Mueen

مُعين

Saudi Sign Language Translation

Group members:

Hadeel Bakhshwen 2006299

Jood Kawther 2009599

Shahad Alharthi 2105261

Almaha abdulwahab 2009018

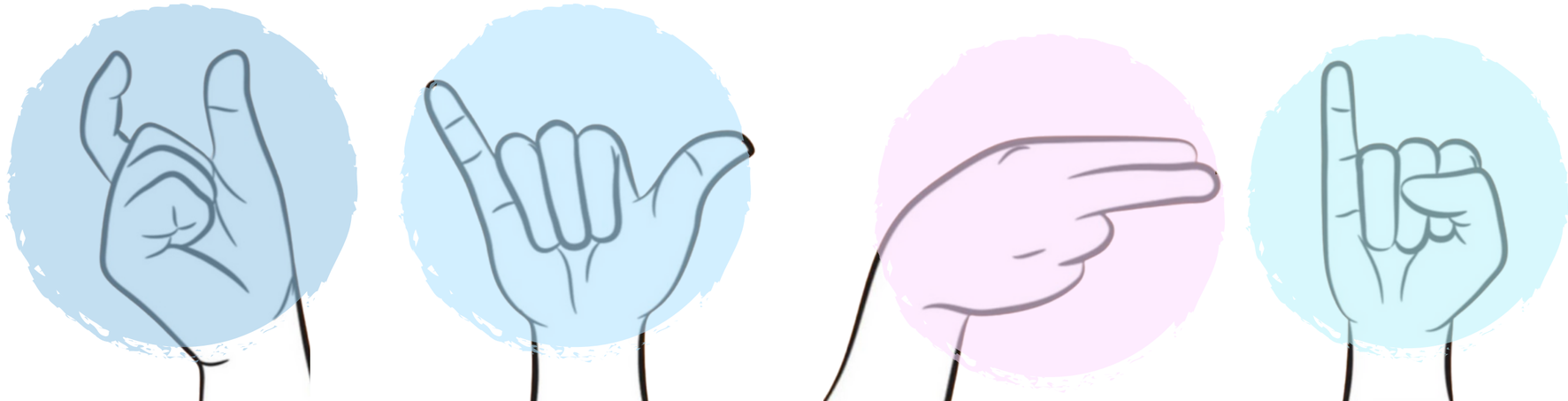
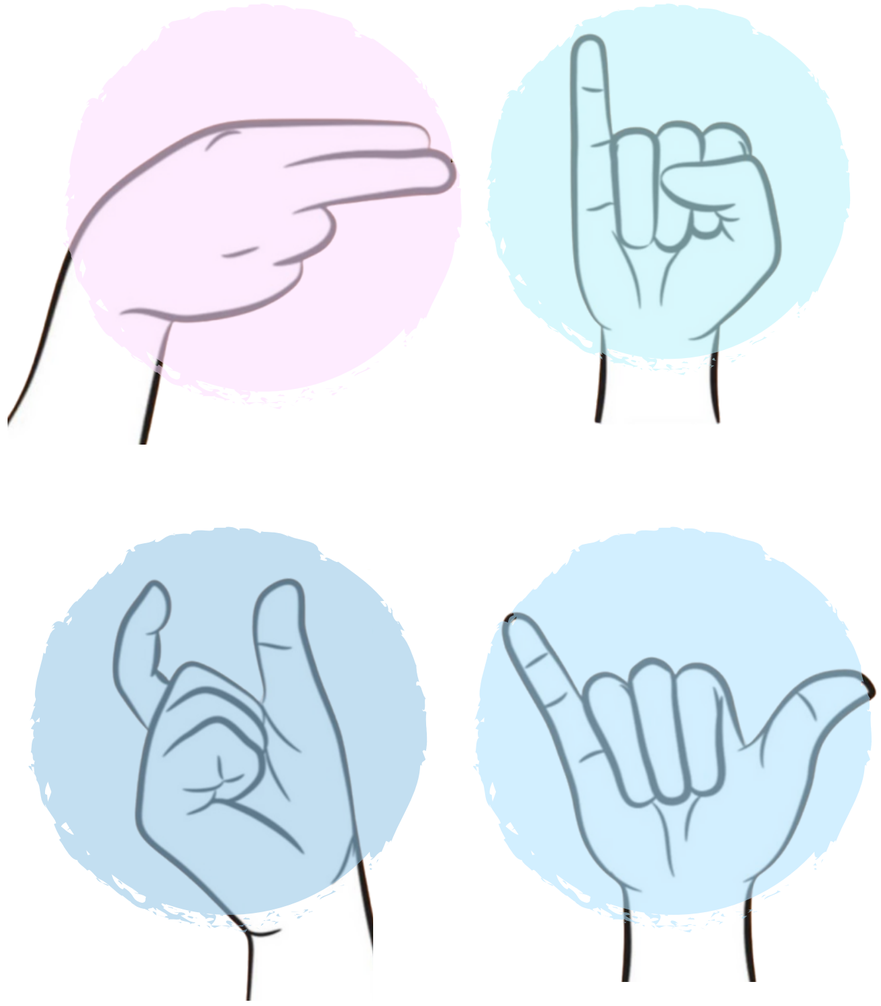


TABLE OF CONTENTS

01

Introduction

By: Hadeel Bakhshwen

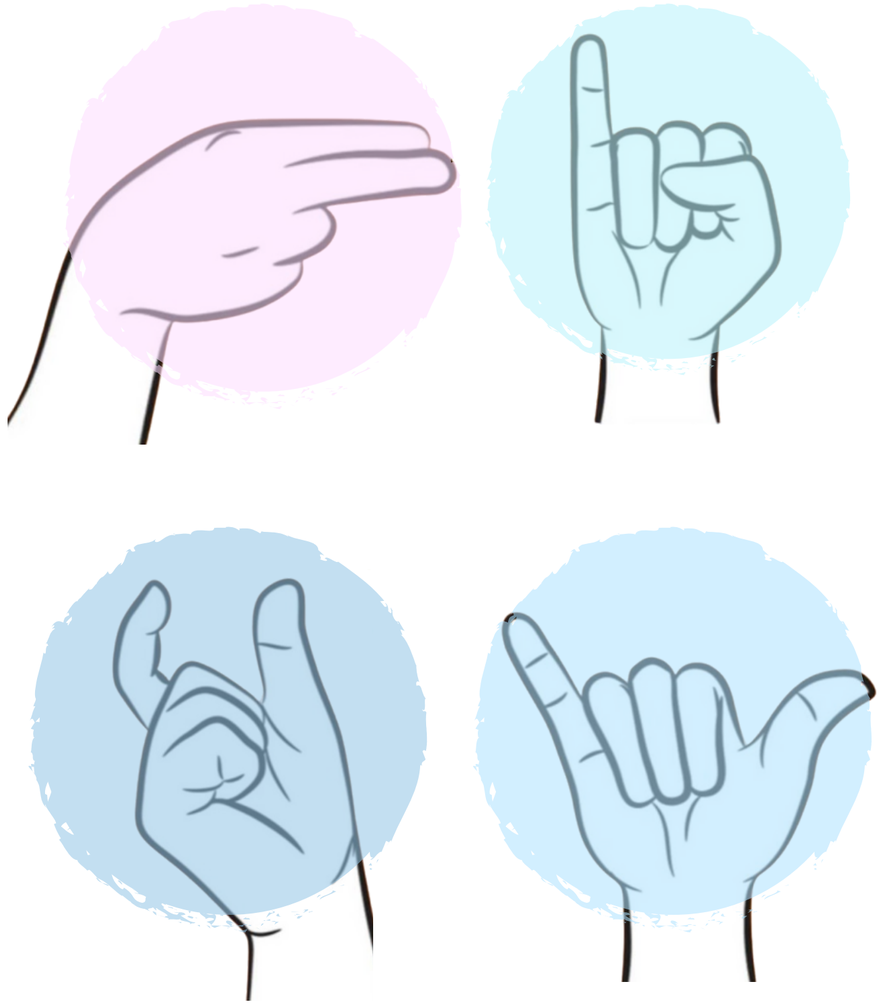


Mueen App is a specialized communication tool for dental clinics, bridging the gap between hearing-impaired patients and medical staff. It translates sign language into text and optionally into audio for smoother interactions.

