Detailed Design Document

SLution

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 key components that work in unison to deliver seamless functionality. The **User Interface (UI)** is implemented as an Android application, providing users with an interactive platform to engage with the system. It facilitates login, sign-in, profile updates, and the ability to upload images. Additionally, the UI enables gesture recognition through the device camera and displays the corresponding translations to users in real time.

The **Backend Services** are responsible for handling API requests, managing the database, and processing gesture recognition through advanced machine learning models. These services ensure smooth communication between the application's components and manage essential data operations.

The **Database**, powered by Firebase, serves as the primary storage solution for the application. It securely stores user profiles, uploads user images, gesture libraries, favorite sentences, and conversation histories, enabling efficient data retrieval and management.

The **Gesture Recognition Models** form the backbone of the application's translation capabilities. A Convolutional Neural Network (CNN) is utilized to recognize gestures from the camera feeds, while a Long Short-Term Memory (LSTM) model translates the recognized

gestures into meaningful text. These models are optimized for accuracy and real-time performance.

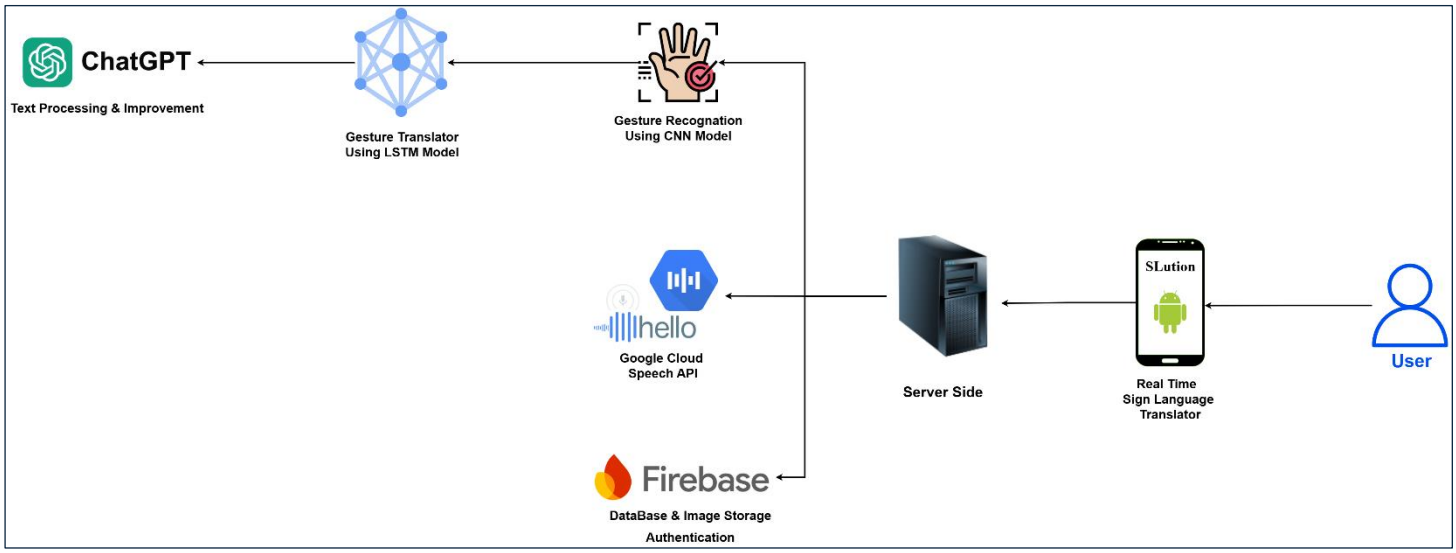
Finally, the application integrates **ChatGPT**, which enhances the translated text to ensure better comprehension and improved communication for users. This integration refines the raw outputs from the gesture recognition models, delivering polished and user-friendly translations.

