

SIGN LANGUAGE RECOGNITION WITH COMPUTER VISION

By

Tee Wei Heng

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2024

REPORT STATUS DECLARATION FORM

Title: Sign Language Recognition with Computer Vision

Academic Session: 202401

I TEE WEI HENG

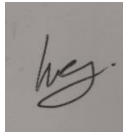
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

62, Jalan Putri 6, Taman Putrimas,

86200 Simpang Renggam, _____

Johor _____

Ng Hui Fuang

Supervisor's name

Date: 26/4/2024

Date: 26/4/2024

**FACULTY/INSTITUTE* OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26/04/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that _____ Tee Wei Heng_____ (ID No:
____20ACB02066____) has completed this final year project/ dissertation/ thesis* entitled
“Sign Language Recognition with Computer Vision” under the
supervision of _____ Dr. Ng Hui Fuang_____ (Supervisor)
Faculty/Institute* of Information and Communication Technology

I understand that University will upload softcopy of my final year project / dissertation/
thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to
UTAR community and public.

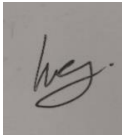
Yours truly,



(Tee Wei Heng)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**Sign Language Recognition with Computer Vision**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____  _____

Name : _____ Tee Wei Heng _____

Date : _____ 26/4/2024 _____

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my deepest gratitude to my supervisor, Dr. Ng Hui Fuang, for affording me the opportunity to embark on the development of a sign language recognition mobile application. Throughout this journey, Dr. Ng's unwavering support and exceptional guidance have been instrumental in my development.

Furthermore, I wish to extend my heartfelt appreciation to my family, who have consistently stood by my side, providing mental support. Finally, I wish to express my gratitude to my friends for their continuous support and inspiring ideas.

ABSTRACT

This project centres on crafting a web application that recognizes sign language, accessible across different devices and operating systems. Its primary objective is to convert American Sign Language (ASL) gestures into text, facilitating communication for the hearing impaired. Leveraging Computer Vision (CV), the project aims to equip computers with the ability to interpret visual cues. Addressing the inherent challenges of sign language, including its limited universality and usage, the project unfolds in seven key stages: data collection, feature engineering, data preparation, model design and training, testing and evaluation, and application development. Noteworthy testing results showcase a robust 95% training accuracy and an 85% testing accuracy. The envisioned web application boasts a feature-rich interface encompassing gesture recognition, user ratings, translation history management, account administration, and an extensive sign language dictionary. Clarifying the system's functionality, diagrams are employed for enhanced comprehension. In the implementation phase, meticulous attention is paid to hardware and software configurations, with detailed setup instructions provided. System operations are thoroughly elucidated, alongside candid discussions on encountered challenges such as data quality issues and processing constraints. To ensure the system's reliability, a comprehensive testing regimen is executed, spanning video frame capture, model prediction, and web application feature validation. In summary, this project marks a good stride towards enhancing communication accessibility for the hearing impaired.

TABLE OF CONTENT

ABSTRACT.....	VI
LIST OF FIGURES.....	X
LIST OF TABLES.....	IX
LIST OF ABBREVIATIONS	XI
CHAPTER 1 INTRODUCTION.....	1
1.1 Problem Statement and Motivation	1
1.2 Research Objectives.....	2
1.3 Project Scope and Direction	3
1.4 Contributions	4
1.5 Report Organization.....	5
CHAPTER 2 LITERATURE REVIEWS	6
2.1 Review of the Technologies	6
2.1.1 Dataset	6
2.1.2 MediaPipe Holistic	8
2.1.3 OpenCV	9
2.1.4 Long Short-Term Memory (LSTM)	10
2.1.5 Firebase Web Hosting.....	12
2.2 Review of Existing Sign Language Recognition Systems	13
2.2.1 Arabic Sign Language Alphabets Translator (ArSLAT).....	13
2.2.2 Real Time Indian Sign Language Recognition (RTISLR) System.....	14
2.2.3 ASL Translator: A Real-Time System for Recognition of American sign Language by using Deep Learning.....	16
2.2.4 DataFlair Sign Detection System.....	19
2.2.5 Sign Language to Text Conversion System (SL2T System) ...	21
2.2.6 Summary of Existing Sign Language Recognition Systems ...	23
CHAPTER 3 PROPOSED METHOD.....	26
3.1 System Development Methodology	26
3.2 System Design	27

3.2.1 System Architecture Diagram.....	27
3.2.2 Use Case Diagram and Description.....	29
3.2.3 Activity Diagram	31
CHAPTER 4 SYSTEM DESIGN	36
4.1 System Block Diagram	36
4.2 Data Collection	37
4.2.1 Overview.....	37
4.2.2 Requirements	37
4.2.3 Implementation Details.....	37
4.3 Feature Engineering.....	40
4.3.1 Overview.....	40
4.3.2 Requirements	40
4.3.3 Implementation Details.....	40
4.4 Data Preparation	43
4.4.1 Overview.....	43
4.4.2 Requirements	43
4.4.3 Implementation Details.....	43
4.5 Model Design, Training, and Hyperparameter Tuning	45
4.5.1 Overview.....	45
4.5.2 Requirements	45
4.5.3 Implementation Details.....	45
4.6 Model Testing and Evaluation.....	49
4.6.1 Overview.....	49
4.6.2 Requirements	49
4.6.3 Implementation Details.....	49
4.7 Application Development and Model Integration	54
4.7.1 Overview.....	54
4.7.2 Requirements	54
4.7.3 Implementation Details.....	54
CHAPTER 5 SYSTEM IMPLEMENTATION	56
5.1 Hardware Setup	56
5.2 Software Setup.....	57
5.3 System Operation.....	58
5.3.1 Overview.....	58
5.3.2 Operations.....	58
5.4 Implementation Issues and Challenges.....	63

5.4.1 Overview.....	63
5.4.2 Issues of MS-ASL Dataset.....	63
5.4.3 Issues of Dataset	64
5.5 Concluding Remark	65
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	66
6.1 System Verification Plan	66
6.1.1 Human Body Capture via Camera.....	66
6.1.2 Landmarks Extraction via MediaPipe Holistic.....	67
6.1.3 American Sign Language Recognition Model	68
6.1.4 SignSpeak Web Application.....	68
6.1.5 Account Management Module	69
6.1.6 Recognition Module	69
6.1.7 Dictionary Module.....	70
6.1.8 Transaction History Management Module	70
6.2 Testing Setup and Result	71
6.2.1 Human Body Capture via Camera.....	71
6.2.2 Landmarks Extraction via MediaPipe Holistic.....	73
6.2.3 American Sign Language Recognition Model	77
6.2.4 Account Management Module	80
6.2.5 Dictionary Module.....	82
6.2.6 Transaction History Management Module	84
6.3 Project Challenges	86
6.4 Objective Evaluation	87
6.5 Concluding Remark	87
CHAPTER 7 CONCLUSION AND RECOMMENDATION	89
7.1 Conclusion	89
7.2 Recommendations.....	89
REFERENCES	1

LIST OF FIGURES

Figure Number	Title	Page
Figure 2-1	User Interface of ASLLRP Sign Bank	7
Figure 2-2	Overview of MediaPipe Holistic Pipeline	8
Figure 2-3	Structure of LSTM Cell	10
Figure 2-4	The Structure of BiLSTM Model	11
Figure 2-5	The Structure of Seq2Seq LSTM Model	12
Figure 2-6	Result of Translation of ‘Ra’, ‘Lam’, ‘Kaf’ Letters	13
Figure 2-7	Examples of Recognizable Signs.	15
Figure 2-8	Sample Result of a Sign	15
Figure 2-9	Results of Test Run for ‘3’, ‘5’, ‘I’, ‘7’	17
Figure 2-10	The Inputs and Outputs of DataFlair SDS	19
Figure 2-11	User Interface of SL2T System	21
Figure 3-1	Waterfall model	26
Figure 3-2	System Architecture Diagram	27
Figure 3-3	Use Case Diagram	29
Figure 3-4	Translate Sign Language Activity Diagram	31
Figure 3-5	Browse Sign Language Dictionary Activity Diagram	33
Figure 3-6	Login User Account Activity Diagram	34
Figure 3-7	Create User Account Activity Diagram	35
Figure 4-1	System Block Diagram	36
Figure 4-2	JSON Files of MS-ASL Dataset	38
Figure 4-3	The Structure of Sample Json File	38
Figure 4-4	The Directory Structure of the Finalized Dataset	39
Figure 4-5	The Difference between BGR (Left) and RGB (Right)	40
Figure 4-6	The Directory Structure of Landmarks	42
Figure 4-7	Pseudocode of Frame Sampling Algorithm	44
Figure 4-8	Model Architecture	45
Figure 4-9	Step Decay Learning Rate Scheduler Settings	46
Figure 4-10	Early Stopping Mechanism Settings	47

Figure 4-11	Keras Tuner Configuration Setting	47
Figure 4-12	Multiclass Confusion Matrix	51
Figure 4-13	Bar Chart for Precision Scores	52
Figure 4-14	Bar Chart for Recall Scores	53
Figure 5-1	Create Account	58
Figure 5-2	Login Account	59
Figure 5-3	Username on Top Right Corner	59
Figure 5-4	Default Username on Top Right Corner	60
Figure 5-5	Recognition Mode	60
Figure 5-6	Learning Mode	61
Figure 5-7	Dictionary Module with Search Feature	61
Figure 5-8	Dialog with Video Clip	62
Figure 5-9	Transaction History Table with Pagination Feature	62
Figure 5-10	Dialog Showing Record Details and Delete Button	63

LIST OF TABLES

Table Number	Title	Page
Table 2-1	The Summary of Strengths and Limitations of Existing SLR Systems	23
Table 4-1	Default Configuration Options for MediaPipe Holistic	41
Table 4-2	Set of Hyperparameters for Hyperparameter Tuning	47
Table 4-3	Optimal Set of Hyperparameters	48
Table 4-4	List of Classes and Corresponding Indices	50
Table 5-1	Specifications of laptop	56
Table 5-2	Software Setup	57
Table 6-1	Test Cases for Evaluating Human Body Capture	66
Table 6-2	Test Cases for Evaluating Landmarks Extraction	67
Table 6-3	Test Cases for Evaluating Landmarks Extraction	68
Table 6-4	Test Cases for Evaluating SignSpeak Web Application	68
Table 6-5	Test Cases for Evaluating Account Management Module	69
Table 6-6	Test Cases for Evaluating Recognition Module	69
Table 6-7	Test Cases for Evaluating Dictionary Module	70
Table 6-8	Test Cases for Evaluating Transaction History Management Module	70
Table 6-9	Test Case for Testing Frame Capturing Method	71
Table 6-10	Test Case for Testing Frame Capturing Method in Capturing Human Body	72
Table 6-11	Test Case for Testing Frame Capturing Method in not Capturing Human Body	72
Table 6-12	Test Case for Testing Landmarker's Ability in Detecting Holistic Keypoints	73
Table 6-13	Test Case for Testing Landmarker's Ability in Detecting Face and Pose Keypoints	74
Table 6-14	Test Case for Testing Landmarker's Ability in Detecting Left Hand Keypoints	75

Table 6-15	Test Case for Testing Landmarker’s Ability in Detecting Right Hand Keypoints	76
Table 6-16	Test Case for Testing System’s Ability to Recognize from Front	77
Table 6-17	Test Case for Testing System’s Ability to Recognize from Side	78
Table 6-18	Test Case for Testing System’s Ability to Recognize in Low Light	79
Table 6-19	Test Case for Testing the Input Validation Mechanism	80
Table 6-20	Test Case for Testing User Authentication Mechanism	81
Table 6-21	Test Case for Testing the Searching Feature when Dealing with Alphabetical Query Term	82
Table 6-22	Test Case for Testing the Searching Feature when Dealing with Numerical Query Term	83
Table 6-23	Test Case for Testing Pagination Feature	84
Table 6-24	Test Case for Testing Deletes History Record Feature	85

LIST OF ABBREVIATIONS

<i>SLR</i>	Sign Language Recognition
<i>ASL</i>	American Sign Language
<i>CV</i>	Computer Vision
<i>HAR</i>	Human Action Recognition
<i>WHO</i>	World Health Organization
<i>AI</i>	Artificial Intelligence
<i>ArSLAT</i>	Arabic Sign Language Alphabets Translator
<i>RTISLR</i>	Real Time Indian Sign Language Recognition
<i>LSTM</i>	Long Short-Term Memory
<i>TP</i>	True Positive
<i>FP</i>	False Positive
<i>TN</i>	True Negative
<i>FN</i>	False Negative
<i>URL</i>	Uniform Resource Locator
<i>FPS</i>	Frame Per Second

CHAPTER 1 INTRODUCTION

This section begins by highlighting the importance of communication in human life and society. It acknowledges the prevalence of verbal communication but identifies a significant group, the deaf and hard of hearing, who face communication challenges due to their physical condition. The section also emphasizes the use of sign language as an alternative means of communication for this group and identifies factors contributing to difficulties in sign language communication. It concludes by presenting the motivation for the project, which is to develop a Human Action Detection (HAR) system, particularly a SLR mobile application, to bridge the communication gap between sign language users and non-sign language users.

1.1 Problem Statement and Motivation

Communication is an essential aspect of human life and society as it facilitates the exchange of knowledge and information among individuals while fostering positive relationships. Verbal communication, which relies on spoken words, is the most common form of communication used by people. However, there is one large group of people unable to benefit from it due to their physical condition, and they are deaf and hard of hearing. Therefore, they must choose the other way of communicating to communicate with others, otherwise they will have difficulty in making others to understand what they think. In such condition, **most of them will choose to use sign language as the way to express their thoughts, feelings, and intentions.**

The action of performing sign language is called signing. Signing is primarily used by the deaf and hard of hearing, but it can also be used by hearing people, such as those who cannot speak, those who have disabilities that make speaking difficult, and those with deaf family members, including children of deaf adults. However, **most of the sign language users are facing difficulty in using sign language to facilitate their daily communication efficiently**, especially for those who take sign language as their native language. And there are few contributing factors to this problem.

Firstly, the **lack of ubiquity of sign language** is one of the important factors. This is because most people consider sign language to be difficult to learn and practical application is often limited, resulting in a lower willingness to develop the skill. Secondly, the second contributing factor to this problem is the **lack of universality of sign language**. There are over 300 different sign languages in use by more than 70

million people globally [1] and each sign language has its own grammar, syntax, and vocabulary, thus it is difficult for sign language users to communicate with each other if they do not share the same sign language.

Therefore, the motivation of this project is to **develop a HAR system to address the stated problems by detecting and tracking the motions performed by human in order to understand their intentions by performing interpretations on those motions.** This can be done by developing a SLR mobile application to provide a platform to bridge the communication gap between sign language users and sign language and non-sign language users.

1.2 Research Objectives

The objective of this project is **to develop a web application that can translate ASL into English language using CV technology.**

This system will be able to:

- **Translate sign language actions captured by the camera of the users' handheld devices into their corresponding English words or phrases.**
- **Accurately recognize and translate 50 different ASL actions.**
- **Enable the users to view the graphical representations of key landmarks detected on their bodies in real time.**
- **Present users with a sign language dictionary containing a collection of sign language video clips.**

The system simplifies sign language communication by converting it into English and displaying it on a screen. Users can easily capture and translate sign language actions performed by anyone to enable them to understand the corresponding English words or phrases. As a result, the users can easily grasp the meaning behind the signer's expressions, facilitating effective communication of feelings and intentions.

Besides that, the system can accurately recognize and translate 50 different American Sign Language (ASL) signs and phrases covering essential topics like daily activities, emotions, objects, relationships, and concepts. This broad vocabulary allows effective communication between sign language users and non-signers, empowering users to express their needs, preferences, and inquiries independently. It also supports

CHAPTER 1

educational pursuits and social connections by including signs related to learning and relationships.

Furthermore, the ability to view graphical representations of key body landmarks detected by the system in real-time provides users with visual feedback confirming active body tracking, as well as an understanding of exactly which body parts are being recognized. This visualization feature builds user confidence in the system's functionality, allows users to optimize their positioning and movements for improved accuracy, and enhances transparency and engagement with the body tracking process.

Lastly, this application provides users a sign language dictionary that features an extensive list of sign language actions, with each action accompanied by a clear description and a video illustration demonstrating how to perform the sign with precision. Users can use the dictionary as a reference tool to learn sign language.

In summary, this system enables sign language communication through real-time translation, visual body tracking feedback, and an integrated sign language dictionary. It empowers effective interaction between signers and non-signers while promoting sign language education.

1.3 Project Scope and Direction

In the end, an Android Sign Language Recognition Web Application will be developed. The application will utilise the technology CV to bridge the communication gap between sign language and non-sign language users.

This proposed mobile application **provides text translation for the sign language signs corresponding to various alphabets, words, and phrases in ASL.** The system **translates 50 key ASL signs/phrases spanning daily life** to independent communication between signers and non-signers.

The system **provides real-time visualization of detected body landmarks**, giving users feedback on active tracking. This allows them to optimize their positioning and movements for improved accuracy.

Lastly, the system **provides a collection of video clips of sign language sign along with their corresponding meanings** or translations. Each video clip features a virtual signer demonstrating a specific sign language action in an easily visible environment.

1.4 Contributions

According to the World Health Organization (WHO), over 1.5 billion people worldwide, approximately 19% of the world's population, live with hearing loss, with 5% requiring rehabilitation to address their disabling hearing loss. Worse still, this number is expected to rise to 2.5 billion by 2050. Factors contributing to hearing loss and deafness are diverse, including intrauterine infections, low birth weight, chronic ear infections, and smoking. These factors can affect people at different stages of life, leading to the deaf community comprising individuals across all age groups [2]. Therefore, this means that **this application has a large audience and can benefit many people through its features** since the target audience of this application is deaf and hard of hearing individuals.

Other than that, the communication gap caused by physical condition often leads to distortion of conveyed messages, resulting in misunderstandings, confusion, and associated loss of time and resources. For example, in clinical settings, communication gaps may cause healthcare professionals to experience discomfort when serving deaf patients, impairing their capacity to fulfil their medical responsibilities. Furthermore, inadequate communication between healthcare professionals and deaf patients increases the likelihood of medical errors and results in a loss of trust between doctors and their patients. Thus, this application provides an opportunity for the sign language users to **prevent the consequences of the communication gap that they may face in the future.**

Furthermore, in the realm of sign language translation, the number of mobile applications available on Google Play that can convert sign language actions into text is surprisingly low, with less than five such apps boasting over 1000 downloads. Additionally, a significant portion of sign language-related mobile apps primarily focus on teaching users how to sign rather than offering translation capabilities. This scarcity of suitable options indicates an underserved target audience due to different reasons, presenting an opportunity for this application to **emerge as a good alternative solution** since this application also provides feedback on learning outcomes of their sign language learning.

The previous studies discussed in this field share certain limitations that negatively impact key factors contributing to successful sign language recognition. These

limitations include recognition accuracy, user autonomy, and system practicality. Since most of the previous system can only work based on a single frame, this application **increases the system practicality by accepting video as input, allowing for recognition across several frames**. After that, this application **improves user autonomy by accurately recognizing the signs performed by users in different environments**. This can be done by improving the diversity of datasets used to train the ML model. Lastly, **the recognition accuracy is better in this application**, as it tracks both facial expressions and body movements to collect enough key points and represent sign language gestures accurately.

1.5 Report Organization

The report is organized to guide readers through a comprehensive exploration of the development of a Sign Language Recognition System. It begins with an abstract summarizing the research, followed by lists of figures, tables, and abbreviations for convenient reference. Chapter 1 introduces the research problem, objectives, scope, contributions, and provides an overview of the report's organization. Chapter 2 reviews pertinent literature, while Chapter 3 outlines the proposed methodology. Chapters 4 through 6 delve into system design, implementation, evaluation, and discussion, covering various aspects such as data collection, model design, testing, and challenges faced. Finally, Chapter 7 offers conclusions drawn from the research and recommendations for future endeavors. The report's structure ensures coherence and facilitates readers' understanding and navigation through the presented research.

CHAPTER 2 LITERATURE REVIEWS

This chapter provides a comprehensive review of the technologies and existing systems relevant to sign language recognition. It covers essential datasets such as the MS-ASL and ASLLRP Sign Bank, which serve as valuable resources for training and testing sign language recognition models. The chapter also explores key technologies like MediaPipe Holistic for human pose and hand tracking, OpenCV for computer vision tasks, and Long Short-Term Memory (LSTM) networks for sequence modeling. Additionally, it presents an in-depth analysis of various existing sign language recognition systems, including ArSLAT, RTISLR, ASL Translator, DataFlair Sign Detection System, and SL2T System. The chapter highlights the strengths, limitations, and functionalities of these systems, providing insights into their recognition approaches, accuracy, and real-time capabilities. Overall, this chapter lays the foundation for understanding the current state of sign language recognition technologies and existing systems, paving the way for further research and development in this field.

2.1 Review of the Technologies

2.1.1 Dataset

MS-ASL Dataset [3]

The MS-ASL dataset is a large-scale dataset for American Sign Language (ASL) recognition. It was created by researchers at Microsoft and the Chinese University of Hong Kong. The dataset consists of videos of ASL signs performed by multiple signers. This dataset is expansive, encompassing over 25,000 annotated videos drawn from real-world sign language usage. It was chosen primarily because of its recent publication year and substantial size, rendering it highly suitable for training deep learning models. The dataset is divided into four subsets, namely MS-ASL100, MS-ASL200, MS-ASL500, and MS-ASL1000. The numerical value in each subset's name corresponds to the number of distinct ASL actions involved. For instance, MS-ASL100 comprises video clips representing 100 different ASL actions.

Each of these subsets has been further subdivided into training, testing, and validation sets to facilitate model development and evaluation. According to [12], the dataset

originally had an average of approximately 60 video clips per ASL action. However, due to unforeseen factors like video deletions or restrictions on public access, the actual average number of clips per action is considerably lower. These factors have also led to imbalanced sample proportions among the subsets. Consequently, it was necessary to combine these subsets and reorganize them for more balanced and effective splitting.

ASLLRP Sign Bank [4]

The ASLLRP (American Sign Language Linguistic Research Project) Sign Bank is a resource that provides access to videos of American Sign Language (ASL) signs, along with linguistic annotations. It enables searching and viewing ASL signs produced by multiple ASL signers, both in citation form (isolated signs) and in sentential context. The isolated sign datasets can currently be downloaded from the Sign Bank website, the user interface of which is shown in the figure and the examples segmented from continuous signing corpora will soon also be made available for download.

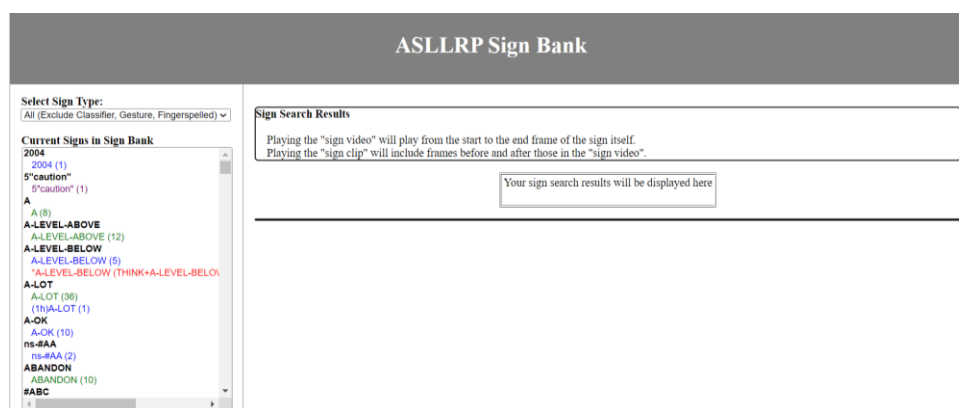


Figure 2-1 User Interface of ASLLRP Sign Bank

After that, the signs in the ASLLRP Sign Bank have been linguistically annotated in a consistent manner across the datasets. The annotations provide information such as the text-based gloss labels for the signs, handshape details (start and end handshapes), sign types (e.g., classifiers, fingerspelled signs), and frame numbers indicating the start and end points of the sign in the video. Basically, this dataset includes citation-form sign datasets from three sources: the BU American Sign Language Lexicon Video Dataset (ASLLVD), Rochester Institute of Technology (RIT), and DawnSignPress (DSP). For each of these datasets, a spreadsheet (CSV file) is provided which contains the annotations for the signs in that dataset.