02

Features

By: Hadeel Bakhshwen

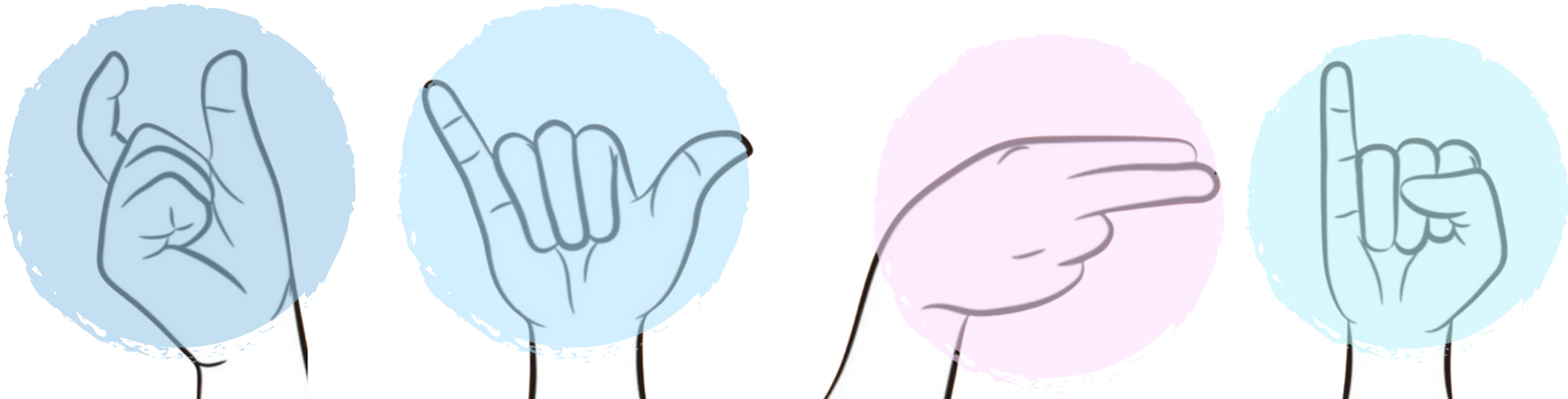


1. The user performs a SSL gesture in front of the device's camera

2. The gesture is analyzed using a trained AI model to recognize its meaning

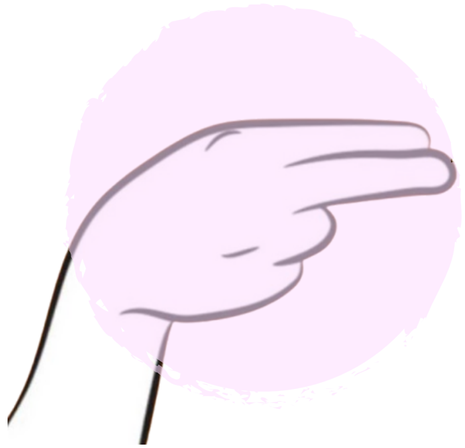
3.Translated gesture appears as text

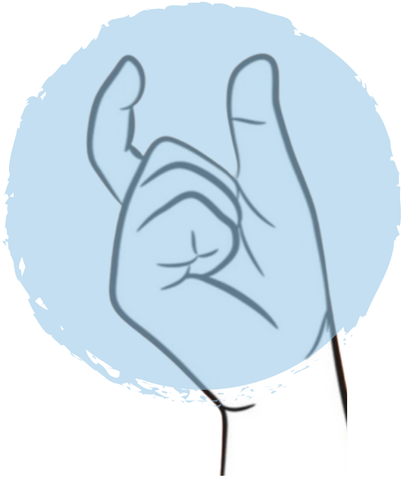
4. Text is converted into speech

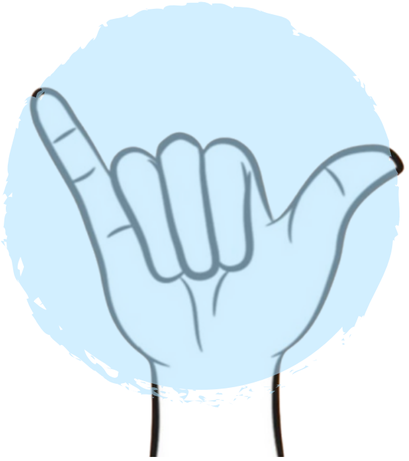


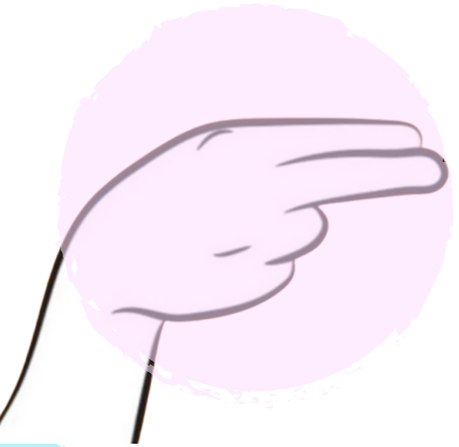
03

Design of the System





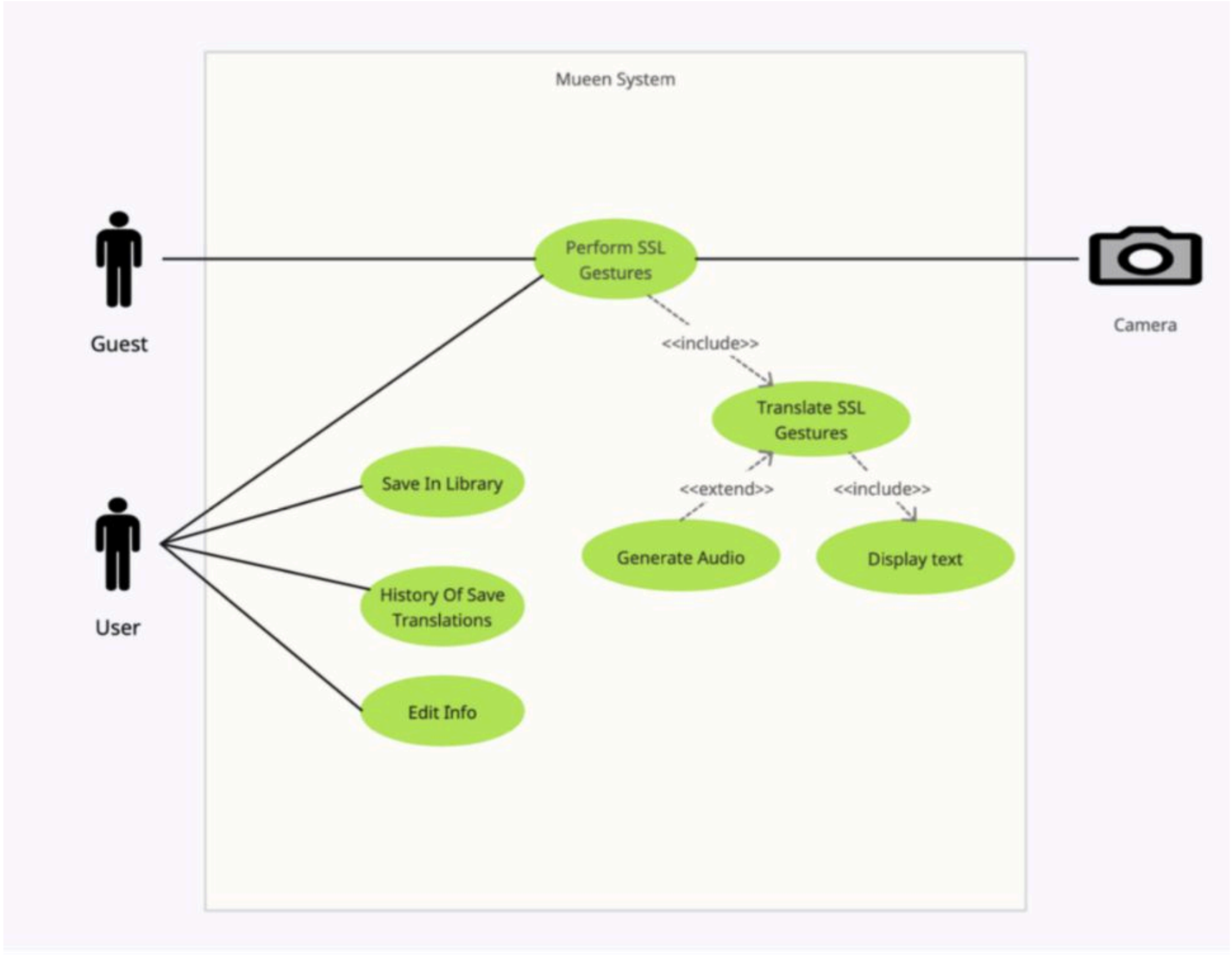


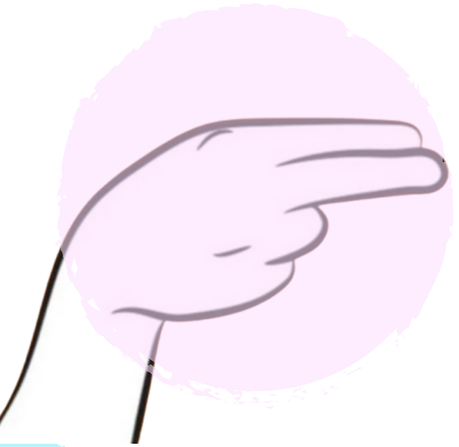


UseCase Diagram

3.A

By: Hadeel Bakhshwen

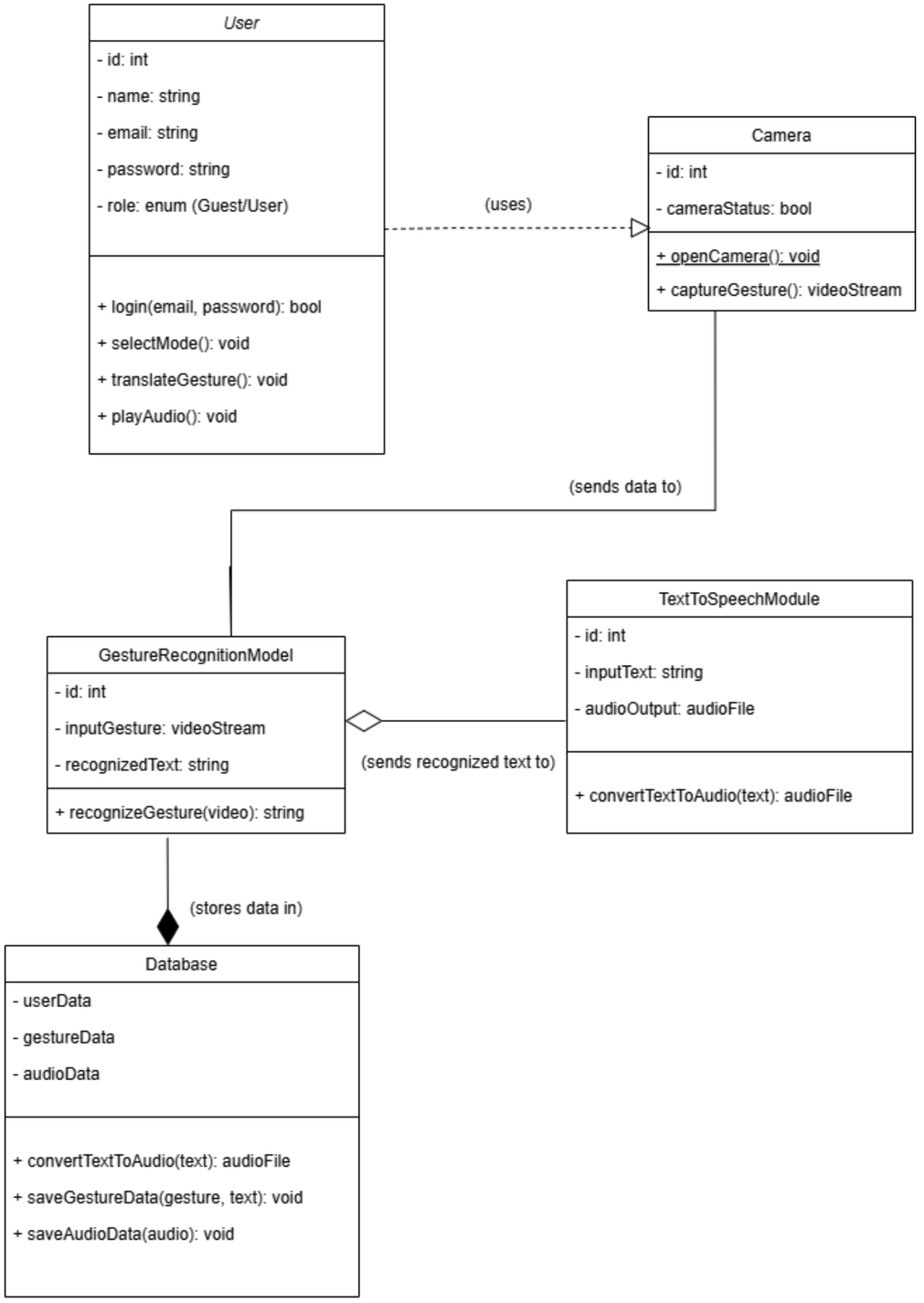

Mueen System
Guest
User
Perform SSL Gestures
Camera
<<include>>
Translate SSL Gestures
Save In Library
<<extend>>
<<include>>
Generate Audio
Display text
History Of Save Translations
