, recognized, translated into text or audio, and saved or displayed to the user. It will also show the flow of data between components like the camera, recognition model, and database.

3.5.1 Use case model:

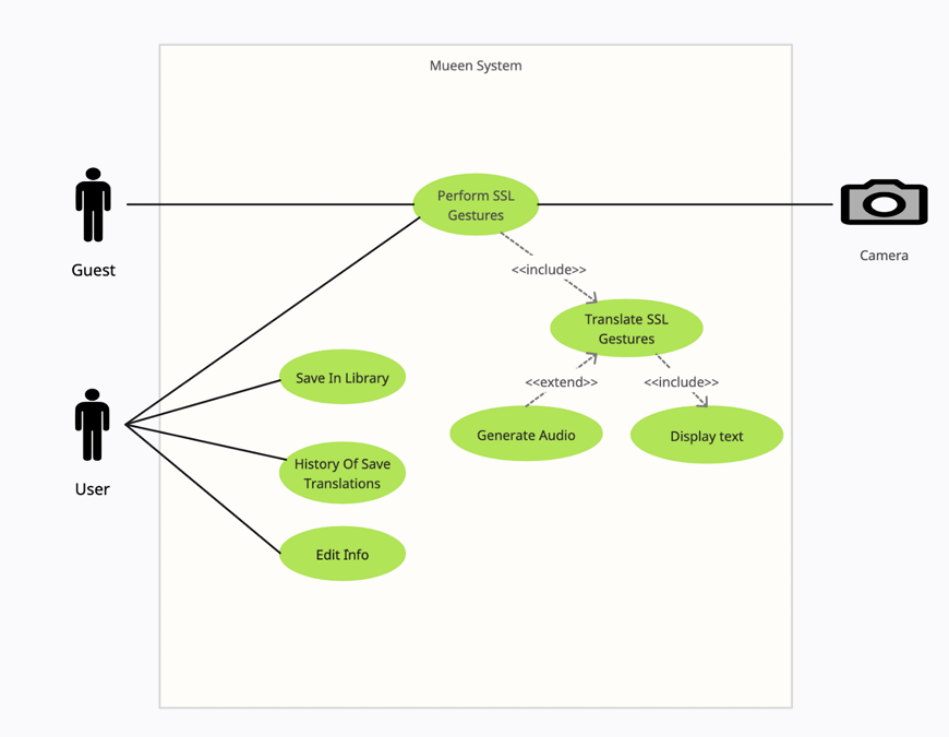


figure 3.11 Use case model

3.5.1.1 Use case description:

Use Case 1: Perform SSL Gestures

- **Brief Description:** Allow users (both guests and registered users) to perform Saudi Sign Language (SSL) gestures, captured through the camera for processing.
- **Actors:** Guest, User
- **Preconditions:** The camera must be active and accessible.
- **Main Flows:**
    1. The user (Guest or Registered) activates the camera.
    2. The user performs the SSL gesture.
    3. The system captures the gesture using the camera.
- **Post Conditions:** The gesture is captured and ready for translation.
- **Alternative Flow:** None

Use Case 2: Translate SSL Gestures

- **Brief Description:** The system processes SSL gestures using the recognition engine and translates them into text or audio outputs in real-time.
- **Actors:** User (Guest or Registered)
- **Preconditions:** SSL gesture must be captured by the camera.

- **Main Flows:**
    1. The captured gesture is sent to the Gesture Recognition Engine.
    2. The gesture is translated into either text or audio.
- **Post Conditions:** The recognized gesture is translated into text/audio.
- **Alternative Flow:** If the system fails to recognize the gesture, a message is displayed indicating failure.

Use Case 3: Display Text

- **Brief Description:** The system displays the translated gesture as text on the screen to provide a readable format for communication.
- **Actors:** User (Guest or Registered)
- **Preconditions:** The gesture has been successfully translated into text.
- **Main Flows:**
    1. The translated text is displayed on the screen.
- **Post Conditions:** The user sees the translated text on the screen.
- **Alternative Flow:** If no text translation is available, a "Translation failed" message is shown.

Use Case 4: Generate Audio

- **Brief Description:** The system converts the translated gestures into audio format using text-to-speech technology for verbal communication.
- **Actors:** User (Guest or Registered)
- **Preconditions:** The gesture has been successfully translated into text.
- **Main Flows:**
    1. The user clicks "Play Audio" to convert the text into audio.
    2. The system plays the audio.
- **Post Conditions:** The user hears the audio translation.
- **Alternative Flow:** If audio generation fails, an error message is displayed.

Use Case 5: Save in Library

- **Brief Description:** Allow registered users to save translated outputs (both gesture and its corresponding text/audio) in their personal library for future reference.
- **Actors:** Registered Users
- **Preconditions:** User must be logged in.
- **Main Flows:**
    1. The user selects the option to save the translation.
    2. The translation is saved in the user's personal library.
- **Post Conditions:** The translation is saved and can be accessed later.
- **Alternative Flow:** If the user is not logged in, the system prompts the user to log in.

Use Case 6: History of Save Translations

- **Brief Description:** The system must provide registered users access to their history of saved translations, enabling them to retrieve and review past translations.
- **Actors:** Registered Users
- **Preconditions:** User must be logged in and have saved translations.
- **Main Flows:**
    1. The user accesses their translation history.
    2. The system retrieves the saved translations.
- **Post Conditions:** The user can view their past translations.
- **Alternative Flow:** If no history is available, a message will indicate that there are no saved translations.

Use Case 7: Edit Info

- **Brief Description:** The system allows registered users to edit their personal account information such as username, email, and preferences.
- **Actors:** Registered Users
- **Preconditions:** User must be logged in.
- **Main Flows:**
    1. The user selects the "Edit Info" option.
    2. The user edits their personal information.
    3. The system saves the updated information.
- **Post Conditions:** The user information is updated in the system.
- **Alternative Flow:** If the user enters invalid data, an error message will be shown.

Use Case 8: Save in Library (For Guest)

- **Brief Description:** Allow guests to save translated gestures, though with limited functionality compared to registered users.
- **Actors:** Guest
- **Preconditions:** Guest must perform gestures.
- **Main Flows:**
    1. Guest performs gestures.
    2. System processes and translates gesture.
    3. The translation can be saved but with restricted access to features compared to registered users.
- **Post Conditions:** The guest translation is saved, but access to certain functionalities (like viewing history) is restricted.
- **Alternative Flow:** None

Mapping Requirements

Use Case Name	Gesture Execution	Gesture Translation	Text Display	Audio Output	Save Translations	View Translation History	Update Account Info	Camera Support	User Access Management
Perform SSL Gestures	✔							✔	
Translate SSL Gestures		✔	✔	✔				✔	
Display Text			✔						
Generate Audio				✔					
Save in Library					✔				✔
History of Saved Translations						✔			✔
Edit Info							✔		✔
Camera Integration	✔	✔						✔	

figure 3.12 Mapping

3.5.2.1 Class Diagram description

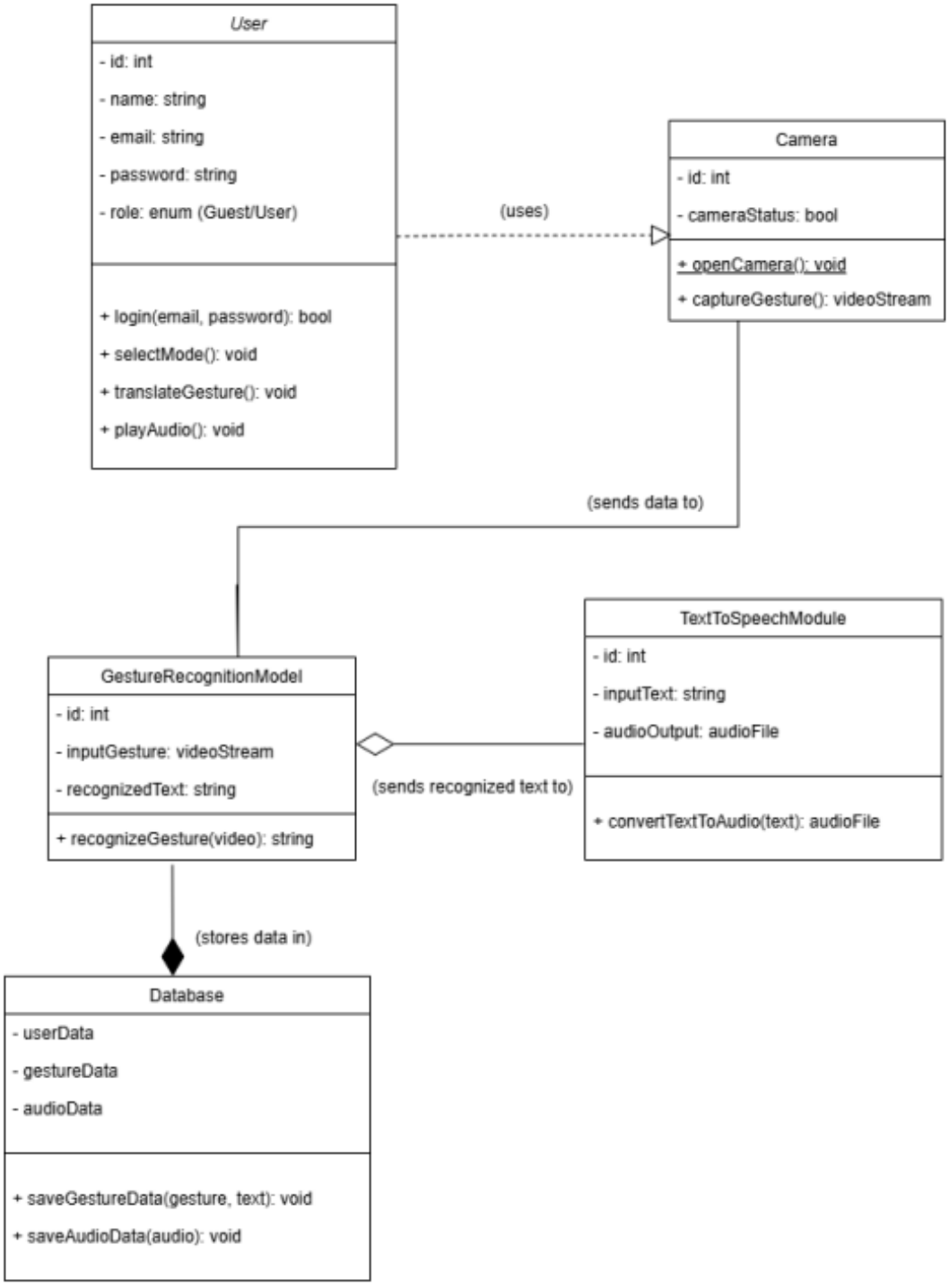


figure 3.13 Class Diagram

User Class

- **login(email, password):** Authenticates user credentials.
- **selectMode():** Allows mode selection (text or audio translation).
- **translateGesture():** Initiates gesture translation into text.
- **playAudio():** Converts recognized text into audio.

Camera Class

- **openCamera():** Activates the camera for gesture capture.
- **captureGesture():** Records gesture input and streams it to the recognition model.

GestureRecognitionModel Class

- **recognizeGesture(video):** Processes video input to produce recognized text.

TextToSpeechModule Class

- **convertTextToAudio(text):** Converts recognized text into an audio file.

Database Class

- **validateUser(email, password):** Verifies user credentials for login.
- **saveGestureData(gesture, text):** Stores gestures and their recognized text.
- **saveAudioData(audio):** Saves generated audio for future use.

3.5.3 Sequence Diagram

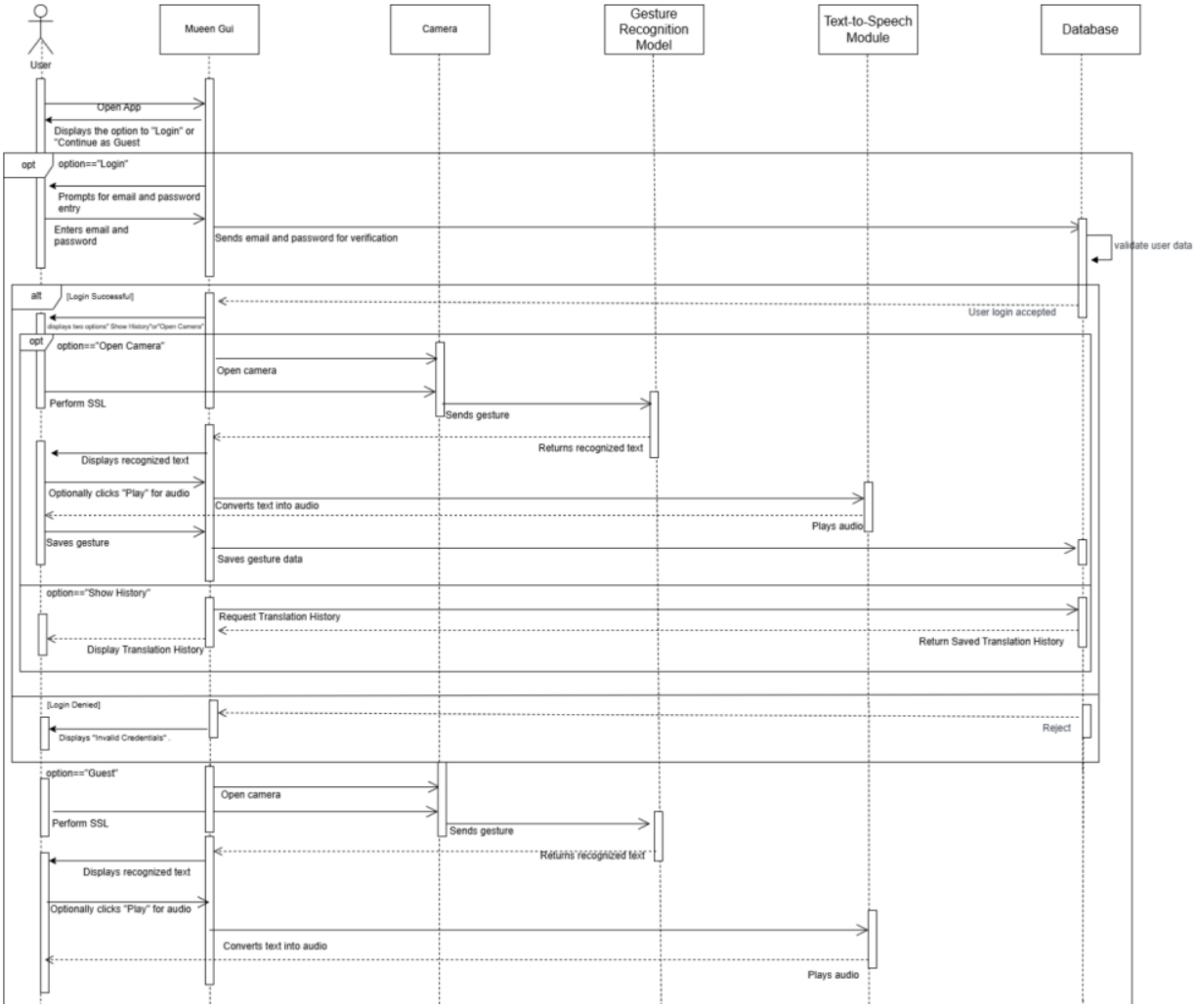


figure 3.14 Sequence Diagram

3.5.3.1 Sequence Diagram Description

Sequence Diagram Description for the Mueen App:
This sequence diagram illustrates the interaction between the various components of the Mueen App, designed to assist hearing-impaired individuals in dental clinics by translating Saudi Sign Language (SSL) into text and audio. The diagram details the application's flow across multiple modules, including Mueen GUI, Camera, Gesture Recognition Model, Text-to-Speech Module, and Database. Below is a step-by-step explanation:

1. **User Authentication:**
    - The user starts the application and is presented with the option to Log in or Continue as Guest.
    - If the user chooses to Log in:
        - The application prompts for an email and password.
        - The entered credentials are sent to the Database for verification.
        - If the credentials are valid, the user is logged in, and their data is retrieved. If invalid, the system displays "Invalid Credentials" and denies access.
2. **Open Camera for Sign Language Recognition:**
    - After logging in (or continuing as a guest), the user selects the option to Open Camera.
    - The camera module activates and captures gestures performed by the user in SSL.
    - The gestures are sent to the Gesture Recognition Model, which processes the input and returns the recognized text.
3. **Display and Play Recognized Text:**
    - The recognized text is displayed on the Mueen GUI for the user and optionally converted into audio through the Text-to-Speech Module.
    - If the user selects the "Play" option, the app plays the audio version of the recognized text.
4. **Save Translation Data:**
    - For logged-in users, the system offers the option to save the translation data (both gesture and its corresponding text/audio) to the Database for future reference.

5. Translation History:

- Logged-in users can choose to view their Translation History.
- A request is sent to the Database to retrieve the saved translation history, which is displayed on the Mueen GUI.

6. Guest Mode:

- If the user selects the Guest Mode, they can access the Open Camera feature without logging in.
- Gestures are captured, recognized, and translated into text and audio.
- However, guest users cannot save translation history or access previously saved data.

3.5.4Normalization

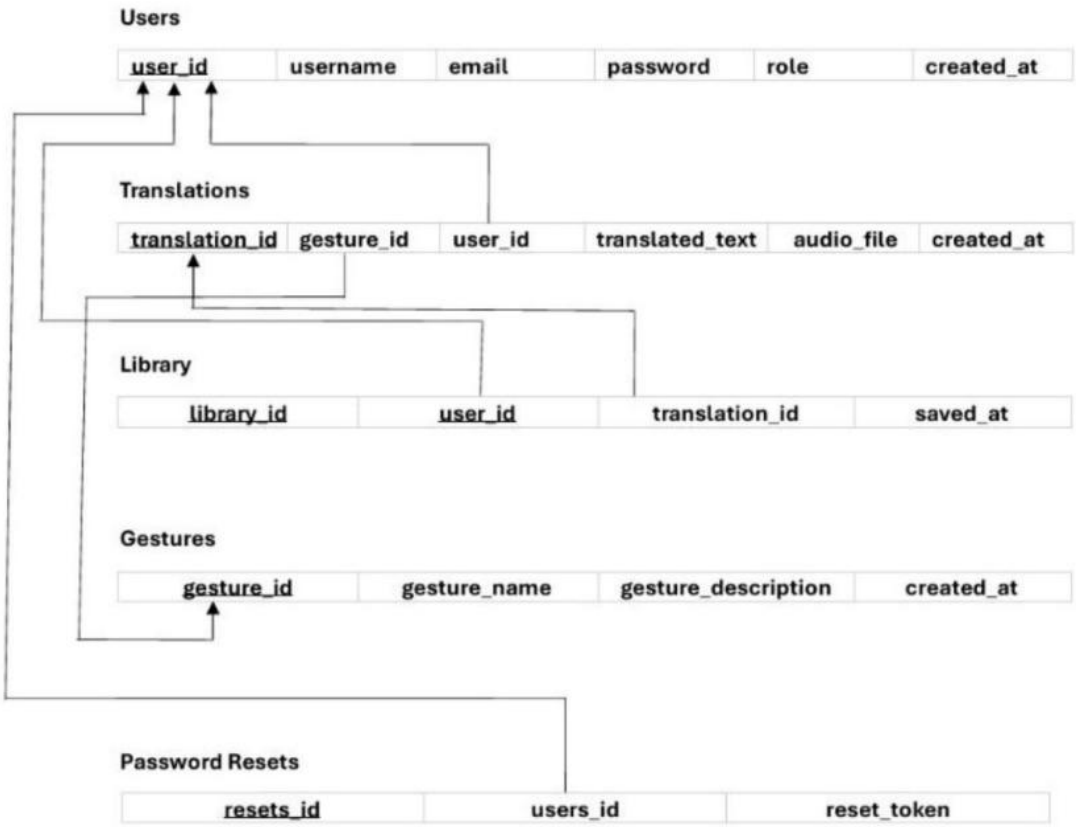
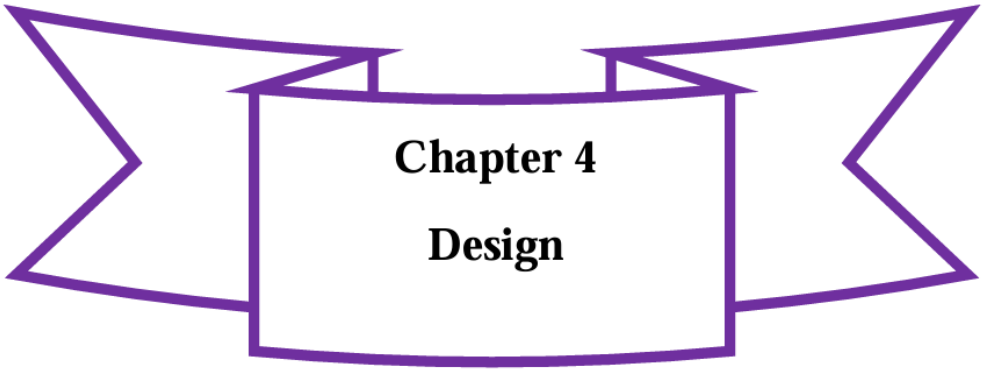


Figure 3.15 Normalization

The normalization applied to the "Mueen" app's database ensures efficient data management by minimizing redundancy and maintaining data integrity. It organizes data into related tables, enabling faster and more accurate queries, such as retrieving user-specific translations or saved gestures. This structure also enhances scalability, allowing for the seamless addition of new features or gesture categories in the future. By reducing duplication, normalization optimizes storage usage and ensures consistent relationships between entities, making the app reliable and efficient in handling real-time translation and user interactions.