Furthermore, ASLLRP Sign Bank uses consistent conventions for labeling and annotating the signs which can help to ensure a one-to-one correspondence between the text-based gloss labels and the sign productions. This consistency is essential for research purposes, such as sign recognition. Additionally, the ASLLRP has provided a spreadsheet that assigns consistent gloss labels to a large portion of the WLASL (Word-Level American Sign Language) dataset, enabling the combination of the ASLLRP and WLASL datasets into a larger resource for sign recognition research.

2.1.2 MediaPipe Holistic

MediaPipe Holistic is an innovative solution developed by Google that allows for simultaneous perception and tracking of human pose, facial landmarks, and hand gestures in real-time on mobile devices. This state-of-the-art system combines and optimizes separate models for these three tasks into a unified, end-to-end pipeline. This pipeline delivers a groundbreaking 540+ keypoints, including 33 for pose, 21 per hand, and 468 for facial landmarks, all while achieving near real-time performance.

The architecture of the MediaPipe Holistic pipeline integrates separate models for pose estimation, face landmarks, and hand tracking. It starts by estimating the human pose using BlazePose's pose detector and keypoint model. Then, using the inferred pose keypoints, regions of interest (ROIs) for each hand and the face are derived. To refine the accuracy of these ROIs, lightweight re-crop models are employed. The input frame is then cropped to these ROIs, and task-specific face and hand models estimate their respective keypoints. Finally, all these keypoints are merged to produce the full 540+ keypoints. The figure illustrates the overview of MediaPipe Holistic pipeline [5].

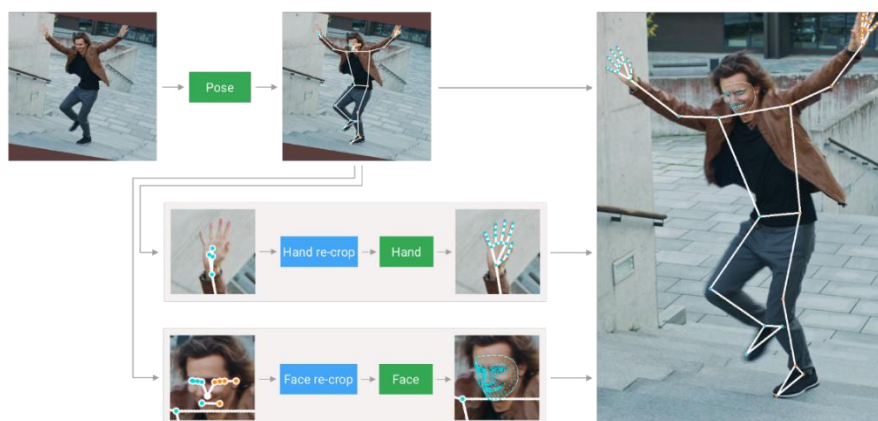


Figure 2-2 Overview of MediaPipe Holistic Pipeline

Next, performance optimization is a key feature of MediaPipe Holistic. The pipeline coordinates up to 8 models per frame, including pose, re-crop, and keypoint models for hands and face. Pre- and post-processing algorithms are optimized by moving computations to the GPU, resulting in approximately 1.5x speedup [5]. The multi-stage nature of the pipeline allows for replacing models with lighter or heavier versions based on performance and accuracy requirements. Additionally, inference on hands and face can be skipped if they are not within the frame bounds, saving compute resources.

Lastly, MediaPipe Holistic is accessible on-device for both mobile (Android, iOS) and desktop platforms. Google provides ready-to-use APIs for research (Python) and web (JavaScript) which facilitates its adoption and experimentation in various projects.

2.1.3 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. [6]

These algorithms can be used for a wide range of tasks such as detecting and recognizing faces, identifying objects, classifying human actions in videos, tracking camera movements, tracking moving objects, extracting 3D models of objects, stitching images together to produce high-resolution panoramas, finding similar images from a database, removing red eyes from flash images, following eye movements, recognizing scenery and establishing markers for augmented reality, and more.

OpenCV has a large user community of over 47 thousand people and an estimated number of downloads exceeding 18 million [6]. The library is used extensively in companies, research groups, and by governmental bodies. Major companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, and Toyota employ the library, along with many startups that make extensive use of OpenCV.

OpenCV has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. It is geared towards real-time vision applications and

takes advantage of SIMD instructions when available. The library is written natively in C++ and has a templated interface that works seamlessly with STL containers. Full-featured CUDA and OpenCL interfaces are also being actively developed.

2.1.4 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special kind of recurrent neural network (RNN) architecture that is designed to model sequential data and capture long-term dependencies more effectively than traditional RNNs. It was introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997 to address the vanishing and exploding gradient problems that conventional RNNs face when learning long-range temporal dependencies [7]. The core idea behind LSTM is to introduce a gating mechanism that regulates the flow of information into and out of a memory cell, allowing it to selectively remember or forget information for long periods of time. This gating mechanism is implemented using specialized units called gates, which are learned during the training process [8].

An LSTM network is composed of a series of interconnected memory cells, each containing four main components. The cell state C_t is the horizontal line running through the LSTM cell, acting as the "memory" of the network. It carries relevant information from previous time steps, allowing it to preserve long-term dependencies in the data. The figure illustrates the structure of an LSTM cell. The yellow horizontal line running through the cell represents the cell state [8].

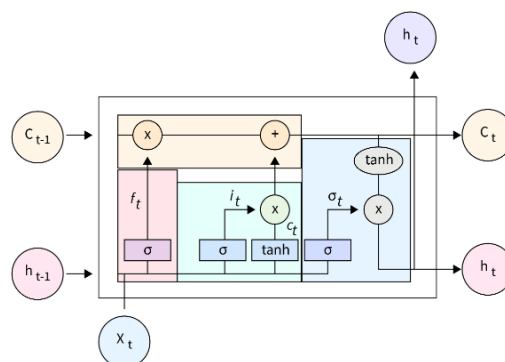


Figure 2-3 Structure of LSTM Cell

The forget gate f_t decides what information from the previous cell state should be kept or discarded. It takes the previous hidden state and current input as inputs and outputs a value between 0 and 1 for each element in the cell state, where 0 means "forget

completely" and 1 means "keep completely." The input gate i_t determines what new information from the current input and previous hidden state should be added to the cell state. It consists of two parts: a sigmoid layer that decides which values to update, and a \tanh layer that creates a vector of new candidate values to add to the state. The output gate O_t controls what information from the updated cell state should be used to compute the hidden state (output) for the current time step. It also filters the information that the LSTM will output, based on the updated cell state [8].

One important variant of LSTM is the Bidirectional LSTM (BiLSTM). In a standard LSTM, the flow of information is only from past to future. However, in many sequence modelling tasks like speech recognition or natural language processing, the context from the future is also important. A BiLSTM processes the input sequence in both the forward and backward directions, with two separate hidden layers, one for each direction. This allows the network to capture dependencies from both past and future contexts, improving its ability to learn from the sequential data [9].

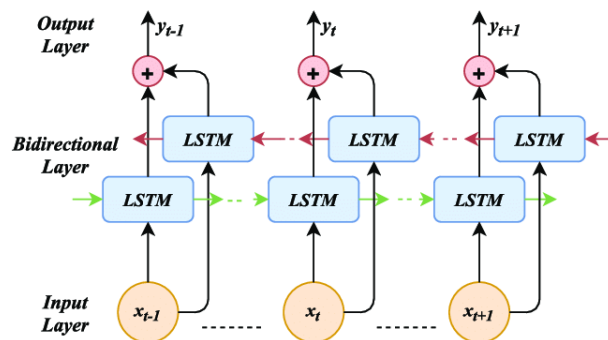


Figure 2-4 The Structure of BiLSTM Model

Another variant is the Sequence-to-Sequence (Seq2Seq) LSTM model, which is commonly used for tasks where the input and output sequences have different lengths, such as machine translation. The Seq2Seq model consists of two components: an encoder LSTM that processes the input sequence and encodes it into a fixed-length context vector, and a decoder LSTM that generates the output sequence one step at a time, using the context vector and the previously generated output as inputs [9].

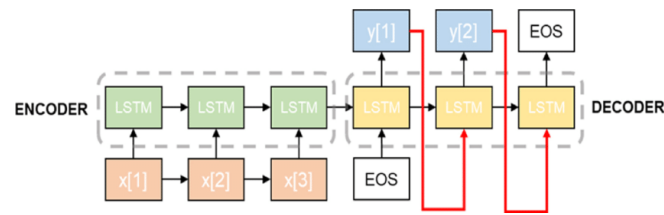


Figure 2-5 The Structure of Seq2Seq LSTM Model

2.1.5 Firebase Web Hosting

Firebase Hosting is a cloud-based web hosting platform offered by Firebase, a comprehensive app development platform owned by Google. It provides a fast, secure, and scalable solution for hosting web applications, static websites, and serverless backend services.

With Firebase Hosting, developers can deploy their web content to a global content delivery network (CDN) with just a single command using the Firebase Command Line Interface (CLI) [10]. This CDN is designed to serve files from the nearest edge location to users, ensuring low latency and fast loading times worldwide. Firebase Hosting automatically compresses files using modern compression algorithms like Brotli, further optimizing performance.

A key advantage of Firebase Hosting is its built-in support for secure connections. All content is served over HTTPS by default, with automatic provisioning of SSL/TLS certificates, eliminating the need for manual configuration [10]. This seamless integration of security measures aligns with modern web standards and user expectations for privacy and data protection.

In addition to static content like HTML, CSS, and JavaScript files, Firebase Hosting also supports serving dynamic content and backend services. Developers can integrate Firebase Hosting with other Firebase products, such as Cloud Functions for server-side logic and Cloud Run for containerized microservices. This versatility allows developers to build and host complete web applications, from the frontend to the backend, within the Firebase ecosystem [10].

In summary, Firebase Hosting is Google's cloud-based platform offering fast, secure hosting with global content delivery. It supports static and dynamic content and integrates seamlessly with other Firebase services for complete web application hosting.

2.2 Review of Existing Sign Language Recognition Systems

2.2.1 Arabic Sign Language Alphabets Translator (ArSLAT)

ArSLAT, a novel system developed by El-Bendary et al. [11], represents a significant advance in the field of Arabic sign language recognition. This innovative system utilizes a vision-based approach to accurately translate manual alphabets into text. The recognition task is achieved through the use of two distinct classifiers: the minimum distance classifier (MDC) and multilayer perceptron (MLP) classifier. Impressively, the system is able to recognize Arabic alphabets with an accuracy of 91.3% using the MDC classifier and 83.7% using the MLP classifier.

Functionalities of ArSLAT

To use ArSLAT, the user must first shoot a video of a person signing a series of letters, with only the hand appearing on camera. When switching from one letter to the next, the user must pause for a certain time (1 second) to allow the system to detect only one frame that actually represents each letter. Once the video is recorded, the user can upload it to the system as input for the recognition process. The results of recognition are displayed row by row, showing the one frame detected that represents each letter along with the corresponding letter in text form.



Figure 2-6 Result of Translation of 'Ra', 'Lam', 'Kaf' Letters [11]

Strengths and Limitations of ArSLAT

To begin with, it is important to acknowledge some of the notable strengths of the ArSLAT system. Firstly, the system is capable of recognizing multiple letters

simultaneously, allowing for efficient and streamlined recognition. As previously mentioned, users can input a video of a person signing a sequence of letters, and the system can accurately recognize each letter within the sequence. Additionally, the ArSLAT system is highly versatile, able to perform recognition in a wide range of environments without the need for specific equipment or clothing. This means that users can utilize the system in unrestricted environments without the need for gloves or other specialized gear.

However, it is equally important to recognize that the ArSLAT system also has several limitations that may affect its overall effectiveness. One notable limitation is that the system relies solely on single frames for recognition, making it difficult to accurately identify more complex sign language actions that require multiple frames for recognition. This can lead to potential misunderstandings or misinterpretations, limiting the system's ability to accurately capture and interpret sign language. Additionally, the system is limited in its ability to accurately localize and extract key points crucial for recognition. Specifically, it can only extract 51 key points to represent hand gestures, which significantly reduces the system's recognition accuracy. Finally, it is important to note that the ArSLAT system cannot perform recognition in real-time. Users must input pre-recorded videos of signed sequences in order for the system to recognize them, rather than signing directly in front of the camera for real-time recognition.

2.2.2 Real Time Indian Sign Language Recognition (RTISLR) System

The Real Time Indian Sign Language Recognition (RTISLR) system, proposed by Rajam et al. [12], is a system designed to recognize one of the south Indian languages with the vision-based approach. The system achieves this by recognizing a set of 32 signs, each representing the binary 'UP' and 'DOWN' positions of the five fingers. The system was trained using 320 images and tested using 160 images, resulting in a remarkable accuracy of 98.125%.

Functionalities of RTISLR system

To perform recognition using this system, users must first activate the system's camera then capture a photo of the sign they wish to recognize. The system then identifies the sign based on the positions of the fingertips of the signing hand, and the result is displayed on the screen.

CHAPTER 2

The recognition process of the RTISLR system utilizes a mechanism that involves using a single hand to represent binary numbers from 0 to 31 in a visual way. This mechanism employs the use of each finger on the hand to represent a binary digit, with lifted fingers representing one and lowered fingers representing zero. By manipulating the positions of the fingers, any binary number between 0 and 31 can be represented. The figure 1-6 explains this mechanism in a clearer way.



Figure 2-7 Examples of Recognizable Signs. [12]

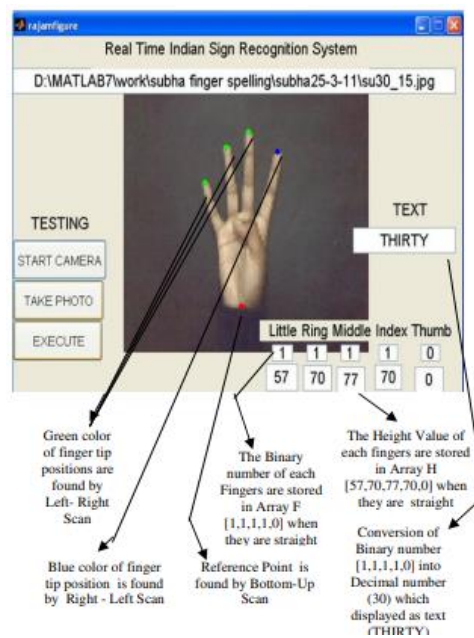


Figure 2-8 Sample Result of a Sign [12]

Strengths and Limitations of RTISLR system

The RTISLR system, despite its real-time sign recognition capabilities, is not without limitations. Its only discernable strength lies in its ability to accurately recognize signs in real-time, although users must manually capture and input images of signs. The following discussion outlines the various limitations of this system in greater detail.

Primarily, the system's sign recognition accuracy is limited to a controlled environment. The user must wear a wristband on the signing hand to prevent arm and palm extension images from interfering with the system's recognition abilities. Additionally, the system's recognition is limited to signs performed with the right hand, as it has been trained solely on images of right-hand gestures. Furthermore, the system can only recognize a single sign at a time, with real-time images serving as the only input for recognition. Furthermore, the RTISLR system and ArSLAT share a significant constraint in their limited ability to accurately localize and extract enough key points from images or video frames. As a result, these systems capture lesser information about sign language gestures, which reduced the accuracy of recognition. Specifically, the RTISLR system extracts only 6 key points, in contrast to ArSLAT's 51, which further exacerbates the recognition accuracy issue. Lastly, several similar limitations of ArSLAT, such as single-frame recognition, are also shared by the RTISLR system. Therefore, it is evident that the RTISLR system's limitations significantly outweigh its strengths.

2.2.3 ASL Translator: A Real-Time System for Recognition of American sign Language by using Deep Learning

ASL Translator is an advanced Sign Language Recognition (SLR) system, initially proposed by Taskiran et al. [13] in 2018. This system is designed to recognize 36 characters of American Sign Language (ASL), comprising 26 letters and 10 numbers. By employing Convolutional Neural Network (CNN) for training and image classification, ASL Translator has achieved a remarkable accuracy of 98.05%. The system has been trained and tested using a dataset collected in 2011 by Massey University, which includes 900 images for the 36 characters, with 25 samples for each character.

Functionalities of ASL Translator

ASL Translator offers users an effortless approach to recognition, without any unnecessary steps or procedures to perform SLR. By simply activating their device's camera and signing ASL in front of it from an appropriate distance, users can easily obtain accurate recognition results. The system then displays the recognition result on the left of the window, providing users with both the first and second guesses for the ASL sign performed.

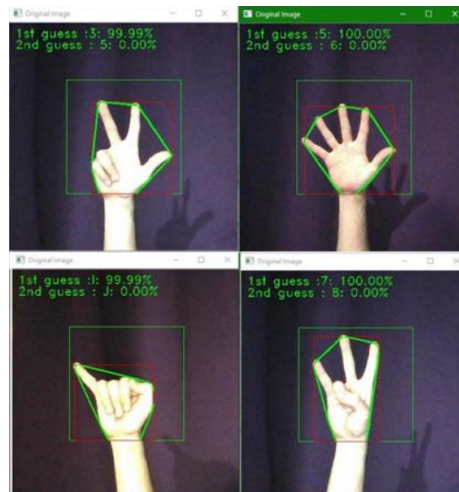


Figure 2-9 Results of Test Run for '3', '5', '1', '7' [13]

Strengths and Limitations of ASL Translator

ASL Translator's strengths are notable and set it apart from other sign language recognition systems. Firstly, it provides users with both the first and second guesses for the result of recognition, along with corresponding probabilities. This additional information enhances the system's usability by providing users with more insight into the level of confidence the system has in its interpretation. This feature is particularly advantageous in situations where accuracy is paramount, as it can help ensure that the message is conveyed correctly. Secondly, ASL Translator supports continuous recognition of multiple signs, enabling users to perform recognition directly after each sign. This feature sets it apart from other sign language recognition systems, such as ArSLAT and RTISLR, which require users to input images or videos for recognition after the completion of a recognition process. Additionally, ASL Translator is capable of performing real-time recognition, making it a highly effective tool for bridging the communication gap between deaf-mute and hearing communities.

CHAPTER 2

However, there are also some limitations of ASL Translator that are worth noting. Similar to other systems, ASL Translator rely on only a single frame to perform recognition, making it unsuitable for recognizing sign language that occurs across a sequence of frames. This can result in inaccurate recognition. Additionally, the system can only perform recognition for a single sign at once and is limited to recognizing signs that are performed solely with the hands. These limitations should be taken into account when using the system, and efforts to address them could enhance its effectiveness in recognizing sign language.

2.2.4 DataFlair Sign Detection System

DataFlair Sign Detection System is a sign detection system proposed by DataFlair [14] in 2023. It is developed using Python, OpenCV, and Keras with a TensorFlow backend, this sign language recognition project can effectively identify numbers from 1 to 10 based on hand gestures captured via webcam. The system's dataset is composed of images captured through the webcam, with 701 images per number for training and 40 for testing. Trained using a Convolutional Neural Network (CNN), the model architecture comprises convolutional, max-pooling, and dense layers. Remarkably, during training, the model attained 100% accuracy on the training set and approximately 81% accuracy on the validation set.

Functionalities of DataFlair Sign Detection System

Users can easily utilize the sign language recognition system by accessing its intuitive interface, which typically involves running the provided Python scripts. Upon execution, the system prompts users to position their hands within the designated region of interest (ROI) captured by a webcam. By performing hand gestures corresponding to numbers 1 through 10 within this ROI, users can then communicate with the system. The system's real-time processing capabilities enable it to instantly recognize and interpret these gestures, displaying the recognized numbers on the video feed in real-time. The figure 2-10 illustrates the behaviour of DataFlair SDS during recognition process.



Figure 2-10 The Inputs and Outputs of DataFlair SDS [14]

Strengths and Limitations of DataFlair Sign Detection System

The sign language recognition (SLR) method described has numerous major advantages and disadvantages. On the plus side, the system is simple and straightforward to implement as it utilizes basic computer vision techniques, such as background subtraction and contour detection. Its ability to perform real-time recognition using webcam feed enhances interactive communication. Moreover, the system's extensibility allows for future enhancements, like recognizing alphabets and additional gestures, through dataset expansion and model retraining.

The system's drawbacks are numerous. For first, it has a limited vocabulary, only recognising numerals 1 through 10, which severely limits its utility for sign language communication. Furthermore, it only recognises single-handed movements, ignoring the complex nature of multi-handed expressions. Its reliance on a controlled environment with steady backgrounds and perfect lighting further limits its usefulness, as real-world settings sometimes provide dynamic and congested environments in which performance may suffer. Furthermore, the system's single-frame recognition technique is ineffective in capturing continuous sign language sequences or motion-based motions. Users must adhere to precise hand positions and distances from the camera, which makes it harder for natural communication. Finally, the lack of ways to resolve occlusions, changing hand sizes, and variable orientations endangers the system's robustness in real-world circumstances.

2.2.5 Sign Language to Text Conversion System (SL2T System)

SL2T System is a desktop application developed by Y. Obi et al. [15] in 2023. It aims to recognize American Sign Language fingerspelling from live camera input and converts them to text in real-time. The system employs computer vision and pattern recognition techniques, using a 2-layer Convolutional Neural Network (CNN) model for gesture classification. The first CNN layer processes the input hand image and predicts the number of frames, while the second layer has algorithms to accurately predict similar letters/symbols. When tested on the authors' dataset for recognizing the 26 ASL alphabet letters, the model achieved an accuracy of 96.3%.

Functionalities of SL2T System

To use this application, users must launch the desktop application and position themselves in front of the computer's webcam, ensuring their hand gestures are clearly visible. The application's GUI displays a box to detect and read the user's ASL hand gestures. Users perform the hand gestures for the letters they want to communicate, keeping their hand within the specified region. The application processes the live camera feed, detects the hand gesture, and uses the trained CNN model to recognize and classify the ASL letter, displaying it in the "Character" row. To form a word, users hold the same gesture for approximately 5 seconds until the letter appears in the "Word" row. An autocorrect feature provides up to 3 suggestions for users to click and correctly form words. The GUI also includes an ASL symbol reference image. Optimal lighting and a plain background are recommended for accurate recognition. The figure 2-5 illustrates the user interface of SL2T system.