Edit Info

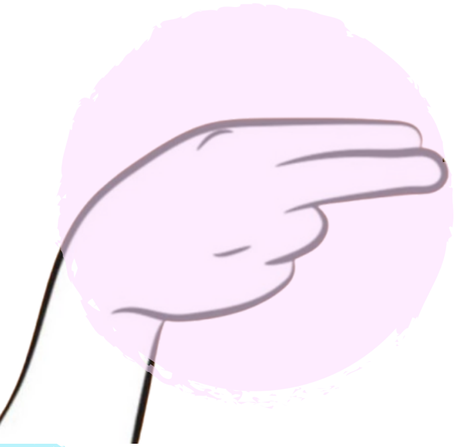


Class Diagram

3.B

By: Shahad Alharthi


User
- id: int
- name: string
- email: string
- password: string
- role: enum (Guest/User)
+ login(email, password): bool
+ selectMode(): void
+ translateGesture(): void
+ playAudio(): void
(uses)
Camera
- id: int
- cameraStatus: bool
+ openCamera(): void
+ captureGesture(): videoStream
(sends data to)
GestureRecognitionModel
- id: int
- inputGesture: videoStream
- recognizedText: string
+ recognizeGesture(video): string
(sends recognized text to)
TextToSpeechModule
- id: int
- inputText: string
- audioOutput: audioFile
+ convertTextToAudio(text): audioFile
(stores data in)
Database
- userData
- gestureData
- audioData
+ convertTextToAudio(text): audioFile
+ saveGestureData(gesture, text): void
+ saveAudioData(audio): void