3.6 Conclusion

The Mueen system can be broken down into two main subsystems. The first subsystem focuses on detecting and translating Saudi Sign Language (SSL) gestures. This process begins when the user or guest activates the camera to capture SSL gestures. The gestures are then processed by the system's translation engine to generate either text or audio outputs. These outputs address communication barriers, particularly in medical and dental settings. The second subsystem is dedicated to user management and personalization. Registered users can save translated data in their personal library, view the history of saved translations, and edit their account details. Guests, in contrast, have access only to real-time translation without the ability to save or manage data. The analysis and design phase demonstrated the use cases for the core functionalities of the system and specified the primary actors, including the User, Guest, and Camera. The traceability matrix mapped the functional requirements to the use cases, ensuring that all necessary features, such as SSL translation, user account management, and data storage, are implemented. Additionally, the non-functional requirements outlined the system's performance, accuracy, security, and user-friendliness to ensure optimal usability and reliability. This chapter provides a detailed overview of the system's design and requirements, establishing the foundation for an efficient and inclusive translation application tailored for the Saudi deaf community



4.1 Introduction

4.2 System prototype

4.3 Database design

4.4 Conclusion

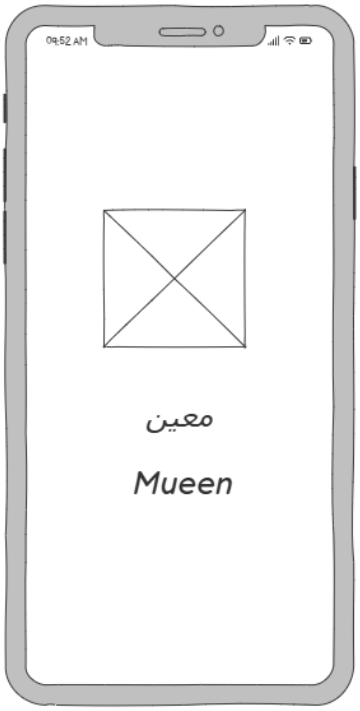
4.1 Introduction:

The designing phase is a critical stage in the project. At this stage, we will brainstorm product ideas based on the user needs. They aim to produce initial concepts in the form of sketches and outline specifications. In this chapter we will draw the system prototype (low and high fidelity) and also design the database.

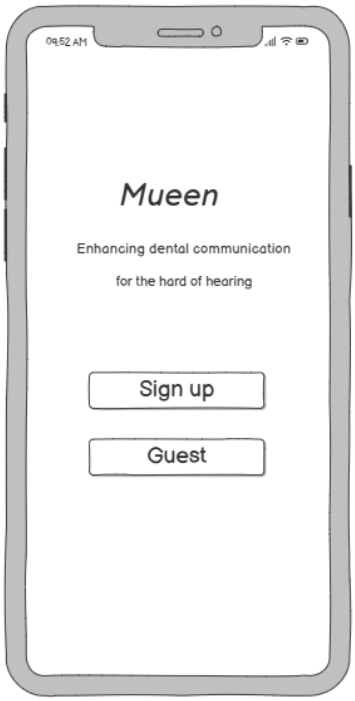
4.2 System prototype

4.2.1 low-fidelity prototype

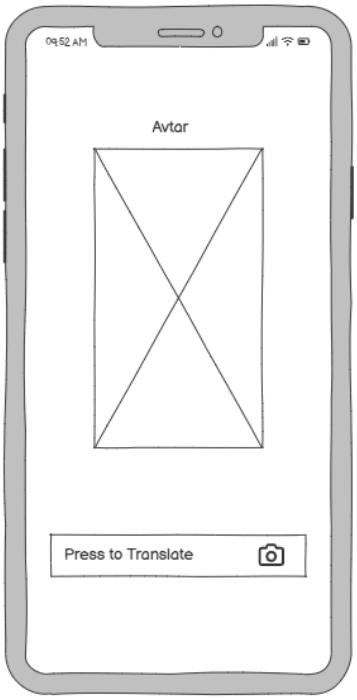
This section presents the initial sketches of the Mueen application. These wireframes illustrate the basic structure and layout of the main screens, including login, signup, home, camera, and settings. The goal is to visualize the user flow and overall design before building the high-fidelity prototype



4.1 Splash Screen



4.2Welcome Screen



4.3Home / Translation Screen

Figure 4.1: Splash Screen

This screen is the splash screen that appears when the user opens the application. It simply displays the app name "Mueen" in both Arabic and English, along with a logo placeholder in the center. Its purpose is to introduce the app in a minimal and clean way before navigating to the next screen.

Figure 4.2: Welcome Screen
The welcome screen presents the main purpose of the app: enhancing dental communication for individuals who are hard of hearing. It includes two buttons"Sign up" for users who want to create an account, and "Guest" for those who prefer to access the app without signing up.

Figure 4.3: Translation Screen
This is the main screen where the user can activate the translation feature. It contains an avatar placeholder and a "Press to Translate" button with a camera icon. When pressed, the app opens the camera to recognize and translate Saudi Sign Language (SSL) gestures in real time to support patient-clinic interaction.

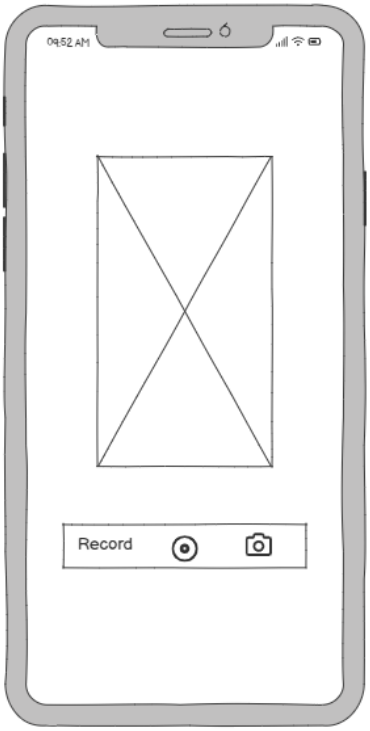


Figure 4.4: Recording Screen

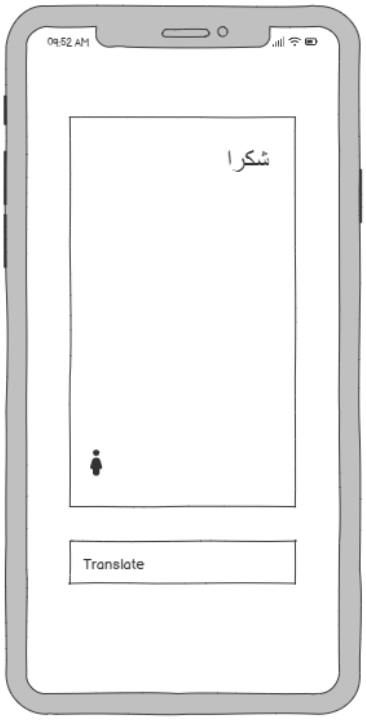


Figure 4.5: Result / Translation Output Screen

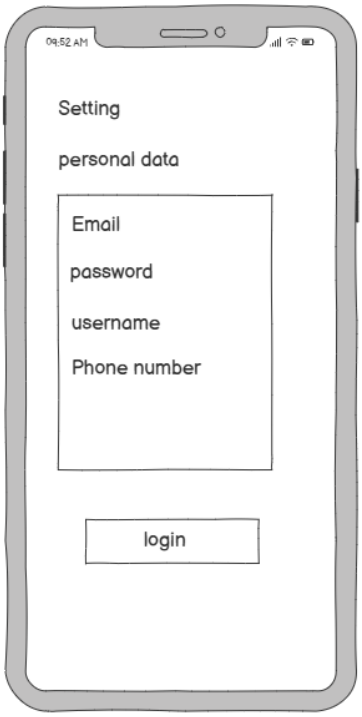


Figure 4.6: Settings Screen

Figure 4.4: Recording Screen
This screen allows users to either record a video using the camera or capture an image. It contains a large camera preview at the top and two buttons at the bottom: one for video recording and another for taking a photo. This is the main interaction point where the user provides sign language input to be translated.

Figure 4.5: Result / Translation Output Screen
After recording or capturing a sign, this screen displays the translated result in text form. In this example, the translated word is "شكراً" (Thank you). The screen also includes a translate button in case the user wants to trigger the translation again.

Figure 4.6: Settings Screen
This screen allows users to view and manage their personal information, such as email, password, username, and phone number. There is also a login button at the bottom, which might be used for authentication if the user is editing their data.

Figure 4.7: Login Screen

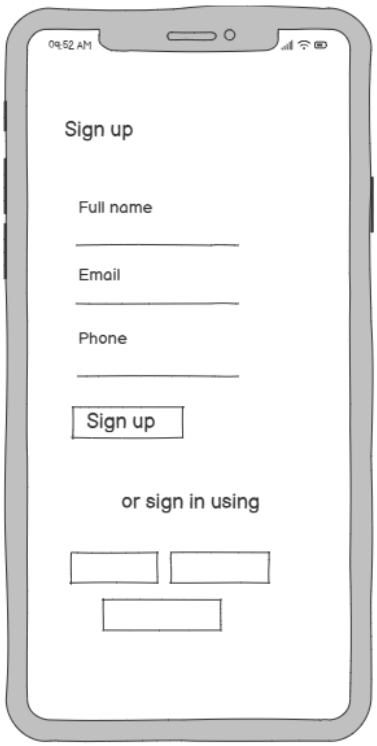


Figure 4.8: Sign-Up Screen

Figure 4.7: Login Screen
This screen allows registered users to log in by entering their email and password. It includes a "Forget password?" link for password recovery and a "Login" button to access the system. Additionally, it offers options to sign in using alternative methods such as social media or third-party services.

Figure 4.8: Sign-Up Screen
This screen is designed for new users who want to create an account. It includes fields for full name, email, and phone number, followed by a "Sign up" button to complete the registration process. Like the login screen, users can also sign up using other services if preferred.

4.3 Database Design:

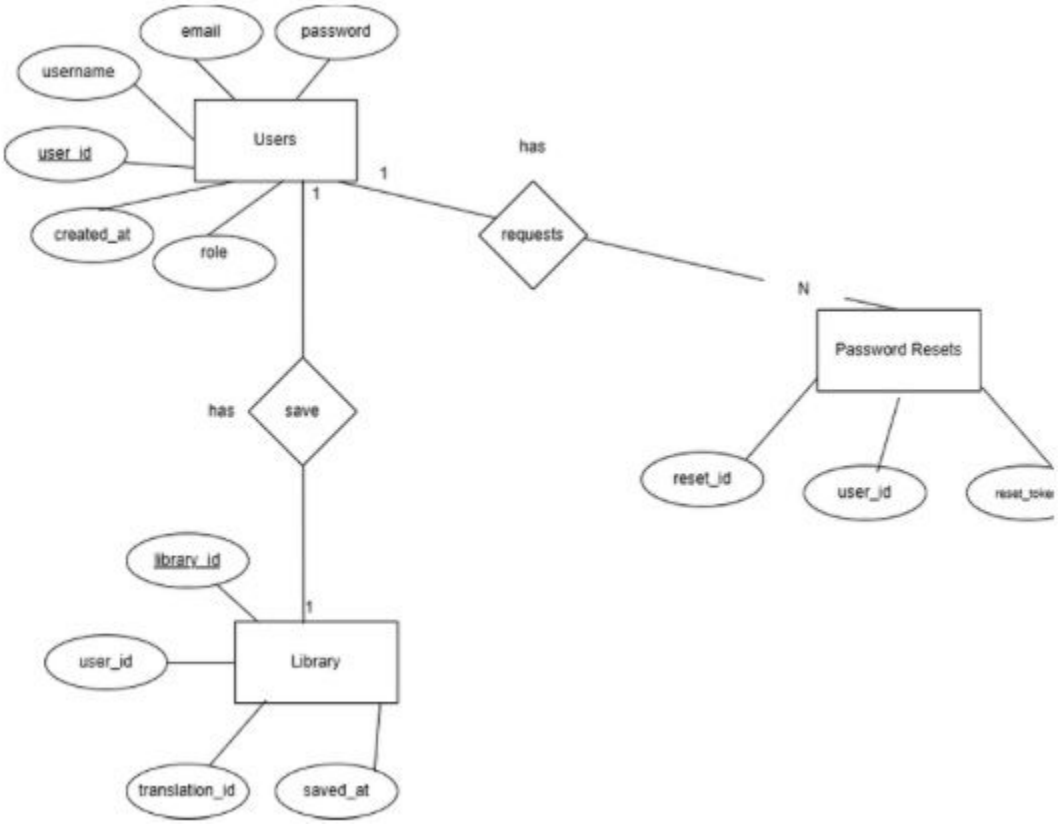
Database design is a method of generating a complete data model of the database. This logical data model has all the required logical and physical design option and physical storage constraint required to create a design in a data definition language.

4.3.1 Logical Database Design:

The logical model defines what the columns in each table are. While writing the logical model, you might also take into consideration the actual database system you're designing.

4.3.1.1 Entity Relationship Diagram:

.



Figuer4.9 ER

4.3.2 Physical Database Design

The physical database design includes the full structure of all data elements used in the system, including field names, data types, relationships, and constraints as implemented in the DBMS. The following tables represent the physical implementation of the Mueen application's database:
Users Table

#	Column	Data type	Not null	Constraint
1	User_ID	int	No	Primary key
2	Username	nvarchar(50)	No	
3	Email	nvarchar(100)	No	
4	Password	nvarchar(255)	No	
5	Role	nvarchar(50)	No	
6	Created_at	datetime	No	

Table4.19: Users Table

Password Resets Table

#	Column	Data type	Not null	Constraint
1	Reset_ID	int	No	Primary key
2	User_ID	int	No	Foreign key (Users)
3	Reset_Token	nvarchar(255)	No	

Figure 4.20: Password Resets Table

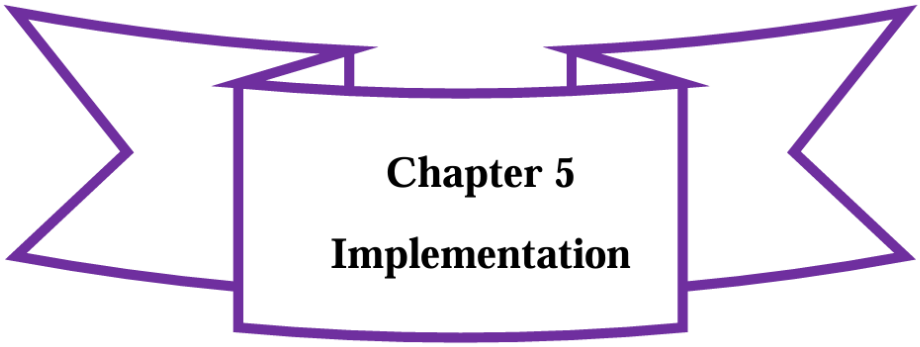
Library Table

#	Column	Data type	Not null	Constraint
1	Library_ID	int	No	Primary key
2	User_ID	int	No	Foreign key (Users)
3	Translation_ID	int	No	Foreign key (Translations)
4	Saved_at	datetime	No	

Figure 4.21: Library Table

4.4 Conclusion

the implementation phase of the "Mueen" application demonstrates the practical realization of our design and development goals. By integrating deep learning models, real-time camera functionality, and a user-friendly interface built with modern tools like Flutter and React Native, we were able to create a solution that supports real-time Saudi Sign Language (SSL) translation within dental clinics. The system was carefully structured with clear UI flows, robust backend communication, and effective data validation. Throughout development, various technical challenges were resolved through debugging and external research, enhancing both stability and user experience. Each screen and function was implemented with the user’s needs in mind, especially focusing on accessibility for the deaf and hard-of-hearing community. This chapter reflects the successful execution of our concept into a functioning, interactive applicatio



5.1 Introduction
5.2 Tools
5.3 System Implementation
5.4 Code Debugging and Troubleshooting
5.5 Packaging and documentation
5.6 Conclusion

5.1 Introduction:

Implementation is the carrying out, execution, or practice of a plan, method, or any design, idea, model, specification, standard, or policy in order to achieve something. It represents the necessary action that follows any preliminary planning to bring a concept into reality [14]. This chapter will cover the essential tools used in developing our mobile application, including the equipment, programming languages used, a description of key user interfaces within the app, and a walkthrough of the app's main functionalities and activities.

5.2 Tools:

The "Mueen" project bridges the communication gap in dental clinics for deaf and hard-of-hearing patients by converting Saudi Sign Language through a combination of text with the option to convert to audio. It employs techniques for deep learning and image processing, user interface design, and optimization, each of which has its advantages and disadvantages.

1. Deep Learning and Image Processing Tools

1.1 OpenCV

OpenCV was utilized at the early stage of the project to extract frames from video files. This step was essential for preparing the dataset that would be used for gesture recognition training. By using OpenCV functions, individual frames containing hand gestures were isolated and saved for further processing.

1.2 MakeSense

After extracting the frames, the MakeSense annotation tool was used to label and annotate the hand gestures in each frame. This labeling process is a crucial step in training any object detection model, ensuring that the system can learn to identify and classify hand movements correctly.

1.3 YOLO (You Only Look Once)

YOLO was the chosen model for gesture detection and recognition. After preparing and annotating the dataset, we trained the YOLO model to detect and classify specific Saudi Sign Language (SSL) gestures from real-time video input. YOLO's speed and accuracy made it a suitable choice for capturing hand gestures efficiently and in real time.

3. User Interface (UI) Development Tools

3.1 Flutter

Flutter is a cross-platform UI toolkit that supports mobile, web, and desktop development from a single codebase. Its customizable widgets and design options enable the creation of a visually appealing and accessible user interface, which is essential for providing a smooth user experience in the "Mueen" application.

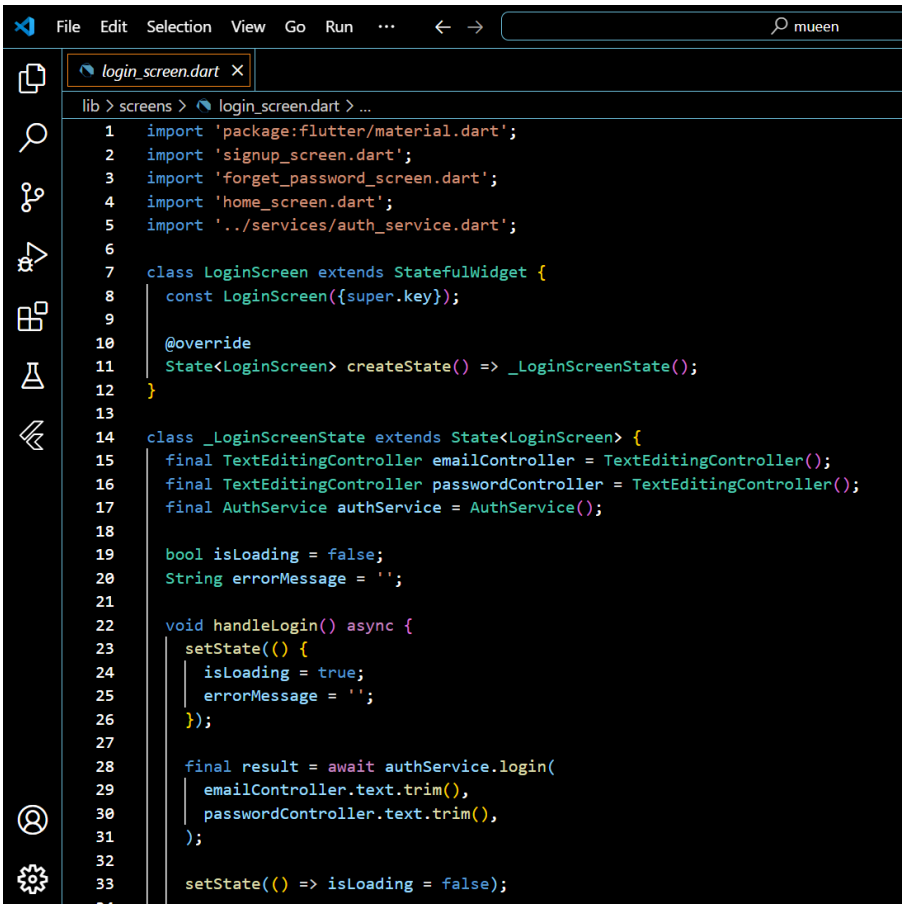
3.2 React Native
React Native is a widely-used framework that enables developers to create cross-platform apps using JavaScript, providing a native-like experience. It's well-suited for rapid development and testing, making it a strong choice for building the Mueen project's UI.

