
# Real-Time Sign Language Transcription
# Final Year Project

**PHASE – I**

---

Submitted by

**Hamza Tariq 70113214**
**Hussnain Tariq 70110706**
**Hamza Akhtar 70111155**

---

**Project Supervisor**

**Sir Sayed Zishan Ali**

**BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING**

**DEPARTMENT OF SOFTWARE ENGINEERING, UNIVERSITY OF LAHORE**

**January 20, 2023**

## FINAL YEAR PROJECT PHASE-I DOCUMENTATION

## STATEMENT OF SUBMISSION

Submitted to the University of Lahore in partial fulfillment of the requirement for the award of degree of Bachelors of Science in Software Engineering (BSSE)

By: Sap ID: _____

By: Sap ID: _____

By: Sap ID: _____

**Project Advisor: _____**

**Sayed Zishan Ali**
Lecturer at University of Lahore
Department of Software Engineering

# Abstract

Sign-language communication plays a crucial role in fostering inclusive communication for individuals with hearing impairments. This paper introduces a pioneering application, the Sign-Language Transcription application, designed to bridge communication gaps by providing real-time transcription of sign language gestures into text. Leveraging advanced gesture recognition and machine learning technologies, the application offers an accessible and user-friendly platform for users to seamlessly transcribe sign language messages.

The Sign-Language Transcription application encompasses a comprehensive system architecture, including a user interface, gesture recognition module, transcription engine, user history tracker, and external messaging integration. The application's innovative real-time transcription feature enables users to receive instantaneous text representations of sign language gestures, fostering efficient and inclusive communication.

# Dedication

This work is dedicated to individuals with hearing impairments, to whom effective communication is both a challenge and a necessity. The Sign-Language Transcription application aims to contribute to overcoming these challenges, providing a tool for enhanced communication and understanding. May it serve as a valuable resource for those seeking more accessible means of expression.

## Acknowledgement

We extend our sincere gratitude to all those who contributed to the development and realization of the Sign-Language Transcription application. Our appreciation goes to the users and experts who provided invaluable insights and feedback throughout the development process. Special thanks to the dedicated team members who worked tirelessly to bring this application to fruition. Additionally, we acknowledge the support from our institutions and the broader community, without whom this project would not have been possible. Your collective efforts have played a crucial role in advancing inclusive communication for individuals with hearing impairments.

Hamza Tariq
Hussnain Tariq
Hammad Akhtar

# Table of Contents

# CHAPTER 1

# INTRODUCTION TO PROBLEM

# Introduction

The objective of this Final Year Project (FYP) is to pioneer an innovative system that transcribes Sign Language into natural language text, bridging communication gaps for the Deaf and hard of hearing community. Sign Language is a rich and complex visual language used by the Deaf, and this project aims to provide a means for the wider population to understand and engage with Sign Language conversations.
In this project, we embark on a unique approach where Sign Language users perform signing while our technology translates their signs into coherent and understandable natural language. The significance of this endeavor lies in facilitating inclusive communication and breaking down the barriers faced by the Deaf community in accessing information and interacting with others.

The foundation of our project begins with the compilation of a comprehensive dataset comprising video recordings of Sign Language conversations. This dataset encompasses various sign languages and dialects, allowing us to create a diverse and representative sample. We meticulously preprocess the video data, ensuring clarity and consistency in the signing gestures.

Next, we delve into the intricacies of Sign Language linguistics and cognitive features. Through advanced computer vision techniques, we extract vital information from the signing videos, including hand movements, facial expressions, and body language. These features play a pivotal role in deciphering the intended message accurately.

Our journey through the project involves the exploration of cutting-edge Natural Language Processing (NLP) models, including recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and state-of-the-art transformer-based models such as BERT and GPT-3. These models serve as the backbone of our system, translating the visual nuances of Sign Language into written text.

Transparency and interpretability are at the core of our system. We dedicate considerable effort to developing methods that provide insights into how the model interprets Sign Language gestures, ensuring that the translation aligns with
the intended meaning. The user interface is thoughtfully designed to be intuitive, making it accessible for Sign Language users to input their signing and receive coherent text translations.

Throughout the project's lifecycle, we remain committed to ethical principles. We prioritize user data privacy, informed consent, and mitigating potential biases in the system. Additionally, we collaborate closely with members of the Deaf community and sign language experts to validate the accuracy and cultural sensitivity of our translations.

The anticipated outcome of our FYP is a robust Sign Language-to-text transcription system. This technology serves as a vital tool for bridging the

communication gap between the Deaf community and the wider society. It empowers Sign Language users to engage in conversations, access information, and participate fully in various aspects of life. However, it's important to underscore that our system should complement, not replace, the importance of human interpreters in certain contexts.

## Purpose

The purpose of developing our project is to address the communication challenges faced by the Deaf and hard of hearing community. The need arises from the complexity of Sign Language, which creates barriers to effective communication. Our Sign Language-to-text transcription system aims to bridge this gap by translating Sign Language gestures into coherent natural language text.

In the market and societal context, our project offers a transformative solution. The system has the potential to improve communication accessibility for the Deaf community, allowing them to engage more effectively with the wider society. This technology goes beyond mere conversation facilitation; it can enhance access to information, education, employment opportunities, and social interactions, contributing to a more inclusive and understanding society.

The anticipated impact is profound, as our project seeks to empower individuals with hearing impairments, enabling them to participate fully in various aspects of life. It aligns with ethical principles, ensuring user data privacy and cultural sensitivity. Importantly, our technology is designed to complement human interpreters, emphasizing a balanced integration of technological advancements and human expertise in communication. Overall, the project aims to bring about positive changes in the lives of the Deaf and hard of hearing individuals, fostering inclusivity and equal participation in society.

## Objectives

1. **Develop a Functional Sign Language-to-Text Transcription System:** Create an advanced system capable of accurately transcribing Sign Language gestures into natural language text, ensuring functionality and reliability.
2. **Diverse and Representative Dataset:** Compile a comprehensive dataset containing video recordings of Sign Language conversations, encompassing various sign languages and dialects to ensure the system's inclusivity and adaptability.
3. **Implement Advanced Computer Vision Techniques:** Utilize cutting-edge computer vision techniques to extract crucial information from signing videos, such as hand movements, facial expressions, and body language, for precise interpretation of Sign Language nuances.
4. **Deploy State-of-the-Art NLP Models:** Explore and integrate state-of-the-art Natural Language Processing (NLP) models, including recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and transformer-based models like BERT and GPT-3, to enhance the accuracy of translations.
5. **Ensure Transparency and Interpretability:** Develop methodologies that provide insights into the model's interpretation of Sign Language gestures, ensuring transparency

and interpretability to align translations with intended meaning.

6. **User-Friendly Interface Design:** Design an intuitive user interface that allows Sign Language users to input their gestures effortlessly and receive coherent text translations, ensuring accessibility and usability.
7. **Adhere to Ethical Principles:** Prioritize user data privacy, informed consent, and address potential biases in the system, maintaining a strong commitment to ethical considerations throughout the project lifecycle.
8. **Collaborate with Deaf Community and Experts:** Work closely with members of the Deaf community and sign language experts to validate the accuracy and cultural sensitivity of translations, incorporating valuable feedback into system refinement.
9. **Empower Deaf Community Engagement:** Empower Sign Language users to actively engage in conversations, access information, and participate fully in various aspects of life, contributing to increased inclusivity and understanding within society.
10. **Complement Human Interpreters:** Emphasize that the system serves as a supplementary tool rather than a replacement for human interpreters, recognizing and respecting the irreplaceable role of human expertise in certain contexts.

## Existing Problems

1. **Limited Availability:** Human interpreters may not be readily available at all times, leading to delays in communication for the Deaf community. This limitation hinders spontaneous and real-time interactions.
2. **Cost:** Hiring professional human interpreters can be expensive, making it difficult for some individuals or organizations to afford constant interpretation services. This financial barrier restricts access to effective communication.
3. **Scalability Issues:** Human interpreters may face challenges in scaling their services to meet the increasing demand, particularly in situations with a large number of Deaf individuals or simultaneous communication needs.
4. **Subjectivity and Variability:** Interpretation can be subjective, and individual interpreters may have different interpretations of the same Sign Language message. This subjectivity can lead to potential miscommunication and misunderstandings.
5. **Privacy Concerns:** In certain situations, relying on human interpreters may raise privacy concerns, especially when discussing sensitive or personal matters. Users may feel more comfortable with technology that ensures data privacy.
6. **Geographical Constraints:** Access to qualified interpreters may be limited

in certain geographic areas, particularly in rural or remote locations. This can result in disparities in communication accessibility.

7. **Training and Certification Challenges:** Ensuring a consistent level of quality among interpreters requires standardized training and certification processes. Variability in interpreter skill levels can impact the quality of communication.

## Proposed Solution

1. **Real-time Accessibility:** Our system aims to provide real-time translation, overcoming the delay associated with the availability of human interpreters. This feature is crucial for spontaneous and immediate communication.
2. **Cost-Effectiveness:** By automating the translation process, our system seeks to offer a cost-effective solution compared to hiring human interpreters. This can increase accessibility for individuals and organizations with budget constraints.
3. **Scalability:** The automated system is designed to be scalable, allowing it to handle a large number of simultaneous translation requests efficiently. This addresses the scalability issues faced by human interpreters.
4. **Consistency and Objectivity:** Machine learning models provide a consistent and objective approach to translation. By reducing subjectivity, our system aims to improve the accuracy and reliability of Sign Language translations.
5. **Privacy Considerations:** Our project places a strong emphasis on user data privacy. Unlike human interpreters, our system can ensure a level of privacy, especially in situations involving sensitive or personal information.
6. **Geographical Accessibility:** As a technology-driven solution, our system can be accessed remotely, promoting geographical accessibility in both urban and remote areas where access to qualified interpreters might be limited.
7. **Continuous Improvement:** Through machine learning, our system can continuously learn and improve its accuracy over time, adapting to various signing styles and nuances. This dynamic learning process enhances the quality of translations.

# Chapter 2

# Software Requirement Specification

## 2.1 Introduction

### 2.1.1 Purpose

The purpose of this Software Requirement Specification (SRS) document is to provide a comprehensive understanding of the Sign-Language Transcription application. It outlines the functional and non-functional requirements, design constraints, and interfaces necessary for the successful development and implementation of the application.

**Intended Audience**

The intended audience for this Software Requirement Specification includes, but is not limited to:

**Development Team:** Software engineers, programmers, and designers who will be involved in the development and implementation of the system.

**Testing Team:** Quality assurance professionals responsible for validating and verifying that the system meets the specified requirements.

**Project Managers:** Individuals overseeing the planning, execution, and monitoring of the project.

**Stakeholders:** Investors, sponsors, and any individuals or organizations with a vested interest in the successful development and deployment of the application.

**Documentation Team:** Writers responsible for creating user manuals, technical documentation, and other related materials.

**End Users:** Individuals who will interact with and benefit from the system, including drivers and administrators.

By addressing the needs of this diverse audience, this SRS aims to ensure a common understanding of the project's objectives, functionalities, and constraints, fostering effective collaboration and successful project outcomes.

### 2.1.2 Scope

The scope of this document encompasses the entire software development life cycle, from the conceptualization of the Sign-Language Transcription application to its deployment and maintenance. It serves as a guide for developers, designers, and stakeholders involved in the project.

### 2.1.3 Definitions, acronyms, and abbreviations

- GUI – Graphical user interface
- DB – Database
- SRS – Software requirement specification
- AI – Artificial Intelligence
- ASL – American Sign-Language

### 2.1.4 References

**REFERENCES**

Slobin, Dan. (1999). Sign language transcription at the morphological level: the Berkeley Transcription System (BTS).

(2023). A Survey on Indian Sign Language Translation Using Artificial Intelligence. 10.1007/978-981-99-3963-3_33.

M. Papatsimouli et al., "Real Time Sign Language Translation Systems: A review study," 2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, Germany, 2022, pp. 1-4, doi: 10.1109/MOCAST54814.2022.9837666.

A Survey of Advancements in Real-Time Sign Language Translators: Integration with IoT Technology

### 2.1.5 Overview

This section provides an overview of the entire Software Requirement Specification, highlighting key chapters and their respective purposes. It aims to offer a quick reference guide for readers

navigating through the document.

## *2.2* Overall description

### 2.2.1 Product Perspective

The Sign-Language Transcription application is designed to operate as an independent system, employing advanced AI and machine learning algorithms for real-time sign language recognition. It interfaces with various devices and platforms, striving for seamless integration into modern communication environments in the future.

.

- **Admin side**
  used for the administration activities just like approval of user's accounts, managing data for analytics, training and monitoring.
- **User side**
  Used to perform the transcription either video or real-time. Users can engage in real-time sign language conversations, with the application accurately transcribing their gestures into text.

#### 2.2.1.1 System Interfaces:

Interaction with AI and machine learning components

#### 2.2.1.2 User Interfaces:

User-friendly screens for gesture input and transcription output.

#### 2.2.1.3 Hardware Interfaces:

The mobile application can be used on android/IOS and web application can be used on any device like laptop, mobile phone as long as it has active internet connection and is compatible with devices featuring cameras for capturing sign language gestures.

#### 2.2.1.4 Software Interfaces:

Software interfaces includes the operating system for mobile, Android or IOS.
For web application, it just needs to have a browser and active internet connection.

#### 2.2.1.5 Memory:

Utilization of primary and secondary memory or database as required.

### 2.2.2 Product Functions

The application performs the following key functions:

- Real-time recognition and transcription of sign language gestures into text.
- User-friendly interface design for both sign language proficient users and those less familiar with sign language.
- Accessibility features, including voice output and customizable font sizes.
- Continuous improvement based on user feedback and technological

More functions are formally defined in the tables below:

#### 2.2.2.1 User functions

*Table 1: User Function - Register*

| **ID** | FR_01 | | | |
|---|---|---|---|---|
| **Name** | User Register/Signup | | | |
| **Description** | **Input** | **Output** | **Requirements** | **Basic Workflow** |
| User shall able to register through application | Username/email<br>Password<br>Full name | Creation of a new account | Database, Internet | Enter these valid inputs for the creation of the account |

*Table 2: User Function - Login*

| **ID** | FR_02 | | | |
|---|---|---|---|---|
| **Name** | Login | | | |
| **Description** | **Input** | **Output** | **Requirements** | **Basic Workflow** |

| | | | | |
|---|---|---|---|---|
| User shall be able to login to the application | Username/email<br>Password | Provide access to dashboard upon successful login | Input validation<br>Account verification<br>Input | Enter inputs if valid, the system will go to it's dashboard |

*Table 3: User Function - Video Transcription*

| **ID** | FR_03 | | | |
|---|---|---|---|---|
| **Name** | Video Transcription | | | |
| **Description** | **Input** | **Output** | **Requirements** | **Basic Workflow** |
| User shall be able to upload a video for transcription | Video of valid format | If video is of acceptable quality/clarity: Text script,<br>an error message otherwise | Internet connectivity, local storage access | User selects the option for video transcription and uploads a video. |

*Table 4: User Function – Real-Time Transcription*

| **ID** | FR_04 | | | |
|---|---|---|---|---|
| **Name** | Real-Time Transcription | | | |
| **Description** | **Input** | **Output** | **Requirements** | **Basic Workflow** |

| | | | | |
|---|---|---|---|---|
| Users shall be able to transcribe in real-time | Camera stream | Text transcription | Internet connectivity, camera access | User selects the option for real-time transcription |

*Table 5: User Function – Track History*

| **ID** | FR_05 | | | |
|---|---|---|---|---|
| **Name** | Track History | | | |
| **Description** | **Input** | **Output** | **Requirements** | **Basic Workflow** |
| User shall be able to look-up past transcripts | Interaction with the history tab/tile | List of past transcripts | Data storage | User would press on the history tab to get a list of past transcriptions |

### 2.2.3 User Characteristics

Users vary in sign language proficiency, technical expertise, and educational backgrounds. The application caters to a diverse user base, ensuring usability for both sign language experts and those less familiar with signing but a familiarity with ASL.

### 2.2.4 Constraints

**Regulatory Policies:** The application must comply with data protection regulations regarding the collection, storage, and processing of user information. User consent for data

processing and clear privacy policies should be implemented.

**Hardware Limitations:** Compatibility and optimization considerations for diverse hardware environments, the application is dependent on camera hardware some of its features.

**Interfaces to Other Applications:** Integration constraints with existing applications and systems.

**Multiple Sign-Languages:** The fact that there are multiple Sign-languages used around the world poses a constraint on the application of which on to cater too.

**Reliability Requirements:** Mandated reliability standards to ensure consistent performance.

**Safety and Security Considerations:** Implementation of measures to address safety and security concerns in the application.

### 2.2.5 Assumptions and dependencies

This section outlines the assumptions made and dependencies identified for the successful implementation:

**Assumptions:**

1. **Stable Network Connectivity:** The assumption is made that users will have stable and reliable network connectivity for real-time data exchange and communication.

2. **Camera Access:** The application assumes that camera access is always available to use certain features.

3. **Familiarity with the language:** The application assumes that the user is familiar with ASL.

4. **User Device Compatibility:** The application assumes compatibility with a range of user devices, including smartphones and tablets, for optimal accessibility.

## Dependencies:

1. **Internal APIs:** The project is dependent on internal API's to communicate the data between applications.

2. **Hardware Components:** Dependencies on specific hardware components, such as cameras.
3. **Third-party Software Libraries:** Dependencies on third-party software libraries, particularly for machine learning like computer vision (CV).

4. **Data Security Protocols:** The project relies on robust data security protocols to ensure the confidentiality and integrity of user and system data.

### 2.2.6 Apportioning of requirements

There is a requirement we have delayed until future version of the system. This include payment method for the users to purchase the subscription.

## 2.3 Specific Requirements

Every system has its own specific requirements according to its nature. The requirements is of two types including functional and non-functional requirements. These are as follow:

### 2.3.1 Functional Requirements

This section is describing the functional requirements at a sufficient level of detail for the designers to a design a system satisfying the user requirement and testes to verify that the system satisfies the requirement.

**User**

- User shall be able to Register
- User shall be able to login
- User shall be able to logout
- User shall be able to receive text transcription of their signed communication
- User shall be able to select video transcription
- User shall be able to select real-time transcription
- User shall be able to track history to past transcriptions
- User shall receive clear visual cues and feedback during sign language recognition and transcription

**Admin**

- Admin shall be able to create, view, edit, and delete user accounts
- Admin shall be able to manage user roles and permissions
- Admin shall be able to export and download conversation data for analysis or archiving
- Admin shall be able to delete conversation data upon request or according to data retention policies

### 2.3.2 Non-Functional requirements

This section outlines the non-functional requirements for the application, ensuring that the system meets certain quality attributes and performance criteria:

1. **Scalability:**

Requirement: The system should handle a large number of concurrent users and vehicles.
Rationale: To accommodate potential growth in user base

2. **Security:**

Requirement: Ensure the security of user data and communication between devices.
Rationale: Protect sensitive information and prevent unauthorized access or manipulation.