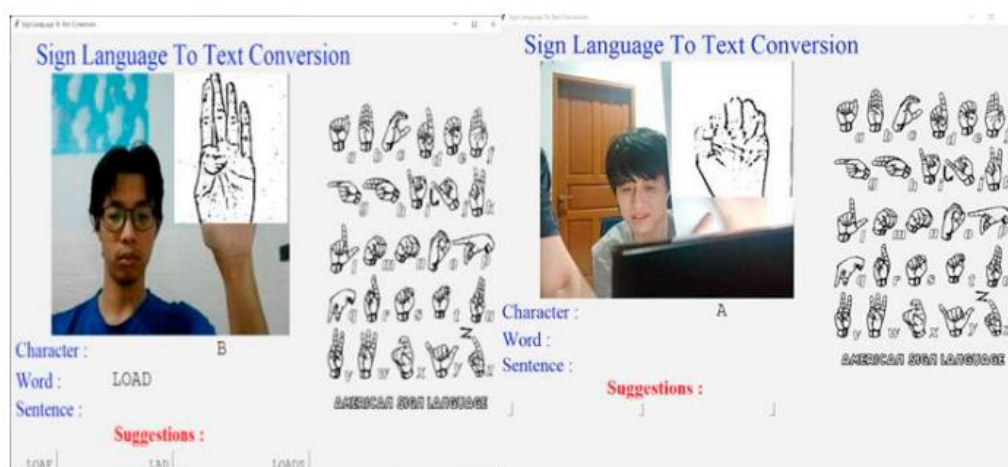


Figure 2-11 User Interface of SL2T System [15]

Strengths and Limitations of SL2T System

One of the notable strengths of SL2T system is its real-time capabilities. The system is designed to recognize and convert sign language gestures into text in real-time which allows for seamless communication between individuals who use sign language and those who do not. This real-time aspect is particularly beneficial in situations where instant communication is crucial, such as in medical emergencies or during in-person conversations. Additionally, the system's desktop application format makes it accessible and user-friendly, as it can be installed and operated on personal computers without the need for specialized hardware or complex setups.

While this system presents promising capabilities, there are certain limitations that should be acknowledged. One significant limitation lies in the scope of gestures it can recognize. The current system is focused on recognizing the 26 letters of the American Sign Language (ASL) alphabet, which is a subset of the comprehensive ASL vocabulary. Recognizing full words, phrases, and nuanced expressions present in ASL may require further advancements in the system's training data and algorithms. Additionally, the accuracy of the system can be influenced by various factors, such as lighting conditions, background complexity, and the quality of the camera used for input. Proper lighting and a plain background are necessary for optimal performance, which may not always be feasible in real-world scenarios. Furthermore, the system's reliance on the user's hand remaining in a specific region of the camera's field of view for a few seconds to form words can potentially hinder natural and fluid communication.

2.2.6 Summary of Existing Sign Language Recognition Systems

Features	ArSLAT	RTISLR	ASL Translator	DataFlair SDS	SL2T System
Category	Isolated	Isolated	Continuous	Continuous	Isolated
Sign Language Variant	Arabic Sign Language	Indian Sign Language	American Sign Language	American Sign Language	American Sign Language
Type of Features Centered	Landmarks	Landmarks	Hand Shape	Hand Shape	Hand Shape
Vision/ Sensor-Based Recognition	Vision-Based	Vision-Based	Vision-Based	Vision-Based	Vision-Based
Number of Recognizable Signs	28	32	36	10	26
Single/ Multi Frame Recognition	Single Frame	Single Fame	Single Frame	Single Frame	Single Frame
Real Time Recognition	No	Yes	Yes	Yes	No
Simultaneous Recognition of Multiple Signs	Yes	No	No	No	No
Unrestricted Recognition	Yes	Yes	Yes	Yes	Yes

Table 2-1 The Summary of Existing Sign Language Recognition System

CHAPTER 2

The table presents a summary of strengths and limitations of various existing sign language recognition systems. It categorizes the systems based on whether they recognize isolated signs or continuous sign language sequences. The systems cover different sign language variants, including Arabic Sign Language (ArSL), Indian Sign Language (ISL), and American Sign Language (ASL).

Most of the systems focus on recognizing hand shapes as the primary feature for sign language recognition, except for ArSLAT and RTISLR, which are landmark-centred. All the listed systems are vision-based, relying on computer vision techniques for sign language recognition. The number of recognizable signs varies across systems, ranging from 10 (DataFlair SDS) to 36 (ASL Translator).

All the systems perform recognition based on single frames or static images, rather than continuous video sequences. Some systems, such as RTISLR, ASL Translator, and DataFlair SDS, are capable of real-time recognition, while others, like ArSLAT and SL2T System, are not designed for real-time operation. Only the ArSLAT system has the capability to recognize multiple signs simultaneously, while the others can recognize one sign at a time.

As mentioned in the table, these systems have certain restrictions or conditions for recognition, such as controlled lighting, specific backgrounds, or limited variation in signer appearance or style. Such restrictions are common in many computers vision-based systems to ensure reliable performance.

The choice of sign language variant and the number of recognizable signs may depend on the target application, the availability of training data, and the specific conditions under which the system is expected to operate. Real-time recognition is an important factor for practical applications, as it enables more natural and efficient communication, but may be more challenging under varying conditions.

While most systems focus on hand shape features, incorporating additional features like hand motion, arm gestures, and non-manual features (facial expressions, head movements) could potentially improve recognition accuracy and robustness, but may also increase the computational complexity and the need for more training data under different conditions.

CHAPTER 2

The performance of these systems is likely to vary based on factors such as lighting conditions, occlusions, background complexity, and the signer's appearance or style, especially if the systems are designed to operate under specific constraints or controlled environments. Unrestricted recognition, as mentioned in the table, may not be entirely accurate, as most practical systems have certain limitations or conditions for optimal performance.

In summary, the table outlines strengths and limitations of various sign language recognition systems, categorized by recognition approach and language variant. Most focus on hand shapes, with limited real-time and simultaneous recognition capabilities. Performance depends on factors like lighting and signer variation, it highlights ongoing challenges in adapting to real-world conditions.

CHAPTER 3 PROPOSED METHOD

This chapter provides an overview of the system development methodology and design for the Sign Language Recognition System. Following the Waterfall model, the development process progresses sequentially through distinct phases, ensuring thorough analysis and minimal requirement changes. The chapter introduces the system architecture, illustrating its components across client-side, server-side, and database layers. Detailed design elements, including use case diagrams and activity diagrams, elucidate user interactions and system processes such as sign language translation, dictionary browsing, and user account management. Overall, this chapter sets the foundation for system development, outlining the structured approach and key functionalities of the proposed system.

3.1 System Development Methodology

The Waterfall model has been selected as the project development methodology for this project. This methodology follows a sequential development process, where each phase must be completed before proceeding to the next. The Waterfall model divides the system development process into five distinct phases, namely requirement, design, implementation, testing, and maintenance, as depicted in Figure 3-1. To ensure minimal changes to the system's requirements during the development process, each phase will undergo rigorous analysis and examination. This is particularly important because the Waterfall model has limited flexibility to revisit previous phases once they have been completed. The main advantage of this methodology is its ability to identify system requirements in detail, minimizing changes to the requirements as the project progresses.

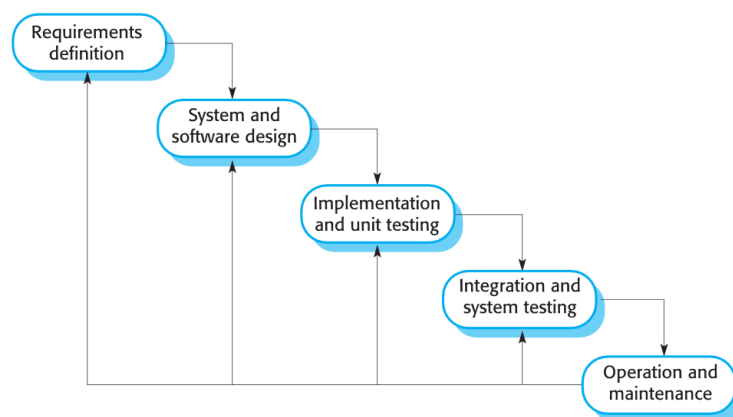


Figure 3-1 Waterfall model

3.2 System Design

3.2.1 System Architecture Diagram

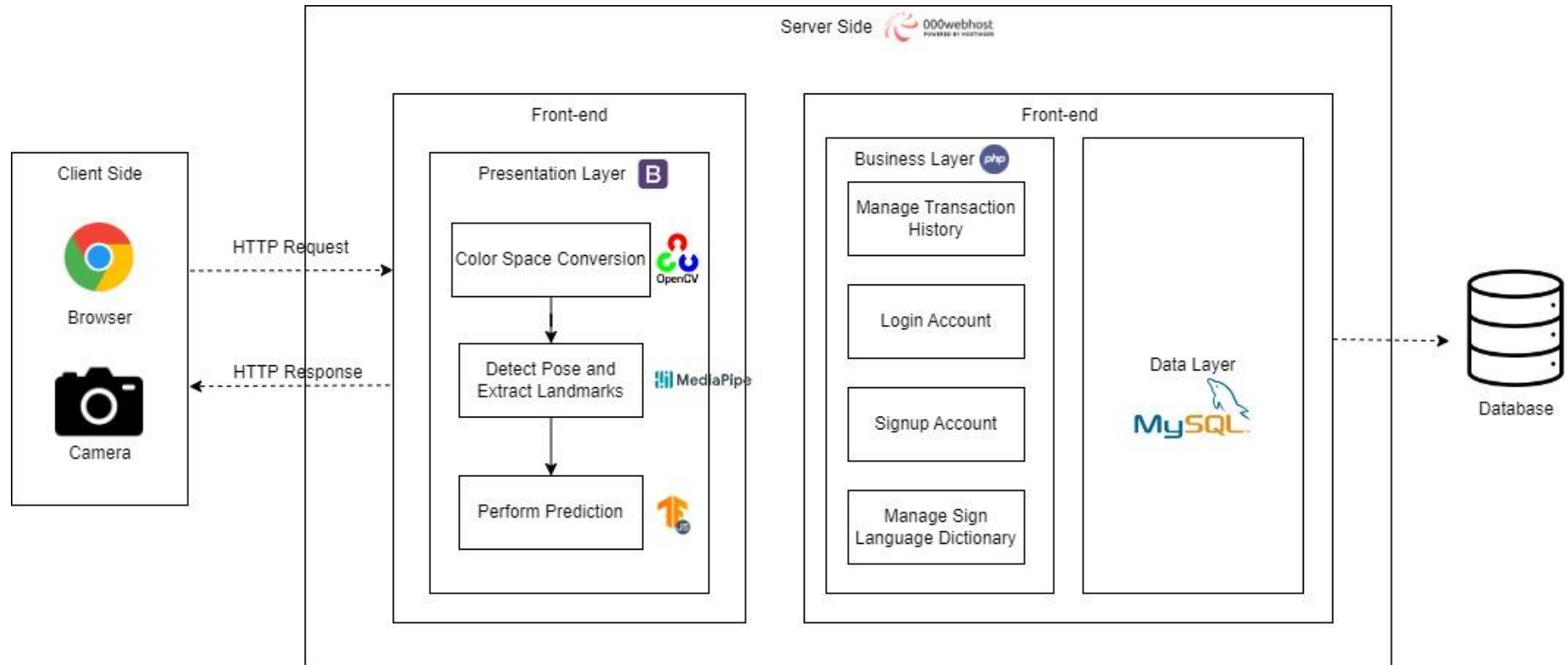


Figure 3-2 System Architecture Diagram

CHAPTER 3

The Figure 3-2 illustrates the architecture of the proposed system, representing the overall structure and components. The diagram is divided into three main sections: Client Side, Server Side, and Database. On the Client Side, there are two components: a Browser and a Camera. The Browser is responsible for sending HTTP requests and receiving HTTP responses from the Server Side, facilitating the user interaction with the web application. The Camera is used to capture images or video streams of sign language actions performed by the user.

The Server Side is further divided into two parts: Front-end and Business Layer. The Front-end consists of the Presentation Layer, it handles the user interface and presentation of information to the users. After that, the Presentation Layer consists of three main components which are Colour Space Conversion Detect Pose and Extract Landmarks and Perform Prediction. The Colour Space Conversion component is responsible for converting the captured image or video data into a suitable colour space format for further processing. The Detect Pose and Extract Landmarks component, represented by the MediaPipe icon, is responsible for detecting the poses and extracting landmarks from the captured sign language actions. The next component in the Front-end is Perform Prediction. This component is responsible for taking the extracted landmarks and feeding them into the classification model to predict the corresponding sign language action or meaning.

On the other hand, the Business Layer on the Server Side contains several components that handle various functionalities of the application. The Manage Transaction History component is responsible for managing and storing the history of sign language recognition transactions. The Login Account and Signup Account components handle user authentication and account management. The Manage Sign Language Dictionary component is responsible for managing and updating the sign language dictionary used by the application.

Lastly, the Data Layer is connected to a Database. This database is used to store user account information, sign language dictionaries, transaction histories, and other relevant data required by the application.

3.2.2 Use Case Diagram and Description

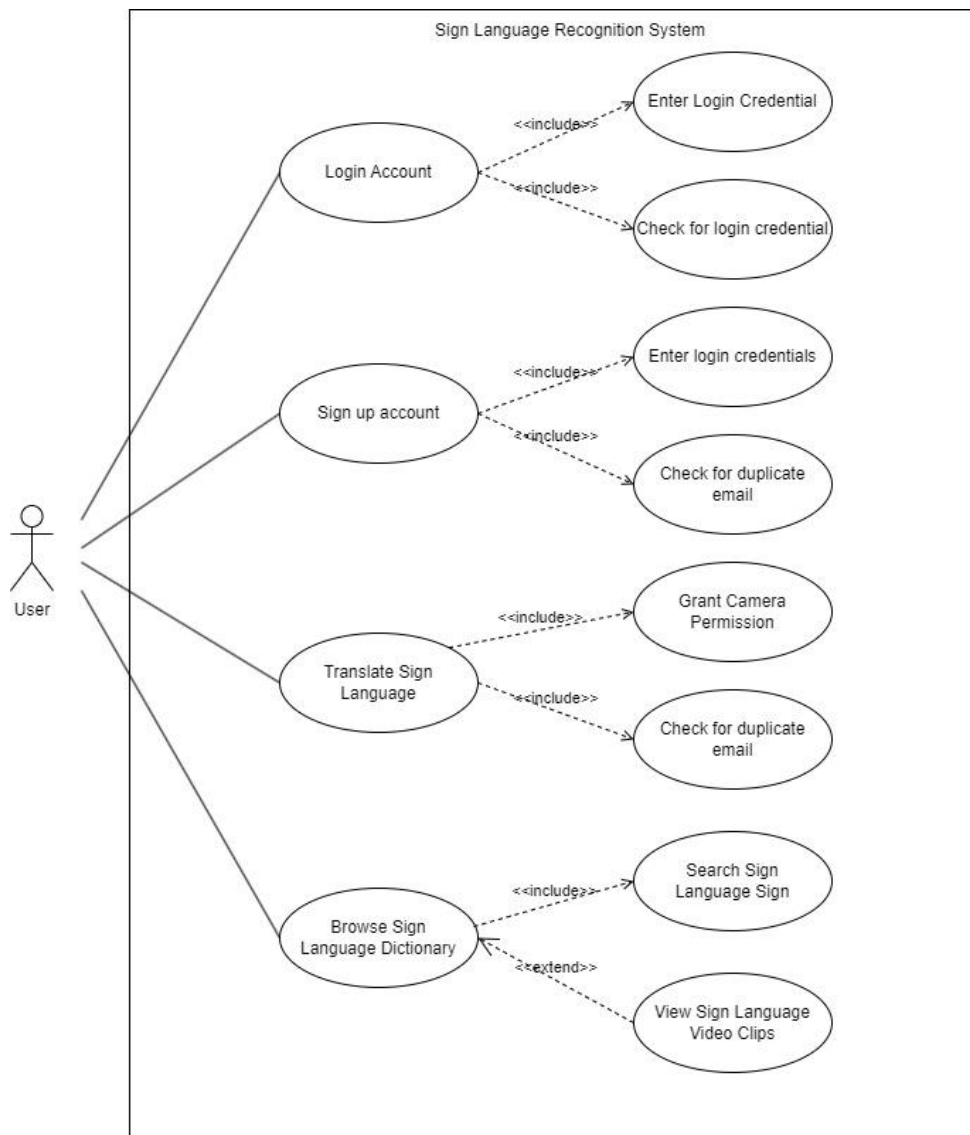


Figure 3-3 Use Case Diagram

The Figure 3-3 illustrates the use case diagram of the proposed system. The primary actors in this system are the Users, who can interact with the different functionalities provided by the system.

One of the core functionalities of the system is account management. Users can either log in to an existing account or sign up for a new account. The "Login Account" use case allows users to enter their login credentials and access their account, while the "Sign up account" use case enables new users to create a new account by providing their credentials and ensuring that their email address is not already registered.

CHAPTER 3

For users interested in learning or practicing sign language, the system offers several features. The "Translate Sign Language" use case allows users to translate between spoken or written language and sign language. The "Browse Sign Language Dictionary" use case provides access to a dictionary or repository of sign language gestures and their corresponding meanings.

The system also includes features for searching and viewing sign language video clips. The "Search Sign Language Sign" use case enables users to search for specific sign language signs or gestures, while the "View Sign Language Video Clips" use case allows users to watch video clips demonstrating various sign language signs and expressions.

For users who wish to use the system's sign language recognition capabilities, the "Grant Camera Permission" use case is necessary. This use case likely involves granting the application access to the device's camera, which is essential for capturing and interpreting sign language gestures.

Overall, the use case diagram illustrates the different functionalities and interactions that users can have with the proposed system, ranging from account management to learning, practicing, and utilizing sign language recognition features.

3.2.3 Activity Diagram

Translate Sign Language Activity

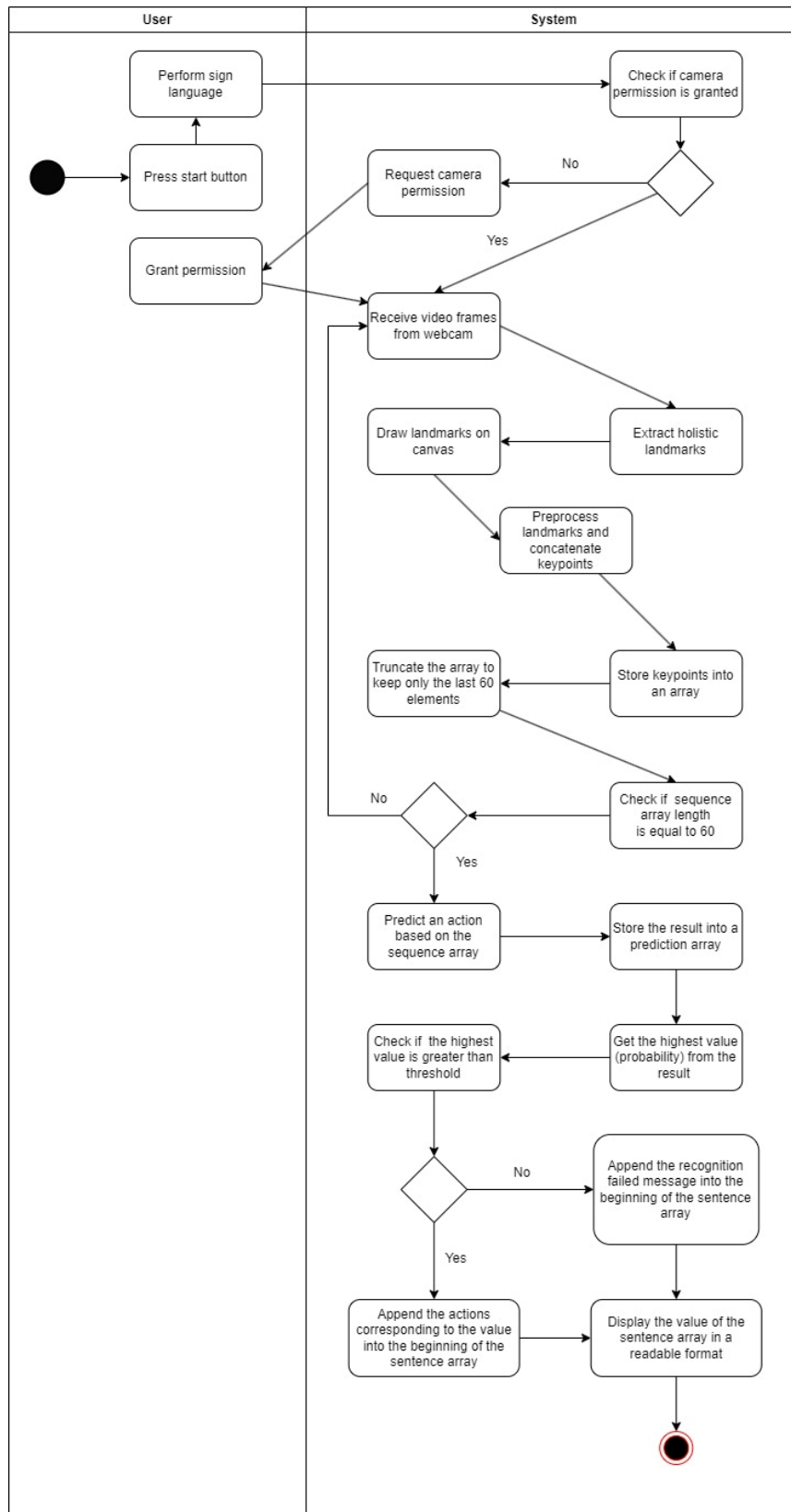


Figure 3-4 Translate Sign Language Activity Diagram

CHAPTER 3

Figure 3-4 illustrates the activity of translating sign language signs into English words. The process begins when the user performs sign language gestures and presses the start button to initiate the sign language recognition system. The system first checks if the camera permission is granted for accessing the webcam. If the camera permission is not granted, the system requests permission from the user to access the webcam. Once the user grants permission, the system can receive video frames from the webcam, capturing the user's sign language gestures.

After that, the system processes the video frames by extracting holistic landmarks and drawing them on a canvas for visualization. These landmarks are then pre-processed and concatenated into keypoints, which are essential for recognizing the sign language gestures. The system truncates the array of keypoints to keep only the last 60 elements, ensuring a fixed sequence length for further processing. It then checks if the sequence array length is equal to 60. If not, the system continues capturing and processing the video frames until the required sequence length is reached. Once the sequence array length is equal to 60, the system predicts an action based on the sequence of keypoints. This prediction is stored in a separate array, representing the recognized sign language gesture or action.

Next, the system then checks if the highest value (probability) in the prediction array is greater than a predefined threshold. If the highest value is below the threshold, it indicates a low confidence in the prediction, and the system appends a recognition failed message to the beginning of the sentence array. If the highest value in the prediction array exceeds the threshold, indicating a confident prediction, the system appends the corresponding action to the beginning of the sentence array. Finally, the system displays the value of the sentence array in a readable format, providing the user with the interpreted sign language gestures or actions.

Overall, this activity diagram outlines the process of obtaining camera permission, capturing video frames, extracting landmarks, preprocessing keypoints, predicting sign language gestures based on sequences of keypoints, and displaying the recognized actions or gestures to the user.

