Software Specification for *Sign Language Detection Application*

Created by:

Mursil Hassan 22i2132

Ishaq ahmed 22k4066

Table of Contents

Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to describe the requirements for the development of "Ez Mart", an online marketplace for buyers and sellers to interact with each other selling and purchasing products online. This document will define the features and functionality of the software system, its constraints, and its performance requirements.

1.2 Document Conventions

This document follows standard conventions for writing SRS documents.

1.3 Intended Audience and Reading Suggestions

This document is intended for software developers, project managers, and stakeholders involved in the development of "Sign Language Gesture Detection App". Readers should have a basic understanding of software development and system design.

1.4 Product Scope

The scope of this project is to develop a sign language detection application, for handicapped users to communicate, to each other, or for people who don't know sign language.

This at the moment has 2 panels, the login page, and the main detection application, it is currently limited to 7 signs, however with enough computing power we can train the same model on multiple signs. The second panel is the main panel, where the users can open the camera and detect signs.

In the future this can be further implemented on video calls.

1.5 References

No external references were used in the creation of this document.

Overall Description

2.1 Product Perspective

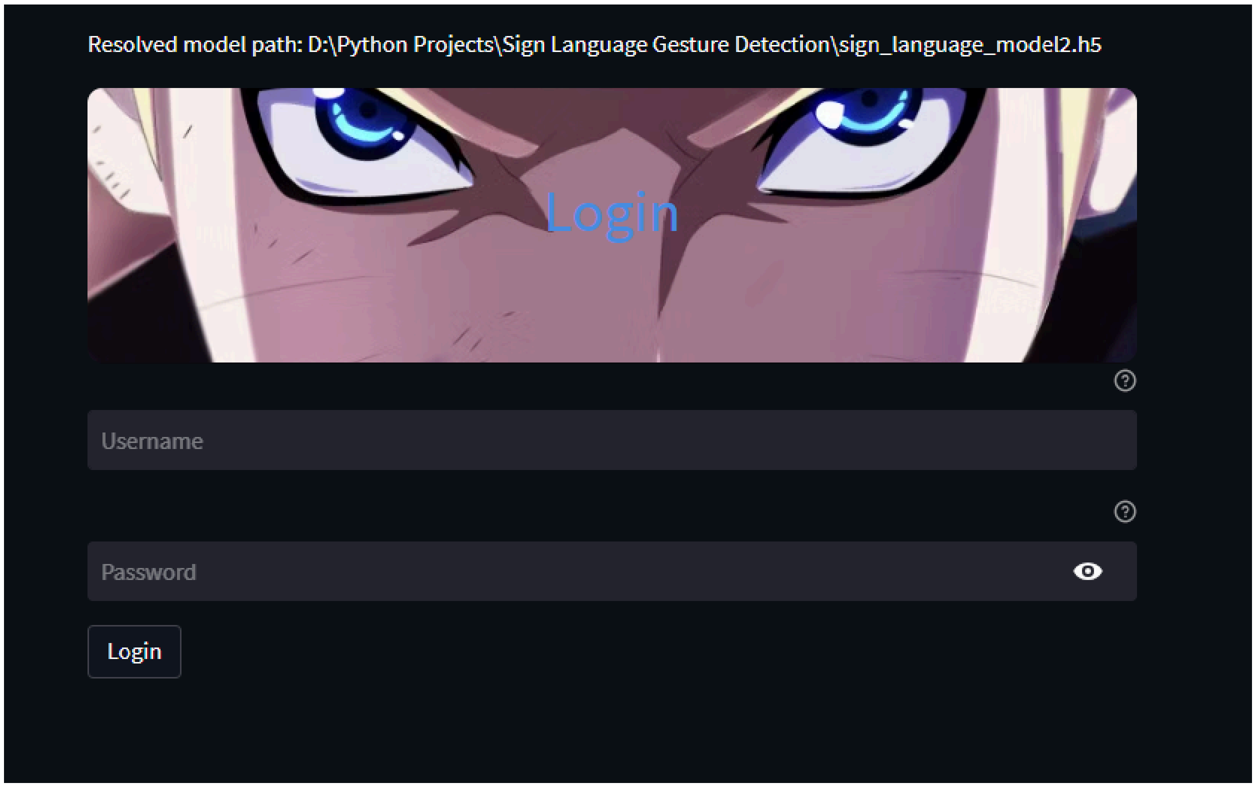
The Real-time Sign Language Detection application leverages advanced machine learning models to accurately identify and classify hand signs in real-time. Utilizing TensorFlow for model predictions and MediaPipe for hand landmark detection, this solution offers a seamless and interactive experience. Streamlit's user-friendly interface enhances accessibility, ensuring ease of use for both developers and end-users. The application serves as a powerful tool for bridging communication gaps, providing immediate, visual translation of sign language gestures. It stands out for its integration of cutting-edge technology and practical usability, promising to revolutionize interactions for the deaf and hard-of-hearing communities.