**3. Reliability:**
Requirement: The system should operate with high accuracy and minimal false positives. The system should operate in real-time with minimal latency and function in various lighting conditions
Rationale: Ensure continuous and reliable service to users.

## 4. Usability:
Requirement: Provide a user-friendly interface for both the mobile and web applications.
Rationale: Enhance user experience and accessibility, regardless of technical expertise.

**5. Accuracy:**
Requirement: detection systems should be highly accurate and adoptive.
Rationale: Ensure precision in the resultant transcript.

**6. Performance:**
Requirement: The system should in real-time with minimal latency.
Rationale: Enhance the system's responsiveness.

**7. Data Storage:**
Requirement: Implement efficient data storage and retrieval mechanisms for historical data.
Rationale: Optimize storage resources and ensure quick access to past data for analysis.

# 3.Appendixes

a. IoT (Internet of Things): A network of interconnected devices that can communicate and share data with each other over the internet.
b. ML (Machine Learning): A subset of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed.
c. SRS (Software Requirement Specification): A document that outlines the functional and non-functional requirements of a software system.

d. Regulatory Compliance: Adherence to laws, regulations, and standards relevant to the development and operation of the software.
e. API (Application Programming Interface): A set of rules that allows one software application to interact with another.

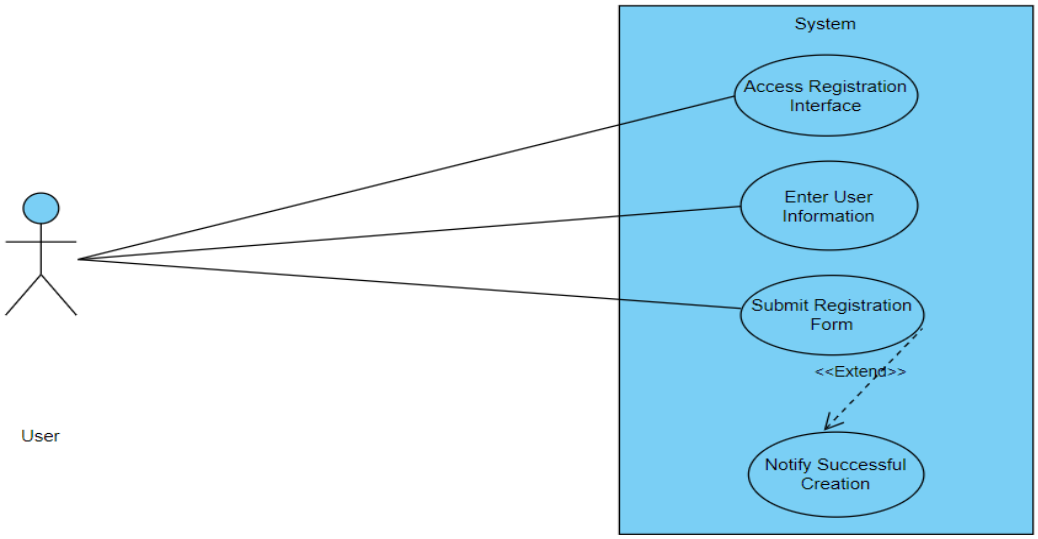
## 4. index



# Chapter 3

# Use-Cases

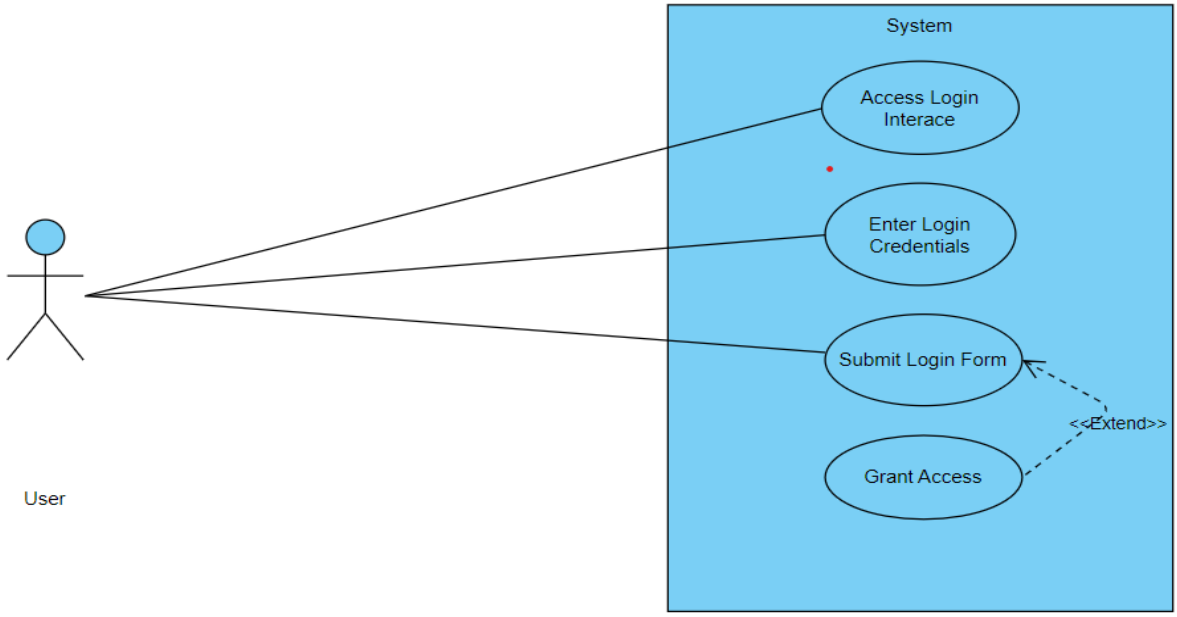
**USE CASE 1:**




| **Use Case ID** | UC_001 |
|---|---|

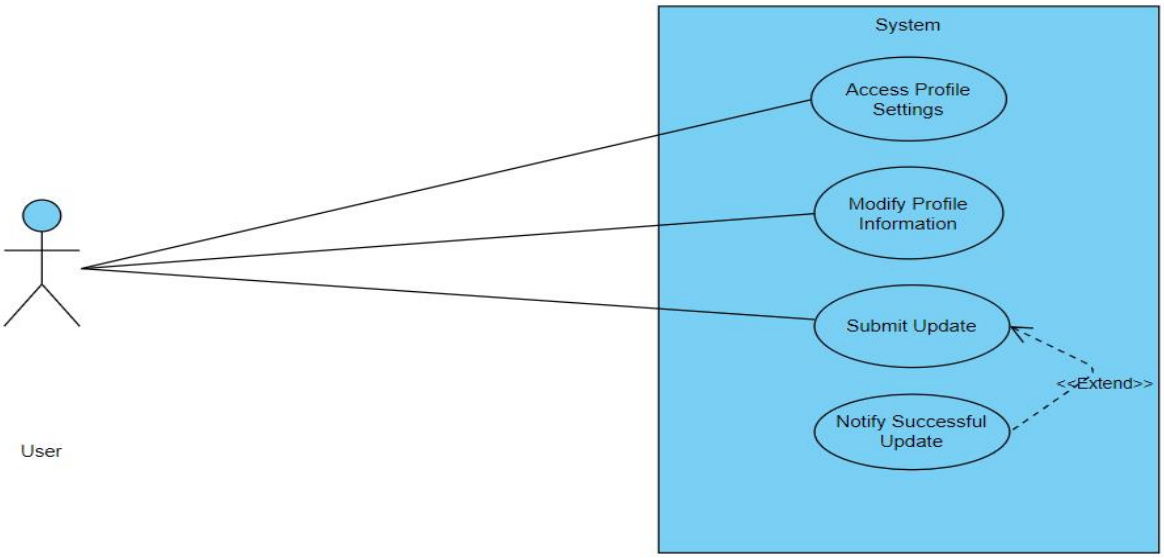
| **Use Case Name** | Registration/Sign up | |
|---|---|---|
| **Description** | This use case involves the process of creating a new account within the Sign Language Transcription System, allowing users to access and utilize the system's features. | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user accesses the registration interface.<br>➢ The user enters the required information, such as username, email, and password.<br>➢ The user submits the registration form.<br>➢ The system notifies the user of successful account creation. | ➢ The system displays the account creation form.<br>➢ The system validates the entered information.<br>➢ The system processes the registration request and creates a new user account. |
| **Alternate Flow** | If the entered information is incomplete or fails validation:<br>➢ The system notifies the user of the validation error.<br>➢ The user corrects the information and resubmits the form.<br>➢ Steps 4 to 8 are repeated. | |

**USE CASE 2:**




| **Use Case ID** | UC_002 | |
|---|---|---|
| **Use Case Name** | Login | |
| **Description** | User can login to the system | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user accesses the login interface.<br>➢ The user enters their username/email and password.<br>➢ The user submits the login form. | ➢ The system displays the login form.<br>➢ The system validates the login credentials.<br>➢ The system processes the login request and grants access to the user. |
| **Alternate Flow** | If the entered credentials are incorrect:<br>➢ The system notifies the user of the authentication failure.<br>➢ The user retries the login with correct credentials.<br>➢ Steps 4 to 6 are repeated. | |

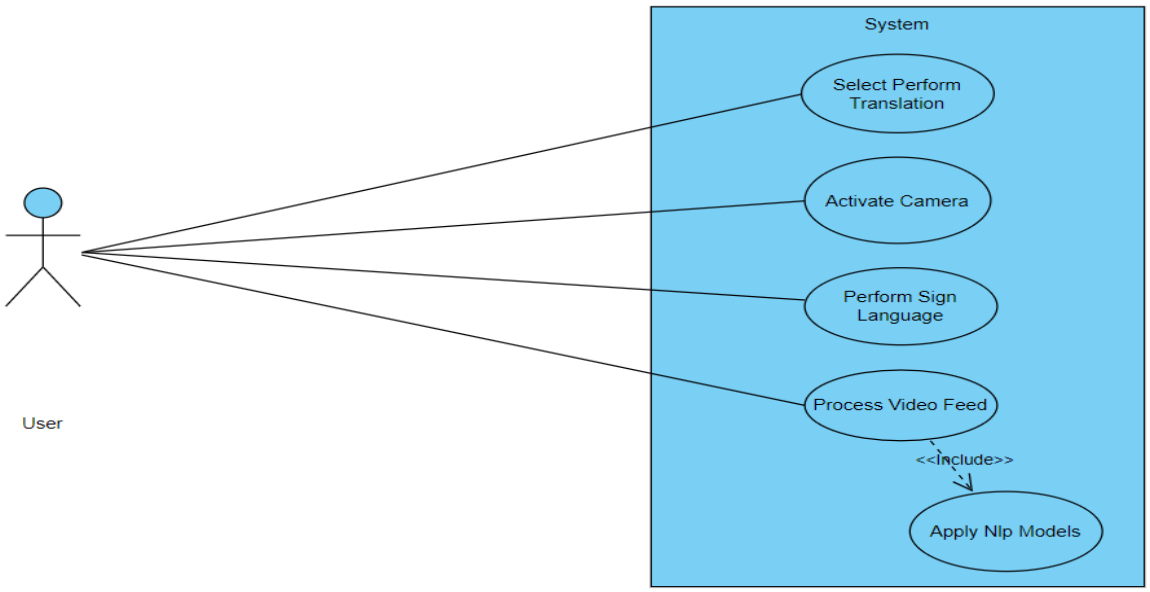
**USE CASE 3:**




| | | |
|---|---|---|
| **Use Case ID** | UC_003 | |
| **Use Case Name** | Update Profile | |
| **Description** | This use case involves the process of updating user profile information. | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must be logged into the system. | |
| **Post-Condition** | The user's profile information is successfully updated. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user navigates to the profile settings.<br>➢ The user modifies the desired profile details (e.g., name, email, or password).<br>➢ The user submits the updated information. | ➢ The system displays the user's current profile information.<br>➢ The system validates the updated information.<br>➢ The system processes the update request and reflects the changes in the user's profile. |
| **Alternate Flow** | If the entered information is incomplete or fails validation:<br>➢ The system notifies the user of the validation error. | |

| | |
|---|---|
| | ➢ The user corrects the information and resubmits the form.<br>➢ Steps 4 to 6 are repeated. |

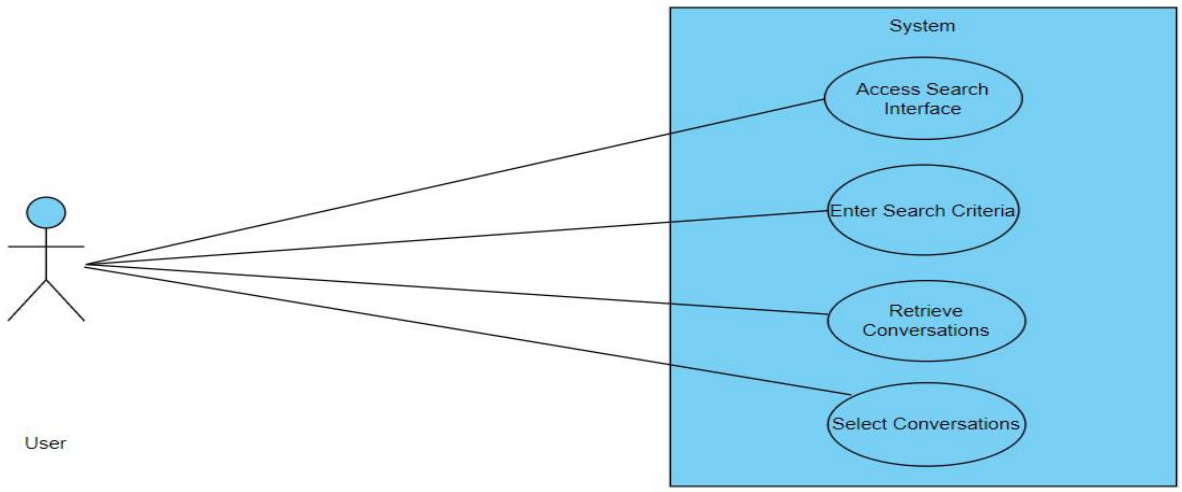
**USE CASE 4:**




| | | |
|---|---|---|
| **Use Case ID** | UC_004 | |
| **Use Case Name** | Perform Sign Language Translation | |
| **Description** | This use case involves the system translating live Sign Language gestures into natural language in real-time. | |
| **Primary Actor** | User (Sign Language User) | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must be logged into the system, and the device must have access to a camera. | |
| **Post-Condition** | The system successfully transcribes the Sign Language gestures into natural language. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user selects the "Perform | ➢ The system activates the camera for |

| | | |
|---|---|---|
| | Translation" option.<br>➢ The user performs Sign Language gestures in front of the camera. | live translation.<br>➢ The system processes the live video feed, extracting key features.<br>➢ The system applies Natural Language Processing (NLP) models to translate gestures into text. |
| **Alternate Flow** | If the system encounters difficulty in recognizing gestures:<br>➢ The system may prompt the user to adjust lighting or perform clearer gestures.<br>➢ Steps 3 to 5 are repeated until successful translation. | |

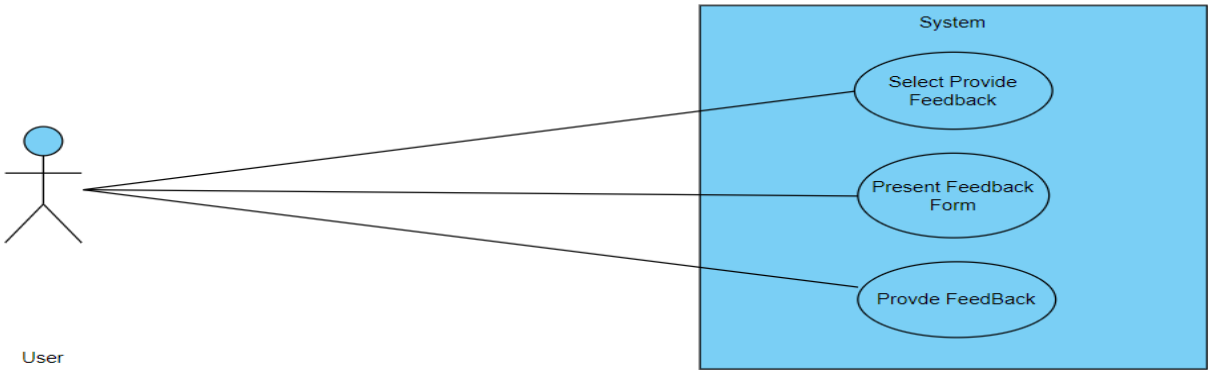
**USE CASE 5:**




| | |
|---|---|
| **Use Case ID** | UC_005 |
| **Use Case Name** | Search Translated Conversations |
| **Description** | This use case involves the user searching for and accessing previously translated Sign Language conversations. |
| **Primary Actor** | User |
| **Secondary Actor** | None |

| | | |
|---|---|---|
| **Pre-Condition** | The user must be logged into the system.. | |
| **Post-Condition** | The user successfully retrieves and views previously translated conversations. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user accesses the "Search Conversations" feature.<br>➢ The user enters search criteria, such as keywords, date, or participants.<br>➢ The user selects a conversation from the search results. | ➢ The system displays a search interface.<br>➢ The system searches the database for relevant translated conversations.<br>➢ The system presents the selected conversation in natural language text. |
| Alternate Flow | If there are no matching conversations:<br>➢ The system notifies the user of no results.<br>➢ The user may refine the search criteria.<br>➢ Steps 4 to 6 are repeated. | |