3.3 Figma
Figma is a top design tool used for creating interactive UI prototypes. It provides robust wireframing and prototyping features, which are crucial for planning and testing the app's interface before development.

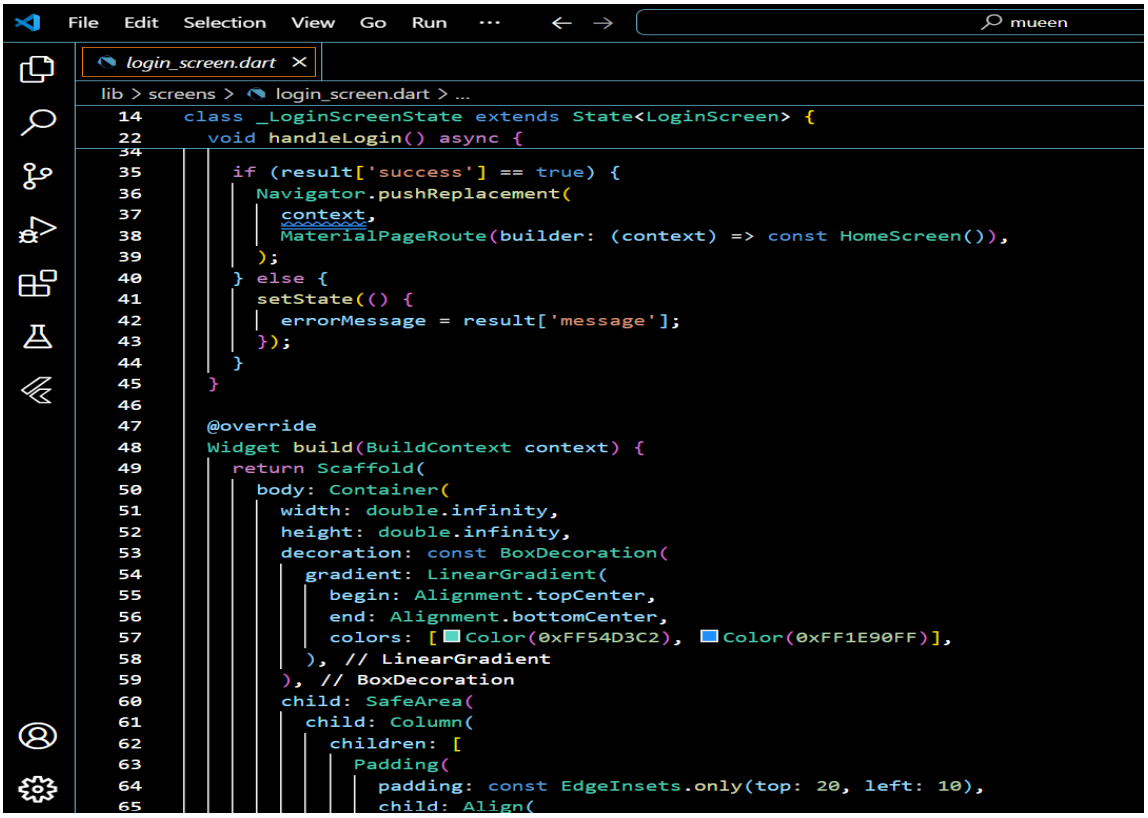
5.3 System Implementation

Login Function:

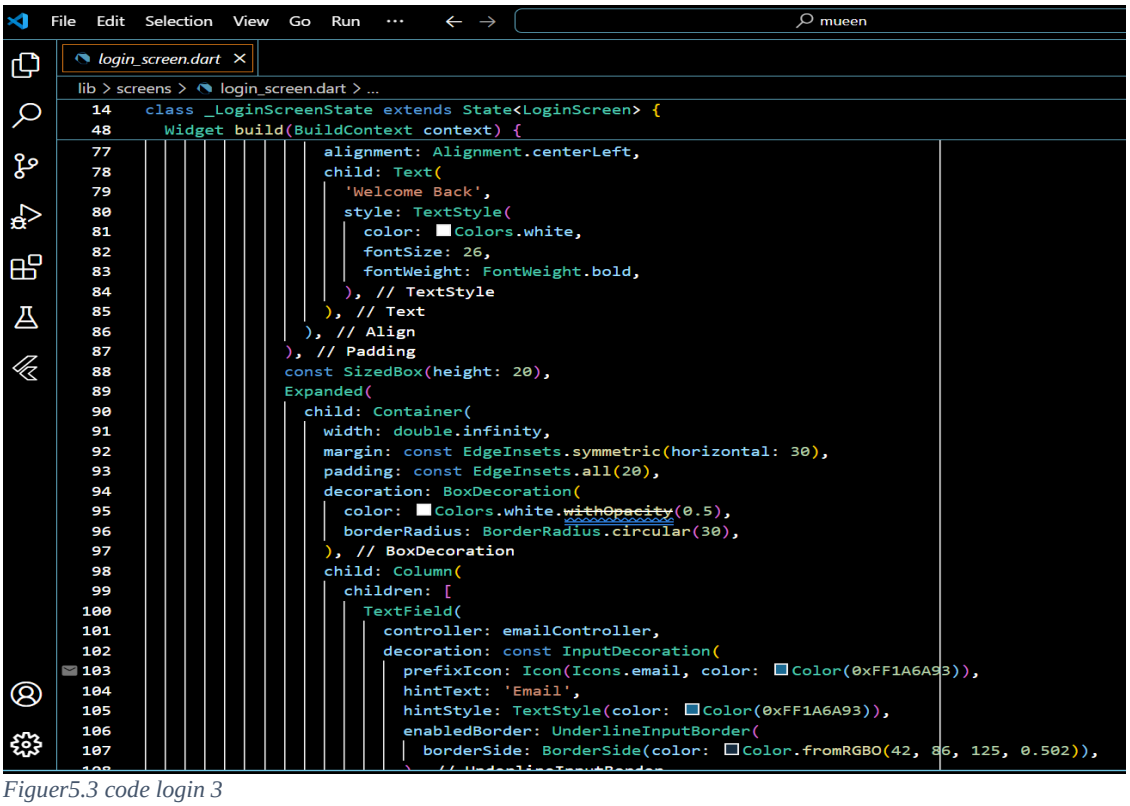
In the "Mueen" app, the login function enables users to either sign up or log in as a guest. After making their selection, they are directed to the translation page. If the user already has an account, they can log in using their registered credentials, ensuring both **new users** and **returning users** can access the app's features efficiently. This process streamlines the user experience by allowing easy access to essential functions such as gesture translation.



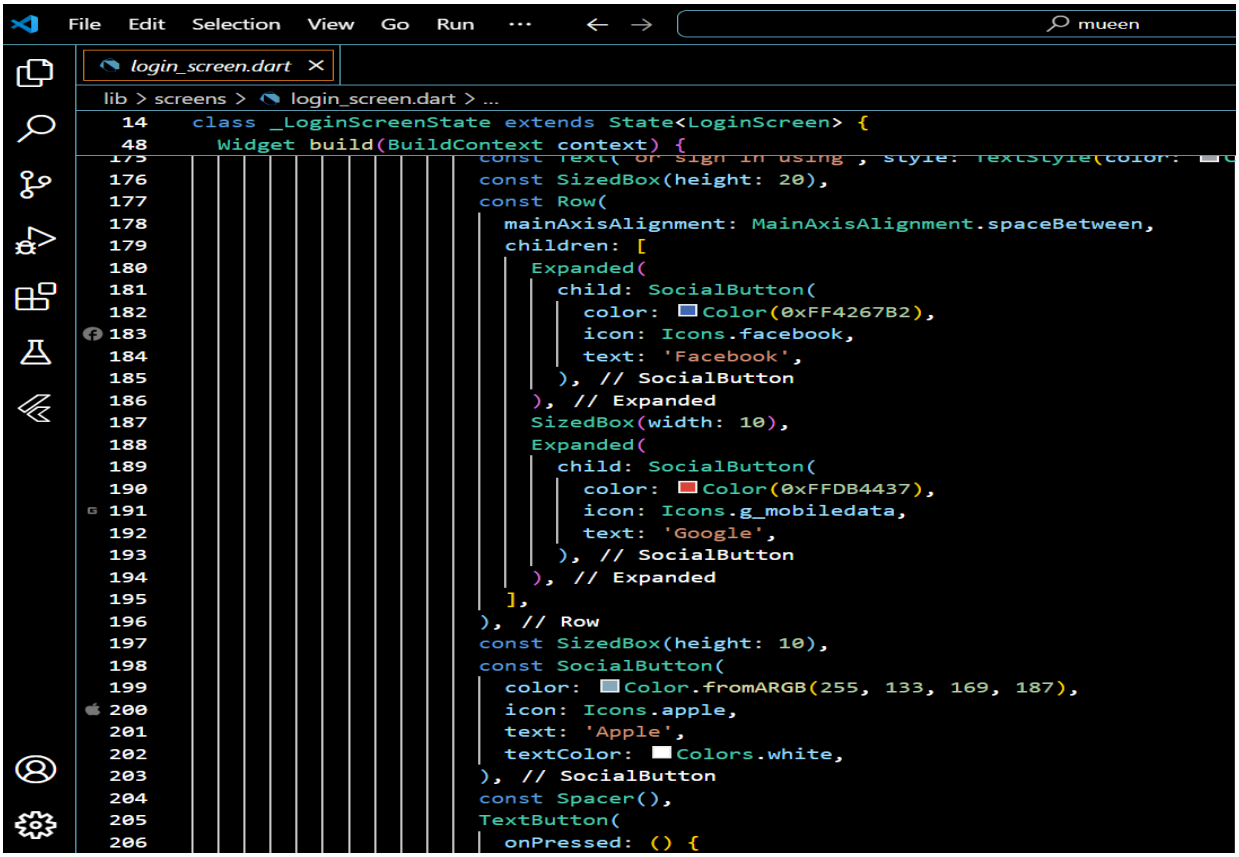
Figuer5.1 code login 1



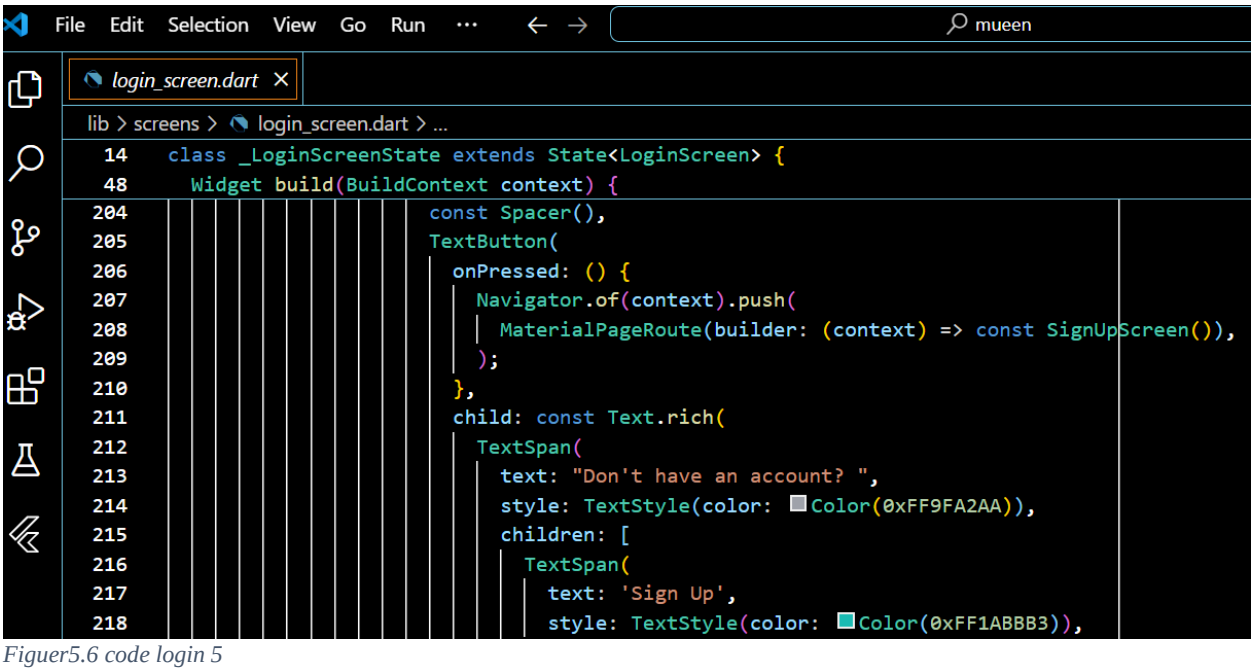
Figuer5.2 code login 2



Figuer5.3 code login 3



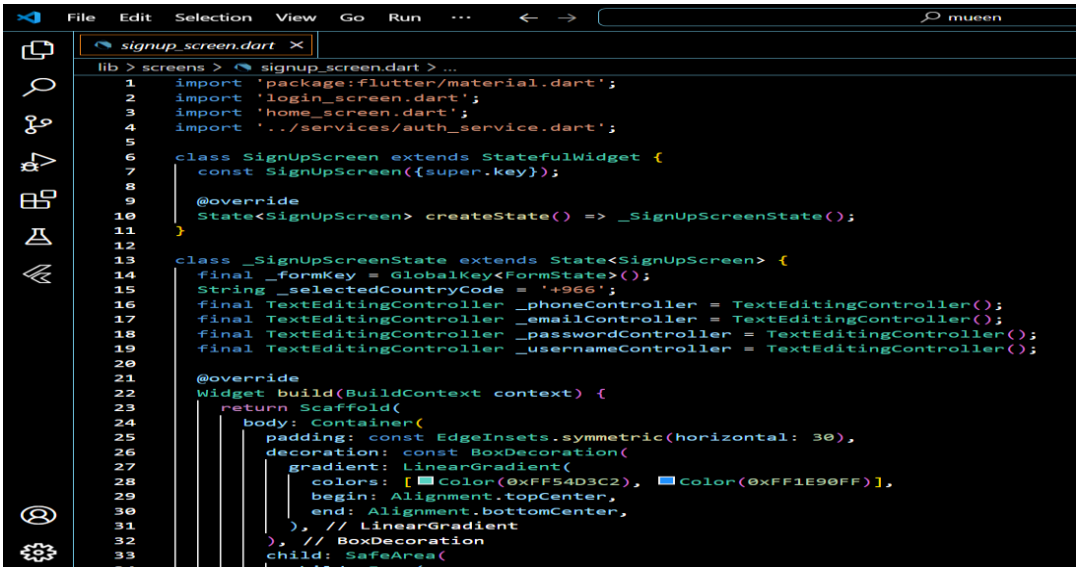
Figuer5.4code login 4



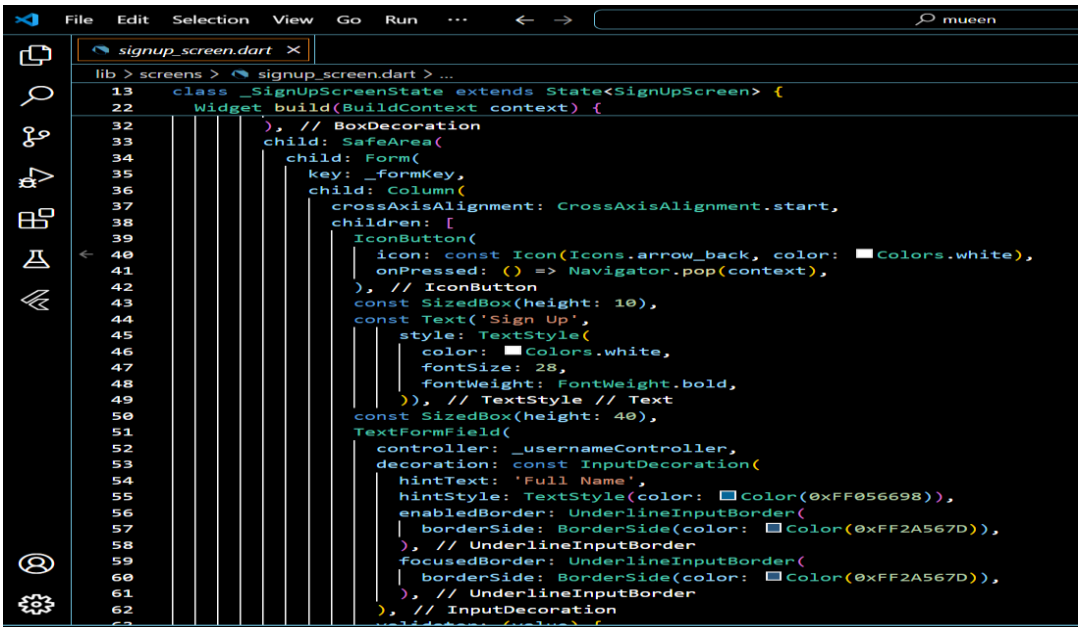
Figuer5.6 code login 5

Signup Function:

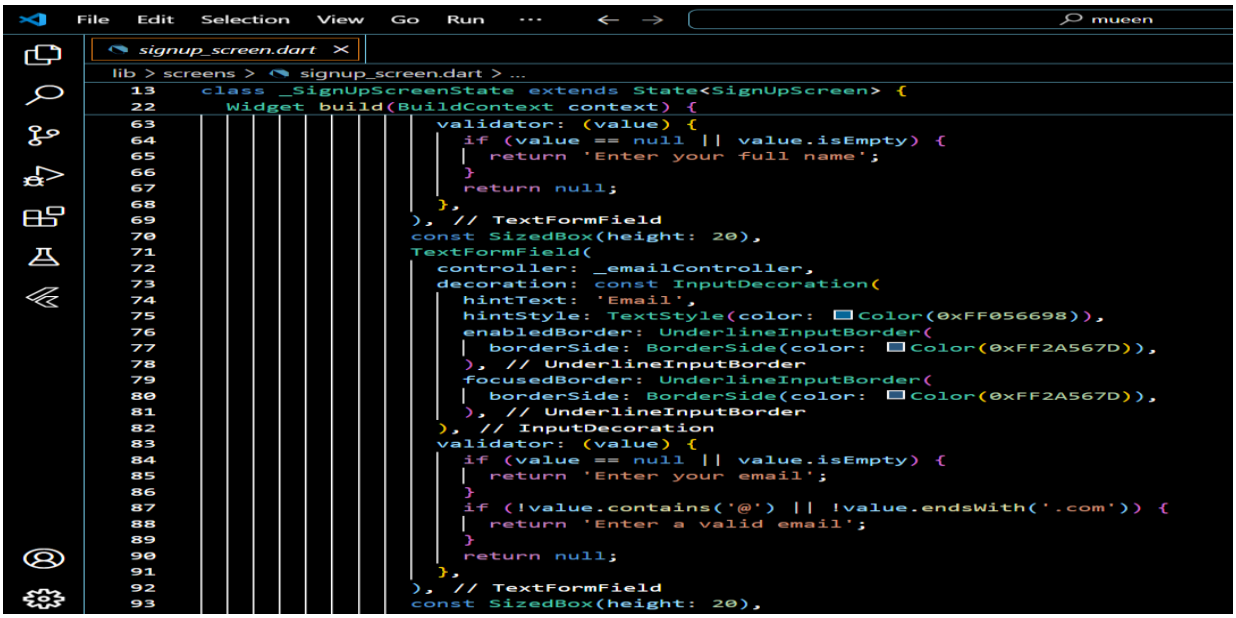
The signup function in the "Mueen" app provides users with the ability to create a new account. By entering necessary details such as name**,** email**,** phone number**,** and password, the system validates the data before allowing the user to register. After a successful signup, users gain access to the app's personalized features, such as saving translations and managing their profile.