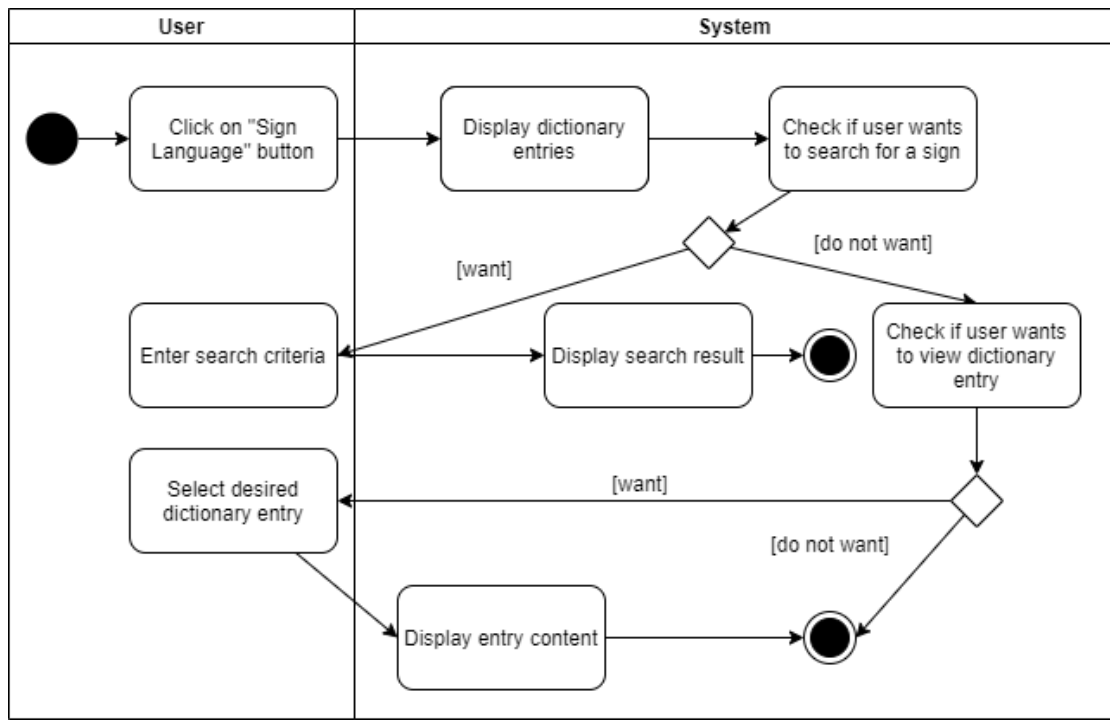
Browse Sign Language Dictionary Activity

Figure 3-5 Browse Sign Language Dictionary Activity Diagram

Figure 3-5 illustrates the process of browsing the sign language dictionary activity diagram. The initial step involves users clicking on the "Sign Language Dictionary" button, which triggers the display of the dictionary entries. Subsequently, the system checks whether the user intends to search for a specific sign. If the user wishes to perform a search, they are prompted to enter the search criteria, which then leads to the display of search results. Upon the completion of the search, the session concludes. Conversely, if the user does not wish to search for a sign, the system proceeds to determine whether the user intends to view any of the dictionary entries. If the user chooses to view a dictionary entry, they are required to select the desired entry for viewing. On the other hand, if the user does not wish to view a dictionary entry, the session concludes directly.

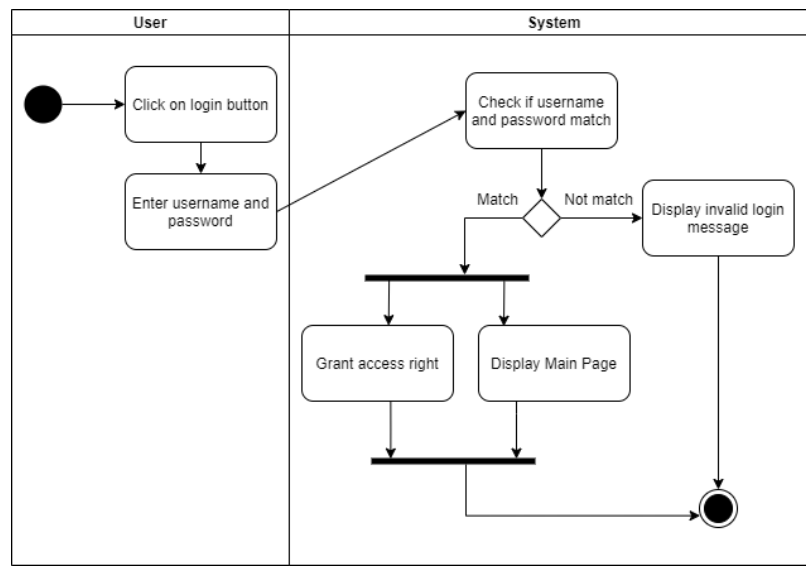
Login User Account Activity

Figure 3-6 Login User Account Activity Diagram

Figure 3-6 illustrates the activity of logging in to a user account. Initially, users are required to click on the “Login” button, which directs them to the login page. On this page, users can enter their username and password, and subsequently submit their credentials. The system then verifies whether the entered username and password match. If there is a match, the system grant access rights and simultaneously displays the main page to the users. Conversely, if the username and password do not match, the system promptly presents an invalid login message to notify the users of the unsuccessful login attempt.

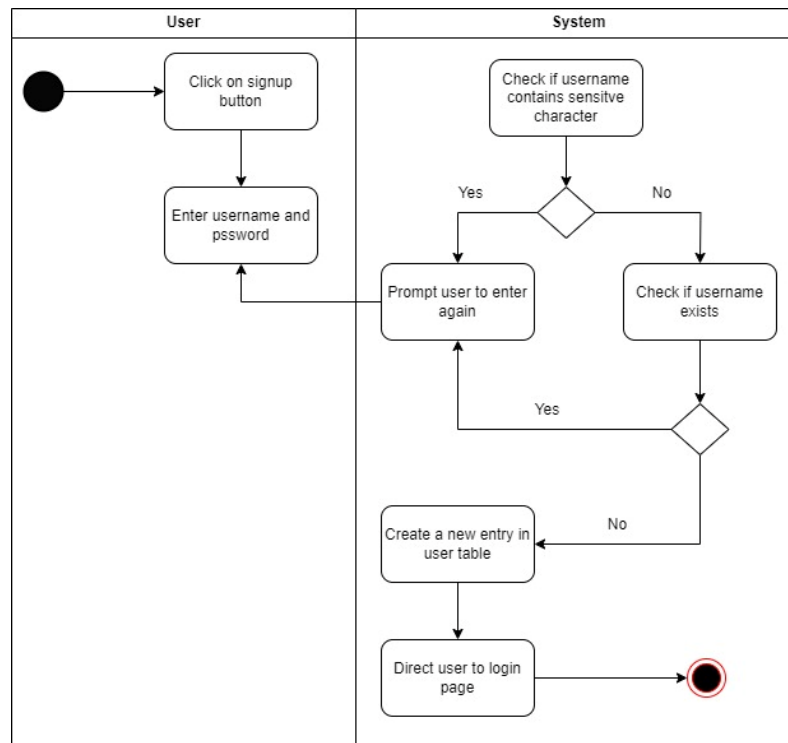
Create User Account Activity

Figure 3-7 Create User Account Activity Diagram

The Figure 3-7 is an activity diagram illustrating the process of creating a new user account in the proposed system. The process begins when the user clicks on the "Sign-up" button, indicating their intent to create a new account. This action triggers the system to perform a check on the username entered by the user to ensure it does not contain any sensitive characters or patterns that could potentially cause security vulnerabilities or conflicts. If the system detects that the proposed username contains sensitive characters, it prompts the user to re-enter the username again. This loop continues until the user provides a valid username that does not violate the system's rules. Once a valid username is entered, the system proceeds to the next step. After validating the username, the system checks if the entered username already exists in the user table or database. If the username is already taken, the system will not allow the user to proceed further, as usernames must be unique within the system. If the username is available and not taken, the system creates a new entry in the user table or database, associating the provided username and password with the new user account. Upon successful creation of the new user account, the system directs the user to the login page. At this point, the user can log in using their newly created credentials and access the system's features and functionalities.

CHAPTER 4 SYSTEM DESIGN

This chapter delves into the detailed process of building and deploying a sign language recognition model as a web application. It begins with data collection, where 50 commonly used signs are selected and supplemented with additional datasets to achieve the desired sample count. Feature engineering follows, extracting body and hand landmarks from video frames for model input. Data preparation standardizes video clips and splits the dataset for training and validation. Model design, training, and hyperparameter tuning involve creating an LSTM neural network, optimizing its architecture and parameters, achieving over 80% test accuracy. Model testing and evaluation assess the model's performance using various metrics, revealing strong training accuracy but some challenges in generalization. Finally, application development and model integration lead to the creation of SignSpeak, a web application featuring webcam-based sign language prediction, a sign dictionary, user authentication, and TensorFlowJS integration for browser-based model execution.

4.1 System Block Diagram

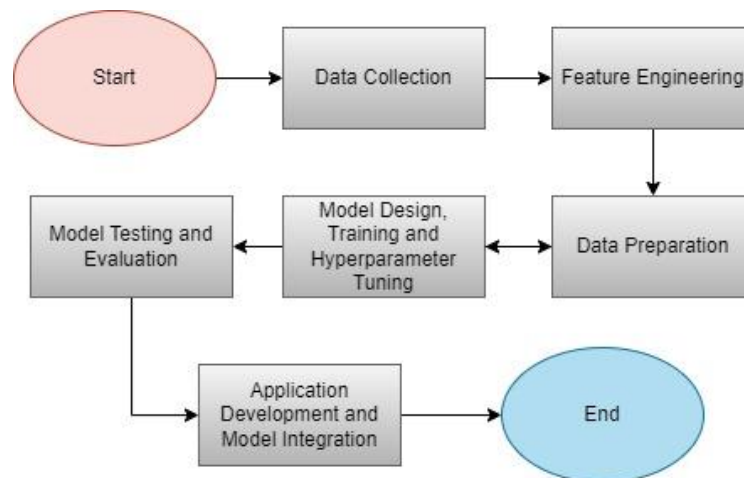


Figure 4-1 System Block Diagram

The block diagram depicts the end-to-end process of building and deploying a sign language recognition model as a web application. It begins with project initiation, followed by data collection, feature engineering, and data preparation. The core stages include model design, training, and hyperparameter tuning. After training, the model undergoes testing and evaluation to assess its performance. Once satisfactory, it moves to application development and integration. Finally, the process concludes with project completion. It's flexible to accommodate unique project needs and challenges.

4.2 Data Collection

4.2.1 Overview

The data collection module involved determining 50 commonly used signs as the target classes, with a desired sample count of 100 per class. The initial ASLLRP Sign Bank dataset provided some samples, but an additional MSAL dataset was leveraged to achieve the target count. The MS-ASL100 subset from MSAL, containing an average of 57.4 samples per class, was selected and supplemented with automatic YouTube video downloads using Python. Personal video recordings were also captured using OpenCV to reach the goal of 100 samples per class, resulting in a final dataset of 5000 samples across 50 sign classes.

4.2.2 Requirements

The anticipated result from this module is a dataset comprising 5,000 samples distributed across 50 sign classes. Each class is neatly organized into its own folder, with each sample residing within its respective class folder. This structured arrangement facilitates seamless and straightforward further processing.

4.2.3 Implementation Details

The first step in data collection module is to determine the actions that the sign language system can recognize. To ensure that the dataset is sufficiently large for effective model training, the desired number of samples for each class is set at 100. The actions are chosen based on two criteria:

The first criterion is that the number of samples in the dataset must be close to the desired number of samples, thus the dataset to be used must be decided first. A dataset called the American Sign Language Linguistic Research Project (ASLLRP) Sign Bank is found to fulfil this requirement. This dataset can be accessed through the official ASLLRP website, where users can view the available signs and the number of video clips for each sign. However, it has a limitation in that it only allows users to manually download the video clips one by one, making the process slow and cumbersome.

The second criterion is that the selected signs must correspond to words commonly used in daily communication to ensure the usability and practicality of the system. With these two criteria, 50 signs that meet the requirements are selected as the classes that can be recognized and translated by the system.

CHAPTER 4

Even though the video clips of the signs have been successfully downloaded and stored, the number of video clips for each sign still does not reach the desired count. Thus, another dataset is needed to achieve the target. In this case, the MSAL dataset is selected to enrich the dataset. This dataset provides several JSON files, as shown in Figure 4-2, which include the necessary information to acquire video clips from YouTube.

```
* `MSASL_train.json`: includes a json file of 16054 train samples.
* `MSASL_test.json`: includes a json file of 4172 test samples.
* `MSASL_val.json`: includes a json file of 5287 validation samples.
* `MSASL_classes.json`: includes a json file of 1000 glosses (words) representing classes in the classification task.
  The first word in the set (`hello`) is class 0 and the last word on the set (`challenge`) is class 999.
* `MSASL_synonym.json`: each row is a list of words that we considered synonym and assign a single class for all of them.
* `C-UDA-0.1_annotated_discussion.pdf`: the Computational Use of Data Agreement (C-UDA) license agreement.
* `README.md`: this file.
```

Figure 4-2 JSON Files of MS-ASL Dataset

The JSON files containing information about the samples are `MSASL_train.json`, `MSASL_test.json`, and `MSASL_val.json`. Each of them has the structure shown in Figure 4-3.

After that, this dataset includes four subsets: MS-ASL100, MS-ASL200, MS-ASL500, and MS-ASL1000. The number refers to the number of classes contained in the dataset. Since MS-ASL100 has the highest number of samples, with an average of 57.4 samples per class, MS-ASL100 is chosen. To obtain this subset, the train, test, and validation sets need to be filtered based on the 'label' key shown in Figure 4-3.

```
{'url': a url link to the video
'start_time': the starting point of the clip in the original video in seconds
'end_time': the starting point of the clip in the original video in seconds
'label': class (an integer between 0 to 1000)
'signer_id': the id of the signer
'box': the body bounding box of the signer such as [y0, x0, y1, x1] where (x0, y0) is up-left corner and (x1,y1) is bottom-right corner.
All the values are normalized (between zero and one) according to width and height.
'text': the gloss for this clip which match the 'label',
'width': height for the original video
'height': height for the original video
'fps': frame per second for the original video
}
```

Figure 4-3 The Structure of Sample Json File

With all the required information obtained, a Python code can be written to automatically download the data from YouTube. To achieve this, libraries such as `pytube` and `moviepy` are utilized. Finally, the average number of samples for each sign in the dataset has reached around 80.

4.3 Feature Engineering

4.3.1 Overview

The feature engineering module converts video frames to RGB color space, configures MediaPipe Holistic with default settings for balanced accuracy and performance, and extracts normalized body and hand keypoint landmarks (pose, left hand, right hand) from each video frame. These 258-dimensional landmark arrays are then saved as .npy files in a directory, serving as the input data for training the deep learning model in the subsequent stage.

4.3.2 Requirements

The expected outcome of this module is a directory containing .npy files that represent the extracted body and hand landmarks from video frames. Each .npy file is a 258-dimensional vector encoding the normalized 3D coordinates of 33 pose keypoints, 21 left hand keypoints, and 21 right hand keypoints for a single frame. This preprocessed landmark data serves as the input dataset for training the deep learning model in the subsequent stages of the pipeline.

4.3.3 Implementation Details

The objective of feature engineering module is to extract the landmarks, which refer to the key points detected on human bodies, such as joints and facial features. These landmarks serve as input for training the deep learning model. To achieve this, the video clips stored locally are read one by one using OpenCV for further processing. Since OpenCV reads videos in the BGR colour space by default, the colour space of each frame must be converted to RGB, as many other libraries and models, including MediaPipe Holistic, expect frames in RGB format. Figure 4-5 illustrates the visual difference between BGR (left) and RGB (right).

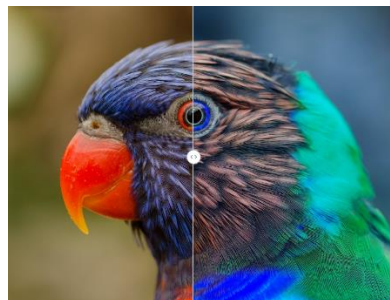


Figure 4-5 The Difference between BGR (Left) and RGB (Right)

Failure to convert the colour format could lead to incorrect colour interpretation and potentially incorrect results in subsequent processing steps, such as landmarks extraction. Afterwards, the configuration options for MediaPipe Holistic are maintained as default. Table 4-1 illustrates these default configuration options.

Table 4-1 Default Configuration Options for MediaPipe Holistic

Configuration Option	Value
static_image_mode	False
model_complexity	1
smooth_landmarks	True
enable_segmentation	False
smooth_segmentation	True
refine_face_landmarks	False
min_detection_confidence	0.5
min_tracking_confidence	0.5

In general, using the default configuration options for MediaPipe Holistic is appropriate for most common use cases, including the current one, where a balance between accuracy and performance is desired. To substantiate this, let's discuss some of the options. With `static_image_mode=False`, MediaPipe Holistic treats the input as a video stream and optimizes for tracking landmarks across frames, reducing computation and latency after the initial detection. Since the current task does not require segmentation masks, keeping `enable_segmentation=False` avoids the additional computation overhead.

Lastly, the keypoints are finally extracted from the video clips on a frame-by-frame basis using MediaPipe Holistic. The output for each frame comprises three components: pose landmarks, left-hand landmarks, and right-hand landmarks. The pose landmarks are a collection of 33 keypoints, with each keypoint having `x`, `y`, `z` coordinates and a visibility value. The `x` and `y` coordinates are normalized between 0.0 and 1.0 based on the image's width and height, respectively. The `z` coordinate denotes the landmark's depth, with the origin set at the wrist, and smaller values indicating proximity to the camera. The `z` magnitude scale is roughly equivalent to `x`. The visibility value ranges

4.4 Data Preparation

4.4.1 Overview

The Data Preparation module aims to standardize and prepare video clip keypoints data for deep learning model training through two main steps: frame sampling and data splitting. In frame sampling, video clips are adjusted to a uniform length of 60 frames to ensure consistent input for the model. This process is automated using provided code, as shown in Figure 4-7. Data splitting involves converting labels to one-hot encoded format and then dividing the dataset for different purposes.

4.4.2 Requirements

The expected outcome of the Data Preparation module is two datasets: one for training (comprising 95% of the data) and one for validation (5% of the data). Both datasets will have video clips standardized to a length of 60 frames and labels in one-hot encoded format, ready for deep learning model training and validation.

4.4.3 Implementation Details

The Data Preparation module emerges as the subsequent topic of discussion, its primary objective being to ensure that the processed samples attain a uniform length, thereby enabling their division into distinct datasets tailored to serve various purposes within the training process. Fundamentally, this module encompasses two pivotal steps: frame sampling and data splitting.

Frame sampling is the step to ensure that all video clips have the same number of frames of keypoints data. Although LSTM models can handle inputs of varying lengths, significant variations in the length of video clips might impact the model's performance. Therefore, before using these keypoints data as input for training the deep learning model, a series of frames must be either truncated or padded to achieve uniformity. The target length for each video clip has been set to 60 frames of data. This value represents the average length of video clips in the dataset. The process of frame sampling is automated using the provided code snippet, as illustrated in Figure 4-7.

CHAPTER 4

```
Set KEYPOINT_DATA_PATH as 'keypoints'
Set T as 60
Set sequence_length as 100

Initialize empty lists: sequences, labels

For each action in actions:
  For each sequence in range from 1 to sequence_length + 1:
    Create an empty list called window
    For each frame_num in the range of frames for the current action and sequence:
      Load the keypoints data for the current action, sequence, and frame_num
      Add the loaded data to the window list

    Convert the window list to a numpy array called array_window

    If the length of array_window is less than T:
      Pad array_window with zeros and truncate it to length T
    Else if the length of array_window is greater than T:
      Sample T frames from array_window and store them in truncated_window
    Else:
      Set truncated_window equal to array_window

    Add truncated_window to the sequences list
    Add the label corresponding to the current action to the labels list
```

Figure 4-7 Pseudocode of Frame Sampling Algorithm

The second step, data splitting, commences with the conversion of labels into a one-hot encoded format. One-hot encoding is a process where categorical variables are converted into a numerical representation, where each category is represented as a binary vector which is a format suitable for training a neural network. For example, if there are only three classes (Bird, Black, Blue), a label of Black would be represented as [0, 1, 0]. After that, the dataset is split into training and testing sets. It takes input features and corresponding labels and splits them into two sets of training data and testing data. In our case, 5% of the data will be reserved for testing, while the rest will be used for training.

4.5 Model Design, Training, and Hyperparameter Tuning

4.5.1 Overview

The module outlines an LSTM neural network tailored for classifying sign language videos into 50 classes. It incorporates BatchNormalization, TimeDistributed Dense, LSTM, Dropout, and Dense layers. Training uses Step Decay Learning Rate Scheduler and Early Stopping, while hyperparameter tuning employs Hyperband and Keras Tuner to find optimal settings, including 'tanh' activation, 1e-2 learning rate, and 'adam' optimizer. This results in a highly effective model for the classification task.

4.5.2 Requirements

In the end, the module delivers an optimized LSTM neural network tailored for classifying sign language video sequences into 50 classes. Hyperparameter tuning via Hyperband and Keras Tuner should identify optimal hyperparameter settings that can make the model to achieve at least 80% of test accuracy.

4.5.3 Implementation Details

The Model Design, Training and Hyperparameter module starts by outlining the Model Design. The architecture is built around the Long Short-Term Memory (LSTM) neural network, known for its effectiveness with sequential data like sign language videos. Figure 4-8 illustrates this model design.

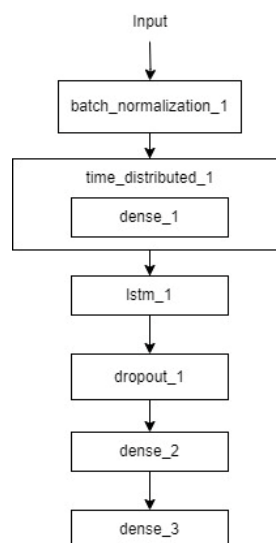


Figure 4-8 Model Architecture

The architecture starts with a BatchNormalization layer, which helps in improving the training efficiency and stability by normalizing the input across batches. This is followed by a TimeDistributed Dense layer with 128 units, which applies a dense layer to each time step of the input sequence individually. This layer acts as a feature extractor, learning relevant representations from the input data. The core of the model is the LSTM layer with 96 units and `return_sequences=False`, which processes the sequential input and captures long-term dependencies within the data. LSTMs are capable of selectively remembering and forgetting information, making them effective for modeling time-series data like sign language sequences. After the LSTM layer, a Dropout layer with a rate of 0.2 is added to prevent overfitting by randomly dropping out a fraction of the units during training. The output from the LSTM layer is then passed through two Dense layers. The first Dense layer has 256 units with a 'tanh' activation function, which introduces non-linearity to the model. The final Dense layer has 50 units with a 'softmax' activation function, which outputs probability scores for each of the 50 classes.

Proceeding to the Model Training phase involves selecting the appropriate training settings for the LSTM model to achieve high test accuracy. Before delving into these settings, it's important to discuss some key deep learning training techniques that have been employed.