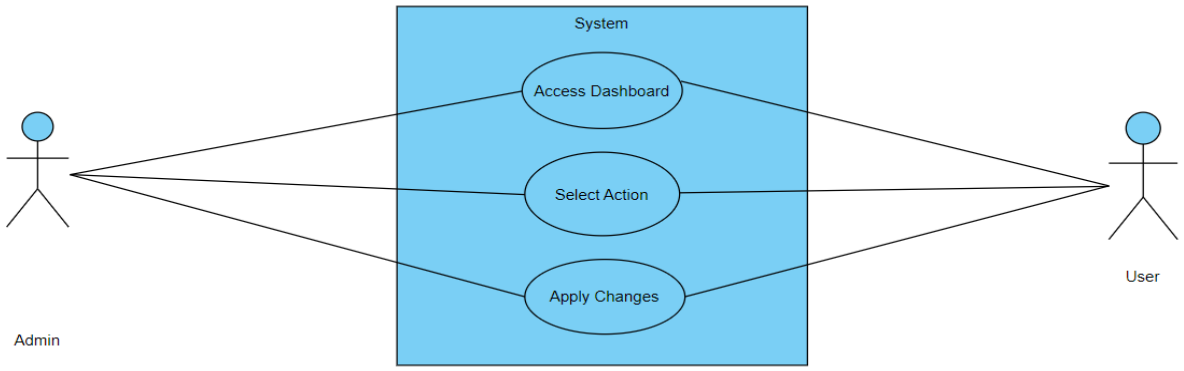
**USE CASE 6:**




| | |
|---|---|
| **Use Case ID** | UC_006 |
| **Use Case Name** | Provide Feedback on Translations |
| **Description** | This use case involves users providing feedback on the accuracy and quality of Sign Language translations. |

| | | |
|---|---|---|
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must be logged into the system and have accessed a translated conversation. | |
| **Post-Condition** | User feedback is recorded and may be used for system improvement. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ While viewing a translated conversation, the user selects the "Provide Feedback" option.<br>➢ The user provides feedback on the accuracy and clarity of the translation. | ➢ The system presents a feedback form.<br>➢ The system records the user's feedback. |
| **Alternate Flow** | If the user chooses not to provide feedback:<br>➢ The system proceeds without collecting feedback.<br>➢ The user continues with their interaction. | |

**USE CASE 7:**




| **Use Case ID** | UC_007 | |
|---|---|---|
| **Use Case Name** | System Administrator Management | |
| **Description** | This use case involves the actions performed by a system administrator for managing user accounts and system configurations. | |
| **Primary Actor** | System Administrator | |
| **Secondary Actor** | User | |
| **Pre-Condition** | The system administrator must be logged into the system.. | |
| **Post-Condition** | Changes to user accounts and system configurations are successfully applied. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The system administrator accesses the administrator dashboard.<br>➢ The system administrator selects an action, such as managing user accounts or configuring system settings. | ➢ The system displays the administrator tools and options.<br>➢ The system processes the administrator's request and applies changes. |
| **Alternate Flow** | If an error occurs during the administrator's action:<br>➢ The system notifies the administrator of the error.<br>➢ The administrator takes corrective actions.<br>➢ Steps 3 to 4 are repeated. | |

# CHAPTER 4
# DESIGN

## 4.1.Architecture Diagram

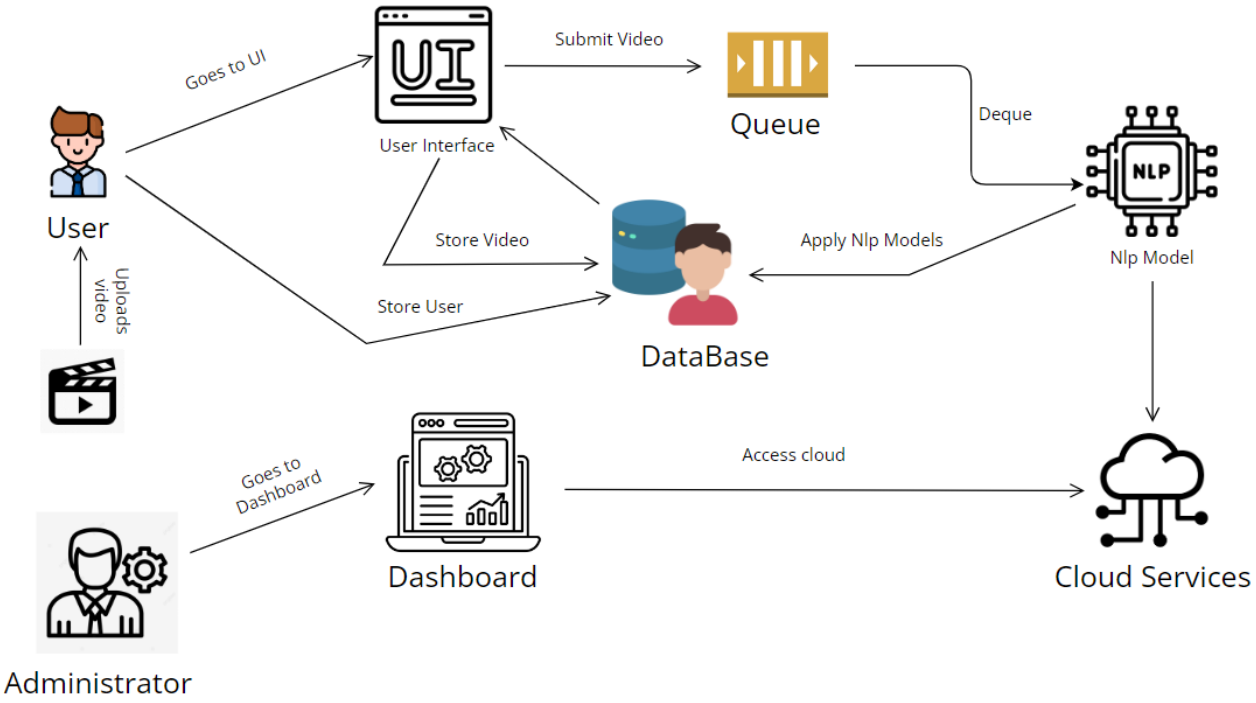
This architecture diagram outlines the components of a Sign Language Transcription System:

1. **User**: Represents individuals interacting with the system, providing user data and submitting sign language videos.
2. **SignLanguageVideo**: Represents videos of sign language submitted by users.
3. **UserDatabase**: Stores user information and sign language videos.
4. **UserInterface**: Web interface for users to interact with the system, displaying user information, and submitting sign language videos.
5. **MessageQueue**: Manages the queue of sign language videos for processing.
6. **TranslationService**: Utilizes NLP models to process sign language videos and generate text.
7. **CloudServices**: Integrates the message queue and translation service.
8. **Administrator**: Represents administrators with access to system management.
9. **AdministratorInterface**: Web interface for administrators to access the dashboard, select actions, and apply changes.

Key Interactions:

- Users submit sign language videos through the UserInterface.
- User data and videos are stored in the UserDatabase.
- MessageQueue manages the queue of videos for processing.
- TranslationService processes videos using NLP models.
- CloudServices integrate the message queue and translation service.
- Administrators manage the system through the AdministratorInterface.

This architecture facilitates the transcription of sign language videos into text, promoting inclusivity and accessibility.




## 4.2.ER Diagram

### 4.2.1. User Entity:

**Attributes: UserId**, **UserName**, **FirstName**, **LastName**, **Email**, **Password**, **RegistrationDate**, **LastLoginDate**, and other user-specific attributes.

**Relationships:**

A User can submit multiple SignLanguageVideos ("Submits").

A User can provide feedback on multiple Translations ("Provides Feedback").

### 4.2.2. SignLanguageVideo Entity:

**Attributes: VideoId**, **UserId** (foreign key), **VideoFile**, **Timestamp**, **SubmissionDate**, **Duration**, **Resolution**, and other video-specific attributes.

**Relationships:**

A User can submit SignLanguageVideos ("Submits" relationship).

Each SignLanguageVideo is associated with a Translation through **VideoId**.

### 4.2.3. Administrator Entity:

**Attributes: AdminId**, **AdminName**, **Email**, **Password**, **Role**, and other admin-specific attributes.

**Relationships:**

An Administrator can review multiple Translations ("Reviews" relationship).

### 4.2.4. Translation Entity:

**Attributes: TranslationId**, **VideoId** (foreign key), **Transcription**, **TranslationDate**, **FeedbackCount**, **AverageFeedbackRating**, **Language**, and other translation-specific attributes.

**Relationships:**

A Translation is associated with a SignLanguageVideo through **VideoId**.

A Translation can have multiple Feedback entries ("Has" relationship).

### 4.2.5. Feedback Entity:

**Attributes: FeedbackId**, **UserId** (foreign key), **TranslationId** (foreign key), **Rating**, **Comment**, **FeedbackDate**, and other feedback-specific attributes.

**Relationships:**

A User can provide feedback on Translations ("Provides Feedback" relationship).

A Translation can have multiple Feedback entries ("Has" relationship).

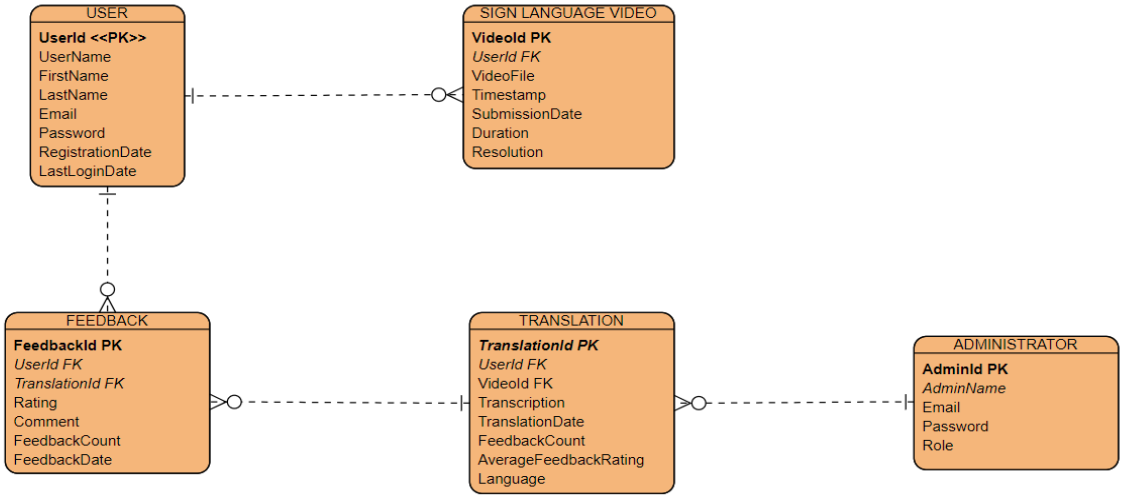



*Figure 1 -ER Diagram*

# Data Dictionary:

### User Entity

| FEILD | Data Type | Description |
|---|---|---|
| UserId **(PK)** | **INT** | Unique identifier for a user. |
| UserName | **VARCHAR** | User's username. |
| FirstName | **VARCHAR** | User's first name. |
| LastName | **VARCHAR** | User's last name. |
| Email | **VARCHAR** | User's email address. |
| Password | **VARCHAR** | User's hashed password. |
| RegistrationDate | **DATE** | Date when the user registered in the system. |
| LastLoginDate | **DATE** | Date of the user's last login. |

### SignLanguageVideo Entity

| FEILD | Data Type | Description |
|---|---|---|
| VideoId **(PK)** | **INT** | Unique identifier for a sign language video. |
| UserId **(FK)** | **INT** | Foreign key referencing the User entity. |

| | | |
|---|---|---|
| VideoFile | **VARCHAR** | File path or identifier for the sign language video. |
| Timestamp | **DATETIME** | Timestamp of when the video was submitted. |
| SubmissionDate | **DATE** | Date when the video was submitted. |
| Duration | **INT** | Duration of the sign language video in seconds. |
| Resolution | **VARCHAR** | Resolution of the video (e.g., HD, 4K). |

**ADMINISTRATOR ENTITY**

| FEILD | Data Type | Description |
|---|---|---|
| AdminId **(PK)** | **INT** | Unique identifier for an administrator. |
| AdminName | **VARCHAR** | Administrator's name. |
| Email | **VARCHAR** | Administrator's email address. |
| Password | **VARCHAR** | Administrator's hashed password. |
| Role | **VARCHAR** | Role or position of the administrator. |

**Translation Entity**

| FEILD | Data Type | Description |
|---|---|---|
| TranslationId **(PK)** | **INT** | Unique identifier for a translation. |
| VideoId **(FK)** | **INT** | Foreign key referencing the SignLanguageVideo entity. |
| Transcription | **TEXT** | Transcription of the sign language video. |
| TranslationDate | **DATE** | Date when the translation was generated. |
| FeedbackCount | **INT** | Number of feedback entries for the translation. |
| AverageFeedbackRating | **FLOAT** | Average rating from user feedback for the translation. |

| | | |
|---|---|---|
| Language | **VARCHAR** | Language of the transcription/translation (e.g., English). |

Feedback Entity

| **FEILD** | **Data Type** | **Description** |
|---|---|---|
| FeedbackId **(PK)** | **INT** | Unique identifier for a feedback entry. |
| UserId **(FK)** | **INT** | Foreign key referencing the User entity. |
| TranslationId **(FK)** | **INT** | Foreign key referencing the Translation entity. |
| Rating | **INT** | User's rating for the translation (e.g., 1 to 5). |
| Comment | **TEXT** | User's comments or feedback on the translation. |
| FeedbackDate | **DATE** | Date when the feedback was submitted. |

## 4.3. Data Flow Diagram

**User (External Agent):**

Represents the external user interacting with the Sign Language Transcription System.

**User Interface Process:**

**Responsibilities:**

Handles user interactions and requests.

## Functions:

Accepts user input for video submission, feedback, and other system interactions.

Provides a user-friendly interface for seamless interaction.

**Output:** Passes user requests and data to the Sign Language Video Processing and Feedback Handling System.

**Sign Language Video Processing Process:**

**Responsibilities:**

Processes sign language videos submitted by users.

**Functions:**

Extracts key features from sign language videos, including hand movements, facial expressions, and body language.

Prepares the video data for further analysis and translation.

**Output:** Sends processed video data to the Translation and Transcription Engine.

**Translation and Transcription Engine Process:**

**Responsibilities:**

Utilizes advanced Natural Language Processing (NLP) models to transcribe sign language into natural language text.

**Functions:**

Applies NLP techniques to understand sign language gestures and expressions.

Translates sign language features into coherent and understandable natural language text.

**Output:** Provides the transcribed text for further use and analysis.

**Feedback Handling System Process:**

**Responsibilities:**

Manages user feedback for system improvement.

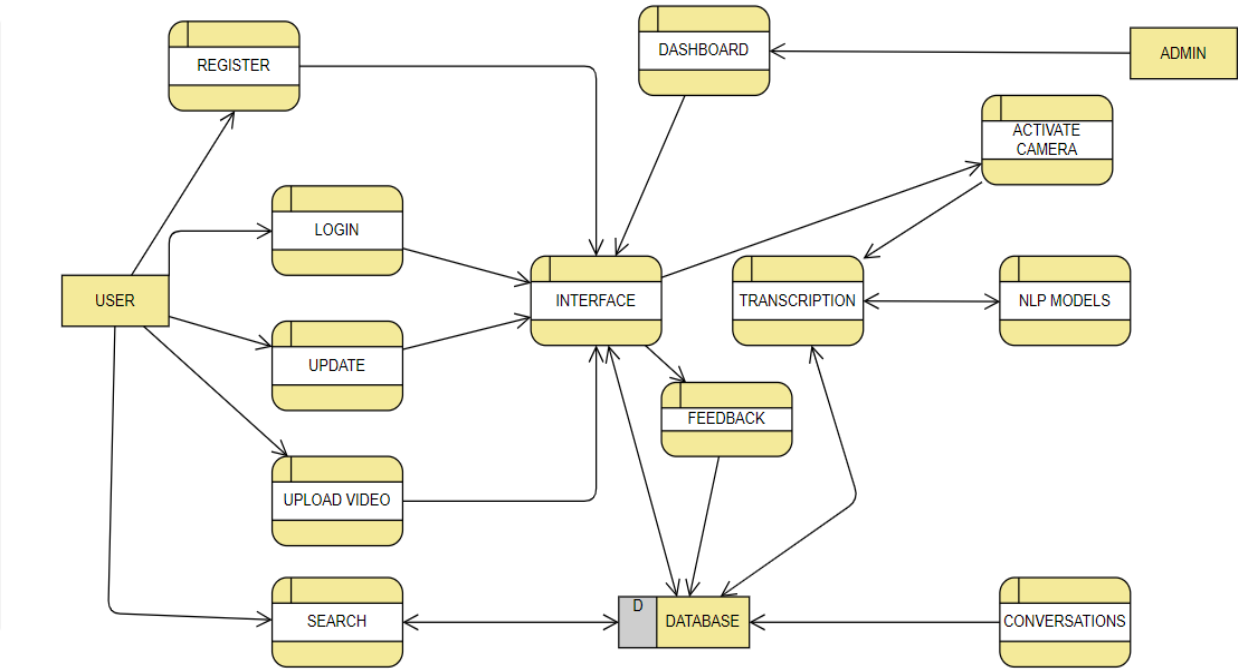
**Functions:**

Collects and processes user feedback on transcriptions and system performance.

Analyzes feedback to identify areas for improvement.

**Output:** Implements system enhancements based on user feedback.

In summary, this DFD outlines the main components and their interactions in the Sign Language Transcription System. Users interact with the User Interface to submit videos and provide feedback. The system processes videos, applies advanced NLP for transcription, and manages feedback to enhance system performance. The directional arrows represent the flow of data and control between these components, providing a visual representation of the system's functionality.




## 4.4. Class Diagram

The class diagram represents the key components and relationships in a Sign Language Transcription System:

1. **User, Administrator, Signer, Translator:**
    - Users interact with the system. Signers and Translators are specialized users with additional attributes.
2. **SignLanguageVideo:**
    - Represents videos submitted by users for sign language transcription.
3. **Translation:**
    - Holds information about the transcriptions of sign language videos.
4. **Feedback:**
    - Captures user feedback on transcriptions.
5. **SignLanguageTranscriptionSystem:**
    - Central system orchestrating UserInterface, VideoProcessing, and FeedbackHandling.
6. **UserInterface:**
    - Allows users to submit videos and feedback.
7. **VideoProcessing:**
    - Processes sign language videos, extracting features for transcription.
8. **TranslationAndTranscriptionEngine:**
    - Transcribes sign language features into written text.
9. **FeedbackHandling:**
    - Analyzes user feedback and suggests system enhancements.
10. **SystemEnhancement:**

- Represents enhancements based on feedback.