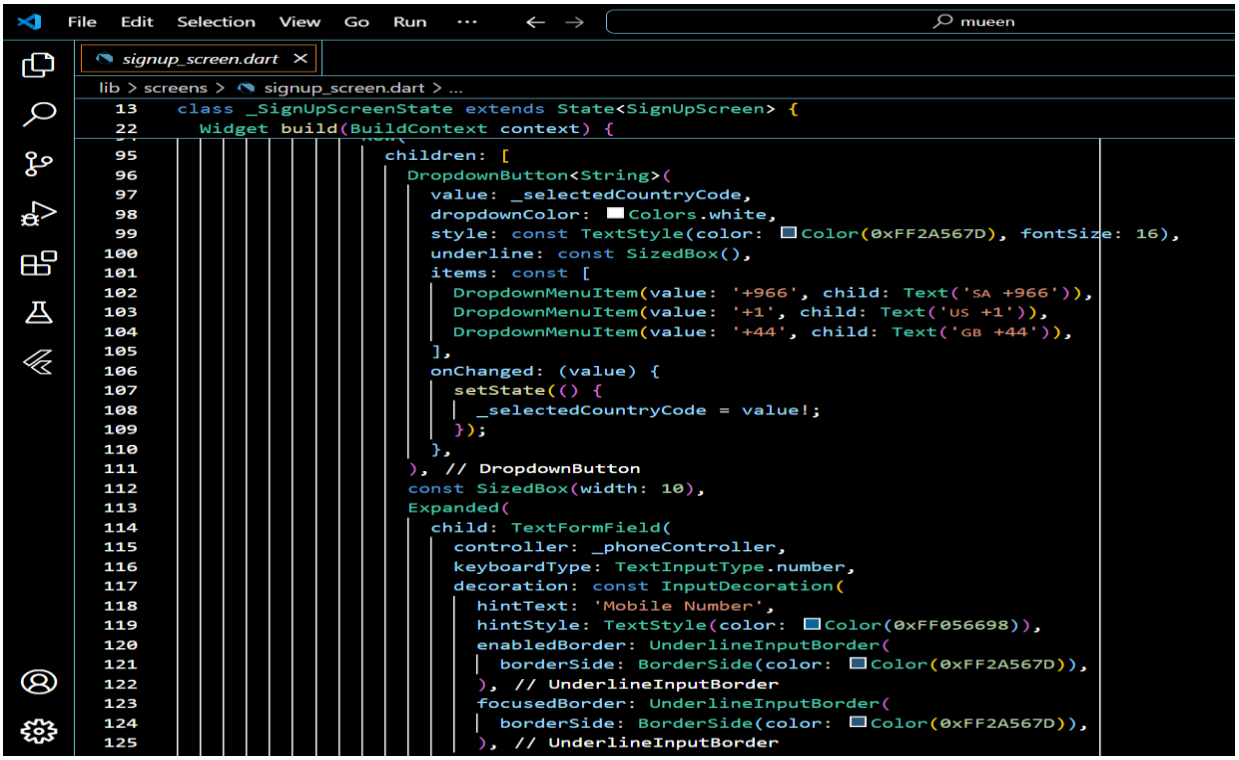
Figuer5.7 Signup code 1



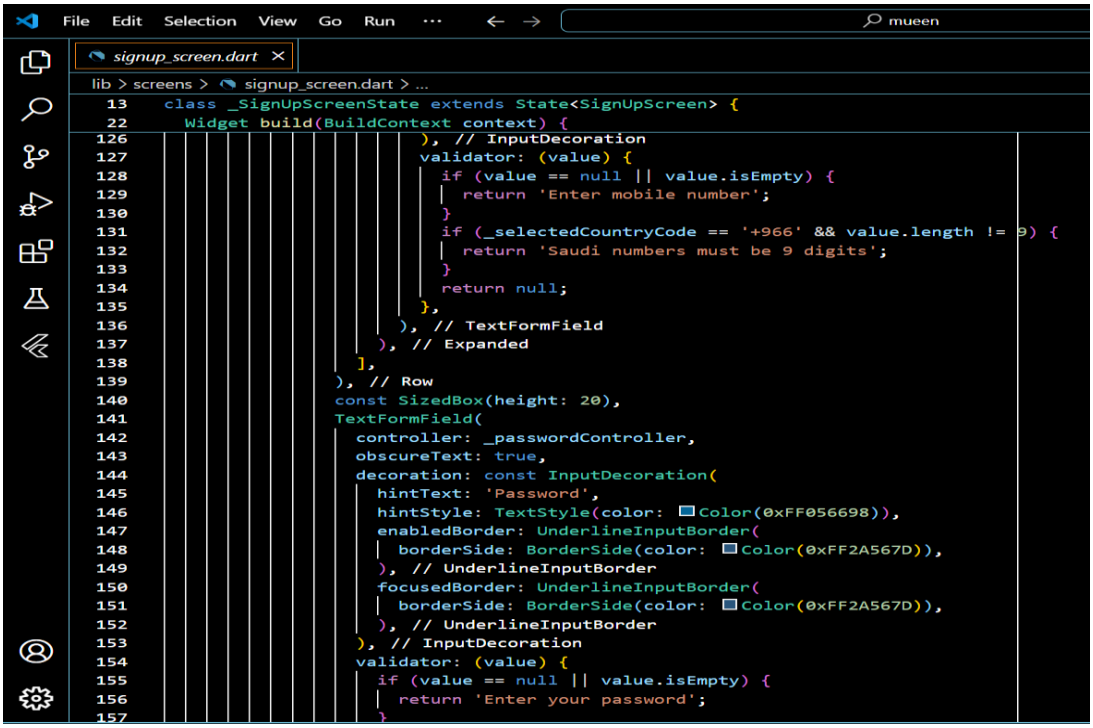
Figuer5.8 Signup code 2



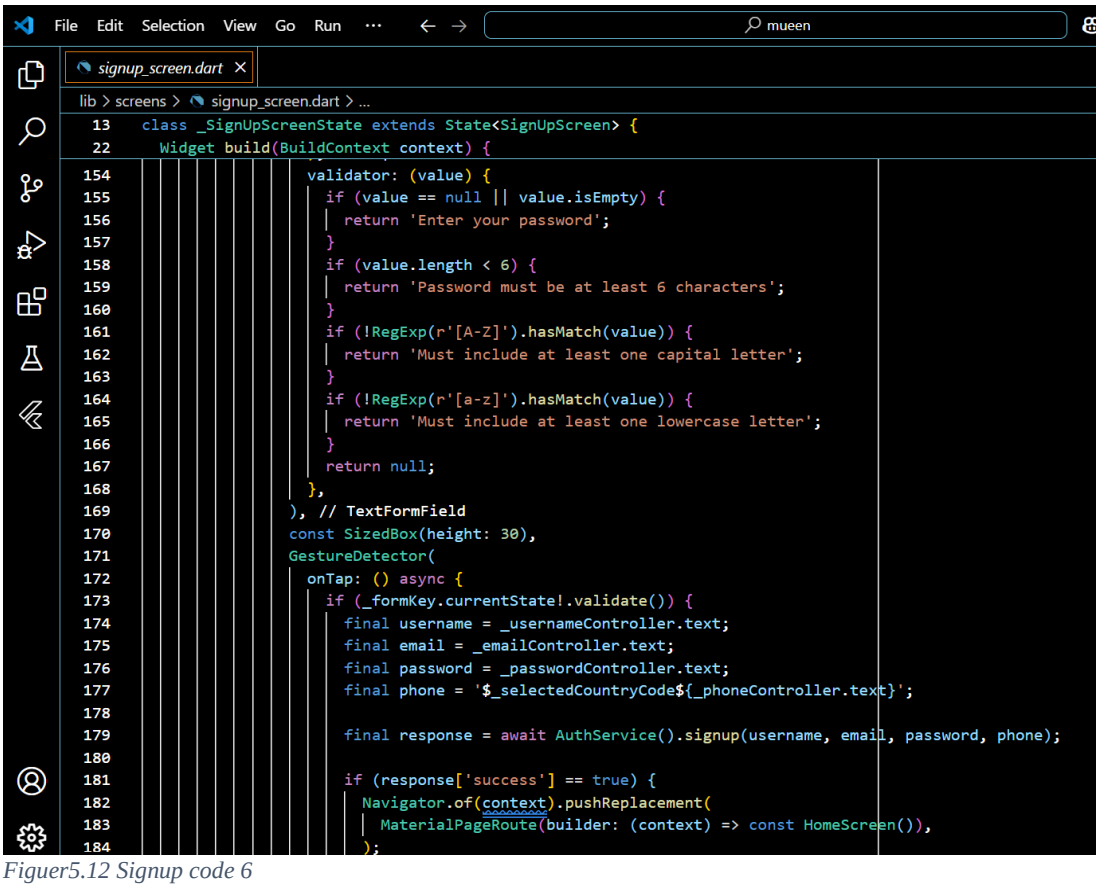
Figuer5.9 Signup code 3



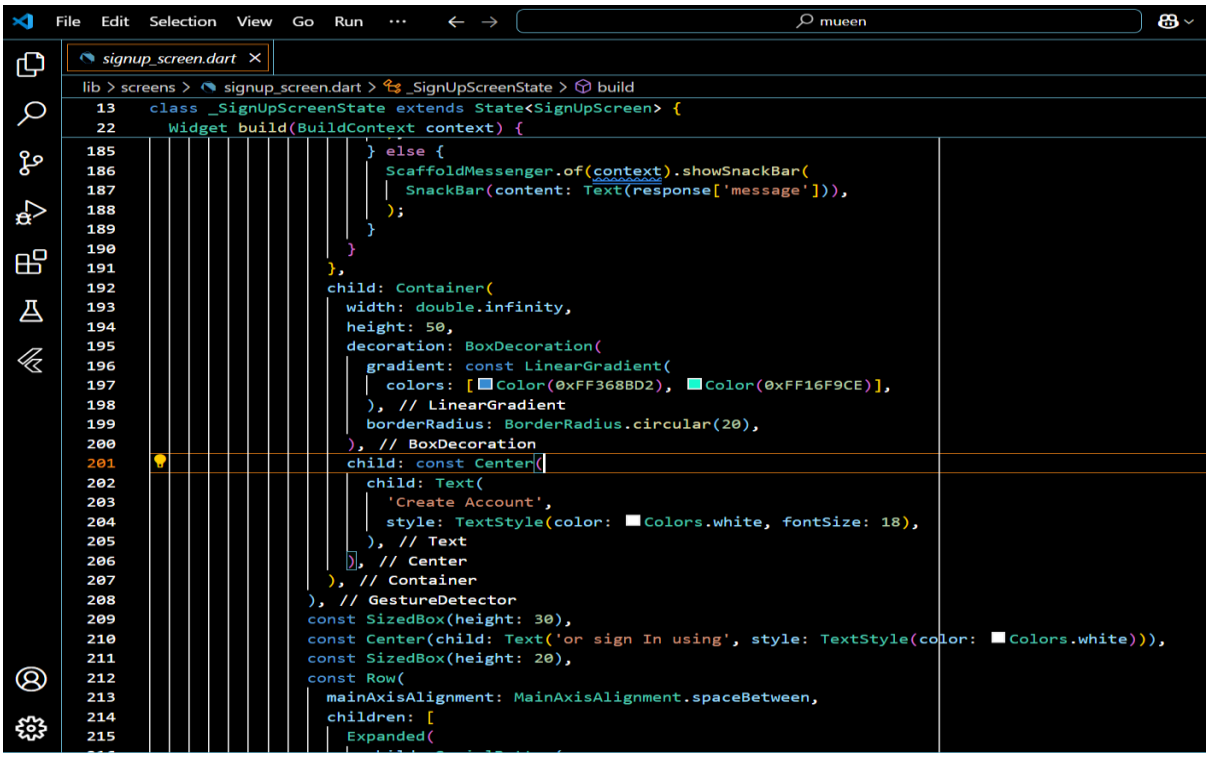
Figuer5.10 Signup code 4



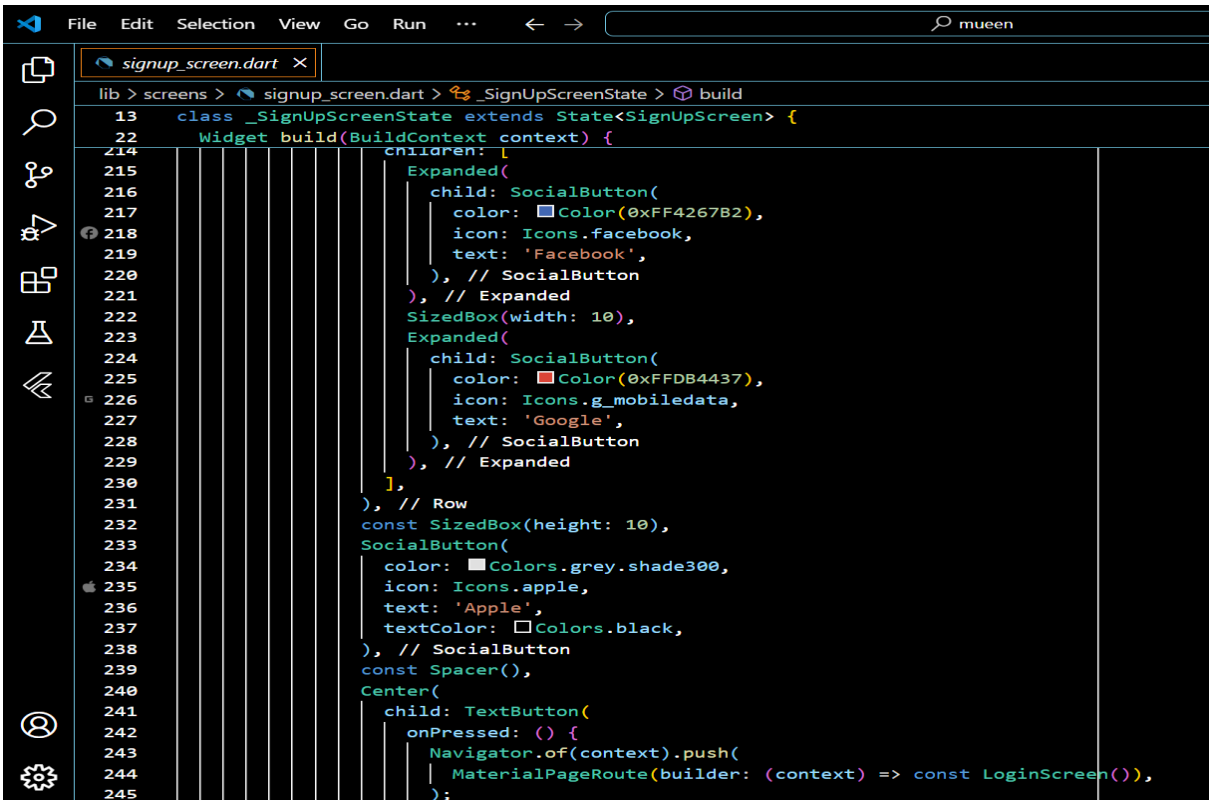
Figuer5.11 Signup code 5



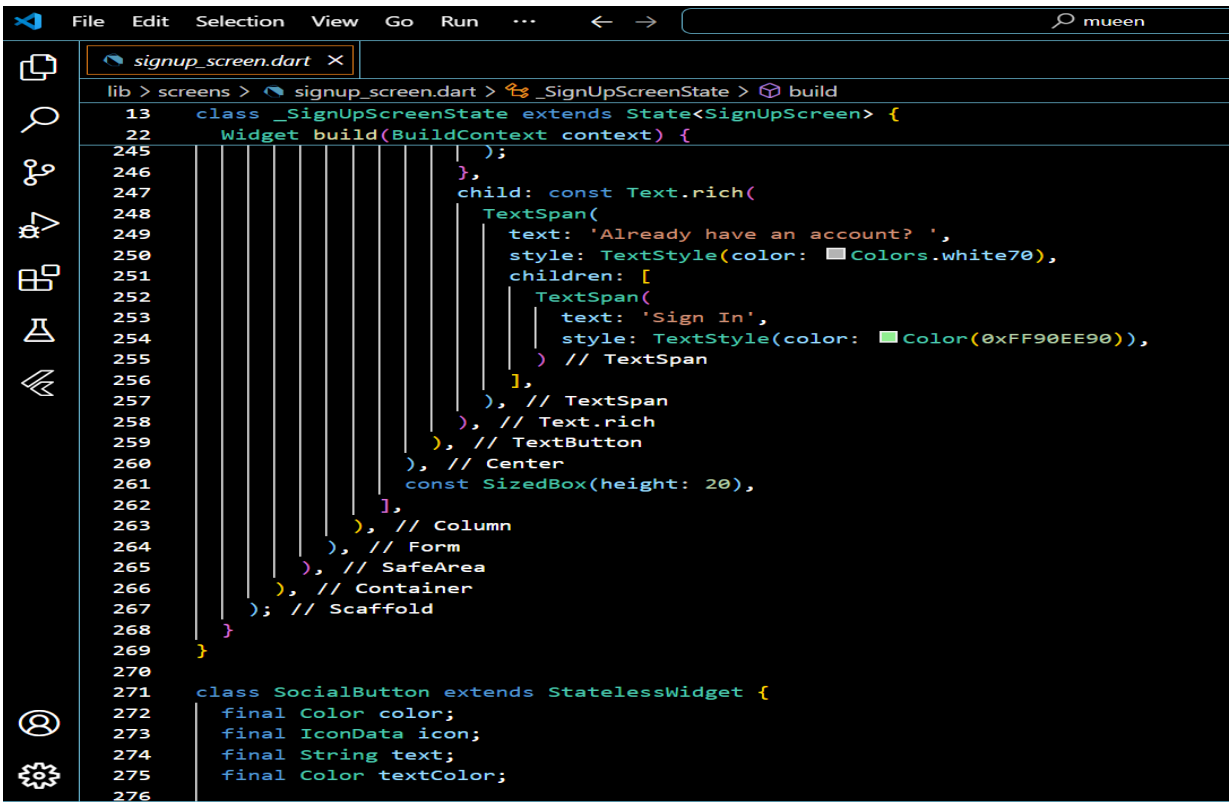
Figuer5.12 Signup code 6



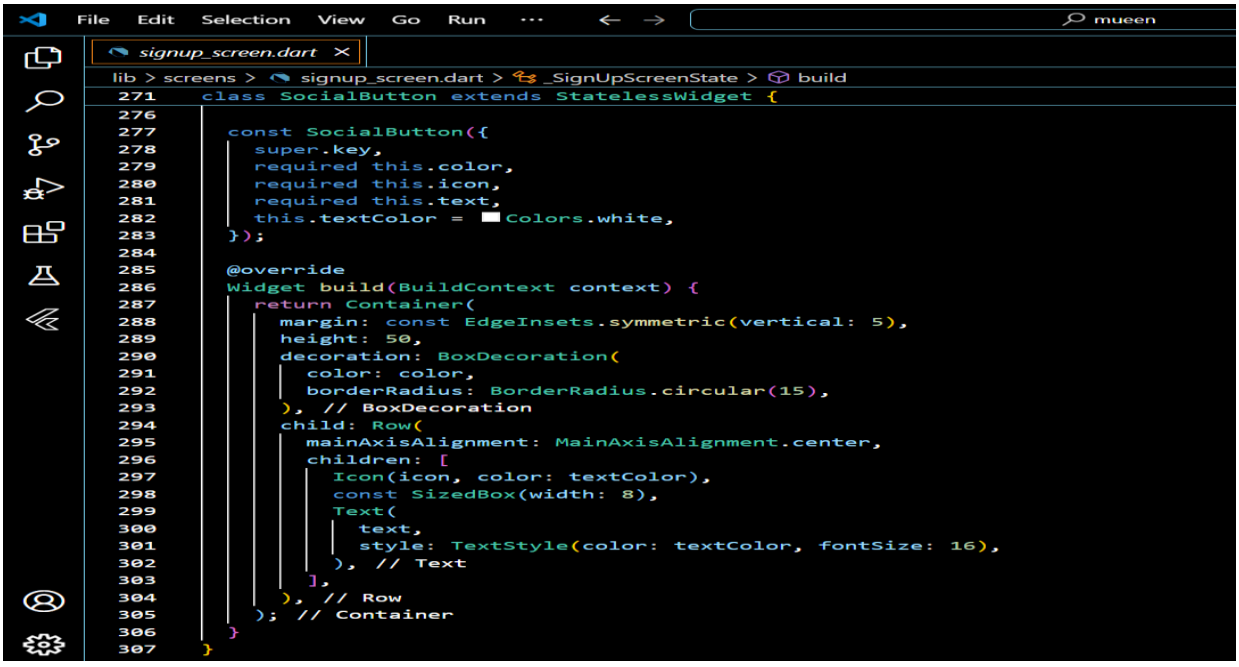
Figuer5.13 Signup code 7



Figuer5.14 Signup code 8



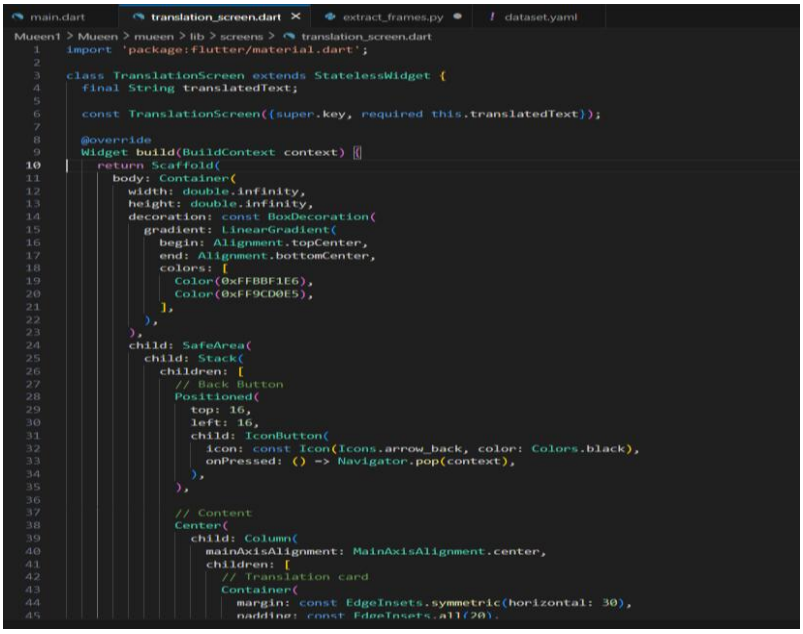
Figuer5.15 Signup code 9



Figuer5.16 Signup code 10

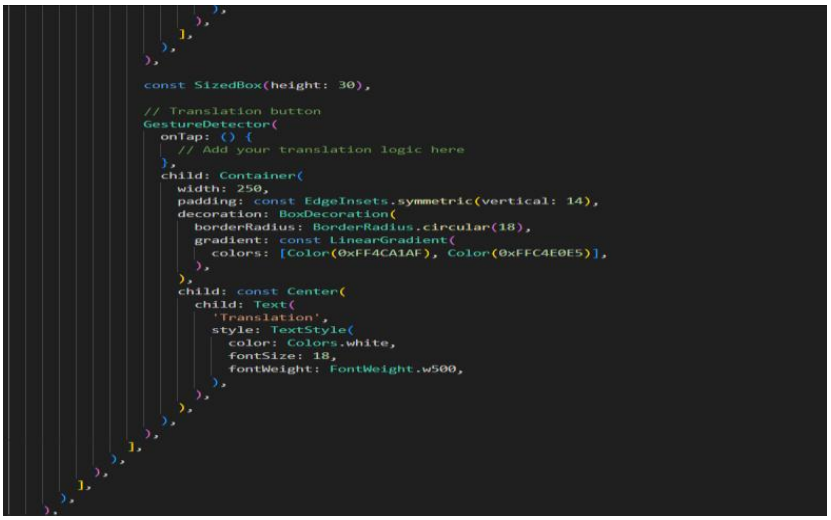
Translation Function:

The translation function in the "Mueen" app is designed to translate **Saudi Sign Language (SSL)** gestures into text or speech. The app uses a camera to capture SSL gestures, which are processed and converted in real-time. This functionality provides a clear and accurate translation of sign language, helping users communicate effectively with non-signing individuals.



Figuer5.17 Translation screen code 1

Figuer5.18 Translation screen code 2

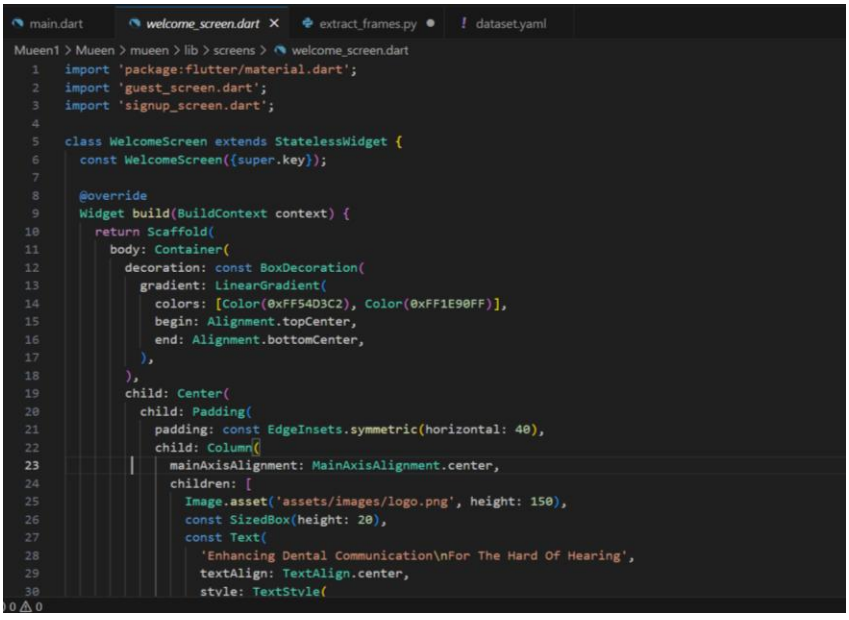


```
                    ),
                  ),
                ],
              ),
            ),

            const SizedBox(height: 30),

            // Translation button
            GestureDetector(
              onTap: () {
                // Add your translation logic here
              },
              child: Container(
                width: 250,
                padding: const EdgeInsets.symmetric(vertical: 14),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(18),
                  gradient: const LinearGradient(
                    colors: [Color(0xFF4CA1AF), Color(0xFFC4E0E5)],
                  ),
                ),
                child: const Center(
                  child: Text(
                    'Translation',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 18,
                      fontWeight: FontWeight.w500,
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
),
```

Figuer5.19 Translation screen code 3

Welcome Screen Function

The Welcome Screen in the Mueen app is the first interface that appears when the user opens the application. It features an attractive design with a gradient background and the app logo to create a professional and welcoming first impression.
The screen includes two main options. The first is Sign Up, where new users can create their own accounts. The second is Guest, which allows users to access the app without signing up. This screen simplifies access and offers a flexible experience that suits all types of users.



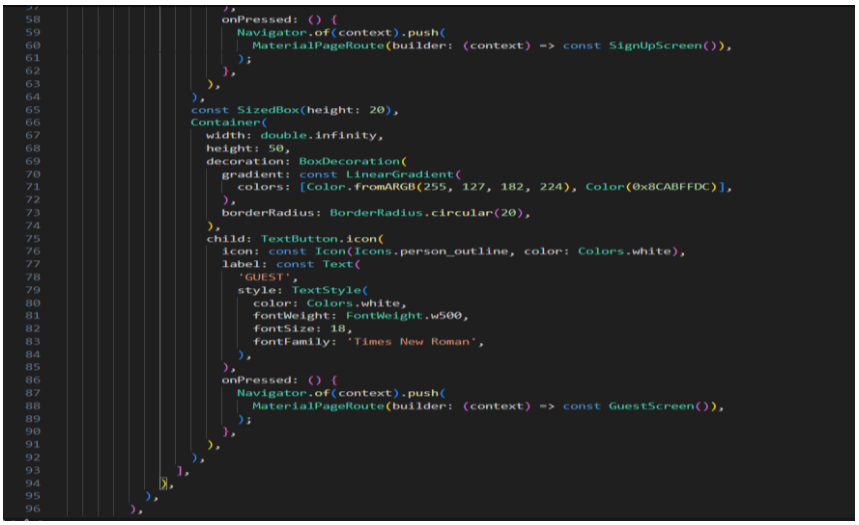
main.dart | welcome_screen.dart | extract_frames.py | dataset.yaml

Mueen1 > Mueen > mueen > lib > screens > welcome_screen.dart

```
1  import 'package:flutter/material.dart';
2  import 'guest_screen.dart';
3  import 'signup_screen.dart';
4
5  class WelcomeScreen extends StatelessWidget {
6    const WelcomeScreen({super.key});
7
8    @override
9    Widget build(BuildContext context) {
10     return Scaffold(
11       body: Container(
12         decoration: const BoxDecoration(
13           gradient: LinearGradient(
14             colors: [Color(0xFF54D3C2), Color(0xFF1E90FF)],
15             begin: Alignment.topCenter,
16             end: Alignment.bottomCenter,
17           ),
18         ),
19         child: Center(
20           child: Padding(
21             padding: const EdgeInsets.symmetric(horizontal: 40),
22             child: Column(
23               mainAxisAlignment: MainAxisAlignment.center,
24               children: [
25                 Image.asset('assets/images/logo.png', height: 150),
26                 const SizedBox(height: 20),
27                 const Text(
28                   'Enhancing Dental Communication\nFor The Hard Of Hearing',
29                   textAlign: TextAlign.center,
30                   style: TextStyle(
```

Figuer5.20 Welcome screen code1

Figuer5.21 Welcome screen code2



Figuer5.22 Welcome screen code3

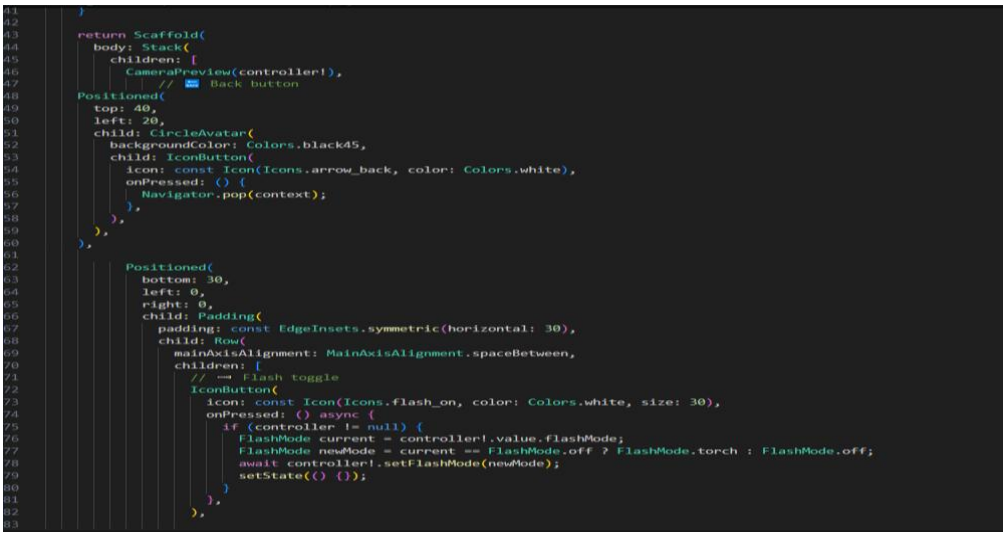
Camera Screen Function