2.2 Product Functions

The following are the functional requirements of the "Sign Language Gesture Detection" software:

2.2.1 Login Interface

The login interface should allow the user to select their user type, whether admin, seller, or customer. Upon successful login, the user should be directed to their respective interface.

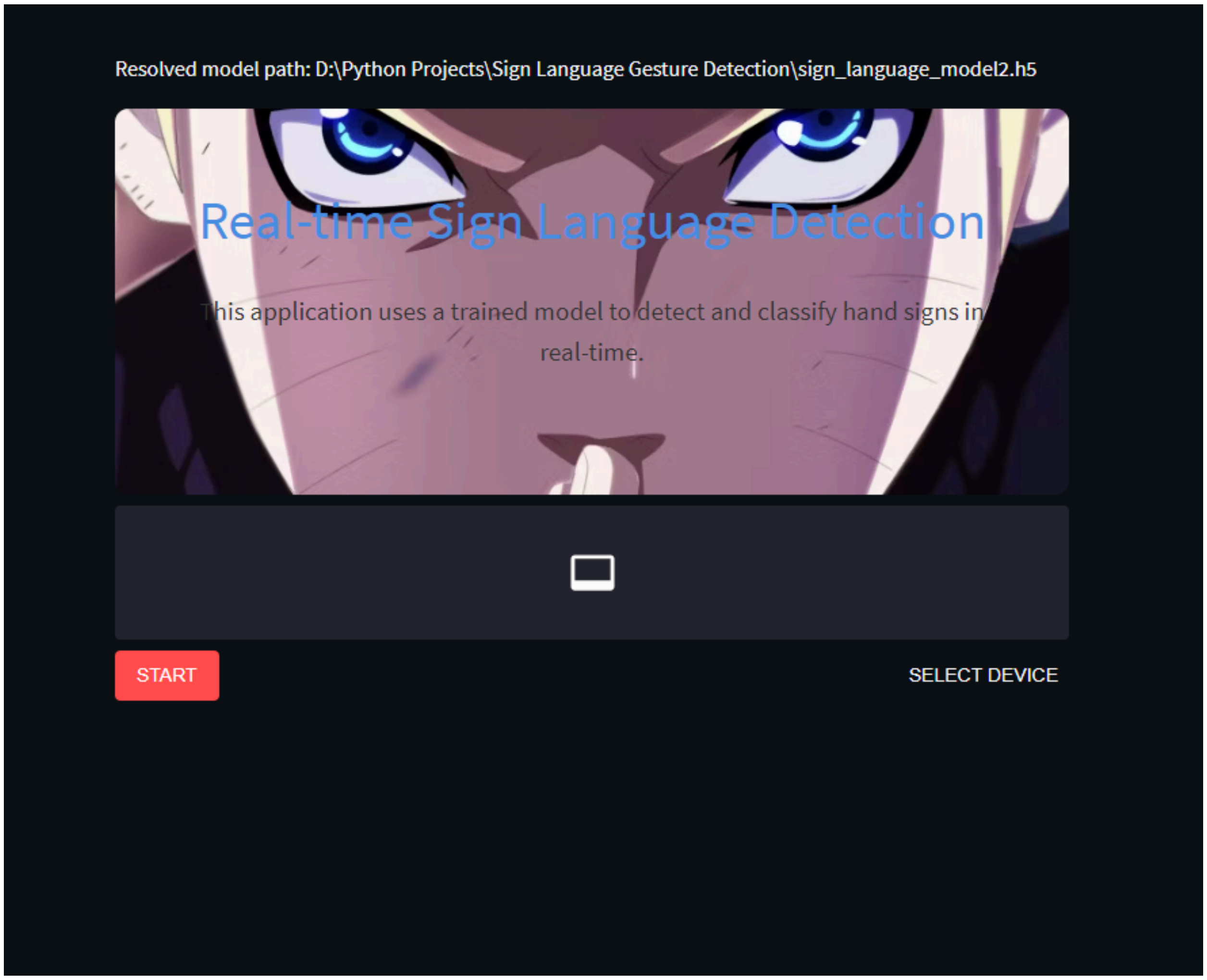


```
Resolved model path: D:\Python Projects\Sign Language Gesture Detection\sign_language_model2.h5
```

2.2.2 Main Interface

The admin interface should provide the following functionalities:

- Open camera: The camera opens
- Detect Signs: Creates bounding boxes with mediapipe
- Model loading: Loads the model
- Detection: Detects the signs live



Resolved model path: D:\Python Projects\Sign Language Gesture Detection\sign_language_model2.h5

Real-time Sign Language Detection

This application uses a trained model to detect and classify hand signs in real-time.