One of these techniques is the Learning Scheduler. This method adjusts the learning rate during training based on a predefined schedule or conditions. The type of Learning Scheduler used here is the Step Decay Learning Scheduler. This scheduler reduces the learning rate at specific intervals or epochs. It is chosen for its ability to provide stable convergence, allowing the learning rate to gradually decrease, enabling the model to find an optimal solution. The setting of the learning scheduler used is depicted in Figure 4-9.

```
def step_decay(epoch):  
    initial_lr = 0.001  
    drop = 0.1  
    epochs_drop = 100  
    lr = initial_lr * (drop ** (epoch / epochs_drop))  
    return lr
```

Figure 4-9 Step Decay Learning Rate Scheduler Settings

The second technique used is Early Stopping. It is a regularization technique used to prevent overfitting by stopping the training process once the model's performance stops improving on a held-out validation dataset. In our case, the early stopping mechanism will monitor validation loss during training, and its setting is depicted in Figure 4-10.

```
early_stopping = EarlyStopping(monitor='val_loss',
                               patience=10,
                               restore_best_weights=True)
```

Figure 4-10 Early Stopping Mechanism Settings

The upcoming section will discuss the training settings, tailored to the specific classification task being addressed. Given that the task is multiclass classification, Cross-Entropy Loss has been chosen as the cost function, while Softmax serves as the activation function.

Hyperparameter tuning is a crucial process in optimizing the performance of machine learning models. It involves exploring a vast space of potential hyperparameter configurations to find the optimal set that yields the best results. The Table 4-2 displays the different values of hyperparameters that have been explored. This process can be time-consuming and resource-intensive, depending on the strategies employed for hyperparameter optimization, such as random search, grid search, and others.

Table 4-2 Set of Hyperparameters for Hyperparameter Tuning

Layers	Hyperparameters	Possible Values
	activation	values=['relu', 'tanh']
	learning_rate	values=[1e-2, 1e-3, 1e-4]
	optimizer	values=['adam', 'rmsprop', 'sgd', 'adamax']
	batch_size	values=['8', '16', '32', '64']
dense_1	units	min_value=64,max_value=352,step=32
lstm_1	units	min_value=64,max_value=352,step=32
dropout_1	rate	values=[0.0, 0.3, 0.5]
dense_2	units	min_value=64,max_value=352,step=32

In the given case, the Hyperband tuning strategy is utilized, which adaptively allocates resources to promising hyperparameter configurations, making it efficient for large

search spaces. To achieve this, Keras Tuner, a powerful tool for automating the hyperparameter tuning process, is employed. Keras Tuner supports various types of hyperparameters, including continuous, discrete, and conditional, providing flexibility in model optimization.

Figure 4-11 illustrates the configuration settings for Keras Tuner in this particular scenario. By leveraging Keras Tuner's capabilities, the process of hyperparameter tuning becomes streamlined and efficient, enabling researchers and developers to explore a broad range of hyperparameter combinations and identify the optimal configuration that maximizes the model's performance.

```
tuner = kt.Hyperband(model_builder,
                    objective='val_accuracy',
                    max_epochs=100,
                    factor=3,
                    directory='dir',
                    project_name='x')
```

Figure 4-11 Keras Tuner Configuration Setting

Lastly, Table 4-3 presents the results of hyperparameter tuning, showcasing the optimal set of hyperparameters that yield the best model, achieving over 80% accuracy on the test dataset.

Table 4-3 Optimal Set of Hyperparameters

Layers	Hyperparameters	Best Value
	learning_rate	1e-2
	optimizer	adam
dense_1	units	128
	activation	tanh
lstm_1	units	96
dropout_1	rate	0.3
dense_2	units	256
dense_3	units	50
	optimizer	softmax

4.6 Model Testing and Evaluation

4.6.1 Overview

This module evaluates the performance and generalization of a multiclass sign language recognition model using accuracy, precision, recall, and a confusion matrix. The model shows a strong 95% training accuracy but 85% testing accuracy, indicating good training performance but some generalization challenges. A table introduces class indices for reference due to space constraints in subsequent visuals. The confusion matrix reveals overall strong performance, though some misclassifications occur. Precision scores highlight mostly high accuracy in classifying positive instances, while recall scores generally show high true positive rates, though with some variations possibly due to factors like sign complexity or data limitations.

4.6.2 Requirements

In the end, a comprehensive analysis of the final model using various performance metrics will provide valuable insights. This will help identify areas for further improvement in future work.

4.6.3 Implementation Details

The primary objective of this module is to evaluate the performance and generalization ability of the model. To achieve this, several performance metrics are utilized, such as accuracy, precision, recall, and a multiclass confusion matrix. These metrics provide insights into how well the model performs on unseen or new data, ensuring that overfitting is avoided.

Firstly, when discussing the accuracy of the model, it's crucial to consider two types: training accuracy and testing accuracy. Training accuracy indicates whether the model has the capability to capture the underlying patterns in the data. On the other hand, testing accuracy assesses how well the model generalizes to unseen data. According to the result, the training accuracy is 95%, while the testing accuracy is 85%. These figures suggest that the model performs well on the training data but may have some room for improvement when it comes to generalization.

CHAPTER 4

After that, the Table 4.4 presents the classes alongside their corresponding indices to enhance the comprehension of the subsequent section. This is because the diagram in the following section labels the classes with indices rather their names due to insufficient space.

Table 4-4 List of Classes and Corresponding Indices

Index	Class	Index	Class	Index	Class	Index	Class	Index	Class
0	eat	10	write	20	mother	30	you	40	tired
1	want	11	milk	21	sister	31	me	41	nice
2	finish	12	orange	22	student	32	what	42	sick
3	drink	13	water	23	friend	33	where	43	hurt
4	like	14	table	24	father	34	when	44	white
5	help	15	pencil	25	doctor	35	how	45	blue
6	need	16	fish	26	grandmother	36	yes	46	yellow
7	lost	17	bird	27	grandfather	37	no	47	black
8	learn	18	cousin	28	family	38	hungry	48	red
9	sit	19	teacher	29	brother	39	deaf	49	green

other classes, particularly classes 5 (help), 11 (milk), and 12 (orange). This could point to a challenge in distinguishing class 10 (write) from these other classes.

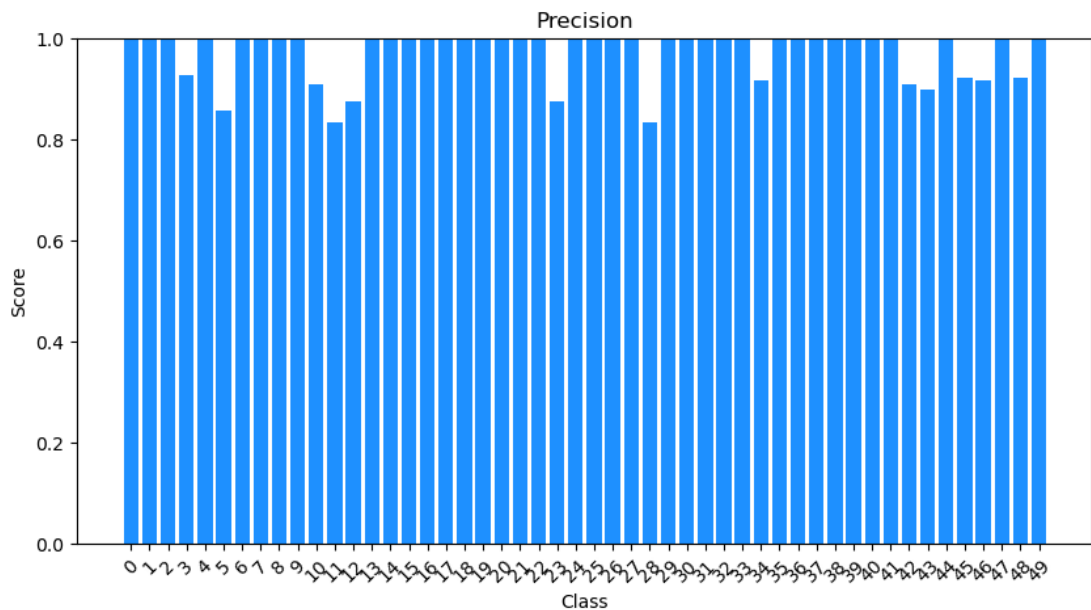


Figure 4-12 Bar Chart for Precision Scores

The Figure 4-12 displays the bar chart that shows the precision scores for all classes in the final model. The x-axis lists the classes, which appear to be numerical labels corresponding to different actions, while the y-axis represents the precision score ranging from 0 to 1. The precision scores for most classes are quite high, with a majority of the bars reaching close to 1.0. However, there are several classes with noticeably lower precision scores, indicated by shorter bars on the chart. A few insights can be extracted from the bar chart. Firstly, the high precision scores for most classes implies a low rate of false positives, which is desirable for an accurate classification system. Secondly, the situation where some classes exhibit lower precision compared to others could be due to factors such as the complexity of certain signs, such as class 11 (milk) and class 23 (friend).

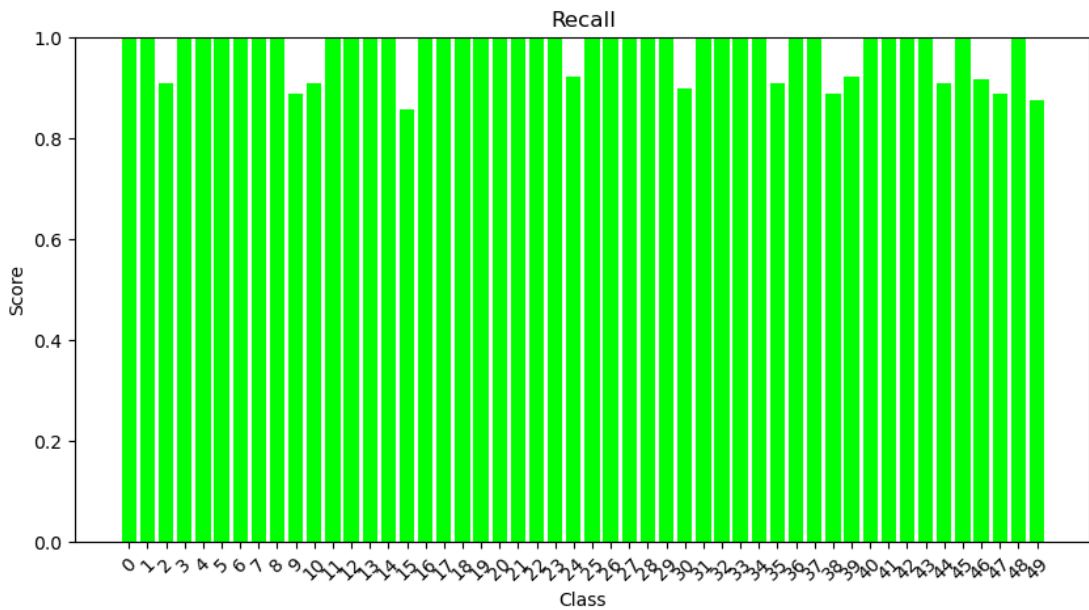


Figure 4-13 Bar Chart for Recall Scores

The Figure 4-13 illustrates the bar chart displaying the recall scores for all classes in the final model. The recall metric measures the model's ability to correctly identify all positive instances of a particular class. Looking at the chart, the recall scores are generally quite high, with most bars reaching close to 1.0. This indicates that the model has a high true positive rate and is successfully capturing the majority of instances for most classes. While the recall is generally high, there are some classes with lower scores. These variations could be due to factors such as class imbalance, complexity or similarity of certain signs, or limitations in the training data or model architecture for those specific classes.

4.7 Application Development and Model Integration

4.7.1 Overview

The final module focuses on developing the front-end user interface, building the back-end system, and integrating the trained model. The front-end, built with Bootstrap 5, features sections for sign language recognition via webcam, accessing the sign dictionary, user login/signup, and form validation. The back end, developed in PHP, integrates with a MySQL database and handles components like transaction history management, account management, and dictionary management. The critical model integration step involves converting the Keras model into the TensorFlowJS Layer format using the TensorFlowJS library, enabling browser-based execution without a server. The converted model files are hosted on GitHub to bypass CORS restrictions. Ultimately, the completed web application is deployed on the 000Webhost platform, making it accessible to users over the internet.

4.7.2 Requirements

The final product is a web application called SignSpeak, it is designed to facilitate interaction with a trained sign language prediction model. Users can access various features such as webcam-based predictions, a sign language dictionary with search capabilities, and user authentication for personalized experiences. Additionally, the application allows users to view labelled sign language videos, manage their accounts, and benefit from a good browsing experience with the model running directly in the browser.

4.7.3 Implementation Details

The Application Development and Model Integration module is the final module to be discussed. It encompasses designing and developing a front-end interface that enables users to interact with the trained model for prediction purposes. Additionally, it involves building a backend system to support the management of the sign language dictionary and user authentication. Lastly, it covers the conversion of the Keras model into the TensorFlowJS Layer format. The following sections will elaborate on these steps in detail.

Firstly, the module starts with the frontend development which focuses on using Bootstrap 5 to design and build interactive user interfaces that allow users to engage

CHAPTER 4

with the model, such as activating the webcam for video frame analysis to make predictions. The application is organized into several key sections for user interaction. The Recognition Section enables users to control the camera, view videos labelled with landmarks, and access recognition results. In addition, this section also supports the prediction of sign language using the trained model in Tensorflow.js format. In the Dictionary Section, a table displays attributes like number, sign name, and video links, providing easy access to sign language resources. Additionally, a search bar in this section allows users to filter and locate specific signs of interest. For user authentication, the Login Section offers a form where users can input their email and password. Account holders can also bookmark frequently accessed signs. Meanwhile, the Signup Section provides forms for new users to register by entering their username, email, and password. Form validation ensures that users input valid information.

Secondly, the next step is backend development. This step utilizes PHP, a backend programming language, to handle the system operations and integration with a MySQL database. Several components have been implemented in order for the application to handle various functionalities. The transaction history management component is implemented to manage and store the history of sign language recognition transactions. The account management component handles user authentication and account management. Lastly, the dictionary component is implemented to manage and update the sign language dictionary used by the application.

The final step of this module is model integration, involving the conversion of the Keras model into the TensorFlowJS Layer format using the TensorFlowJS JavaScript library. This conversion allows the model to be deployed and run in the browser without a backend server. The conversion process generates a directory containing a `model.json` file and binary weight files. The `model.json` file encapsulates the model topology and weight file manifest, enabling direct prediction of sign language actions. However, due to Cross-Origin Resource Sharing (CORS) restrictions, the file cannot be accessed via HTTP requests from the same directory as the accessing code, as it would be considered a cross-origin request. To bypass this limitation, file hosting is implemented by storing the file on GitHub and accessing it via a URL, circumventing the CORS mechanism. Lastly, the completed web application is deployed on 000Webhost platform to making it accessible to users on the internet.

CHAPTER 5 SYSTEM IMPLEMENTATION

This chapter provides a detailed overview of the hardware specifications utilized in the SignSpeak project, focusing on the ASUS TUF Gaming A15 laptop model. The specifications include the processor, operating system, graphics card, memory, storage, and camera, all of which are crucial components for running the sign language recognition system. This chapter sets the foundation for the subsequent sections by outlining the hardware infrastructure necessary for the project's implementation and operation.

5.1 Hardware Setup

- i. Laptop

Description	Specifications
Model	ASUS TUF Gaming A15 – FA506IH
Processor	AMD Ryzen™ 5 4600H Mobile Processor (6-core/12-thread, 11MB Cache, 4.0 GHz max boost)
Operating System	Windows 10 Home
Graphic	NVIDIA® GeForce GTX™ 1650, 4GB GDDR6
Memory	16GB DDR4 RAM
Storage	512GB PCIe® 3.0 NVMe™ M.2 SSD
Camera	720P HD camera

Table 5-1 Specifications of laptop

5.2 Software Setup

Name	Type	Description
Python	Programming Language	Python was chosen as the primary programming language for the project due to its high-level nature, straightforward syntax, and extensive support for machine learning and deep learning through various APIs, libraries, and frameworks. Moreover, the Python community offers substantial support and resources for machine learning and deep learning endeavours.
PHP	Programming Language	A server-side scripting language used for web development to integrate with MySQL database and to handle the back-end operation of the sign language recognition web application.
OpenCV	Python Library	A library that provides a variety of programming functions for image and video processing, object detection, recognition, and other computer vision tasks. It is utilized to convert colour spaces to a format compatible with MediaPipe Holistic.
MediaPipe Holistic	Python Library	A computer vision solution that provides simultaneous detection and extraction of multiple facial landmarks, hand gestures, and body poses in video clips of the datasets.
TensorFlow	Python Library	A machine learning framework developed by Google for building and training the sign language recognition model.
Pytube	Python Library	A library used in the python code to download YouTube videos based on a given URL to build the dataset.
TensorFlowJS	JavaScript Library	A JavaScript library that allows machine learning models, including those from TensorFlow, to run in web browsers.
Bootstrap 5	JavaScript Framework	An open-source front-end framework that provides tools and templates for building responsive and mobile-first websites and web applications.
XAMPP	Software Package	XAMPP is a free and open-source cross-platform software package that provides a local server environment for web development, bundling Apache, MySQL, PHP, and Perl.
000Webhost	Service	A free web hosting service that is used to host the proposed web application. It offers users the ability to host websites without any cost, providing a platform to publish web content online.

Table 5-2 Software Setup

5.3 System Operation

5.3.1 Overview

In an overview, SignSpeak allows users to create accounts with unique email addresses and log in using email and password credentials. After logging in, the username is displayed, granting access to features like saving history. Guests can also log in but without access to history saving. Users can perform sign language actions for recognition and assess their performance in learning mode. A dictionary module lists recognizable actions with a search function, and users can view example videos of actions. Transaction history is displayed in a paginated table, where users can view details and delete records.

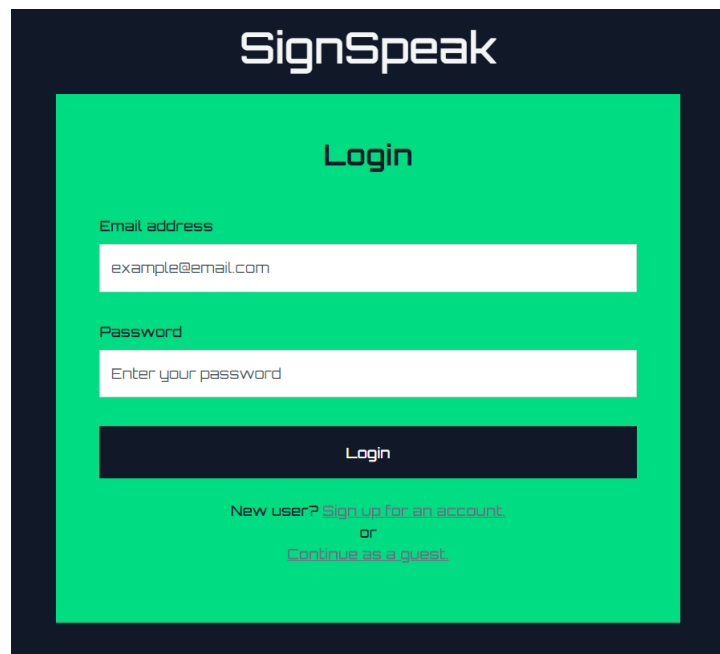
5.3.2 Operations

As shown in Figure 5-1, the users can create an account by entering their name, email address and password before clicking on the ‘Sign Up’ button. However, the email address entered must not be duplicated with the one of any records stored in database.

The image shows a 'Create Account' form with a white background and a dark blue border. The form is titled 'Create Account' in bold black text. It contains three input fields: 'Name' with the placeholder 'Enter your name', 'Email address' with the placeholder 'name@email.com', and 'Password' with the placeholder 'Enter your password'. Below the fields is a dark blue 'Sign Up' button. At the bottom, there is a link: 'Already registered? Login here.'

Figure 5-1 Create Account

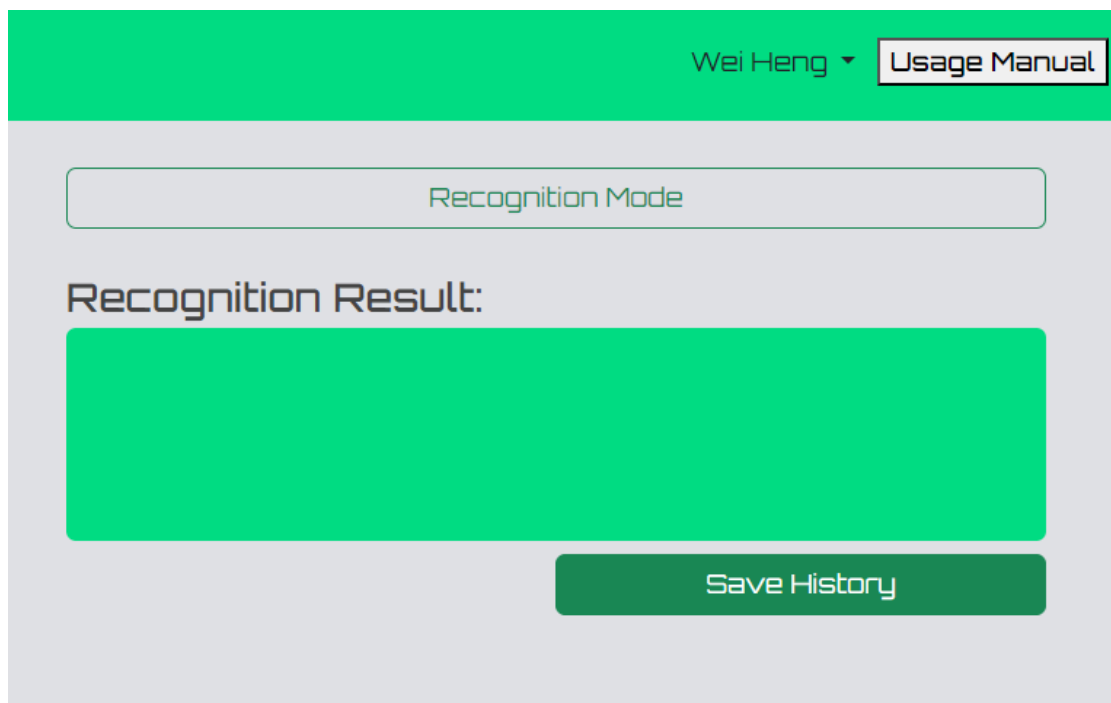
As shown in Figure 5-2, the users can enter their email address and password to login to their account after creating one.



The image shows a login form for SignSpeak. The form is centered on a dark blue background. At the top, the SignSpeak logo is displayed in white. Below the logo, the word "Login" is written in white. The form consists of two input fields: "Email address" with the placeholder text "example@email.com" and "Password" with the placeholder text "Enter your password". Below these fields is a dark blue button labeled "Login". At the bottom of the form, there is a link for "New user? Sign up for an account." and a link for "or Continue as a guest."

Figure 5-2 Login Account

As shown in Figure 5-3, the top right corner of the main page will display the username after the user's login to their account successfully. Users who have an account are eligible to use the save history feature.



The image shows the main page of SignSpeak after a user has logged in. The top navigation bar is dark blue and contains the username "Wei Heng" with a dropdown arrow and a button labeled "Usage Manual". Below the navigation bar, there is a light gray area with a rounded rectangle labeled "Recognition Mode". Below this, the text "Recognition Result:" is displayed, followed by a large dark blue rectangular area. At the bottom right of this area, there is a dark blue button labeled "Save History".

Figure 5-3 Username on Top Right Corner

As shown in Figure 5-4, the user can also login as a guest and the top right corner of the main page will display ‘Account’ as username. However, guest users are not eligible to use the save history feature.