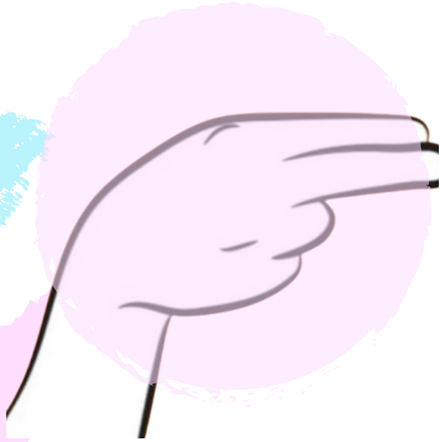


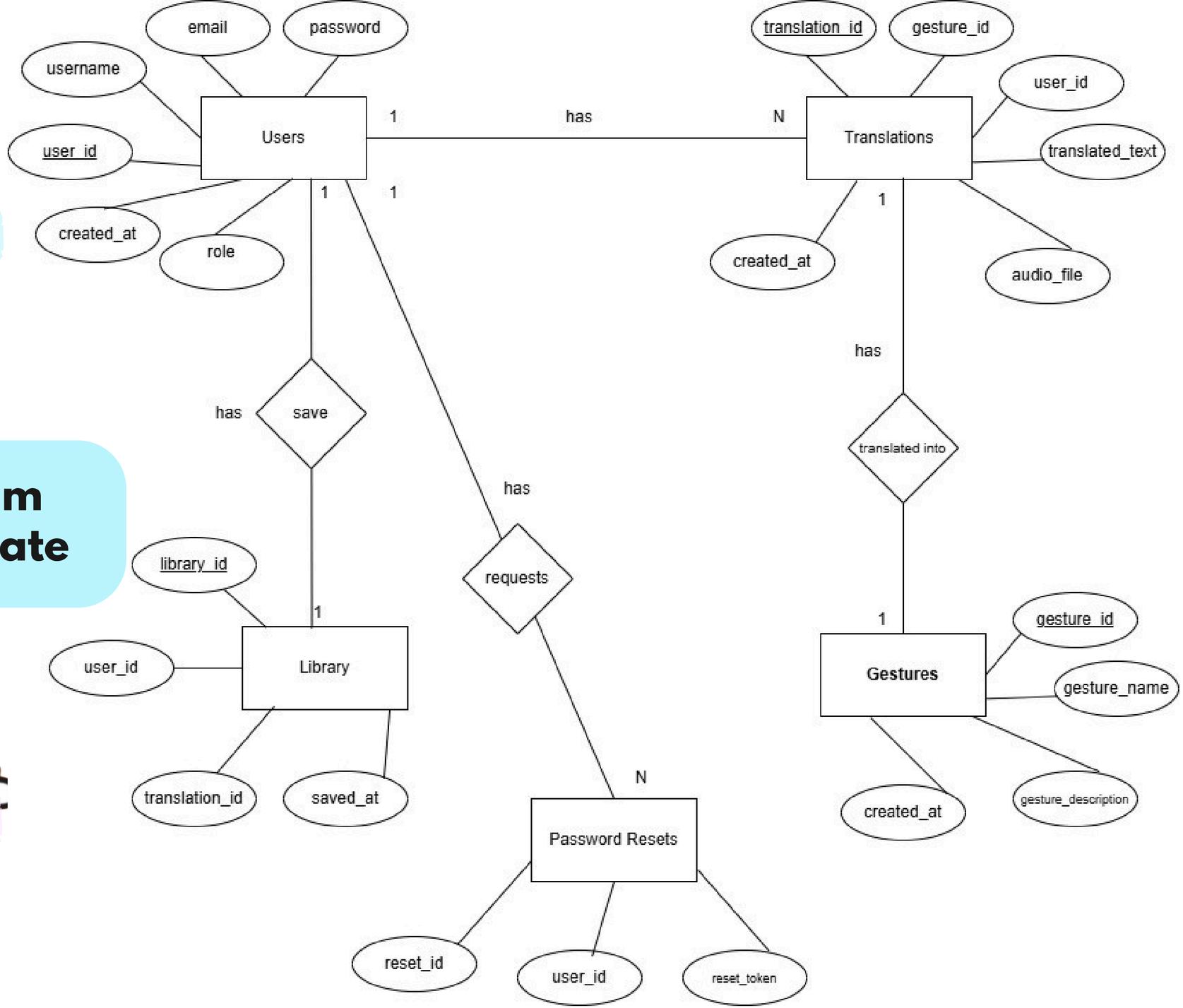
ER Diagram

3.D

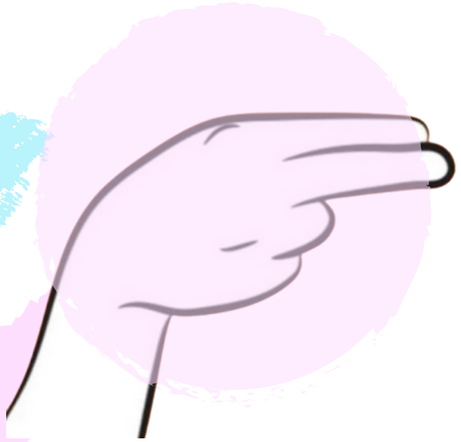
By: Almaha abdulwahab

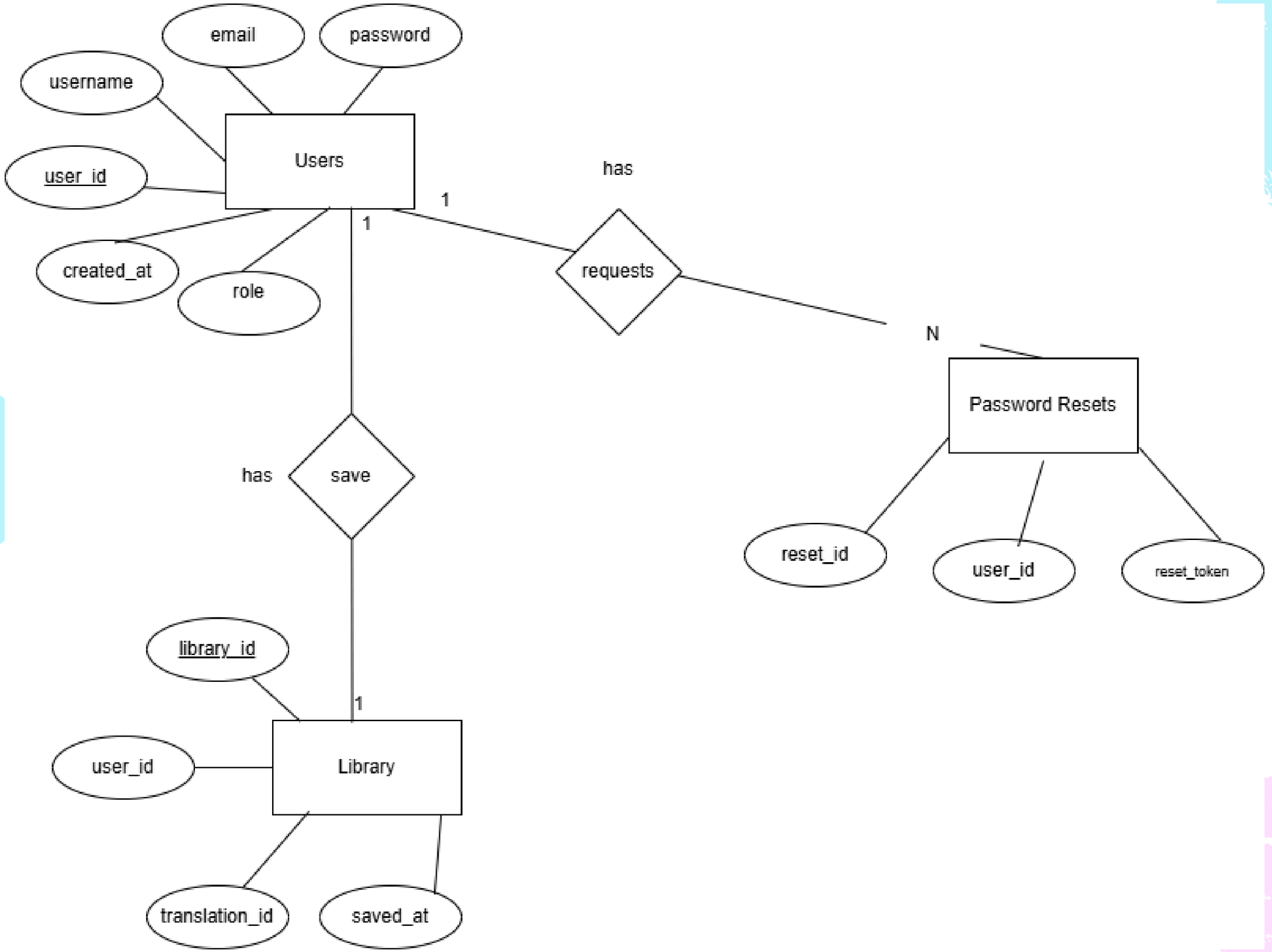
ER Diagram before update





ER Diagram after update

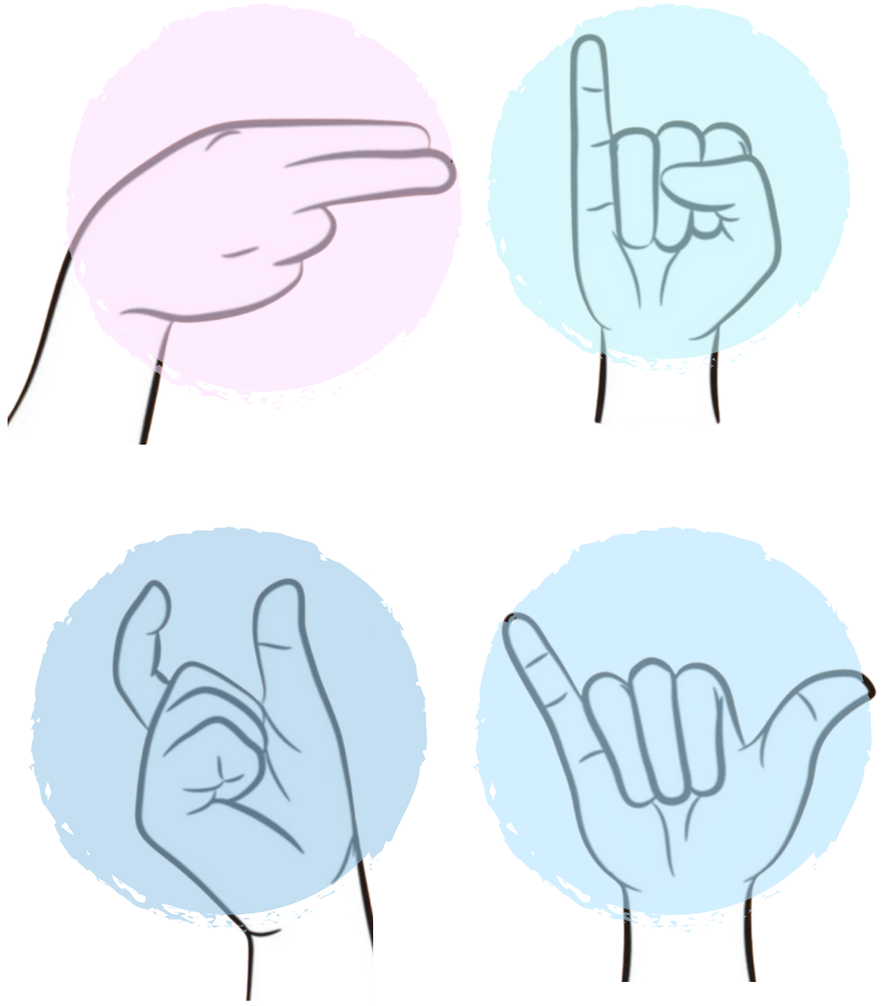




04

Testing

By: Jood Kawther



Usability Testing

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 user 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.

- Users rated the app between 4.8 and 5 stars.
- All tasks were completed without help.
- Overall, the experience was smooth and positive.
- The UI was described as clean, with intuitive flow
- One suggestion: add clearer cues on the translation screen after using the camera.

Usability Testing

User A

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.

User B

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

User C

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.

Acceptance Testing

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

Unit Testing

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:mueen/services/auth_service.dart';
// FakeAuthService to simulate backend behavior
class FakeAuthService extends AuthService {
  @override
  Future<Map<String, dynamic>> login(String email, String password) async {
    if (email.isEmpty || password.isEmpty) {
      return {"success": false, "message": "Email and password required"};
    }
    if (email == "correct@kau.edu.sa" && password == "password123") {
      return {"success": true, "message": "Login successful"};
    }
    if (email == "wrong@kau.edu.sa") {
      return {"success": false, "message": "Invalid email"};
    }
    if (password == "wrongpassword") {
      return {"success": false, "message": "Incorrect password"};
    }
    throw Exception('Server not reachable');
  }
```

```dart
  @override
  Future<Map<String, dynamic>> signup(String username, String email, String password, String phone) async {
    if (username.isEmpty || email.isEmpty || password.isEmpty || phone.isEmpty) {
      return {"success": false, "message": "All fields are required"};
    }
    if (email == "existing@kau.edu.sa") {
      return {"success": false, "message": "Email already exists"};
    }
    return {"success": true, "message": "Sign up successful"};
  }
```

```dart
  @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"};
  }
}
```

```dart
Run | Debug
void main() {
