SLution - Real-Time Sign Language Translation Application

Submitted By:

Ofir Evgi - 207441346

Omer Shukroon - 208540856

Table of Contents

Literature Review + Competitors

Introduction:

Sign language is an essential means of communication for over 70 million people worldwide, including approximately 10,000 to 15,000 individuals in Israel who use Israeli Sign Language (ISL). While there are over 300 unique sign languages globally, communication barriers still persist between sign language users and the general public, making daily interactions challenging.

Our application provides a practical technological solution to this problem by recognizing American Sign Language (ASL) gestures and translating them into English letters in real time. The system focuses on identifying individual letters from static hand signs using image-based detection. The app is designed to support smoother communication in environments such as schools, workplaces, and public settings, promoting greater accessibility through simple, real-time gesture-to-text translation.

Motivation:

In today's digital era, challenges and opportunities coexist in the pursuit of promoting accessibility and equality. For the deaf and hard-of-hearing community, the lack of widespread understanding of sign language by the public creates significant communication and social barriers. This highlights the need for simple and accessible technological solutions that can bridge these gaps.

Our proposed application for real-time sign language recognition and text translation aims to address these challenges. By enabling smoother daily interactions in various institutions, from clinics to workplaces, the app strives to foster genuine connections and make communication more inclusive through smart and innovative technology.

Literature Review – Sign Language Translation:

Communication between sign language users and people who are not familiar with it is a daily challenge for the disability community. Currently, there are applications in the market that offer sign language learning or basic translation of individual words, but there is no solution that provides continuous and complete real-time translation of sign language into text, our application aims to bridge this gap.

This review will examine the existing solutions in the market, their limitations, and the opportunity our application offers to improve accessibility and understanding between people.

Sign language recognition systems are becoming essential tools to bridge communication gaps between hearing-impaired individuals and the broader community. Across multiple studies, innovative technologies like Convolutional Neural Networks (CNNs), Random Forest classifiers, and image processing methods have been developed to address the challenges in this field. Each research offers unique solutions, yet many shares similar technologies, focusing on gesture detection, feature extraction, and real-time processing.

In [2], the researchers designed a system combining the YOLOv5 object detection model and Random Forest Classifier. YOLOv5 excels in detecting ASL gestures, achieving an impressive mAP of 0.995 and recall of 1, while the Random Forest Classifier provides precision of 98% in gesture interpretation. The integration of Generative AI (GenAI) enhances the system's ability to form sentences from recognized signs, further converting them to speech for seamless communication. This dual-model approach tackles challenges like diverse gestures and processing speed, achieving over 95% accuracy in real-time scenarios. Similarly, in [3], CNNs were used to translate ASL gestures into text, focusing on feature extraction from video sequences. While both studies employ advanced machine learning, [3] specifically highlighted the efficiency of pooling and convolutional layers in processing inputs. Lighting and background variations posed challenges, but preprocessing methods helped the system reach accuracy above 90%.

In [4], a system based on Python and OpenCV utilized CNNs to classify hand gestures detected from real-time video. The project focused on user-friendly Human-Computer Interfaces (HCI), avoiding complex environments like uniform backgrounds or gloves. Like [3], it emphasized real-time gesture detection but also converted signs into both text and speech. With over 90% accuracy, this approach demonstrates the growing potential of neural networks for enhancement of communication. Meanwhile, [1] proposed a CNN-based model for real-time translation, combining hand gesture recognition with text and speech output. This study uniquely highlighted the use of multiple convolutional layers, focusing on detecting dynamic and static gestures. The system achieved 90% accuracy, addressing challenges like lighting inconsistency and complex gestures with advanced preprocessing.

In summary, all studies share a focus on using CNNs for their ability to extract visual features and classify gestures effectively. Random Forest classifiers, as seen in [2], and

OpenCV methods, as used in [4], provide alternative or complementary strategies to enhance gesture detection and recognition. Real-time translation into text and speech is a common feature across these systems, highlighting their practical applications in education, healthcare, and daily communication for the hearing-impaired community. Despite differences in datasets and specific technologies, these studies collectively demonstrate how combining machine learning models with preprocessing strategies can achieve highly accurate, inclusive, and accessible solutions.

References:

[1] - Ojha, A., Pandey, A., Maurya, S., Thakur, A., & Dayananda, P. (2023). Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network

[2] - Joshi, H., Gundawar, J., Golhar, V., Gangurde, A., & Yenikkar, A. (2024). Real-Time Sign Language Recognition and Sentence Generation

[3] - Girija, V., Singh, A. K., Sahil, N., Singh, S., & Paul, T. (2024). Sign Language to Text Conversion Using CNN

[4] - Priya, S. A., Nair, A., Madhavan, S., & Issac, K. (2023).

Competitors

SLution	Pocket Sign	Sign Language AI	Issiesign	Deaf Sign	Application / Feature
Free	cheap	Free	Free	Free	**Price**
ASL	ASL	ASL	ISL	ASL	**Languages**
8.0	8.5	9.5	7.5	6.5	**Word Database**
Yes	No	Yes	No	No	**Real-Time Translation**
9.0	9.0	5.5	6.5	8.0	**User-Friendly Interface**
Yes	Yes	Yes	No	No	**Real-Time Hend Detection**

ASL - American Sign Language

Deaf Sign - The application focuses on basic learning of sign language in English. The user interface is very clear and user-friendly but has a small database that includes numbers, letters, days, and about nine general words such as "yes", "no" and "hello." The application allows users to browse through the words, and upon selecting a word from the database, a short video appears demonstrating the gesture that represents the word in sign language. Additionally, the app provides an option to take a short test.