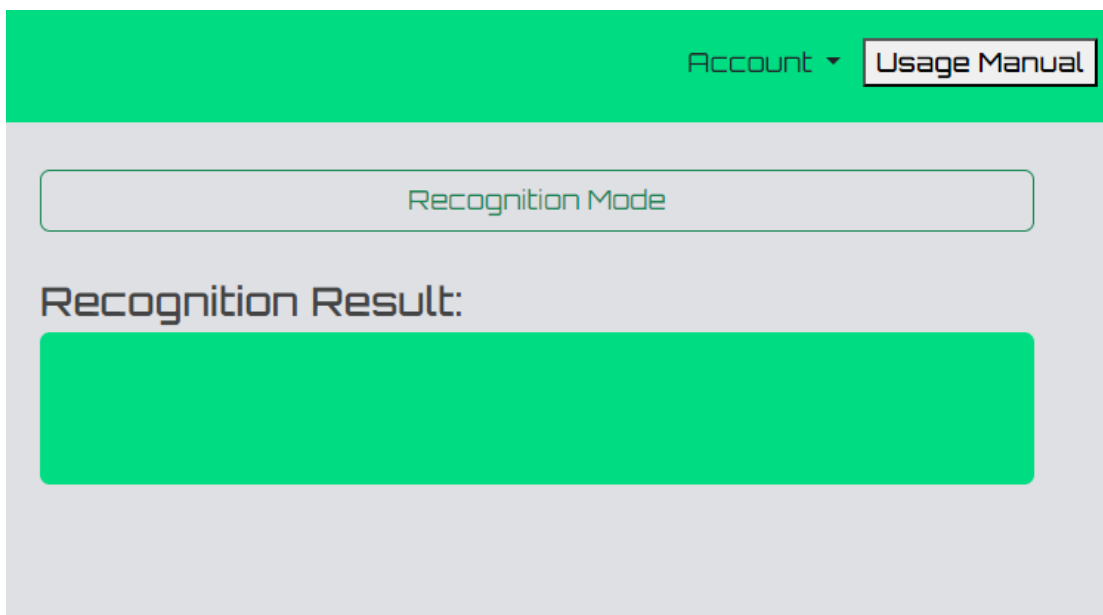


Figure 5-4 Default Username on Top Right Corner

As shown in Figure 5-5, the users can perform sign language actions in front of the camera after clicking on the ‘Start Camera’ button in order to get the recognition result.

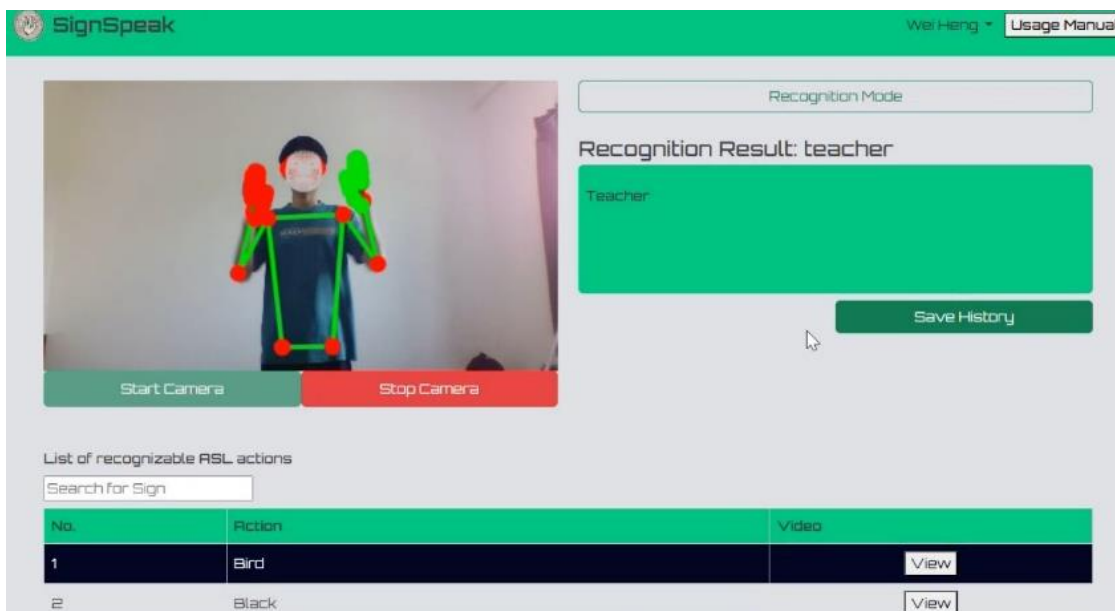


Figure 5-5 Recognition Mode

CHAPTER 5

As shown in Figure 5-6, the users can activate the learning mode by selecting the mode toggle button to get the rating of them in performing certain sign language actions. To switch between different actions, the users can select the available actions from the drop-down list.

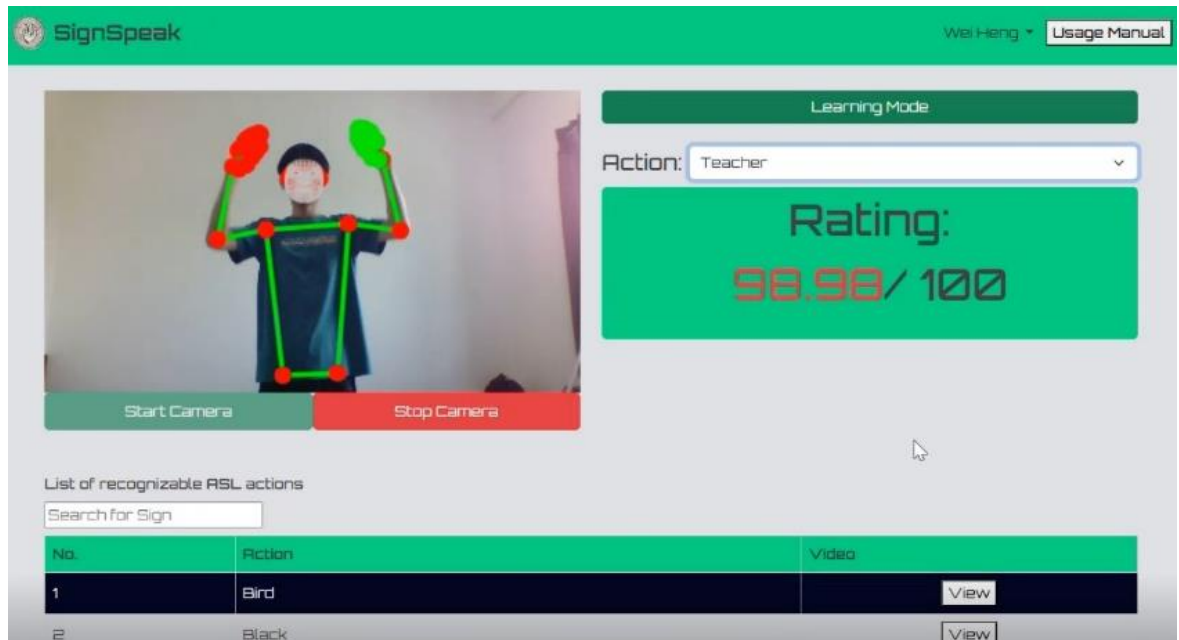


Figure 5-6 Learning Mode

As shown in Figure 5-7, the users can view a list of actions that are recognizable by the system in the dictionary module. The search feature is also provided to allow users to search for a particular action.

List of recognizable ASL actions

No.	Action	Video
1	Bird	View
2	Black	View
3	Blue	View
4	Brother	View
5	Cousin	View
6	Deaf	View
7	Doctor	View
8	Drink	View
9	Eat	View
10	Family	View

Figure 5-7 Dictionary Module with Search Feature

CHAPTER 5

As shown in Figure 5-8, the users can view the example video clips of a particular action in a dialog by clicking on the “View” button.

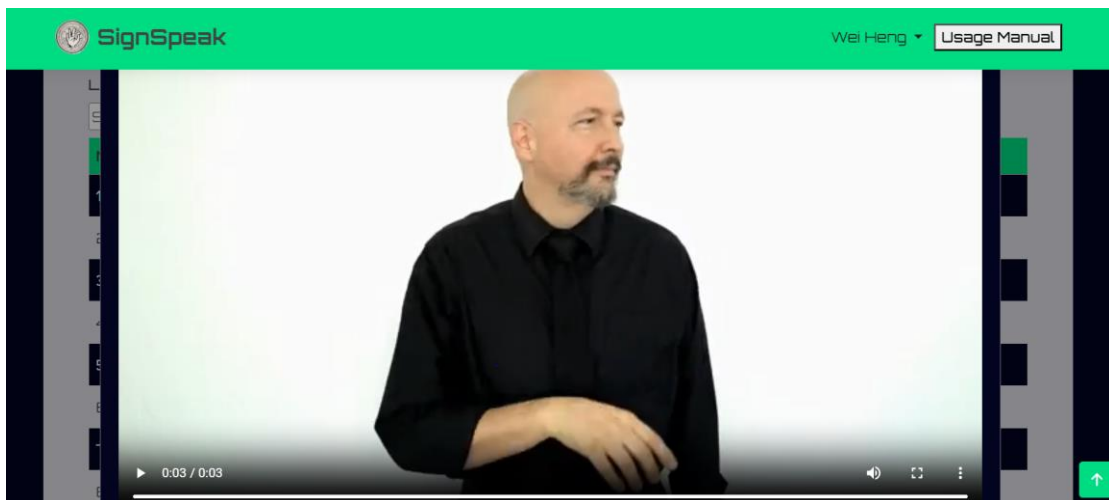
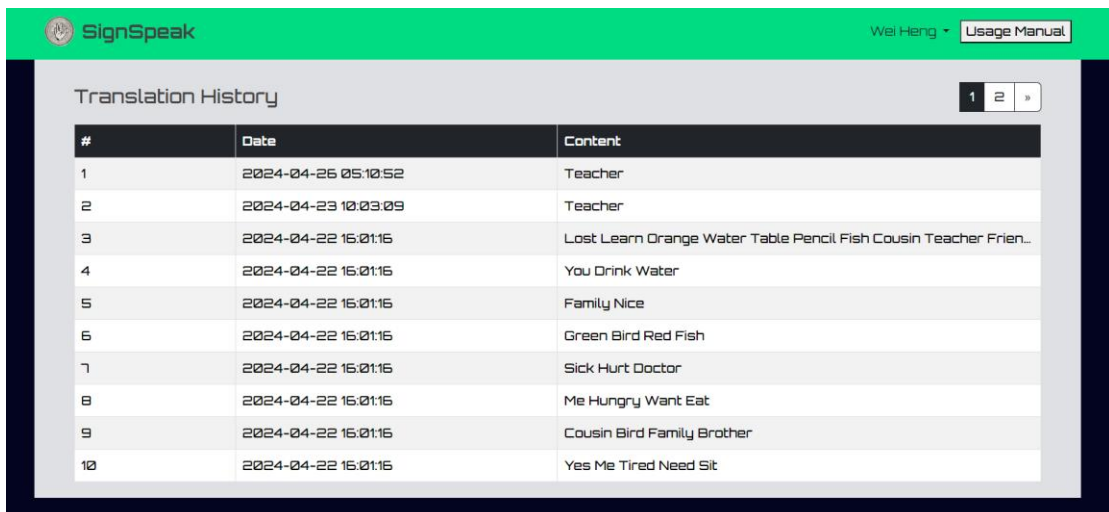


Figure 5-8 Dialog with Video Clip

As shown in Figure 5-9, the users can view the translation history records that they have saved in a table which provides the information such as data and content. The pagination feature also supports in this table to allow the users to view the records page by page.

A screenshot of a web application interface showing a table titled 'Translation History'. The table has three columns: '#', 'Date', and 'Content'. There are 10 rows of data. Above the table, there is a pagination control showing '1 2 >'. The table is styled with a light gray background and dark text.

#	Date	Content
1	2024-04-26 05:10:52	Teacher
2	2024-04-23 10:03:09	Teacher
3	2024-04-22 16:01:16	Lost Learn Orange Water Table Pencil Fish Cousin Teacher Frien...
4	2024-04-22 16:01:16	You Drink Water
5	2024-04-22 16:01:16	Family Nice
6	2024-04-22 16:01:16	Green Bird Red Fish
7	2024-04-22 16:01:16	Sick Hurt Doctor
8	2024-04-22 16:01:16	Me Hungry Want Eat
9	2024-04-22 16:01:16	Cousin Bird Family Brother
10	2024-04-22 16:01:16	Yes Me Tired Need Sit

Figure 5-9 Transaction History Table with Pagination Feature

As shown in Figure 5-10, the users can view the details of a record by clicking row of the records. Also, the users can delete the record by clicking on the ‘Delete’ button.



Figure 5-10 Dialog Showing Record Details and Delete Button

5.4 Implementation Issues and Challenges

5.4.1 Overview

During the system implementation phase, various critical challenges arose at different stages. These challenges have the potential to lead to issues like subpar system performance and user dissatisfaction. Nonetheless, effective solutions have been proposed to address these challenges, which will be elaborated upon in the subsequent section.

5.4.2 Issues of MS-ASL Dataset

One of the critical challenges in developing an effective sign language recognition model lies in the data collection process, specifically when dealing with the MS-ASL dataset. The issues associated with this dataset can have significant implications on the model's performance and accuracy.

In the dataset, some samples showcase the signer performing multiple sign actions in succession rather than just a single sign. Moreover, a minority of samples depict the signer repeating the same sign action multiple times. Most concerning is that certain video samples have been inaccurately labelled with incorrect sign annotations. These issues contribute to a dataset of low quality, deviating from the assumption that each clip should depict a single instance of a signer performing a single sign. Training a model on such flawed data, where signs are either compounded, repeated, or mislabelled, can result in poor performance and inaccuracies in sign recognition and interpretation.

A proposed solution is to manually review every video clip, trim out unwanted portions with multiple or repeated signs, and discard any samples with wrong annotations.

However, this manual curation process is extremely labour-intensive and time-consuming, requiring significant human effort and resources.

In essence, the data collection phase for the MS-ASL dataset presents major challenges due to the inconsistencies and errors present, which can undermine the model's training. While manual cleaning can enhance data quality, it comes at a high cost in terms of the time and resources required for such an undertaking.

5.4.3 Issues of Dataset

The second significant challenge in training the model was dealing with the varying lengths of time steps in the video samples. Time steps refer to the number of frames that the video samples contain. While it is reasonable to provide the LSTM model with input sequences of different time steps by catering them in different batches, as the model is design to handle such variations, the difference in the number of frames across video samples was substantial, with some samples having up to 100 frames more than others. And it can lead to two major consequences.

Firstly, there is a difficulty in learning temporal patterns. LSTM models are specifically designed to capture temporal dependencies in sequences. However, with a substantial difference in time steps, the model can struggle to effectively learn these dependencies. Shorter sequences may end up being forgotten or overshadowed by longer sequences during training, which can result in a suboptimal representation of the temporal patterns. Second, there's a risk of overfitting. The model might be overfit to the longer sequences, giving them undue weight while underfitting or even ignoring the shorter ones. This imbalance can lead to a biased model that performs well on certain sequence lengths but poorly on others, compromising its generalization capability.

To address these issues, a data preparation step called frame sampling was introduced. Frame sampling allows for truncating or padding the video samples to a desired length, thereby achieving uniformity in the input sequences. This ensures that the model receives a more balanced representation of sequences during training, enabling it to better capture the temporal patterns and prevent overfitting or underfitting to specific sequence lengths.

By implementing frame sampling, the model can be trained on a consistent number of frames across all video samples. This not only helps in learning the temporal

dependencies more effectively but also improves the overall performance and generalization capabilities of the sign language recognition model.

5.5 Concluding Remark

In summary, this project encountered challenges with data quality and model compatibility. Fixing dataset issues required extensive manual work, highlighting the importance of good data gathering. The model faced difficulties with varying video lengths, prompting the addition of frame sampling for consistent learning. Despite these hurdles, the implemented solutions demonstrated the system's resilience. Additionally, successful setup of hardware, software, and a user-friendly web app was achieved. These challenges provide valuable insights for improving sign language technology in the future.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

This chapter outlines a comprehensive set of test cases designed to evaluate various aspects of the SignSpeak system. These test cases cover key functionalities such as human body capture via camera, landmarks extraction using MediaPipe Holistic, American Sign Language (ASL) recognition model performance, the SignSpeak web application, account management module, recognition module, dictionary module, and transaction history management module. Each test case specifies the expected results, ensuring thorough evaluation of the system's functionality and performance. Additionally, this chapter provides insights into the testing setup and results, demonstrating the reliability and effectiveness of the system across different scenarios. Despite challenges such as limitations in mobile application development and computational constraints on mobile devices, the SignSpeak system successfully achieves its defined objectives, paving the way for more accessible and inclusive communication tools for the deaf and hard-of-hearing communities.

6.1 System Verification Plan

In this section, test cases and their expected results are presented to evaluate all aspects of SignSpeak. These test cases were identified during system development and reviewed during the implementation stage.

6.1.1 Human Body Capture via Camera

Table 6-1 Test Cases for Evaluating Human Body Capture

ID	Test Case	Expected Result
1	Method to capture frame is executed	Captured video frame is displayed as video stream on the window.
2	User stands in front of the camera	User is captured inside the video frame.
3	User stands outside the range of the camera	User is not captured inside the video frame.

6.1.2 Landmarks Extraction via MediaPipe Holistic

Table 6-2 Test Cases for Evaluating Landmarks Extraction

ID	Test Case	Expected Result
1	Video frame showing a full human body is passed to the landmarker	<ul style="list-style-type: none"> • Full body landmarks are drawn on the canvas. • No landmarks for any body components are empty.
2	Video frame showing only face and pose is passed to the landmarker	<ul style="list-style-type: none"> • Face and pose landmarks are drawn on the canvas. • Left and right hands' landmarks are empty.
3	Video frame showing only left hand is passed to the landmarker	<ul style="list-style-type: none"> • Left hand's landmarks are drawn on the canvas. • Landmarks of face, pose, and right hand are empty.
4	Video frame showing only right hand is passed to the landmarker	<ul style="list-style-type: none"> • Right hand's landmarks are drawn on the canvas. • Landmarks of face, pose, and left hand are empty.
5	Video frame showing no human body is passed to the landmarker	<ul style="list-style-type: none"> • No landmarks for any body components are drawn on the canvas. • Full body landmarks are empty.

6.1.3 American Sign Language Recognition Model

Table 6-3 Test Cases for Evaluating Landmarks Extraction

ID	Test Case	Expected Result
1	Signing 'teacher' from a frontal angle	'teacher' is one of the classes that the model can predict. The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
2	Signing 'teacher' from a side angle	The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
3	Signing 'teacher' in low light	The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
4.	Signing 'air' from frontal angle	'air' is one of the actions that the model cannot predict. The model outputs a probability distribution where the values at all indexes are lower than 0.8.

6.1.4 SignSpeak Web Application

Table 6-4 Test Cases for Evaluating SignSpeak Web Application

ID	Test Case	Expected Result
1	User enters the URL of SignSpeak into a browser	The Login page of SignSpeak is rendered.
2	User enters the URL of SignSpeak into a browser after logging into an account	The Main page of SignSpeak is rendered.

6.1.5 Account Management Module

Table 6-5 Test Cases for Evaluating Account Management Module

ID	Test Case	Expected Result
1	User selects 'Sign Up' button after entering all required information	An account is created, and the user is directed to the Main page.
2	User selects 'Sign Up' button after entering an email that is already duplicated in the database	A message is displayed informing the user that the email already exists.
3	User selects 'Login' button after enters a combination of email and password that matches a record in the user table of the database.	The user is directed to the main page of his account.
4.	User enters a combination of email and password that does not match a record in the user table of the database.	A message is displayed informing the user that the combination is incorrect.

6.1.6 Recognition Module

Table 6-6 Test Cases for Evaluating Recognition Module

ID	Test Case	Expected Result
1	User selects the 'Start Camera' button	Captured video frame is displayed as video stream on the canvas.
2	User selects the 'Stop Camera' button	The canvas turns black.
3	User selects the 'Recognition Mode' toggle button	The learning mode is activated, and the 'bird' action is selected as the default sign currently being rated. The rating is also displayed.

6.1.7 Dictionary Module*Table 6-7 Test Cases for Evaluating Dictionary Module*

ID	Test Case	Expected Result
1	User selects the 'View' button in the table row for 'teacher'	Video clip of signing 'teacher' is displayed in a dialog.
2	User enters 't' as the search query in the search bar.	The history records with the sign name that matches 't' are displayed.
3	User enters '1' as the search query in the search bar.	No history records are displayed.
4.	User enters 'air' as the search query in the search bar.	No history records are displayed.

6.1.8 Transaction History Management Module*Table 6-8 Test Cases for Evaluating Transaction History Management Module*

ID	Test Case	Expected Result
1	User clicks on 'Save Translation History' button	The history record is displayed as a record of the translation history table.
2	User has more than 10 history records	The history records are displayed in the table, with 10 records per page.
3	User selects the 'Next' pagination button.	The history records of the next page are displayed in the table.
4.	User selects the first record of the translation history table.	The details of the history record are displayed in a dialog, along with a 'Delete' button.
5.	User selects the 'Delete' button in the dialog of first history record.	The first record is deleted from the database.

6.2 Testing Setup and Result

In this section, some of the important test cases will be discussed to ensure the reliability of the system.

