Together App: An AI-Enhanced Sign Language Translator

Amal Mohammed Elsaeed Ibrahim
Mennaallah Aymen Awadallah
Mahmoud Mohamed Wahied Elden Refaei
Course: Mobile Computing

Institution: Helwan University

May 12, 2025

Project Report

Contents

1 Project Overview

The Together App is a pioneering mobile application designed to foster seamless communication between the deaf and hard-of-hearing community and the broader population. By leveraging artificial intelligence (AI), the app provides real-time translation between spoken language and sign language, creating an inclusive platform that bridges communication gaps. Developed using Flutter for cross-platform compatibility, the app supports both Android and iOS devices, ensuring wide accessibility. The source code for the Together App is available on GitHub at https://github.com/WAHiiED/TogetherApp.git, providing access to the app's implementation details and facilitating further development or contributions.

The app integrates a suite of features, including:

- **AI-Powered Translation**: Converts spoken words to sign language videos and interprets sign language videos into text.
- **Video Communication**: Facilitates secure, peer-to-peer video meetings with real-time translation capabilities.
- **User Management**: Offers personalized profiles, favorites, and history tracking for a tailored user experience.
- **Dictionary and Learning Tools**: Provides a comprehensive sign language dictionary for users to learn and reference signs.

The primary objective of the Together App is to empower users by providing intuitive tools that enhance accessibility and promote inclusivity. By combining cutting-edge AI with robust backend services, the app delivers a reliable and user-friendly experience for individuals with diverse communication needs.

2 Technology Stack

The Together App is built using a carefully selected technology stack to ensure performance, scalability, and ease of development. The key components include:

- **Flutter**: A cross-platform framework used for developing the app's user interface (UI) and core logic, enabling a consistent experience on Android and iOS.
- **Firebase**: A comprehensive backend platform providing:
  - *Authentication*: Secure user login and signup with email/password authentication.
  - *Firestore*: Real-time NoSQL database for storing user profiles, favorites, and translation history.
  - *Storage*: Cloud storage for media files such as sign language videos and audio

recordings.

- **Flask**: A lightweight Python framework hosting the AI model server, responsible for processing translation requests (voice-to-sign and sign-to-text).

- **WebRTC**: A real-time communication framework enabling secure, peer-to-peer video calls with low latency.

- **SharedPreferences**: A local storage solution for caching user preferences and lightweight data, improving offline functionality.

This technology stack ensures a modular and maintainable codebase, with each component optimized for its specific role in the app's ecosystem.

3 System Architecture

The Together App employs a modular, client-server architecture to deliver a seamless user experience. The architecture is divided into three primary layers:

- **Frontend (Flutter)**: The client-side layer, built with Flutter, handles the user interface, navigation, and local logic. It communicates with backend services to fetch and display data, such as sign language videos and user profiles.

- **Backend (Firebase)**: Firebase serves as the cloud-based backbone, managing user authentication, data storage, and file hosting. Firestore ensures real-time synchronization of user data, while Firebase Storage securely stores media files.

- **AI Server (Flask)**: A dedicated Flask server processes AI-driven translation tasks. It receives audio or video inputs, applies machine learning models for translation, and returns the results (e.g., sign language videos or text predictions).

Additionally, WebRTC facilitates encrypted, peer-to-peer video communication, allowing users to conduct secure meetings with real-time translation. The architecture is designed for scalability, with Firebase handling dynamic workloads and the Flask server optimized for AI inference tasks.

4 Feature Descriptions

The Together App offers a rich set of features designed to enhance communication and accessibility. Below is a detailed description of each feature:

4.1 Splash Screen

The splash screen serves as the app's entry point, displaying the Together App logo and initializing the application. It checks the user's authentication status using Firebase Authentication and routes them to the appropriate screen (login or home). The splash screen ensures a smooth onboarding experience by handling initial setup tasks, such as

loading cached data via SharedPreferences.

4.2 Home Screen

The home screen is the central hub of the app, providing quick access to key functionalities:

- **Translation Tools**: Buttons to access word-to-sign, voice-to-sign, and sign-to-word translation features.
- **Video Meeting Setup**: An option to initiate or join a secure video call.
- **Navigation Bar**: Links to user profile, favorites, and history sections.

The home screen is designed for intuitive navigation, with a clean layout and prominent call-to-action buttons to guide users to their desired features.

4.3 Login and Signup

The app uses Firebase Authentication for secure user management. The login screen allows existing users to sign in with their email and password, while the signup screen collects additional details, such as:

- User role (e.g., deaf, hard-of-hearing, or general user).
- Date of birth (DOB) for personalized content recommendations.

Upon successful signup, user data is stored in Firestore, and the app navigates to the home screen. A link to switch between login and signup screens enhances usability.