Sign Language AI - The application focuses on translating sign language into text in English. The user interface is clear but not very inviting or well-organized. However, it has a large database that includes almost every word related to daily conversations and beyond. The application offers three main options:

Real-time translation from sign language to English:
The app uses the camera to recognize hand movements and translates them into words (translating one word at a time).

Translation from English to sign language:
After typing a desired word, a short video appears demonstrating the gesture in sign language.

Browsing and searching words in the general word database:
Users can browse through the entire word database using search or filter by category.

Pocket Sign - The application focuses on teaching sign language in English. The user interface is very clear and user-friendly, with a large database that includes almost every word necessary to learn sign language. The lessons are organized by topics such as objects, food, actions, measurements, clothing, etc., which helps track progress in learning the language.

The application allows users to search for a word in the existing database, displaying a short video that demonstrates the gesture representing the word in sign language. Additionally, the app provides an option to join a lesson on a specific topic.

Another feature is the sign language recognition mode. In this mode, words and videos of their corresponding gestures in sign language are displayed based on a specific topic. Using the phone's camera, the app detects the user’s gestures, and when the user performs the gesture correctly, the system recognizes it and moves on to the next word.

IssieSign - This app is designed for learning Israeli Sign Language (ISL). The user interface of the app is quite basic, but the words are organized into categories. For example, if we want to learn how to say "nose" or "mouth," we can select the "Body Parts" category, and a video will then demonstrate how to sign that word in ISL.

Users have the option to create their own word folder, compiling a collection of words they choose, as well as a "Favorites" folder.

The app's vocabulary covers essential words that are important to know, and users can search for specific words. Additionally, users can add new words to the vocabulary by recording a video of themselves signing the word.

SLution – In conclusion, there are apps designed for learning sign language that make it possible for more people to learn the language and communicate with individuals with disabilities. However, these apps have a limited target audience, which still leaves communication between sign language speakers and non-speakers lacking. There are also translation apps, but they are often limited to translating individual words, making it harder to bridge the communication gap between these two worlds.

Each app currently has its own disadvantages, whether it's a small database, an uninviting or unintuitive UI, or the lack of real-time hand recognition. This is where SLution comes into

action, offering a system that enables real-time translation of sign language gestures into individual letters. Our app combines this with a user-friendly and clear UI, allowing users to track recent conversations, save favorite phrases, and directly translate text into multiple languages. It also features the ability to expand the database over time.

We believe that with our solution, we can help overcome these communication barriers and provide people with disabilities the ability to connect with the world more easily and confidently, without fear or shame.

Detailed Requirements

Stakeholders

End Users:

- Individuals with disabilities who communicate using sign language.
- People who want to communicate with sign language users (e.g., family, friends and colleagues)
- Government institutions and large companies that aim to make their services more accessible to individuals with disabilities in an easy and efficient way.

Administrators:

Includes developers, investors, and system managers responsible for:

- Managing and maintaining the application's database, infrastructure, and features.
- Monitoring system performance, fixing bugs, and ensuring application quality.
- Overseeing the project's success and its future development.

Functional Requirements

End Users:

Registration and Login:

- Users can register in the app with personal details (e.g., email and password).
- Users can securely log into the app.

User Profile Management:

- Users can update their personal details (e.g., name, email, password).
- When the user selects their preferred sign language in their profile, the system will automatically set the default text language for conversations to the chosen language.

Navigation from the Home Screen:

Users can access the following functions from the home screen:

Conversation History:

- Users can browse saved conversations.
- Users can resume a saved conversation.
- Users can mark sentences as favorites for future use.

Favorite Sentences:

- Users can view a list of sentences marked as favorites.
- Users can remove sentences from the list.

Starting a New Conversation:

- Users can start a new conversation.

In the conversation:

- Users can translate sign language into text in real-time using the camera.
- Users can type messages using the keyboard.
- Users can convert speech to text.
- Users can access the list of favorite sentences for quick use of saved phrases.

Conversation Translation Screen:

The screen is divided into two sections:

- Top section: Displays the camera view (front or rear).
- Users can see themselves performing sign language.
- Users can approve or edit the translated sentence.

Administrators:

Database Management:

Update and manage the database of gestures and sentences.

User Management:

Manage user accounts (e.g., resetting passwords, deleting accounts).

System Maintenance:

Monitor and fix errors related to gesture recognition or system performance.

Feature Development:

Test and implement new features based on user feedback.

Non-Functional Requirements

Performance:

- The system will process sign language gestures in real-time with a delay of no more than 5-10 seconds.
- The system will achieve at least 90% accuracy in gesture recognition under standard conditions.

Scalability:

- The system will support increased user loads without performance degradation.
- The database of gestures and sentences can be expanded to include additional signs or languages.

Reliability:

- The system will ensure 99.9% availability for continuous operation.
- Backup mechanisms will ensure recovery from unexpected failures.

Security:

- All user data, including personal details and translations, will be encrypted during transmission and storage.
- The application will comply with relevant data security regulations (e.g., GDPR).

Usability:

The app interface will be intuitive and suitable for users with varying levels of technological proficiency.

Accessibility features will include:

- Font size adjustment.
- High contrast themes.

Compatibility:

- The app will be compatible with Android.
- The app will support front and rear cameras with various resolutions.

