

WEB APPLICATION FOR LIPREADING

LAU YEE LIN

UNIVERSITI TUNKU ABDUL RAHMAN

WEB APPLICATION FOR LIPREADING

LAU YEE LIN

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Science (Honours) Software
Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Lau Yee Lin

ID No. : 2005403

Date : 12 September 2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled "**WEB APPLICATION FOR LIPREADING**" was prepared by **LAU YEE LIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : *kckhor*

Supervisor : Khor Kok Chin

Date : 1/10/2024

Signature : _____

Co-Supervisor : _____

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, LAU YEE LIN. All right reserved.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Khor Kok Chin for his invaluable advice, guidance and his enormous patience throughout the development of my project. He has given me support and shared his valuable life experiences in encouraging me to face the challenges encountered throughout the work of the project.

Next, I would like to extend my appreciation to the juniors of my same course, for their kindness and helpfulness in contributing their time and efforts for me to collect data required in the project.

In addition, I also felt deeply grateful to my loving parents and friends who had helped and encouraged me to went through the tough times in my academic journey.

Finally, I am grateful to the Department of Computing (DC) for providing the information, resources, and facilities required for me to experience a smooth academic study and be able to complete the development project.

ABSTRACT

Communication happens every day in our daily lives. However, there are conditions where the communication occurs in an environment which impedes the listener from listening to the message clearly. Therefore, this project aims to develop a web application that can perform lipreading using an existing deep learning model, LipCoordNet. It allows users to upload video to the web application and the application will generate text and video output for the users to visualize the speech instead of listening to the sounds. The users can choose to download the predicted text to their own device for future usage. Based on the output of the lipreading, the average word error rate (WER) and character error rate (CER) of an Asian speaker and a Native speaker is calculated, resulting in the average WER and CER value of the Asian speaker being higher than that of the Native speaker. To reduce the WER and CER of the sentences spoken by Asian speakers, efforts have been made in trying to train the LipCoordNet model with the Asian speakers dataset. 270 Asian speaker dataset has been collected with 27 Asian speakers speaking 10 sentences each. For the evaluation of the usability of the web application, five respondents are selected to participate in the system usability testing and contribute to the system usability scale (SUS) score. The SUS score obtained is 87.5, indicating that the system receives a grade A with the adjective rating of Excellent.

TABLE OF CONTENTS

DECLARATION	3
APPROVAL FOR SUBMISSION	4
ACKNOWLEDGEMENTS	6
ABSTRACT	7
TABLE OF CONTENTS	1
LIST OF TABLES	6
LIST OF FIGURES	8
LIST OF SYMBOLS / ABBREVIATIONS	13
LIST OF APPENDICES	14

CHAPTER

1	INTRODUCTION	15
1.1	General Introduction	15
1.2	Importance of the Study	15
1.3	Problem Statement	16
1.3.1	External Environment Impeding Listening of Human's Voice during a Communication	16
1.3.2	Manual Lipreading and Manual Recording in Human-based Lipreading	17
1.4	Aim and Objectives	18
1.5	Scope and Limitation of the Study	18
1.5.1	Targeted Users	19
1.5.2	Modules Covered	19
1.5.3	Development Tools, Languages and Frameworks	20
1.5.4	Limitations	20
1.6	Proposed Solution	21
1.7	Proposed Approach	23

2	LITERATURE REVIEW	25
2.1	Overview of Lipreading	25
2.1.1	Why Lipreading is Important?	25
2.1.2	Lipreading Methods	26
2.1.3	Summary of Lipreading	27
2.2	Traditional machine learning models	27
2.2.1	Overview of traditional machine learning models	27
2.2.2	Description of various traditional machine learning algorithms	28
2.2.3	Problems of existing traditional machine learning models	29
2.2.4	Summary of traditional machine learning models	29
2.3	Deep learning models	30
2.3.1	Reasons of choosing deep learning models	30
2.3.2	Architecture of deep learning model	31
2.3.3	Type of deep learning models	32
2.3.4	Summary of deep learning models	33
2.4	Options for lipreading detection	34
2.5	The existing methods in lipreading	36
2.6	Dataset Overview	69
2.6.1	GRID Dataset	69
2.6.2	LRW Dataset	70
2.6.3	Summary of Datasets	71
2.7	Evaluation Metrics	71
2.7.1	Metrics to Measure Lipreading Model Performance	71
2.7.2	Summary of Evaluation Metrics	74
2.8	System Usability Scale (SUS)	74
2.8.1	Guidelines in Obtaining SUS Score	74
2.8.2	Interpretation of SUS Score	76
2.8.3	Reason of Using SUS as Usability Measurement Tool	77

	2.8.4 Summary of SUS	77
3	METHODOLOGY AND WORK PLAN	78
3.1	Introduction	78
3.2	Summary of Model Workflow	78
3.2.1	Data Collection	79
3.2.2	Data Preprocessing	80
3.2.3	Data Augmentation	82
3.2.4	Apply Pre-trained Model	82
3.2.5	Performance Evaluation	83
3.3	Project Planning and Scheduling	91
3.3.1	Work Breakdown Structure (WBS)	91
3.3.2	Gantt Chart	94
3.4	Software Development Methodology	98
3.4.1	Requirements	98
3.4.2	Planning	98
3.4.3	Iteration Initialization	99
3.4.4	Design	99
3.4.5	Implementation	99
3.4.6	System Testing	100
3.4.7	Retrospective	100
3.5	Technologies and Development Tools	100
3.5.1	Visual Studio Code	100
3.5.2	Python	100
3.5.3	ReactJS framework	100
3.5.4	Flask framework	101
3.5.5	SQLite	101
3.6	Summary	101
4	PROJECT SPECIFICATION	102
4.1	Introduction	102
4.2	Functional Requirements	102
4.3	Non-Functional Requirements	103
4.4	Use Case Diagram	105
4.5	Use Case Description	106
4.6	Initial Prototype	112

4.6.1	Lipreading Model Sample Output	112
4.6.2	Web Application Low Fidelity Interface Design	114
4.7	Summary	117
5	SYSTEM DESIGN	118
5.1	Introduction	118
5.2	System Architecture Design	118
5.2.1	ReactJS architecture	119
5.2.2	Flask architecture	121
5.3	Database design	121
5.3.1	Data dictionary	121
5.4	Conclusion	122
6	SYSTEM IMPLEMENTATION	123
6.1	Introduction	123
6.2	Project Setup	123
6.2.1	SQLite Database Setup	123
6.2.2	Lipreading Model Setup	126
6.3	Web Application Implementation	127
6.3.1	Video Upload Module	129
6.3.2	Lipreading Module	132
6.3.3	Output Video Module	137
6.3.4	Transcription Module	139
6.3.5	Preview Sample Module	142
6.3.6	FAQ Module	144
6.3.7	Contact Module	145
6.4	Conclusion	150
7	SYSTEM TESTING	151
7.1	Introduction	151
7.2	Unit Testing	151
7.2.1	Video Upload Module	152
7.2.2	Lipreading Module	158
7.2.3	Output Video Module	159
7.2.4	Transcription Module	162
7.2.5	Preview Sample Module	165

7.2.6	FAQ Module	167
7.2.7	Contact Module	167
7.3	System Usability Testing	170
7.3.1	Test Scenarios	172
7.3.2	Test Results of System Usability Testing	174
7.4	Conclusion	178
8	CONCLUSION AND RECOMMENDATION	179
8.1	Conclusion	179
8.2	Limitations and recommendations for future works	181
REFERENCES		184

LIST OF TABLES

<p>Table 2.1: A summary of findouts related to lipreading based on the previous works by other researchers.</p> <p>Table 2.2: The CER and WER values of LipNet and LipCoordNet model. The LipCoordNet model has a lower CER and WER value for the overlapped speakers compared to LipNet.</p> <p>Table 3.1: List of sentences spoken by an Asian speaker and a Native speaker and their respective prediction results.</p> <p>Table 3.2: Results of the WER and CER for the five sentences spoken by an Asian speaker and a Native speaker. Average WER and CER have also been calculated and presented in the table.</p> <p>Table 4.1: Functional requirements of the lipreading web application.</p> <p>Table 4.2: Non-functional requirements of the lipreading web application.</p> <p>Table 4.3: Upload Pre-recorded Video Use Case.</p> <p>Table 4.4: Perform Lipreading Use Case.</p> <p>Table 4.5: View Sample Output Use Case.</p> <p>Table 4.6: View Frequently Asked Questions Use Case.</p> <p>Table 4.7: Provide Suggestions Use Case.</p> <p>Table 5.1: Data dictionary for contact collection.</p> <p>Table 6.1: Modules of the Web Application for Lipreading.</p> <p>Table 7.1: Unit testing of video upload module.</p> <p>Table 7.2: Unit testing of lipreading module.</p> <p>Table 7.3: Unit testing of output video module.</p> <p>Table 7.4: Unit testing of transcription module.</p> <p>Table 7.5: Unit testing of preview sample module.</p> <p>Table 7.6: Unit testing of FAQ module.</p> <p>Table 7.7: Unit testing of contact module.</p>	<p>36</p> <p>68</p> <p>84</p> <p>85</p> <p>102</p> <p>104</p> <p>106</p> <p>107</p> <p>109</p> <p>110</p> <p>111</p> <p>122</p> <p>127</p> <p>152</p> <p>158</p> <p>159</p> <p>162</p> <p>165</p> <p>167</p> <p>167</p>
--	---

Table 7.8: User Satisfaction Survey template that consists of ten rating-scale questions and three short answer questions.	171
Table 7.9: Usability Testing Scenarios.	172
Table 7.10: General guideline in interpreting the average SUS score.	175
Table 7.11: Summary of the user satisfaction survey results which include the total score and SUS score for each respondents. Based on the SUS scores, the average SUS score, grade, and adjective rating are obtained.	176
Table 7.12: Answers by the respondents about the most liked features.	176
Table 7.13: Answers by the respondents about the most disliked features.	177
Table 7.14: Answers by the respondents about the suggestions to improve the system.	177
Table 8.1: Limitations and recommendations of the system.	181

LIST OF FIGURES

Figure 1.1: Pre-processing steps of lipreading model.	22
Figure 1.2: The overview of the system. It includes some important features such as uploading video and downloading transcripts at the frontend. In the backend of the system, it contains a Flask application to serve as the backend server, which able to interact with the SQLite database as well as the LipCoordNet model for the lipreading feature.	23
Figure 1.3: Overview of the PXP methodology phases.	24
Figure 2.1: LRW dataset sample of the mouth region extracted from a male speaker (top row) and a female speaker (bottom row) pronouncing the word ‘about’.	71
Figure 2.2: Sample of the SUS questionnaire containing the guideline in calculation of SUS score. The tick symbols within the scale of each questions are the sample of user’s answer. The numbers written on the rightmost part of the questionnaire are the computed score for each questions. The value 22 represents the total score which is the sum of all computed scores whereas the value 55 is the SUS score that is done by multiplying 2.5 with total score.	75
Figure 2.3: 3 different approaches to interpret SUS score, by using adjective ratings, grade scale, and acceptability ranges.	76
Figure 3.1: Flowchart of the workflow of LipCoordNet model.	78
Figure 3.2: Code snippet in charge of obtaining the video frames and storing them in array.	79
Figure 3.3: Code snippet related to the face alignment and frame resizing.	80
Figure 3.4: Face detector and shape predictor used to extract lip coordinates.	80
Figure 3.5: Code snippet related to the extraction of lip coordinates.	81
Figure 3.6: Code snippet related to normalization of lip coordinates and storing them in PyTorch tensor.	81
Figure 3.7: Code of HorizontalFlip function and ColorNormalize function.	82

Figure 3.8: The screenshot of the videos of the Native speaker and the Asian speaker speaking the sentences.	83
Figure 3.9: Code to calculate the WER and CER. The predicted text and actual text should be updated with the sentences that want to be calculated for WER and CER.	84
Figure 3.10: Folders containing the videos of the 27 speakers.	86
Figure 3.11: Sample of the video frames converted from a video of an Asian speaker reading a sentence.	86
Figure 3.12: Code snippet in lips_coords_extractor.py file which uses shape_predictor_68_face_landmarks_GTX.dat pretrained model as the shape predictor to obtain lip landmark coordinates.	87
Figure 3.13: Sample content in the lip landmark coordinates file.	87
Figure 3.14: Example of the information required in the alignment file to train the LipCoordNet model. It consists of the start time of a word (in milliseconds), end time of a word (in milliseconds), and the word itself.	88
Figure 3.15: Screenshot of the output by the online forced aligner tool.	89
Figure 3.16: A csv file showing the start time and end time of the words at the green colour columns. This csv file is downloaded together with the .TextGrid file when using the online forced aligner tool.	89
Figure 3.17: Gantt chart of this project.	97
Figure 4.1: Use case diagram of the lipreading web application.	105
Figure 4.2: Sample input video to test the model. The spoken words are ‘BIN RED AT S NINE AGAIN’.	113
Figure 4.3: Predicted result of the video is ‘PLACE RED AT S SIX AGAIN’.	113
Figure 4.4: Sample output video with the predicted sentence overlayed in the video.	114
Figure 4.5: Interface design of the home page of the web application.	114
Figure 4.6: Interface design of the upload page of the web application.	115
Figure 4.7: Interface design of the lipread page of the web application.	116

Figure 5.1: The overview of the system architecture design.	118
Figure 5.2: The high level architecture of React application.	120
Figure 6.1: The code snippet for defining and creating database and ‘Contact’ table.	125
Figure 6.2: Location of the lipreadDB.db file.	125
Figure 6.3: The ‘Contact’ table displayed in DB Browser for SQLite.	126
Figure 6.4: The required dependencies and their respective versions to be installed.	127
Figure 6.5: Home page of the web application which include some descriptions of the application and some navigation buttons.	129
Figure 6.6: The video requirements are listed on the webpage for users to refer before they upload their video to the system.	130
Figure 6.7: The sample video for users to refer before before they upload their video to the system.	130
Figure 6.8: The video upload module contains the features for users to choose a video file for lipreading, preview the uploaded video, and start the lipreading process.	131
Figure 6.9: The code snippet for the user interface shown in Figure 6.8. It includes the logic in handling the file uploaded by the users and displaying the successfully uploaded video to the screen.	132
Figure 6.10: The code snippets of the ‘handleSubmit’ function related to the frontend logic in processing the lipreading.	133
Figure 6.11: The code snippet of handling the ‘/lipread’ POST request in Flask application.	134
Figure 6.12: Screenshot of the loading overlay.	135
Figure 6.13: Screenshot of the code related to the loading overlay.	135
Figure 6.14: Code snippet related to the logic of cancel loading overlay.	135
Figure 6.15: Details of the video setting that have been performed on the captured video before uploading to the web application.	136
Figure 6.16: The complete user interface for the lipreading output page.	137

Figure 6.17: Frontend code in obtaining and displaying the output video.	138
Figure 6.18: Backend code in responding client request with the output video file (output.mp4).	138
Figure 6.19: Screenshot of the output video file rendered in the browser which contains the transcriptions being overlayed on the video.	139
Figure 6.20: Code snippet related to displaying of fetched transcription text at the textarea.	139
Figure 6.21: Flask application code related to the accessing of /transcription endpoint via a GET request.	140
Figure 6.22: Screenshot of the transcription displayed at the textarea.	141
Figure 6.23: Code snippet used to handle the logic of modifying the transcriptions at the textarea.	141
Figure 6.24: Screenshot of the lipreading output webpage to indicate the ‘Download Transcriptions’ button.	142
Figure 6.25: Code snippet related to the logic in handling download transcriptions.	142
Figure 6.26: The download text file content, which is same as what has been modified at the textarea of the web application.	142
Figure 6.27: The screenshots of the web application for the preview sample module.	143
Figure 6.28: The code snippet used to develop preview sample module.	144
Figure 6.29: Screenshots of the web application FAQ module.	144
Figure 6.30: Code snippet for developing FAQ module with the first question and answer division being shown.	145
Figure 6.31: Screenshot of the web application contact module.	145
Figure 6.32: Sample of error message being shown when a user does not enter the name credential and presses the submit button.	146
Figure 6.33: Sample of valid user input which is ready to be submitted.	146
Figure 6.34: Code snippet of the contact module.	148

Figure 6.35: Code snippet of the backend logic of handling the contact form submission.	149
Figure 6.36: The output of the Flask application to show that the POST request was made successfully.	149
Figure 6.37: The name, email, and message data submitted in the web application are successfully stored in the database.	150
Figure 7.1: The respondent, Tee Szen Yong is conducting the usability testing physically.	170
Figure 7.2: Screenshots of the video taken by the respondents during the usability testing.	174

LIST OF SYMBOLS / ABBREVIATIONS

<i>TP</i>	true positive
<i>TN</i>	true negative
<i>FP</i>	false positive
<i>FN</i>	false negative
<i>CER</i>	character error rate
<i>C</i>	characters
<i>S</i>	substitutions error
<i>D</i>	deletions error
<i>I</i>	insertions error
<i>C_N</i>	total number of characters in reference
<i>WER</i>	word error rate
<i>W</i>	words
<i>S</i>	substitutions error
<i>D</i>	deletions error
<i>I</i>	insertions error
<i>W_N</i>	total number of words in reference

LIST OF APPENDICES

Appendix A: Informed consent form	190
Appendix B: Usability test responses	195

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Communication is unavoidable in human society as it supports people in five major goals, which are exchanging information, expressing feelings, sharing imaginations, meeting social expectations, and influencing others (Britannica Kids, n.d.). When interpersonal communication is conducted, it involves at least two parties, one acting as a sender and one acting as a receiver. The sender initiates the communication by encoding the message in an understandable form to the receiver who receives the sender's message followed by interpreting the message (Nordquist, 2020).