The Camera Screen in the Mueen app allows users to interact directly with the device camera. It provides a live preview using the rear camera and gives users the ability to either record a video or capture a photo.

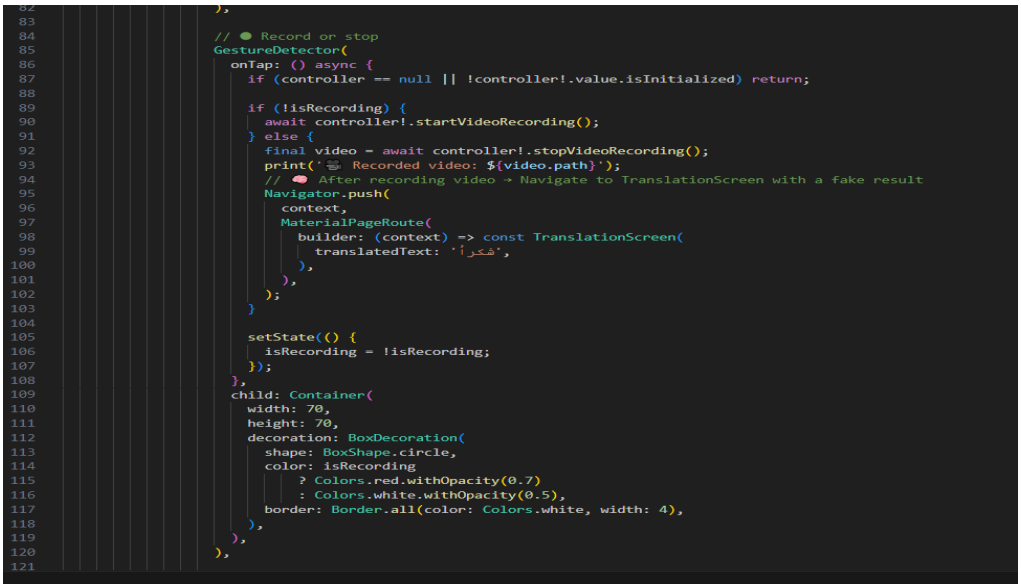
After recording or capturing, the app automatically navigates to the Translation Screen with a sample translated result, creating a smooth transition between capturing input and viewing translation.

The screen includes a simple interface with intuitive controls such as a record button, a capture button, a flash toggle, and a back button for easy navigation. This feature is essential in enabling real-time interaction with sign language and supports the app's goal of improving communication for the hard of hearing.

Figuer5.23 Camera screen code1

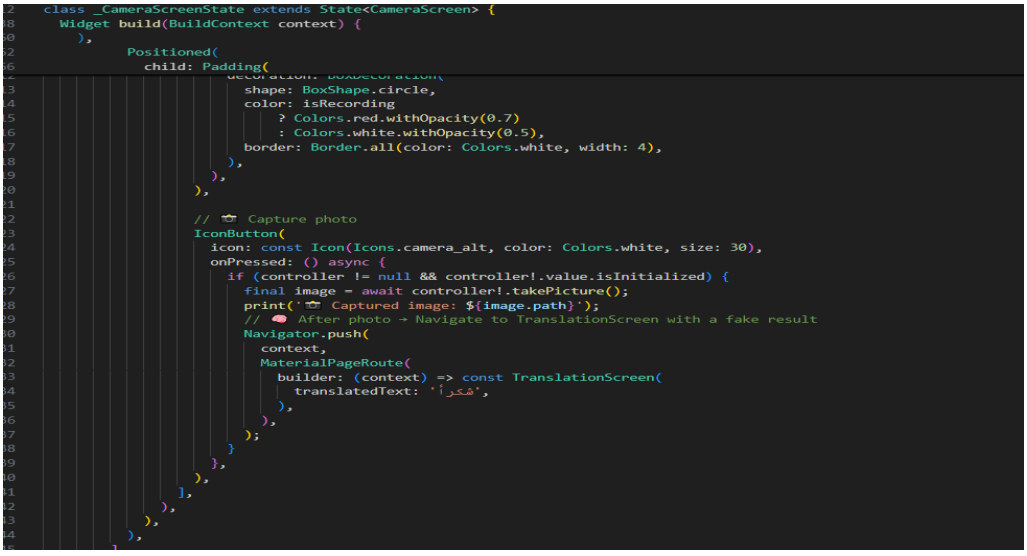


Figuer5.24 Camera screen code2



```
),

// ● Record or stop
GestureDetector(
  onTap: () async {
    if (controller == null || !controller!.value.isInitialized) return;

    if (!isRecording) {
      await controller!.startVideoRecording();
    } else {
      final video = await controller!.stopVideoRecording();
      print('📸 Recorded video: ${video.path}');
      // 🧠 After recording video → Navigate to TranslationScreen with a fake result
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const TranslationScreen(
            translatedText: 'شكراً',
          ),
        ),
      );
    }

    setState(() {
      isRecording = !isRecording;
    });
  },
  child: Container(
    width: 70,
    height: 70,
    decoration: BoxDecoration(
      shape: BoxShape.circle,
      color: isRecording
          ? Colors.red.withOpacity(0.7)
          : Colors.white.withOpacity(0.5),
      border: Border.all(color: Colors.white, width: 4),
    ),
  ),
),
```

Figuer5.25 Camera screen code3

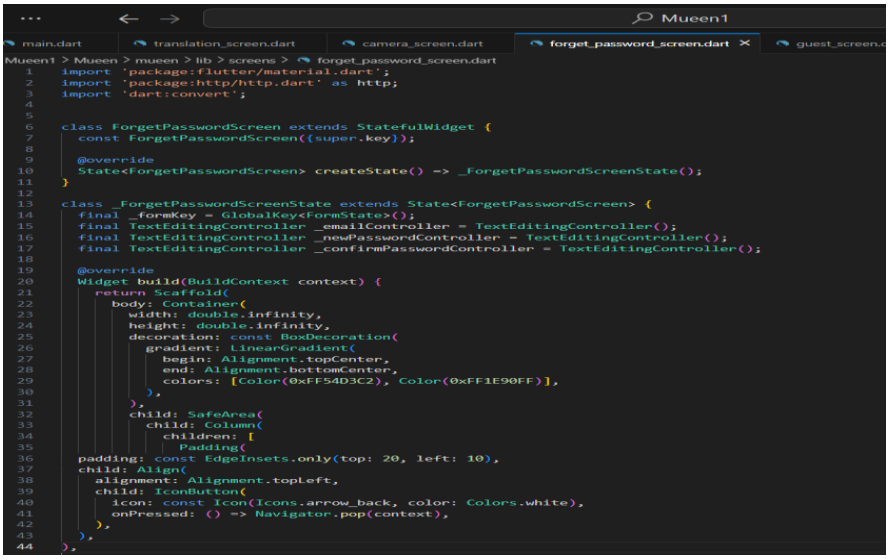


```
class _CameraScreenState extends State<CameraScreen> {
  Widget build(BuildContext context) {
    ),
          Positioned(
            child: Padding(
                  shape: BoxShape.circle,
                  color: isRecording
                      ? Colors.red.withOpacity(0.7)
                      : Colors.white.withOpacity(0.5),
                  border: Border.all(color: Colors.white, width: 4),
                ),
              ),
            ),

            // 📸 Capture photo
            IconButton(
              icon: const Icon(Icons.camera_alt, color: Colors.white, size: 30),
              onPressed: () async {
                if (controller != null && controller!.value.isInitialized) {
                  final image = await controller!.takePicture();
                  print('📸 Captured image: ${image.path}');
                  // 🧠 After photo → Navigate to TranslationScreen with a fake result
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => const TranslationScreen(
                        translatedText: 'شكراً',
                      ),
                    ),
                  );
                }
              },
            ),
          ],
        ),
      ),
    ),
```

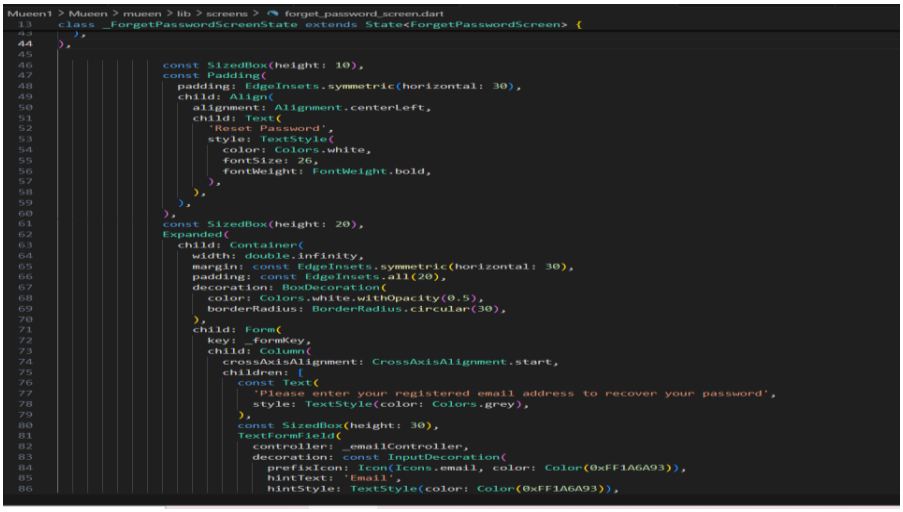
Figuer5.26 Camera screen code4

Forget Password Screen Function

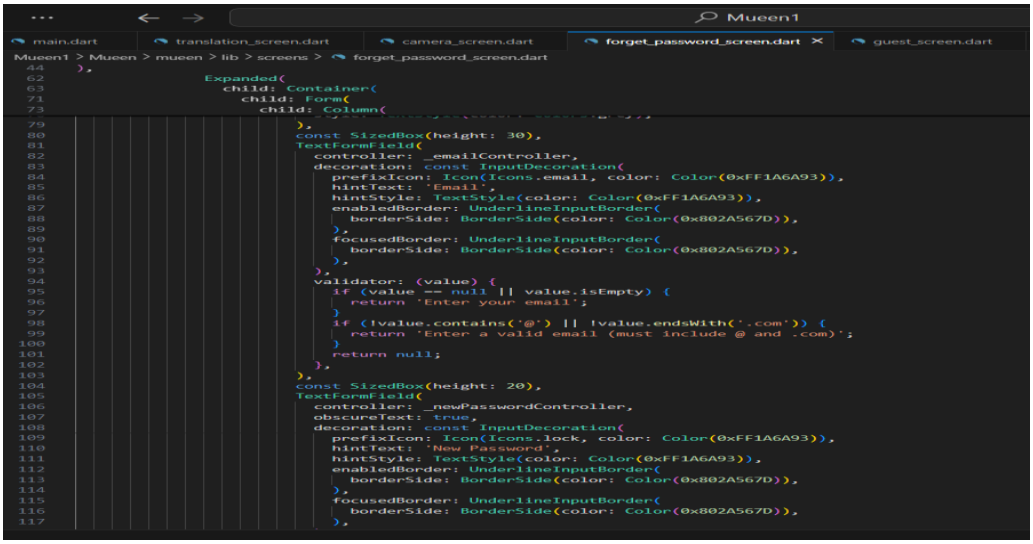
The Forget Password screen in the application provides users with a simple and secure way to reset their password in case they forget it. The interface includes input fields for the user's registered email address, a new password, and confirmation of the new password. The form is protected with validation rules to ensure that the email is in the correct format and that the new password meets the required strength, including having both uppercase and lowercase letters. Once the form is submitted, the app sends the data to the backend server using an HTTP POST request to update the password. Feedback is given to the user through clear success or error messages, and if the reset is successful, the user is redirected to the login screen. The design uses a smooth color gradient and intuitive layout to enhance usability and provide comfortable user experience.