  Run | Debug
  group('AuthService Login Tests', () {
    Run | Debug
    test('Login successful with correct credentials', () async {
      final authService = FakeAuthService();
      final response = await authService.login('correct@kau.edu.sa', 'password123');
      expect(response['success'], true);
      expect(response['message'], "Login successful");
    });

    Run | Debug
    test('Login failed with wrong email', () async {
      final authService = FakeAuthService();
      final response = await authService.login('wrong@kau.edu.sa', 'password123');
      expect(response['success'], false);
      expect(response['message'], "Invalid email");
    });

    Run | Debug
    test('Login failed with wrong password', () async {
      final authService = FakeAuthService();
      final response = await authService.login('correct@kau.edu.sa', 'wrongpassword');
      expect(response['success'], false);
      expect(response['message'], "Incorrect password");
    });

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

    Run | Debug
    test('Login fails due to server error', () async {
      final authService = FakeAuthService();
      try {
        await authService.login('any@kau.edu.sa', 'anyPassword');
        fail('Expected an exception to be thrown');
      } catch (e) {
        expect(e.toString(), contains('Server not reachable'));
      }
    });
  });

```

```dart

  Run | Debug
  group('AuthService Sign Up Tests', () {
    Run | Debug
    test('Sign up successful with valid inputs', () async {
      final authService = FakeAuthService();
      final response = await authService.signup('user1', 'user1@kau.edu.sa', 'password123', '0555555555');
      expect(response['success'], true);
      expect(response['message'], "Sign up successful");
    });

    Run | Debug
    test('Sign up failed with existing email', () async {
      final authService = FakeAuthService();
      final response = await authService.signup('user2', 'existing@kau.edu.sa', 'password123', '0555555555');
      expect(response['success'], false);
      expect(response['message'], "Email already exists");
    });

    Run | Debug
    test('Sign up failed with empty fields', () async {
      final authService = FakeAuthService();
      final response = await authService.signup('', '', '', '');
      expect(response['success'], false);
      expect(response['message'], "All fields are required");
    });
  });

```