However, there are always communication barriers which become obstacles to effective communication. Hence, this project serves to resolve the difficulties faced by the receiver in receiving the sender's message, specifically to resolve the noisy and silent environment that impedes the complete receiving of the message. It involves the application of a visual speech recognition technique, as known as lipreading, by integrating a lipreading model, the LipCoordNet model into a web application which allows interpretation of human lip patterns into texts. By having this lipreading feature, people are able to visualize speech sounds instead of listening to the sounds (Better Health Channel, n.d.).

Although there are various tutorials in teaching humans to lipread, sometimes normal speech is too fast for humans to lipread easily and many speech patterns are similar, leading to confusion and doubt (Better Health Channel, n.d.). Therefore, with the help of deep learning, the lipreading process can be done automatically and more effective communication can be enhanced.

1.2 Importance of the Study

This project contributes a practical application of the lipreading by developing a web application with the major feature of lipreading. It helps to resolve the obstacles of communication by providing a lipreading feature that can translate some simple words into text form and allow visualize of the transcriptions via

output video. With the help of this web application, people will be able to cope with the challenges faced during communication, specifically in the noisy environments and at the silent environment which requires low speaking volume.

1.3 Problem Statement

The major problems that this project would like to address are related to the external environment that affects the effective communication. With the help of lipreading which is a visual speech recognition method, the obstacles can be reduced. However, human-based lipreading still causes some undesirable burdens to both lipreaders and users. The details of the problems are described below.

1.3.1 External Environment Impeding Listening of Human's Voice during a Communication

The external environment always impedes communication from going smoothly. There are always bad acoustic conditions that cause people to fail to listen clearly to what others saying. Due to the bad acoustic conditions might involve many different possible factors, in this project the bad acoustic conditions are specified to the loud background noise environment and the silent environment.

The most common situation where people cannot listen clearly to what others are saying is when people speak in environments with loud background noise. This environment causes the human's speaking voice to be covered by the background noise (Xu et al., 2018). In this environment, the surrounding noise is in high volume, accompanied by multiple simultaneous speakers speaking, which means that many conversations are happening together. This causes the listeners to fail to listen clearly to their respective speakers.

Another common situation where people are unable to listen clearly to what others are saying is when the people are staying in an environment that fixed the rule to keep silent. These environments can be libraries, museums, meetings, or conferences. In this environment, the speech volume is low and people may whisper to others, causing the listener to fail to listen clearly to each of the spoken words. Communication may be done by silently mouthing the words, but it is hard to understand.

Another example can be found in police and detective agencies, in which they may collect some videos that consist of the key evidence. However, due to some limited functionality, the recorded videos do not have sound. This limitation causes the inspection team to fail to collect the words said by the suspect or victim and lose the key evidence (121 Captions, n.d.).

Although people can lipread by themselves to support the understanding of the sentences, human lipreading is extremely challenging, particularly in a lack of context. Apart from the lips, tongue, and teeth, the majority of lipreading actuations are latent and impossible to distinguish when context is absent (Assael et al., 2016). Hence, an automated lipreading feature can perform lipreading even in an environment with loud background noise or unclear voice, helping those in need to enjoy more convenient communication.

1.3.2 Manual Lipreading and Manual Recording in Human-based Lipreading

In human-based lipreading, people can lipread the lip, tongue, and teeth movements to understand the sender's messages. However, not everyone has the proficiency and ability in lipreading manually with a high accuracy. In a project in which the performance of an artificial-intelligence lipreading system and human lipreading is compared, the former achieved an accuracy of 93.4% whereas the latter predicted only 52.3% correctly (Condliffe, 2016). This proves that many people fail to make use of this lipreading technique in communication.

Also, sometimes people may need to record the messages for future reference. In order to record the messages, some lipreaders have to manually record the translated sentences into various multimedia elements such as texts, audio, or videos. However, this is a very time-consuming and tedious process as the lipreaders have to type the texts one by one or record an audio or video. It is also inconvenient for the users who refer to the translated audio or videos as they might need to refer a few times to capture every single translated word.

Moreover, if the lipreader is a deaf person, the human-based lipreading process is very exhausting and may cause concentration fatigue to the lipreader. Deaf individuals have to make use of various attention mechanisms to interpret and understand the surrounding condition (Khalifa, 2018).

Therefore, if there is a web application that possesses features such as automatic lipreading and enables downloading of the translated texts, the shortcomings of human-based lipreading can be addressed by avoiding the manual lipreading and manual recording process, saving time, effort and improving efficiency at the same time.

1.4 Aim and Objectives

This project aims to develop an application for reading lips using a deep learning model, LipCoordNet and translate it into a form that is easily understood by people.

The objectives of this project are:

- i. To study an existing deep learning model, LipCoordNet for lipreading based on its suitability with this project.
- ii. To implement the deep learning model, LipCoordNet in a web application.
- iii. To evaluate the usability of the web application using System Usability Scale (SUS) after completed the development of the application.

1.5 Scope and Limitation of the Study

This project covers the development of a web application for reading human lips who are speaking English language sentences and translating them into English language transcripts for users to keep a copy of the text file. The lip recognition and translation are to be done by implementing the deep learning model that is suitable for this project, which is LipCoordNet model among the studied models. The selection of the deep learning model to be applied in this project is done by researching many other models. By considering the popularity of the model in the academic research community of speech recognition, the performance of the model, and the practical implementation perspective of the model, finally LipCoordNet is selected as the model that best fits with the system. The LipCoordNet model will perform the lipreading process and produce the predicted results into output videos and transcriptions that can be easily understood. It is then integrated into the web application.

1.5.1 Targeted Users

The target users for this project include any individuals who are interested in uploading their videos to the web application to use the lipreading feature and view the translated transcripts. These users can be deaf or hearing-impaired individuals who wish to make use of the automatic lipreading feature to assist them in understanding what other people saying. Besides, the users can be academic researchers who want to observe the result of the lipreading in this web application. It is also suitable to be used by individuals who possess videos without audio or with large background noise audio. They can use the web application to lipread their videos so that they can understand the spoken words in the video.

1.5.2 Modules Covered

Below are the lists of modules covered in the web application:

a. Video upload module

Users can upload their pre-recorded videos to the web application. Each upload supports only one video. If users want to choose other videos, they have to discard the original selected video first. The uploaded video can be playback using the video control features such as play, pause, full screen mode, picture in picture mode, change playback speed, and change volume.

b. Lipreading module

When users choose to begin the lipreading, the system will execute the deep learning model to lipread, translate into transcripts and generate output video.

c. Output video module

The output of the lipreading is the output video that contains the transcriptions overlayed on the video. Users can playback the video and use the video control features such as play, pause, full screen mode, picture in picture mode, change playback speed, and change volume.

d. Transcription module

The output of the lipreading is the transcription that is shown on the screen for users to refer to. The users are allowed to edit the translated transcripts if they identify any mistakes in the transcriptions. They can also download the transcription text file into their device if they want to have a copy for their own reference or usage. If users do not download, the web application will not save history transcript record.

e. Preview sample module

Users who wonder about the sample output that will be generated by the lipreading can refer to this module which provides some images and descriptions texts about the output of the lipreading.

f. FAQ module

Users who have enquiries about the web application can refer to this module to look for the answers of their questions.

g. Contact module

Users who wish to provide suggestions or recommendations to improve the web application can send their message to the support team by providing their name, email, and suggestions.

1.5.3 Development Tools, Languages and Frameworks

Visual Studio Code is used as the Integrated Development Environment (IDE) in developing this project. ReactJS framework is used to support the building of the user interfaces and handles the user interaction. The Flask framework coded in Python language is used as the backend server to process requests from frontend and handle the server-side logic. The selected lipreading model, LipCoordNet, is developed using the Python language. SQLite is used for managing data in the database. The visualization of the data is supported by the DB Browser for SQLite tool.

1.5.4 Limitations

For the limitations of this project, the duration of the lipreading process is around 1 minute. The performance of lipreading in terms of the word error rate

(WER) and character error rate (CER) will be higher for Asian speakers than the Native speakers. The lipreading of the sentences that the system currently supports must be in a specific 6-word structure. Also, this project is specifically designed to lipread English language sentences. Therefore, it does not support lipreading in any other language.

Besides that, the video uploaded to the web application shall be recorded with the speaker's lips facing the camera under good lighting conditions and without a shaking camera or cluttered background to ensure that the lipreading can be performed successfully and to optimize the performance of the lipread model. The system currently only supports the lipreading of video that consists of a single speaker. Also, this application only supports lipread of one video per upload. It does not support lipreading of several videos together at the same time.

1.6 Proposed Solution

The problems mentioned in the problem statement above can be addressed by developing a web application which contains a lipreading feature. The lipreading feature is developed by applying an existing deep learning model, LipCoordNet coded in Python language. The web application with this lipreading feature serves as the platform for users to upload videos and view output. In this project, the users can capture video that contains people speaking to be uploaded to the web application. Since the lipreading does not depends on the audio, the system can support lipreading of video without sound or with loud background noise. After successfully uploading the video, whenever users press the begin button, the automated lipreading feature of the system will be initiated. The model will generate translated texts and videos and return them to the web application to be displayed.

The deep learning model, LipCoordNet is specifically designed for lipreading. From the original video, it will execute certain pre-processing steps including facial landmarks detection, image warping and cropping to extract the mouth Region of Interests (ROIs). Figure 1.1 shows the pre-processing steps of the lipreading model.

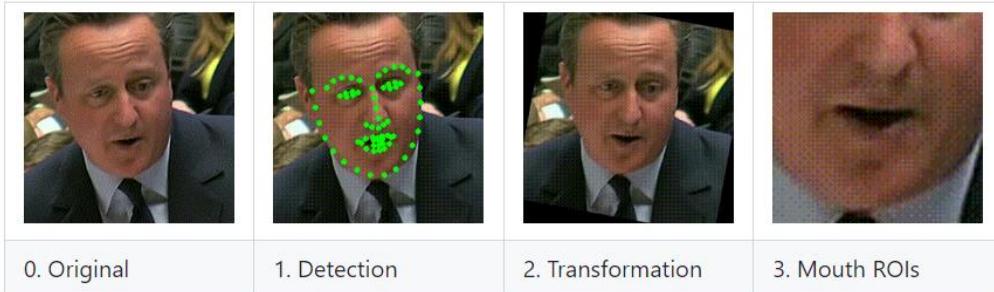


Figure 1.1: Pre-processing steps of lipreading model.

Source: Pingchuan et al. (n.d.)

Apart from the lipreading feature, this application also provides some other necessary features to resolve the problems encountered. There is a download feature in this app which allows users to download the translated transcripts into their own devices. This enables users to open the downloaded transcripts when they want to refer back next time. It helps to resolve the time-consuming problem caused by human-based lipreading. This is because human-based lipreading will need to record down the output of the lipreading manually.

The frontend development of web application is done using ReactJS framework. The backend server used in this project is Flask framework in Python language. The database used is SQLite which can be accessed using DB Browser for SQLite to view the data. Figure 1.2 below shows the overview of the system architecture.

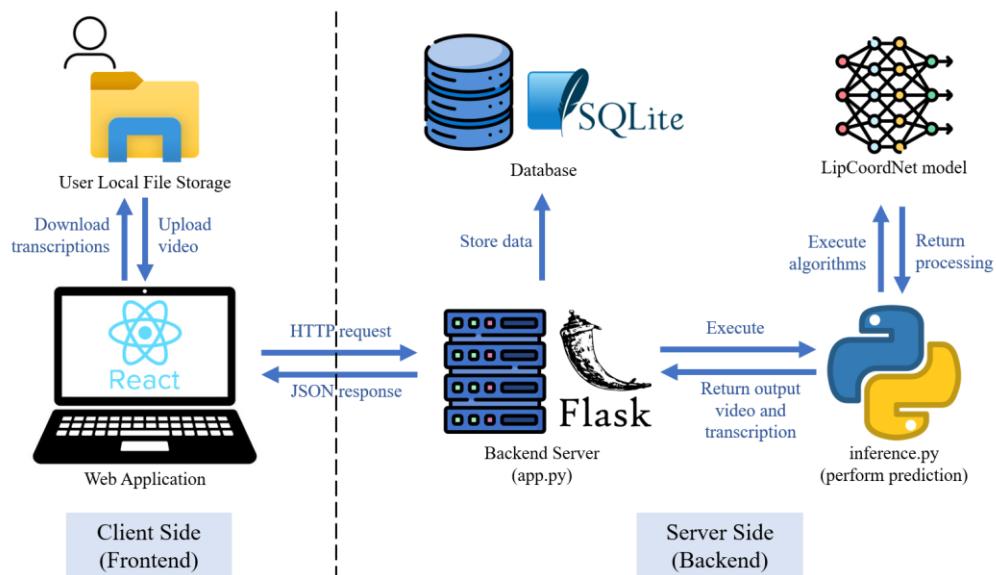


Figure 1.2: The overview of the system. It includes some important features such as uploading video and downloading transcripts at the frontend. In the backend of the system, it contains a Flask application to serve as the backend server, which able to interact with the SQLite database as well as the LipCoordNet model for the lipreading feature.

1.7 Proposed Approach

The software development methodology used in this project is Personal Extreme Programming (PXP). PXP is an agile methodology applied by autonomous developers that combines Extreme Programming (XP) with the Personal Software Process (PSP). The combination of these two methodologies in forming the PXP methodology can enhance autonomous developers' performance and quality by automating developer tasks and carrying out regular retrospections on a daily basis (Ilieva et al., 2009).

PXP methodology is selected because it suggests an iterative approach to developing the project which promotes higher flexibility and responsiveness to changes. Moreover, the efficient development practices used in Extreme Programming can improve project planning as well as product quality control (Asri et al., 2018).

Throughout the process, log files are maintained in which information related to the task planning, task duration, and test cases are recorded.

During the first phase which is the requirements phase, the autonomous developer collects functional and non-functional requirements and records them in the documentation. In the following planning phase, the developer refers to the requirement documentation to create a collection of tasks and smaller subtasks. Each task is estimated with a duration. Important design decisions are made in this phase, such as the programming language, development framework, and others.

Next, the iteration initiation phase begins the iteration, and each iteration take between two to three weeks. Each iteration includes processes such as design, implementation, and system testing.

Retrospective indicates the end of a process iteration. All data collected in the preceding phase were examined to have an improvement proposal. If all

requirements have been met and there are no more open defects, the project is complete (Ilieva et al., 2009). Figure 1.3 shows an overview of the phases of PXP methodology.

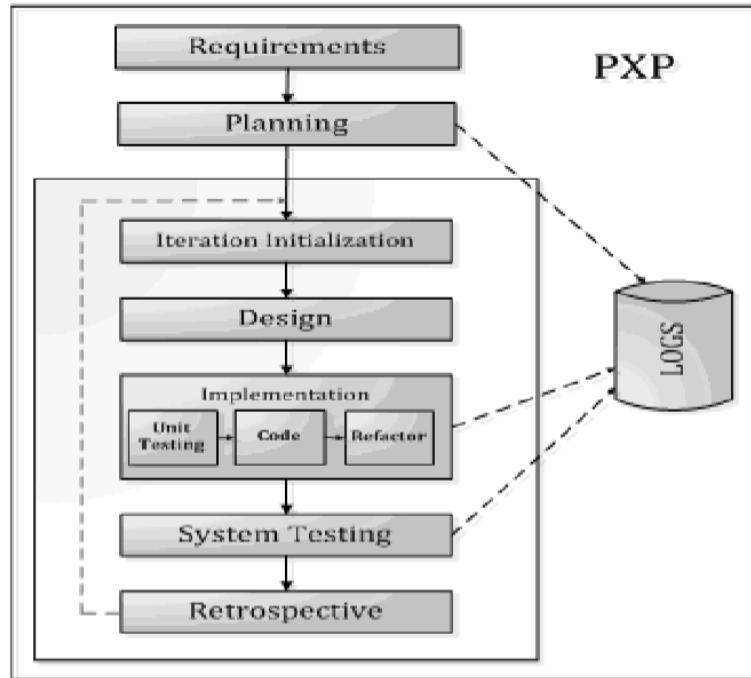


Figure 1.3: Overview of the PXP methodology phases.