2.3 Files and Characteristics

The Product has 4.py files, including model building, data capture, data augmentation, and live inferencing. Another class is the vidtransformer class that detects hand signs, by making the bounding box.

2.4 Operating Environment

Sign Language Gesture Detection here on referred to as SLGD, is based on python.

- Python
- TensorFlow
- MediaPipe
- OpenCv
- Streamlit

2.5 Design and Implementation Constraints

- Hardware Limitations: The application requires a high-definition camera and a system with sufficient processing power to handle real-time video streaming and inference. Older or less powerful hardware may struggle to deliver smooth and accurate performance.
- Internet Connectivity: For optimal performance, especially when deployed on the web, the application relies on a stable and high-speed internet connection. Poor connectivity can lead to latency issues and degraded user experience.
- Model Size and Performance: The trained TensorFlow model used for sign language detection needs to be optimized for both accuracy and speed. Large models may offer better accuracy but can slow down the real-time processing. Balancing model size with performance is crucial.
- Lighting Conditions: The accuracy of hand detection and gesture recognition is highly dependent on the lighting conditions. Poor or inconsistent lighting can affect the

model's ability to correctly identify hand landmarks and gestures, leading to reduced accuracy.

- Background Complexity: The application performs best in environments with simple and uncluttered backgrounds. Complex or dynamic backgrounds can interfere with the detection of hand landmarks, thereby impacting the overall reliability of the sign language recognition.

2.6 User Documentation

The SLGD software system must have comprehensive user documentation that is easily accessible and understandable to all user classes. The user documentation should provide step-by-step instructions on how to use the software system, including how to log in, and use the application.

2.7 Assumptions and Dependencies

The following assumptions and dependencies are made during the development of the Ez Mart software system:

- User Knowledge: It is assumed that users have a basic understanding of how to operate a web application and access webcam functionality.
- Environment Setup: Users will operate the application in a well-lit environment with minimal background distractions to ensure accurate gesture recognition.
- Device Compatibility: Users will access the application on devices equipped with a functional high-definition webcam and sufficient processing power to handle real-time video streaming and model inference.
- Stable Internet Connection: It is assumed that users have access to a stable and high-speed internet connection for seamless real-time interaction, especially if the application is web-based.
- Model Training Data: The underlying machine learning model is trained on a sufficiently large and diverse dataset of sign language gestures to ensure high accuracy and generalization.

- **Dependencies**:
- Hardware Requirements: The application is dependent on the availability of high-definition webcams and hardware capable of handling intensive computations for real-time processing.
- Software Libraries: The application relies on several software libraries and frameworks, including TensorFlow for machine learning, MediaPipe for hand landmark detection, OpenCV for image processing, and Streamlit for building the user interface.
- Third-Party Services: For web deployment, the application may depend on third-party cloud services for hosting, which should provide sufficient computational resources and uptime guarantees.
- Browser Compatibility: The application's performance and user experience depend on the compatibility with modern web browsers that support WebRTC and other necessary web technologies.
- Data Privacy and Security: Ensuring data privacy and security is critical, especially for applications involving video streaming. Compliance with relevant data protection regulations and best practices is a key dependency.

External Interface Requirements

3.1 User Interfaces

Login Screen: Simple interface with styled input fields for username and password, and a login button.

Main Interface:

Title and Description: Displays the application’s purpose.

Video Feed: Shows the real-time webcam feed.

Detection Output: Overlays detected hand landmarks and predicted gestures on the video.

3.2 Hardware Interfaces

Webcam: Requires a high-definition webcam for capturing video.

Processing Unit: Needs a device with sufficient processing power for real-time video processing and inference.

3.3 Software Interfaces

TensorFlow: For loading and running the sign language recognition model.

MediaPipe: For detecting hand landmarks in the video.

OpenCV: For image processing tasks like resizing and color conversion.

Streamlit: For creating the web-based user interface.