```dart

  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");
    });
  });
}
```

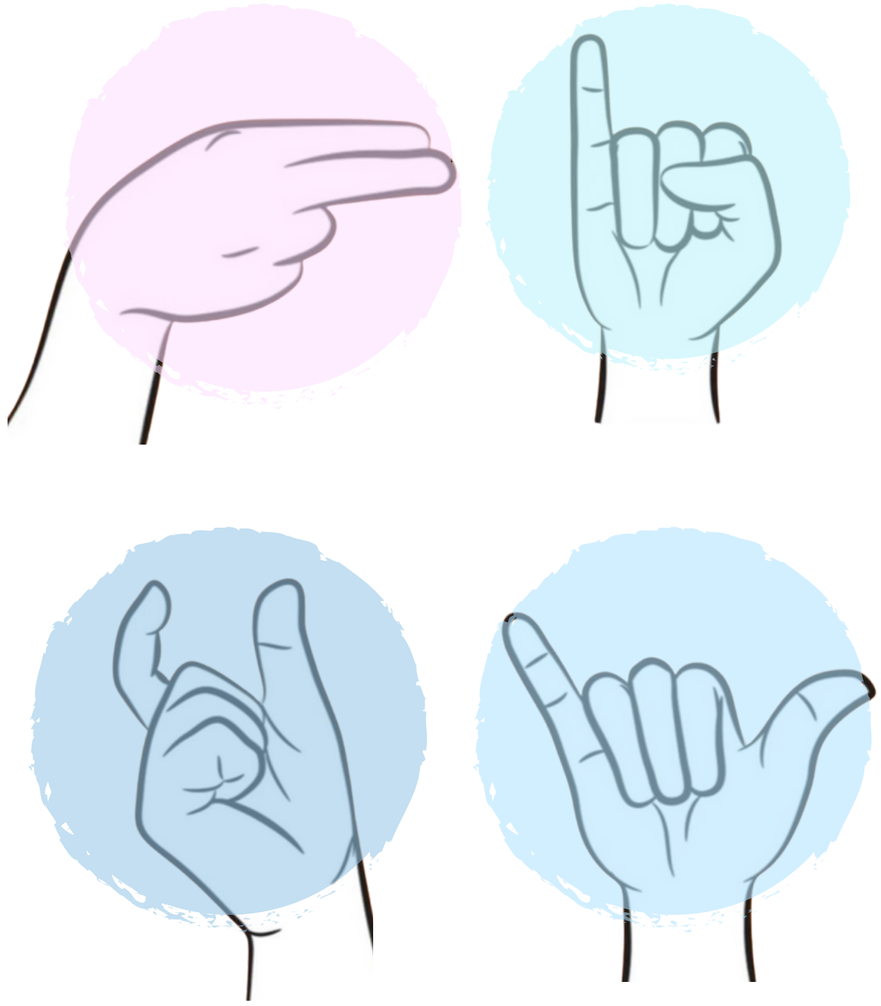
Integration Testing

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

05

Project Limitation

By: Almaha abdulwahab

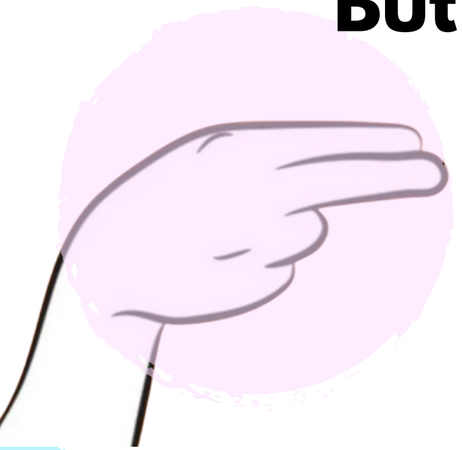


1

We used a small scope of around 50 signs only. However, many other important words are used in dental clinics that are not included in our system.

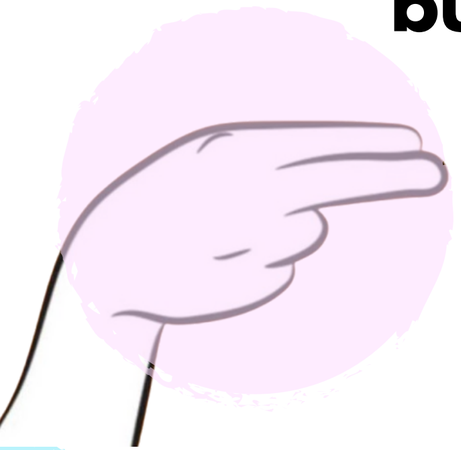
2

Our application uses the camera to detect signs.
But since we trained the model with only two people, the accuracy is low when other users try it.



3

**We planned to convert the sign into voice,
but the voice feature did not work in the final version.**



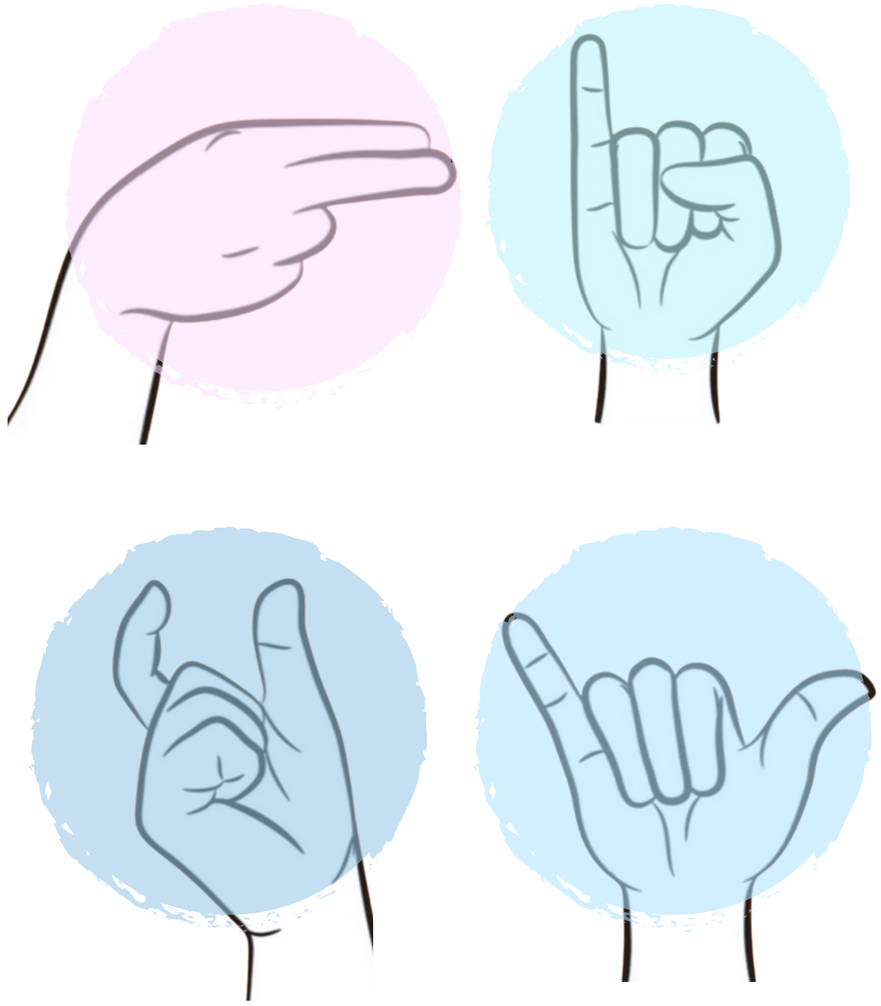
4

The camera needs good lighting and a clear background. If the place is too dark or messy, the app may not detect the sign correctly.

06

Future Work

By: Almaha abdulwahab



1

We plan to increase the scope of our app.