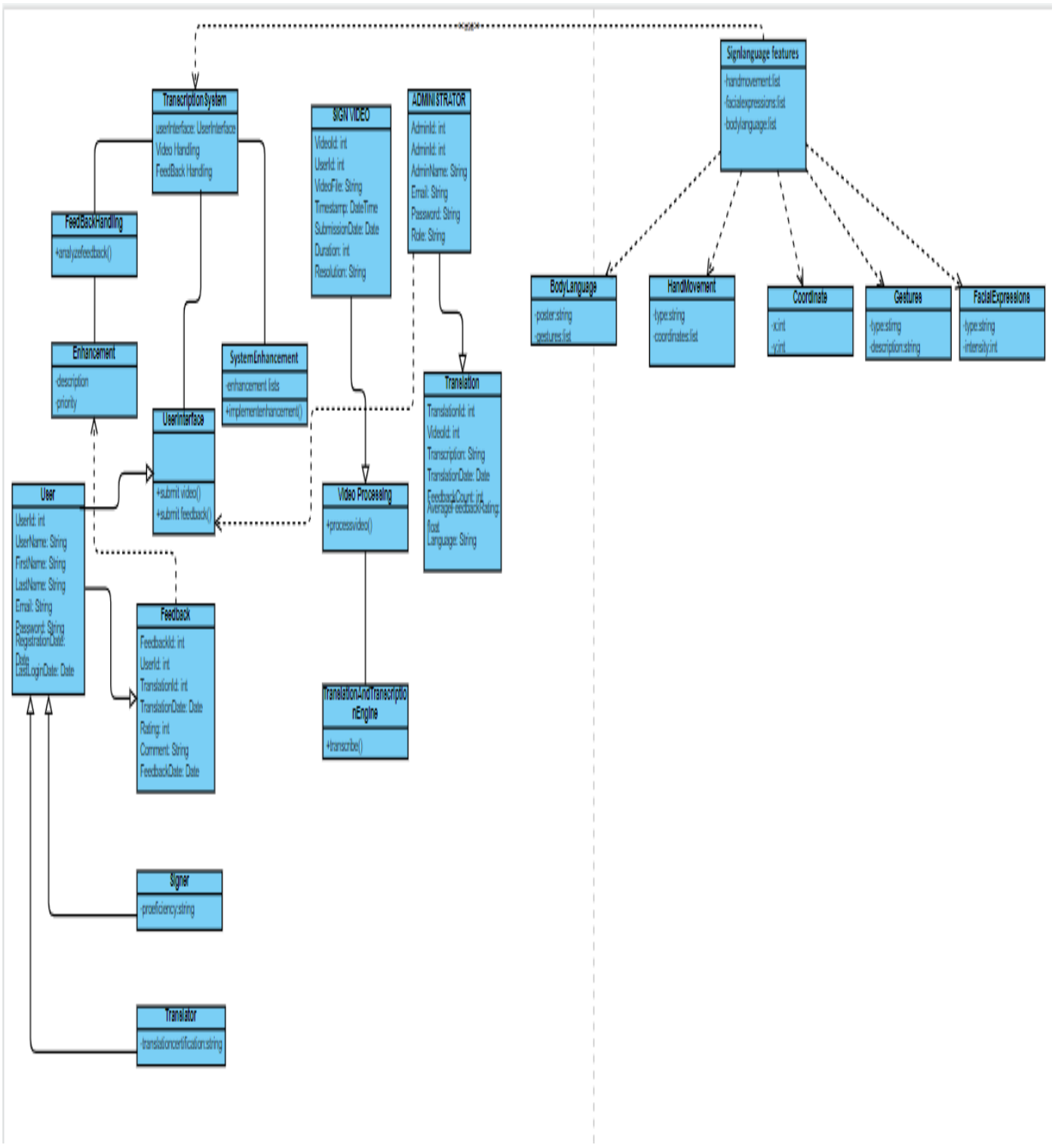
11. **Enhancement:**

- Describes specific system improvements.

12. **SignLanguageFeatures, HandMovement, FacialExpression, BodyLanguage, Coordinate, Gesture:**

- Classes related to the features extracted from sign language videos.

The diagram uses associations to show how these classes interact, and it includes inheritance to depict specialized user roles. The overall structure provides a foundation for building a comprehensive Sign Language Transcription System.


Signlanguage features
-handmovement:list
-facialexpressions:list
-bodylanguage:list
TranscriptionSystem
userInterface: UserInterface
Video Handling
FeedBack Handling
SIGN VIDEO
VideoId: int
UserId: int
VideoFile: String
Timestamp: DateTime
SubmissionDate: Date
Duration: int
Resolution: String
ADMINISTRATOR
AdminId: int
AdminId: int
AdminName: String
Email: String
Password: String
Role: String
FeedBackHandling
+analyzefeedback()
BodyLanguage
-poster:string
-gestures:list
HandMovement
-type:string
-coordinates:list
Coordinate
-x:int
-y:int
Gestures
-type:string
-description:string
FacialExpressions
-type:string
-intensity:int
Enhancement
-description
-priority
SystemEnhancement
-enhancement lists
+implementenhancement()
Translation
TranslationId: int
VideoId: int
Transcription: String
TranslationDate: Date
FeedbackCount: int
AverageFeedbackRating: float
Language: String
UserInterface
+submit video()
+submit feedback()
User
UserId: int
UserName: String
FirstName: String
LastName: String
Email: String
Password: String
RegistrationDate: Date
LastLoginDate: Date
Video Processing
+processvideo()
Feedback
FeedbackId: int
UserId: int
TranslationId: int
TranslationDate: Date
Rating: int
Comment: String
FeedbackDate: Date
Translation-andTranscription Engine
+transcribe()
Signer
-proficiency:string
Translator
-translationcertification:string