Streamlit WebRTC: For handling real-time video streaming and transformations.

3.4 Communication Interfaces

Internet Connection: Stable, high-speed connection for web-based deployment.

WebRTC Protocol: Utilized for real-time video streaming between the client’s webcam and the application.

System Features

The following section describes the features and functionality of the Ez Mart system:

4.1.1 User Login

Users can enter their credentials and log in to the system.

4.1.2 User Registration(future)

Users can register by providing basic information such as name, email address, and password.

4.1.3 User Profile Management

Users can manage their profile information, including name, email address, and password.

4.2 Real-time Gesture Detection

The application will provide functionality to detect and classify sign language gestures in real-time. The following features will be supported:

4.2.1 Video Feed Processing

The application will capture and process the real-time video feed from the user's webcam.

4.2.2 Hand Landmark Detection

The application will detect hand landmarks using MediaPipe to identify hand positions and gestures.

4.2.3 Gesture Classification

The application will classify the detected hand gestures using a trained TensorFlow model and display the corresponding sign language interpretation.

4.3 User Interface

The application will provide an intuitive and user-friendly interface for interaction. The following features will be supported:

4.3.1 Display Video Feed

The application will display the real-time video feed along with detected hand landmarks and gesture classification overlays.

4.3.2 Gesture Interpretation

The application will display the interpreted sign language gesture as text on the video feed.

Other Nonfunctional Requirements

- **Non-Functional Requirements**
- The following section describes the non-functional requirements for the Real-time Sign Language Detection application:

  5.1 Performance Requirements

- Real-time Processing: The application must process video input and detect gestures in real-time with minimal latency to ensure a seamless user experience.
- Accuracy: The gesture detection and classification must be highly accurate, with a target accuracy rate of at least 95%.

  5.2 Usability Requirements

- User-Friendly Interface: The application must provide an intuitive and easy-to-use interface, accessible to users of all ages and technical backgrounds.
- Accessibility: The application should be accessible to users with disabilities, providing clear visual feedback and easy navigation.

  5.3 Reliability Requirements

- Uptime: The application should have a high uptime, with availability of at least 99.5%.
- Error Handling: The application must handle errors gracefully, providing meaningful error messages and allowing users to recover from errors without losing data.

  5.4 Security Requirements

- Data Privacy: The application must ensure that all user data, including personal information and video feeds, is securely stored and transmitted, complying with relevant data protection regulations.
- Authentication: The application must provide secure authentication mechanisms to protect user accounts.

  5.5 Maintainability Requirements

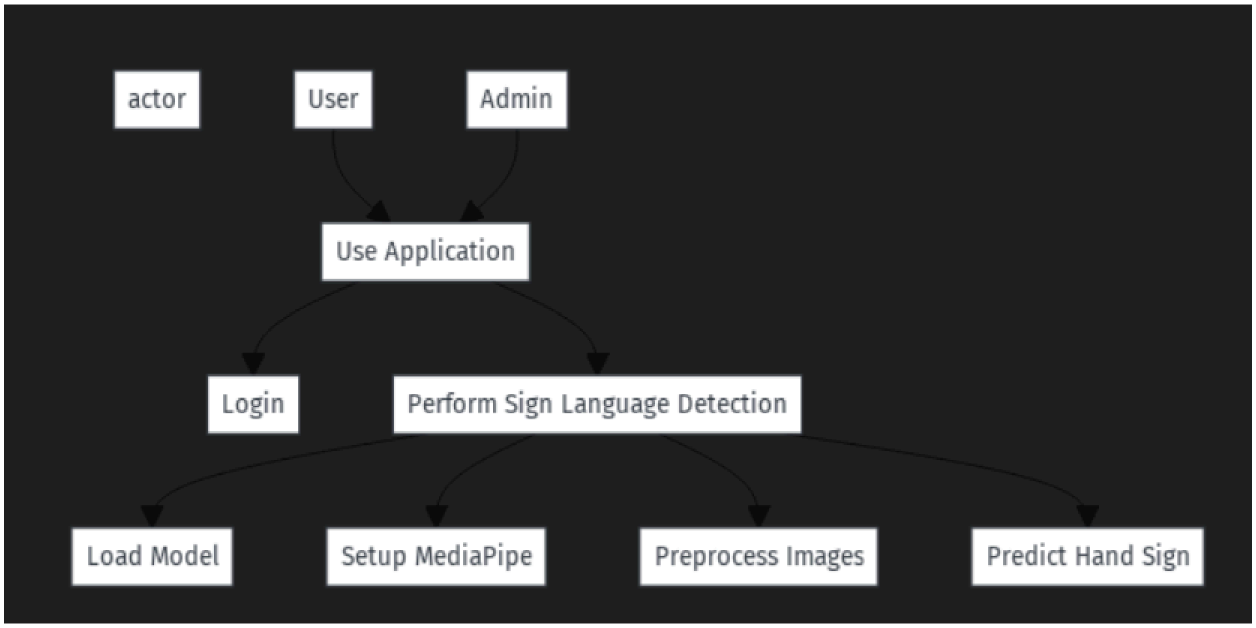
- Modular Design: The application should be designed with modular components to facilitate easy maintenance and updates.
- Documentation: Comprehensive documentation must be provided for both users and developers, including user guides and API documentation.