It will not be limited to dental clinics only, but can be used in other medical fields too.

2

We will improve the camera detection to work better in different lighting and positions.

3

We want to support different Arabic dialects, not just the Saudi dialect. This will make the app more useful in other Arabic-speaking countries.

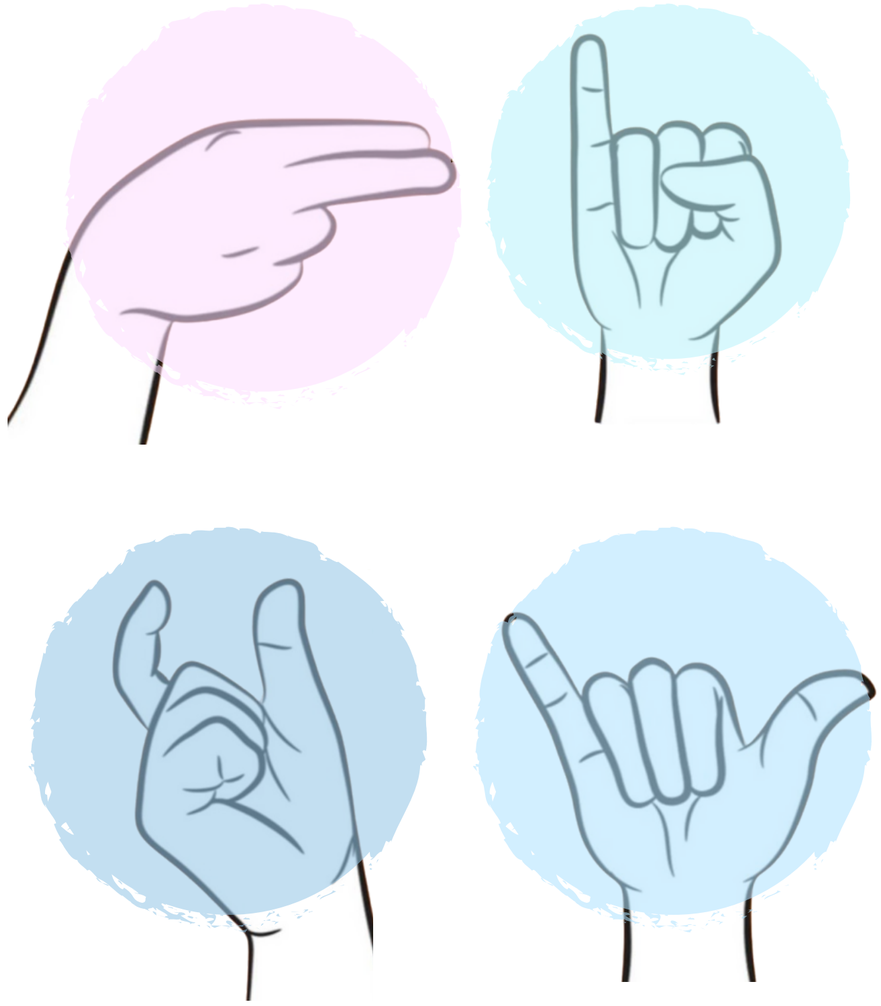
4

We aim to adapt the app for use not only in hospitals and clinics, but also in schools for deaf students, to help teachers communicate with them more effectively.

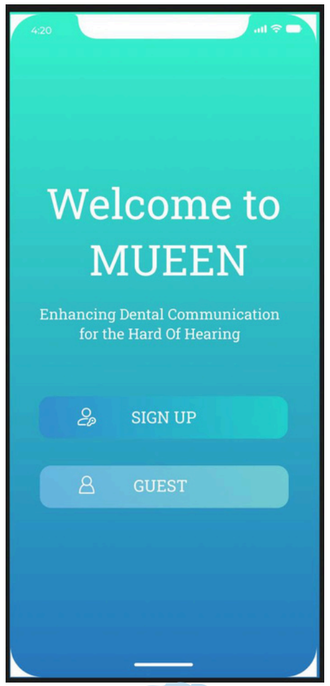
07

Results

By: Shahad Alharthi

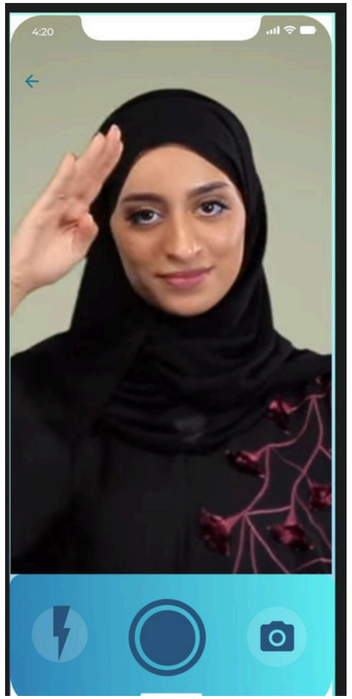



مُعين
Mueen

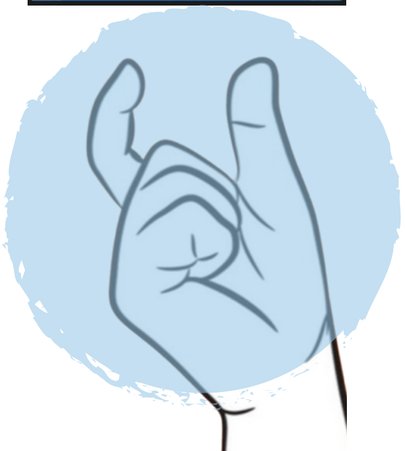

Welcome to
MUEEN
Enhancing Dental Communication
for the Hard Of Hearing
SIGN UP
GUEST

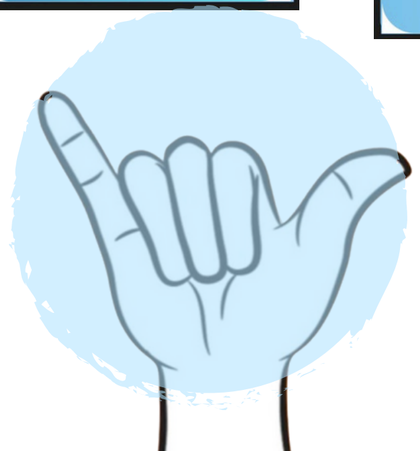

Press to translate

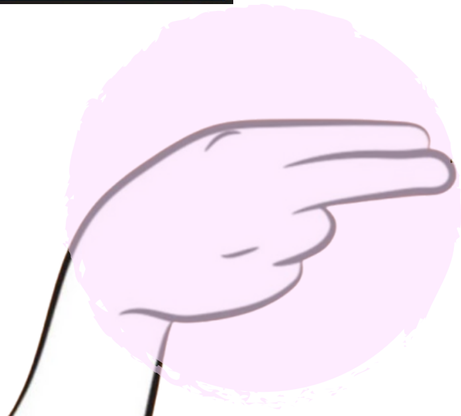

Press to translate

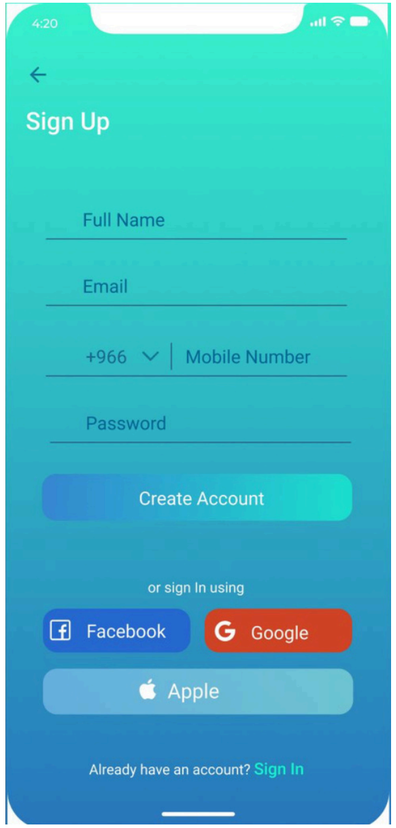


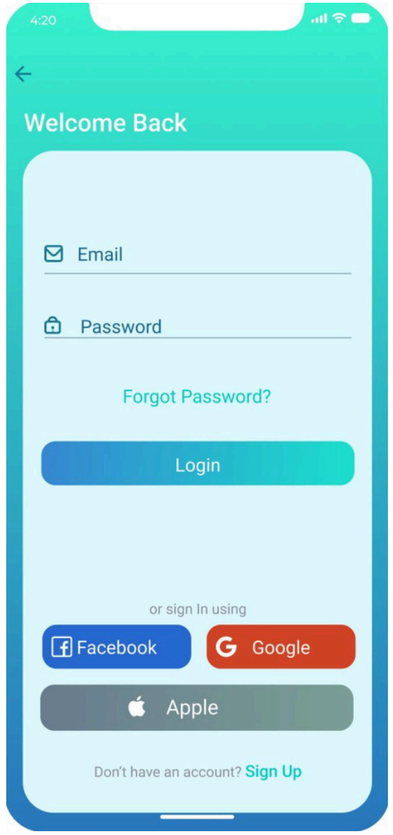

شكرا
Translation

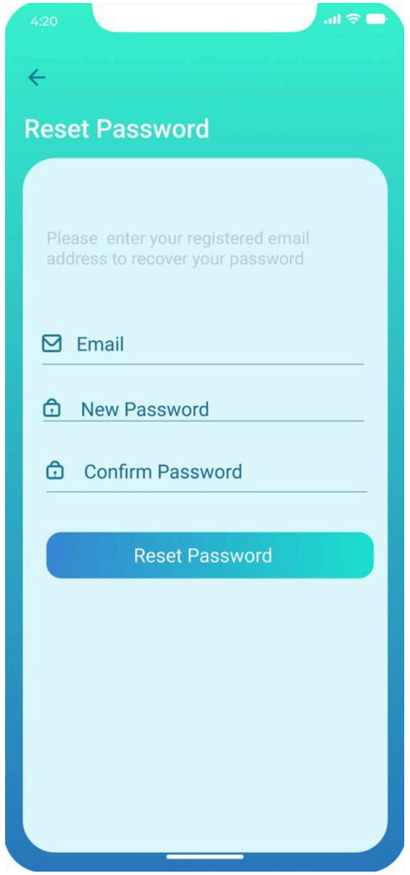