4.4 Profile Management

The profile management feature allows users to view and edit their personal information, including name, email, role, and DOB. Editable text fields are synced with Firestore in real-time, ensuring data consistency across devices. This feature empowers users to maintain up-to-date profiles, which are used to personalize the app experience, such as tailoring sign language recommendations.

4.5 Favorites

The favorites screen displays a list of user-saved signs, retrieved from both local storage (SharedPreferences) and Firestore. Users can:

- Search for specific signs using a search bar.
- Remove signs from their favorites list.
- View detailed sign information, including videos and descriptions.

This feature enhances user engagement by allowing quick access to frequently used signs, with seamless integration between local and cloud data.

4.6 Word to Sign

The word-to-sign feature enables users to input text and receive corresponding sign language videos. The process involves:

- Users enter a word or phrase in a search bar.
- The app queries Firestore for matching sign data, including videos, descriptions, and related signs.
- Users can download videos for offline use or add signs to their favorites.

This feature is powered by a comprehensive sign language dictionary, making it a valuable learning tool for users unfamiliar with sign language.

4.7 Voice to Sign

The voice-to-sign feature allows users to record or upload audio, which is translated into sign language videos. The workflow is as follows:

- Users record audio using the app's built-in recorder or upload an audio file.
- The audio is sent to the Flask server, where an AI model processes it to generate a sign language video.
- The video is returned to the app and displayed to the user.

This feature leverages advanced speech recognition and AI translation, providing real-time communication support for users who rely on sign language.

4.8 Sign to Word

The sign-to-word feature enables users to upload or record a sign language video and receive a text prediction. The process includes:

- Users record a video using the app's camera or upload a pre-recorded video.
- The video is sent to the Flask server, where an AI video inference pipeline analyzes the signs and generates a text prediction.
- The predicted text is displayed to the user, with options to save the prediction to their history.

This feature is particularly useful for deaf or hard-of-hearing users communicating with non-signers, as it translates sign language into written text in real time.

4.9 Meeting Setup and Room

The meeting setup feature allows users to initiate or join a video call. Key functionalities include:

- **Camera Preview**: Users can preview their camera feed and toggle audio/video settings.
- **Room Code**: A unique code is generated for each meeting, which users can copy or share to invite others.
- **Confirmation**: Starting the call transitions to the video call room with real-time communication.

This feature uses WebRTC for secure, peer-to-peer connections, ensuring privacy and low latency during calls.

4.10 In-call Chat and Video

The in-call video room provides an interactive environment for real-time communication. Features include:

- **Local and Remote Streams**: Displays video feeds for all participants.
- **Mic and Camera Toggles**: Users can enable or disable their microphone and camera during the call.
- **In-room Chat**: A text-based chat system allows users to send messages during the call, with messages displayed in a scrollable chat window.

The in-call experience is enhanced by real-time translation capabilities, where spoken or signed inputs are translated to facilitate communication between users with different communication preferences.

4.11 App Screens

The Together App's user interface is designed for accessibility and ease of use, with each screen tailored to specific functionalities. Below is a detailed description of the key screens, accompanied by their respective screenshots where available:

4.11.1 Splash Screen

The splash screen is the first screen users see upon launching the app. It prominently displays the Together App logo, featuring a circular emblem with interconnected hands, symbolizing unity and communication. The screen performs essential initialization tasks, such as checking the user's authentication status via Firebase Authentication. Based on this, it routes users to either the login screen (for unauthenticated users) or the home screen (for authenticated users). The screen uses SharedPreferences to load cached user preferences, ensuring a fast and seamless startup experience. The design is minimalistic,

with a light background featuring a subtle pattern of small blue dots, and no loading indicator is visible in this static view.

4.11.2 Home Screen

The home screen serves as the app's central navigation hub, designed for intuitive access to core features. It features a clean layout with large, tactile buttons for initiating a new meeting ("NEW MEETING") and accessing the word-to-sign translation feature ("WORD TO SIGN"), which also displays a 5-star rating. A bottom navigation bar labeled "TOGETHER" includes icons for Home, Favorites, and Profile sections. The screen uses a light color scheme with a subtle dotted background pattern, and the buttons are prominently labeled with bold text and icons (e.g., a hand gesture for "WORD TO SIGN"). Accessibility features like high-contrast text and scalable fonts accommodate diverse user needs.

4.11.3 Login Screen

The login screen features input fields for email and password, with a "Log In" button that triggers Firebase Authentication. A "Forgot Password?" link and a "Sign Up" link provide additional options. The screen uses a consistent design with clear labels, error messaging for invalid inputs, and a back button to navigate to the signup screen, ensuring a smooth onboarding process.