## 4.5.Activity Diagrams

### 1. Register

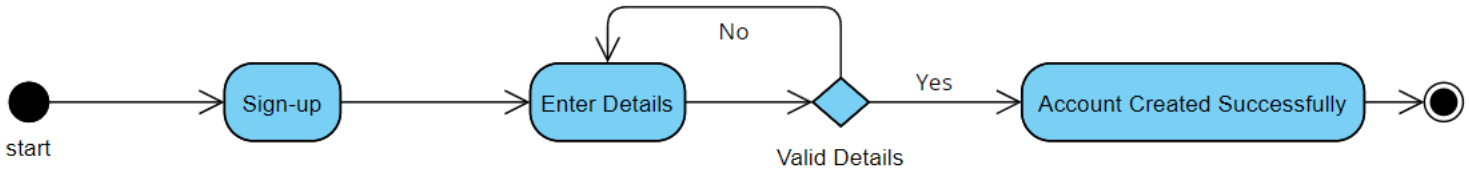



*Figure 2- Requirement FR_01*

### 2. Login

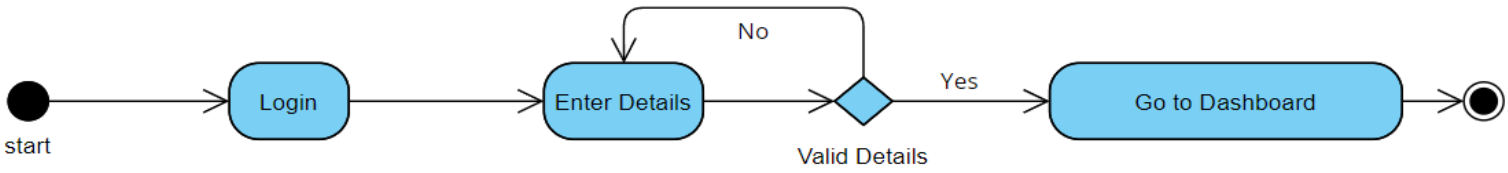



*Figure 3- Requirement FR_02*

### 3. Video Transcription

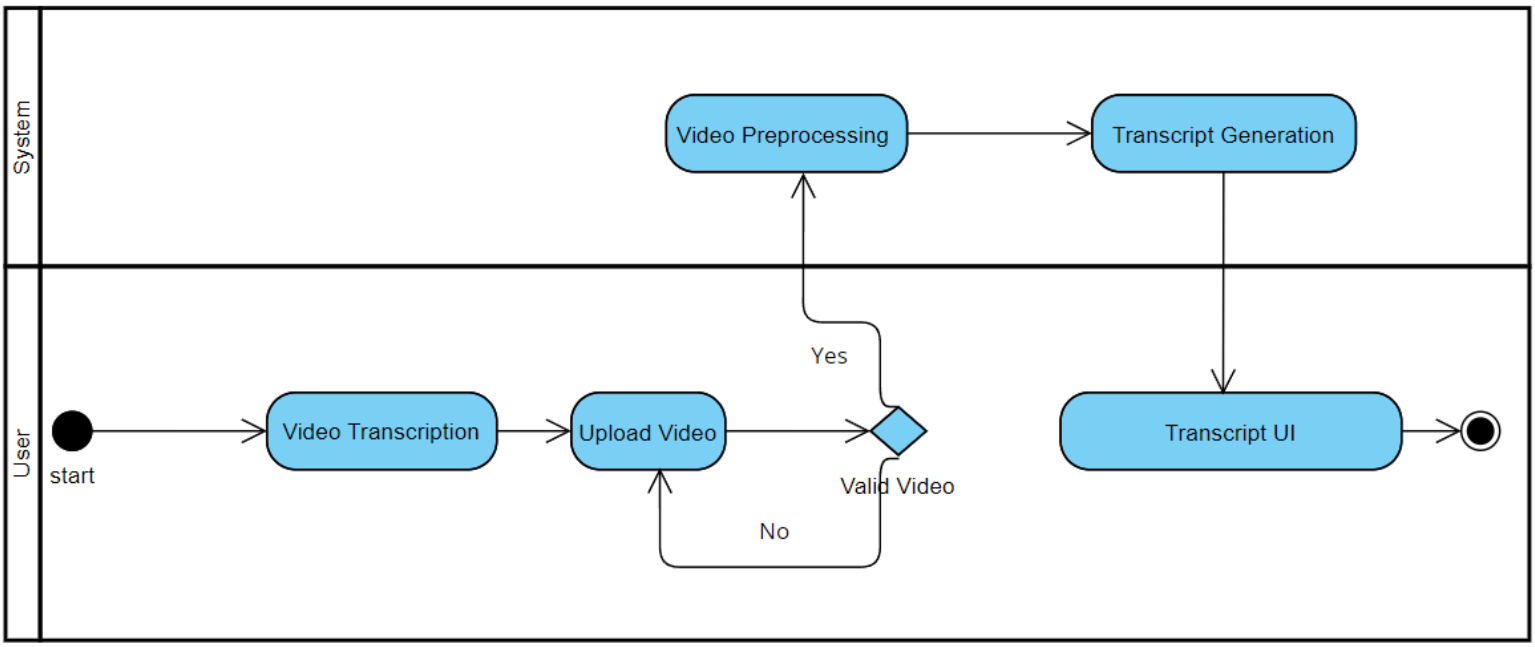



*Figure 4- Requirement FR_03*

### 4. Real Time Transcription

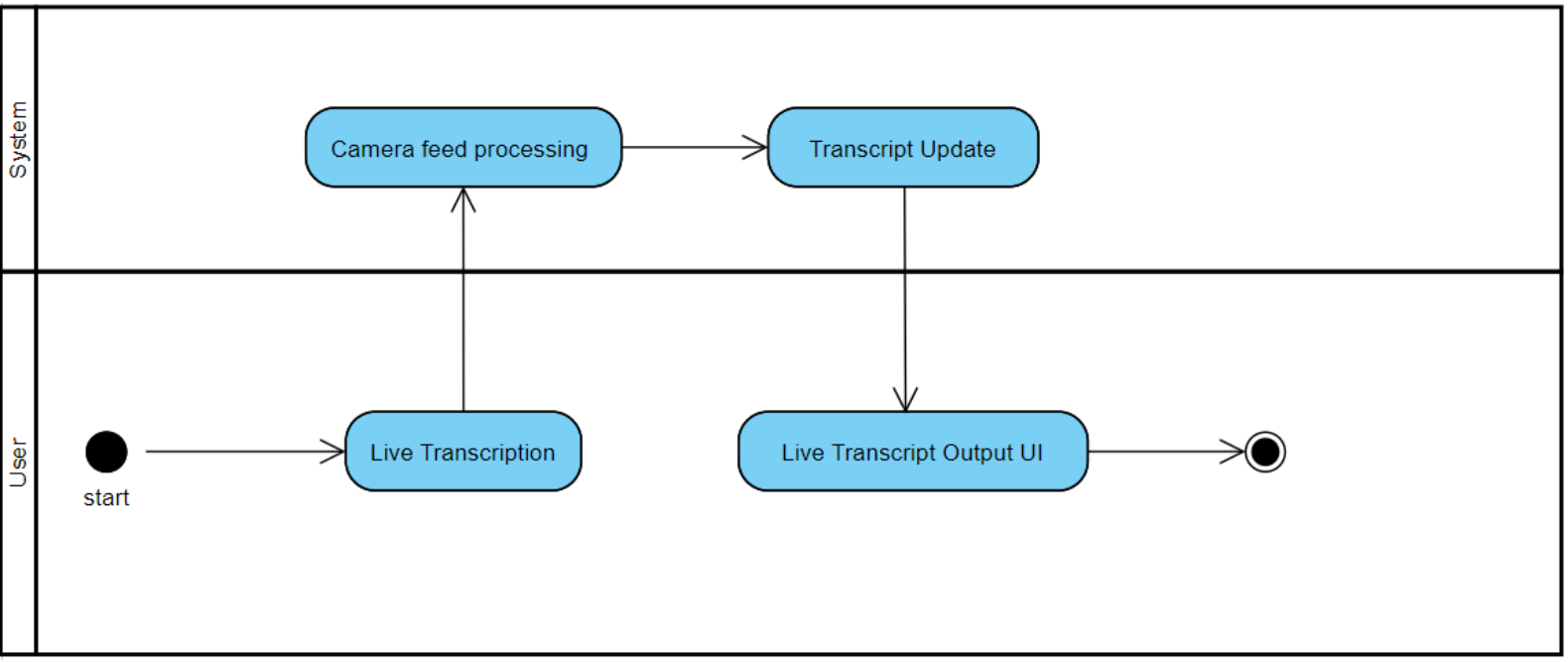



*Figure 5- Requirement FR_04*

### 5.Track History

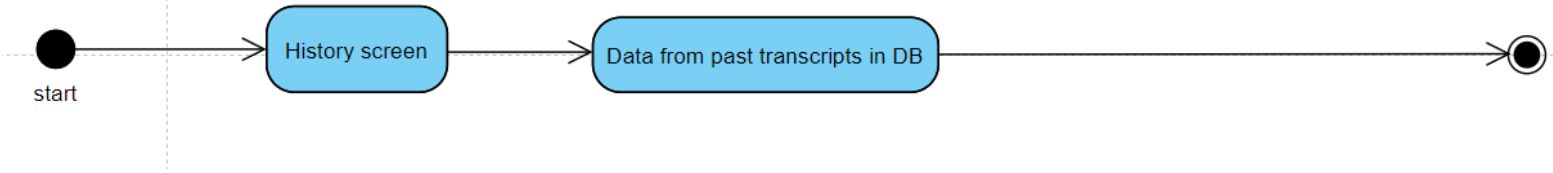



*Figure 6- Requirement FR_05*

## 4.6.Sequence Diagram

### 1. Sign-up

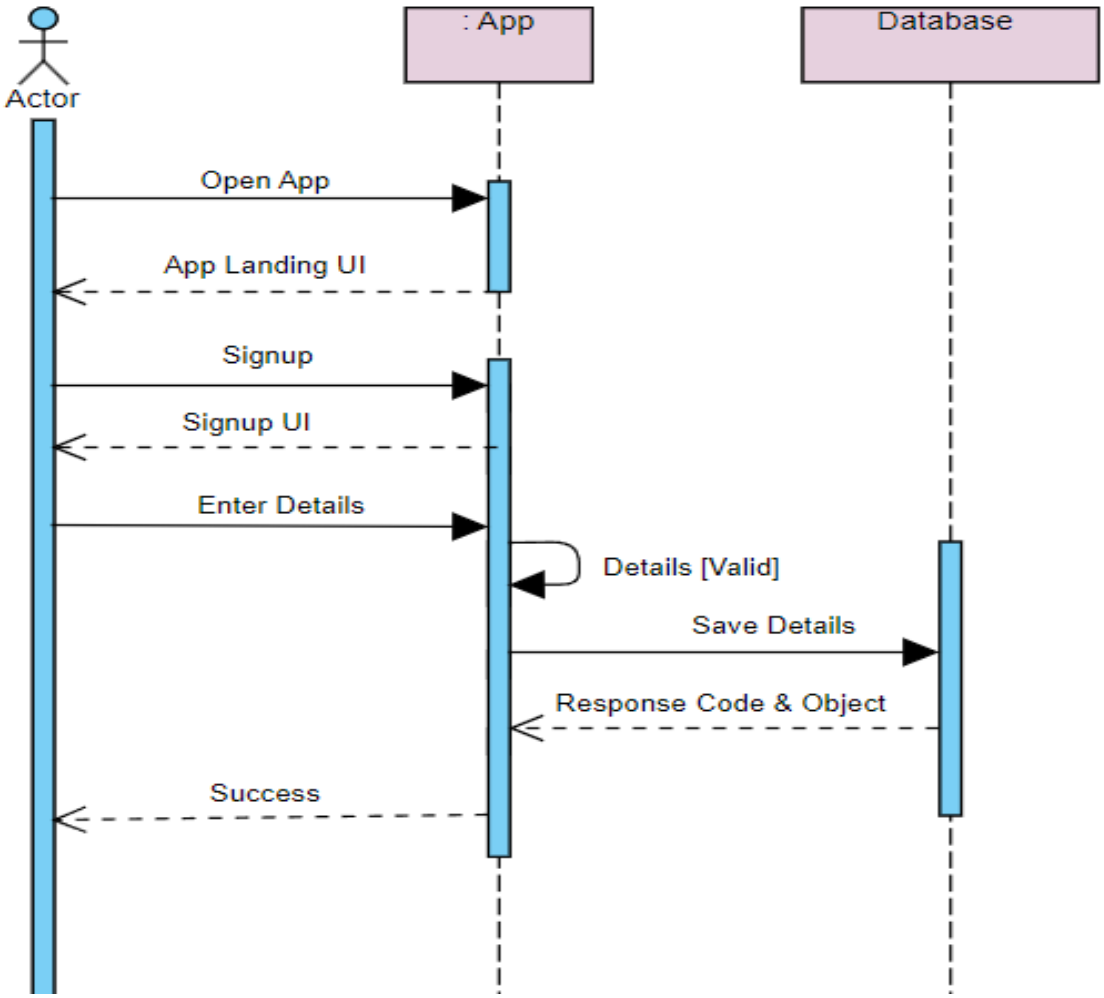



*Figure 7 - Requirement FR_01*

**2. Login**

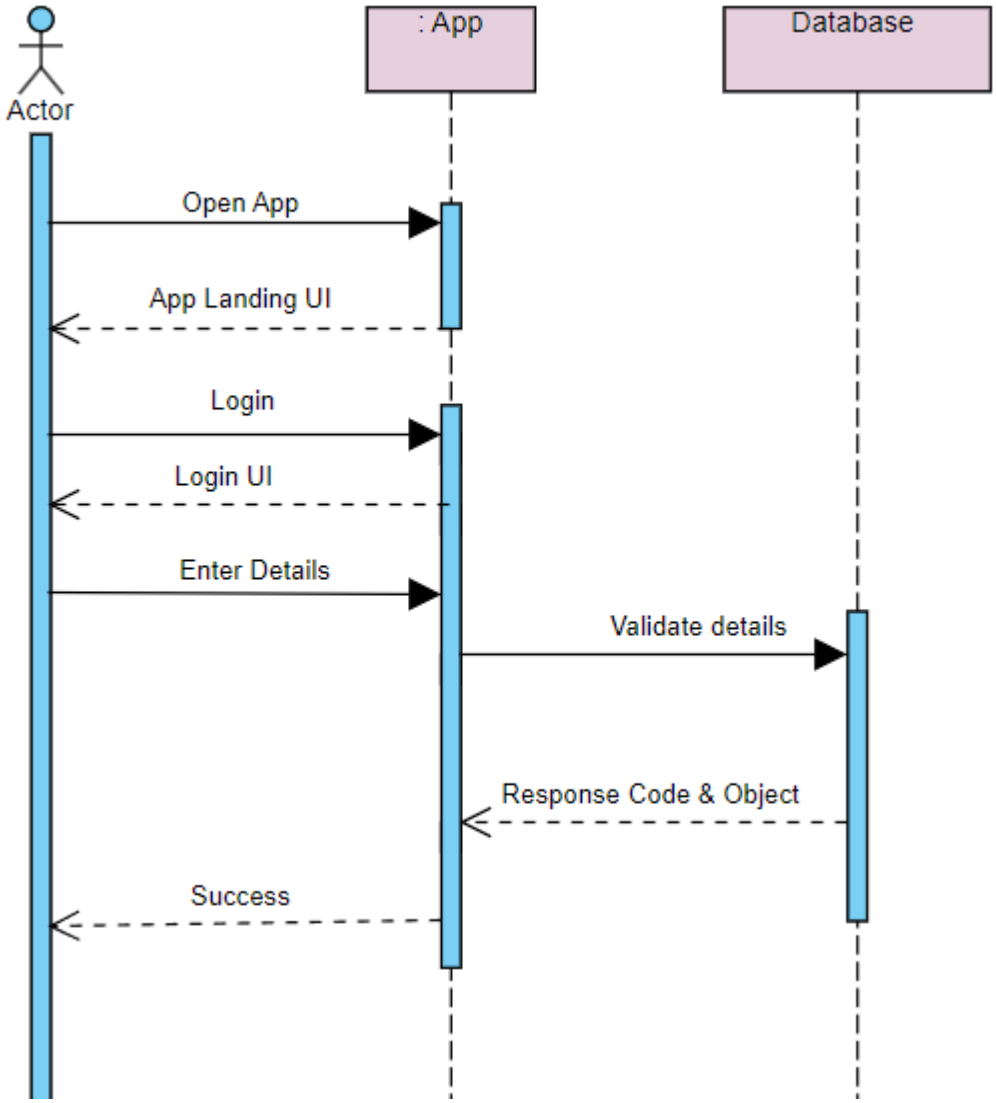



*Figure 8- Requirement FR_02*

### 3. Video Transcription

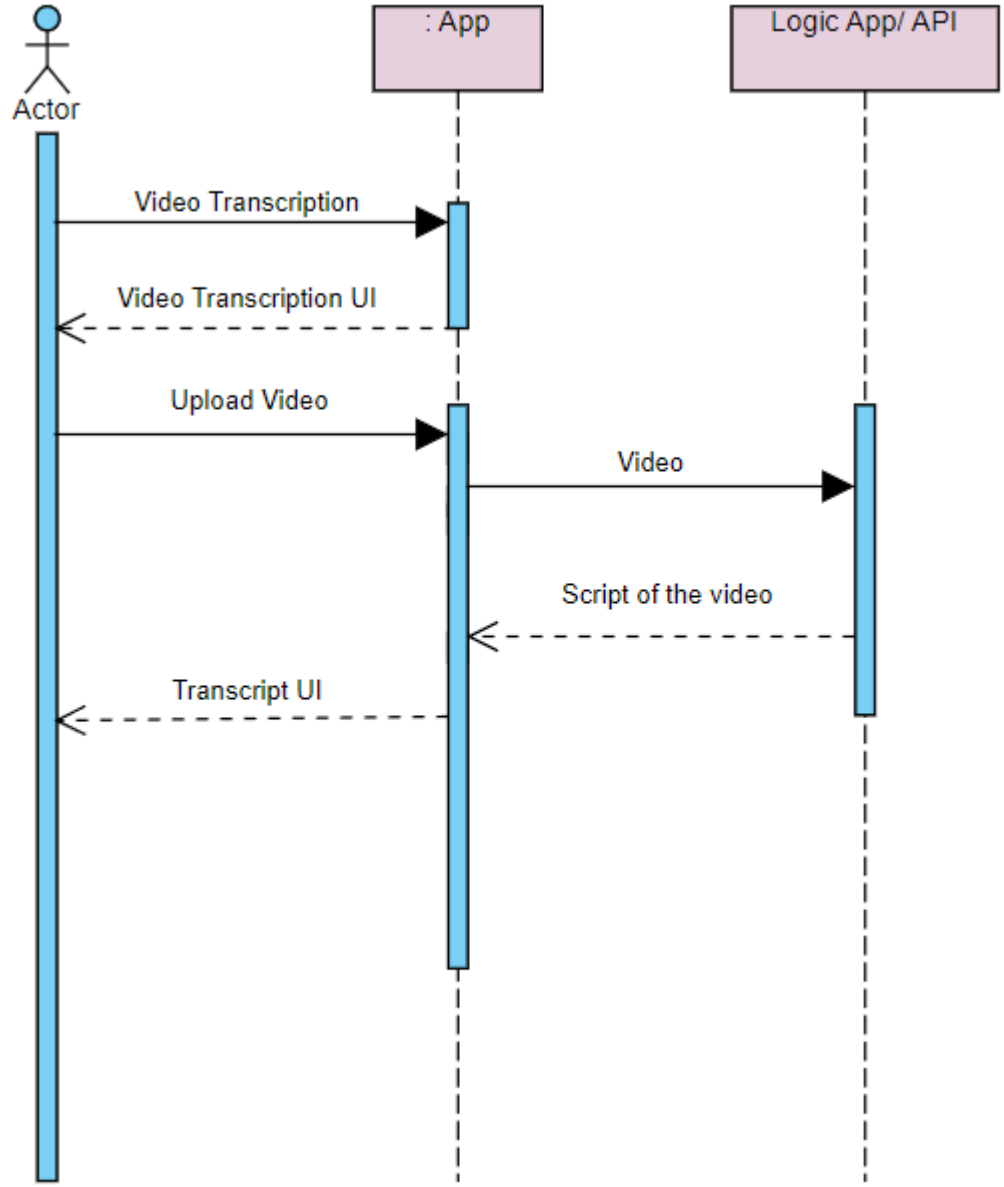



*Figure 9 - Requirement FR_O3*

### 4. Real-Time Transcription

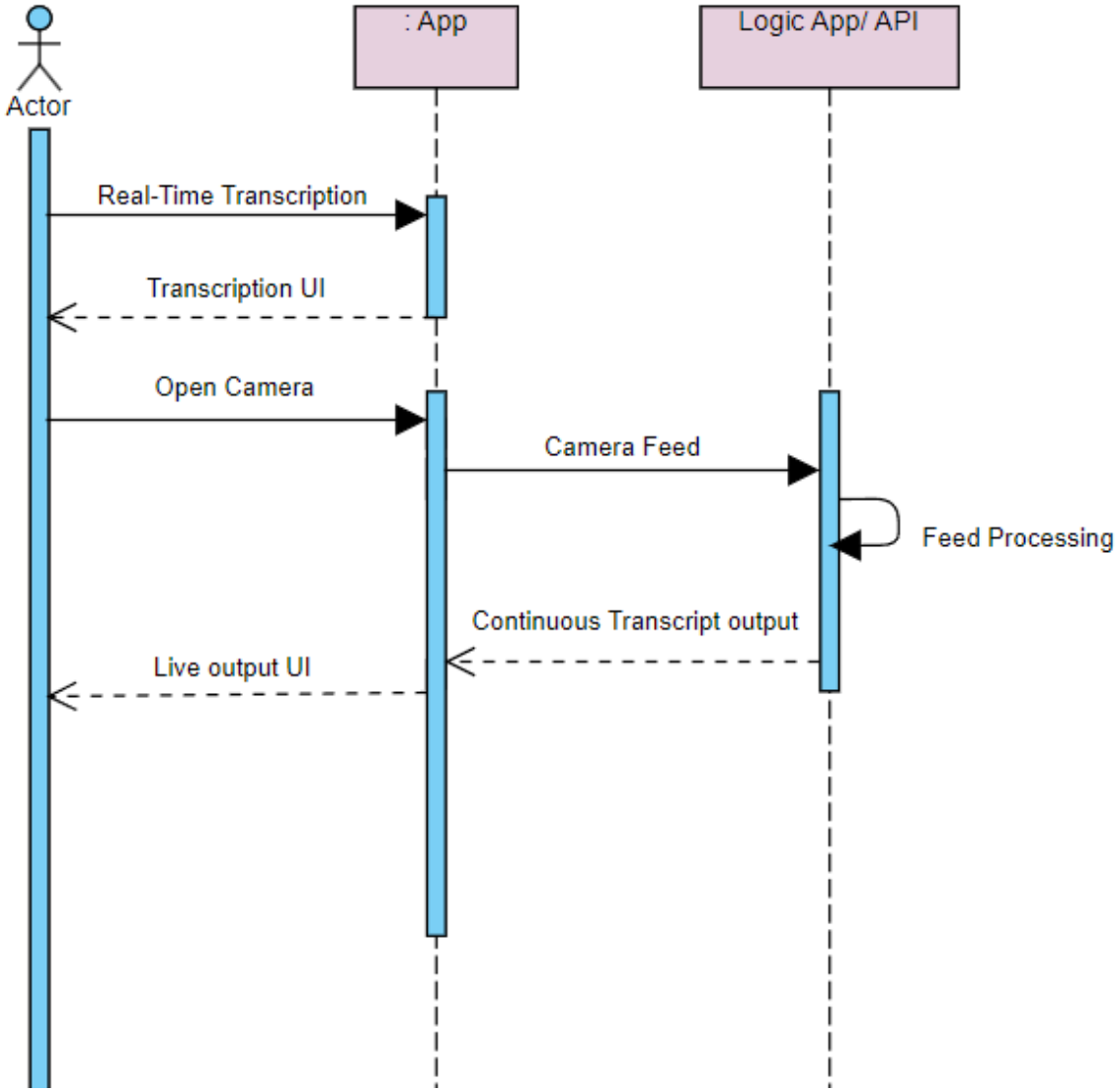


*Figure 9 - Requirement FR_O4*

**5. Track History**

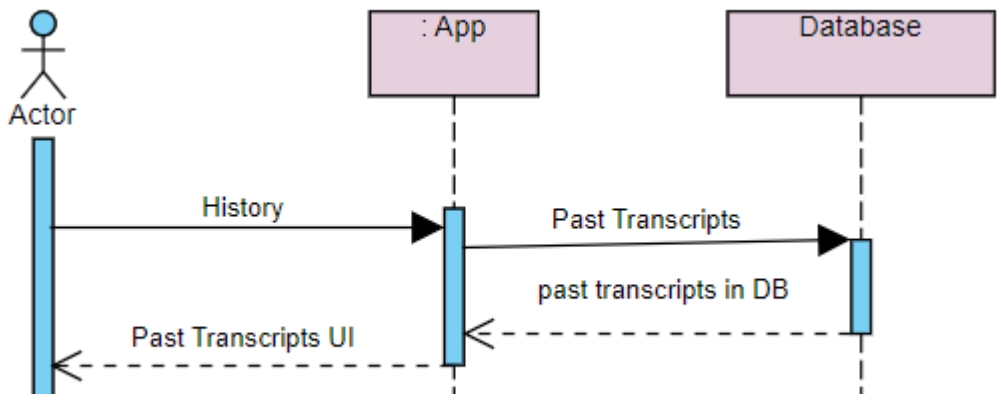



*Figure 10 - Requirement FR_O5*

## 4.6 Collaboration Diagram

1. **Register**

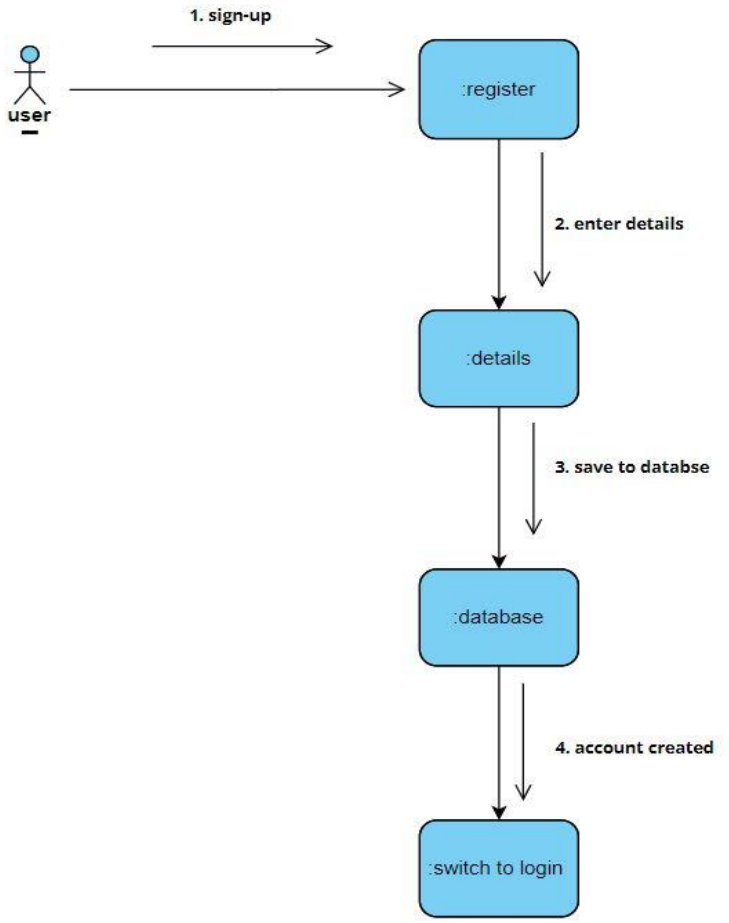



*Figure 111- Requirement FR_01*

**2. Login**

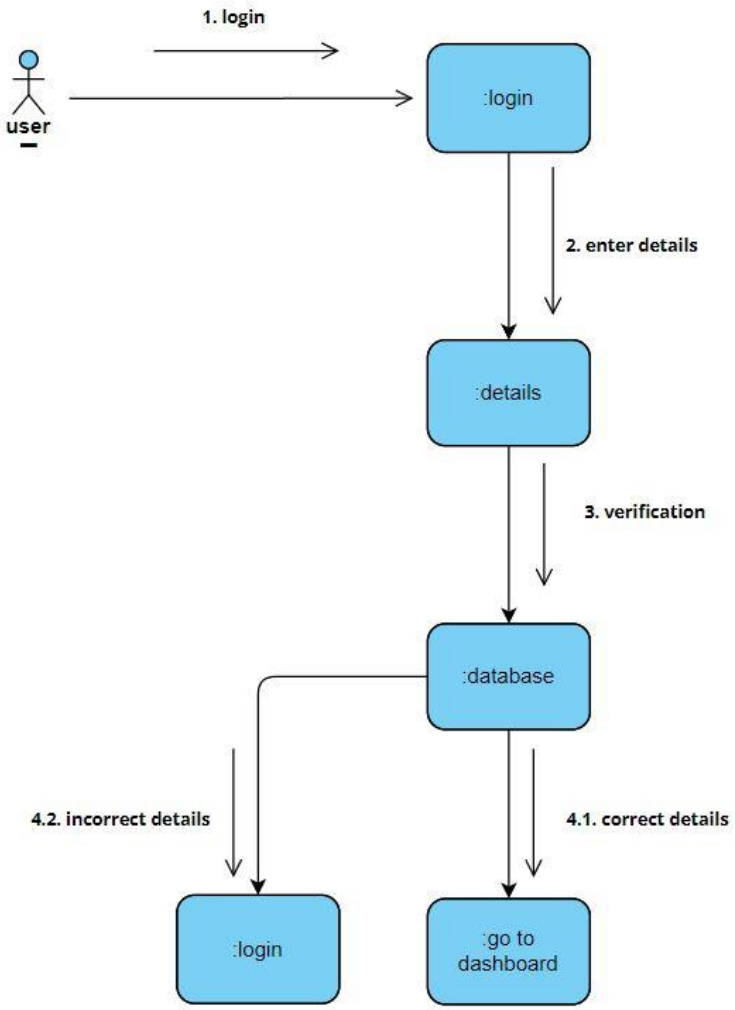



*Figure 12- Requirement FR_02*

## 3. Video Transcription

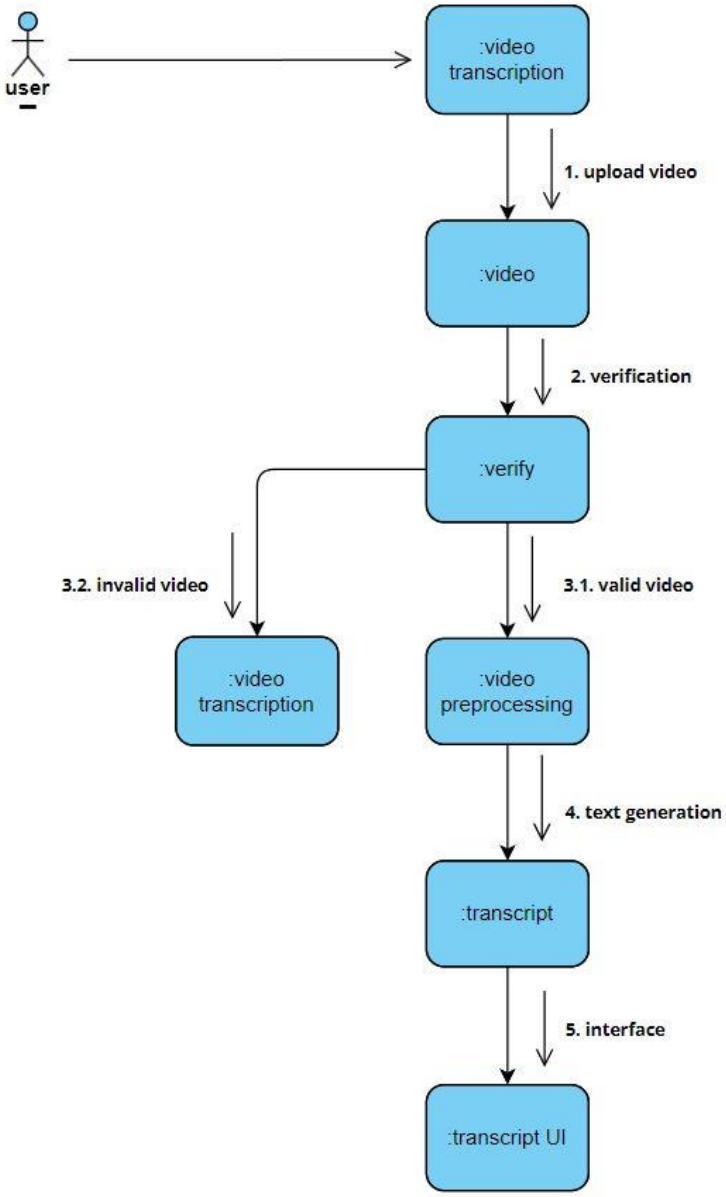



*Figure 13- Requirement FR_03*

## 4. Real-Time Transcription

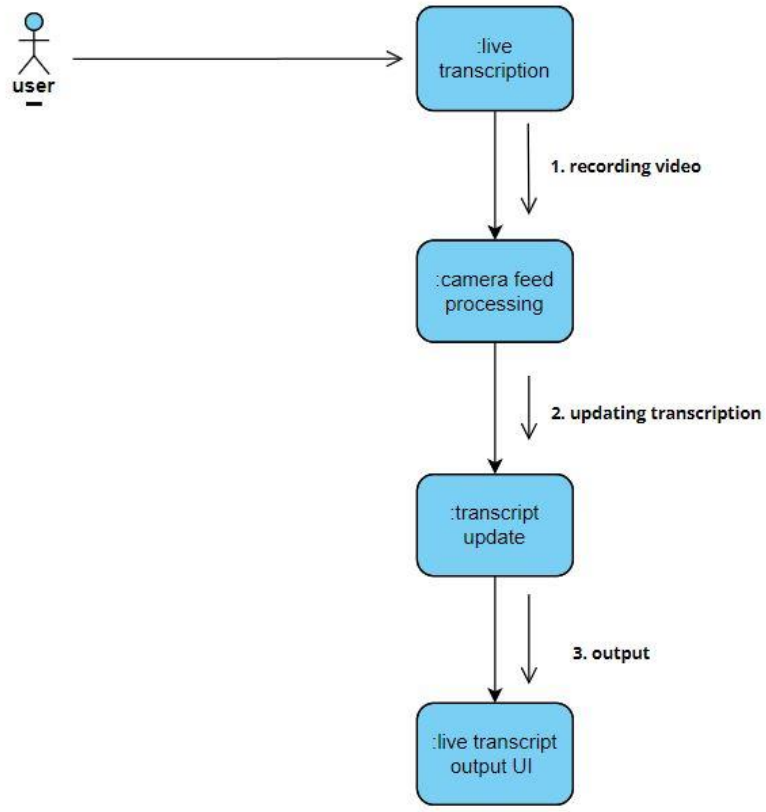



*Figure 14- Requirement FR_04*

## 5. Track History

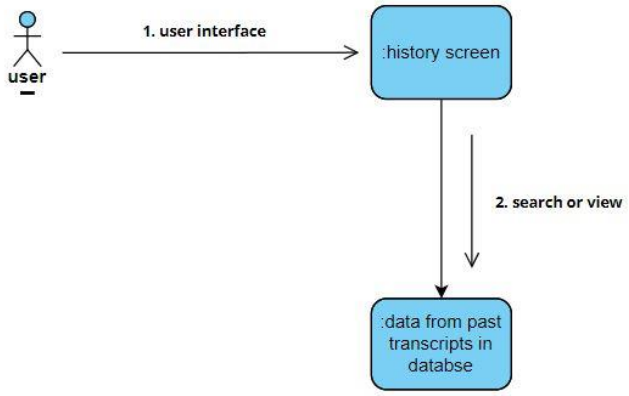
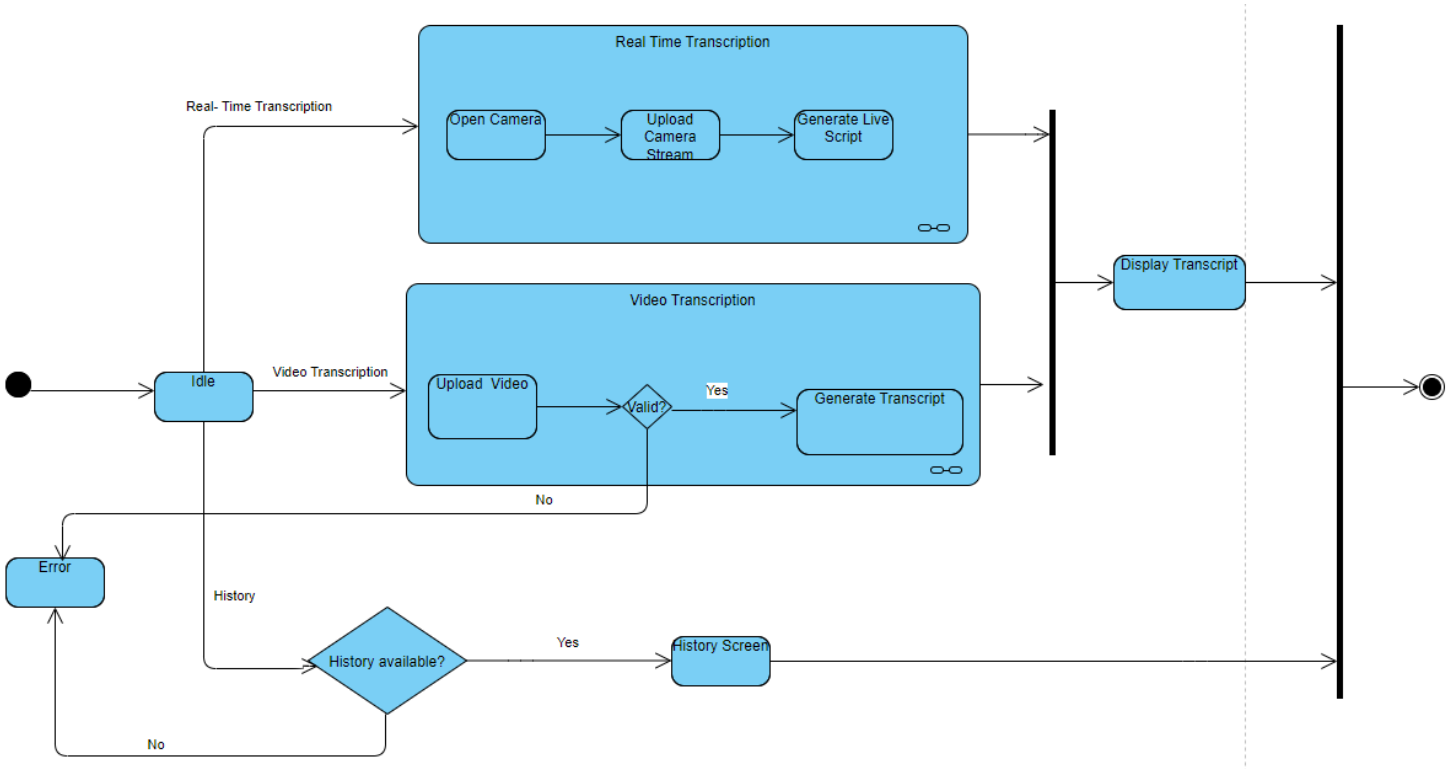
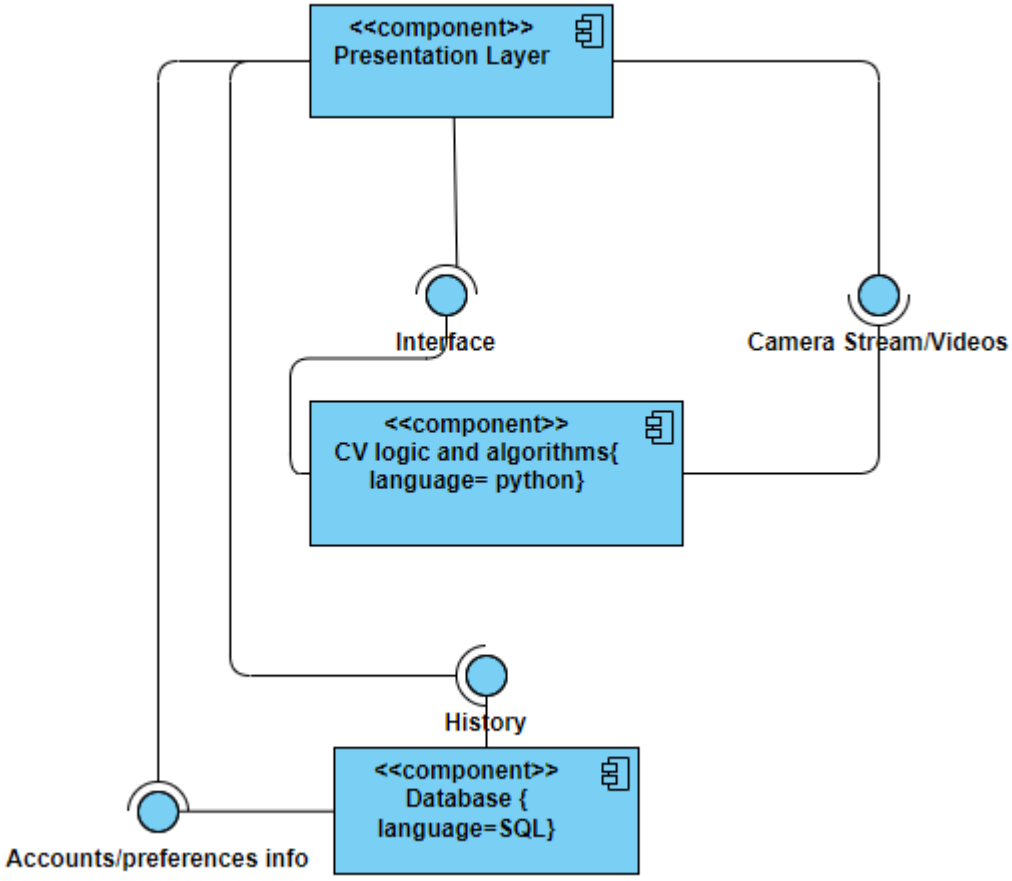



*Figure 15- Requirement FR_05*

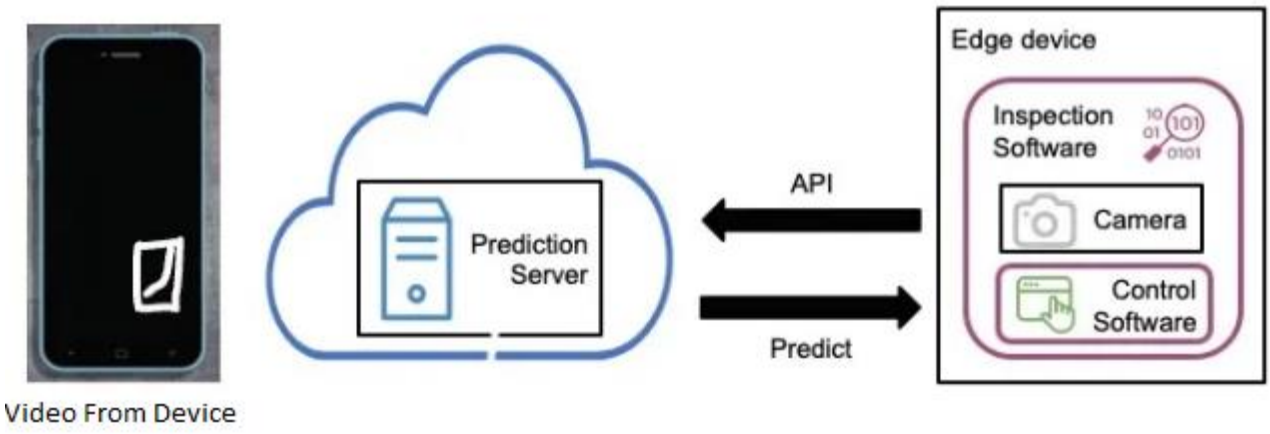
## 4.7.State Transition Diagram



## 4.8. Component Diagram


<<component>>
Presentation Layer
Interface
Camera Stream/Videos
<<component>>
CV logic and algorithms{
language= python}
History
<<component>>
Database {
language=SQL}
Accounts/preferences info

## 4.9.Deployment Diagram



# CHAPTER 5

# TESTING

## 5.1 Test Case Specification

### Login

| Positive Test Case | |
|---|---|
| ID | LOGIN_POSITIVE |
| Priority | High |
| Description | To verify user authentication to system. |
| Reference | Functional Requirement reference |
| Users | Administrator. |
| Pre-requisites | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access. |
| Steps | A Open the web link to system.<br>B Enter login id<br>C Enter Password.<br>D Press Login. |
| Input | Login id and password |
| Expected result | Successfully enters the system and main home page opens. |
| Status | Tested, passed. |

| Negative Test Case | |
|---|---|
| ID | TC_LOGIN_FAILURE |
| Priority | High |
| Description | To verify user authentication to system. |
| Reference | Functional Requirement reference |
| Users | Administrator. |
| Pre-requisites | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access. |
| Steps | A Open the web link to system.<br>B Enter login id.<br>C Enter Password.<br>D Press Login. |
| Input | Incorrect Login id or password or deactivated credentials. |
| Expected result | Does not allows access to system features and notifies the error. |
| Status | Tested, passed. |

## Signup

| Positive Test Case | |
|---|---|
| **ID** | LOGIN_POSITIVE |
| **Priority** | High |
| **Description** | To verify user authentication to system. |
| **Reference** | Functional Requirement reference |
| **Users** | New users |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access. |
| **Steps** | A Open the application<br>B Navigate to the registration/signup page<br>C Enter valid username/email, password, full name, address, and cell number<br>D Click on the "Register" button |
| **Input** | Valid username/email, valid password, valid full name |
| **Expected result** | Successfully enters the system and main home page opens. |
| **Status** | Tested, passed. |

| Negative Test Case | |
|---|---|
| **ID** | TC_LOGIN_FAILURE |
| **Priority** | High |
| **Description** | To verify user authentication to system. |
| **Reference** | Functional Requirement reference |
| **Users** | New users. |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access. |
| **Steps** | A Open the application<br>B Navigate to the registration/signup page<br>C Enter valid username/email, password, full name, address, and cell number<br>D Click on the "Register" button |
| **Input** | Invalid/incomplete username/email, invalid/incomplete password, invalid/incomplete full name, invalid/incomplete address, invalid/incomplete cell number<br>. |
| **Expected result** | Does not allows access to system features and notifies the error. |
| **Status** | Tested, passed. |

## Uploading Video

| **Positive Test Case** |
|---|

| | |
|---|---|
| **ID** | TC_UPLOAD _POSITIVE |