5.6 Portability Requirements

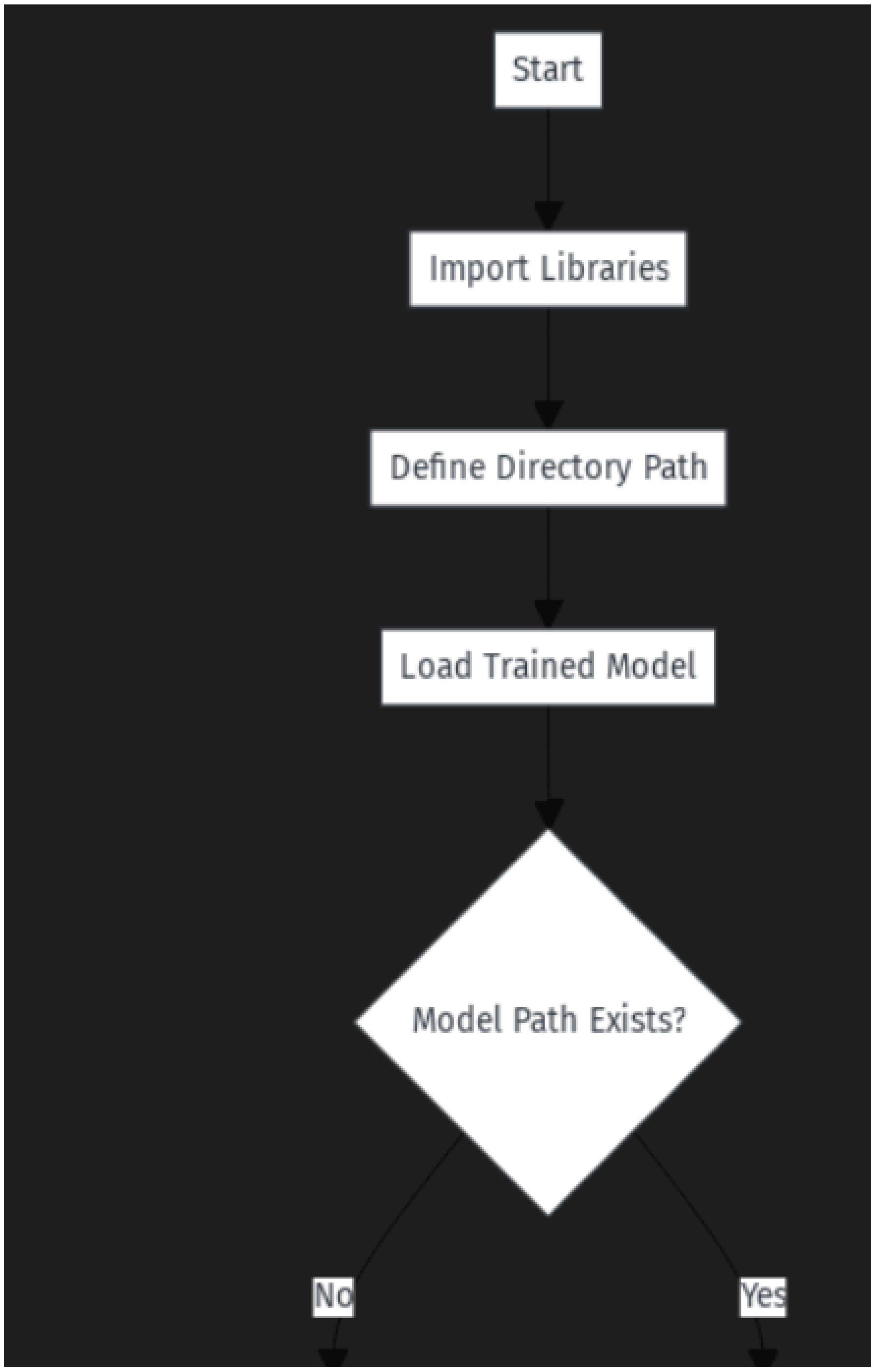
- Cross-Platform Compatibility: The application must be compatible with multiple operating systems, including Windows, macOS, and Linux.
- Browser Compatibility: The web-based application must be compatible with major web browsers, including Chrome, Firefox, Safari, and Edge.
- These non-functional requirements ensure that the Real-time Sign Language Detection application is performant, user-friendly, reliable, secure, maintainable, and portable.