4.11.4 Signup Screen

The signup screen collects user details, including user role (e.g., deaf, hard-of-hearing, general user) and date of birth, stored in Firestore upon submission. It includes a "Sign Up" button and a back button to navigate to the login screen. The design is consistent with the login screen, with clear labels and error messaging for invalid inputs.

4.11.5 Profile Screen

The profile screen displays user information, including name, email, role, and date of birth, in a structured layout. Editable text fields allow users to update their details, with changes synced to Firestore in real-time. The screen includes a "Save" button to confirm edits and a "Log Out" option. The design emphasizes clarity, with sectioned fields and a user avatar placeholder, enhancing personalization and usability.

4.11.6 Favorites Screen

The favorites screen presents a scrollable list of saved signs, each displayed as a card with the sign name and a favorite icon. The provided screenshot shows one saved sign, "Hello," with a red heart icon indicating it's favorited. A search bar at the top, labeled "Search favorites...," allows users to filter signs, querying both local SharedPreferences and Firestore data. Each card includes options to view details or remove from favorites (though only the heart icon is visible in the screenshot). The screen's layout is list-based with a light background and subtle dotted pattern, ensuring accessibility with tappable cards and clear typography.

Figure 1: Splash Screen of the Together App, displaying the logo with interconnected hands.

Figure 2: Home Screen of the Together App, showcasing buttons for starting a new meeting and accessing translation tools.

Figure 3: Login Screen of the Together App, featuring email and password input fields.

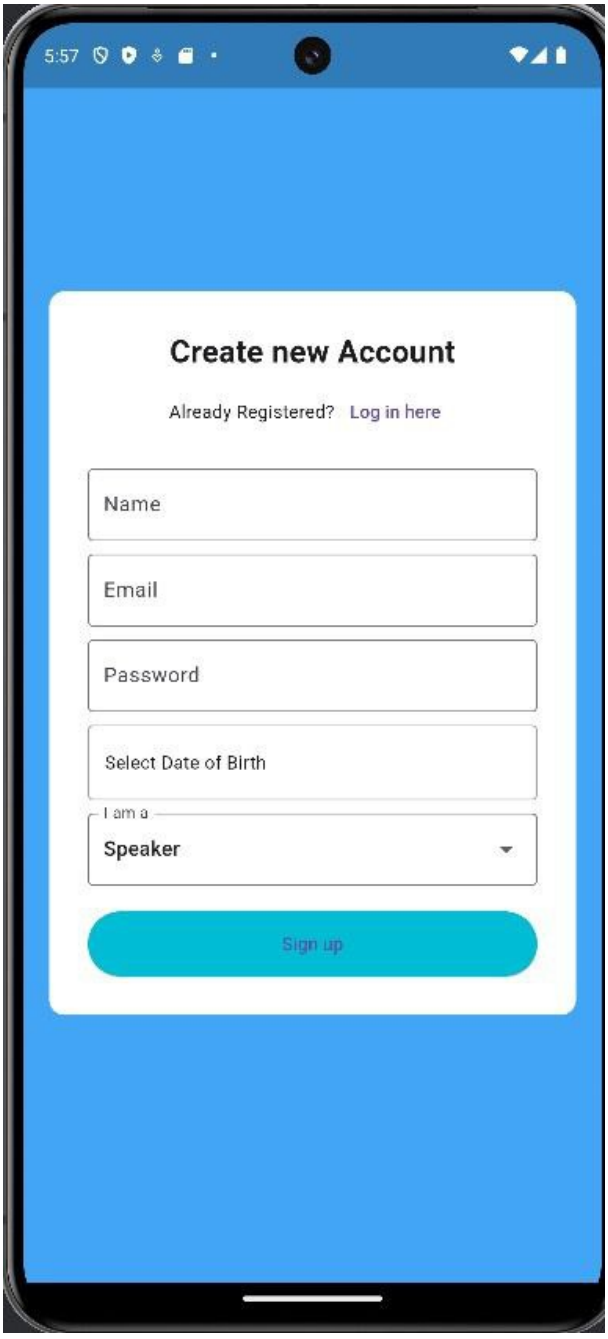

Figure 4: Signup Screen of the Together App, collecting user role and date of birth.