| **Priority** | High |
| **Description** | Verify that the user can upload a video file (mp4) |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | D System is online.<br>E User must have active login credentials provided by system administrator.<br>F User has internet access. |
| **Steps** | E Open the application<br>F Navigate to the Upload Video Section<br>G Select Valid mp4 format file from storage<br>H Click on the "Upload" button |
| **Input** | Valid MP4 file |
| **Expected result** | The video is successfully uploaded and displayed on the page, “Get Transcript” button appears |
| **Status** | Tested, passed. |

| **Negative Test Case** | |
|---|---|
| **ID** | TC_UPLOAD _FAILURE |
| **Priority** | High |
| **Description** | Verify that the user can upload a video file (mp4) |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | G System is online.<br>H User must have active login credentials provided by system administrator.<br>D User has internet access. |
| **Steps** | I Open the application<br>J Navigate to the Upload Video Section<br>K Select invalid/corrupted mp4 format file from storage or any other file format.<br>E Click on the "Upload" button |
| **Input** | Corrupted MP4 file, or Non-MP4 files |
| **Expected result** | Nothing uploads, no “Get Transcript” |
| **Status** | Tested, passed. |

# Camera

| Positive Test Case | |
|---|---|
| **ID** | TC_CAMERA_POSITIVE |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users |

| | |
|---|---|
| **Pre-requisites** | I System is online.<br>J User must have active login credentials provided by system administrator.<br>K User has internet access.<br>L User's device has a camera of webcam privileges attached |
| **Steps** | L Open the application<br>M Navigate to the Real Time Transcription Button |
| **Input** | Permission to use camera |
| **Expected result** | Successfully enters the real time transcription screen |
| **Status** | Tested, passed. |

| Negative Test Case | |
|---|---|
| **ID** | TC_CAMERA _FAILURE |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users. |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access.<br>D User's device has a camera of webcam privileges attached |
| **Steps** | A Open the application<br>B Navigate to the Real Time Transcription Button |
| **Input** | Does not allow camera permissions |
| **Expected result** | Does not allows access to real time transcription screen and notifies the error. |
| **Status** | Tested, passed. |

| Negative Test Case | |
|---|---|
| **ID** | TC_CAMERA _FAILURE_02 |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users. |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access.<br>D User's device does not have a camera of webcam privileges attached |

| | |
|---|---|
| **Steps** | A Open the application |
| **Input** | .- |
| **Expected result** | No Real Time Transcription Button Visible |
| **Status** | Tested, passed. |

## Downloading Transcripts

| Positive Test Case | |
|---|---|
| **ID** | TC_TRACRIPTS _POSITIVE |
| **Priority** | High |
| **Description** | To verify the system generates a transcript. |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access.<br>D A valid video has been procced successfully or a real time session has been ended |
| **Steps** | A Click on “Get Transcript” button if visible |
| **Input** | - |
| **Expected result** | Successfully downloads a .txt transcript and opens an editable dialog |
| **Status** | Tested, passed. |

| Negative Test Case | |
|---|---|
| **ID** | TC_TRACRIPTS _POSITIVE |
| **Priority** | High |
| **Description** | To verify the system generates a transcript. |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | A System is online.<br>B User must have active login credentials provided by system administrator.<br>C User has internet access.<br>D A valid format video has been procced successfully or a real time session has been ended |

| | |
|---|---|
| **Steps** | B If the feed could not a understood for the whole length of the video or session, the system will notify user with error. |
| **Input** | valid format video or real time session with no sign-language |
| **Expected result** | If the feed could not a understood for the whole length of the video or session, the system will notify user with error. |
| **Status** | Tested, passed. |

## 5.2 Black Box Testing

Black Box Testing, alternatively termed Behavioral Testing, is a software testing approach wherein the tester is unaware of the internal structure, design, or implementation details of the item under examination. The tests conducted within this methodology can pertain to either functional or non-functional aspects, although they typically focus on functionality.