Source: Ilieva et al. (2009)

CHAPTER 2

LITERATURE REVIEW

The advancement of lipreading technology is thanks to the efforts of researchers from the past until today. From the human manual technique develop to traditional machine learning and deep learning techniques in the lipreading detection field, a variety of research has been done and recorded in academic papers or reports. In this section, lipreading, traditional machine learning, deep learning, the existing works, and many other research details will be discussed to understand the theories, techniques, and efforts made by the researchers.

2.1 Overview of Lipreading

Lipreading has been introduced by Ponce de Leén in the year of 1504 to the deaf (Nemani et al., 2023). Since then, humans have begun to explore the techniques of lipreading and come out with a lot of approaches. People are interested in discovering this field due to its significance to the society. Thanks to the resurgence of the neural network, lipreading is undergoing huge advancements. In this section, the importance of lipreading as well as some traditional and recent methods in lipreading will be discussed to understand the reasons of people interested in research in this field and the improvements that have been made from the past until now.

2.1.1 Why Lipreading is Important?

Lipreading allows verbal communication to happen in the absence of an audible acoustic signal. The most common scenario where lipreading plays an important role is when the environment has large interfering background noise, people will have to visualise the patterns of the lips, mouths, and tongues to understand the communication. It also enables individuals with hearing impairments issue to communicate with others. People who do not understand sign language can choose to lipread instead. Besides that, there are some scenarios where lipreading should be used to understand silent speech. For instance, people communicate using silent speech in public quiet places such as libraries or theatres where undisturbing communication is necessary. There are also some

scenarios where people would like to communicate some confidential information in public places but it is inconvenient to voice out, they may need to communicate via silent speech (Wand et al., 2016). With the usage of lipreading, silent speech can be interpreted and understood by people, keeping the communication ongoing.

Apart from that, lipreading of surveillance videos to collect forensic evidence during crime-fighting is also very useful (Bowden et al., n.d.). Numerous innovative applications, including enhanced biometric authentication, interaction between humans and machines, and speech intelligibility in noisy settings, have been made possible by research on lipreading (Nemani et al., 2023). Therefore, lipreading plays an important role on many different occasions in helping people to understand the content of the communication when an audio signal is absent.

2.1.2 Lipreading Methods

Researchers have put much efforts to discover various methods of lipreading. The lipreading methods can be categorized into 2 categories, based on the methods of feature extraction.

The first category is the feature extraction based on traditional means by using a two-stage approach. The first step is done by extracting the mouth region of interest (ROI) in order to extract features from that region by using a certain feature extraction algorithm. The extracted features are processed and encoded to become feature vectors with equal length using Discrete Cosine Transform (DCT) and Principal Component Analysis (PCA) (Hao et al., 2020). DCT is one of the most popular feature extractors. The second step is to feed the feature into Hidden Markov Models (HMM) which is a dynamic classifier for classification purposes. The problem is that this kind of classifier depends on the conditional independent assumption. The long-term dependencies are typically failed to be modelled (Xu et al., 2018).

The second category is based on the deep learning method. Thanks to the recent advancement of deep learning, the research on the lipreading approach has the chance to enjoy better improvement. In the initial deep learning works, the two-stage approach of the traditional method is still followed but the feature extraction stage is replaced by using deep autoencoders whereas the

classification is replaced using Long-Short Term Memory (LSTM) networks (Martinez B et al., 2020). However, the major difference between the new deep learning methods and the traditional methods is that the two-stage approach is replaced by using an end-to-end trainable system made up of front-end and back-end neural networks (Weng & Kitani, 2019). It applies convolutional layers or fully connected for the feature extraction and the extracted features are fed into the attention architectures or the recurrent neural network (RNN). This new approach ensures that the learned features have a strong relation with the particular task that the network was trained on (Weng & Kitani, 2019).

2.1.3 Summary of Lipreading

Since people realized the importance of lipreading, people have put efforts to discover this field. In the traditional lipreading approach, it uses a two-stage approach to extract features from mouth ROIs. Based on the extracted features, it performs classification. However, with the advancements of deep learning, this approach is gradually replaced by an end-to-end deep learning approach which results in a better performance and has better strengths compared to the traditional approach. It is believed that with the rapid technological development these days, the research on lipreading will undergo further advancement and have better impacts on society to resolve more problems associated with lipreading.

2.2 Traditional machine learning models

In this section, traditional machine learning theories and some widely used algorithms will be discussed to have an understanding of human efforts in exploring Artificial Intelligence technology.

2.2.1 Overview of traditional machine learning models

Machine learning is a subset of Artificial Intelligence that uses algorithms and statistical models to process data, identify data patterns, and predict outcomes based on the input dataset(Amazon Web Services, n.d.). In this digital era in which data plays a significant role, machine learning is widely applicable in many fields to solve problems, identify opportunities, unlock revenue streams, and drive the growth of businesses. With the presence of this automation

technology done by machine learning, the previous traditional data analysis processes which are much more time-consuming can be replaced and further improved by this more effective method.

2.2.2 Description of various traditional machine learning algorithms

The machine learning algorithms can be organized into several groups such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, transduction, and learning to learn (Nasteski, 2017). In this section, supervised learning and unsupervised learning algorithms are chosen to be discussed.

Supervised learning has the classes to be predetermined and it categorizes data based on the prior information. This algorithm uses training datasets to teach models and produce outputs. The training dataset is made up of inputs, as known as features, and correct outputs, as known as labels (Google Cloud, n.d.). The classification model is a type of supervised learning that is more practically applicable and widely used. It predicts output variables or categorical labels according to the input data and group data based on the prediction result. The classification model is suitable to be used for categorical output variables. Some alternatives to represent classification models include support vector machines (SVM), decision trees, algebraic functions, and so on (Nasteski, 2017). SVM classifies data by constructing an optimal hyperplane that can maximize the distance between two classes of data points in an N-dimensional space (IBM, 2023). The decision tree algorithm builds a tree-like structure that recursively splits data into subsets until reaching a stopping criteria, in which the final goal is to obtain the attribute that can maximize the information gain (GeeksforGeeks, n.d.).

Another type of supervised learning is the regression model. It is used to study the relationship between two or more dependent and independent variables. Examples of common regression algorithms are linear regression, polynomial regression, and logistical regression (IBM, n.d.-a). Linear regression is done by using the independent variable to predict the value of the dependent variable whereas logistical regression performs prediction on the probability of an event or observation to produce a binary outcome.

Unsupervised learning does not use labelled data like supervised learning. It discovers patterns based on the unlabeled data for clustering purposes. This technique is suitable for cases in which the common properties of a dataset are unsure. Some examples of unsupervised learning algorithms include hierarchical clustering, k-means clustering, and Gaussian mixture models (IBM, n.d.-b). Hierarchical clustering is done by grouping objects with similar characteristics to the same group and the clusters are represented with a dendrogram. For k-means clustering algorithm, it uses k to set the number of pre-defined clusters and assign data points to the closest centroid identified in each cluster to become the pre-defined clusters (Javatpoint, n.d.).

2.2.3 Problems of existing traditional machine learning models

Even though traditional machine learning is a powerful technology to analyze data and perform predictions, still there are some problems identified in these existing models. Traditional machine learning algorithms have limited capability in handling complex and unstructured data. Although it can improve its performance when new data are taken in, it still requires some ongoing human intervention for improvements. It would be a challenge if there was no expert to help in interpreting the results and eliminating uncertainty (Amazon Web Services, n.d.). It also requires human work to clean data, select features, and tune the models. Also, the accuracy of machine learning algorithms is limited because they can only perform as good as the quality and quantity of the training data(Drug Discovery Pro, n.d.). Therefore, there are still some limitations in the traditional machine learning models.

2.2.4 Summary of traditional machine learning models

The presence of traditional machine learning models leverages the ability of humans to process and analyse data to help improve their businesses or resolve the challenges faced. Great efforts have been put into discovering various different machine learning algorithms that are suitable to be applied in different cases. However, there are still some problems being identified with the existing traditional machine learning models. Therefore, in the next section, deep learning which is an evolution of machine learning will be discussed.

2.3 Deep learning models

Along with the revolution of technology, deep learning model architecture has become more and more mature, examples like computer vision, natural language process, speech recognition, and others now can be done using deep learning models. The characteristics of the deep learning model are it can progressively learn more and more features about the input data and be able to recognize it using its multiple layers of interconnected neurons (Salakhutdinov et al., 2013). These characteristics of the deep learning model cause it to excel in various kinds of fields including image classification, object detection and recognition, and speech recognition. Hence, the presence of deep learning models is contributing to the advancement of different fields of study. Researchers can make good use of deep learning models to train, optimize, and overcome different problems in the field across different domains.

2.3.1 Reasons of choosing deep learning models

In the early 2000s, there were scientists such as Geoffrey Hinton, Yann LeCun, and Yoshua Bengio who studied the deep learning field (Piper, 2019). However, during that time, due to the limitations in terms of the cost of processing power and the complexity of datasets, the research progress is restricted. However, in the recent 20 years, the limitations have been resolved with the advancement of technologies, therefore deep learning has undergone a resurgence and getting back the attention of the researchers.

Firstly, unstructured data like images, audio, and text can be handled extremely well by deep learning models and even at high accuracy and efficiency. There are various kinds of deep learning models with each of them having its speciality for different kinds of data, for instance, Convolutional Neural Networks (CNN) can be used for images, Recurrent Neural Networks (RNN) can be used for sequential data, or even Transformer-based models can be used for language processing (Questions and Answers in MRI, n.d.). These models have the capability of learning complex features and patterns directly from the data without the need for manual feature engineering, thereby, making them outperform all the other similar technologies like machine learning (LinkedIn, 2024). Its capability to carry out different tasks like language

understanding, speech recognition as well as image recognition, and classifying makes it handy for researchers.

Aside from that, deep learning models have great scalability and flexibility. Data are normally difficult to handle due to their volume and complexity, but this can all be done easily by deep learning models as deep learning models can handle large-scale and complex datasets due to the architecture characteristics (Kansal, 2021). Therefore, it can be a great tool for handling big data. Furthermore, despite the large fields and domains in the world, deep learning models have great flexibility that allows them to adapt to different kinds of fields and domains. Whether it is natural language processing, image recognition, classification, or time-series analysis, deep learning models can be adjusted to deal with specific problems in different fields (Nanos, 2024). For example, there's a transfer learning feature in deep learning that allows the pre-trained deep learning models to learn new fields without the need to retrain the whole model. This high scalability and flexibility make it a great choice for researchers and people to use.

In simple words, the choice of using deep learning is preferable by most people due to its high performance and efficiency at handling unstructured data as well as its high scalability and flexibility that makes it a robust tool to handle different domains and fields of data.

2.3.2 Architecture of deep learning model

Deep learning models are complex for human understanding, the structures of deep learning models, the way deep learning models work, and the architecture of deep learning models are confusing. However, below is a brief explanation of the architecture of the deep learning model.

Deep learning models have a backbone that defines their structure, where the core of the backbone is the Artificial Neural Network (ANN), this ANN consist of interconnected layers of neurons which includes the input layer, hidden layers as well as the output layer (Madhavan & Jones, 2017). The input data will then be processed by one layer and passed to another layer progressively. There are different kinds of layers in deep learning models, and each of the layers has its own purpose. For instance, the convolutional layer is for image processing, pooling layers are used for downsampling, recurrent

layers are for sequential data as well as normalization layers are used for standardizing input (Madhavan & Jones, 2017).

Aside from the layers in the deep learning model, the activation function is also a crucial component of deep learning architecture. In order to learn complex features and patterns from input data, activation functions like Rectified Linear Unit (ReLU) can be used to determine the output of complex functions, thereby improving the capability of the model to learn how the data are interrelated (Madhavan & Jones, 2017). In order to carry out the deep learning model's training, there are also several hyperparameters that can be tuned to enhance and adjust the model's training capability, activations, weight, biases, epoch, learning rate, or even optimizer can be changed and tuned regularly to fit the input data.

Next, deep learning models have forward propagation and backward propagation architecture. The forward propagation is where input data undergoes the activation function, biases and weight progressively and produces the result of prediction, while the backward propagation is done after the forward propagation is complete to adjust the weights and biases and check the prediction errors (Stackademic, 2023).

In short, the complex architecture of deep learning enables it to learn the features and patterns of input data. Depending on the requirement, different kinds of models will be developed and finetuned to achieve their respective goals.

2.3.3 Type of deep learning models

There are different kinds of deep learning models with each of them being created to handle different types of data and purposes, 3 of them are the Convolutional Neural Networks (CNNs), Transformer-based models, and Recurrent Neural Networks (RNNs). The explanations for them are as below. When it comes to computer vision tasks, Convolutional Neural Networks (CNN) is a decent model for dealing it, this is because CNN can extract the features and patterns from the visual data efficiently and effectively. CNN can also apply learnable filters on input images and detection features like edge detection, pattern detection as well as texture detection (AI & Insights, 2023). These

unique features makes CNN extremely effective at handling tasks like object detection and image classification.

Next, transformer-based models are deep learning models for natural language processing (NLP) tasks. This model is different compared to the other models as it does not rely on previous connection information in a sentence, instead, it uses self-attention to comprehend different words in a sentence (Gillioz et al., 2020). Hence, making the model comprehend the word and sentences effectively and efficiently. For instance, it can handle different NLP tasks like language translation, text generation as well as document summarization easily. The obvious real-life applications are the BERT(Bidirectional Encoder Representation From Transformer) and GPT (Generative Pre-Trained Transformer). These two models have great performance at understanding language and sentences (Gillioz et al., 2020).

Apart from that, Recurrent Neural Networks (RNN) is a famous model in deep learning model that handles sequential data. RNN can retain the memory of previous inputs when processing sequential data, thus making it better at processing sequential data compared to the old feedforward networks(IBM, n.d.-a). For instance, this can handle tasks like language modelling and speech recognition. There are now newer versions of the model like LSTM and GRU networks that are created to overcome the ‘vanishing gradient problem’ in old RNN (IBM, n.d.-a).

2.3.4 Summary of deep learning models

To summarize, deep learning models are preferred by many people as they can handle unstructured data effectively and have great scalability and flexibility. Multiple components and layers like convolutional, pooling, recurrent, and normalization form together to make the deep learning model a great tool for processing different kinds of data. The popular models in deep learning models are the Convolutional Neural Networks (CNN) that is used for image tasks, Transformer-based models which are used to carry out natural language tasks as well and Recurrent Neural Networks (RNN) which are used for multiple tasks like speech recognition, sentiment analysis as well as language modelling. Hence, these deep learning models work together to help in the advancement of different fields.

2.4 Options for lipreading detection

With the advancement of technologies, there are more options for selecting suitable method for lipreading detection. As what has been discussed in section 2.2 and section 2.3 about the capability of traditional machine learning models and deep learning models, these techniques can be applied in the lipreading field as well.

There are two stage approach of lip reading detection using traditional machine learning, which the first step of feature extraction and the second step of classification. In the feature extraction stage, the mouth region of interest (ROI) is extracted to obtain the features in order to encode into feature vectors with equal length. This can be done by using Discrete Cosine Transform (DCT), it separates the images into parts based on Principal Component Analysis (PCA) to reduce the complexity of data (Hao et al., 2020). DCT is a transformation technique whereas PCA is a type of unsupervised machine learning algorithm. These two DCT technique and PCA machine learning algorithm are the commonly used feature extractors. After extracting the features, the next stage pass the features into Hidden Markov Models (HMM) for classification purposes.

Aside from that, the deep learning method can also be used in lipreading detection. Similar to the traditional machine learning approach, the initial deep learning approach is made up of two stages but the feature extraction stage is replaced using deep autoencoders and the classification stage is replaced using Long-Short Term Memory (LSTM) networks (Martinez B et al., 2020). In the new deep learning approaches for lipreading, an end-to-end trainable system is proposed. This system is made up of front-end and back-end neural network. Feature extraction is done by the convolutional layers or fully connected layer whereas classification is done by RNN or attention architectures (Weng & Kitani, 2019).

Researchers prefer to use deep learning approaches for lipreading detection nowadays because it ensures that the learned features have a strong relationship with the trained task (Weng & Kitani, 2019). Also, when using the traditional machine learning model, there is a problem with the usage of HMM as the classifier because it depends on the conditional independent assumption, causing the long-term dependencies to fail to be modelled (Xu et al., 2018).

Besides that, the manual feature extraction process in traditional machine learning methods can be overcome by deep learning using the machine-autonomous feature extraction way (Xing et al., 2023). The end-to-end network in deep learning helps to eliminate the manual feature engineering process. Another reason deep learning is preferred nowadays is because of the presence of large-scale datasets such as LRW and GRID datasets which will be discussed in section 2.6. Deep learning models have a higher capability in handling these large-scale datasets.

Therefore, with the higher capabilities and better performance of the deep learning approaches in lipreading detection, deep learning is preferred compared to the machine learning approach. Nevertheless, the approaches of traditional machine learning are still being cherished because they contribute to the achievements in lipreading today.

2.5 The existing methods in lipreading

Table 2.1: A summary of findouts related to lipreading based on the previous works by other researchers.