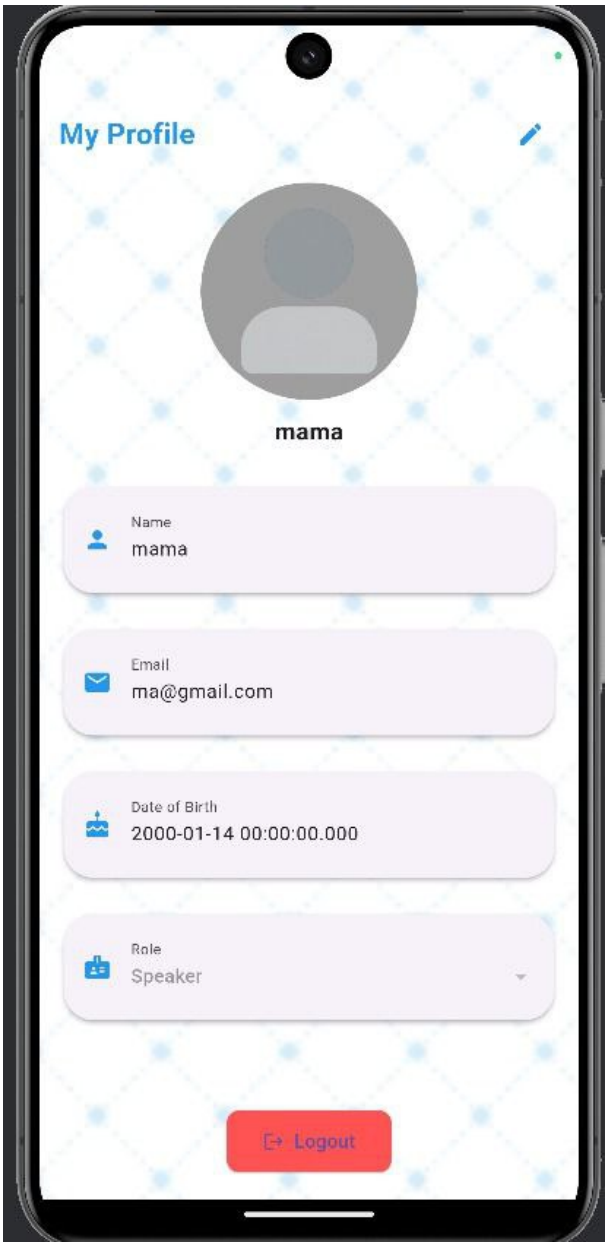

Figure 5: Profile Screen of the Together App, displaying editable user information.

Figure 6: Favorites Screen of the Together App, showing a list with the saved sign “Hello.”

4.11.7 Word to Sign Screen

The word-to-sign screen features a search bar where users input text to query the sign language dictionary. Results are displayed as a card with a sign video, textual description, and related signs. Buttons allow users to download the video for offline use or add the sign to favorites. The screen includes a "Clear" button to reset the search and a back button for navigation. The design prioritizes quick access to results, with a responsive layout that adapts to different screen sizes.

4.11.8 Voice to Sign Screen

The voice-to-sign screen provides options to record or upload audio for translation into sign language videos. The screenshot shows two buttons: "Upload Audio" and "Record Voice." Below these, a recording interface is active, displaying "Record your voice message" with a waveform animation, a timer set at "00:00," and buttons for "Start Recording" (in red) and "Stop Recording" (in gray, currently disabled). The screen's interface is streamlined, with a light background and subtle dotted pattern, large buttons, and a progress indicator during recording, ensuring ease of use for all users.

4.11.9 Sign to Word Screen

The sign-to-word screen allows users to record or upload a sign language video for text prediction. It includes a "Record Video" button to access the device camera and an "Upload Video" button for file selection, with drag-and-drop support. After processing by the Flask server's AI pipeline, the predicted text is displayed, with an option to save the prediction to history. The screen's design emphasizes simplicity, with a preview window for recorded videos and clear feedback during processing.

4.11.10 Meeting Setup Screen

The meeting setup feature includes two related screens: the initial setup screen and the new meeting configuration screen.

Start or Join Meeting Screen: This screen allows users to either start a new meeting or join an existing one. The screenshot shows the Together App logo at the top, followed by the title "Start or Join Meeting." Two buttons are provided: "New Meeting" and "Join Meeting," both with icons (a camera for "New Meeting" and a link for "Join Meeting"). The screen has a light background with a dotted pattern, maintaining a consistent design theme.

New Meeting Screen: After selecting "New Meeting," users are taken to a screen to configure the call. The screenshot shows the title "NEW MEETING" with a camera preview area (currently blank in the image). Below this, the "WORD TO SIGN" button is visible, indicating additional services accessible from this screen. The bottom navigation bar includes Home, Favorites, and Profile options, consistent with the home screen.

Note that the new meeting screen in the screenshot does not fully align with the described functionality (e.g., camera preview and room code sharing), suggesting this image may represent an earlier stage of the meeting setup process. The description below reflects the intended functionality.