Figuer5.27 Forget password code1



Figuer5.28 Forget password code2




```
),
Expanded(
  child: Container(
    child: Form(
      child: Column(
        ),
        const SizedBox(height: 30),
        TextFormField(
          controller: _emailController,
          decoration: const InputDecoration(
            prefixIcon: Icon(Icons.email, color: Color(0xFF1A6A93)),
            hintText: 'Email',
            hintStyle: TextStyle(color: Color(0xFF1A6A93)),
            enabledBorder: UnderlineInputBorder(
              borderSide: BorderSide(color: Color(0x802A567D)),
            ),
            focusedBorder: UnderlineInputBorder(
              borderSide: BorderSide(color: Color(0x802A567D)),
            ),
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Enter your email';
            }
            if (!value.contains('@') || !value.endsWith('.com')) {
              return 'Enter a valid email (must include @ and .com)';
            }
            return null;
          },
        ),
        const SizedBox(height: 20),
        TextFormField(
          controller: _newPasswordController,
          obscureText: true,
          decoration: const InputDecoration(
            prefixIcon: Icon(Icons.lock, color: Color(0xFF1A6A93)),
            hintText: 'New Password',
            hintStyle: TextStyle(color: Color(0xFF1A6A93)),
            enabledBorder: UnderlineInputBorder(
              borderSide: BorderSide(color: Color(0x802A567D)),
            ),
            focusedBorder: UnderlineInputBorder(
              borderSide: BorderSide(color: Color(0x802A567D)),
            ),
```

Figuer5.29 Forget password code3

```
            borderSide: BorderSide(color: Color(0x802A567D)),
          ),
        ),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Enter your new password';
          }
          if (value.length < 6) {
            return 'Password must be at least 6 characters';
          }
          if (!RegExp(r'[A-Z]').hasMatch(value)) {
            return 'Must include at least one capital letter';
          }
          if (!RegExp(r'[a-z]').hasMatch(value)) {
            return 'Must include at least one lowercase letter';
          }
          return null;
        },
      ),
      const SizedBox(height: 20),
      TextFormField(
        controller: _confirmPasswordController,
        obscureText: true,
        decoration: const InputDecoration(
          prefixIcon: Icon(Icons.lock, color: Color(0xFF1A6A93)),
          hintText: 'Confirm Password',
          hintStyle: TextStyle(color: Color(0xFF1A6A93)),
          enabledBorder: UnderlineInputBorder(
            borderSide: BorderSide(color: Color(0x802A567D)),
          ),
          focusedBorder: UnderlineInputBorder(
            borderSide: BorderSide(color: Color(0x802A567D)),
          ),
        ),
        validator: (value) {
          if (value != _newPasswordController.text) {
            return 'Passwords do not match';
          }
          return null;
        }
```

```
                child: Column(
                  return 'Passwords do not match';
                }
                return null;
              },
            ),
            const SizedBox(height: 30),
            GestureDetector(
              onTap: () async {
    if (_formKey.currentState!.validate()) {
      final response = await http.post(
        Uri.parse("http://10.0.2.2/mueen_backend/reset_password.php"),
        headers: {'Content-Type': 'application/x-www-form-urlencoded'},
        body: {
          "email": _emailController.text,
          "password": _newPasswordController.text,
        },
      );

      final data = json.decode(response.body);

      if (data["success"] == true) {
        // إظهار رسالة نجاح
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text("✅ ${data["message"]}")),
        );

        // التوجيه إلى صفحة تسجيل الدخول
        Future.delayed(const Duration(seconds: 1), () {
          Navigator.pop(context); // Assuming login screen is in previous page
        });
      } else {
        // إظهار رسالة خطأ
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text("❌ ${data["message"]}")),
        );
      }
    }
  },

                child: Container(
```

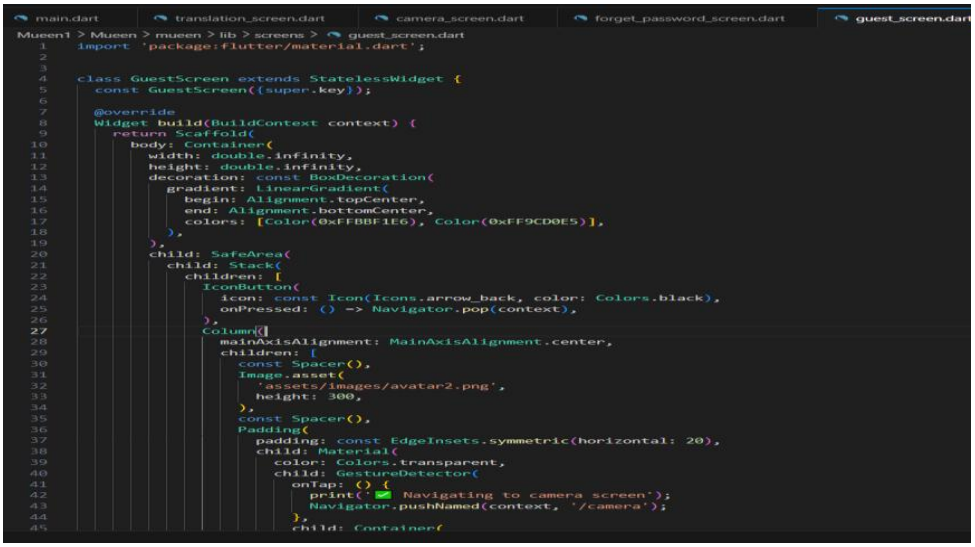
Figuer5.30 Forget password code4

```
    } else {
      // إظهار رسالة خطأ
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("❌ ${data["message"]}")),
      );
    }
  }
},

                child: Container(
                  width: double.infinity,
                  height: 50,
                  decoration: BoxDecoration(
                    gradient: const LinearGradient(
                      colors: [Color(0xFF368BD2), Color(0xFF16F9CE)],
                    ),
                    borderRadius: BorderRadius.circular(20),
                  ),
                  child: const Center(
                    child: Text(
                      'Reset Password',
                      style: TextStyle(color: Colors.white, fontSize: 18),
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
  }
}
```

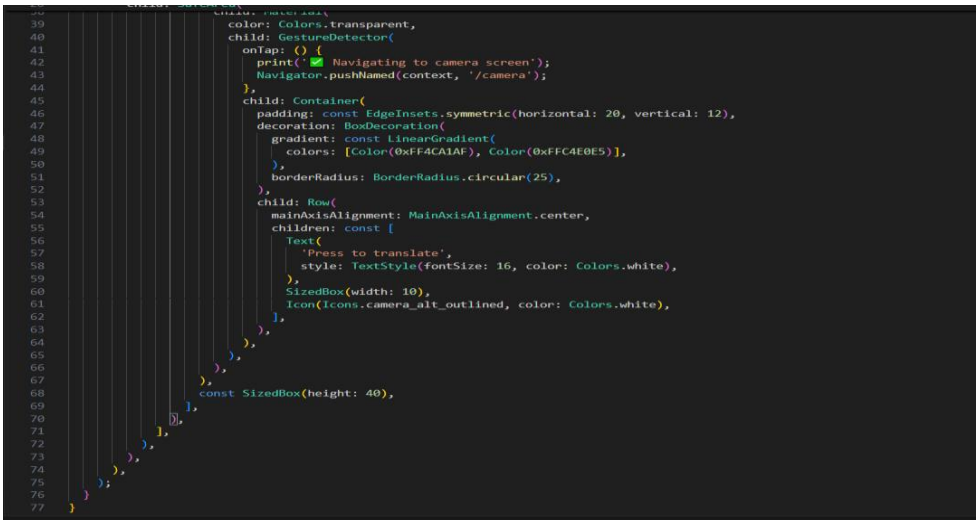
Figuer5.31 Forget password code5

The Guest Screen Function

The Guest Screen provides a welcoming interface for users who are accessing the app as guests. It features a simple and clean design with a soft gradient background and a central avatar image to create a friendly user experience. At the bottom of the screen, there is a prominent button labeled “Press to translate” that includes a camera icon, inviting the user to begin the translation process. When this button is pressed, the app navigates to the camera screen, allowing the guest to use the camera for sign language translation or other visual input. The screen also includes a back button at the top and left corner for easy navigation. Overall, the design is intuitive and visually appealing, ensuring accessibility for all users.



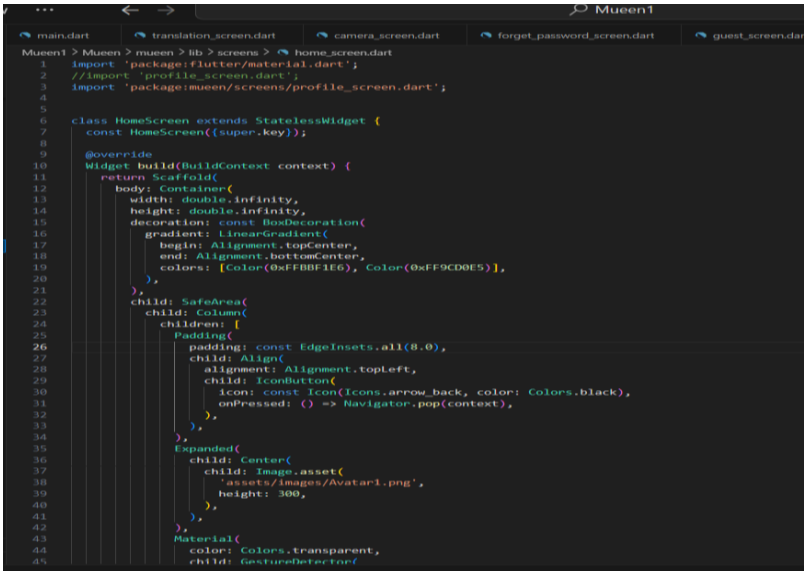
Figuer5.32 Guest screen code 1



Figuer5.33 Guest screen code 2

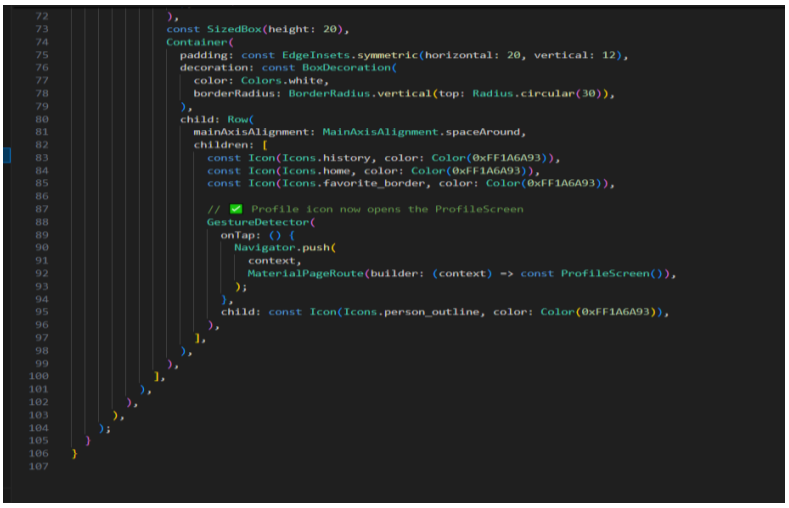
The Home Screen

The Home Screen serves as the central hub of the application, offering users a smooth and user-friendly interface. The design features a calming gradient background and a centered avatar image that creates a welcoming feel. Users are presented with a prominent button labeled "Press to translate," which directs them to the camera screen for initiating the translation process. At the bottom of the screen, a navigation bar is included with icons for history, home, favorites, and profile. Tapping the profile icon leads the user to the Profile Screen, allowing access to personal settings or user information. The interface is designed to be both functional and visually engaging, ensuring intuitive navigation and ease of use.



Figuer5.34 Home screen code 1

Figuer5.35 Home screen code 2

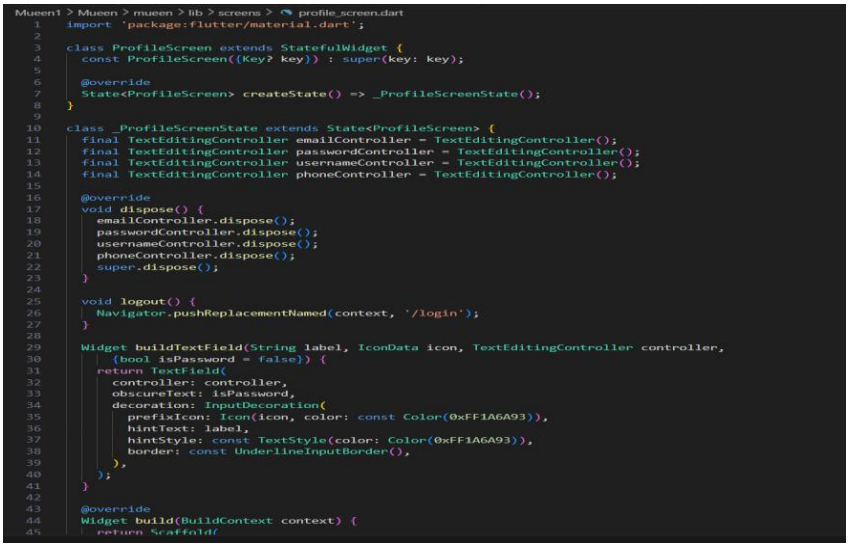


```
            ),
            const SizedBox(height: 20),
            Container(
              padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
              decoration: const BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.vertical(top: Radius.circular(30)),
              ),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: [
                  const Icon(Icons.history, color: Color(0xFF1A6A93)),
                  const Icon(Icons.home, color: Color(0xFF1A6A93)),
                  const Icon(Icons.favorite_border, color: Color(0xFF1A6A93)),

                  // ✅ Profile icon now opens the ProfileScreen
                  GestureDetector(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const ProfileScreen()),
                      );
                    },
                    child: const Icon(Icons.person_outline, color: Color(0xFF1A6A93)),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    ),
  );
  }
}
```

Figuer5.36 Home screen code 3

The Profile Screen Function

The Profile Screen provides users with access to their personal settings in an intuitive and visually cohesive layout. At the top, the interface displays a "Settings" title and a "Personal Data" section, where users can view or edit their email, password, username, and phone number using neatly designed input fields. Below that, users are presented with several quick-access options such as language settings, feedback, sharing the app, and rating the app, each represented by a clear icon and label. A stylish "Log Out" button allows users to exit their session and return to the login screen. The bottom navigation bar maintains visual consistency with the rest of the app and allows users to navigate between key sections like home, history, favorites, and profile. The overall design is user-friendly and built with gradient colors that match the app’s aesthetic theme.