No.	Ref.	Number of citations	Dataset	Techniques used	Problems / Objectives	Hyperparameter	Achievements	Performance	Remarks
1	(Yargic & Dogan, 2013)	37	• Color names in Turkish language	<ul style="list-style-type: none"> • MS Kinect camera • MS Kinect Face Tracking SDK • Data Preprocessing 	<ul style="list-style-type: none"> • To develop lipreading application for education purpose of hearing- 	• -	<ul style="list-style-type: none"> • MS Kinect sensor is used to identify the points on the lips. Each angle of the points is then classified to identify the best angle features 	Classification success rate: <ul style="list-style-type: none"> • All angle features are used: 78.22% • 4 best angle features are used: 	MS Kinect camera is used to identify mouth feature instead of using model for

				<ul style="list-style-type: none"> • Lip activation detection • Word segmentation • KNN classifier 	impaired children			72.44% (Faster computation time)	feature extraction
2	(Assael et al., 2016)	465	• GRID	<p>LipNet</p> <ul style="list-style-type: none"> • STCNN • Bi-GRU • CTC 	<ul style="list-style-type: none"> • Current research on end-to-end trained models only perform word classification 	<ul style="list-style-type: none"> • Optimize Ada m • Mini-batches: 50 	<ul style="list-style-type: none"> • Sentences with varying lengths for both input and output are able to be processed 	<ul style="list-style-type: none"> • Sentence-level word accuracy: 95.2% • Unseen Speakers CER: 6.4% 	

					ation but not at the sentence level	• LR: 10^{-4}		• Unseen Speakers WER: 11.4% • Overlapp ed Speakers CER: 1.9% • Overlapp ed Speakers WER: 4.8%	
3	(Wand et al., 2016)	262	• GRID	• LSTM	• To apply a compact neural	• LR: 0.02 • Early stopp	• The architecture achieved better accuracy than the standard SVM	• Word accuracy: 79.6%	

					network architect ure in place of the complete visual speech recognit ion pipeline	ing with 10 epoch hs delay	classifier that applies conventional features	Accuracy on non-letter words is higher then letter words	
4	(Petridis S et al., 2017)	150	<ul style="list-style-type: none"> • OuluVS2 • CUAVE 	<ul style="list-style-type: none"> • LSTM • BLSTM (fusion of streams) 	<ul style="list-style-type: none"> • Traditional lipreading system consists of two 	<p>RBM training</p> <ul style="list-style-type: none"> • Epochs: 20 • Mini - 	<ul style="list-style-type: none"> • A model that can jointly learn to extract feature and perform classification is presented with great performance at the same time 	<ul style="list-style-type: none"> • Classification accuracy: 84.5% (OuluVS 2) 	

					stages which are feature extractio n and classific ation	batch size: 100 • LR: 0.1 (Ber noulli i- Bern oulli RB Ms) • LR: 0.00 1 (1 layer)	Classificatio n accuracy: 78.6% (CUAVE)	
--	--	--	--	--	---	--	--	--

5	(Shillingford et al., 2018)	194	• LSVSR (3886 hours video)	• Data processing pipeline • V2P model • speech decoder	• Previous deep learning models for lipreading do not tackle the setting of open-vocabulary continuation recognition	• Optimize Ada m Batc h size: 128 • LR: 10^{-4} • 1 st mom entum coefficien t: 0.9	• Largest VSR dataset is constructed • The model achieved significant reductions in WER with the use of larger vocabulary dataset • The model supports extension of vocabulary size without the need of retraining deep network	• WER: 40.9% • CER: 28.3% • PER: 33.6%	
---	-----------------------------	-----	----------------------------	---	--	--	---	--	--

						<ul style="list-style-type: none"> • 2nd mom entum coefficien t: 0.999 			
6	(Xu et al., 2018)	134	<ul style="list-style-type: none"> • GRID 	<ul style="list-style-type: none"> • LCANet • Cascaded attention-CTC decoder • 3D CNN • highway network • BiGRU 	<ul style="list-style-type: none"> • Existing approaches have high error rates when dealing with 	<ul style="list-style-type: none"> • Optimizer: Ada m • ILR: 0.0001 	<ul style="list-style-type: none"> • Partially overcomes the weakness of the conditional independence assumption of CTC • Faster convergence • Weight parameter in the loss function do not have to be tuned 	<ul style="list-style-type: none"> • CER: 1.3% • WER: 2.9% • BLEU: 97.4% 	-

				<ul style="list-style-type: none">• Preprocessing• Data augmentation	wild data applied models such as CTC and attention-based seq2seq, too emphasis on local information and too	<ul style="list-style-type: none">• Batch size: 64• Epochs: 50	<ul style="list-style-type: none">• Output is directly generated from the softmax output		
--	--	--	--	---	---	---	--	--	--

					flexible in forecasti ng proper alignme nt, respecti vely				
7	(Abrar et al., 2019)	11	• MIRACL- VC1	• VGG Net • TensorFlow to support running model in app	• The use of deep learning algorith ms has not been widely adopted	• Lear ning algor ithm: SGD • M ini batch	• Prototyped an Android app that can execute the model in real time on any smartphone using cloud computing	• Training accuracy: 94.86% • Validatio n accuracy: 93.82% • Testing accuracy:	

8	(Koum paroulis & Potami anos, 2019)	11	• TCD- TIMIT corpus	• MobiLipNet s (visual feature extraction) • TDNN (temporal modeling)	• Deep architect ures' computa tional burden h as gone	• -	• High computationally efficient and accurately recognised architectures are suggested	• MobiLip NetV2 contains 106 times les s paramete	60.00% (Lower because testing data contains speakers that do not seen in training)

				<ul style="list-style-type: none">• WFST (decoding)• ResNet-10 (face detection)	<p>neglecte d</p> <ul style="list-style-type: none">• Deep learning models are less efficient to compute and store			<p>rs and is 37 times quicker than ResNet, resulting in just a 0.07% decrease in absolute WER</p> <ul style="list-style-type: none">• MobiLip NetV2 has 4.27% higher in absolute	
--	--	--	--	--	--	--	--	--	--

								WER than 3D- CNN, 20 times smaller in size, and 12 times higher computat ional efficienc y	
9	(Lu & Li, 2019)	76	<ul style="list-style-type: none"> • A self-created dataset of unprofessional English 	<ul style="list-style-type: none"> • CNN • Attention-based LSTM 	<ul style="list-style-type: none"> • To improve the performance of 	<ul style="list-style-type: none"> • Batch size: 50 	<ul style="list-style-type: none"> • The extracted mouth ROIs exhibit great resilience and fault tolerance 	<ul style="list-style-type: none"> • Accuracy : 88.2% (3.3% greater than 	Dataset is not using professional pronunciation

			pronunciation of number from 0 to 9	general CNN-RNN	<ul style="list-style-type: none"> • LR: 0.00 • ROI: 224 x 224 pixel s 	<ul style="list-style-type: none"> • The link between the mouth ROIs in video frames is connected and enhanced 	general CNN-RNN)	tion, hence it is more suitable for actual application
10	(Margam et al., 2019)	19	<ul style="list-style-type: none"> • Grid dataset • Indian English dataset (In-En) (self-collected) 	<ul style="list-style-type: none"> • YOLOv2 (lip detection) • 3D-2D-CNN-BLSTM • BLSTM-HMM • w-CTC 	<ul style="list-style-type: none"> • To improve performance 	<ul style="list-style-type: none"> • Optimize: Ada m • LR: 0.00 • Batch 	<ul style="list-style-type: none"> • The proposed architecture achieved better performance than the state-of-the-art models 	<ul style="list-style-type: none"> • 3D-2D-CNN-BLSTM w-CTC has 55.1% better performance than LCANet

				<ul style="list-style-type: none">• ch-CTC• Feature duplication	size: 32		(Grid seen test set) <ul style="list-style-type: none">• 3D-2D-CNN-BLSTM w-CTC has 24.5% better performance than LipNet (Grid unseen test set)• 3D-2D-CNN-	
--	--	--	--	--	-------------	--	---	--

								BLSTM w-CTC has 25.0% better performa nce than 3D-2D- CNN BLSTM- HMM (In-En)	
11	(Mesbah et al., 2019)	70	<ul style="list-style-type: none"> • AVLetters • OuluVS2 • BBC LRW 	<ul style="list-style-type: none"> • HCNN 	<ul style="list-style-type: none"> • Models have high computational 	<ul style="list-style-type: none"> • AVL etters - Epochs: 5000 	<ul style="list-style-type: none"> • The model aids in gaining training time by decreasing the dimensionality of video images 	<p>Accuracy:</p> <ul style="list-style-type: none"> • AVLetter s: 59.23% • OuluVS2 : 93.72% 	

					require ments	- Batch size: 520 • Oulu VS2 - Epochs: 1000 - Batch size: 1070 • BBC LR W - Epochs: 105 • - Batch	<ul style="list-style-type: none">The model integrates the Hahn moments as the first layer, greatly reducing the dimensionality of images and lowering the computing cost	<ul style="list-style-type: none">BBC LRW: 58.02%	
--	--	--	--	--	------------------	---	---	---	--

						h size: 300			
12	(Saitoh & Kuboka wa, 2019)	7	• SSSD	<ul style="list-style-type: none"> • LiP25w web application • Use dlib method for facial feature points detection • motion-based feature extraction • GRU (recognition) 	<ul style="list-style-type: none"> • Lack of practical system of lipreading 	• -	<ul style="list-style-type: none"> • Developed word-level lipreading web application • The application able to recognize words or phrases that have been registered in the database 	<ul style="list-style-type: none"> • Accuracy : 73.4% 	

				<ul style="list-style-type: none"> TensorFlow (build and train GRU) 					
13	(Weng & Kitani, 2019)	90	<ul style="list-style-type: none"> LRW Kinetics (for 2nd round pre-training) 	<ul style="list-style-type: none"> Frontend: I3D (3D-CNN) Backend: Bi-LSTM Two-stream (optical flow stream + grayscale video stream) 	<ul style="list-style-type: none"> To replace the state-of-the-art model frontend shallow 3D CNNs + deep 2D CNNs with deep 3D 	<ul style="list-style-type: none"> - 	<ul style="list-style-type: none"> The classification accuracy is improved It is discovered that utilising optical flow input alone can yield performance that is comparable to that of employing grayscale video as input alone. As a result, the two inputs are combined to create a two-stream network that further enhance the performance 	<ul style="list-style-type: none"> Validation accuracy: 84.11% Testing accuracy: 84.07% 	

					CNNs (2 streams) • No previous work make use of optical flow input which is proved to be useful in completing video tasks				
--	--	--	--	--	--	--	--	--	--

14	(Feng et al., 2020)	72	<ul style="list-style-type: none"> • LRW • LRW-1000 	<ul style="list-style-type: none"> • ResNet-18 • BiGRU • Squeeze-and-Extract module • Data augmentation: MixUp • Cosine learning rate scheduling • Label smoothing • Word boundary 	<ul style="list-style-type: none"> • Only source code or brief descriptive on of training strategies were given • Many deep learning models customize their own data 	<ul style="list-style-type: none"> • Optimize r: Ada m • ILR: 3e – 4 • Weight ht • Many decay: 1e – 4 • Batch size: 32 (Sing 	<ul style="list-style-type: none"> • Factors that can improve lipreading models are analysed and better performance is achieved • Reduced temporal jittering in face video • New basic training pipeline is built 	<ul style="list-style-type: none"> • Accuracy for LRW: 88.4% • Accuracy for LRW-1000: 55.7% 	-
----	---------------------	----	---	---	--	---	--	---	---

					processing or training procedure	le GPU)			
						<ul style="list-style-type: none">• Mini mal learning rate: 1e - 6• Loss function: cross - entropy			

15	(Fengh our et al., 2020)	54	• BBC LRS2	Overall system performs 2 tasks using 2 neural network architectures: <ul style="list-style-type: none"> • 1st: viseme classifier • 2nd: word detector 	• The system must employ a restrictive number of visemes as classes to lipread sentence(s) which includes a broad variety	Viseme classifier: • Optimize Ada m ILR: 10^{-3} Epochs: 2000	• The suggested model is quite resilient to changes in illumination levels. The system lipreads using visemes rather than words or ASCII letters, hence fewer classes are needed overall, which reduces computational bottlenecks. • Words that did not appear during the training phase can be classified using a viseme-based lip reading technique	• VER: 4.6% • CER: 23.1% • WER: 35.4% • SAR: 33.4% • CPU time: 37 hours	
----	-----------------------------------	----	------------	--	---	---	--	---	--

					of vocabulary and identify words that might not have been included in system training				
16	(Martin ez B et al., 2020)	272	<ul style="list-style-type: none"> • LRW • LRW-1000 	<ul style="list-style-type: none"> • ResNet-18 • MS-TCN • Variable-length 	<ul style="list-style-type: none"> • Training time costs 3 weeks 	<ul style="list-style-type: none"> • Epochs: 80 • Optimize 	<ul style="list-style-type: none"> • Simplified training process • Enhanced model's performance 	<ul style="list-style-type: none"> • Classification accuracy for LRW: 85.3% 	-

				augmentatio n	GPU- time	r: Ada		• Classific ation accuracy for LRW- 1000: 41.4%	
				• Cosine scheduler		• Initia l learn ing rate (ILR): 3e - 4			
				• Pre-training 10% hardest words		• Weig ht deca y: 1e - 4			
						• Batc h			

						size: 32			
17	(May et al., 2021a)	109	<ul style="list-style-type: none"> • LRW • LRW-1000 	<ul style="list-style-type: none"> • ShuffleNet v2 • DS-TCN • Self-distillation • Knowledge distillation • Cosine annealing schedule • Data augmentation 	<ul style="list-style-type: none"> • Deep learning models • Cost • Computational cost • Limited computational capacity 	<ul style="list-style-type: none"> • Optimizer: Ada mW • Epochs: 80 • ILR: 3e - 4 • Weight decay: 1e - 4 	<ul style="list-style-type: none"> • Lightweight models are trained with no significant reduction in performance • Reduce computational cost by 8.2 times and number of parameters by 3.9 times 	<ul style="list-style-type: none"> • Accuracy for LRW: 88.5% • Accuracy for LRW-1000: 46.6 % 	-

						<ul style="list-style-type: none"> • Mini-batch : 32 			
18	(Tsouro unis et al., 2021)	21	<ul style="list-style-type: none"> • LRGW-10 (Greek language simulating a clinical scenario) • LRW-500 	<ul style="list-style-type: none"> • ALSOS • ResNet-18 • CNN • MS-TCN • FC 	To examine the advantages of switching between spatial and spatiotemporal convolution s in order to extract useful characteristi	<ul style="list-style-type: none"> • Optimize r: Ada mW • LR: 0.00 03 • Weigt deca 	<ul style="list-style-type: none"> • The module which uses the advantages of 2 words by alternating 2D and 3D convolutions enables both spatial and spatiotemporal information encoded directly on image space, and its incorporation into the lipreading system enh 	<ul style="list-style-type: none"> • Classification accuracy: 56.3% (LRGW-10) • Classification accuracy: 87.0% (LRW-500) 	

					cs from the lipreading	0.00 01	ances the performance sequences		
19	(Lu et al., 2022)	121	A group of fruit names	<ul style="list-style-type: none"> • Triboelectric sensors installed on mask • dilated RNN model (prototype learning) 	<ul style="list-style-type: none"> • Noncontact visual methods are affected by things obscuring them, head motion, light intensity, and 	Epochs: 500	<ul style="list-style-type: none"> • Muscle motion captured by non-invasive and contact sensors can address issues encountered by the vision-based solutions <p>Number of model parameters are reduced and training efficiency is improved</p>	<ul style="list-style-type: none"> • Test accuracy: 94.5% 	

					facial angles <ul style="list-style-type: none">• Lip motion dataset are limited				
20	(Sheng et al., 2022)	24	<ul style="list-style-type: none">• LRW• LRS2• LRS3	<ul style="list-style-type: none">• ASST-GCN• C3D_ResNe t18• Transformer• Two-stream	Currently available deep learning models for lip reading concentrate on examining the appearance	<ul style="list-style-type: none">• Optimize Ada m• LR: 10^{-6}	The suggested approach has the ability to automatically learn temporal and spatial information from videos	<ul style="list-style-type: none">• Accuracy : 85.7% (LRW)• CER: 36.2% (LRS2)• WER: 55.7% (LRS2)	

					and optical flow of videos, but they do not fully explore the properties of lip movements			<ul style="list-style-type: none"> • CER: 42.9% (LRS3) • WER: 62.7% (lrs3) 	
21	(HuggingFace, n.d.-b)	-	EGCLLC (enhanced version of GRID corpus)	<ul style="list-style-type: none"> • LipCoordNet • Input: image sequence and coordinates sequence • Bi-GRU 	<p>Improve sentence-level predictions' precision by adding geometric context to</p> <ul style="list-style-type: none"> • Total epoch: 51 • Training hardware: 	<ul style="list-style-type: none"> • Dual input system (raw image sequences + lip landmark coordinates) • Enhanced spatial resolution via detailed landmark tracking • Outperforms LipNet and PyTorch 	<ul style="list-style-type: none"> • WER: 1.7% • CER: 0.6% • validation dataset loss: 0.0256 		

					the LipNet model	NVI DIA GeF orce RTX 3080 12G B	implementation of LipNet		
22	(Nemani et al., 2023)	16	LIPAR (customized dataset based on MIRACL-VC1)	<ul style="list-style-type: none"> • Data preprocessing • Model training and validation (3D-CNN) • Android deployment (Cloud 	<p>The majority of proposed solutions do not have deployed technologies to carry out VSR</p>	-	<ul style="list-style-type: none"> • Word-level speaker independent VSR is performed • a prototype of an autonomous smart mobile application is developed for edge devices without the need for human involvement 	<ul style="list-style-type: none"> • Training accuracy: 80.2% • Testing accuracy: 77.9% 	

				servers' hardware specificatio ns: Standard A2v2)					
--	--	--	--	---	--	--	--	--	--

Table 2.1 lists the summary of findings collected from the papers written by the researchers about the existing methods in lipreading. In this project, the papers chosen to be studied are majority published from the year 2019 until 2023. This is because it is understandable that novel approaches with better performance will be proposed associated with the rapid advancement of technology, therefore focus should be placed on more recent published papers. However, some classic papers that have been published for years but have received many citations from other researchers are still being studied because these papers contribute theories, methodologies, or findings that are significant to the research areas.