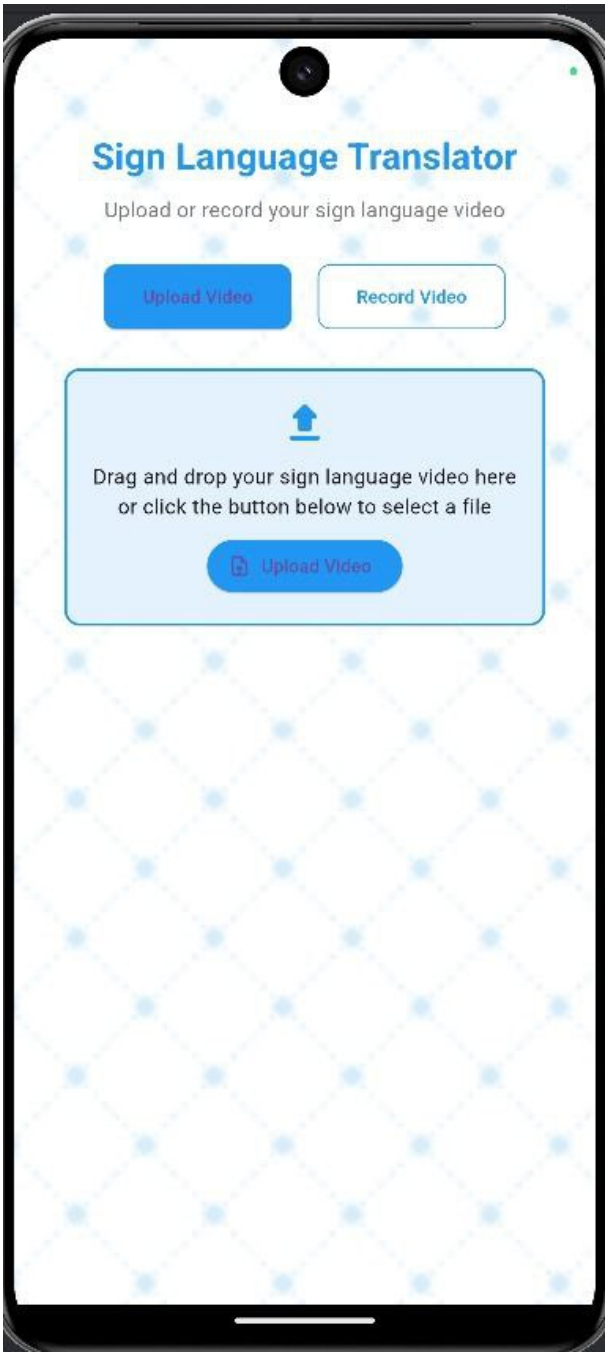

Figure 7: Word to Sign Screen of the Together App, displaying a sign language video result.

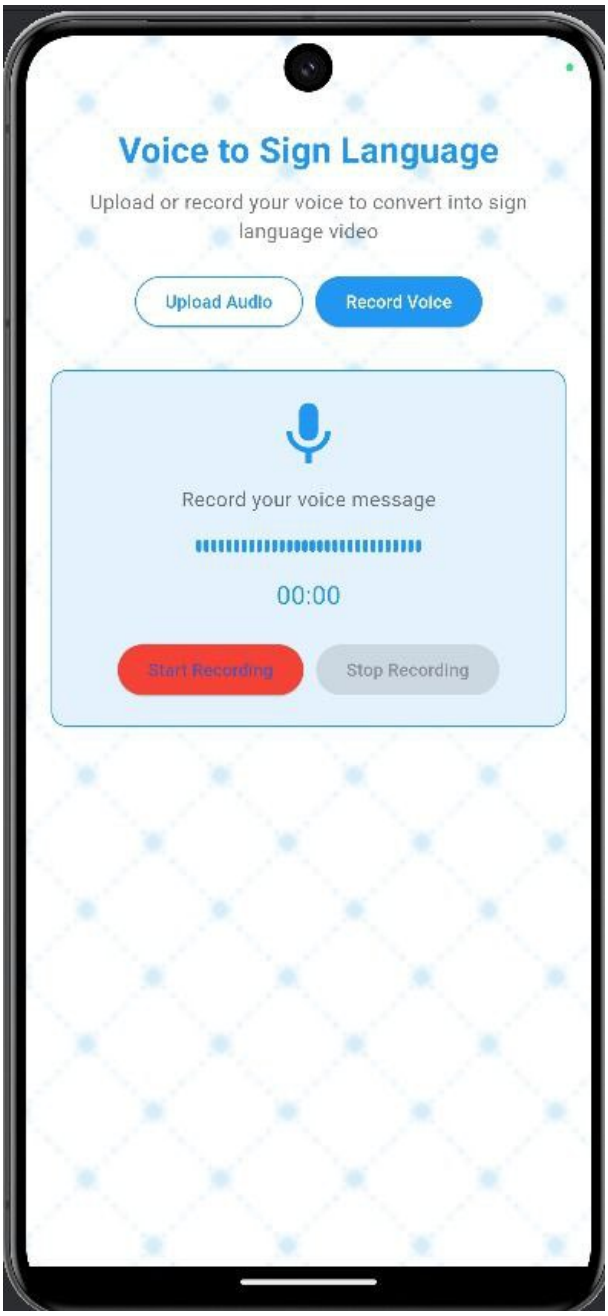

Figure 8: Voice to Sign Screen of the Together App, featuring audio recording and upload options.