6.2.1 Human Body Capture via Camera

Table 6-9 Test Case for Testing Frame Capturing Method

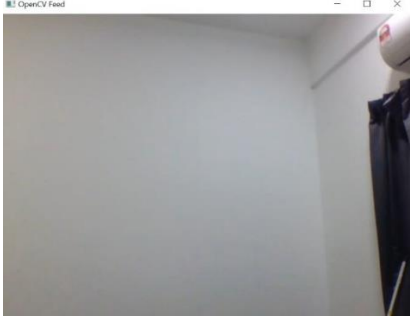
Test Case Name	Testing frame capturing method
Test Case Description	Method to capture frame is executed
Expected Output	Captured video frame is displayed as video stream on the window.
Output	
Result (Pass/Fail)	Pass

Table 6-10 Test Case for Testing Frame Capturing Method in Capturing Human Body


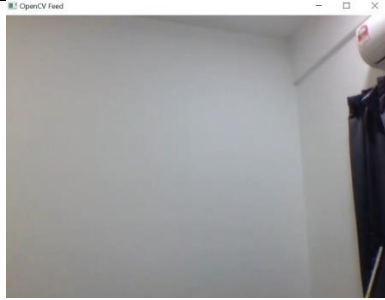
Test Case Name	Testing frame capturing method in capturing human body
Test Case Description	User stands in front of the camera
Expected Output	User is captured inside the video frame
Output	
Result (Pass/Fail)	Pass

Table 6-11 Test Case for Testing Frame Capturing Method in not Capturing Human Body

Test Case Name	Testing frame capturing method in not capturing human body
Test Case Description	User stands outside of the range of the camera
Expected Output	User is not captured inside the video frame
Output	
Result (Pass/Fail)	Pass

6.2.2 Landmarks Extraction via MediaPipe Holistic

Table 6-12 Test Case for Testing Landmarker's Ability in Detecting Holistic Keypoints

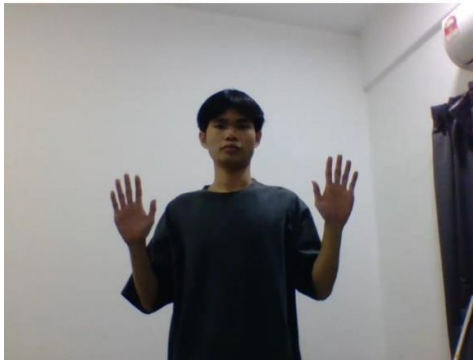
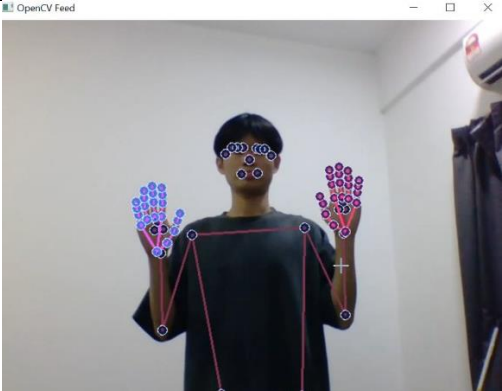
Test Case Name	Testing the landmarker's ability in detecting holistic keypoints
Test Case Description	Video frame showing a full human body is passed to the landmarker
Expected Output	<ul style="list-style-type: none"> • Full body landmarks are drawn on the canvas. • No landmarks for any body components are empty.
Input	 <p>A video frame from an OpenCV Feed showing a person standing in a room with their hands raised. The person is wearing a dark t-shirt. The background is a plain white wall.</p>
Output	 <p>The same video frame as the input, but with holistic keypoints detected. The keypoints are represented by small circles in various colors (blue, purple, red, green) connected by lines. The keypoints are located on the person's face, hands, and torso.</p>
Result (Pass/Fail)	Pass

Table 6-13 Test Case for Testing Landmarker's Ability in Detecting Face and Pose Keypoints


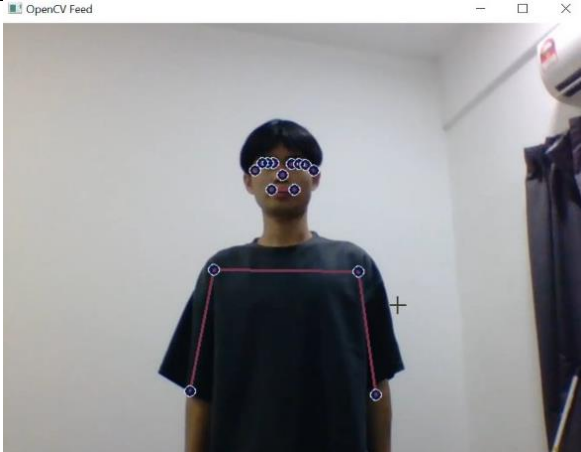
Test Case Name	Testing the landmarker's ability in detecting face and pose keypoints
Test Case Description	Video frame showing only face and pose is passed to the landmarker
Expected Output	<ul style="list-style-type: none"> • Face and pose landmarks are drawn on the canvas. • Left and right hands' landmarks are empty.
Input	 <p>The input image shows a person standing in a room, centered in the frame. The person is wearing a dark t-shirt. The background is a plain wall with a door visible on the right side.</p>
Output	 <p>The output image shows the same person as the input, but with several keypoints detected. The face is marked with a set of blue dots connected by lines, representing facial landmarks. The pose is marked with a red rectangle around the torso and shoulders, with blue dots at the corners, representing body keypoints. There are no markers on the hands.</p>
Result (Pass/Fail)	Pass

Table 6-14 Test Case for Testing Landmarker's Ability in Detecting Left Hand Keypoints

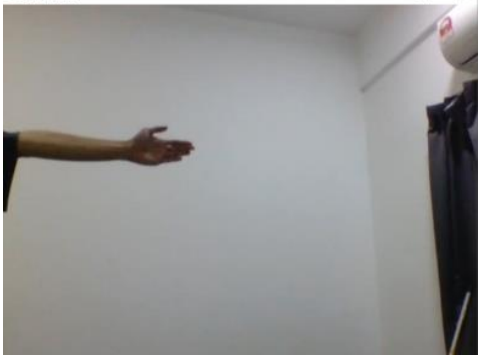

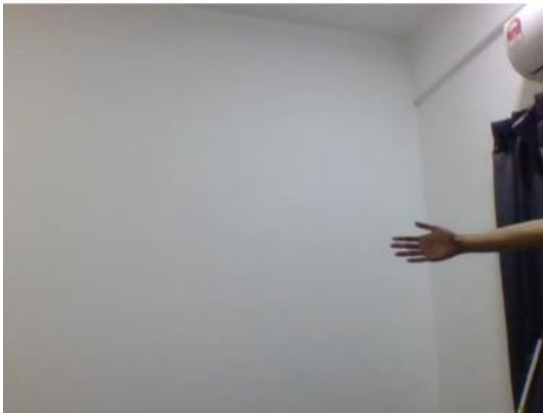
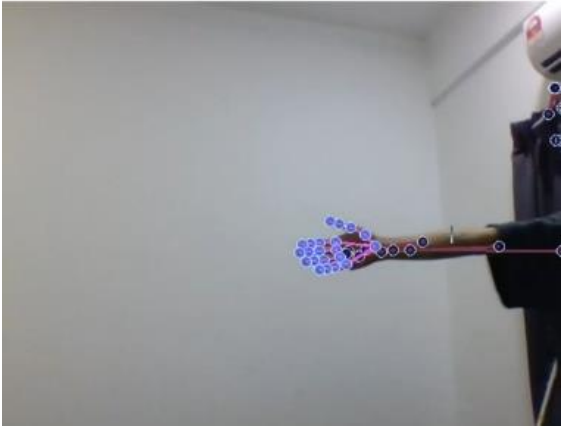
Test Case Name	Testing the landmarker's ability in detecting left hand keypoints
Test Case Description	Video frame showing only left hand is passed to the landmarker
Expected Output	Left hand's landmarks are drawn on the canvas. Landmarks of face, pose, and right hand are empty.
Input	
Output	
Result (Pass/Fail)	Pass

Table 6-15 Test Case for Testing Landmarker's Ability in Detecting Right Hand Keypoints

Test Case Name	Testing the landmarker's ability in detecting right hand landmarks
Test Case Description	Video frame showing only right hand is passed to the landmarker
Expected Output	Right hand's landmarks are drawn on the canvas. Landmarks of face, pose, and left hand are empty.
Input	 <p>The input image shows a person's right hand extended in a room with white walls. The hand is the only object of interest for the landmark detection process.</p>
Output	 <p>The output image shows the same right hand with several blue circular keypoints overlaid on it, representing the detected landmarks. The keypoints are concentrated on the hand and wrist, while the rest of the scene remains unprocessed.</p>
Result (Pass/Fail)	Pass

6.2.3 American Sign Language Recognition Model

Table 6-16 Test Case for Testing System's Ability to Recognize from Front

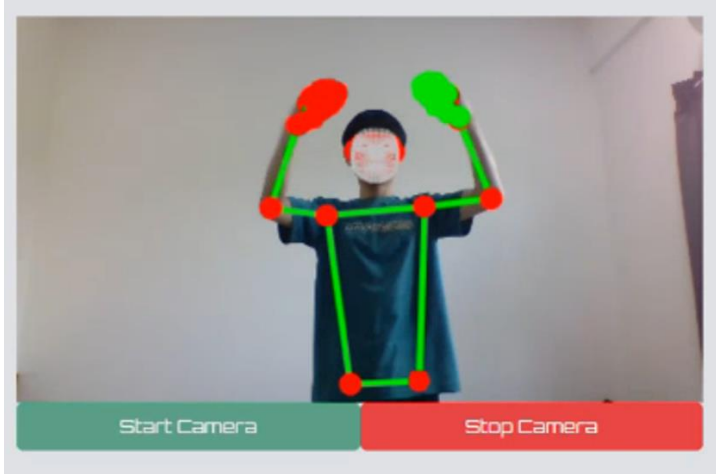
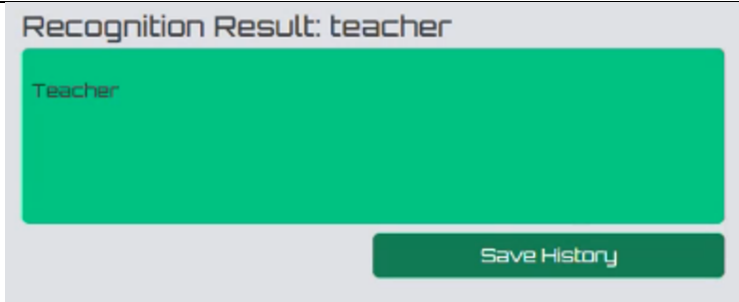
Test Case Name	Testing the system's ability to recognize from the front
Test Case Description	Signing 'teacher' from a frontal angle
Expected Output	'teacher' is one of the classes that the model can predict. The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
Input	
Output	
Result (Pass/Fail)	Pass

Table 6-17 Test Case for Testing System's Ability to Recognize from Side

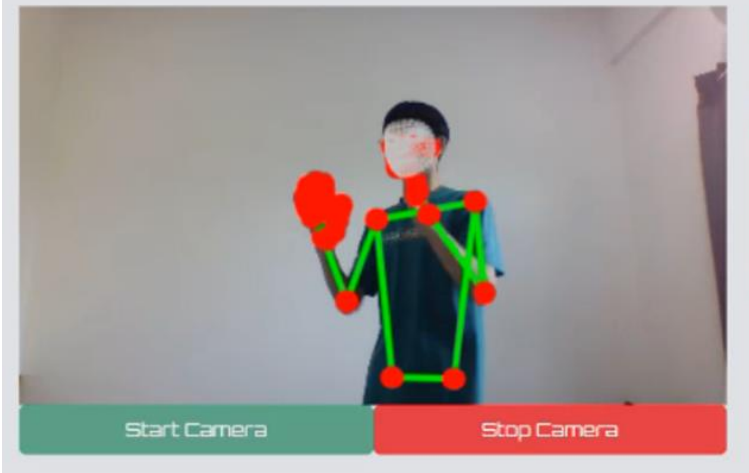
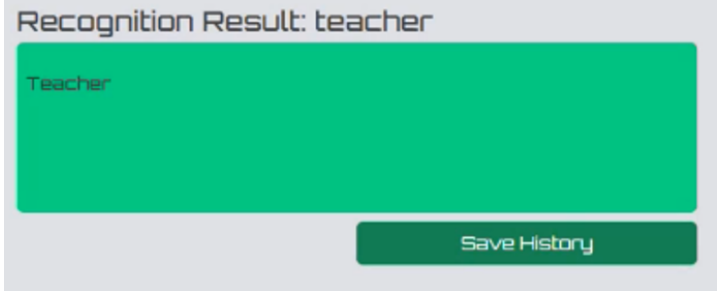
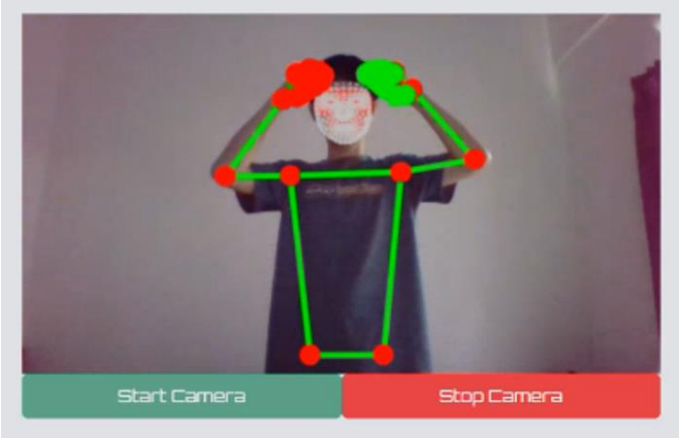
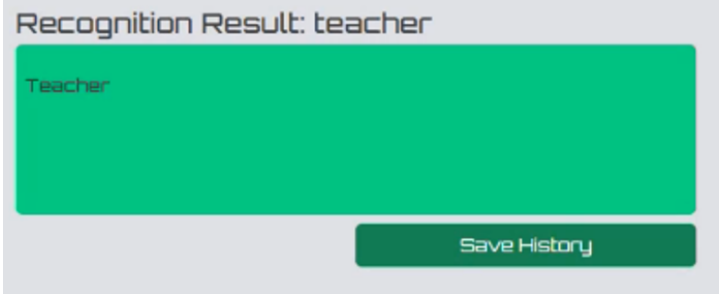
Test Case Name	Testing the system's ability to recognize from the side
Test Case Description	Signing 'teacher' from a side angle
Expected Output	'teacher' is one of the classes that the model can predict. The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
Input	
Output	
Result (Pass/Fail)	Pass

Table 6-18 Test Case for Testing System's Ability to Recognize in Low Light

Test Case Name	Testing the system's ability to recognize in low light
Test Case Description	Signing 'teacher' in low light
Expected Output	The model outputs a probability distribution where the value at the index corresponding to 'teacher' is the highest.
Input	
Output	
Result (Pass/Fail)	Pass

6.2.4 Account Management Module

Table 6-19 Test Case for Testing the Input Validation Mechanism

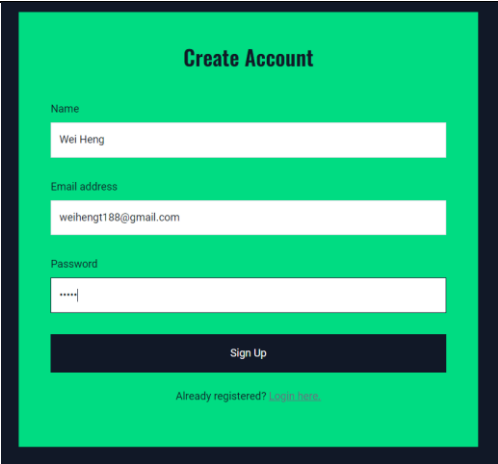
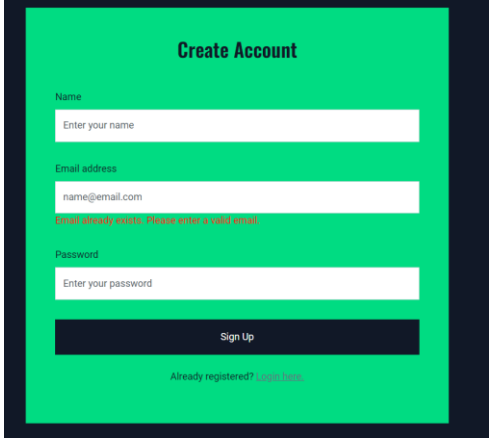
Test Case Name	Testing the input validation mechanism	
Test Case Description	User selects the 'Sign Up' button after entering an email that is already duplicated in the database	
Expected Output	A message is displayed informing the user that the email already exists.	
Input		
Output		
Result (Pass/Fail)	Pass	

Table 6-20 Test Case for Testing User Authentication Mechanism

Test Case Name	Testing the user authentication mechanism	
Test Case Description	User selects 'Login' after enters a combination of email and password that does not match a record in the user table of the database.	
Expected Output	A message is displayed informing the user that the combination is incorrect.	
Input		
Output		
Result (Pass/Fail)	Pass	

6.2.5 Dictionary Module

Table 6-21 Test Case for Testing the Searching Feature when Dealing with Alphabetical Query Term

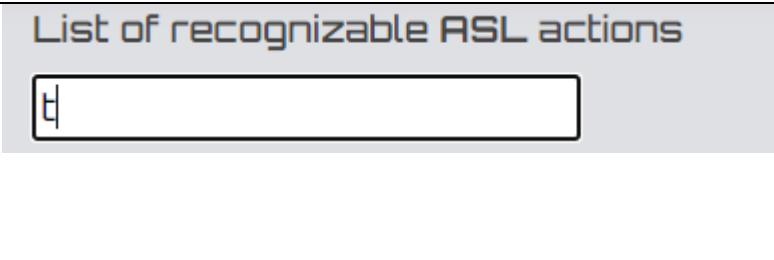
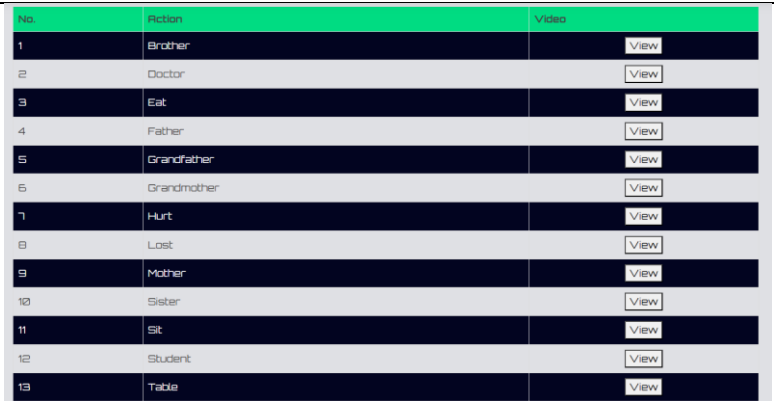
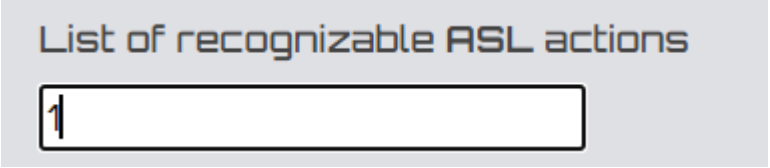
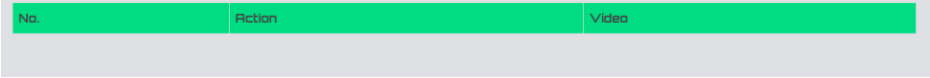
Test Case Name	Testing the searching feature when dealing with alphabetical query term																																										
Test Case Description	User enters 't' as the search query in the search bar.																																										
Expected Output	The history records with the sign name that matches 't' are displayed.																																										
Input																																											
Output	 <table border="1"> <thead> <tr> <th>No.</th> <th>Action</th> <th>Video</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Brother</td> <td>View</td> </tr> <tr> <td>2</td> <td>Doctor</td> <td>View</td> </tr> <tr> <td>3</td> <td>Eat</td> <td>View</td> </tr> <tr> <td>4</td> <td>Father</td> <td>View</td> </tr> <tr> <td>5</td> <td>Grandfather</td> <td>View</td> </tr> <tr> <td>6</td> <td>Grandmother</td> <td>View</td> </tr> <tr> <td>7</td> <td>Hurt</td> <td>View</td> </tr> <tr> <td>8</td> <td>Lost</td> <td>View</td> </tr> <tr> <td>9</td> <td>Mother</td> <td>View</td> </tr> <tr> <td>10</td> <td>Sister</td> <td>View</td> </tr> <tr> <td>11</td> <td>Sit</td> <td>View</td> </tr> <tr> <td>12</td> <td>Student</td> <td>View</td> </tr> <tr> <td>13</td> <td>Table</td> <td>View</td> </tr> </tbody> </table>	No.	Action	Video	1	Brother	View	2	Doctor	View	3	Eat	View	4	Father	View	5	Grandfather	View	6	Grandmother	View	7	Hurt	View	8	Lost	View	9	Mother	View	10	Sister	View	11	Sit	View	12	Student	View	13	Table	View
No.	Action	Video																																									
1	Brother	View																																									
2	Doctor	View																																									
3	Eat	View																																									
4	Father	View																																									
5	Grandfather	View																																									
6	Grandmother	View																																									
7	Hurt	View																																									
8	Lost	View																																									
9	Mother	View																																									
10	Sister	View																																									
11	Sit	View																																									
12	Student	View																																									
13	Table	View																																									
Result (Pass/Fail)	Pass																																										

Table 6-22 Test Case for Testing the Searching Feature when Dealing with Numerical Query Term

Test Case Name	Testing the searching feature when dealing with numerical query term
Test Case Description	User enters '1' as the search query in the search bar.
Expected Output	No history records are displayed.
Input	
Output	
Result (Pass/Fail)	Pass