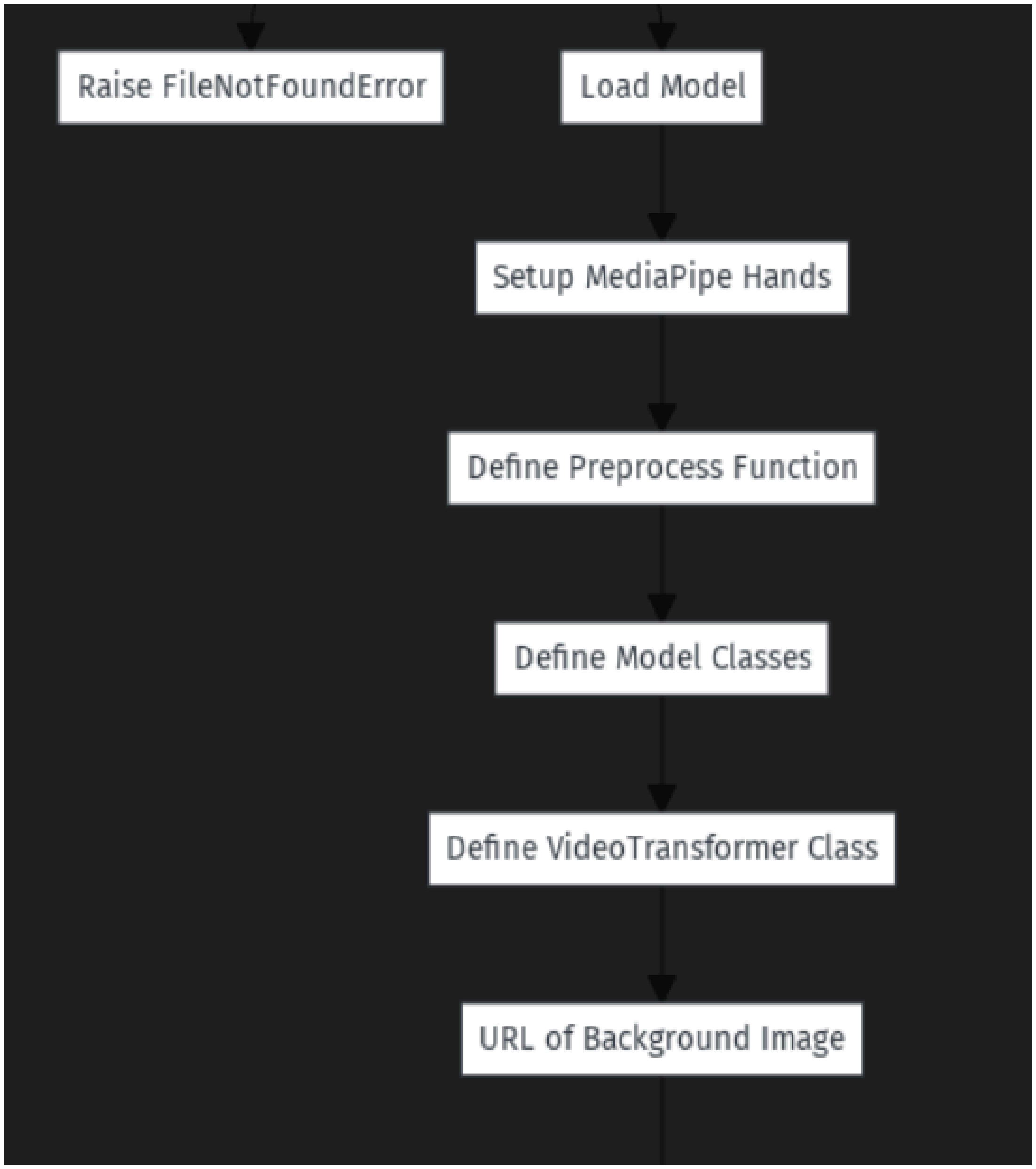
Use Case and Activity Diagram

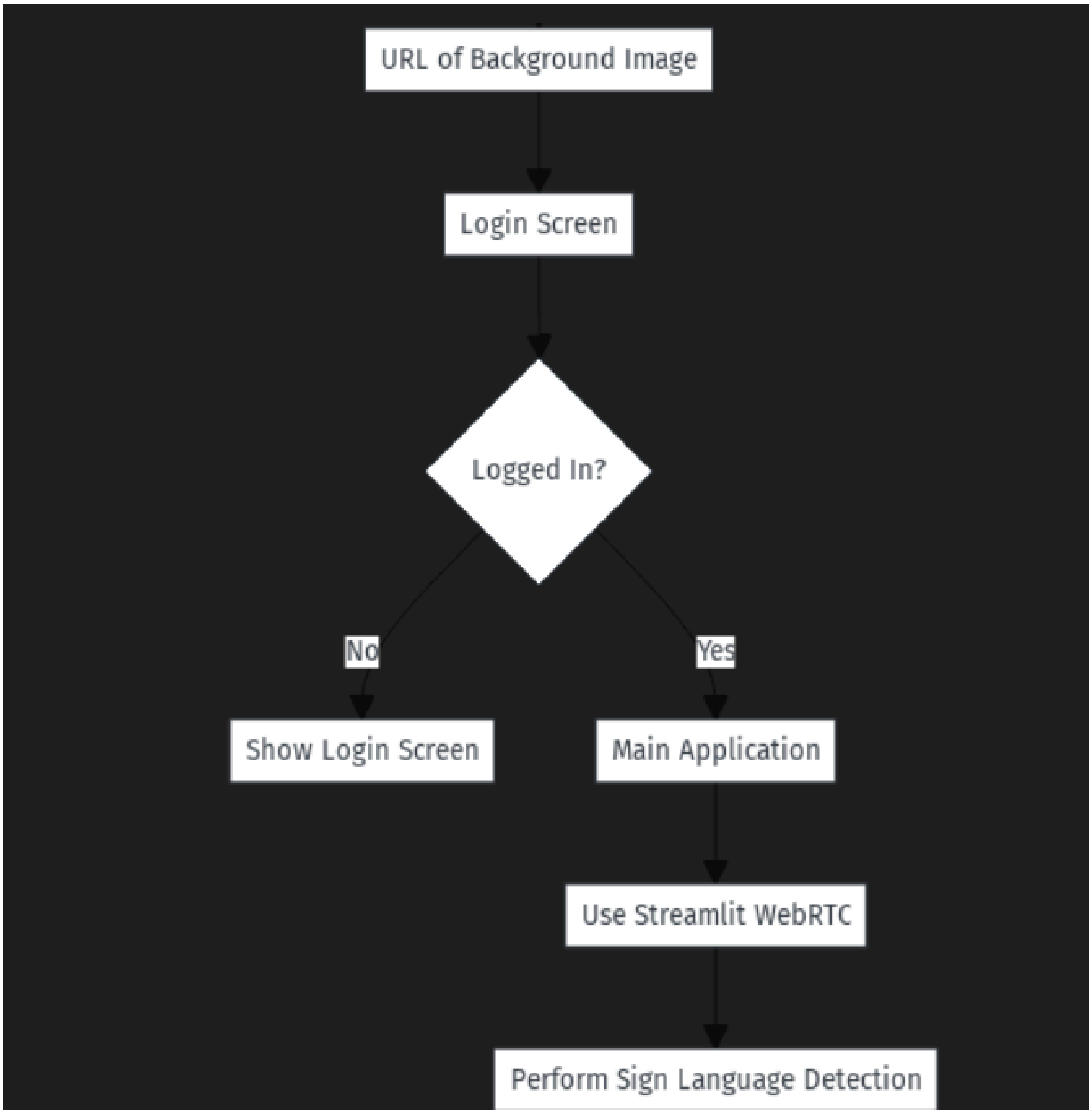
6.1 Use Case diagram



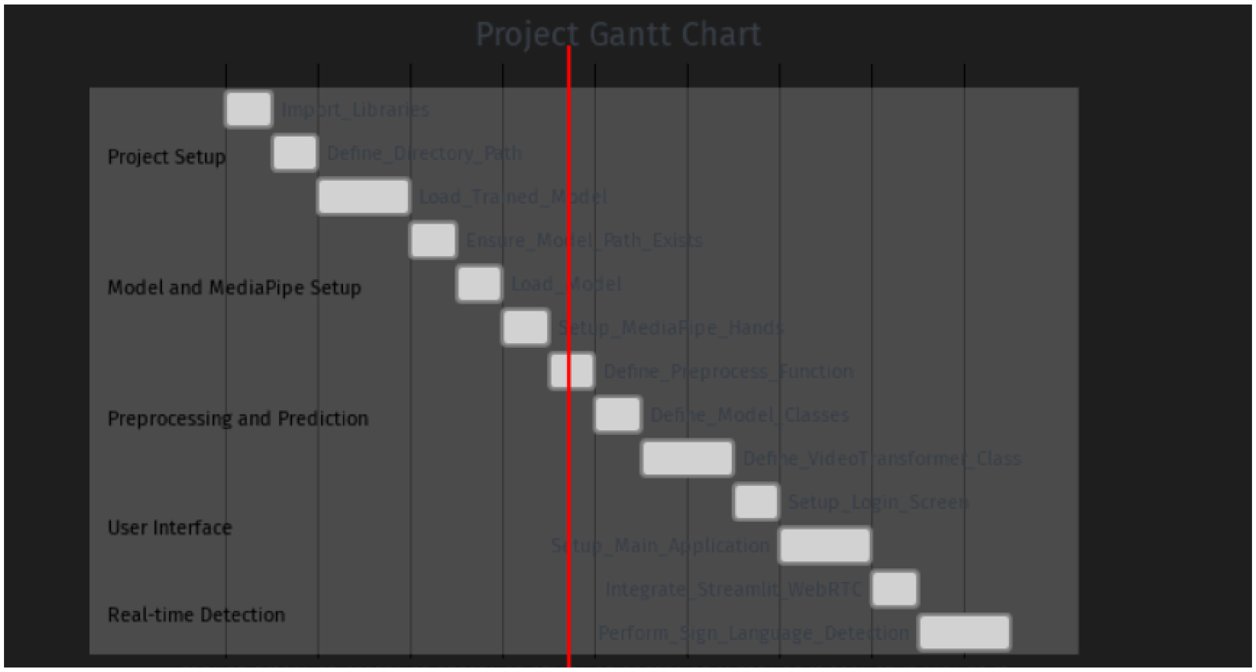
6.2 Activity diagram


Start
Import Libraries
Define Directory Path
Load Trained Model
Model Path Exists?
No
Yes


Raise FileNotFoundError
Load Model
Setup MediaPipe Hands
Define Preprocess Function
Define Model Classes
Define VideoTransformer Class
URL of Background Image


URL of Background Image
Login Screen
Logged In?
No
Yes
Show Login Screen
Main Application
Use Streamlit WebRTC
Perform Sign Language Detection

Gantt Chart



Testing

7.1 Introduction

The testing section of this SRS report outlines the testing approach and results for the login, add, remove, and delete features of the SLGD project. This section includes the test cases and their expected results, as well as the actual results obtained during testing.

7.2 Objectives