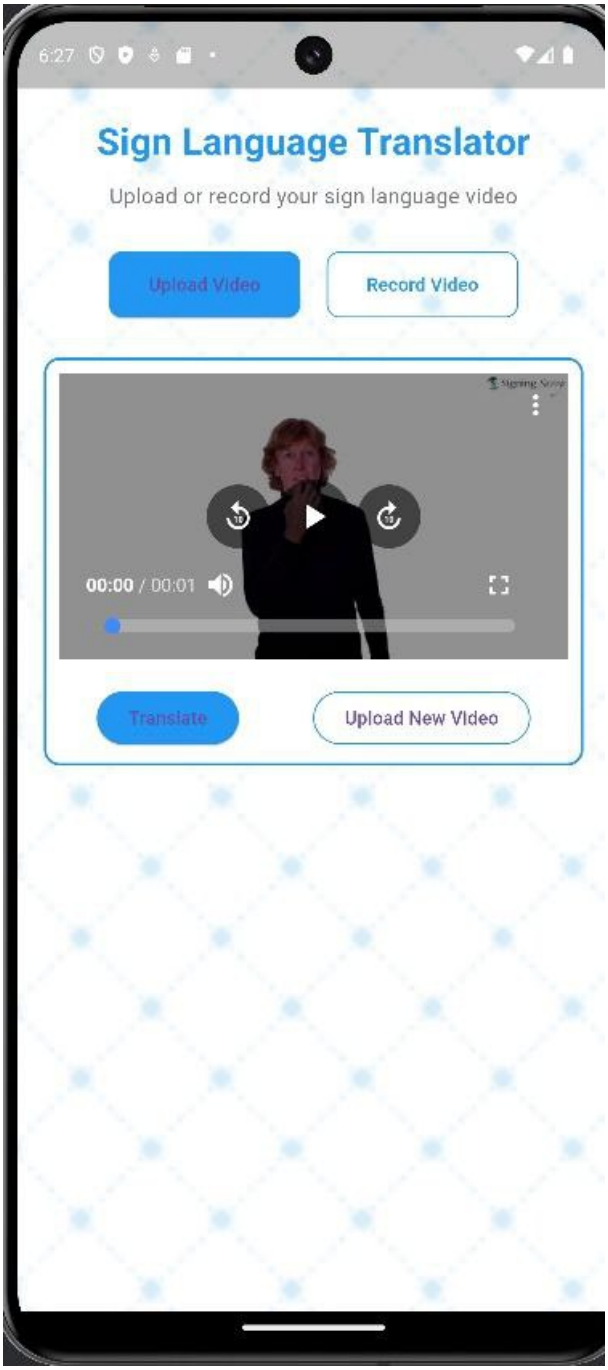

Figure 9: Sign to Word Screen of the Together App, showing video recording and upload options.

Figure 10: Start or Join Meeting Screen of the Together App, offering options to start or join a meeting.