Maintenance:

- The app's architecture will be modular, allowing components to be updated without disrupting the entire system.

- The code will be well-documented to facilitate future development and bug fixes.

Localization:

- The system will support English in the user interface.

Data Management:

- The system will automatically save the last 10 conversations.

- Sentences marked as favorites will be stored separately for quick retrieval.

Detailed Design Document

1. Introduction:

1.1 Purpose:

This document defines the detailed design of the Real-Time Sign Language Translator application. The purpose of the application is to enable users to translate sign language gestures into text, facilitating communication for individuals with disabilities and their broader communities.

1.2 Overview:

This document provides a detailed design for the Real-Time Sign Language Translator application. The system facilitates real-time gesture recognition and translation into text, enabling effective communication between sign language users and non-users. It details the system components, their interfaces, the data flow, and the dependencies required for successful implementation and operation.

2. System Overview:

The Real-Time Sign Language Translator is composed of several core components that work together to provide real-time translation of sign language letters into text. The User Interface (UI) is developed as an Android application, offering users a smooth and accessible platform for interaction. It supports user authentication, profile management, and gesture recognition using the device's camera.
The gesture recognition system is embedded directly within the mobile application and relies on an **ensemble of four Convolutional Neural Network (CNN) models** that were trained together. For each frame captured by the camera, the system uses a softmax layer in each model to generate a probability vector over the possible letter classes. The softmax outputs from all four models are averaged to produce a final prediction vector, from which the most probable letter is selected. To ensure translation accuracy and reduce noise, the system displays a letter only after it has been consistently predicted in 10 consecutive frames.
The Firebase backend is used for managing user authentication, storing user profiles, conversation history, and favorite phrases. All gesture processing is performed locally on the device to ensure low latency and support offline usage.
This architecture ensures fast, reliable, and accessible communication for users, especially in environments where internet access may be limited.

2.1 Model Architecture and Datasets

To implement accurate sign language recognition, the SLution application utilizes an **ensemble of Convolutional Neural Network (CNN) models** trained from scratch using two distinct **American Sign Language (ASL)** image datasets. The first dataset, obtained from Kaggle, contains a diverse collection of hand gesture images captured from multiple users under varying lighting conditions, distances, and backgrounds. The second dataset, curated by Krishna Paachajanya, consists of images of a single hand performing each letter with slight variations and is organized into three variants: original images, white-background images with black hand contours, and black background images with white hand contours.

Four CNN models were trained in total and used in the final application. Two models (**sign_cnn_model_tf** and **sign_cnn_model_tf2**) were trained on the first dataset using images resized to 64×64 pixels. Their architecture includes three convolutional layers followed by dense and dropout layers, with differences in training epochs (15 vs. 25) and the use of data augmentation. Two additional models (sign_cnn_model_final_tf and sign_cnn_model_final_tf2) were trained on a merged dataset that combines only the original images from both sources. These models utilize a deeper architecture with four convolutional layers, batch normalization, and larger input dimensions (300×300). They were trained for 20 and 13 epochs respectively, with varying levels of data augmentation.

For each frame captured by the camera, the system uses a **softmax layer** in each model to generate a **probability vector** over the possible letter classes. The softmax outputs from all four models are **averaged** to produce a final prediction vector, from which the **most probable letter** is selected. To ensure translation accuracy and reduce noise, the system displays a letter only after it has been consistently predicted in **10 consecutive frames.**

Two additional models were developed **sign_cnn_dataCE_model_tf** and **sign_cnn_dataAT_model_tf** trained on the specialized contour-based variants of the second dataset. Although these models showed promising results in isolated evaluations, they were not integrated into the app due to the lack of a reliable method for real-time **hand segmentation** or **contour extraction** on mobile devices. Nevertheless, their development reflects an effort to explore and evaluate advanced preprocessing pipelines that could be incorporated in future versions.

All models were developed and trained using **TensorFlow** and deployed to the Android application via **TensorFlow Lite** for efficient on-device inference.

3. Design Considerations:

The system is designed with several practical considerations and constraints to ensure reliable and efficient functionality. It assumes that users have Android devices equipped with a functioning camera and basic processing capabilities sufficient for running on-device machine learning models.