The objective of this testing section is to ensure that the login, add, remove, and delete features of the SLGD project are functioning as intended and meeting the requirements outlined in the SRS report. By conducting rigorous testing, we aim to identify any defects or issues that may impact the functionality of the system and provide recommendations for improvement.

7.3 Testing Methods

The testing methods: include accuracy and real time monitoring so far done manually, no unit tests are sufficient for this application.

Glossary:

SRS: Software Requirements Specification. It is a document that describes the functionality and features of a software system.

Admin: A user type that has access to the system's administrative functionalities, such as managing sellers and viewing transaction records.

Login Page: The initial page that prompts the user to select their user type and provide login credentials

User Interface: The visual and interactive components of the software that allow users to interact with the system.

Hardware Interface: The physical components of the system that enable communication between the software and external devices.

Software Interface: The protocols and standards used for communication between software components or with external systems.

Communication Interface: The methods and protocols used for communication between different entities in the system.

Performance Requirements: The expected performance metrics of the system, including speed, reliability, and scalability.

Safety Requirements: The measures taken to ensure the safety and security of the system and its users.

Security Requirements: The measures taken to protect the system from unauthorized access and data breaches.

Software Quality Attributes: The characteristics of the software that affect its usability, maintainability, and reliability.

Mediapipe: Library used for landmarks

Tensorflow: Library used for model building

OpenCv: Library for camera control

Streamlit: Library for UI