Most of the paper's objectives are to propose their own lipreading model that can achieve performance better than the state-of-the-art result while some of the purposes are to resolve the computational cost and computational capacity issue. The researchers have included their study of literature review as known as related works in their own papers so that they can compare their works with others to understand the performance of their works. Some common datasets used by the researchers include GRID, LRW, OuluVS2, LRS2, and MIRACL-VC1. Some researchers choose to prepare and use their own dataset instead of the publicly available datasets. For the performance evaluation metrics, researchers choose to use classification accuracy, VER, WER, CER, PER, BLEU, SAR, training accuracy, validation accuracy, and testing accuracy in measuring their proposed models' performance. Besides that, the hyperparameters such as epoch, learning rate, batch size, and optimizer used in each model vary according to the selection of the researchers, however, the optimizer chosen by them are either Adam or AdamW.

Although each proposed models have their own architecture design and choice of networks, overall, the proposed models follow the lipreading detection steps of data preprocessing to extract mouth ROIs and continue with feature extraction and classification to obtain the output which is the decoded lip result. It is identified that the proposed models perform the classification in different forms, initially, isolated speech segments are classified in the form of digits and letters, and after that moving towards word classification, sentence-level sequence prediction, and classification based on phonemes and visemes. LipNet model is a great example of a lipreading model performing classification at the

sentence-level using an end-to-end trainable approach (Assael et al., 2016). From the table of research, it is identified that the feature extraction tasks are commonly performed by CNN, 3D-CNN, and ResNet whereas the classification tasks are commonly performed by LSTM, Bi-LSTM, attention-based LSTM, TCN, and Bi-GRU.

In this project, the lipreading model chosen to be used in achieving the lipreading task is the LipCoordNet model. It is an advanced version of the LipNet model with the integration of lip landmark coordination to form a dual input model. Initially, the LipNet model was the target model that was aimed to be applied. This is because, among the papers studied, it is identified that LipNet is a famous model that has been studied by many other researchers. Many of the researchers compared their own model with the performance of the LipNet model. The paper of LipNet also achieved a high number of citations from other researchers. Based on the last knowledge updated in November 2024, the LipNet paper has been cited by 465 other scholarly works in Google Scholar, indicating a high impact within the academic research community of speech recognition. However, the reason for choosing LipCoordNet as the model in this project is because it is identified that it outperforms the LipNet model. The LipCoordNet model has lower values of CER and WER for overlapped speakers compared to the LipNet model. The evaluation metrics results of LipNet and LipCoordNet are shown in Table 2.2.

Table 2.2: The CER and WER values of LipNet and LipCoordNet model. The LipCoordNet model has a lower CER and WER value for the overlapped speakers compared to LipNet.

Scenario	Image Size (W x H)	CER	WER
Unseen speakers (LipNet)	100 x 50	6.4%	11.4%
Overlapped speakers (LipNet)	100 x 50	1.9%	4.8%
Overlapped speakers (LipCoordNet)	128 x 64	0.6%	1.7%

Modified from Source: Assael et al. (2016) and HuggingFace (n.d.-b)

Another reason for choosing the LipCoordNet model is considered from the practical perspective, which is because of its ease of implementation.

Pretrained models are targeted to be found in this project so that extensive training starting from the beginning can be avoided. However, among the pre-trained models found, many of them are outdated as they are using the older version of dependencies. This problem may hinder the compatibility of the model with the web application that will be developed in this project. Therefore, the LipCoordNet model is chosen as this project model.

2.6 Dataset Overview

According to the review in section 2.5, GRID and LRW are the two most popular datasets used in the research community. In this section, the characteristics of these 2 datasets will be studied in detail to gain deeper insights.

2.6.1 GRID Dataset

The GRID audio-visual dataset is commonly used in lipreading tasks. It is made up of audio and video recordings which contain 18 male speakers and 16 female speakers. It has 34 speakers speaking 1000 sentences each, making up 34000 sentences in total, and it created the data with recording length of 28 hours (Assael et al., 2016).

This dataset is made up of 51 unique vocabularies, each of which forms the category of letter, digit, command, colour, preposition, and adverb respectively. 6-word structure of command (4) + colour (4) + preposition (4) + letter (25) + digit (10) + adverb (4) are the fixed sentence structures spoken (Wand et al., 2016). The numbers specified in the parentheses are the number of alternative words available to be selected randomly to form the sentence. For instance, “set red with m six please” is a sentence formed with the 6-word structure (Margam et al., 2019). The letter W takes longer to pronounce, thus it's excluded from the sentence (Wand et al., 2016). Therefore, in total, 64000 sentences can be formed using these 51 words by using the calculation formula of ${}^4C_1 \times {}^4C_1 \times {}^4C_1 \times {}^{25}C_1 \times {}^{10}C_1 \times {}^4C_1$ (Xu et al., 2018). The duration of each sentence is 3 seconds, and it costs 25 frames per second (fps), therefore 75 frames make up each of the sentences.

2.6.2 LRW Dataset

Lip Reading in the Wild (LRW) dataset is a large word-level lipreading dataset made up of classes with 500 unique English words collected from BBC programmes such as talk shows and news. In this dataset, there are up to 1000 utterances of words spoken by the speakers. There are 29 frames in each utterance, which is equal to 1.16 seconds for the length of each utterance (May et al., 2021b). Figure 2.1 shows the sample from the LRW dataset in which a male and a female speaker utter the word “about”. This dataset is composed of short clips of the videos, of which 488766 of them are for the training set, 25000 are for the validation set, and 25000 are for the testing set (Weng & Kitani, 2019).

One of the characteristics of the LRW dataset which causes it to be known as a challenging dataset is that it has high variability in its speakers, head pose, and illumination (Martinez B et al., 2020). Another challenge is this dataset contains pairs of words that have similarities in their visemes. This similarity is caused by the usage of 23 pairs of singular and plural forms of nouns as well as 4 pairs of present and past tenses of verbs. An example of the singular and plural forms word pair that appear in this dataset is “benefit” and “benefits”. Whereas for the present and past tenses word pair, the example is “allow” and “allowed” (Weng & Kitani, 2019). This challenge increases the confusion level between the words to be recognized. However, the most challenging part of this dataset is the 500 target words to be recognized are surrounded by other irrelevant words (Weng & Kitani, 2019).



Figure 2.1: LRW dataset sample of the mouth region extracted from of a male speaker (top row) and a female speaker (bottom row) pronouncing the word ‘about’.

Source: Mesbah et al. (2019)

2.6.3 Summary of Datasets

We have studied 2 commonly used lipreading datasets which are the GRID and LRW datasets. We choose to study these 2 datasets due to their popularity in being applied by other researchers to their existing works in lipreading. From our studies, we have collected some information regarding the physical characteristics and challenges of the datasets. Some of the physical characteristics include the number of data in the datasets, the duration or frame of the data, the number of speakers, the structure of the data, as well as the type of the data. We believe that the previous researchers chose to use either of these datasets because they would like to address certain complexity and challenges of these two datasets by using their proposed models and approaches. Therefore, the choice of the dataset to be used for training and testing the lipreading model has to be considered based on the ability of the chosen model to address the complexity of the dataset.

2.7 Evaluation Metrics

Evaluation metrics are some quantitative measures applied in deep learning models to assess their performance. By having the result of evaluation metrics, a comparison can be made with the state-of-the-art models to identify the strengths and weaknesses of the applied model in order to make further improvements in the future. In this section, 3 commonly used evaluation metrics in lipreading models will be identified and discussed.

2.7.1 Metrics to Measure Lipreading Model Performance

The performance of the lipreading model can be measured by various metrics such as classification accuracy, word error rate (WER), character error rate (CER), viseme error rate (VER) and bilingual evaluation understudy (BLEU). In this paper, the 3 most common evaluation metrics to measure lipreading model performance will be chosen to be discussed in detail. Therefore,

according to the existing methods in lip reading in section 2.5, the classification accuracy, WER, and CER are chosen to be studied.

Classification accuracy is the ratio of the quantity of correct predictions to the total input sample count. Its simplicity in the calculation, straightforwardness in representing the model performance in a single number, and universal applicability in various contexts cause many researchers interested in applying this metric in their model performance evaluation (Furbush, 2021). The equation below shows the formula to calculate classification accuracy (Google for Developers, n.d.).

$$\text{Classification accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of input samples}} \quad (2.1)$$

In binary classification context, the formula to calculate classification accuracy can be computed in terms of true positive (TP), false positive (FP), false negative (FN), and true negative (TN). The formula is as shown in equation below (Google for Developers, n.d.):

$$\text{Classification accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.2)$$

where

TP = true positive

TN = true negative

FP = false positive

FN = false negative

Models aim to achieve more TP and TN because they represent that the predicted value is the same as the actual value. FP indicates that the value is expected to be negative but it is predicted as positive. FN is also undesirable because the actual value is supposed to be positive but the model predicts it as negative. Overall, a higher accuracy value is desired because it indicates a better performance of the model as the model made more correct predictions.

Apart from measuring lipreading model performance based on classification accuracy, error rate (ER) is also suitable for measuring the model

performance. This metric is the complement of accuracy. In the context of speech recognition, CER and WER are especially useful because they are the error rate metrics measured by computing the overall edit distance to evaluate the accuracy of the model (Fenghour et al., 2020). A comparison of actual speech with decoded speech is done to determine the misclassifications. The formula used to calculate CER and WER are shown as equations below (Fenghour et al., 2020). The computation of WER is similar to CER, it is just that it replaces the computation from character level to word level.

$$CER = \frac{C_S + C_D + C_I}{C_N} \quad (2.3)$$

where

CER = character error rate

C = characters

S = substitutions error

D = deletions error

I = insertions error

C_N = total number of characters in reference

$$WER = \frac{W_S + W_D + W_I}{W_N} \quad (2.4)$$

where

WER = word error rate

W = words

S = substitutions error

D = deletions error

I = insertions error

W_N = total number of words in reference

A smaller CER value indicates a higher prediction accuracy of the lipreading model. Similarly, the smaller the value of WER, the higher the prediction accuracy of the lipreading model (Xu et al., 2018). Therefore, the model will aim to achieve CER and WER values to be as small as possible.

2.7.2 Summary of Evaluation Metrics

In short, due to its simplicity and effectiveness, Classification accuracy is the most commonly used metric that is widely applicable to many different contexts. ER is the complement of accuracy, in which CER and WER are the metrics specifically applied in the speech recognition context. A high value of classification accuracy and a low value of CER and WER are pursued in order to achieve a better performance model.

2.8 System Usability Scale (SUS)

System Usability Scale is a reliable evaluation tool invented by John Brooke in the year 1986, which is used to evaluate the usability of products and services such as web applications, mobile applications, software, and others (usability.gov, n.d.). In this project, SUS is selected as the tool to evaluate the usability of the lipreading web application. Below section are the guidelines for using this approach and the reasons for choosing it .

2.8.1 Guidelines in Obtaining SUS Score

This subjective assessment of usability is done by distributing a questionnaire which consists of 10 questions to the respondents. In the questionnaire, the respondents are required to choose their answers from the 5 response options ranging from 0 to 4, which represent the meaning of “strongly disagree” to “strongly agree” respectively (usability.gov, n.d.).

After collecting the responses from the respondents, the SUS score has to be computed. Since there are 10 questions in the questionnaire, the calculation is done by subtracting the score filled in by respondents with 1 for questions 1, 3, 5, 7, and 9. For questions 2, 4, 6, 8, and 10, the calculation is done by subtracting 5 with the score filled in by respondents. When the calculation is completed, an addition should be made to each computed score from each question. Next, the sum of the value should be multiplied by 2.5 in order to obtain the SUS score (Brooke, 1995). Figure 2.2 shows an example of the calculation of the SUS score.

						Strongly disagree						Strongly agree
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>							4
1.	1	2	3	4	5							
2.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>							1
3.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							1
4.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							4
5.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							1
6.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							2
7.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							1
8.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>							1
9.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>							4
10.	<input type="checkbox"/>	<input checked="" type="checkbox"/>						3				

Total score = 22

SUS Score = $22 * 2.5 = 55$

Figure 2.2: Sample of the SUS questionnaire containing the guideline in calculation of SUS score. The tick symbols within the scale of each questions are the sample of user's answer. The numbers written on the rightmost part of the questionnaire are the computed score for each questions. The value 22 represents the total score which is the sum of all computed scores whereas the value 55 is the SUS score that is done by multiplying 2.5 with total score.

Source: Brooke (1995)

2.8.2 Interpretation of SUS Score

The SUS score ranges from 0 to 100. After computing and obtaining the SUS score, the usability of the product or service can be compared with the grade ranking shown in Figure 2.3. A higher score is more desirable as it means better usability of the products or services. A lower score means that the products or services have to be improved in terms of their usability.

There are several approaches that can be used in interpreting the SUS score, either using adjective ratings, grade scales, or acceptability ranges. According to research, adjective ratings are highly correlated with the SUS score. It rates the usability by using some adjective words such as “worst imaginable”, “poor”, “good”, and so on. This approach helps to communicate the interpretation to the professionals who do not specialize in human factors (Bangor et al., 2009). The grade scale is similar to the traditional school grading system in which “A” represents a superior performance in usability while “F” represents a failing performance. For the acceptability ranges approach, it evaluates the usability in terms of “acceptable” and “not acceptable”. The score region from 50 to 70 is a marginal range in which “marginal low” is better than “not acceptable” while “marginal high” is worse than “acceptable” (Sauro, 2018).

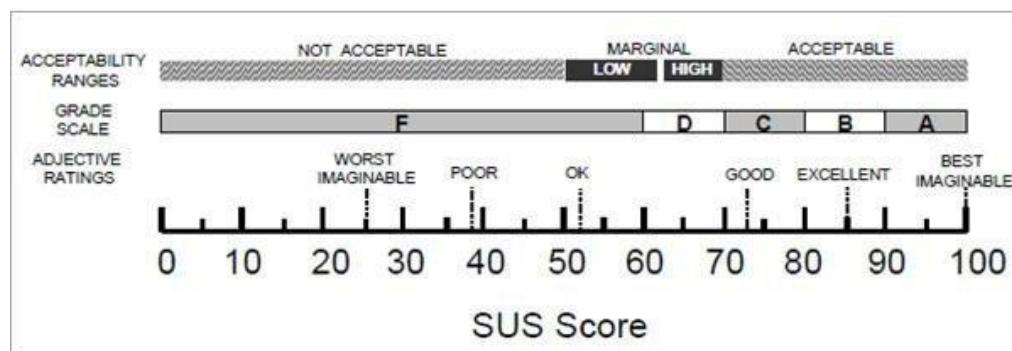


Figure 2.3: 3 different approaches to interpret SUS score, by using adjective ratings, grade scale, and acceptability ranges.

Source: Bangor et al. (2009)

2.8.3 Reason of Using SUS as Usability Measurement Tool

A requirement of having a generalized usability measurement tool is highly appreciated so that it can evaluate the system usability in many contexts. ISO 9241-11 recommends that usability can be measured in terms of the system effectiveness, efficiency, and satisfaction of users (Brooke, 1995). However, the metrics required to measure these classes are very precise and highly dependent on the context of use. The negative effect of this context-specificity is it raises the difficulties in comparing system usability among different systems. Hence, SUS is an optimum choice for assessing usability in a generalized manner because it supports cross-system comparison (Brooke, 1995).

Besides that, the evaluation of usability in industries is usually not cost-effective. Also, imagine if a user is required to fill in an evaluation form which lasts up to an hour after using a certain product or experiencing a certain service. It is undoubtedly that the user will feel frustrated. Therefore, a quick and simple but reliable measurement method which requires only less effort and cost to complete the whole evaluation is needed (Brooke, 1995). The SUS method satisfies the need for a speedy and low-cost assessment method which requires only 10 questions and some simple calculation steps to obtain the usability result. In addition, SUS is applicable to a small sample size of respondents to achieve a reliable result (usability.gov, n.d.).