### 5.2.1 **Equivalence Partitioning (EP)**

Equivalence Partitioning (EP) is a highly employed technique aimed at reducing the volume of necessary test cases for system evaluation. This method is widely utilized to categorize inputs into equivalence classes, thereby streamlining the testing process.

**Login**

| Inputs | Valid Partitions | Invalid Partitions |
|---|---|---|
| Username/Email | Compulsory field. Case insensitive. Contains alphabets [A-Z, a-z], numeric keys [0-9], special keys [., _]. Must contain @ sign and domain suffix (e.g., .com). | Empty field. Doesn't contain @ or domain suffix. Contains invalid characters. |
| **Password** | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 5 characters. Contains invalid characters (based on specific system rules). |

**Signup**

| Inputs | Valid Partitions | Invalid Partitions |
|---|---|---|
| **Full Name** | Compulsory field. Contains more than 1 character. Contains alphabets [A-Z, a-z] and spaces. | Empty field. Contains only numeric or special characters. |
| **Email** | Compulsory field. Case sensitive. Valid email address. Alphabets [A-Z, a-z], numeric keys [0-9], special keys [., _]. Contains @ sign and .com. | Empty field. Doesn't contain @ or .com. |
| **Password** | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 8 characters. |
| **Confirm Password** | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 8 characters. |

### 5.2.2 Boundary Value Analysis (Character Count)

**Login**

| Inputs | Minimum boundary | | Maximum boundary | |
|---|---|---|---|---|
| | Min - 1 | Min | Max | Max + 1 |
| Email | 11 | 12 | 254 | 255 |
| Password | 7 | 8 | 254 | 255 |

**Signup**

| Inputs | Minimum boundary | | Maximum boundary | |
|---|---|---|---|---|
| | Min - 1 | Min | Max | Max + 1 |
| Email | 11 | 12 | 254 | 255 |
| Password | 7 | 8 | 254 | 255 |
| Name | 3 | 4 | 254 | 255 |

### 5.2.3 Decision Table Testing (DT)

Decision table is a testing method, which aims to ensure that each one of the possible branches from each decision point is executed at least once and thereby ensuring that all reachable code is executed.

**Signup**

| Inputs | Values | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 |
|---|---|---|---|---|---|---|
| Name | T/F | T | F | T | T | F |
| Email | T/F | T | T | F | T | F |
| Password | T/F | T | T | T | F | F |
| Status | T/F | Signup | Error | Error | Error | Error |

**Login**

| Inputs | Values | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 |
|---|---|---|---|---|---|---|
| Email | T/F | T | T | F | T | F |
| Password | T/F | T | T | T | F | F |
| Status | T/F | Login | Error | Error | Error | Error |

**Video Upload**

| Inputs | Values | Case 1 | Case 2 |
|---|---|---|---|
| Video | Valid/Invalid | Valid | Invalid |
| Status | Uploaded/Error | Uploaded | Error |

### 5.2.4 State Transition Testing

State transition testing is a methodical approach used to validate the behavior of a system when it undergoes changes in state based on various events or inputs. This testing technique ensures that all possible states and transitions are covered, helping to identify any unexpected behaviors or errors in the system's workflow.

| State | Name | Events | New User | Existing User | Login | Signup | Video Upload | View Transcript | Logout |
|---|---|---|---|---|---|---|---|---|---|
| **S1** | Start | Open App | S1 | S1 | - | - | - | - | - |
| **S2** | User Registration | Submit Registration | S2 | - | - | S2 | - | - | - |
| **S3** | User Login | Submit Login | - | S3 | S3 | - | - | - | - |
| **S4** | Video Upload | Upload Video | - | - | S4 | - | S4 | - | - |
| **S5** | Request Transcript | Request Transcription | - | - | - | - | S5 | S5 | - |
| **S6** | View Transcript | View Transcription | - | - | - | - | - | S6 | - |
| **S7** | Logout | Logout | - | - | - | - | - | - | S7 |

### 5.2.5 Use Case Testing

Signup

| **Use Case ID** | UC_001 | |
|---|---|---|
| **Use Case Name** | Registration/Sign up | |
| **Description** | This use case involves the process of creating a new account within the Sign Language Transcription System, allowing users to access and utilize the system's features. | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user accesses the registration interface.<br>➢ The user enters the required information, such as username, email, and password.<br>➢ The user submits the registration form.<br>➢ The system notifies the user of successful account creation. | ➢ The system displays the account creation form.<br>➢ The system validates the entered information.<br>➢ The system processes the registration request and creates a new user account. |

| | |
|---|---|
| **Alternate Flow** | If the entered information is incomplete or fails validation:<br>➢ The system notifies the user of the validation error.<br>➢ The user corrects the information and resubmits the form.<br>➢ Steps 4 to 8 are repeated. |

## Login

| | | |
|---|---|---|
| **Use Case ID** | UC_002 | |
| **Use Case Name** | Login | |
| **Description** | User can login to the system | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user accesses the login interface.<br>➢ The user enters their username/email and password.<br>➢ The user submits the login form. | ➢ The system displays the login form.<br>➢ The system validates the login credentials.<br>➢ The system processes the login request and grants access to the user. |
| **Alternate Flow** | If the entered credentials are incorrect:<br>➢ The system notifies the user of the authentication failure.<br>➢ The user retries the login with correct credentials.<br>➢ Steps 4 to 6 are repeated. | |

## Real Time Transcription

| | |
|---|---|
| **Use Case ID** | UC_004 |
| **Use Case Name** | Perform Sign Language Translation |
| **Description** | This use case involves the system translating live Sign Language gestures into natural language in real-time. |
| **Primary Actor** | User (Sign Language User) |
| **Secondary Actor** | None |

| | | |
|---|---|---|
| **Pre-Condition** | The user must be logged into the system, and the device must have access to a camera. | |
| **Post-Condition** | The system successfully transcribes the Sign Language gestures into natural language. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
| | ➢ The user selects the "Real-Time Transcription" option.<br>➢ The user performs Sign Language gestures in front of the camera. | ➢ The system activates the camera for live translation.<br>➢ The system processes the live video feed, extracting key features.<br>➢ The system applies Natural Language Processing (NLP) models to translate gestures into text. |
| **Alternate Flow** | If the system encounters difficulty in recognizing gestures:<br>➢ The system may prompt the user to adjust lighting or perform clearer gestures.<br>➢ Steps 3 to 5 are repeated until successful translation. | |

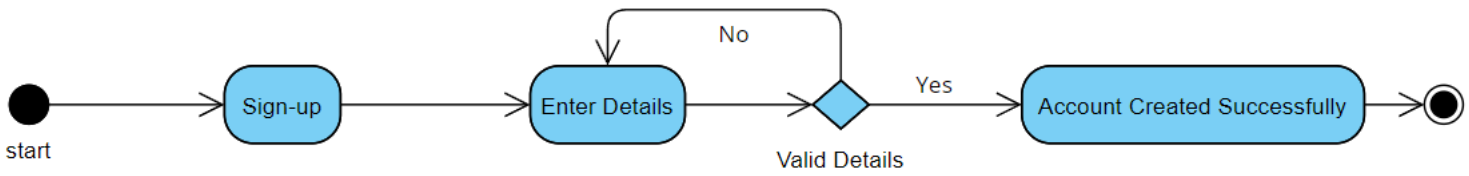
## 5.3 White Box Testing

White testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other name of glass box testing is clear testing, open box testing, logic drives testing or path driven testing or structural testing

### 5.3.1 Cyclomatic Complexity

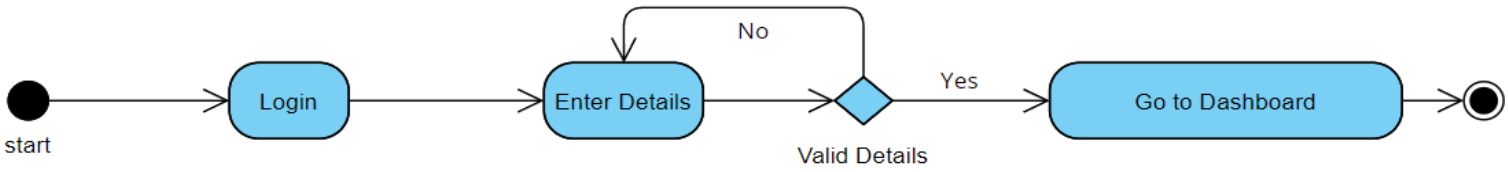
Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measure the number the linearly-independent paths through a program module. Lower the program's Cyclomatic complexity, lower the risk to modify and easier to understand

#### 5.3.1.1 Sign-up



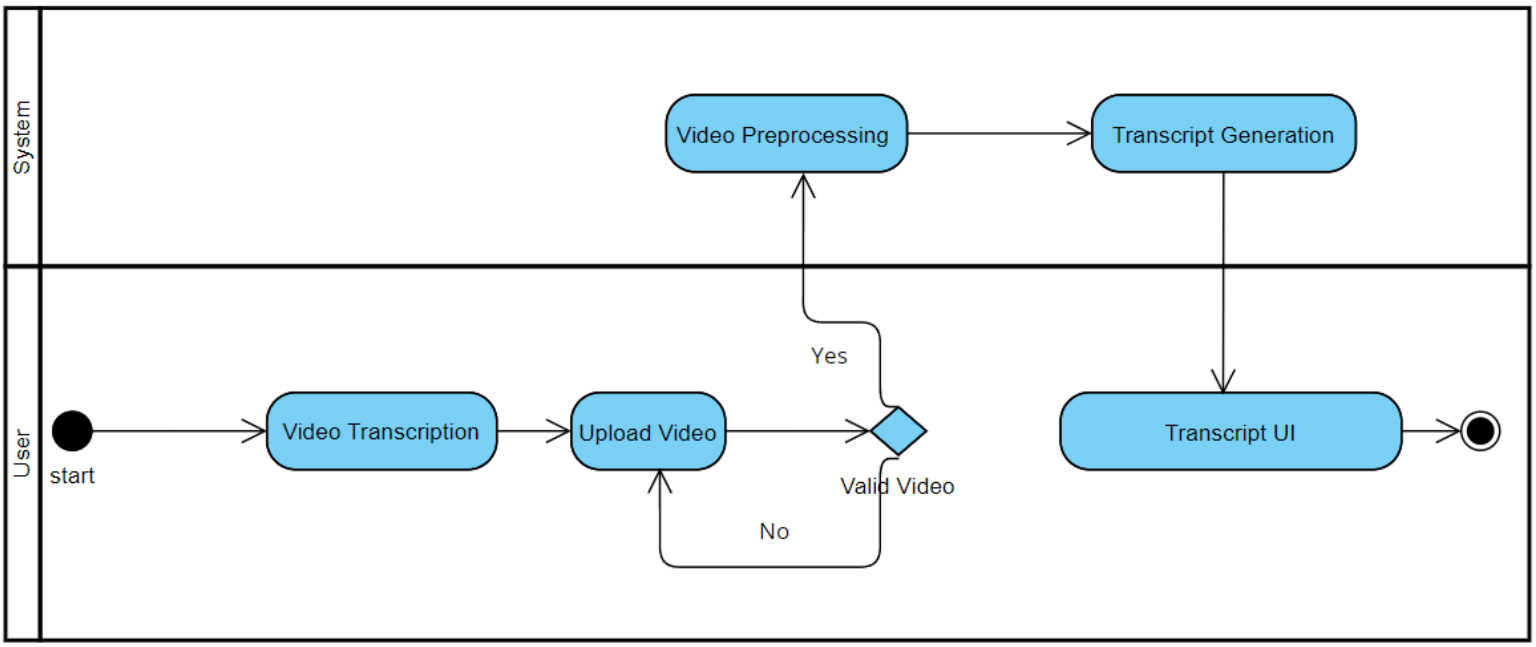
| **Cyclomatic Complexity** | |
|---|---|
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 6 |
| Cyclomatic Complexity (M) | **6 - 6 + 2 = 2** |

#### 5.3.1.2 Login



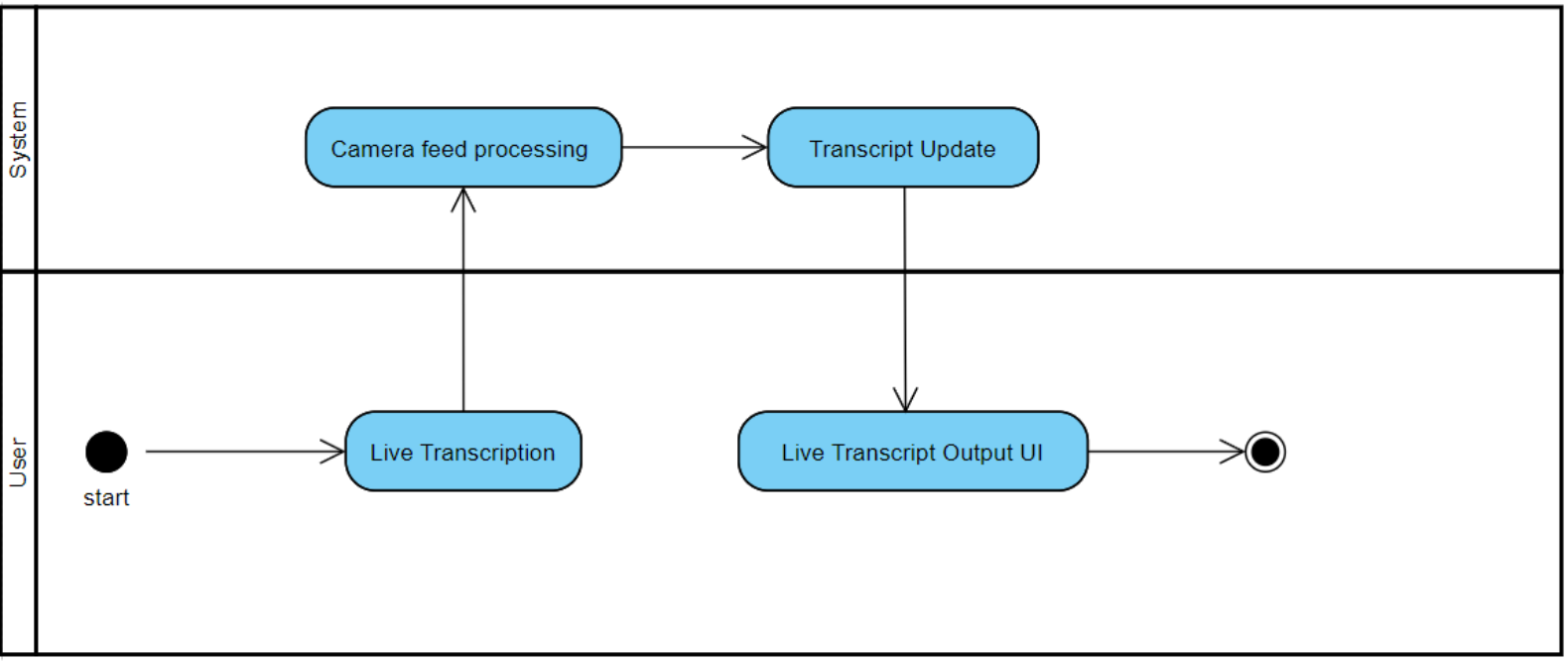

| **Cyclomatic Complexity** | |
|---|---|
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 6 |
| Cyclomatic Complexity (M) | **6 - 6 + 2 = 2** |

#### 5.3.1.3 Video Transcription




| **Cyclomatic Complexity** | |
|---|---|
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 8 |
| Number of Transitions (E) | 8 |
| Cyclomatic Complexity (M) | **8 - 8 + 2 = 2** |

#### 5.3.1.4 Real Time Transcription




| **Cyclomatic Complexity** | |
|---|---|
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 5 |
| Cyclomatic Complexity (M) | **5 - 6 + 2 = 1** |

## 5.4 Performance Testing

Performance Testing is a detailed assessment of the software program's execution speed, reaction to user inputs, ability to provide the expected results, the manner in which it optimizes the resources, and its capacity to manage growing loads without a significant decline in performance. While functional testing is aimed at detecting errors in the software's features and capabilities, the purpose of performance testing is to identify performance issues that may affect the usability of the software and its ability to perform optimally under regular usage conditions. undefined

**Speed:** This aspect evaluates the ability of the software application to respond with speed to the actions of the users or system events. It ensures that the users do not have to wait for a long time or encounter any form of delay when using the software.

**Scalability:** This also measures how well the software can handle the load of increasing number of users or transactions and still perform optimally. It assists in defining the maximum number of users that a given software can accommodate before its efficiency starts to decline.

**Stability:** This dimension measures the ability of the software to perform in a steady manner when subjected to different loads and usage patterns. It aims at establishing any vulnerability or irregularity that may arise when the software is under different loads or usage.

## 5.5 Stress Testing

Stress testing is a specific type of software testing aimed at determining the stability of a software system in conditions that it was not designed to handle, or in other words, testing the software beyond its normal limit. Still, it is important to note that stress testing is most important for applications that are vital or heavily used but can be applied to any software system.

Stress testing is particularly focused on the examination of the system stability, its capability to work in under adverse conditions, such as high loads, resource depletion, and other kinds of failures. While testing is usually concerned with ensuring that a piece of software behaves as required under normal conditions, stress testing is more interested in finding the system's flaws or the ways it malfunctions when stressed to its limits.

Key objectives of stress testing include:

**Availability**: Determining the system's accessibility and its ability to perform during peak usage or when there are competing demands on resources, so that users can access and use the system.

**Robustness**: Determining the software's capability to remain stable and performant when under pressure or in a hazardous environment and not freeze up or suffer through catastrophic malfunctions.

**Error Handling**: Verifying the capacity of the system to alert the user or administrator and prevent or correct the errors or exceptions that may occur during periods of high stress on the application, while also preventing the loss or corruption of data.

## 5.6 System Testing

System Testing is a broad type of testing that checks the total effective working and compatibility of a software product as a component of the entire computer based system. This differs from unit testing or integration testing that are done on a component or a subsystem level, system testing aims at testing the software within its complete environment, including the black-box testing.