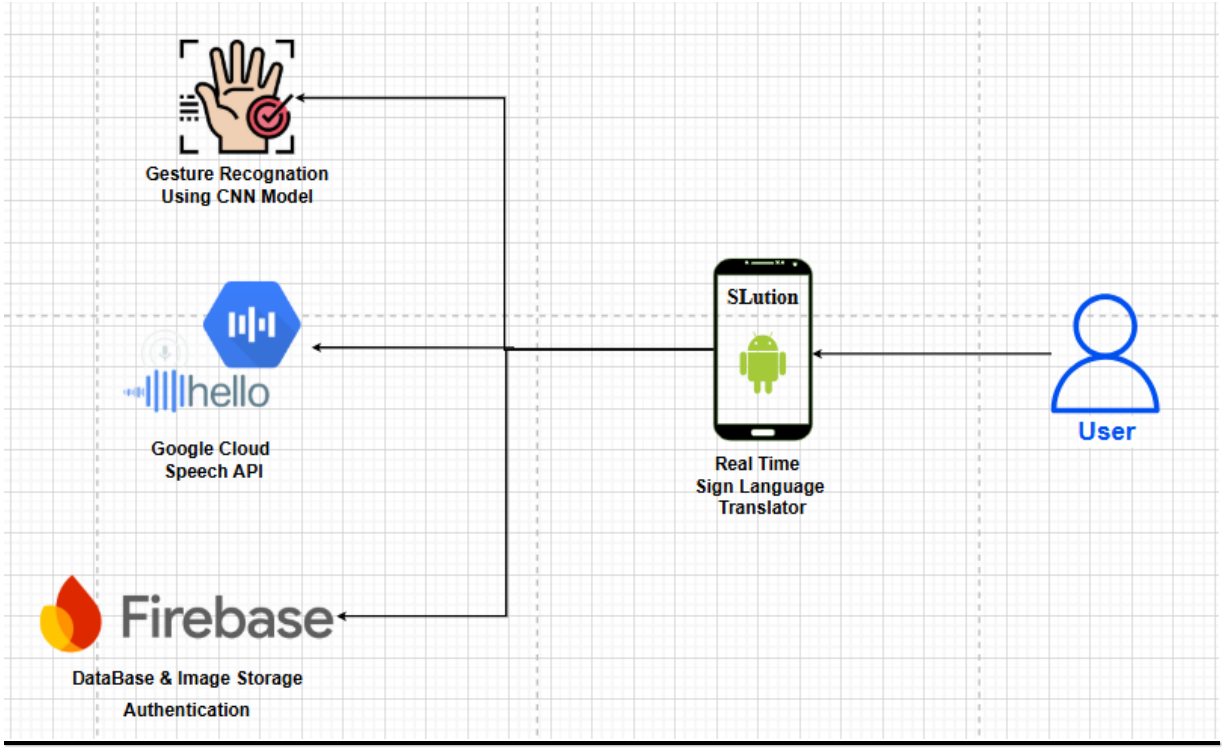
All gesture recognition is performed locally within the mobile application, eliminating the need for external servers or backend processing. The application does not rely on continuous internet access for its core functionality, although connectivity is used for features such as user authentication and cloud data storage via **Firebase**.

To achieve accurate real-time translation, the system processes camera input at high frequency and uses an ensemble of CNN models. A predicted letter is only displayed after consistent detection across 10 consecutive frames, ensuring robustness and reducing false positives.

The application currently **supports American Sign Language (ASL)** only, with the focus on letter-level recognition. Design choices prioritize low-latency processing, user accessibility, and ease of use, particularly for deaf and hard-of-hearing individuals seeking practical communication support.

4. System Architecture:

4.1 High-Level Diagram:



4.2 Components:

The Real-Time Sign Language Translator consists of five main components that work together to deliver smooth and accurate user experience. The User Interface is implemented as an Android application, serving as the primary platform for user interaction. It provides features such as user registration, login, profile management, and real-time gesture recognition using the device's camera, with translations displayed directly on the screen. The gesture recognition is handled by a set of Convolutional Neural Network (CNN) models embedded within the app. These models operate in an ensemble, averaging predictions from each frame and confirming a letter only after it has been detected consistently across 10 consecutive frames, thereby ensuring high accuracy and reliability. For voice-based interaction, the application integrates the Google Cloud Speech-to-Text API, allowing users to convert spoken input into text. Data management, including user authentication, conversation history, and favorite phrases, is handled via Firebase, providing secure and synchronized storage of user information. All components are integrated within the app to provide a fast, responsive, and user-friendly translation experience.

5. Component Design:

The Real-Time Sign Language Translator application is designed with a modular structure, where each component serves a distinct purpose and interacts seamlessly with others. Below is a detailed breakdown of the components, their interfaces, and the flow of data through the system:

5.1 User Interface Module:

Purpose: Provides an interactive platform for users to perform tasks such as logging in, updating profiles, and initiating gesture-based translations.

Displays real-time translations of sign language gestures into text.

Input: User credentials (e.g., email, password), profile updates, and camera-captured gestures.

Output: Translated text from gestures or voice commands.

Dependencies: Relies on the server-side API for gesture translation and Firebase for authentication and data storage.

5.2 Server-Side Module:

Purpose: Acts as the core processing unit, managing communication between the user interface, machine learning models, and Firebase.

Handles data processing, authentication, and coordination of gesture recognition tasks.

Input: Requests from the User Interface (e.g., gesture data, voice input).

Output: Processed and translated text sent back to the user.

Dependencies: Integrates with the Gesture Recognition Module, Google Cloud Speech API, and Firebase.

5.3 Gesture Recognition Module:

Purpose: Utilizes an ensemble of Convolutional Neural Network (CNN) models embedded in the mobile application to detect hand gestures from video frames and translate them into corresponding letters. The system ensures accuracy by requiring consistent detection of the same letter across 10 consecutive frames before displaying it.

Input: Video frames captured via the device camera.

Output: Recognized letters mapped to text.

Dependencies: Operates locally within the Android application, relying on real-time camera input from the User Interface.

5.4 Google Cloud Speech API:

Purpose: Converts spoken input into text, enabling additional interaction methods for users.

Input: Voice commands from the user.

Output: Text representation of the spoken input.

Dependencies: Integrated with the server-side logic to process and manage voice commands.

5.5 Database Module:

Purpose: Powered by Firebase, it securely stores user profiles, conversation histories, gesture data, and favorite sentences.

Input: Data requests for user details, conversations, or favorite phrases.

Output: Data retrieved and provided to the requesting component.

Dependencies: Linked with the User Interface and Server-Side Module for data storage and retrieval.