2.8.4 Summary of SUS

SUS is an optimum method for measuring the usability of the web application in this project because of its generalised method of assessment, cost-saving and quick approach. A detailed explanation of the steps to conduct the usability test via the SUS approach is described in this section starting from preparing the questionnaire to interpreting the SUS score.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, the methodology flowchart of the system algorithm will be described. Next, the Work Break Structure (WBS) and Gantt Chart used for project planning and scheduling are included in this chapter. Besides that, the phases in the chosen software development methodology, which is Personal Extreme Programming (PXP) will be discussed. Also, the tools and technologies used in the development of the system are included in this chapter as well.

3.2 Summary of Model Workflow

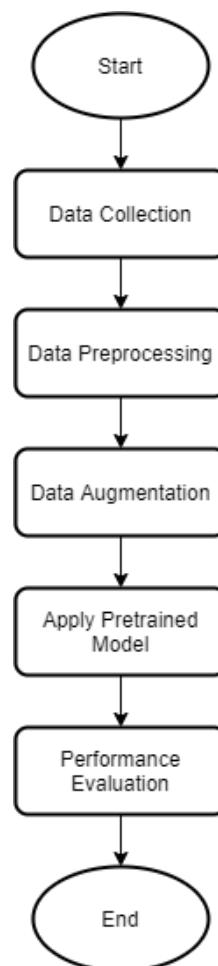


Figure 3.1: Flowchart of the workflow of LipCoordNet model.

Figure 3.1 shows the workflow of the LipCoordNet model, which is the deep learning model used in this project for lipreading purposes. The LipCoordNet model is an advanced version of the LipNet model which integrates the lip landmark coordinates as the second input. The reason for choosing the LipCoordNet model is because its performance is better than the LipNet model as well as its compatibility with the web application of the project. The details of the reason for selecting the LipCoordNet model can be found in the text description of section 2.5. The details of each step in the model workflow will be described in the subsections below.

3.2.1 Data Collection

Data collection is the acquiring of raw data from various sources. The collected data will be processed in the steps later so that it can be fed into the deep learning model for training.

The data collection step done in this lipreading model is to look for the JPEG image files stored in a directory. These image files are the frames of a video. OpenCV imread function is used to read the image files and store them in the array. The collected frames are ready for the lip coordinates extraction process.

```
def load_video(file, device: torch.device):
    if not os.path.exists("samples"):
        os.makedirs("samples")

    p = os.path.join("samples")
    output = os.path.join("samples", "%04d.jpg")
    cmd = "ffmpeg -hide_banner -loglevel error -i {} -qscale:v 2 -r 25 {}".format(
        file, output
    )
    os.system(cmd)

    files = os.listdir(p)
    files = sorted(files, key=lambda x: int(os.path.splitext(x)[0]))

    array = [cv2.imread(os.path.join(p, file)) for file in files]
```

Figure 3.2: Code snippet in charge of obtaining the video frames and storing them in array.

3.2.2 Data Preprocessing

Data preprocessing is the processing of the collected raw data so that it suits the deep learning model. Preprocessing of data is also necessary to eliminate the defects in the raw data such as missing values, noises, and unsuitable format.

The video frames collected are filtered to ensure they are all valid images. Next, the facial landmarks are detected and stored in a list. After that, face alignment is done on the face in the frame via transformation and the face region is resized to a size of 128 pixels width and 64 pixels height respectively. This is to ensure that the model will receive a consistent size of input data.

```

array = list(filter(lambda im: not im is None, array))

fa = face_alignment.FaceAlignment(
    face_alignment.LandmarksType._2D, flip_input=False, device=device.type
)
points = [fa.get_landmarks(I) for I in array]

front256 = get_position(256)
video = []
for point, scene in zip(points, array):
    if point is not None:
        shape = np.array(point[0])
        shape = shape[17:]
        M = transformation_from_points(np.matrix(shape), np.matrix(front256))

        img = cv2.warpAffine(scene, M[:2], (256, 256))
        (x, y) = front256[-20:].mean(0).astype(np.int32)
        w = 160 // 2
        img = img[y - w // 2 : y + w // 2, x - w : x + w, ...]
        img = cv2.resize(img, (128, 64))
        video.append(img)

```

Figure 3.3: Code snippet related to the face alignment and frame resizing.

In the following step, the x and y coordinates of lip landmarks are extracted. To extract the lip coordinates, it will require the face detector and shape predictor.

```

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(
    "LipCoordNet/lip_coordinate_extraction/shape_predictor_68_face_landmarks_GTX.dat"
)

```

Figure 3.4: Face detector and shape predictor used to extract lip coordinates.

However, before performing face detection, the image frames are resized to 600 x 500 pixels and converted into grayscale colour first.

```

def extract_lip_coordinates(detector, predictor, img_path, prev_rects):
    image = cv2.imread(img_path)
    image = cv2.resize(image, (600, 500))
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    rects = detector(gray)
    retries = 3
    while retries > 0:
        try:
            assert len(rects) == 1
            break
        except AssertionError as e:
            retries -= 1
            rects = detector(gray) # Retry face detection

    if retries == 0:
        if prev_rects is not None:
            print(f"Using previous rects for frame {img_path} due to face detection failure.")
            rects = prev_rects
        else:
            print(f"Skipping frame {img_path} due to face detection failure and no previous rects.")
            return None

    for rect in rects:
        # apply the shape predictor to the face ROI
        shape = predictor(gray, rect)
        x = []
        y = []
        for n in range(48, 68):
            x.append(shape.part(n).x)
            y.append(shape.part(n).y)
    return [x, y]

```

Figure 3.5: Code snippet related to the extraction of lip coordinates.

The lip coordinates are then normalized, and stored in the PyTorch tensor so that they are ready for use in the lipreading model. The lip coordinates are represented using points 48 to 67.

```

x_coords, y_coords = coords_result
normalized_coords = []
for x, y in zip(x_coords, y_coords):
    normalized_x = x / width
    normalized_y = y / height
    normalized_coords.append((normalized_x, normalized_y))
coords.append(normalized_coords)
coords_array = np.array(coords, dtype=np.float32)
coords_array = torch.from_numpy(coords_array)

```

Figure 3.6: Code snippet related to normalization of lip coordinates and storing them in PyTorch tensor.

The annotation files are also preprocessed by filtering out the unwanted labels including "SIL" and "SP" and then converting them to the uppercase format. The filtered annotations are then converted into numerical array representation. Padding is done on the array to ensure the length consistency.

3.2.3 Data Augmentation

Data augmentation is the process of enriching the training data by using the existing data to create new data. It is usually done to avoid overfitting so the model has better performance.

Horizontal flip of video frames is done as the data augmentation step of this model. First of all, a random floating point number within the range of 0 to 1 is generated. Next, if the random number has a value more than 0.5, it uses the array slicing method to reverse the width dimension of the images to achieve a horizontal flip. With the horizontal flipping, more images with different orientations are created, which can increase the variations in the training data.

Besides that, colour normalization is done on the video frames by dividing each colour value by 255 so that the values fall within the range between 0 and 1. This technique can reduces the effect of lighting and other color variations found in the dataset.

```

def HorizontalFlip(batch_img, p=0.5):
    # (T, H, W, C)
    if random.random() > p:
        batch_img = batch_img[:, :, ::-1, ...]
    return batch_img

def ColorNormalize(batch_img):
    batch_img = batch_img / 255.0
    return batch_img

```

Figure 3.7: Code of HorizontalFlip function and ColorNormalize function.

3.2.4 Apply Pre-trained Model

Application of pre-trained model such as LipCoordNet model in this project is done after completing data preprocessing steps. This model is specially designed for lipreading purposes, thus it is suitable to be applied in this project which aims to implement lipreading features in the web application. The training process of this LipCoordNet model is similar to the original LipNet model, but it is combined with the input of landmark coordinates. The training method of the model is done by using the pre-trained weights from the LipNet model as the training starting point. This pre-trained weight created a new layers for landmark coordinates for training (HuggingFace, n.d.).

Enhanced GRID Corpus with Lip Landmark Coordinates (EGCLLC), an enhanced version of GRID dataset that is added with the lip landmark coordinates dataset, are the training data used to train the model. The main reason to add this landmark is to assist and provide information related to the positional key points located around the lips (HuggingFace, n.d.). In short, this GRID alignments, lip images, and lip coordinate are the key component that made up EGCLLC dataset.

After setting the path to the input video frames and the directory to save the output video, the lipreading inference can be performed using the pre-trained model by running the inference.py file. The path to the input video frames is the location where the video containing lip movements to be predicted is stored. It will generate an output video which contains the predicted texts overlaying on the frames of the video afterwards.

3.2.5 Performance Evaluation

Performance evaluation is an important process of measuring the model's performance by using different evaluation metrics. Since the dataset used to train the pretrained model, LipCoordNet, is all Native speakers, it is suspected that the predicted transcription for Asian speaker may be less accurate than the Native speaker. Therefore, a further analysis and interpretation has been done on the Asian speaker and Native speaker.

In this project, the performance evaluation is done by calculating the Word Error Rate (WER) and Character Error Rate (CER) via a comparison between actual sentence and predicted sentence. An Asian speaker and a Native speaker have been invited to read out five sentences.



Figure 3.8: The screenshot of the videos of the Native speaker and the Asian speaker speaking the sentences.

The five actual sentences are listed in the table below. After collected the videos of them reading the sentences, the inference.py script has been used to predict the transcriptions.

Table 3.1: List of sentences spoken by an Asian speaker and a Native speaker and their respective prediction results.

Sentence	Actual Sentence	Predicted Sentence (Asian)	Predicted Sentence (Native)
s1	BIN BLUE BY Z NINE SOON	BIN BLUE BY Z NINE AGON	BLIN BLUE BY Z NINE AGAIN
s2	BIN GREEN WITH I ONE AGAIN	BIN GREEN WITH V OINE PAGAIN	BIN GREEN WITH I ONE AGAIN
s3	PLACE RED WITH D SEVEN AGAIN	PLACE RED WITH C SERO AGAIN	PLACE RED WITH C SEVEN AGAIN
s4	PLACE WHITE IN P SIX NOW	PLACE WHITED IN B SIX AGAIN	PLACE WHITE IN B SIX NOW
s5	SET BLUE AT T EIGHT PLEASE	SET BLUE AT E EIGHT PLEASE	SET BLUE AT X SIX PLEASE

Based on the predicted transcriptions, the word error rate (WER) and character error rate (CER) have been calculated by using PyTorch.

```
from torchmetrics.text import WordErrorRate, CharErrorRate
preds = ["BIN BLUE BY Z NINE AGON"]
target = ["BIN BLUE BY Z NINE SOON"]
wer = WordErrorRate()
cer = CharErrorRate()
print("WER: ", wer(preds, target))
print("CER: ", cer(preds, target))
```

Figure 3.9: Code to calculate the WER and CER. The predicted text and actual text should be updated with the sentences that want to be calculated for WER and CER.

After completed calculating all WERs and CERs, the results have been summarised in the table below.

Table 3.2: Results of the WER and CER for the five sentences spoken by an Asian speaker and a Native speaker. Average WER and CER have also been calculated and presented in the table.

	Asian Speaker					Ave rage	Native Speaker					Ave rage
Sent ence	s1	s2	s3	s4	s5		s1	s 2	s3	s4	s5	
WER	0.1 667	0.5 000	0.3 333	0.5 000	0.1 667	0.33 33	0.3 333	0 667	0.1 667	0.1 667	0.3 333	0.20 00
CER	0.0 870	0.1 154	0.1 429	0.2 917	0.0 385	0.13 51	0.2 174	0 357	0.0 417	0.0 923	0.1 74	0.09

Note: s = Sentence

Average WER and CER are calculated for both Asian speaker and Native speaker by taking the sum of their WER results or CER results divided by five. It is noticed that the Asian speaker has an average WER of 0.3333, which is higher than the average WER of the Native speaker, which is 0.2000. Also, the average CER of the Asian speaker is 0.1351, which is also higher than the average CER of the Native speaker, which is 0.0974.

Based on the result, it can be interpreted that the lipreading model can perform more accurate prediction for Native speakers than for Asian speakers. The lower WER and CER suggests that the lipreading model has better accuracy when interpreting the sentence spoken by the Native speaker. It is reasonable for the Asian speakers to have a lower WER and CER value because the LipCoordNet model is trained using the EGCLLC dataset in which the speakers of the dataset are all Native speakers. All of the speakers spoke English as their mother tongue language and encompassed the English accents (Cooke et al., 2006).

Therefore, an intention of training the LipCoordNet model with some Asian dataset has been made. To train the model, there are three types of data

that are required to prepare, which include the lip images, lip coordinates, and the alignments file.

The lip images are the video frames of the videos. In order to prepare the videos, 27 Asian speakers are invited to read out ten sentences and the videos are being recorded.



Figure 3.10: Folders containing the videos of the 27 speakers.

After collected the videos, video editing process has been done on the videos to crop out each sentence to become one video. Each of the videos are then being converted into video frames by using the online Video to JPG converter tool.



Figure 3.11: Sample of the video frames converted from a video of an Asian speaker reading a sentence.

Next, the second data to prepare is the lip coordinates, which is the landmark coordinates of the lips extracted from the captured videos. To generate the lip coordinates file, it will make use of the dlib library to use the face detector and shape predictor. The shape predictor used here is the shape_predictor_68_face_landmarks_GTX.dat pretrained model. This model will provide the x and y coordinates of the facial landmarks. Figure below shows

part of the code in lips_coords_extractor.py file in which it sets the shape predictor with the pretrained model and uses the shape predictor to obtain the coordinates.

```

def generate_lip_coordinates(speakers):
    file_path_sep = "\\"
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor(
        "shape_predictor_68_face_landmarks_GTX.dat"
    )
    for speaker in speakers:
        print("hi", speaker)
        videos = glob.glob(speaker + "/*")
        for video in videos:
            print(video)
            frames = glob.glob(video + "/*.jpg")
            vid = {}
            try:
                frames = sorted(
                    frames,
                    key=lambda x: int(x.split(file_path_sep)[-1].split(".") [0]),
                )
                for frame in frames:
                    retry = 3
                    while retry > 0:
                        try:
                            coords = extract_lip_coordinates(detector, predictor, frame)
                            break
                        except Exception as e:
                            retry -= 1
                            print("Error: ", video)
                            print(e)
                            print("retrying...")
                vid[frame.split(file_path_sep)[-1].split(".")[0]] = coords
            vid_path = video.split(file_path_sep)
            save_path = (
                LIP_COORDINATES_DIRECTORY
                + "\\"
                + vid_path[-2]
                + "\\"
                + vid_path[-1]
                + ".json"
            )
        )
    )

```

Figure 3.12: Code snippet in lips_coords_extractor.py file which uses shape_predictor_68_face_landmarks_GTX.dat pretrained model as the shape predictor to obtain lip landmark coordinates.

The lip landmark coordinates are saved in the json file format. Figure below shows the example of lip landmark coordinates json file, in which the keys represent the frame numbers whereas the values represent the x and y lip landmark coordinates.

```

0 bbb9s.json x
D:\backend\selftrain>LipCoordNet>lip.coordinates>s1> 1 bbb9s.json ...
1 [{"frame number": 309, 315, 321, 325, 330, 336, 342, 336, 338, 325, 328, 314, 312, 321, 325, 330, 339, 330, 325, 320}, {"214, 232, 213, 212, 211, 212, 214, 223, 226, 227, 226, 222, 214, 215, 216, 215, 215, 215, 219, 219, 219}], "y lip landmark coordinates"

```

Figure 3.13: Sample content in the lip landmark coordinates file.

The third data to prepare for training the LipCoordNet model with Asian speaker dataset is the alignments file. The alignments file is the file that contains information about the timing and alignment of words spoken in a video. Each line in the alignment file should represent the start and end time of a word

in the video. Figure below shows an example of the information required in the alignment file to train the LipCoordNet model. It consists of the start time of a word (in milliseconds), end time of a word (in milliseconds), and the word itself.

```

bbaf2n.align ×
D:\> EGCLLC > GRID_alignments
1 0 23750 sil
2 23750 29500 bin
3 29500 34000 blue
4 34000 35500 at
5 35500 41000 f
6 41000 47250 two
7 47250 53000 now
8 53000 74500 sil
9

```

Figure 3.14: Example of the information required in the alignment file to train the LipCoordNet model. It consists of the start time of a word (in milliseconds), end time of a word (in milliseconds), and the word itself.

There are 2 input files that are required to obtain the alignment output, which are the waveform audio file and the transcript text file. Since there are 27 speakers each speaking 10 sentences, there would be a total of 270 alignment files need to be created, which also means 270 waveform audio files need to be prepared. Although the online forced aligner tool by University of Wisconsin-Milwaukee has been tried out to obtain the alignment file, the downloaded output is in the .TextGrid file format. The .TextGrid file needs to be parsed into .align file format to support the model training. Also, the start time and end time in the .TextGrid file are displayed in the unit second. They have to be converted into the unit millisecond.