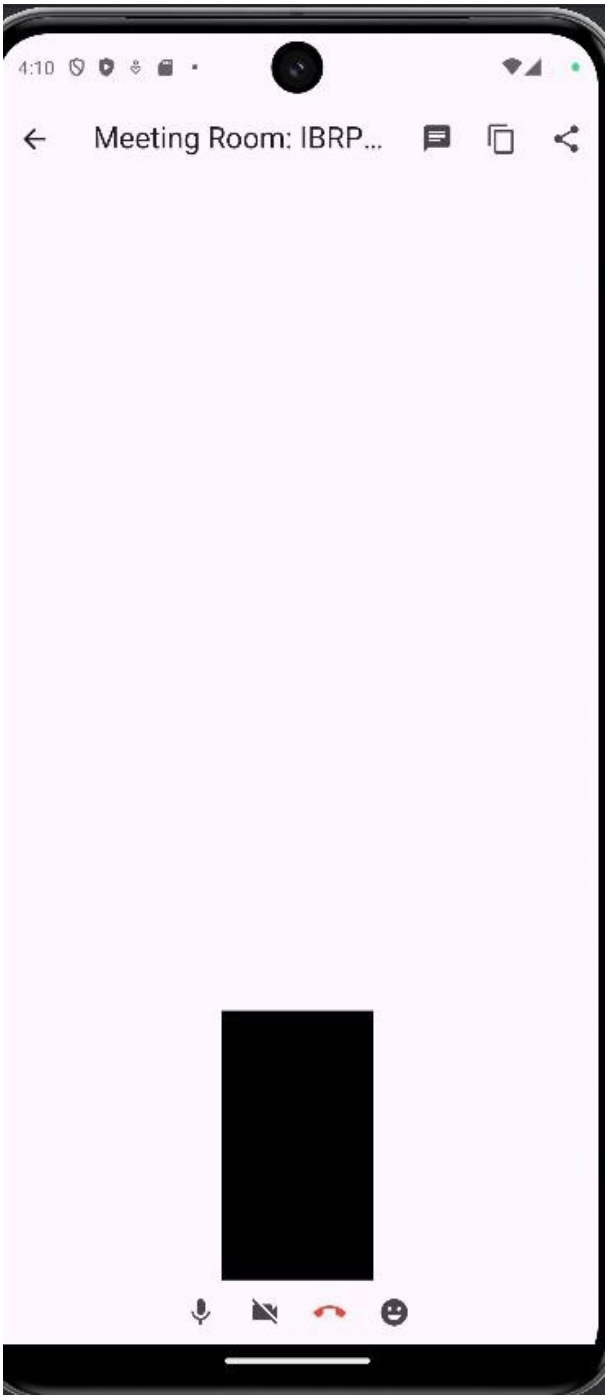

Figure 11: New Meeting Screen of the Together App, displaying the meeting setup interface.

The meeting setup screen enables users to configure a video call. It features a camera preview window with toggle switches for audio and video, a generated room code with "Copy" and "Share" buttons, and a "Start Call" button to enter the call room. The screen includes a "Cancel" button to return to the home screen. The layout is compact, with centered controls and high-contrast elements to ensure accessibility.

4.11.11 In-call Video and Chat Screen

The in-call screen provides a real-time video call interface. The screenshot shows the title "Meeting Room: IBRP..." with icons for microphone, camera, and sharing options. The video stream area is blank (no video feeds are displayed), but a chat window at the bottom shows a message "mama: hellooooo" and a text input field labeled "Type a message" with a send button. The screen supports real-time translation overlays, though not visible in this static view. The design prioritizes clarity, with a split layout for video streams (when active) and a scrollable chat history.

5 Conclusion

The Together App represents a significant advancement in mobile computing for accessibility. By integrating AI-powered translation, real-time video communication, and robust user management, the app addresses critical communication barriers faced by the deaf and hard-of-hearing community. Key achievements include:

- **Inclusive Design**: A user-friendly interface tailored to diverse communication needs.
- **Real-time Translation**: AI-driven features that enable seamless speech-to-sign and sign-to-text conversion.
- **Secure Communication**: Encrypted video calls and robust authentication for user privacy.
- **Scalable Architecture**: A modular design that supports future enhancements and scalability.

The project demonstrates the application of mobile computing principles, including cross-platform development, cloud integration, and AI-driven functionality. The Together App not only serves as a practical tool for communication but also sets a foundation for future innovations in accessible technology.

6 Appendix: Code Samples

The following code samples illustrate key functionalities of the Together App, with detailed comments to explain the implementation.