5.7 Data Flow:

1. Users interact with the User Interface by logging in, capturing gestures, or providing voice input.
2. Gesture data is processed locally on the device using embedded CNN models
3. For voice input, the Google Cloud Speech API converts the audio into text.
4. All user and system data, such as profiles and conversation histories, is securely stored and retrieved from the Database Module.

6. Data Design:

6.1 Database Schema:

Users Table:

- Columns:
  - userID (Primary Key, int): Unique identifier for the user.
  - name (varchar): Full name of the user.
  - email (varchar, unique): Email address of the user.
  - password (varchar): Encrypted password of the user.
  - preferredSignLanguage (varchar): User's preferred sign language.
- Relationships:
  - Linked to the Conversations Table and through userID.

Conversations Table:

- Columns:
  - conversationID (Primary Key, int): Unique identifier for the conversation.
  - userID (Foreign Key, int): References the userID in the Users Table.
  - createdDate (datetime): Date and time the conversation was created.
  - sentences (JSON Array): List of sentences associated with the conversation.
- Relationships:
  - Linked to the Users Table and Sentences Table.

Sentences Table:

- Columns:
    - sentenceID (Primary Key, int): Unique identifier for the sentence.
    - text (varchar): The actual text of the sentence.
    - language (varchar): Language of the sentence.
    - isFavorite (boolean): Whether the sentence is marked as a favorite.
- Relationships:
    - Linked to the Conversations Table.

TextTranslation Table:

- Columns:
    - translationID (Primary Key, int): Unique identifier for the translation.
    - sourceText (varchar): Original text before translation.
    - translatedText (varchar): Translated text.
    - sourceLanguage (varchar): Source language of the text.
    - targetLanguage (varchar): Target language of the text.
- Relationships:
    - Linked to Conversations Table for sentence translations.

Gestures Table:

- Columns:
    - gestureID (Primary Key, int): Unique identifier for the gesture.
    - gesturePattern (varchar): Representation of the gesture (e.g., image path or encoded data).
    - associatedText (varchar): Text associated with the gesture.
- Relationships:
    - Linked to the SignLanguageProcessor.

Admins Table:

- Columns:
    - adminID (Primary Key, int): Unique identifier for the admin.

- name (varchar): Admin's name.
- email (varchar): Admin's email address.

- Relationships:
  - Admins can manage Users Table, Conversations Table, and Gestures Table.

DatabaseManager Table:

- Columns:
  - databaseID (Primary Key, int): Unique identifier for the database manager instance.
  - usersTable (JSON Array): References all users in the system.
  - conversationsTable (JSON Array): References all conversations.
  - sentencesTable (JSON Array): References all sentences.
  - gesturesTable (JSON Array): References all gestures.
  - favoritesTable (JSON Array): References all favorite sentences.
- Purpose:
  - This table acts as an orchestrator, managing data interactions across all other tables.

UImanager Table:

- Columns:
  - uiManagerID (Primary Key, int): Unique identifier for the UI manager instance.
  - themeSettings (varchar): Stores UI theme settings (e.g., dark mode).
  - fontSize (int): User's preferred font size.
- Relationships:
  - Handles settings for the Users Table.

SignLanguageProcessor Table:

- Columns:
  - processorID (Primary Key, int): Unique identifier for the processor instance.
  - language (varchar): Language supported by the processor.
  - gestureDatabase (JSON Array): References gestures used in recognition.

- Relationships:
    - Linked to the Gestures Table for gesture recognition and processing.

VoiceProcessor Table:

- Columns:
    - processorID (Primary Key, int): Unique identifier for the voice processor.
    - translateLanguage (varchar): Target language for voice translation.
    - voiceToSentence (JSON): Voice input converted to sentence format.
- Relationships:
    - Linked to the Conversations Table and Sentences Table for voice-to-text translations.

6.2 Storage Considerations:

Indexing:

- Fields like userID, email, conversationID, and gestureID are indexed to improve query performance.