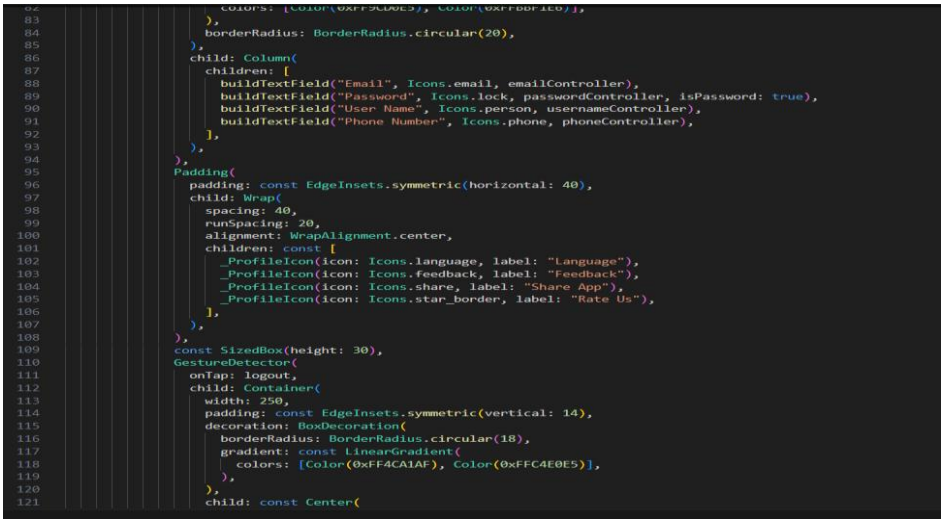


```
Mueen1 > Mueen > mueen > lib > screens > profile_screen.dart
import 'package:flutter/material.dart';

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

  @override
  State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController usernameController = TextEditingController();
  final TextEditingController phoneController = TextEditingController();

  @override
  void dispose() {
    emailController.dispose();
    passwordController.dispose();
    usernameController.dispose();
    phoneController.dispose();
    super.dispose();
  }

  void logout() {
    Navigator.pushReplacementNamed(context, '/login');
  }

  Widget buildTextField(String label, IconData icon, TextEditingController controller,
      {bool isPassword = false}) {
    return TextField(
      controller: controller,
      obscureText: isPassword,
      decoration: InputDecoration(
        prefixIcon: Icon(icon, color: const Color(0xFF1A6A93)),
        hintText: label,
        hintStyle: const TextStyle(color: Color(0xFF1A6A93)),
        border: const UnderlineInputBorder(),
      ),
    );
  }

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

Figuer5.37 Profile screen code 1

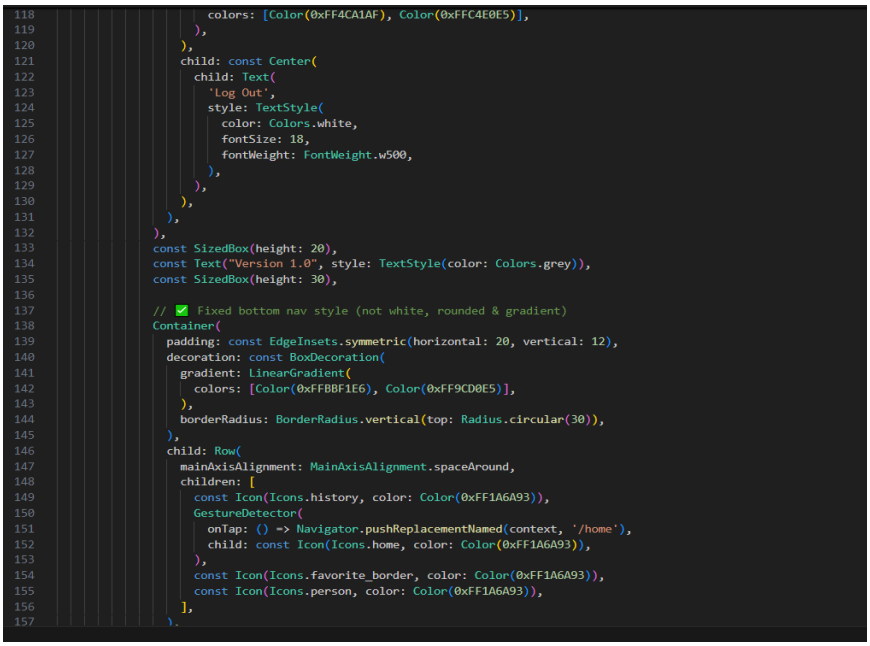
```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [Color(0xFFBBF1E6), Color(0xFF9CD0E5)],
          ),
        ),
        child: SafeArea(
          child: Column(
            children: [
              // 👇 Settings aligned to left
              Padding(
                padding: const EdgeInsets.fromLTRB(20, 30, 20, 0),
                child: Align(
                  alignment: Alignment.centerLeft,
                  child: const Text(
                    'Settings',
                    style: TextStyle(
                      fontSize: 28,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFF1A6A93),
                    ),
                  ),
                ),
              ),
              const SizedBox(height: 10),
              const Text(
                'Personal Data',
                style: TextStyle(fontSize: 20, color: Color(0xFF1A6A93)),
              ),
              Container(
                margin: const EdgeInsets.all(20),
                padding: const EdgeInsets.all(20),
                decoration: BoxDecoration(
                  gradient: const LinearGradient(
                    colors: [Color(0xFF9CD0E5), Color(0xFFBBF1E6)],
                  ),
```

Figuer5.38 Profile screen code 2



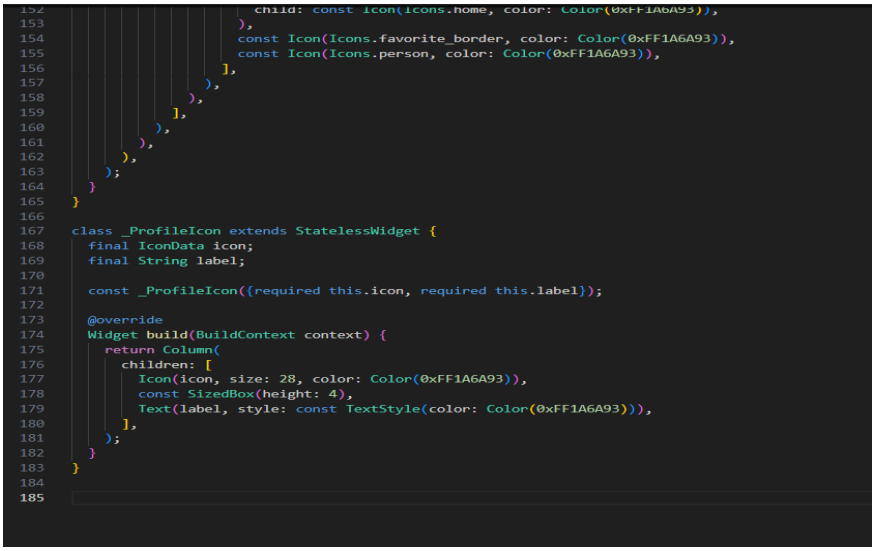
```
                  ),
                  borderRadius: BorderRadius.circular(20),
                ),
                child: Column(
                  children: [
                    buildTextField("Email", Icons.email, emailController),
                    buildTextField("Password", Icons.lock, passwordController, isPassword: true),
                    buildTextField("User Name", Icons.person, usernameController),
                    buildTextField("Phone Number", Icons.phone, phoneController),
                  ],
                ),
              ),
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 40),
                child: Wrap(
                  spacing: 40,
                  runSpacing: 20,
                  alignment: WrapAlignment.center,
                  children: const [
                    _ProfileIcon(icon: Icons.language, label: "Language"),
                    _ProfileIcon(icon: Icons.feedback, label: "Feedback"),
                    _ProfileIcon(icon: Icons.share, label: "Share App"),
                    _ProfileIcon(icon: Icons.star_border, label: "Rate Us"),
                  ],
                ),
              ),
              const SizedBox(height: 30),
              GestureDetector(
                onTap: logout,
                child: Container(
                  width: 250,
                  padding: const EdgeInsets.symmetric(vertical: 14),
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(18),
                    gradient: const LinearGradient(
                      colors: [Color(0xFF4CA1AF), Color(0xFFC4E0E5)],
                    ),
                  ),
                  child: const Center(
```

Figuer5.39 Profile screen code 3



```
                    colors: [Color(0xFF4CA1AF), Color(0xFFC4E0E5)],
                  ),
                ),
                child: const Center(
                  child: Text(
                    'Log Out',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 18,
                      fontWeight: FontWeight.w500,
                    ),
                  ),
                ),
              ),
            ),
          ),
          const SizedBox(height: 20),
          const Text("Version 1.0", style: TextStyle(color: Colors.grey)),
          const SizedBox(height: 30),

          // ✅ Fixed bottom nav style (not white, rounded & gradient)
          Container(
            padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
            decoration: const BoxDecoration(
              gradient: LinearGradient(
                colors: [Color(0xFFBBF1E6), Color(0xFF9CD0E5)],
              ),
              borderRadius: BorderRadius.vertical(top: Radius.circular(30)),
            ),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: [
                const Icon(Icons.history, color: Color(0xFF1A6A93)),
                GestureDetector(
                  onTap: () => Navigator.pushReplacementNamed(context, '/home'),
                  child: const Icon(Icons.home, color: Color(0xFF1A6A93)),
                ),
                const Icon(Icons.favorite_border, color: Color(0xFF1A6A93)),
                const Icon(Icons.person, color: Color(0xFF1A6A93)),
              ],
            ),
```

Figuer5.40 Profile screen code 4



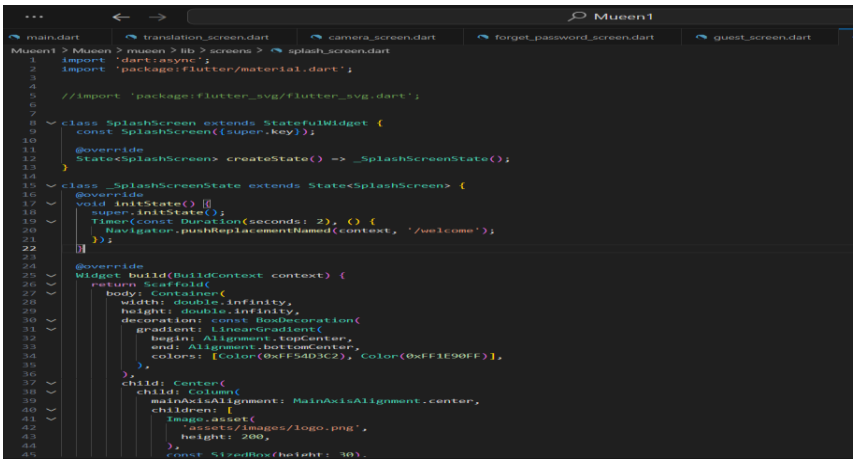
```
                  child: const Icon(Icons.home, color: Color(0xFF1A6A93)),
                ),
                const Icon(Icons.favorite_border, color: Color(0xFF1A6A93)),
                const Icon(Icons.person, color: Color(0xFF1A6A93)),
              ],
            ),
          ),
        ),
      ],
    ),
  );
  }
}

class _ProfileIcon extends StatelessWidget {
  final IconData icon;
  final String label;

  const _ProfileIcon({required this.icon, required this.label});

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Icon(icon, size: 28, color: Color(0xFF1A6A93)),
        const SizedBox(height: 4),
        Text(label, style: const TextStyle(color: Color(0xFF1A6A93))),
      ],
    );
  }
}

```

Figuer5.41 Profile screen code 5

The Splash Screen Function

The Splash Screen serves as the visual entry point to the Mueen application, designed to provide a smooth and engaging user experience during the app's initial loading phase. It features a full-screen gradient background with a modern color blend that reflects the app's aesthetic theme. At the center of the screen, the app's logo is prominently displayed, followed by the application name in both Arabic ("مُعين") and English ("Mueen") using large, clean typography to establish brand identity.

This screen appears briefly for a few seconds before automatically navigating to the Welcome Screen, giving users a seamless transition while the app prepares necessary resources. The design is minimalistic and user-friendly, avoiding unnecessary clutter while delivering a professional and visually appealing first impression. The overall purpose of the Splash Screen is to reinforce branding and enhance the perceived performance of the application by offering a polished startup experience.



Figuer5.42 The Splash Screen code 1

Figuer5.43 The Splash Screen code 2

5.4 Code Debugging and Troubleshooting

While we were building the *Mueen* app, we faced some problems and tried different ways to solve them:

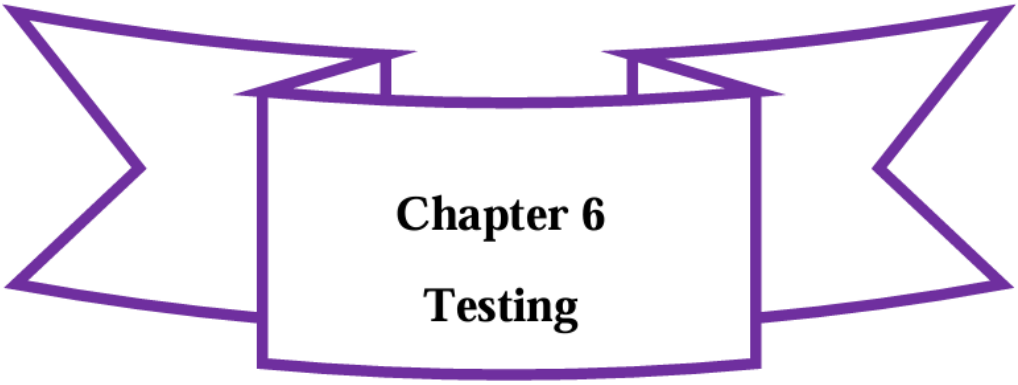
- One of the issues was with the **camera**. Sometimes the screen stayed black. After checking online, we found that we needed to make sure the camera was initialized correctly and that we had camera permissions in the Android settings.
- We also had **navigation issues**. For example, when we tried to go from the camera screen to the translation screen, it didn't work. We fixed this by using the right route and checking the screen constructor.
- In the **forms** (login, signup, and reset password), we had problems with validation. We added simple conditions like checking if the email has "@" and if the password is strong.
- Finally, when connecting to the **backend**, we couldn't reach the server using localhost. After some searching, we used the IP 10.0.2.2 for the emulator, and the problem was solved.

5.5 Packaging and documentation

Class Name	Description
SplashScreen	Show the logo at the start, then move to the welcome screen.
WelcomeScreen	Let's the user choose to sign up or continue as a guest.
LoginScreen	Allow users to log in using their email and password.
SignUpScreen	Used to create a new account with name, email, phone, and password.
ForgetPasswordScreen	Help users reset their password by entering their email and a new password.
HomeScreen	Main screen after login. It includes access to the camera and navigation bar.
GuestScreen	Simple home screen for users without an account.
CameraScreen	Use the camera to take videos or photos for translation.
TranslationScreen	Shows the result of the sign translation.
ProfileScreen	Let's the user update email, password, name, and phone.
_ProfileIcon	Used to display icons like language, feedback, etc. in the profile screen.
AuthService	Contains login and signup functions to connect with the backend.
SocialButton	A custom button used for Facebook, Google, or Apple login options.

5.6 Conclusion

The implementation of the "Mueen" app successfully transformed our design into a functional system that translates Saudi Sign Language in real time. We used deep learning tools, camera integration, and intuitive interfaces to support deaf patients in dental clinics. Despite some challenges, the final app provides a smooth and accessible user experience.



In this chapter, we go over all the testing we did to make sure the Mueen app works properly and gives a smooth user experience. Our team focused on five types of testing: **Acceptance**, **Integration**, **Usability**, **Unit**, and **System Testing**. Each one helped us check the app from a different angle — from backend connection to overall flow and how real users felt using it.

6.1 Acceptance Testing

Acceptance testing was done to check that the app's key features work the way users expect. We ran different test cases for the main screens like login, signup, password reset, guest access, profile editing, and logout. We also tested how the app handles invalid actions — like leaving fields empty or entering wrong inputs — just to make sure it doesn't crash or act unexpectedly.

Test Case ID	**Description**	**Steps**	**Expected Result**	**Actual Result**	**Status**
AT-1	Login with valid credentials	Open app → Enter correct email/password → Tap login.	User navigates to HomeScreen.	Successfully navigated to HomeScreen.	Pass
AT-2	Login with invalid credentials	Open app → Enter wrong email/password → Tap login.	Show error message.	Error message appeared: "Invalid email or password"	Pass

AT-3	Successful signup	Tap SignUp → Fill all fields correctly → Submit.	Account created, navigate to HomeScreen.	Account successfully created; navigated to HomeScreen.	Pass
AT-4	Signup with missing fields	Leave fields empty → Tap signup.	Show validation error.	If fields are left empty, specific prompts like 'Enter your name' will appear.	Pass
AT-5	Navigate as guest	Tap "Continue as Guest".	Navigate to guestScreen	Navigate to guestScreen.	Pass
AT-6	Password recovery request	Tap "Forgot Password?" → Enter email and new password → Submit.	Success message appears.	Success message appears: Password updated successfully	Pass
AT-7	Edit Profile successfully	Login → Go to Profile → Edit and save changes.	Changes saved and displayed.	Nothing happens	Failed
AT-8	Logout function works	Tap logout button.	Navigate to LoginScreen.	Navigate to LoginScreen.	Pass
AT-9	User attempts to reuse existing email during signup	Tap SignUp → Enter an email that already exists in the database → Submit.	Show error: "Email already registered" and prevent account creation.	Show error: "Email already registered" and prevent account creation.	Pass
AT-10	App handles invalid actions safely	Input wrong data → Navigate fast.	No crashes occur.	No crashes occur.	Pass

Table6.1 Application Functional Test Cases Summary

Feedback:
All the major flows passed except one issue with the Edit Profile screen — the changes didn't show even though the save button was used. Everything else worked correctly and as expected. This edit issue wasn't critical, but we flagged it for fixing later.

6.2 Integration Testing

This test was to make sure the frontend and backend were communicating properly. We mainly tested login and signup integration with the backend database, and we also tested the connection with the AI translation model.

Test Case ID	**Description**	**Steps**	**Expected Result**	**Actual Result**	**Status**
IT-1	Login backend integration	Submit correct login form.	User authenticated from backend.	Login authenticated via backend.	Pass
IT-2	Signup backend integration	Submit signup form.	User info saved in database.	New user entry created in database.	Pass
IT-3	Model integration for translation	Capture sign input.	A new translation should be generated and displayed for each newly captured sign	Only the first captured sign is translated correctly; all following signs repeat the first output.	(Pending)/ Failed

Table6.2 Integration Testing Results

Feedback:
Login and signup backend tests worked well. The data sent from the app was saved correctly and responses were received as expected. The only issue was with the translation model — it only gave a proper result for the first captured sign. After that, it repeated the same output no matter what the user did. We know this is a key part of the app, so fixing it is a priority.

6.3 Usability Testing

Usability testing was a key part of evaluating the Mueen app, especially since it was developed for a specific group — individuals who are deaf or have speech disabilities. To make sure the app truly meets their needs, we conducted testing sessions with three users from this community. Each user interacted with the app independently, without any prior guidance or explanation.

The goal was to observe how smoothly they could complete core tasks such as signing up, logging in, accessing the app as a guest, editing their profile, logging out, navigating to the translation screen, and using the translation feature. Their feedback helped us better understand whether the app's design, navigation, and overall experience were intuitive and accessible for real users.

User A - Task Results

Task	Success	Time Taken	Notes
Signup	☑	30s	Completed easily, no confusion.
Login	☑	15s	Understood fields clearly.
Guest access	☑	2s	Button was clear.
Edit profile	☑	20s	Editing fields felt clear and natural.
Logout	☑	3s	Logout icon intuitive.
Navigate to Translation	☑	8s	Easy to access from camera.
Use translation UI	☑	1s	Understood function.
Rate app usability	—	—	**5/5** – Clear flow, smooth experience, no issues.

Table 6.3: Usability Testing Results – User A Task Performance

Comment: User A moved quickly and confidently through all screens. They didn't ask for help once and rated the experience a full 5 out of 5. Their feedback confirmed that the app layout and flow made sense.

User B - Task Results

Task	Success	Time Taken	Notes
Signup	☑	43s	Clear process, no issues.
Login	☑	18s	Understood quickly.
Guest access	☑	1s	Navigation was easy.
Edit profile	☑	18s	All clear
Logout	☑	5s	Easy to find.
Navigate to Translation	☑	10s	Found via camera.
Use translation UI	☑	2s	Process felt smooth.
Rate app usability	—	—	**4.9/5** – Great experience

Table 6.4: Usability Testing Results – User B Task Performance

Comment: User B also had no trouble at all. Their only note was that everything felt "clear and simple." They finished all tasks smoothly and gave a very high rating.

User C - Task Results

Task	Success	Time Taken	Notes
Signup	☑	36s	Form was clear.
Login	☑	17s	Smooth login.
Guest access	☑	2s	Easy to locate.
Edit profile	☑	22s	Didn't notice Save feedback.
Logout	☑	4s	Very clear.
Navigate to Translation	☑	9s	Logical flow.
Use translation UI	☑	1s	Wanted more visual cues.
Rate app usability	—	—	**4.8/5** – Very usable with minor design suggestion.

Table 6.5: Usability Testing Results – User C Task Performance

Comment: User C completed everything successfully, but mentioned two things: they weren't sure if their profile changes were saved (which aligns with a previously known issue), and they wanted clearer visual feedback before the translation screen.

Usability Summary Table

Task	% of Users Successful	Common Notes
Signup	100%	All completed signup. With no problems
Login	100%	Smooth for all users.
Guest Access	100%	Clear and accessible.
Edit Profile	95%	One users unsure about Save confirmation.
Logout	100%	Very easy to find.
Navigate to Translation	100%	All used Camera first; flow was clear.
Use Translation UI	99%	Some wanted better visual cues.

Table 6.6: Usability Summary Based on Task Completion and Feedback

Overall Feedback

All users completed every task independently, which was a strong indicator of good accessibility and usability — especially considering that all participants were from the deaf or speech-disabled community. Most of the feedback was very positive, and users especially liked how clean and simple the interface was.

The only two small things that came up were:

- A need for better confirmation feedback after editing the profile.
- A clearer visual cue after using the translation feature in the camera screen (like an animation or message that tells the user it worked).

The overall user ratings ranged between **4.8 and 5.0**, which gave us confidence that the app is on the right track in terms of both design and functionality.

6.4 System Testing

Test Case	Description	Steps	Test Data (Inputs)	Expected Result	Actual Result	Result
1	Successful login with correct credentials	Open app ➔ Enter correct email & password ➔ Tap Login	Email: correct@kau.edu.sa Password: password123	User logged in and navigated to Home Screen	User logged in and navigated to Home Screen	PASS
2	Failed login with wrong email	Open app ➔ Enter wrong email ➔ Tap Login	Email: wrong@kau.edu.sa Password: password123	Error message: "Invalid email" appears	Error message: "Invalid email" appears	PASS
3	Failed login with wrong password	Open app ➔ Enter correct email but wrong password ➔ Tap Login	Email: correct@kau.edu.sa Password: wrong password	Error message: "Incorrect password" appears	Error message: "Incorrect password" appears	PASS
4	Forgot password - reset with existing email	Open app ➔ Tap "Forgot Password" ➔ Enter existing email and new password ➔ Confirm new password ➔ Tap Reset	Email: existing@kau.edu.sa New Password: NewPassword123	Password reset successful, success message displayed	Password reset successful, success message displayed	PASS
5	Forgot password - reset with empty fields	Open app ➔ Tap "Forgot Password" ➔ Leave fields empty ➔ Tap Reset	Email: (empty) Password: (empty)	Error: "Email and password required"	Error: "Email and password required"	PASS
6	Forgot password - reset with non-existing email	Open app ➔ Tap "Forgot Password" ➔ Enter unknown email ➔ Tap Reset	Email: notfound@kau.edu.sa Password: NewPassword123	Error: "User not found"	Error: "User not found"	PASS
7	Sign up new user successfully	Open app ➔ Tap "Sign Up" ➔ Fill all fields ➔ Tap Register	Username: new User Email: newuser@kau.edu.sa Password: password123 Phone: 0555555555	Account created successfully, success message shown	Account created successfully, success message shown	PASS
8	Sign up failed - email already exists	Open app ➔ Tap "Sign Up" ➔ Enter existing email ➔ Tap Register	Username: new User Email: existing@kau.edu.sa Password: password123 Phone: 0555555555	Error: "Email already exists"	Error: "Email already exists"	PASS
9	Translation - Open camera and detect sign	Open app ➔ Login ➔ Navigate to Translation screen ➔ Allow camera access ➔ Show a known sign to camera	Hand sign shown to camera	App detects the sign and displays the correct text	App detects the sign and displays the correct text	PASS
10	Translation - Show no sign	Open app ➔ Login ➔ Navigate to Translation screen ➔ Show empty hand (no sign)	No hand sign shown	App shows "No sign detected"	App shows "No sign detected"	PASS

Table 6.7: Functional Test Cases for Mueen Application

This table represents the system testing results for the main features of the Mueen application. Each test was designed to check how the app responds to different inputs and user actions, such as login, password reset, sign up, and gesture translation. The results show that the system performed correctly in all cases and met the expected outcomes, which confirms that the application is stable and functions as intended.

6.5 Unit Testing