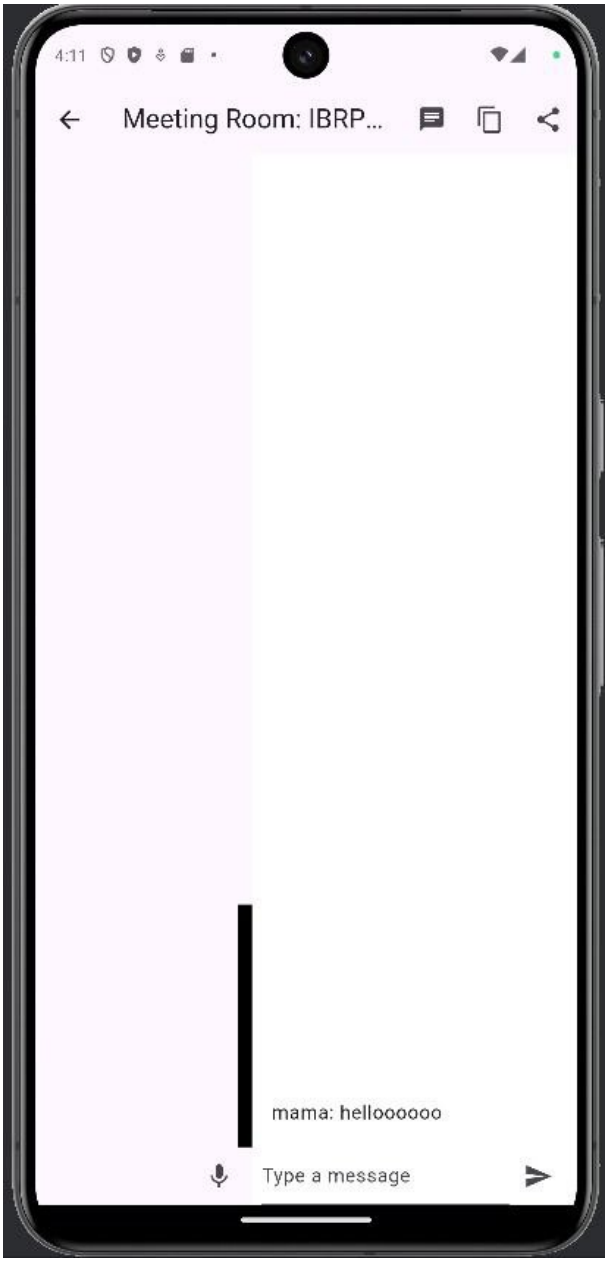

Figure 12: In-call Video and Chat Screen of the Together App, showing the chat window with a message.

6.1 AuthService: Login and Signup

This code handles user authentication using Firebase Authentication, including login and signup functionalities.

```
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Login method to authenticate existing users
  Future<String?> login({
    required String email,
    required String password,
  }) async {
    try {
      // Attempt to sign in with email and password
      await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return "Success"; // Indicate successful login
    } on FirebaseAuthException catch (e) {
      return e.message; // Return error message if authentication fails
    }
  }

  // Signup method to create new users
  Future<String?> signup({
    required String email,
    required String password,
  }) async {
    try {
      // Create a new user with email and password
      UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return "Success"; // Indicate successful signup
    } on FirebaseAuthException catch (e) {
      return e.message; // Return error message if signup fails
    }
  }
}
```

6.2 ApiService: Audio and Video Upload

This code handles uploading audio and video files to the Flask server for translation processing.

```
import 'package:http/http.dart' as http;
import 'dart:io';

class ApiService {
  static const String baseUri = 'http://your-flask-server.com/api';

  // Upload audio file for voice-to-sign translation
  static Future<String?> uploadAudio(File audioFile) async {
    final uri = Uri.parse('$baseUri/voice2sign');
    final request = http.MultipartRequest('POST', uri);
    request.files.add(await http.MultipartFile.fromPath('audio', audioFile.path));
    final response = await request.send();
    return response.statusCode == 200
        ? await response.stream.bytesToString()
        : 'Upload failed';
  }

  // Upload video file for sign-to-word translation
  static Future<String?> uploadSignVideo(File videoFile) async {
    final uri = Uri.parse('$baseUri/translate');
    final request = http.MultipartRequest('POST', uri);
    request.files.add(await http.MultipartFile.fromPath('video', videoFile.path));
    final response = await request.send();
    return response.statusCode == 200
        ? await response.stream.bytesToString()
        : null;
  }
}
```

6.3 FirestoreService: Manage Favorites and Predictions

This code manages saving favorites and translation predictions to Firestore.

```
import 'package:cloud_firestore/cloud_firestore.dart';

class FirestoreService {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  // Save a sign to the user's favorites collection
  Future<void> saveFavorite({
    required String userId,
    required String signName,
  }) async {
    await _firestore
```

```
        .collection('users')
        .doc(userId)
        .collection('favorites')
        .add({
      'signName': signName,
      'timestamp': FieldValue.serverTimestamp(),
    });
  }

  // Save a translation prediction to the user's predictions collection
  Future<void> savePrediction({
    required String userId,
    required String prediction,
  }) async {
    await _firestore
        .collection('users')
        .doc(userId)
        .collection('predictions')
        .add({
      'prediction': prediction,
      'timestamp': FieldValue.serverTimestamp(),
    });
  }
}
```

6.4 WebRTCService: Initialize Video Call

This code initializes a WebRTC video call using the flutter$_{w}ebrtcpackage$.

```
import 'package:flutter_webrtc/flutter_webrtc.dart';

class WebRTCService {
  final RTCPeerConnection _peerConnection;
  final RTCVideoRenderer _localRenderer = RTCVideoRenderer();
  final RTCVideoRenderer _remoteRenderer = RTCVideoRenderer();

  WebRTCService(this._peerConnection);

  // Initialize local and remote video renderers
  Future<void> initializeRenderers() async {
    await _localRenderer.initialize();
    await _remoteRenderer.initialize();
  }

  // Create an offer for the video call
  Future<void> createOffer() async {
    RTCSessionDescription offer = await _peerConnection.createOffer({
      'offerToReceiveAudio': 1,
```

```
      'offerToReceiveVideo': 1,
    });
    await _peerConnection.setLocalDescription(offer);
    // Send offer to the remote peer (implementation depends on signaling server)
  }
}
```