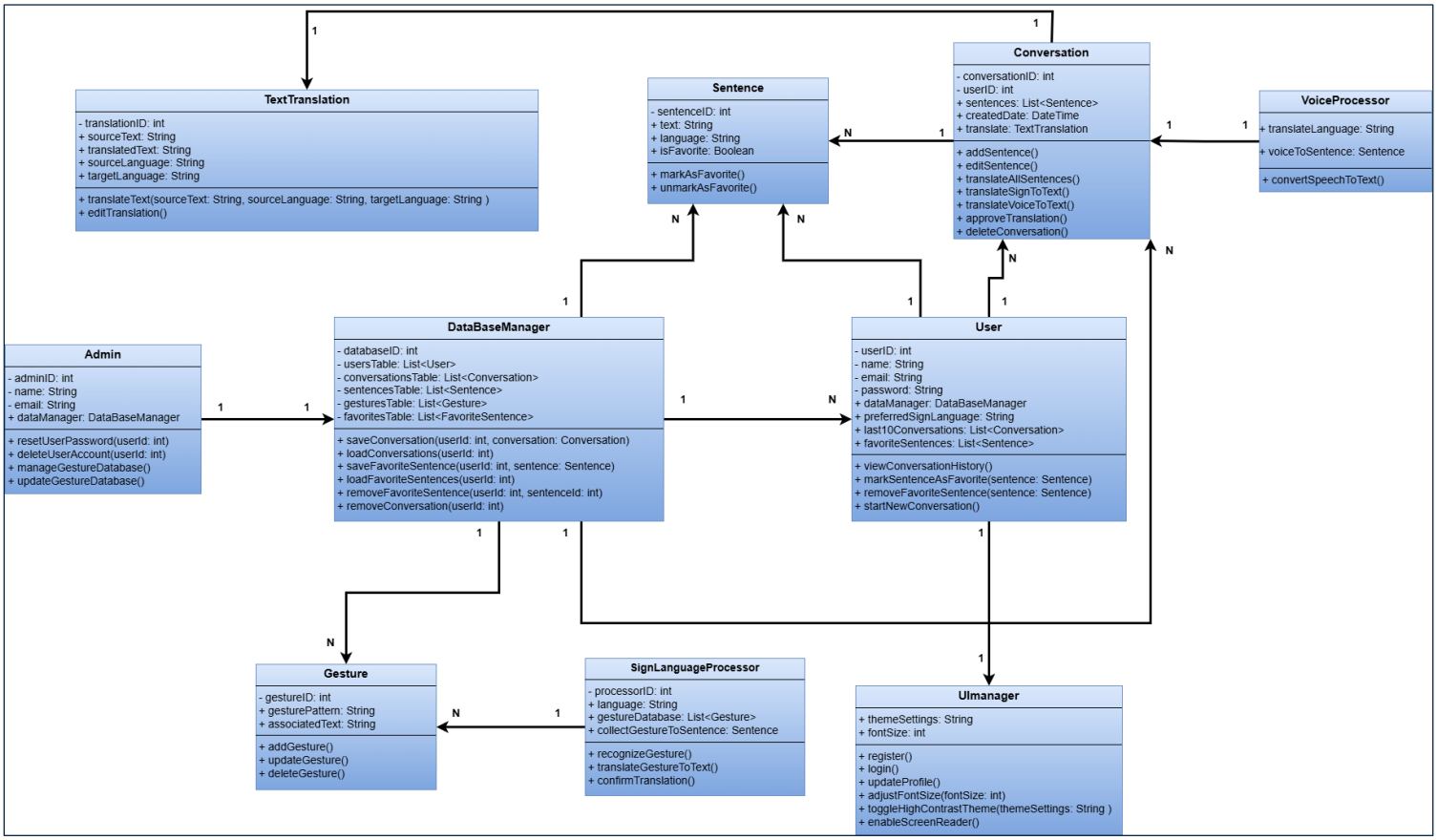
Backup:

- Automatic daily backups of the Firebase database ensure recovery in case of failure.

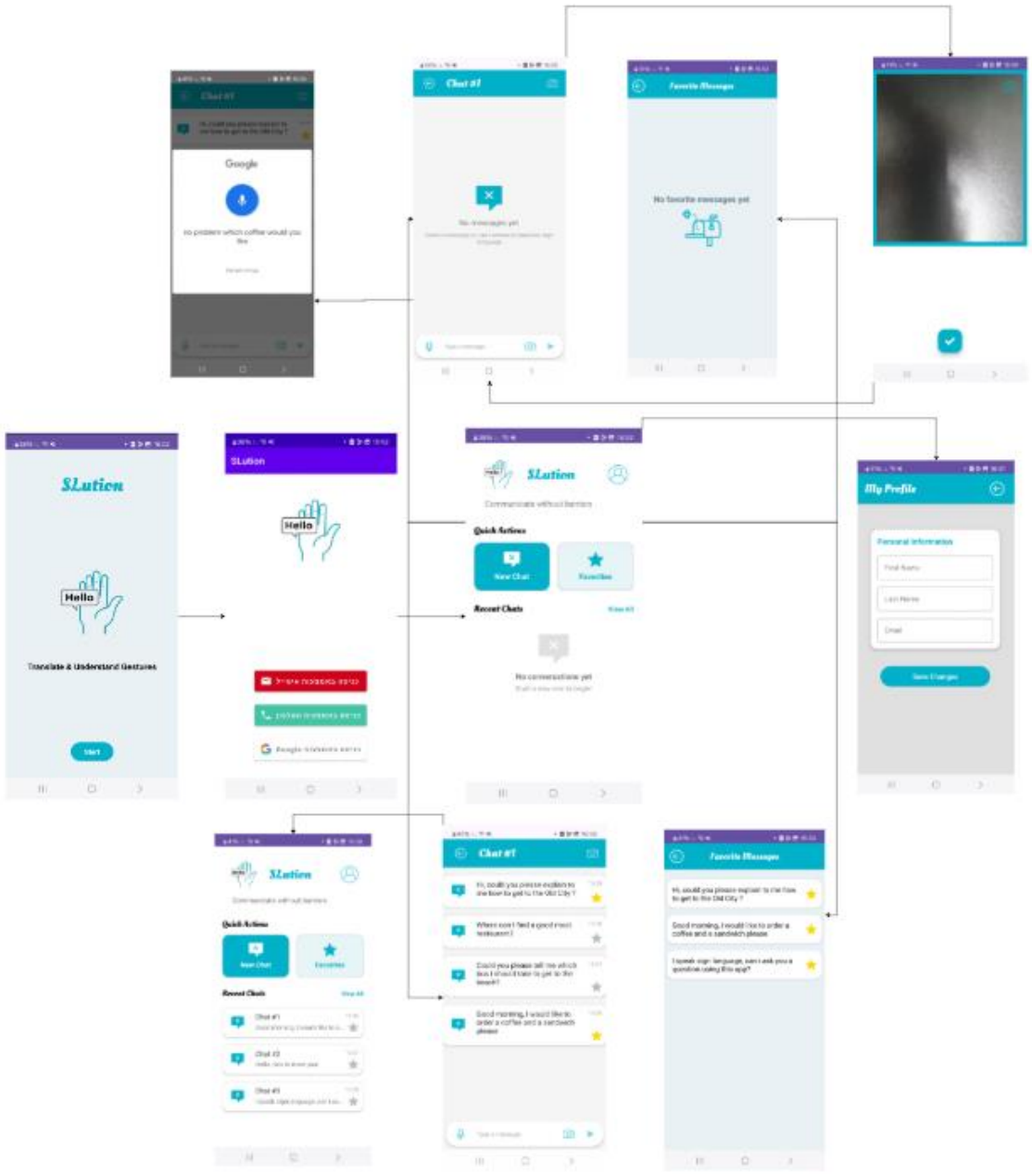
Data Retrieval:

- Real-time syncing between the app and Firebase ensures that updates to user profiles, conversations, and gestures are reflected immediately.

7. Detailed Class/Function Design:



7.1 Application Flow Diagram:



8. Security Design:

To ensure the safety, reliability, and compliance of the Real-Time Sign Language Translator application, the following security measures and principles have been adopted:

8.1 Authentication:

Firebase Authentication:

- The application uses Firebase Authentication to securely manage user sign-in and identity.
- Authentication methods include email/password and Google account login, handled entirely by Firebase's secure backend.
- Firebase handles all password encryption and secure token generation, protecting against session hijacking and brute-force attacks.
- Secure authentication tokens are automatically issued and managed by Firebase to maintain protected user sessions.

8.2 Encryption:

Secure Communication via HTTPS/TLS:

- All data transmitted between the mobile application and Firebase services is encrypted using HTTPS and TLS protocols.
- This ensures that user data, including authentication credentials and conversation history, is securely transmitted and protected from interception.

8.3 Compliance:

GDPR and Privacy Compliance:

- The application complies with GDPR principles by using Firebase's built-in privacy and data protection features.
- Only the minimum required user data is stored to operate the application's functionality.
- Users can request the deletion of their personal data, and Firebase provides mechanisms to support such requests, promoting user transparency and control.

Software Test Plan (STP)

Introduction

This Software Test Plan outlines the strategy, scope, and responsibilities for testing the SLution mobile application. SLution is a real-time sign language to text translation app designed to bridge communication gaps using camera-based gesture recognition, voice-to- text, and Firebase-backed chat features. The goal of this plan is to ensure the system meets its functional requirements and delivers a reliable, accessible user experience.

The app is also planned to integrate with a separate machine learning model for sign language recognition, which is currently under development and will be tested separately before full integration.

Test Items

The software components to be tested include:

- Welcome screen and login flow
- Home screen with chat listing
- Chat management and UI
- Voice-to-text functionality using Google Speech API
- Sign language translation via camera interface
- Favorite messages system
- User profile management
- Firebase Authentication and Realtime Database integration
- External Machine Learning model for gesture recognition (integration pending)

Features to be Tested

- User authentication through email, phone number, and Google accounts.
- Creating new chat sessions and displaying chat history.
- Sending typed or voice-translated messages.
- Accessing the camera to translate sign language into text.
- Saving messages to favorites and viewing them in a dedicated screen.
- Editing and saving personal user profile information.
- Reading and writing data from/to Firebase.
- Switching between front and back camera during translation.
- Integration of camera input with ML-based gesture recognition.
- Displaying recognized signs as text in the UI.

Features Not to be Tested

- Actual sign language gesture recognition model (if not implemented yet)
- iOS compatibility (the app is Android-specific)
- Multilingual support beyond English and Hebrew
- Performance under extreme conditions (e.g., no internet or low battery)
- Server-side analytics or external data integrations outside of Firebase
- Full machine learning-based sign language recognition.

Testing Strategy

The testing process will include the following types:

- Unit Testing: Focused on internal logic such as validation, data storage, and chat model handling.
- Integration Testing: Ensuring that components interact correctly (e.g., camera fragments with chat logic, Firebase with UI).
- System Testing: End-to-end testing of real usage scenarios including message creation, recognition, and navigation.
- Acceptance Testing: Verification that the app meets all project requirements and functions as expected for real users.

Testing will primarily follow a black-box approach using both emulators and real Android devices.

Future integration testing will be planned for the machine learning model once incorporated into the app, including data flow validation and performance testing under real-world conditions.

Test Environment

The testing environment includes:

- Android smartphone (Android 10 or above) with front and back cameras
- Android Studio as the development and testing IDE
- Firebase Realtime Database and Firebase Authentication services
- Google Speech-to-Text API
- Internet connection for cloud features
- Emulator and physical device for parallel testing

For ML testing, additional tools such as TensorFlow may be used during the integration phase.

Responsibilities

- Developer: Unit and integration testing, Firebase data validation
- Tester (or self): Manual UI and system testing
- Peer Reviewer: Acceptance testing and bug reporting

Schedule

The testing timeline is expected to be:

- Test case design: 1 day
- Unit and integration testing: 3 days
- System and acceptance testing: 2 days
- Bug fixing and retesting: as needed during the process
- Test report preparation: 2 days

The full testing phase is estimated to take approximately 8 to 10 working

days. ML integration testing: to be done (after model is finalized)

Risks and Contingencies

- If internet access is unavailable, some features like Firebase sync or voice translation may fail. The app should display appropriate messages and allow retries.

- If the user denies camera permissions, translation features will be disabled. The app should explain the need for permissions and guide the user to enable them.