Figure 3.15: Screenshot of the output by the online forced aligner tool.

Source: University of Wisconsin-Milwaukee (n.d.)

	A	B	C	D	E	F	G	H	I	J
1	file	word	phoneme	time_1third	time_mid	time_2third	time_start	time_end	duration	F0_avg
2	sound	BIN	B	0.49138322	0.506349206	0.521315193	0.461451247	0.551247166	0.089795918	132.2926749
3	sound	BIN	IH1	0.581179138	0.596145125	0.611111111	0.551247166	0.641043084	0.089795918	154.678878
4	sound	BIN	N	0.68760393	0.710884354	0.734164777	0.641043084	0.780725624	0.13968254	155.9219534
5	sound	BLUE	B	0.810657596	0.825623583	0.840589569	0.780725624	0.870521542	0.089795918	137.4690339
6	sound	BLUE	L	0.880498866	0.885487528	0.89047619	0.870521542	0.900453515	0.029931973	153.4844069
7	sound	BY	UW1	0.980272109	1.020181406	1.060090703	0.900453515	1.139909297	0.239455782	184.6058124
8	sound	BY	B	1.179818594	1.199773243	1.219727891	1.139909297	1.259637188	0.119727891	115.1978605
9	sound	BY	AY1	1.356084656	1.40430839	1.452532124	1.259637188	1.548979592	0.289342404	106.3258024
10	sound	Z	Z	1.951398337	1.968027211	1.984656085	1.91814059	2.017913832	0.099773243	115.7778457
11	sound	Z	IY1	2.064474679	2.087755102	2.111035525	2.017913832	2.157596372	0.13968254	136.4905152
12	sound	NINE	N	2.267346939	2.277324263	2.287301587	2.24739229	2.307256236	0.059863946	121.9060628
13	sound	NINE	AY1	2.367120181	2.397052154	2.426984127	2.307256236	2.486848073	0.179591837	132.072029
14	sound	NINE	N	2.516780045	2.531746032	2.546712018	2.486848073	2.576643991	0.089795918	133.9936455
15	sound	SOON	S	2.65313681	2.69138322	2.72962963	2.576643991	2.806122449	0.229478458	147.2660806
16	sound	SOON	UW1	2.869312169	2.900907029	2.93250189	2.806122449	2.99569161	0.189569161	304.5609945
17	sound	SOON	N	3.005668934	3.010657596	3.015646259	2.99569161	3.025623583	0.029931973	0

Figure 3.16: A csv file showing the start time and end time of the words at the green colour columns. This csv file is downloaded together with the .TextGrid file when using the online forced aligner tool.

The major problem encountered in preparing the data is the tasks in obtaining 270 alignment files has exceeded the workload affordable by an individual developer within the stipulated time frame. The waveform audio file and transcript text file have to be uploaded one by one in order to obtain the alignment file. The alignment file also needs to be processed further before it can be used for training. The EGCLLC dataset used to train the LipCoordNet model is actually the GRID audio-visual sentence corpus but with further enhancement. However, this GRID audio-visual sentence corpus is collected and postprocessed by a group of researchers from the University of Sheffield (Cooke et al., 2006). Therefore, the effort that can be made by individual developer is limited.

However, without the alignment files, the training of the model will be directly impacted negatively. This is because without the accurate alignments, the model unable to learn the mapping between the speaker's lip movements and the spoken words. Without the alignment file, the training of the model failed to proceed. After considering the time limitation and the limited personal ability, the result of having the Asian speaker to have lower WER and CER values than the Native speaker becomes the limitation in this project.

3.3 Project Planning and Scheduling

3.3.1 Work Breakdown Structure (WBS)

0.0 Web Application Development for Lip Reading

1.0 Project Initialization

 1.1 Preliminary Planning

 1.1.1 Project Title Registration

 1.1.2 Study Project Background

 1.1.3 Define Problem Statement

 1.1.4 Identify Project Aim and Objectives

 1.1.5 Define Project Scope

 1.1.5.1 Define Target Users of Project

 1.1.5.2 Define Modules Covered in Project

 1.1.5.3 Define System Scope

 1.1.5.4 Define Project Development Tools and Technologies

 1.1.5.5 Identify Project Limitations

 1.1.6 Propose Project Solution

 1.1.7 Propose Project Approach

 1.2 Conduct Literature Review

 1.2.1 Study Concepts of Lipreading

 1.2.2 Study the Traditional Machine Learning Models

 1.2.3 Study the Deep Learning Models

 1.2.4 Study Lipreading Detection Options using Traditional and Deep Learning Methods

 1.2.5 Review Existing Similar Lipreading Algorithms

 1.2.6 Review Popular Datasets used in Lipreading

 1.2.7 Review Common Evaluation Metrics used in Lipreading

 1.2.8 Review System Usability Scale used for Usability Test

 1.3 Methodology and Work Plan

 1.3.1 Create Flowchart of Model Algorithm

 1.3.2 Develop Work Breakdown Structure (WBS)

 1.3.3 Develop Gantt Chart

 1.3.4 Explain Software Development Methodology

1.3.5 Identify Software Development Tools and Technologies

1.4 Project Planning and Design

1.4.1 Prepare Requirement Specification

1.4.1.1 Identify Functional Requirements of Lipreading Web Application

1.4.1.2 Identify Non-Functional Requirements of Lipreading Web Application

1.4.2 Design Use Case Diagram

1.4.3 Prepare Use Case Description

1.4.4 Create Initial Prototype

1.4.4.1 Prepare Model Algorithm

1.4.4.2 Design Web Application User Interface

2.0 System Development

2.1 System Architecture Design

2.2 Database Design

2.3 Implement Web Application

2.3.1 First Iteration: Apply Deep Learning Model

2.3.1.1 Iteration Initialisation

2.3.1.2 Design

2.3.1.3 Implementation

2.3.1.4 System Testing

2.3.1.5 Retrospective

2.3.2 Second Iteration: Develop Web Application Functionality

2.3.2.1 Iteration Initialisation

2.3.2.2 Design

2.3.2.3 Implementation

2.3.2.4 System Testing

2.3.2.5 Retrospective

2.3.3 Third Iteration: Integrate Deep Learning Model into Web Application

2.3.3.1 Iteration Initialisation

2.3.3.2 Design

2.3.3.3 Implementation

2.3.3.4 System Testing

2.3.3.5 Retrospective

3.0 System Testing

3.1 Unit Testing

3.2 System Testing

3.3 Usability Testing using System Usability Scale (SUS)

3.4 User Acceptance Testing

4.0 Project Closure

4.1 System Deployment

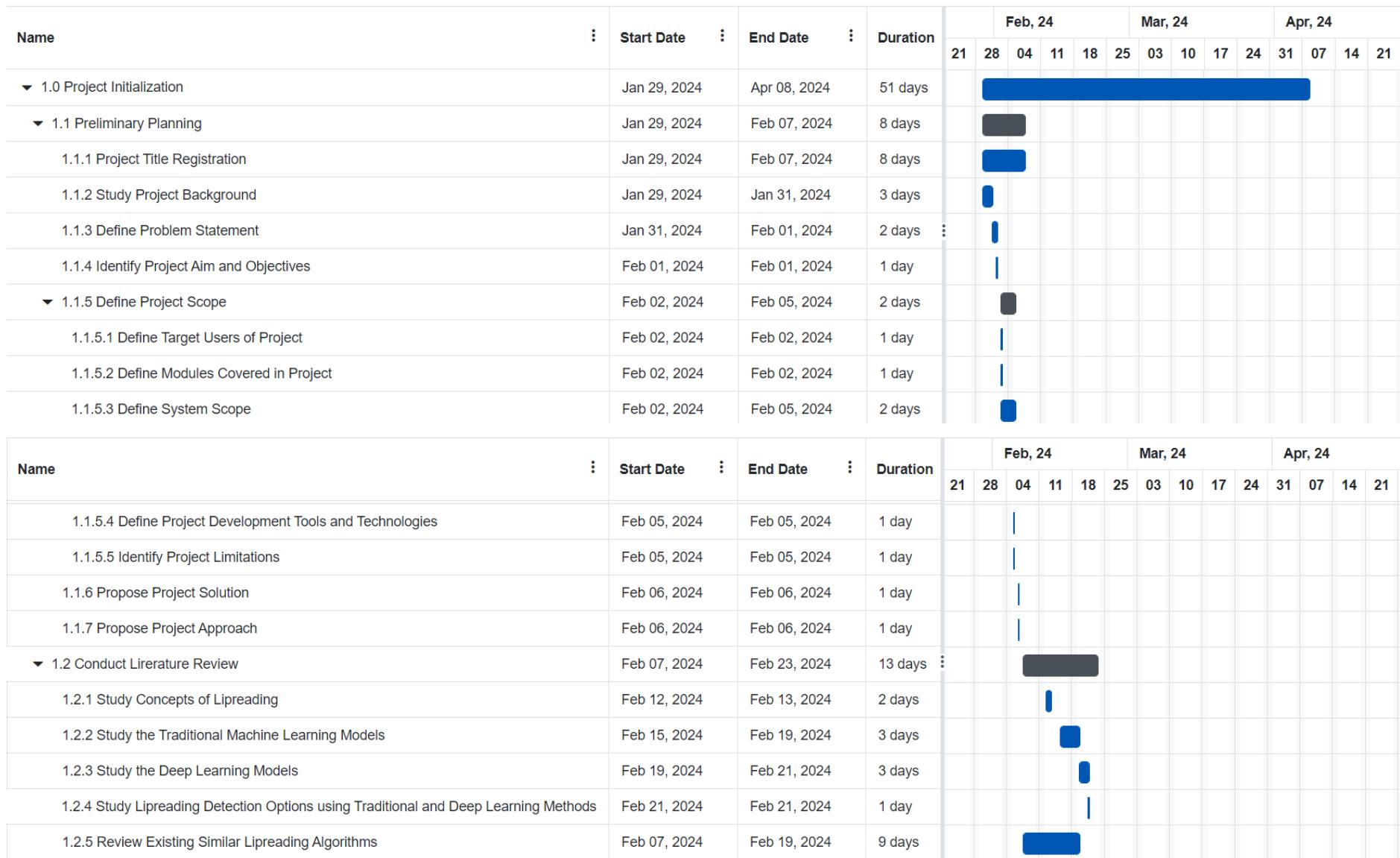
4.2 System Monitoring

4.3 Report Finalization

4.4 Review Lessons Learnt

3.3.2 Gantt Chart

Figure 3.17 shows the planned schedule of this project by using gantt chart.



Name	Start Date	End Date	Duration	Jun, 24		Jul, 24		Aug, 24		Sep				
				26	02	09	16	23	30	07	14	21	28	
▼ 2.0 System Development	May 27, 2024	Aug 14, 2024	58 days											
2.1 System Architecture Design	May 27, 2024	May 31, 2024	5 days	■										
2.2 Database Design	Jun 03, 2024	Jun 07, 2024	5 days		■									
▼ 2.3 Implement Web Application	Jun 10, 2024	Aug 14, 2024	48 days											
2.3.1 First Iteration: Apply Deep Learning Model	Jun 10, 2024	Jul 01, 2024	16 days											
2.3.1.1 Iteration Initialisation	Jun 10, 2024	Jun 10, 2024	1 day			■								
2.3.1.2 Design	Jun 10, 2024	Jun 14, 2024	5 days		■									
2.3.1.3 Implementation	Jun 13, 2024	Jun 28, 2024	12 days						■					
2.3.1.4 System Testing	Jun 10, 2024	Jul 01, 2024	16 days						■					
2.3.1.5 Retrospective	Jul 01, 2024	Jul 01, 2024	1 day							■				
2.3.2 Second Iteration: Develop Web Application Functionality	Jul 02, 2024	Jul 19, 2024	14 days						■					
2.3.2.1 Iteration Initialisation	Jul 02, 2024	Jul 02, 2024	1 day				■							
2.3.2.2 Design	Jul 02, 2024	Jul 04, 2024	3 days				■							
2.3.2.3 Implementation	Jul 03, 2024	Jul 18, 2024	12 days				■							
2.3.2.4 System Testing	Jul 02, 2024	Jul 19, 2024	14 days				■							
2.3.2.5 Retrospective	Jul 19, 2024	Jul 19, 2024	1 day					■						
2.3.3 Third Iteration: Integrate Deep Learning Model into Web Application	Jul 22, 2024	Aug 14, 2024	18 days											
2.3.3.1 Iteration Initialisation	Jul 22, 2024	Jul 22, 2024	1 day							■				
2.3.3.2 Design	Jul 22, 2024	Jul 24, 2024	3 days							■				
2.3.3.3 Implementation	Jul 24, 2024	Aug 13, 2024	15 days							■				
2.3.3.4 System Testing	Jul 22, 2024	Aug 14, 2024	18 days							■				
2.3.3.5 Retrospective	Aug 14, 2024	Aug 14, 2024	1 day								■			

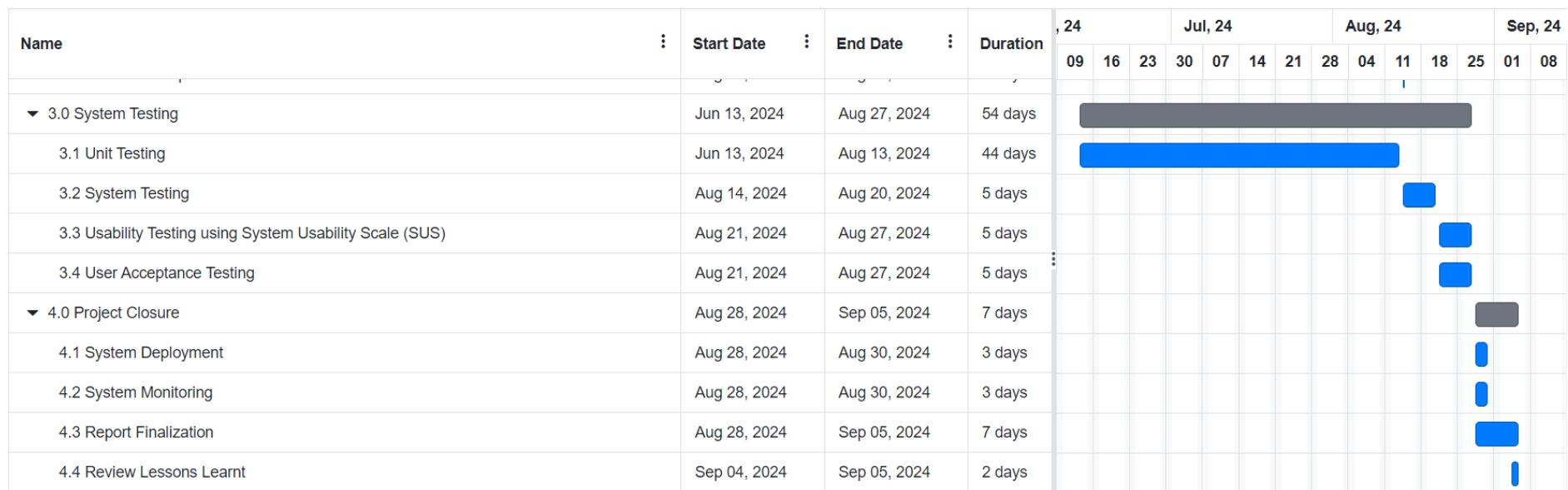


Figure 3.17: Gantt chart of this project.

3.4 Software Development Methodology

The software development methodology chosen for this project is Personal Extreme Programming (PXP). PXP methodology is made up of a combination of methodologies which are Extreme Programming (XP) and Personal Software Process (PSP). This methodology is specifically suitable to be applied by autonomous developers, who are independent software developers who do not work as part of a team.

Although PSP is also specifically designed for autonomous developers' use, deployment of PSP in a real project is very difficult. This is because it requires good knowledge regarding the process specification in order to apply correctly. Also, PSP costs much effort in preparing and maintaining the documentation as well as data (Ilieva et al., 2009). However, the workload of autonomous developers is very high and they should not spend their efforts on implementing heavy processes of the methodology.

Therefore, a lighter methodology which is the PXP is proposed so that it has an easier software development process to follow. It retains the basic principles of PSP and combines the efficient development practices from XP to ensure better performance in project planning as well as maintaining product quality. Less documentation and maintenance efforts are required in PXP compared to PSP. In this section, each phase of the PXP development methodology will be discussed.

3.4.1 Requirements

In the requirements phase, the functional and non-functional requirements are gathered and documented. In this project, the requirements are collected by referring to the existing applications with lipreading features as well as referring to the research papers that have implemented lipreading applications.

3.4.2 Planning

In the second phase of PXP, a set of tasks and subtasks are created based on the collected requirements. Schedules are assigned to these tasks and subtasks in which the total duration of the subtasks is made up of the duration of the parent tasks. Planning is also conducted to decide the development tools, integrated development environment (IDE), programming languages, deep learning model

for lipreading purposes, and others that are used in developing the web application for the lipreading project.