3. Design Considerations:

This system is built under certain assumptions, constraints, and standards to ensure robust and efficient functionality. The primary assumptions include users having stable internet access and devices capable of supporting camera functionality. Firebase is assumed to handle authentication and data storage reliably. The backend is designed and implemented using Python, ensuring flexibility and compatibility with the system's requirements. The constraints define the system's boundaries, such as the need for real-time processing with a delay of less than three seconds and initial support being limited to ISL and ASL sign languages. To maintain consistency and security, the system adheres to RESTful API standards for backend communication and complies with OWASP security guidelines to ensure safe data transmission.

4. System Architecture:

4.1 High-Level Diagram:



4.2 Components:

The Real-Time Sign Language Translator is composed of six major components working together to ensure efficient and seamless operation. The **User Interface** is an Android application that serves as the primary platform for user interaction. It enables functionalities such as login, profile updates, and gesture recognition using the camera. Translated text is displayed in real time, providing users with immediate feedback. The **Server Side**, developed in Python, manages communication between the client, machine learning models, and the database. It is responsible for all data processing while ensuring secure and efficient operations. The **Google Cloud Speech API** enables spoken input to be converted into text, allowing users to interact with the application through voice commands. The **Database**, powered by Firebase, handles data storage for user profiles, uploaded images, favorite sentences, and conversation histories, while also providing secure authentication and image storage. The **Gesture Recognition** component employs CNN for detecting gestures from camera feeds and LSTM models for processing and translating gestures into text, ensuring accurate and meaningful translations. Lastly, the application integrates with **ChatGPT** to refine and enhance the translated text, ensuring that the output is clear, polished, and user-friendly.

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 machine learning models (CNN and LSTM) to detect gestures from video feeds and convert them into meaningful text.

Input: Video frames captured via the device camera.

Output: Recognized gestures mapped to corresponding text.

Dependencies: Requires real-time video data from the User Interface and model processing on the server.

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.

5.6 ChatGPT Integration Module:

Purpose: Refines and enhances the translated text from gestures or voice input, ensuring better readability and accuracy for the user.

Input: Raw translations generated by the Gesture Recognition Module or Speech API.

Output: Polished and user-friendly translated text.

Dependencies: Processes output from machine learning models and communicates the refined text back to the Server-Side Module.

5.7 Data Flow:

1. Users interact with the User Interface by logging in, capturing gestures, or providing voice input.
2. Gesture data is sent to the Server-Side Module, which processes it using the Gesture Recognition Module (CNN and LSTM models).
3. For voice input, the Google Cloud Speech API converts the audio into text.
4. The raw translations are refined by the ChatGPT Integration Module and sent back to the user through the User Interface.
5. 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 Favorites Table 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 and Favorites 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 Data Flow:

User Interaction:

- Users interact with the UI to login, create conversations, and process gestures or voice inputs.

Gesture Recognition:

- Captured gestures are processed via the SignLanguageProcessor, using data from the Gestures Table.

Voice Input:

- Voice commands are processed by the VoiceProcessor and converted to text.

Conversation Management:

- Sentences and translations are saved in the Conversations Table, while favorites are added to the Favorites Table.

Data Retrieval:

- All interactions with user profiles, conversation histories, and gesture recognition are synchronized in real-time via the DatabaseManager.

6.3 Storage Considerations:

Indexing:

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

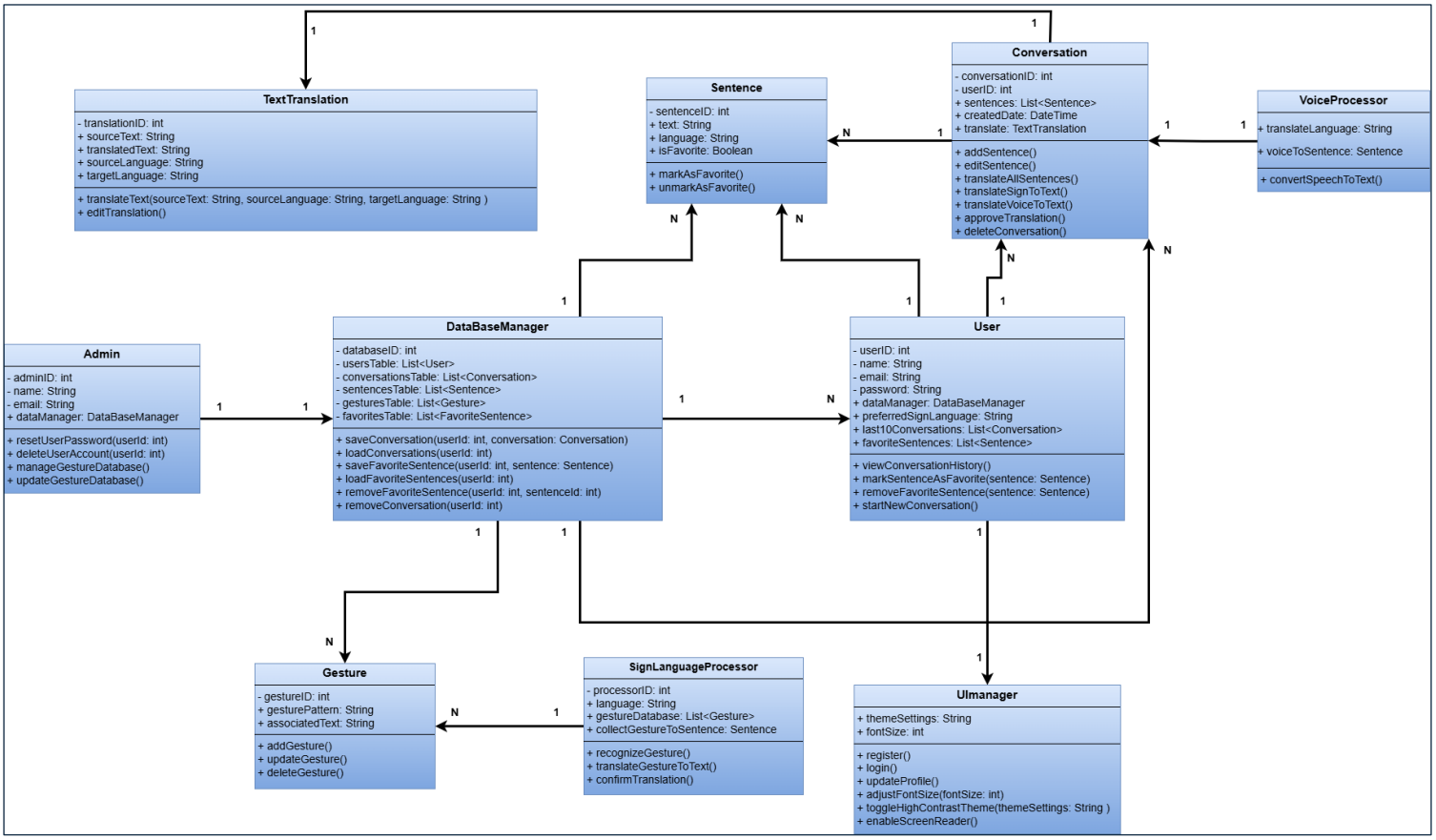
Backup:

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

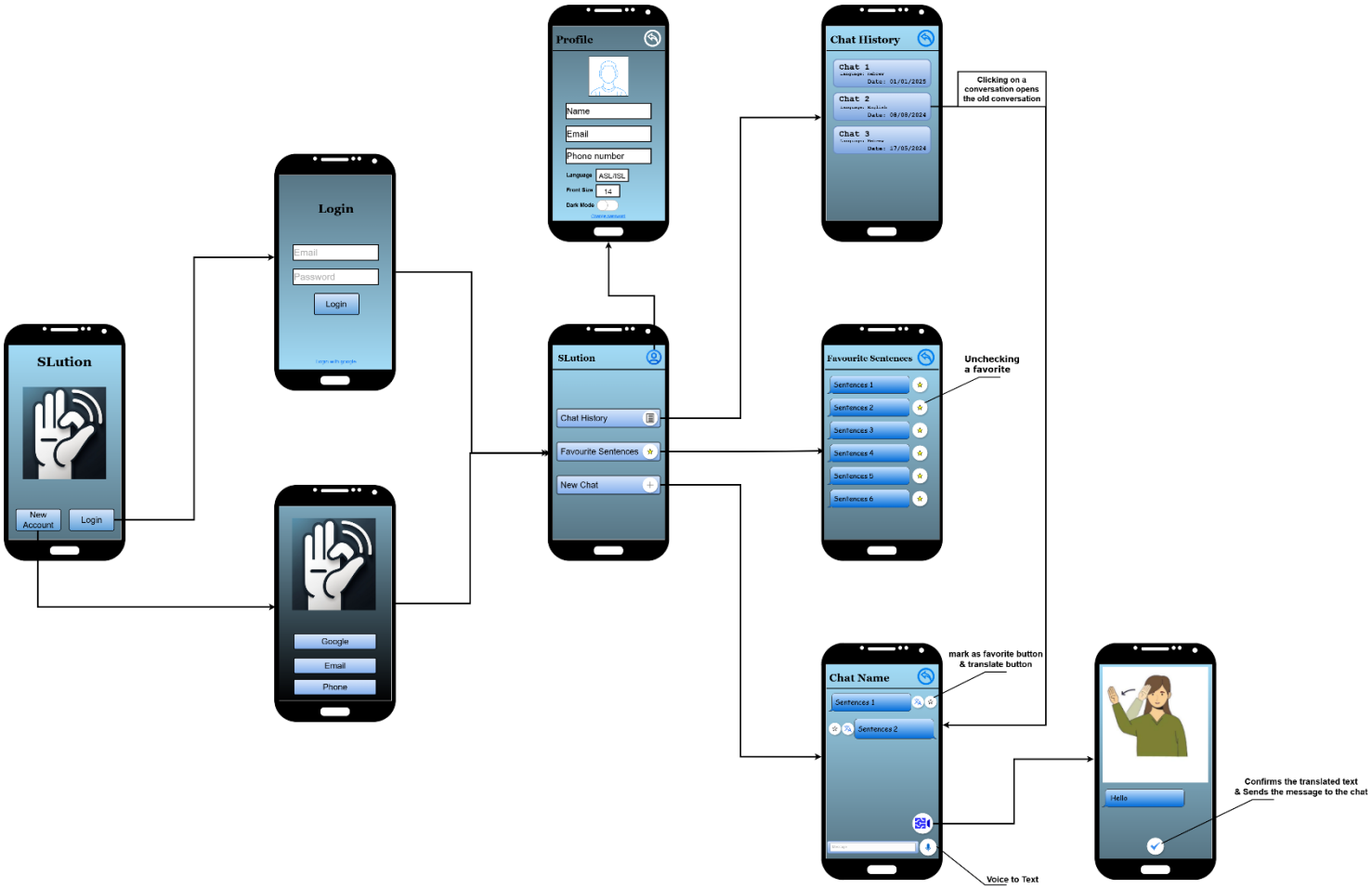
Data Retrieval:

- 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:

OAuth2-based Secure Login:

- Users authenticate securely using OAuth2 protocols, which include token-based authentication for enhanced security.
- This method minimizes the risk of session hijacking and unauthorized access.

Password Hashing with bcrypt:

- User passwords are securely hashed using bcrypt, a robust algorithm designed to prevent brute-force attacks.
- Hashing ensures that even if the database is compromised, raw passwords cannot be retrieved.

8.2 Encryption:

End-to-End Encryption:

- All data transmitted between the client and server is encrypted using secure protocols such as HTTPS and TLS.
- This ensures that sensitive user data, such as login credentials and translations, is protected from interception during transmission.

8.3 Compliance:

GDPR Compliance:

- The application complies with GDPR regulations to ensure the privacy and protection of user data.
- Data minimization principles are implemented by only storing the data necessary for operation.
- Users have the ability to request the deletion of their data, ensuring transparency and control over personal information.