- Google Speech API may be unsupported on some devices. The app will allow fallback to manual typing.

- Some older Android versions may not support CameraX or Firebase. Testing will focus on devices running API level 29+.

- User profile data may be lost if Firebase access is interrupted. Data is stored both locally (SharedPreferences) and in the cloud to minimize loss.

- Delays in integrating the ML model may postpone full gesture recognition functionality. If so, the camera feature will remain limited to basic translation placeholders until integration is complete.

Software Test Design (STD)

Information

This Software Test Design document provides detailed test cases for verifying the functionality of the SLution mobile application, which enables real-time translation of sign language into text using camera input, voice recognition, and Firebase services.

Test Cases

Test Case ID	Description	Preconditions	Test Steps	Expected Result	Actual Result
TC-001	Test user registration via email	Server running, Firebase initialized	1. Open login page 2. Choose email registration 3. Enter valid email and password 4. Submit	User account is created and redirected to home screen	Feature works as expected
TC-002	Login using Google account	Google account available on device	1. Open login screen 2. Click Google login 3. Select account 4. Grant permission if needed	User successfully logged in and redirected to home screen	Feature works as expected
TC-003	Start new chat and display chat history	User logged in, Firebase initialized	1. Click New Chat' 2. Enter message 3. Send message 4. Return to chat later	Chat appears in history and displays previous messages	Feature works as expected

TC-004	Send message using voice input	Microphone permission granted	1. Click microphone icon. 2. Speak message 3. Confirm transcription 4. Send	Message appears in chat as transcribed	Feature works as expected
TC-005	Use camera to recognize sign and display text	Camera permission granted, ML model integrated	1. Open translation screen 2. Show sign in camera. 3. Confirm translation	Translated text appears on screen	Feature works as expected
TC-006	Save message to favorites	Message visible in chat	1. Click star icon next to message	Message saved and appears in favorites screen	Feature works as expected
TC-007	Edit and save user profile	User logged in	1. Open profile screen 2. Change name/email 3. Click save	Profile updated successfully in Firebase and UI	Feature works as expected
TC-008	Switch between front and back camera	Camera opened in translation screen	1. Click switch camera icon	Camera view switches correctly	Feature works as expected
TC-009	Test Firebase read/write operations	Firebase configured	1. Send message 2. View message saved to Firebase 3. Refresh and confirm persistence	Messages persist and sync correctly across sessions	Feature works as expected
TC-010	Display recognized signs as text from ML model	ML model integrated and camera enabled	1. Sign gesture in front of camera 2. Wait for recognition 3. Confirm displayed result	Recognized gesture is shown as accurate text	Feature works as expected

Conclusion

Project Summary

SLution is a mobile application designed to bridge the communication gap between sign language users and the hearing community by translating American Sign Language (ASL) gestures into English letters in real time. The app targets practical, everyday use and enables more accessible communication for the deaf and hard-of-hearing community using modern computer vision techniques implemented directly on mobile devices.

Key Achievements and Design Strengths

The project implements an ensemble of Convolutional Neural Networks (CNNs) running locally within the Android application to perform fast and accurate hand gesture recognition. By averaging predictions across multiple CNN models and displaying a result only after consistent recognition in 10 consecutive frames, the system ensures reliable and noise-resistant translation. The application also integrates Google Speech-to-Text for voice input and uses Firebase for secure user authentication, profile storage, and conversation management. The user interface is designed to be intuitive, accessible, and responsive, with features such as favorite sentences, recent conversations, and camera-based gesture detection. The system architecture supports offline gesture processing and online synchronization, providing a flexible and scalable foundation for future development.

Technical Architecture Excellence

SLution architecture is streamlined and modular, with all core components including gesture recognition, user interaction, and data handling integrated within the mobile application itself. The use of Firebase for authentication and cloud storage ensures data security and scalability, while communication is encrypted via HTTPS and TLS protocols. Google Speech to Text offers an alternative input method, and the application’s design ensures compatibility across Android devices with different hardware capabilities. A robust testing strategy was executed, covering unit, integration, system, and acceptance testing to validate the app's functionality.

Challenges and Considerations

Key challenges included optimizing gesture recognition to run efficiently on device without external servers, and ensuring stable recognition across different lighting conditions, backgrounds, and hand shapes. Future challenges may include supporting additional sign languages and expanding from letter-level recognition to full word or sentence-level interpretation. Market competition with existing apps also necessitates strategic differentiation through usability, accuracy, and real-time capabilities.

Impact and Future Potential

SLution has the potential to make a significant social impact by improving communication in education, healthcare, customer service, and social interactions. Its architecture allows for future scaling, both in terms of language support and advanced model improvements. With over 70 million sign language users globally, SLution addresses a real need in the accessibility technology space and offers a reliable, scalable, and user-centered solution.

Recommendations for Success

To ensure long-term success, it is recommended to continuously improve gesture recognition accuracy based on real-world user feedback, expand the gesture database, and explore collaborations with educational institutions and organizations in the deaf community. Enhancing accessibility features and ensuring adherence to international standards will further strengthen user trust and adoption.

Final Assessment

SLution is a thoughtfully designed, technically sound, and socially meaningful application. Its reliance on on-device processing, user-friendly design, and real-time performance sets it apart from other solutions in the field. While there is still room to grow especially in extending language coverage and vocabulary depth the project demonstrates a strong foundation and clear potential to become a leading tool in the domain of sign language translation and accessibility technology.