System testing is carried out with one principal goal, and that is to validate that the software under test complies with the requirements and functions in the intended environment. Due to the nature of contact of many software systems with other software components, hardware devices, or other systems, et cetera, ensures that tested system integrates well and works in harmony with the other systems within the broader system architecture.

Key aspects of system testing include:

**Completeness:** Ensuring that the whole of the software product and any component, element or part of it performs its specified function and is compliant to the defined system requirement.

**Integration:** Verification of various forms of software conjunction and information exchange at the interfaces between the modules, parts or sub-systems.

**Interfacing:** Verification of the interfaces and communication interfaces used by the software in its interaction with other systems, databases or any hardware interfaces.

## 5.7 Regression Testing

Regression testing can be described as an important quality assurance engineering method which checks if recent changes or modifications to a software code base result in the creation of a new undesirable effect or if it has affected other desirable components or aspects of the software in a negative manner. It entails rerunning a selected set of a previously run test cases to check on the stability of a software after integrating change code into it.

The main goal of regression testing is to minimize the probability of encountering regression defects – they can occur because of code entries or deletions, bugs, or other changes made to the system. Thus, regression testing enables to preserve the stability, reliability, and the essential similarity of the software product in the consequent releases or iterations while retesting the influenced portions of the software.

Key principles of regression testing include:

**Verification:** Ensuring that new code changes or update do not impact the capability of the previous functionalities of the software or avert any kind of hindrance, loss in overall performance.

**Risk Mitigation:** Assessing and managing possible risks which may occur when making changes to the code; the goal is to maintain code purity and usability over the course of the software production.

**Efficiency:** Improving the efficiency of execution of regression tests highlighting the important test cases, utilizing the automated tools, and avoiding repetition to gain faster results and more effective release processes.

# CHAPTER 6

# Tools and Techniques

This chapter provides comprehensive details of the tools, languages, applications, and libraries employed in developing a real-time sign language transcription system. These elements are crucial in structuring and designing the system as they enhance the project's remodeling and establish a common transcription process.

**Languages Used in Development:**

The primary languages utilized in the development of the system are:

**Python**

Python serves as the main programming language for the dashboard, backend, and frontend development. The choice of Python is driven by its readability and ease of modification, allowing developers to avoid the intricacies of other languages. Additionally, Python's versatility enables the integration of code written in languages more suited for specific tasks, such as web development or machine learning libraries. Key areas where Python is employed include:

**Backend Development**: Python's robust frameworks and libraries support server-side logic, database interactions, and API development.

**Data Processing and Manipulation**: Python's rich ecosystem of data processing libraries, such as Pandas and NumPy, facilitates efficient data handling and manipulation.

**Frontend Development**: While Python is not traditionally used for frontend development, tools like Flask enable the creation of dynamic web interfaces.

**HTML/CSS**

HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are essential for developing the layout and visual design of the web interface. They ensure that the graphical user interface is smooth, user-friendly, and responsive. Key aspects include:

**HTML**: Provides the structural framework of the web pages, defining elements such as headers, paragraphs, forms, and buttons.

**CSS**: Enhances the appearance and usability of the web interface by applying styles, such as colors, fonts, and layouts, ensuring a consistent look and feel across different devices and screen sizes.

**Applications and Tools:**

Several applications and tools facilitate the development process, assisting in coding, debugging, version control, and deployment.

**Flask**

Flask is a lightweight web framework for Python, used to develop the web application's frontend. It provides a simple yet powerful framework for creating web interfaces. Key features include:

- **Flexibility**: Flask's modular design allows developers to choose the components they need, promoting a clean and maintainable codebase.
- **Extensibility**: A wide range of extensions are available for integrating additional functionality, such as form handling, authentication, and database interaction.
- **Simplicity**: Flask's straightforward and concise syntax makes it easy for developers to create robust web applications with minimal boilerplate code.

**Git and GitHub**

Git is a version control system that helps track changes in the codebase. GitHub is a platform for hosting code repositories, facilitating collaboration and version management. Key aspects include:

- **Version Control**: Git allows developers to maintain a detailed history of code changes, enabling efficient tracking, branching, and merging of code.
- **Collaboration**: GitHub's features, such as pull requests and code reviews, support collaborative development and ensure code quality.
- **Hosting and Deployment**: GitHub provides tools for continuous integration and deployment, automating the process of building, testing, and deploying applications.

**Visual Studio Code (VS Code)**

VS Code is a source code editor used for writing and debugging code. It supports a wide range of extensions that enhance coding efficiency and productivity. Key features include:

- **IntelliSense**: Advanced code completion and syntax highlighting help developers write code more efficiently and accurately.
- **Debugging**: Integrated debugging tools allow developers to set breakpoints, inspect variables, and step through code.

- **Extensions**: A vast library of extensions adds support for various languages, frameworks, and tools, enhancing the development experience.

**Google Colab**

Google Colab provides powerful computing resources and an interactive environment for training the convolutional neural network (CNN) used in the real-time sign language transcription project. Its seamless integration with Google Drive and pre-installed libraries like TensorFlow and OpenCV streamline data processing, model development, and collaboration. Key features include:

- **Cloud-Based**: Colab offers access to high-performance GPUs and TPUs, enabling efficient training of complex models without requiring local hardware.
- **Collaboration**: Colab's notebook interface allows multiple users to collaborate in real-time, facilitating knowledge sharing and teamwork.
- **Pre-Installed Libraries**: With pre-installed machine learning libraries, such as TensorFlow and Keras, developers can quickly prototype and train models.

**Libraries and Extensions:**

Various libraries and extensions are incorporated to support web development, data processing, and security.

**Mediapipe**

Mediapipe is an open-source framework developed by Google for building multimodal, cross-platform applied machine learning pipelines. It provides customizable, efficient solutions for various computer vision tasks, such as face detection, hand tracking, and pose estimation, by leveraging pre-built components called calculators.

Mediapipe is widely used for creating real-time, high-performance applications in domains like augmented reality, gesture recognition, and human-computer interaction.

**Flask-WTF**

Flask-WTF is an extension that integrates Flask and WTForms, simplifying the use and validation of forms in Flask applications. Key features include:

- **Form Handling**: Flask-WTF provides an easy way to create and manage web forms, reducing the complexity of form validation and processing.
- **CSRF Protection**: Built-in CSRF (Cross-Site Request Forgery) protection enhances the security of web forms by preventing unauthorized actions.
- **Seamless Integration**: The extension seamlessly integrates with Flask, allowing developers to quickly add form functionality to their applications.

**WTForms**

WTForms is a flexible form validation and rendering library for Python web applications. It is used for designing and validating web forms. Key features include:

- **Form Rendering**: WTForms provides a variety of field types and validation options, enabling developers to create complex forms with ease.
- **Custom Validators**: Developers can create custom validators to enforce specific business rules and requirements.
- **Extensibility**: WTForms is highly extensible, allowing developers to integrate it with various web frameworks and libraries.

**Flask-MySQLdb**

Flask-MySQLdb is an extension for Flask to work with MySQL databases. It provides rudimentary instructions on how to connect to a MySQL server and perform database queries. Key features include:

- **Database Connectivity**: Simplifies the process of connecting Flask applications to MySQL databases, enabling efficient data storage and retrieval.
- **Query Execution**: Provides tools for executing SQL queries, managing transactions, and handling database errors.
- **Integration**: Seamlessly integrates with Flask, allowing developers to incorporate database functionality into their applications with minimal effort.

**Bcrypt**

Bcrypt is an efficient password hashing program used in Python to add and verify password digests, enhancing the security of the application. Key features include:

- **Strong Security**: Bcrypt uses a computationally intensive hashing algorithm, making it resistant to brute-force attacks.
- **Salting**: Automatically generates a unique salt for each password, further increasing security by preventing rainbow table attacks.
- **Ease of Use**: Simple API for hashing and verifying passwords, making it easy to implement secure authentication in web applications.

**OpenCV**

OpenCV (Open Source Computer Vision Library) is used for image analysis and processing. It acts as the primary input/output unit and handles video feed and pre-processing frames for the neural network. Key features include:

- **Image Processing**: Provides a wide range of tools for image manipulation, including filtering, transformation, and feature extraction.
- **Video Analysis**: Supports video capture and processing, enabling real-time analysis of video streams.
- **Machine Learning**: Includes machine learning algorithms and tools for training and deploying models for computer vision tasks.

# Chapter 7

# Summary and Conclusion

# Project Overview

## Introduction

The aim of this project is to develop a software application that can transcribe conventional sign language in real time. This technology aims to significantly improve communication for the deaf and hard-of-hearing community. By leveraging the power of machine learning and web technologies, the project facilitates the live capture and processing of sign language gestures via webcam, converting them into text in real time. This tool is especially useful in overcoming communication barriers in various social and occupational settings, thereby promoting inclusivity and accessibility.

## Objectives

1. **Enhance Communication**: Provide a reliable tool for real-time transcription of sign language, thereby bridging the communication gap between sign language users and non-users.
2. **User-Friendly Interface**: Design an intuitive and accessible user interface that caters to individuals with varying levels of technical expertise.
3. **Real-Time Processing**: Ensure the system can handle real-time video input and provide immediate text transcriptions.
4. **Security**: Implement robust security measures to protect user data and credentials.
5. **Community Engagement**: Incorporate feedback from the deaf and hard-of-hearing community to refine and improve the system.

## System Architecture

The system is built using a combination of frontend and backend technologies, integrating various tools and libraries to achieve its objectives.

**Frontend Development**

The frontend is built using Flask, a lightweight web framework for Python, providing an intuitive and user-friendly interface. Key features of the frontend include:

- **Live Video Input**: Users can provide real-time video input through a webcam. The system processes this input to recognize and transcribe sign language gestures.
- **Real-Time Transcription**: The system displays text transcriptions of sign language gestures as they are performed, providing immediate feedback to the user.

The user interface is designed to be simple and accessible, ensuring ease of use for individuals with varying levels of technical expertise. The integration of HTML, CSS, and JavaScript enhances the interactivity and visual appeal of the application. Key elements of the frontend development include:

- **HTML**: Provides the structural framework of the web pages, defining elements such as headers, paragraphs, forms, and buttons.
- **CSS**: Enhances the appearance and usability of the web interface by applying styles, such as colors, fonts, and layouts, ensuring a consistent look and feel across different devices and screen sizes.

**Backend Development**

The backend of the system is robust, leveraging a comprehensive dataset and advanced machine learning models to accurately interpret sign language gestures. Key components of the backend include:

- **Dataset**: The system uses a large, annotated dataset of sign language gestures, covering a wide range of signs and phrases. This dataset is crucial for training the machine learning model to recognize various gestures accurately.

- **Machine Learning Model**: A convolutional neural network (CNN) is trained on the dataset to recognize and interpret sign language gestures. The CNN architecture is chosen for its effectiveness in image and video recognition tasks.
- **API**: A RESTful API is developed using Flask-RESTful to handle the transcription process. The API processes video frames, utilizes the trained CNN model to predict the corresponding text, and returns the transcriptions to the frontend in real time.

The backend also incorporates Flask-MySQLdb for database interactions, storing user data and transcriptions securely. Bcrypt is used for password hashing, ensuring the security and integrity of user credentials.

- **Flask-MySQLdb**: This extension provides an easy way to interact with MySQL databases from Flask applications. It supports CRUD operations (Create, Read, Update, Delete) and helps in managing user data and transcriptions.
- **Bcrypt**: Used for securely hashing passwords, Bcrypt ensures that user credentials are stored securely, protecting against potential security breaches.

**Google Colab**

Google Colab plays a crucial role in the development of the machine learning model. It provides several key benefits, including:

- **Powerful Computing Resources**: Google Colab offers free access to GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), significantly speeding up the training process of the CNN model. These resources are essential for handling the computational demands of training deep learning models.
- **Integration with Jupyter Notebooks**: Google Colab uses Jupyter Notebooks, which facilitate interactive coding, data visualization, and documentation. This feature allows for:
    - **Interactive Development**: Code can be run in segments, making it easier to debug and refine the machine learning model.

- **Data Visualization**: Visual tools and libraries can be used within the notebook to visualize data and model performance, aiding in better understanding and optimization of the model.
- **Documentation**: Notes and explanations can be included directly alongside the code, making the development process more transparent and the notebook a comprehensive documentation of the project's progression.
- **Collaboration**: Google Colab supports collaboration, allowing multiple users to work on the same notebook simultaneously. This feature is beneficial for team-based projects, facilitating shared development and peer review.

## Implementation and Testing

The implementation of the system focuses on achieving high accuracy and performance. Extensive testing is conducted to ensure the reliability and effectiveness of the system:

### Performance Testing

The system's performance is tested to ensure it can handle real-time video input and provide timely transcriptions. This involves evaluating the system's latency and responsiveness under different conditions:

- **Latency Measurement**: Measuring the time taken from video frame capture to text transcription, ensuring minimal delay.
- **Scalability Testing**: Assessing the system's ability to handle multiple simultaneous users and varying video input quality.
- **Resource Utilization**: Monitoring CPU, GPU, and memory usage to ensure efficient use of computing resources.

**User Testing**

Feedback from users, including members of the deaf and hard-of-hearing community, is gathered to refine the system. User testing helps identify usability issues and areas for improvement, ensuring that the system meets the needs of its target audience:

- **Usability Surveys**: Conducting surveys to gather feedback on the user interface, ease of use, and overall experience.
- **Focus Groups**: Organizing focus groups with sign language users to observe their interactions with the system and gather in-depth feedback.
- **Accessibility Testing**: Ensuring that the system is accessible to users with different levels of technical expertise and varying physical abilities.

**Security Considerations**

Security is a critical aspect of the system, particularly in protecting user data and ensuring the integrity of the transcription process. Key security measures include:

- **Authentication and Authorization**: Implementing robust authentication mechanisms to verify user identity and authorization to access certain features.
- **Data Encryption**: Encrypting sensitive data, such as user credentials and personal information, both in transit and at rest.
- **Regular Audits**: Conducting regular security audits and vulnerability assessments to identify and mitigate potential risks.

**Conclusion**

The real-time sign language transcription system developed in this project demonstrates the potential of combining machine learning and web technologies to enhance communication for the deaf and hard-of-hearing community. By integrating Python, Flask, and a robust backend API, the system provides an effective solution for translating sign language into text, promoting inclusivity and accessibility in various environments. The use of Google Colab for model training ensures efficient development and optimization of the machine learning model, resulting

in a reliable and user-friendly application. This project highlights the importance of technological innovation in addressing communication barriers and fostering a more inclusive society.

## Future Work

While the current system provides a solid foundation, there are several areas for future enhancement:

1. **Extended Vocabulary**: Expanding the dataset to include more signs and phrases, covering a broader range of vocabulary.
2. **Multilingual Support**: Adding support for multiple languages, allowing the system to transcribe sign language into different spoken and written languages.
3. **Mobile Application**: Developing a mobile version of the application to increase accessibility and convenience for users.
4. **Gesture Recognition Accuracy**: Continuously improving the machine learning model to increase the accuracy of gesture recognition, particularly for complex and nuanced signs.
5. **Integration with Assistive Technologies**: Integrating the system with other assistive technologies, such as hearing aids and speech-to-text applications, to provide a comprehensive communication solution.

## Impact

The development of this real-time sign language transcription system has the potential to significantly impact the lives of deaf and hard-of-hearing individuals. By providing a reliable and accessible tool for communication, the system can:

- **Improve Social Interactions**: Enhance everyday interactions by enabling seamless communication between sign language users and non-users.
- **Facilitate Employment Opportunities**: Improve accessibility in the workplace, allowing deaf and hard-of-hearing individuals to participate more fully in professional environments.
- **Enhance Educational Experiences**: Support inclusive education by providing real-time transcription in classrooms, enabling deaf and hard-of-hearing students to follow along with lectures and discussions.
- **Promote Inclusivity**: Foster a more inclusive society by breaking down communication barriers and promoting understanding and interaction between different communities.

In conclusion, this project not only showcases the power of technology in addressing communication challenges but also emphasizes the importance of inclusivity and accessibility in today's digital age. By continuing to refine and expand the system, we can ensure that it remains a valuable tool for the deaf and hard-of-hearing community, making a meaningful difference in their daily lives.

# Chapter 8

# User Manual

## Overview:

The login authentication screen allows users to log in to the system securely. To access this screen, users must have an active username and password provided by the system administrator, along with the web login address.

### Steps to Log In:

**Open Login Webpage:** .1

Launch your preferred web browser (e.g., Google Chrome, Mozilla Firefox).

Press Enter to navigate to the login page.

**Enter Login Details:** .2

On the login page, you will see fields to enter your username and password.

Type your email in the "email" field.

Type your password in the "Password" field. (Note: Passwords are case-sensitive.)

**Click Login:** .3

After entering your login credentials, click on the "Login" button located below the login form.

Alternatively, you can press the "Enter" key on your keyboard after entering your password to submit the login form.

**Done:** .4

If the entered username and password are correct and valid, you will be successfully logged in to the system.

You will be redirected to the application's dashboard or homepage, depending on the system's configuration.

Now you can access the various features and functionalities available within the application according to your user role and permissions.

### Steps to Sign Up:

**Access Sign-Up Page:** .1

Open your preferred web browser.

Enter the web address provided for signing up.

Press Enter to navigate to the sign-up page.

**Provide User Information:** .2

On the sign-up page, you will find fields to input your personal information.

Enter your desired username in the "Username" field. (Note: Usernames may be subject to availability and character restrictions set by the system.)

Choose a strong and secure password for your account. Enter the chosen password in the "Password" field. (Note: Passwords are case-sensitive and should contain a mix of letters, numbers, and special characters for enhanced security.)

Confirm your password by retyping it in the "Confirm Password" field.

**Submit Sign-Up Form:** .3

Once you have provided all necessary information, proceed to submit the sign-up form.

Click on the "Sign Up" or "Create Account" button located below the sign-up form.

**Verification and Confirmation:** .4

After submitting the sign-up form, the system will process your request.

If the provided information meets the system's requirements and no errors are encountered, your account will be successfully created.

**Accessing the System:** .5

Once your account is successfully created and verified (if applicable), you can proceed to log in to the system using your newly created username and password.

Refer to the "Login Functionality" section of this user manual for instructions on how to log in.

**Active Email and Password:**

- Email: (70110706@student.uol.edu.pk)
- Password: (12345678)

**Troubleshooting**

If you encounter any issues logging in, ensure that you have entered the correct username and password. Passwords are case-sensitive, so ensure that Caps Lock is not enabled and that you are typing your password correctly.

If you have forgotten your password or are unable to log in, contact your system administrator for assistance. They can reset your password or provide further guidance on accessing the system.

By following these steps, users can log in securely to the application and access its various features and functionalities.