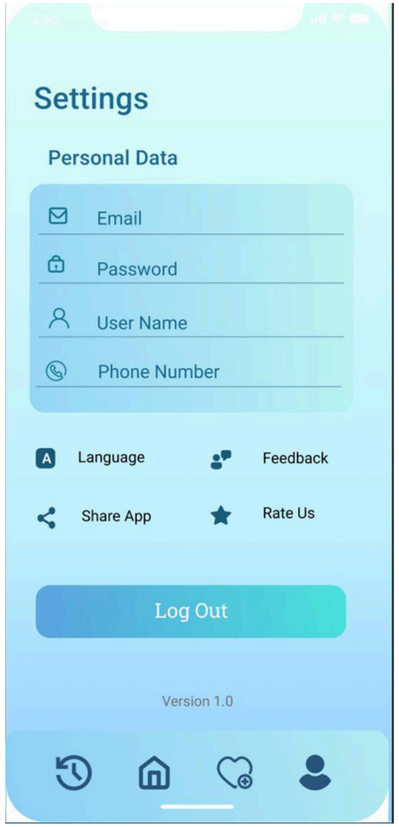





4:20
Sign Up
Full Name
Email
+966
Mobile Number
Password
Create Account
or sign In using
Facebook
Google
Apple
Already have an account? Sign In


4:20
Welcome Back
Email
Password
Forgot Password?
Login
or sign In using
Facebook
Google
Apple
Don't have an account? Sign Up

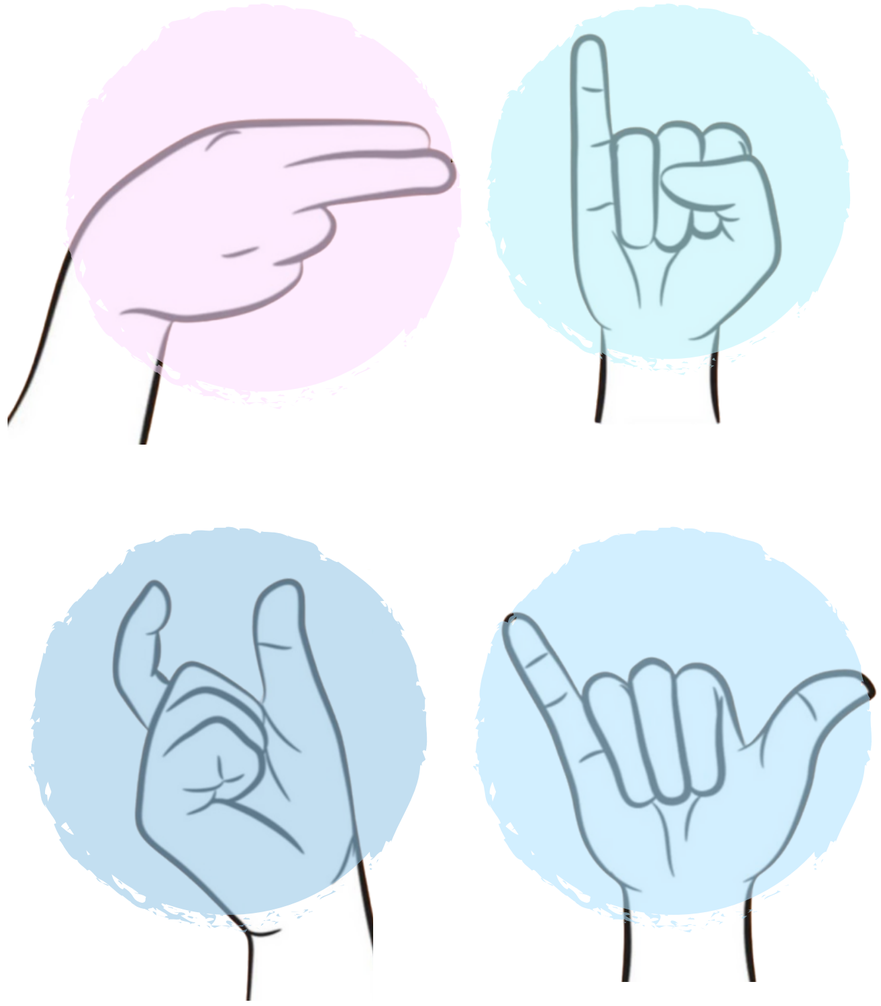

4:20
Reset Password
Please enter your registered email address to recover your password
Email
New Password
Confirm Password
Reset Password


Settings
Personal Data
Email
Password
User Name
Phone Number
Language
Feedback
Share App
Rate Us
Log Out
Version 1.0

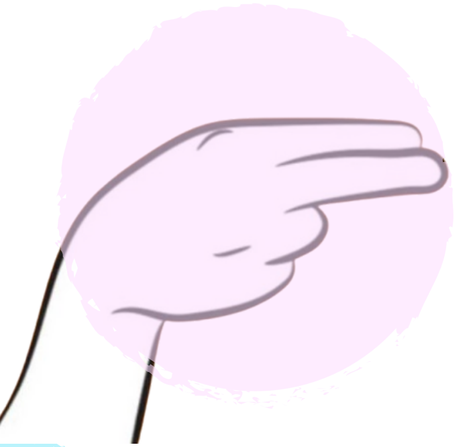
08

Conclusions

By: Jood Kawther



Mueen was built to bridge communication gaps — starting with dental clinics, but aiming much further. This is just the beginning of what inclusive technology can look like.
Thank you for being part of our journey.



And well, just to end it our way:

```
void main() {
  print("Mueen project: Completed ✅");
  print("Future goals: Just getting started 🚀");
}
```