6.2.6 Transaction History Management Module

Table 6-23 Test Case for Testing Pagination Feature

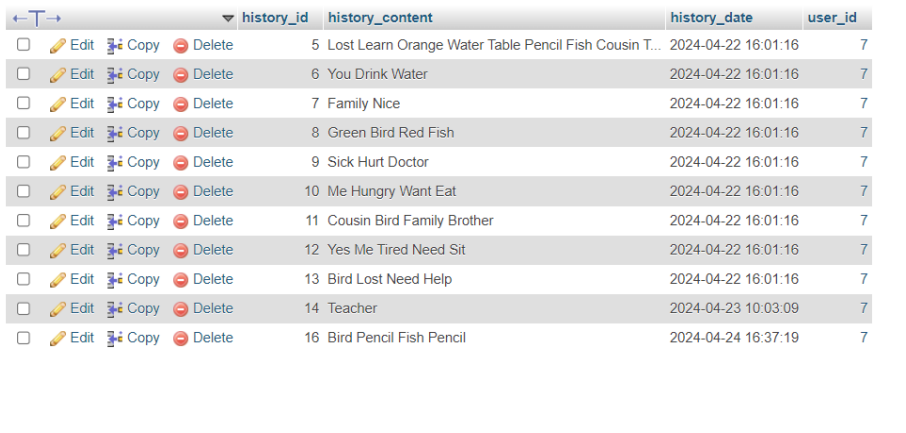
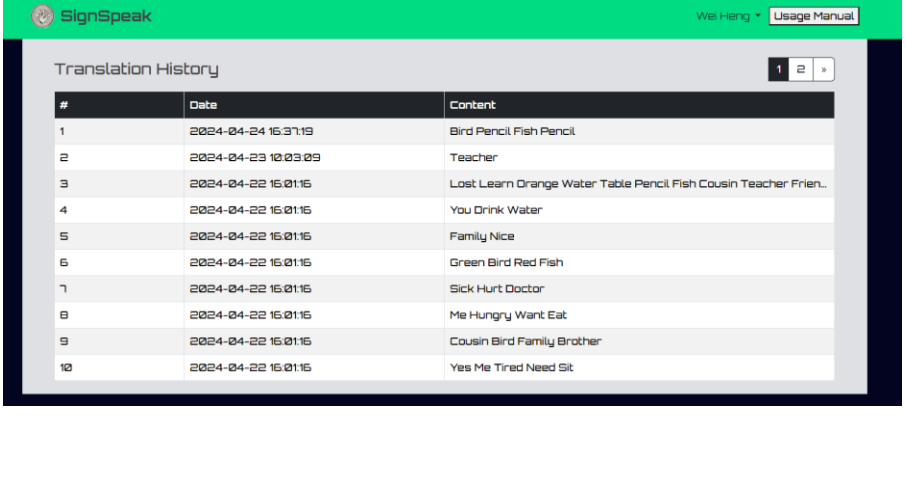
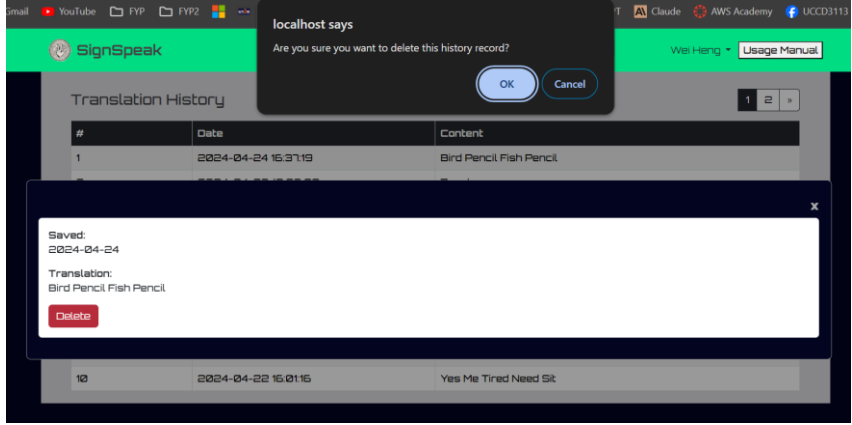
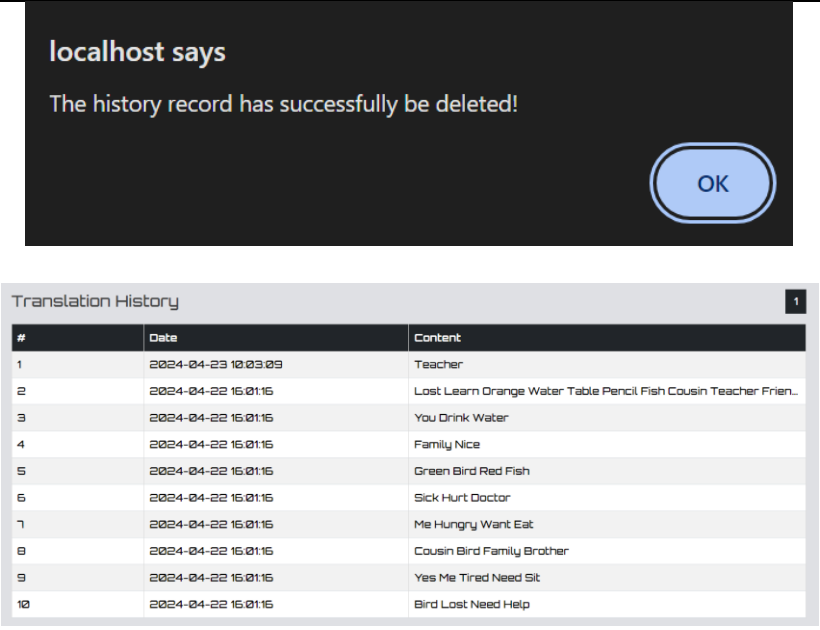
Test Case Name	Testing pagination feature																																																
Test Case Description	User has more than 10 history records																																																
Expected Output	The history records are displayed in the table, with 10 records per page.																																																
Input	 <table border="1"> <thead> <tr> <th>history_id</th> <th>history_content</th> <th>history_date</th> <th>user_id</th> </tr> </thead> <tbody> <tr><td>5</td><td>Lost Learn Orange Water Table Pencil Fish Cousin T...</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>6</td><td>You Drink Water</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>7</td><td>Family Nice</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>8</td><td>Green Bird Red Fish</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>9</td><td>Sick Hurt Doctor</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>10</td><td>Me Hungry Want Eat</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>11</td><td>Cousin Bird Family Brother</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>12</td><td>Yes Me Tired Need Sit</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>13</td><td>Bird Lost Need Help</td><td>2024-04-22 16:01:16</td><td>7</td></tr> <tr><td>14</td><td>Teacher</td><td>2024-04-23 10:03:09</td><td>7</td></tr> <tr><td>16</td><td>Bird Pencil Fish Pencil</td><td>2024-04-24 16:37:19</td><td>7</td></tr> </tbody> </table>	history_id	history_content	history_date	user_id	5	Lost Learn Orange Water Table Pencil Fish Cousin T...	2024-04-22 16:01:16	7	6	You Drink Water	2024-04-22 16:01:16	7	7	Family Nice	2024-04-22 16:01:16	7	8	Green Bird Red Fish	2024-04-22 16:01:16	7	9	Sick Hurt Doctor	2024-04-22 16:01:16	7	10	Me Hungry Want Eat	2024-04-22 16:01:16	7	11	Cousin Bird Family Brother	2024-04-22 16:01:16	7	12	Yes Me Tired Need Sit	2024-04-22 16:01:16	7	13	Bird Lost Need Help	2024-04-22 16:01:16	7	14	Teacher	2024-04-23 10:03:09	7	16	Bird Pencil Fish Pencil	2024-04-24 16:37:19	7
history_id	history_content	history_date	user_id																																														
5	Lost Learn Orange Water Table Pencil Fish Cousin T...	2024-04-22 16:01:16	7																																														
6	You Drink Water	2024-04-22 16:01:16	7																																														
7	Family Nice	2024-04-22 16:01:16	7																																														
8	Green Bird Red Fish	2024-04-22 16:01:16	7																																														
9	Sick Hurt Doctor	2024-04-22 16:01:16	7																																														
10	Me Hungry Want Eat	2024-04-22 16:01:16	7																																														
11	Cousin Bird Family Brother	2024-04-22 16:01:16	7																																														
12	Yes Me Tired Need Sit	2024-04-22 16:01:16	7																																														
13	Bird Lost Need Help	2024-04-22 16:01:16	7																																														
14	Teacher	2024-04-23 10:03:09	7																																														
16	Bird Pencil Fish Pencil	2024-04-24 16:37:19	7																																														
Output	 <table border="1"> <thead> <tr> <th>#</th> <th>Date</th> <th>Content</th> </tr> </thead> <tbody> <tr><td>1</td><td>2024-04-24 16:37:19</td><td>Bird Pencil Fish Pencil</td></tr> <tr><td>2</td><td>2024-04-23 10:03:09</td><td>Teacher</td></tr> <tr><td>3</td><td>2024-04-22 16:01:16</td><td>Lost Learn Orange Water Table Pencil Fish Cousin Teacher Frien...</td></tr> <tr><td>4</td><td>2024-04-22 16:01:16</td><td>You Drink Water</td></tr> <tr><td>5</td><td>2024-04-22 16:01:16</td><td>Family Nice</td></tr> <tr><td>6</td><td>2024-04-22 16:01:16</td><td>Green Bird Red Fish</td></tr> <tr><td>7</td><td>2024-04-22 16:01:16</td><td>Sick Hurt Doctor</td></tr> <tr><td>8</td><td>2024-04-22 16:01:16</td><td>Me Hungry Want Eat</td></tr> <tr><td>9</td><td>2024-04-22 16:01:16</td><td>Cousin Bird Family Brother</td></tr> <tr><td>10</td><td>2024-04-22 16:01:16</td><td>Yes Me Tired Need Sit</td></tr> </tbody> </table>	#	Date	Content	1	2024-04-24 16:37:19	Bird Pencil Fish Pencil	2	2024-04-23 10:03:09	Teacher	3	2024-04-22 16:01:16	Lost Learn Orange Water Table Pencil Fish Cousin Teacher Frien...	4	2024-04-22 16:01:16	You Drink Water	5	2024-04-22 16:01:16	Family Nice	6	2024-04-22 16:01:16	Green Bird Red Fish	7	2024-04-22 16:01:16	Sick Hurt Doctor	8	2024-04-22 16:01:16	Me Hungry Want Eat	9	2024-04-22 16:01:16	Cousin Bird Family Brother	10	2024-04-22 16:01:16	Yes Me Tired Need Sit															
#	Date	Content																																															
1	2024-04-24 16:37:19	Bird Pencil Fish Pencil																																															
2	2024-04-23 10:03:09	Teacher																																															
3	2024-04-22 16:01:16	Lost Learn Orange Water Table Pencil Fish Cousin Teacher Frien...																																															
4	2024-04-22 16:01:16	You Drink Water																																															
5	2024-04-22 16:01:16	Family Nice																																															
6	2024-04-22 16:01:16	Green Bird Red Fish																																															
7	2024-04-22 16:01:16	Sick Hurt Doctor																																															
8	2024-04-22 16:01:16	Me Hungry Want Eat																																															
9	2024-04-22 16:01:16	Cousin Bird Family Brother																																															
10	2024-04-22 16:01:16	Yes Me Tired Need Sit																																															
Result (Pass/Fail)	Pass																																																

Table 6-24 Test Case for Testing Deletes History Record Feature

Test Case Name	Testing deletes history record feature
Test Case Description	User selects the ‘Delete’ button in the dialog of first history record.
Expected Output	The first record is deleted from the database.
Input	
Output	
Result (Pass/Fail)	Pass

6.3 Project Challenges

In today's world, technology plays a pivotal role in bridging communication gaps and fostering inclusivity. One such endeavor is the development of an application that aims to translate American Sign Language (ASL) into English text. This innovative project not only demonstrates the power of modern technology but also highlights the challenges encountered during its development process. By addressing these challenges, the application strives to provide a seamless and accessible experience for individuals who communicate through ASL.

The first challenge encountered is relevant to the feature engineering module. The primary aim of this project was to develop an intuitive Android mobile application capable of translating American Sign Language (ASL) into English. The choice of a mobile application was driven by its offline accessibility, ensuring users can utilize the application without the need for an internet connection, thereby offering high mobility. During the development phase, the system utilized MediaPipe Holistic, a framework designed to detect the human body and extract holistic landmarks. Unfortunately, MediaPipe Holistic doesn't support programming languages commonly used for mobile application development, such as Java or Kotlin. Instead, it is compatible only with Python and JavaScript. To overcome this limitation and ensure the project's progression, the objective was redefined. The focus shifted from an Android mobile application to a web-based sign language recognition application. While this approach may not offer the same benefits of offline accessibility that a mobile app provides, it introduces its advantages. Web applications can be accessed from any device with a web browser, irrespective of the operating system.

After that, the second challenge is the limited computational capacity of mobile phones. The web application handles both landmark extraction and sign language recognition on the front-end, requiring significant computational power from the device. Consequently, the application often experiences lag and becomes less user-friendly on mobile phones, reducing its accessibility and usability.

The development of the ASL-to-English translation mobile application represents a significant step towards promoting inclusivity and removing communication barriers. Despite the challenges faced, such as the limitations of MediaPipe Holistic's compatibility with mobile development languages and the computational constraints of

mobile devices, the project underscores the importance of leveraging technology to empower individuals and foster understanding. As technology continues to evolve, it is crucial to address these challenges and explore innovative solutions that enhance accessibility and usability for all users, regardless of their communication preferences or abilities.

6.4 Objective Evaluation

All the objectives defined in Chapter 1 were successfully achieved. For the first objective, the web application can capture video frames, detect the human body, and extract holistic landmarks to perform predictions using a model trained from scratch. The web application is accessible from any handheld device via a browser. Regarding the second objective, the web application can recognize and translate 50 ASL actions covering different aspects of life with a test accuracy exceeding 80%. The third objective has also been achieved, with the web application capturing frames and drawing landmarks on the detected human body with multiple colors in real-time. Lastly, the web application achieved the fourth objective by providing a sign language dictionary section with information on recognizable actions and a search feature for users to easily locate and view corresponding video clips.

In conclusion, the proposed solutions effectively address the challenges faced in developing a practical and efficient sign language recognition system for mobile devices. By leveraging emerging technologies, refining underlying models, and ensuring computational efficiency, this work paves the way for more accessible and user-friendly communication tools for the deaf and hard-of-hearing communities.

6.5 Concluding Remark

In summary, the rigorous system verification process, involving comprehensive test cases and real-world scenarios, has validated the robustness and effectiveness of the developed sign language recognition system. Despite the challenges encountered during the project, such as technological constraints and computational limitations, the successful achievement of all defined objectives demonstrates the system's ability to bridge communication gaps and foster inclusivity. By leveraging emerging technologies, refining underlying models, and prioritizing computational efficiency, this work lays a solid foundation for more accessible and user-friendly communication tools tailored to the needs of the deaf and hard-of-hearing communities. The insights

CHAPTER 6

gained from this project pave the way for further advancements in the field, ultimately contributing to a more inclusive society where communication barriers are minimized.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

This chapter summarizes the successful development of the Sign Language Recognition System (SignSpeak), highlighting its significance in improving accessibility for the hearing impaired. This chapter offers potential solutions to address challenges such as limited phone computational capacity, including utilizing high-powered devices or backend servers. Additionally, it suggests future development avenues, such as leveraging MediaPipe Holistic for mobile applications and adopting Transformer models with expanded datasets to enhance system performance and user experience.

7.1 Conclusion

In conclusion, the report has detailed the development journey of the Sign Language Recognition System (SignSpeak). By examining existing literature, implementing a systematic methodology, and overcoming challenges in design and implementation, a functional system has been successfully created. This achievement marks a significant step forward in improving accessibility for the hearing impaired. Moving forward, ongoing improvements to this system promise to further enhance its usability and impact, ultimately benefiting individuals who rely on sign language for communication.

7.2 Recommendations

The first recommendation is to address the second challenge mentioned in Chapter 6.3, which is the limited computational capacity of mobile phones. This can help provide the project with greater usability and practicality. There are some potential solutions to the challenge, but they come with their own limitations.

The first solution suggests running the web application on high-powered mobile phones. However, tests on mid-range Android and iOS devices showed poor performance, indicating that only high-end phones would effectively run the application. This approach raises the bar for users, as they would need a high-end phone to use the application smoothly.

The second solution involves moving the landmark extraction module and model to a backend server, making them accessible via an API. While this could reduce the computational load on the mobile device, it introduces a new issue: the requirement for a large bandwidth. Continuous transmission of video frames over the internet would be

necessary to ensure real-time prediction, potentially leading to increased data usage and latency problems.

The second recommendation is to develop a sign language recognition mobile application using MediaPipe Holistic when it supports the programming languages commonly used for mobile development, such as Java and Kotlin. This is because MediaPipe Holistic might be optimized to be suitable for use in mobile phones at that time, thus providing a good user experience without requiring users to have high-end performance phones.

Lastly, the third recommendation is to improve the methodology of developing the model by using a Transformer model and increasing the size of the dataset. This is because the Transformer model has been proven to have better capability to capture temporal patterns than LSTM models, which can help users communicate using the system in a more contextual situation. Increasing the size of the dataset can help the system recognize more sign language actions and generalize well to new data, making it more stable and useful.

In conclusion, the recommendations address challenges in a sign language recognition project. To overcome limited phone computational capacity, options include high-end devices or backend servers, each with drawbacks. Future use of MediaPipe Holistic aims to optimize user experience without high-end phones. Additionally, employing Transformer models and expanding datasets improves pattern recognition and system stability.

REFERENCES

- [1] “International Day of Sign Languages,” *United Nations*. [Online]. Available: <https://www.un.org/en/observances/sign-languages-day#:~:text=Collectively%2C%20they%20use%20more%20than,informally%20when%20travelling%20and%20socializing>. [Accessed: 22-Apr-2023].
- [2] “Deafness and hearing loss,” *World Health Organization*. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>. [Accessed: 21-Apr-2023].
- [3] “MS-ASL,” Microsoft Research, <https://www.microsoft.com/en-us/research/project/ms-asl/> (accessed Apr. 13, 2024).
- [4] C. Neidle and A. Opoku, “ASLLRP sign Bank - Boston University,” Boston University, <https://www.bu.edu/asllrp/rpt20/asllrp20.pdf> (accessed Apr. 13, 2024).
- [5] “MediaPipe holistic - simultaneous face, hand and pose prediction, on device,” Google Research, <https://research.google/blog/mediapipe-holistic-simultaneous-face-hand-and-pose-prediction-on-device/> (accessed Apr. 13, 2024).
- [6] GfG, “What is opencv library?,” GeeksforGeeks, <https://www.geeksforgeeks.org/opencv-overview/> (accessed Apr. 13, 2024).
- [7] S. Saxena, “What is LSTM? introduction to long short-term memory,” Analytics Vidhya, <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/> (accessed Apr. 13, 2024).
- [8] J. Brownlee, “A gentle introduction to long short-term memory networks by the experts,” MachineLearningMastery.com, <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/> (accessed Apr. 13, 2024).
- [9] M. Phi, “Illustrated guide to LSTM’s and GRU’s: A step by step explanation,” Medium, <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (accessed Apr. 13, 2024).
- [10] “Firebase hosting,” Google, <https://firebase.google.com/docs/hosting> (accessed Apr. 13, 2024).
- [11] N. El-Bendary, H. M. Zawbaa, M. S. Daoud, A. E. Hassanien and K. Nakamatsu, “ArSLAT: Arabic Sign Language Alphabets Translator,” 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), Krakow, Poland, 2010, pp. 590-595, doi: 10.1109/CISIM.2010.5643519.

APPENDIX

- [12] P. S. Rajam and G. Balakrishnan, "Real time Indian Sign Language Recognition System to aid deaf-dumb people," 2011 IEEE 13th International Conference on Communication Technology, Jinan, China, 2011, pp. 737-742, doi: 10.1109/ICCT.2011.6157974.
- [13] M. Taskiran, M. Killioglu and N. Kahraman, "A Real-Time System for Recognition of American Sign Language by using Deep Learning," 2018 41st International Conference on Telecommunications and Signal Processing (TSP), Athens, Greece, 2018, pp. 1-5, doi: 10.1109/TSP.2018.8441304.
- [14] "Sign language recognition using python and opencv," DataFlair, <https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/> (accessed Apr. 7, 2024).
- [15] Y. Obi, K. S. Claudio, V. M. Budiman, S. Achmad, and A. Kurniawan, "Sign language recognition system for communicating to people with disabilities," *Procedia Computer Science*, vol. 216, pp. 13–20, 2023, doi: 10.1016/j.procs.2022.12.106.

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 2
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done reviewing the work of FYP 1.
- Done reviewing the work to be done in FYP2.

2. WORK TO BE DONE

- Search for other datasets.
- Combine the datasets to create a bigger dataset.
- Decide the 50 actions.

3. PROBLEMS ENCOUNTERED

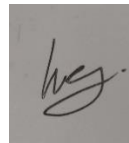
- Most of the datasets are not well-structured.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 4
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done searching for other datasets.
- Done combining the datasets to create a bigger datasets.
- Done deciding the 50 actions.

2. WORK TO BE DONE

- Extract keypoints from the combined dataset.
- Store the dataset in structured manner.

3. PROBLEMS ENCOUNTERED

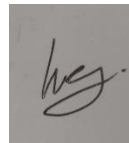
- No.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 6
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done extracting keypoints from the combined dataset.
- Done storing the dataset in structured manner.

2. WORK TO BE DONE

- Designs the network architecture.
- Trains the models.

3. PROBLEMS ENCOUNTERED

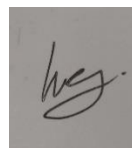
- Face difficulty in choosing the best architecture.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 8
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done designing the network architecture.
- Done training the models.

2. WORK TO BE DONE

- Perform hyperparameter tuning.
- Develop web application.
- Integrate the final model to web application.

3. PROBLEMS ENCOUNTERED

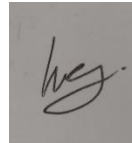
- No.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 10
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done performing hyperparameter tuning.
- Done developing web application.
- Done integrating the final model to web application.

2. WORK TO BE DONE

- Test the web application.
- Write report.

3. PROBLEMS ENCOUNTERED

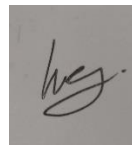
- No.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 3	Study week no.: 12
Student Name & ID: Tee Wei Heng 20ACB02066	
Supervisor: Dr. Ng Hui Fuang	
Project Title: Sign Language Recognition with Computer Vision	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Done testing the web application.
- Done writing report.

2. WORK TO BE DONE

- Prepare for presentation.

3. PROBLEMS ENCOUNTERED

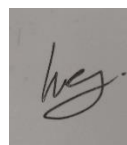
- No.

4. SELF EVALUATION OF THE PROGRESS

- Satisfied.



Supervisor's signature



Student's signature

POSTER

SIGN LANGUAGE RECOGNITION WITH COMPUTER VISION

BY TEE WEI HENG 20ACB02066
Supervised by Dr. Ng Hui Fuang



Wholly owned by UTAR Education Foundation
(Co. No. 578227-M)
DU012(A)



OBJECTIVES

- Develop a sign language recognition web applications
- Train a model that can accurately recognize 50 different ASL actions.
- Provide a sign language dictionary

PROBLEM STATEMENT

- Sign Language users are facing difficulty in daily communication

CONTRIBUTIONS

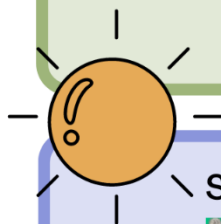
- Builds a large ASL Dataset with 5000 samples for 50 classes
- Developed a comprehensive sign language recognition web application
- Trained a sign language model with test accuracy over 80%

MODEL ACCURACY

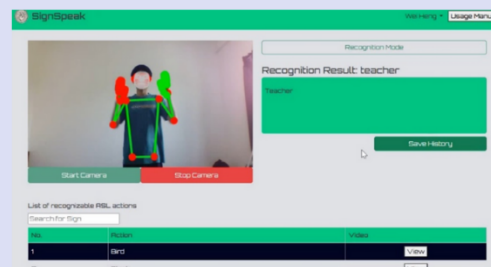
Train Accuracy: 95%
Test Accuracy: 85%

MODEL ARCHITECTURE

- Long Short Term Memory (LSTM) Neural Network



SYSTEM IMPLEMENTATION



The system supports:

- Sign Language Recognition
- Action Rating
- Account Management
- Translation History Management

PLAGIARISM CHECK RESULT

20ACB02066_FYP2

ORIGINALITY REPORT

11%

SIMILARITY INDEX

9%

INTERNET SOURCES

6%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1

eprints.utar.edu.my

Internet Source

1%

2

cse.anits.edu.in

Internet Source

1%

3

123dok.com

Internet Source

1%

4

hdl.handle.net

Internet Source

<1%

5

ai.googleblog.com

Internet Source

<1%

6

www.analyticsinsight.net

Internet Source

<1%

7

www.mdpi.com

Internet Source

<1%

8

Abdelaziz Testas. "Distributed Machine Learning with PySpark", Springer Science and Business Media LLC, 2023

Publication

<1%

link.springer.com

PLAGIARISM CHECK RESULT

9	Internet Source	<1 %
10	machinelearningmodels.org Internet Source	<1 %
11	google.github.io Internet Source	<1 %
12	ijircce.com Internet Source	<1 %
13	docshare.tips Internet Source	<1 %
14	Eleni-Ioanna Koutsovili, Ourania Tzoraki, Nicolaos Theodossiou, George E. Tsekouras. "Early Flood Monitoring and Forecasting System Using a Hybrid Machine Learning-Based Approach", ISPRS International Journal of Geo-Information, 2023 Publication	<1 %
15	medium.com Internet Source	<1 %
16	mihealthandsafety.org Internet Source	<1 %
17	github.com Internet Source	<1 %
18	fastercapital.com Internet Source	<1 %

PLAGIARISM CHECK RESULT

19	Zhanjun Hao, Yuejiao Wang, Zhenyi Zhang, Xiaochao Dang. "EarHear: Enabling the Deaf to Hear the World via Smartphone Speakers and Microphones", IEEE Internet of Things Journal, 2024 Publication	<1 %
20	www.mirlabs.org Internet Source	<1 %
21	analyticsindiamag.com Internet Source	<1 %
22	doczz.net Internet Source	<1 %
23	www.egekongreleri.org Internet Source	<1 %
24	www.queppelin.com Internet Source	<1 %
25	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2018 Publication	<1 %
26	in.store.asus.com Internet Source	<1 %
27	umpir.ump.edu.my Internet Source	<1 %

PLAGIARISM CHECK RESULT

Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Tee Wei Heng
ID Number(s)	2002066
Programme / Course	Bachelor of Computer Science (HONOURS)
Title of Final Year Project	Sign Language Recognition with Computer Vision

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)
Overall similarity index: <u> 11 </u> % Similarity by source Internet Sources: <u> 9 </u> % Publications: <u> 6 </u> % Student Papers: <u> 0 </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
Parameters of originality required, and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Ng Hui Fuang

Date: 26/4/2024

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB02066
Student Name	Tee Wei Heng
Supervisor Name	Dr. Ng Hui Fuang

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
n/a	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 26/6/2024