```
@override
Future<Map<String, dynamic>> resetPassword(String email, String newPassword) async {
  if (email.isEmpty || newPassword.isEmpty) {
    return {"success": false, "message": "Email and password required"};
  }
  if (email == "existing@kau.edu.sa") {
    return {"success": true, "message": "Password reset successful"};
  }
  return {"success": false, "message": "User not found"};
}
}
```

Figure 6.5.1 Reset Password Function Logic

```
Run | Debug
group('AuthService Reset Password Tests', () {
  Run | Debug
  test('Reset password successful with existing email', () async {
    final authService = FakeAuthService();
    final response = await authService.resetPassword('existing@kau.edu.sa', 'NewPassword123');
    expect(response['success'], true);
    expect(response['message'], "Password reset successful");
  });

  Run | Debug
  test('Reset password with empty fields', () async {
    final authService = FakeAuthService();
    final response = await authService.resetPassword('', '');
    expect(response['success'], false);
    expect(response['message'], "Email and password required");
  });

  Run | Debug
  test('Reset password with non-existing email', () async {
    final authService = FakeAuthService();
    final response = await authService.resetPassword('notfound@kau.edu.sa', 'NewPassword123');
    expect(response['success'], false);
    expect(response['message'], "User not found");
  });
});
}
```

Figure 6.5.2 Unit Test Scenarios for Reset Password

```
PS C:\Users\DELL\Downloads\Mueen1\Mueen\mueen> flutter test
00:01 +11: All tests passed!
```

Figure 6.5.3 Unit Test Execution Result in Terminal

To ensure the reliability of the reset password functionality in the Mueen application, a set of unit tests was created. These tests cover different scenarios including a successful password reset with a valid email, an attempt with empty fields, and an attempt with a non-existing email. The tests were written using Flutter's built-in test package, and the results confirmed that all cases passed successfully. This demonstrates the correctness and stability of the reset logic under various input conditions.

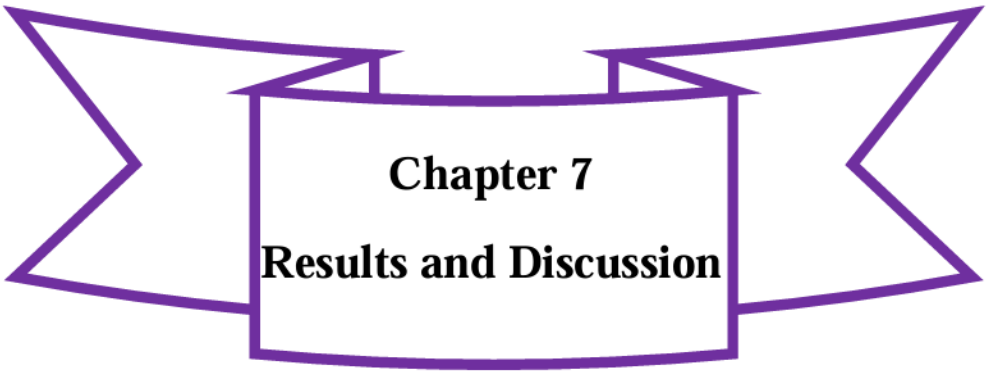
6.6 Conclusion

Testing was one of the most important stages in developing the Mueen app. It helped us make sure that the app wasn't just working, but actually ready for real users — especially those from the deaf and speech-disabled community, who the app was built for.

We carried out five types of testing: acceptance, integration, usability, unit, and system testing. Each one gave us a different layer of insight. Acceptance testing confirmed that all the main features worked the way they were supposed to. Integration testing showed us that the frontend and backend were communicating correctly. Usability testing gave us real feedback from actual users in our target audience, which made the results more meaningful. Unit testing helped us make sure each part of the code was solid before connecting everything. And system testing brought it all together by checking the full user journey from start to finish.

Yes, there were a couple of issues — like the translation output repeating and the profile save feedback not being clear — but those were either fixed or noted for the next version. What matters is that by the end of this phase, we were confident that the app was working, stable, and actually helpful for the people it was meant to support.

In short, testing helped us move from a working prototype to a real, usable app. And that made all the difference.



7.1 Introduction

7.2 Interfaces of our App

7.3 Accomplished Objectives

7.4 Work Limitation

7.5 Conclusion

7.1 Introduction

This chapter highlights the main results of our work on the Mueen application, focusing on the app's interface design, the objectives we achieved, and the limitations we encountered during development. It reflects both the technical outcomes and our experience building a solution for real users specifically, deaf individuals who often face communication barriers in dental clinics. The results here are based on what we implemented, tested, and refined over the course of the project.

7.2 Interfaces of our website

The Mueen application consists of several screens that guide the user through a smooth and accessible experience. Each screen was designed with simplicity, clarity, and the needs of deaf and speech-disabled users in mind.

Below are the main interfaces, along with a brief explanation of each:

1. Splash Screen

figer7.1 splash screen

Displays the app's logo briefly before navigating to the Welcome screen. It sets the tone for the user experience and gives a smooth entry point into the app.

2. Welcome Screen

figier7.2 welcome screen

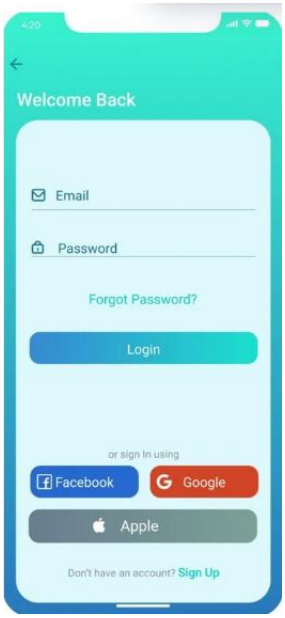
Offers users two main options: sign up, or continue as a guest. This screen is simple and acts as the starting point for navigation.

3. Guest Screen

figer7.3 guest screen

Allows users to access a limited version of the app without creating an account. It’s designed for quick access when needed.

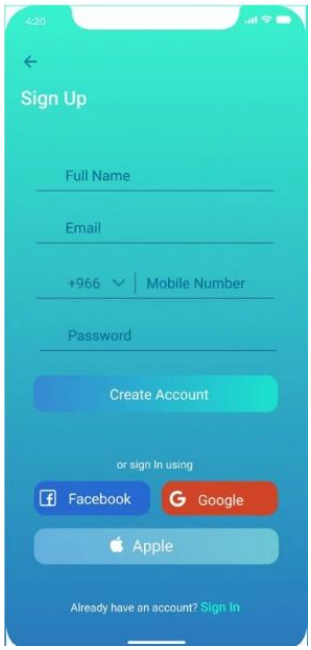
4. Login Screen



Figuer7.4 login

Registered users can enter their email and password to access the app. Field validation and backend authentication are included to ensure secure access.

5. Signup Screen



figuer7.5 singup

New users can register by filling in the required information. The form includes validation for email, password, and mobile number.

6. Home Screen

fiauer7.6 Home screen

Acts as the central hub of the app. From here, users can access the translation feature by camera, view their profile, or log out. The layout is clean and icon-based.

8. Camera Screen

figuer7.7 camera screen

This screen opens the device’s camera and allows users to capture a sign gesture. It includes buttons for flash, capture, and starting the recognition process. This is the first step in the translation flow.

9. Translation Screen

figuer7.8 Translation screen

Displays the translation result of the captured sign in text form and audio playback for added accessibility.

10. Profile Screen

figuer7.9 profile scree

Allows users to view and edit their account information such as email, username, phone number, and password. Additional options include changing language, sending feedback, and logging out.

Each screen was implemented based on the high-fidelity Figma design and adjusted after testing with real users to make sure the app was easy to understand and interact with — especially for users with hearing or speech disabilities.

7.3 Achievement Objective:

1. **Facilitate communication:** Ensure real-time communication between deaf patients and dental practitioners by translating SSL into text and audio.
2. **Reduce reliance on interpreters:** Minimize the need for interpreters in clinics, reducing costs and increasing accessibility.
3. **Enhance healthcare experience:** Provide a more inclusive, comfortable experience for deaf patients in dental settings.

How We Achieved:

- Built the Mueen app with live sign-to-text-audio translation using the camera.
- Trained and integrated a gesture recognition model using a custom SSL dataset.
- Designed a clean and simple UI so that users could navigate easily, even without guidance.
- Connected the app to a backend database for user management and future scalability.
- Included feedback from real deaf users in the testing phase to ensure the app worked as intended.

7.4 Work Limitation:

While we were able to achieve our core objectives and deliver a working version of Mueen, there were still a few limitations that came up during development and testing. These didn't take away from the value of the app, but they helped us clearly see where improvements can be made in future versions.

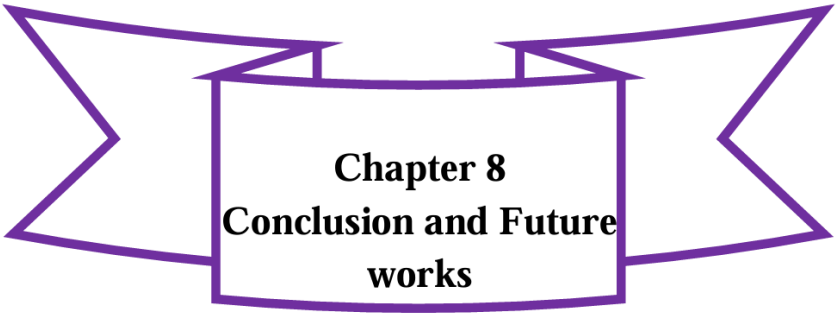
1. **Limited Sign Coverage**
   We built a custom dataset of approximately **80 Saudi Sign Language signs**, focused specifically on terms used in dental clinics. While this covered the basic communication needed in that setting, it's still a small portion of SSL. There are many other important words and phrases that could be added to make the app more useful in broader scenarios.
2. **Model Generalization Issues**
   Since the model was trained using signs from just two individuals, it works best with gestures that match their style. Accuracy noticeably decreases when tested with different users who sign differently in terms of speed, hand shape, or body positioning. More data from diverse signers would help improve this.
3. **Camera Sensitivity**
   The app relies heavily on the device's camera to detect signs. However, it requires **good lighting and a clear background** to function properly. If the environment is too dark, crowded, or if the user is too far from the camera, the recognition accuracy drops.

4. **Limited Testing Scope**
   While we conducted usability testing with real deaf users, the number of participants was small. More extensive testing with a wider range of users would give us better feedback and help identify other usability concerns we may have missed.

Despite these limitations, the Mueen app performed well under realistic conditions and proved that the core idea works. These constraints didn't stop us — they simply gave us a clear direction for what to work on next.

7.5 Conclusion:

This chapter presented the main outputs of the Mueen app, including the user interface, the objectives we achieved, and the limitations we encountered during development. Overall, the app successfully met its core goal of supporting communication for deaf patients in dental clinics using real-time sign language translation. While there are still areas for improvement, the results show that Mueen is a strong starting point with real potential for impact and future growth.



8.1 Introduction
8.2 Difficulties We Faced
8.3 Changes
8.4 Future Works
8.5 Conclusion

8.1 Introduction

This chapter wraps up the final phase of our project and reflects on the full journey of developing **Mueen**, a mobile app that translates Saudi Sign Language into text and audio, designed to support deaf and speech-disabled individuals in real-life situations. Our goal from the start was to build something meaningful that could actually help people communicate, starting with dental clinics, where clear communication is critical but often unavailable for the deaf community.

In the introduction, we defined the problem and clearly stated what we were trying to solve. We knew that interpreter availability is limited, and we wanted to create a solution that would help users communicate independently. In the literature review, we explored existing sign language translation technologies, compared them to our idea, and found that none of them fully addressed the needs of Saudi users in specific settings like clinics.

We moved on to the analysis phase, where we collected real feedback through questionnaires and interviews with our target users. We identified both functional and non-functional requirements, and visualized system behavior using use case diagrams and data flow diagrams.

In the design phase, we created both low- and high-fidelity prototypes, mapped out the interface flow, and designed the database. The implementation chapter covered how we built the app using Flutter, PHP, and MySQL, including the process of creating our **own dataset** from scratch to train the translation model.

During testing, we applied multiple types: acceptance, integration, usability, unit, and system testing. We tested with real users from the deaf community and gathered feedback that helped us fine-tune the experience. Finally, we presented our results, explained what worked and what didn’t, and made decisions on what to improve or change moving forward.

This chapter brings everything together, from the challenges we faced to what’s next for Mueen and reflects on what this project really meant to us, beyond just code and diagrams.

8.2 Difficulties We Faced

Throughout the development of the Mueen application, we faced several challenges, some technical, some time-related, and others related to the uniqueness of the problem we were trying to solve. Since we were building something that didn't already exist in the local context, we had to work through unfamiliar territory and figure out solutions along the way.

One of the biggest challenges was the **creation of the dataset**. There were no ready-made datasets available for Saudi Sign Language that suited our app's use case in a clinical setting. So, instead of relying on existing sources, we made the decision to build our own dataset entirely from scratch. This involved:

- Identifying the relevant signs specific to dental clinic conversations
- Coordinating with a sign language expert to record accurate gesture samples
- Capturing and organizing sign videos and labeling them correctly
- Structuring the dataset in a way that could be used to train the translation model

This part of the project took a lot of effort, planning, and trial-and-error. It was time-consuming, but it was necessary and it helped us gain a deeper understanding of how important data quality is to AI applications.

Other key difficulties we faced included:

- **Integrating the model with the Flutter application**: Aligning the model output with the app's structure was technically tricky and required multiple revisions.
- **Connecting features together smoothly**: From login and signup to camera access and translation output, we had to make sure each function worked well on its own and didn't break when combined with others.
- **Time constraints**: Balancing the implementation, documentation, testing, and presentation deadlines was a real challenge especially while building a unique solution with limited resources.

Despite all of this, we managed to overcome the major roadblocks by staying organized, splitting responsibilities as a team, and supporting each other during critical moments. The fact that we created the dataset ourselves, got the model running, and delivered a working app under pressure is something we're genuinely proud of.

8.3 Changes

As the project progressed, several changes were made both in technical execution and overall scope to help keep the app within reach while still maintaining its purpose. These changes were driven by practical limitations, feedback from our supervisors, and input from real stakeholders in the deaf community.

One of the biggest changes was **adjusting the scope of the app**. From the beginning, our vision for Mueen was ambitious: we wanted to build a general-purpose translation app that could help deaf and speech-disabled individuals communicate in any environment hospitals, airports, schools, government offices, and more. It wasn't just about the idea; it was something we felt strongly about.

However, during the proposal phase, the committee advised us to narrow the scope to something more specific and achievable within the time, tools, and resources we had. Instead of choosing randomly, we reached out directly to **organizations and individuals from the deaf community** to ask where the need was most critical. Based on their feedback, the two most urgent areas were **hospitals and law firms**. After further consideration and discussion, we decided to focus on **dental clinics** as a focused starting point. This environment allowed us to control the dataset vocabulary, design relevant user flows, and build a solid version of the app that could later be expanded to other sectors.

In addition to narrowing the scope, we made several adjustments based on what we learned along the way:

- **Translation model handling**: After noticing repeated outputs during testing, we modified how the model interacts with the app to better prepare it for future updates and to reduce repetition.
- **UI and UX refinements**: We simplified navigation and made visual improvements to buttons, screen flow, and feedback cues, all based on what our deaf users shared during testing sessions.
- **Technical simplifications**: Some backend processes and transitions between screens were streamlined to reduce bugs and improve performance on real devices.

These changes weren't a step back, they were a way to focus on delivering a quality, working solution without losing sight of the bigger goal. The full vision for Mueen to serve in a wide range of public and professional spaces is still at the heart of the project. We just chose to begin where the need was real and where the app could make an immediate difference.

8.4 Future Work

Although Mueen currently focuses on sign language translation in dental clinics, this is just the beginning. The long-term goal has always been to create an inclusive solution that can serve deaf and speech-disabled individuals across different sectors of life. Based on user needs, technical learnings, and future potential, we identified several directions for development.

- **Wider medical field support**: We aim to expand the app beyond dental clinics to cover other medical environments such as emergency rooms, general hospitals, pharmacies, and specialized clinics. These settings involve critical communication where immediate understanding is essential for safety and care quality.
- **Stronger model training and real-time performance**: One of the top priorities is to improve the translation model. We plan to train it using a larger and more diverse dataset ideally with more sign samples, real-user inputs, and better gesture variation. This will help increase accuracy, reduce output repetition, and support continuous real-time recognition.
- **Improved camera detection and environment adaptability**: Currently, lighting conditions, camera angle, or background movement can affect how signs are captured. In future versions, we want the app to adjust more intelligently to different environments, ensuring smoother and more reliable sign recognition regardless of where or how it's used.
- **Use in non-medical sectors**: One of our biggest goals is to expand Mueen to serve in places like **schools for deaf students**, **government service centers**, **airports**, and **law firms**. These were all mentioned as high-priority settings during our research and interviews with the deaf community. In schools, for example, teachers and students could communicate more naturally without needing an interpreter for every interaction.
- **User customization and accessibility features**: We also plan to add more personalization features such as translation speed control, repeat/recapture options, gesture guides, and interface adjustments to ensure the app works for different age groups and literacy levels within the deaf community.

Ultimately, we believe that Mueen can grow from a dental clinic tool into a national accessibility platform, one that opens doors for inclusion, independence, and equal communication across multiple sectors of society

8.5 Conclusion

This chapter reflected on the final stages of the Mueen project including the challenges we faced, the changes we made, the future directions we're planning, and most importantly, the overall journey we went through to reach this point.

Mueen started as a bold and ambitious idea: to build an application that could break communication barriers between deaf or speech-disabled individuals and people in public services. We faced a lot of questions early on, how will it work? Can it really be done? How do we make it useful for real people in real settings?

And yet, step by step, we moved from idea to execution.

From defining the problem in the introduction, analyzing similar solutions in the literature review, and gathering real user needs in the analysis chapter, to designing, building, testing, and finally evaluating our system, every stage taught us something new. We built our own dataset from scratch, adapted the scope based on feedback, and tested the app with members of the deaf community to make sure we were building something for them, not just about them.

Yes, there were obstacles, technical ones, time-related ones, and even emotional ones when we had to let go of certain ideas. But with every challenge, we learned to problem-solve, collaborate, and push forward as a team.

This project wasn't just about software. It was about creating something with purpose. Something human.

In the end, we didn't just deliver a mobile application. We delivered a message that inclusion matters, and that technology should serve *everyone*, not just the majority.

Mueen is ready to be used in dental clinics, but its journey doesn't stop there. What we built here is only the beginning of what we hope will grow into a larger platform that supports the deaf community in many areas of life, from hospitals and schools to public services and beyond.

We're proud of what we created. And more than that, we're proud of how far we've come, not just as students, but as a team who turned a vision into reality.