3.4.3 Iteration Initialization

The iteration of the project begins at this phase. In every iteration, the major tasks to be focused on will be selected. The selected tasks will be concentrated throughout the iteration. In this project, there are three iterations, which include the applying of the deep learning model, the development of the web application, and the integration of the deep learning model into the web application. Each iteration takes around two to three weeks to complete.

3.4.4 Design

The iteration begins with the design phase. In this project, the design phase for the first iteration is to select the lipreading model that is suitable to be applied to this project's web application, which is LipCoordNet. In the development of web application iteration, it involves the test case design. The iteration of integration of the deep learning model into the web application involves the design of system architecture and the communication between the application and model.

3.4.5 Implementation

The next phase after design is the implementation phase in which actual development of code and implementation of the previous designs are conducted at this phase. For the first iteration, inferencing is performed by the LipCoordNet model to lipread and produce the output in the form of text and video. For the second iteration, the functionalities of the web application are developed. In the third iteration, the lipreading model is applied to the web application in which the lipreading process of the model can be triggered from the web application and the lipreading results can be viewed from the application. Overall, the implementation phase involves three major steps which are unit testing, code generation, and code refactoring. In order to exit the implementation phase, the system should be error-free and pass all unit tests. Any errors identified in this phase will be modified and recorded in the log files.

3.4.6 System Testing

In the system testing phase, the developed system is tested completely to ensure it meets all written project requirements. If there are any errors identified in this phase, they are recorded in the log files with the error details and the errors are fixed.

3.4.7 Retrospective

Retrospective phase indicates the end of an iteration. In this phase, the variations between estimated and actual task completion duration are identified. This is to determine the causes of the delays so that better estimation can be done in future projects. If the system has fulfilled all the necessary requirements without open defects, the project development will be marked as an end with the product release. Otherwise, a new iteration will begin by moving to the Iteration Initialization phase again.

3.5 Technologies and Development Tools

3.5.1 Visual Studio Code

The code editor chosen to be used in developing the code in this project is Visual Studio Code. It supports application development in various languages such as HTML, CSS, JavaScript, Python, and so on. It is used to develop the ReactJS application code, Flask application code, and run the development server in this project.

3.5.2 Python

Python is used as the programming language of the deep learning lip reading model in this project. It offers a large ecosystem of libraries suitable for data processing and matrix math such as NumPy, TensorFlow, Keras, PyTorch, and so on, making it suitable for implementing deep learning algorithms such as the lipreading model in this project. It is also used as the language of the Flask application in this project to serve as the backend server.

3.5.3 ReactJS framework

ReactJS is chosen as the frontend framework for developing the user interface of the web application and handling the user interaction. The reason for

choosing this JavaScript framework is because it has a strong community and ecosystem that provides rich resources. It also supports the combination with other libraries to build the complete application. Examples of some libraries used in the React web application in this project includes React Router for navigation and Font Awesome for icons. The component-based architecture of React also promotes code reuse, ensuring a faster development process.

3.5.4 Flask framework

Flask framework is chosen as the backend server of the web application to handle the requests from frontend and handle the server-side logic. It manages the communication between frontend and the lipreading model by receiving HTTP request and return with JSON response. It is also used in this application to configure database and create database table. Flask is chosen as the backend framework of this project due to its lightweight and minimalist structure. Also, it is developed in Python, therefore it is easily integrated with the lipreading model code coded in Python as well.

3.5.5 SQLite

SQLite is chosen as the database used in managing the data in this application. It is selected to be used in this project due to its serverless and lightweight nature. To visualize the stored data via a graphical user interface, the DB Browser for SQLite is used.

3.6 Summary

To sum up, this chapter discussed the workflow of the LipCoordNet model starting from the data collection step until the performance evaluation step. Next, the project plan and the planned schedule are also included in this chapter by using the Work Breakdown Structure (WBS) and Gantt Chart to have an overview of the development phases, helping to track the project's progress. After that, it discussed the methodology used in developing the web application of this project, which is Personal Extreme Programming (PXP). Lastly, the development tools and technologies used in the development of this project are also included in this chapter as they are the necessary resources required to support the development of the application.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter defines the requirement specifications including the functional and non-functional requirements. After collecting and recording the requirements into the requirement specifications, a use case diagram is used to illustrate the system specification and the details are described in the use case descriptions. Also, an initial prototype containing the lipreading model output and a simple web application user interface design are included to show the feasibility of lipreading.

4.2 Functional Requirements

The functional requirements of the system are gathered based on the existing similar lipreading applications and according to the research papers that have prototyped or implemented the lipreading web or mobile application. The functional requirements of the lipreading web application are listed in Table 4.1 below:

Table 4.1: Functional requirements of the lipreading web application.

Module	ID	Functional Requirements
Video Upload Module	FR1	The web application shall allow users to upload the pre-recorded video containing lip movements from their own devices.
	FR2	The web application shall display the video requirements on the screen.
	FR3	The web application shall allow users to manage the playback of their uploaded video using the play, pause, full screen mode, picture-in-picture mode, set volume, and set video playback speed features.
	FR4	The web application shall allow users to manage the playback of the sample video using the play, pause,

		full screen mode, picture-in-picture mode, set volume, and set video playback speed features.
Lipreading Module	FR5	The web application shall be able to lipread the video using the deep learning model applied in the system.
	FR6	The web application shall allow users to cancel the lipreading process.
Output Video Module	FR7	The web application shall display the output video on the screen.
	FR8	The web application shall allow users to manage the playback of the output video using the play, pause, full screen mode, picture-in-picture mode, set volume, and set video playback speed features.
Transcription Module	FR9	The web application shall display the predicted transcriptions on the screen.
	FR10	The web application shall allow users to edit the predicted transcriptions.
	FR11	The web application shall allow users to download the predicted transcriptions to their own devices.
Preview Sample Module	FR12	The web application shall display the sample output of lipreading on the screen.
FAQ Module	FR13	The web application shall display the frequently asked questions on the screen.
Contact Module	FR14	The web application shall allow users to submit their name, email, and suggestions to be stored in the database.

4.3 Non-Functional Requirements

Non-functional requirements define the attributes that determine the quality of the system apart from its specific behaviours. It is usually used to determine the system's capabilities and constraints(AltexSoft, n.d.). The non-functional requirements of the lipreading web application include:

Table 4.2: Non-functional requirements of the lipreading web application.

ID	Non-functional Requirements	Category of NFR
NFR1	The web application should have less than 0.5 of WER and CER in its lipreading prediction of the video.	Accuracy
NFR2	The web application should have a simple and intuitive interface with clear buttons, icons, and simple instructions.	Usability
NFR3	The web application should take less than 3 seconds for the time taken to upload and download videos.	Performance
NFR4	The web application should be compatible with the web browsers of Google Chrome, Microsoft Edge, and Mozilla Firefox.	Compatibility

4.4 Use Case Diagram

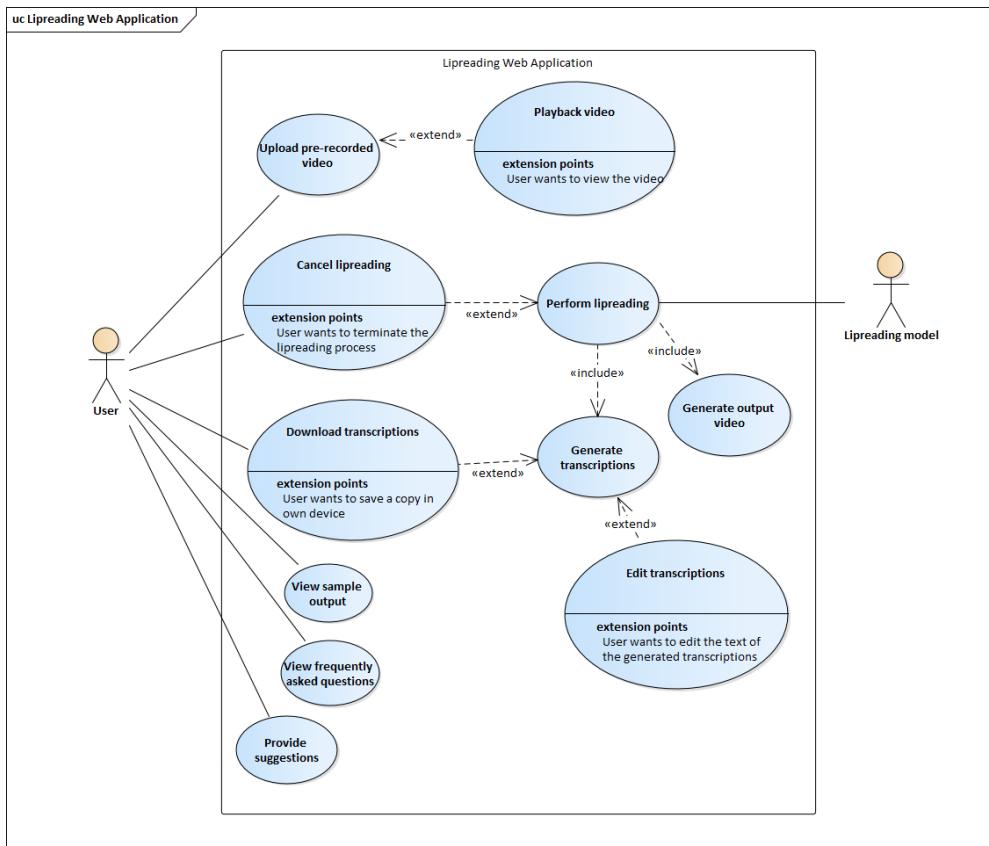


Figure 4.1: Use case diagram of the lipreading web application.

4.5 Use Case Description

Table 4.3: Upload Pre-recorded Video Use Case.

Use Case Name: Upload pre-recorded video		ID: UC01	Importance Level: High		
Primary Actor: User		Use Case Type: Detail, Essential			
Stakeholders and Interests:					
User - wants to lipread the video to understand the spoken words in the video					
Brief Description: User selects the pre-recorded video from its own device to be uploaded to the system.					
Trigger:					
User selects video and presses the upload button					
Relationships:					
Association : User Include : N/A Extend : Playback video Generalization : N/A					
Normal Flow of Events:					
1. The user prepares his/her own video containing the lip movement that he/she wants to lipread. 2. The user stores the pre-recorded video in his/her own device. 3. The user presses the upload button to select the pre-recorded video that he/she wants to upload to the system for lipreading. If the upload failed, the system will display error message to the screen, and user can choose to upload another video. 4. The system takes several seconds to upload the video from the user's device to the web application.					

<p>5. The system displays the successfully uploaded video to the screen.</p> <p>6. If the user changes his/her mind and wants to upload another video, the flow of events will be executed again starting from main flow 3.</p>
<p>Sub-flows:</p> <p>N/A</p>
<p>Alternate/Exceptional Flows:</p> <p>E-1:</p> <p>1. The user can playback the successfully uploaded video if he/she wants to view the video via the application.</p>

Table 4.4: Perform Lipreading Use Case.

Use Case Name: Perform lipreading	ID: UC02	Importance Level: High		
Primary Actor: Lipreading model	Use Case Type: Detail, Essential			
<p>Stakeholders and Interests:</p> <p>Lipreading model – executes the algorithm to perform lipreading of video and generate transcriptions and output video to the user</p> <p>User – wants to view, edit or download the generated transcriptions; wants to cancel the lipreading process</p>				
<p>Brief Description: This use case describes about the process of lipreading of video and the generation of output transcriptions and video.</p>				
<p>Trigger: User presses the button that inform the system to start to lipread the uploaded video.</p>				

<p>Relationships:</p> <table> <tbody> <tr> <td>Association</td><td>: Lipreading model</td></tr> <tr> <td>Include</td><td>: Generate transcriptions, Generate output video</td></tr> <tr> <td>Extend</td><td>: Cancel lipreading, Edit transcriptions, Download transcriptions</td></tr> <tr> <td>Generalization</td><td>: N/A</td></tr> </tbody> </table>	Association	: Lipreading model	Include	: Generate transcriptions, Generate output video	Extend	: Cancel lipreading, Edit transcriptions, Download transcriptions	Generalization	: N/A	
Association	: Lipreading model								
Include	: Generate transcriptions, Generate output video								
Extend	: Cancel lipreading, Edit transcriptions, Download transcriptions								
Generalization	: N/A								
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The user presses the button that initiate the lipreading feature of the system. 2. The lipreading model starts to execute its algorithm to lipread the video. 3. The system displays a loading overlay. 4. The lipreading model generates transcriptions. 5. The lipreading model generates output video. 6. The system hides the loading overlay. 7. The system displays the generated transcriptions on the screen. 8. The system displays the generated output video on the screen. 									
<p>Sub-flows:</p> <p>N/A</p>									
<p>Alternate/Exceptional Flows:</p> <p>E-1:</p> <ol style="list-style-type: none"> 1. If the user wants to cancel the lipreading process, the user can click on the cancel button shown at the loading overlay. 2. The system hides the loading overlay. 3. The lipreading process is terminated. 4. The system displays a message on screen indicating that the lipreading process is terminated. <p>E-2:</p>									

- | |
|--|
| <ol style="list-style-type: none"> 1. If the user wants to edit the generated transcriptions, the user can hover the mouse to the transcription location. 2. The mouse cursor turns into text cursor. 3. The user selects the text that he/she want to edit and type the correct texts. |
|--|

E-3:

- | |
|--|
| <ol style="list-style-type: none"> 1. If the user wants to have a copy of the generated transcriptions in their own device, the user can presses the download button to download the transcriptions. 2. The user selects the location where the transcriptions will be saved at. 3. The system saves the transcriptions into the user's device. |
|--|

Table 4.5: View Sample Output Use Case.

Use Case Name: View Sample Output		ID: UC03	Importance Level: High		
Primary Actor: User		Use Case Type: Detail, Essential			
Stakeholders and Interests:					
User – wants to view the sample output of the lipreading before trying out the lipreading feature in the web application					
Brief Description: This use case describes about the process of user viewing the sample output of the lipreading.					
Trigger: User navigates to the preview sample output page.					
Relationships:					
Association : User Include : N/A Extend : N/A Generalization : N/A					

Normal Flow of Events:
1. The user presses the preview sample output button from the home page or from the navigation bar.
2. The system displays the sample output of the lipreading which include some images, description texts, and a video.
3. The user scrolls at the webpage to view the sample output.
Sub-flows:
N/A
Alternate/Exceptional Flows:
N/A

Table 4.6: View Frequently Asked Questions Use Case.

Use Case Name: View Frequently Asked Questions	ID: UC04	Importance Level: High		
Primary Actor: User	Use Case Type: Detail, Essential			
Stakeholders and Interests: User – has enquiries about the web application and wants to view the frequently asked questions to look for answers				
Brief Description: This use case describes about the process of user viewing the frequently asked questions and their respective answers.				
Trigger: User navigates to the frequently asked question page.				

Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization : N/A
Normal Flow of Events:
<ol style="list-style-type: none"> 1. The user presses the FAQ button from the navigation bar. 2. The system displays a list of frequently asked questions. 3. The user scrolls through the webpage to look for the questions that they wonder. 4. The user clicks on the questions that they want to see their answers. 5. The system displays the answer for the question that the user selects.
Sub-flows:
N/A
Alternate/Exceptional Flows:
N/A

Table 4.7: Provide Suggestions Use Case.

Use Case Name: Provide Suggestions	ID: UC05	Importance Level: High		
Primary Actor: User	Use Case Type: Detail, Essential			
Stakeholders and Interests:				
User – wants to provide suggestions or recommendations to improve the web application				
Brief Description: This use case describes about the process of user sends the suggestion message to the system.				

Trigger: User navigates to the contact us page.												
<p>Relationships:</p> <table> <tr> <td>Association</td> <td>:</td> <td>User</td> </tr> <tr> <td>Include</td> <td>:</td> <td>N/A</td> </tr> <tr> <td>Extend</td> <td>:</td> <td>N/A</td> </tr> <tr> <td>Generalization</td> <td>:</td> <td>N/A</td> </tr> </table>	Association	:	User	Include	:	N/A	Extend	:	N/A	Generalization	:	N/A
Association	:	User										
Include	:	N/A										
Extend	:	N/A										
Generalization	:	N/A										
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The user presses the contact us button from the navigation bar or home page. 2. The user enters his or her name. 3. The user enters his or her email. 4. The user enters the suggestions. 5. The user clicks on the submit button. If the input of the name, email, or suggestions is left empty, the system displays error message. If the input of the email is in invalid format, the system displays error message. 6. The system displays success message. 												
<p>Sub-flows:</p> <p>N/A</p>												
<p>Alternate/Exceptional Flows:</p> <p>N/A</p>												

4.6 Initial Prototype

4.6.1 Lipreading Model Sample Output

This initial prototype of the lipreading model is developed to prove the feasibility of the lipreading feature that will be implemented in the web application. This initial prototype makes use of the pre-trained LipCoordNet

model. The model has been trained using the EGCLLC dataset, an enhanced version of the GRID corpus by adding the lip landmark coordinates. Figures 4.2, 4.3, and 4.4 show the sample input video and output result of the lipreading prototype.

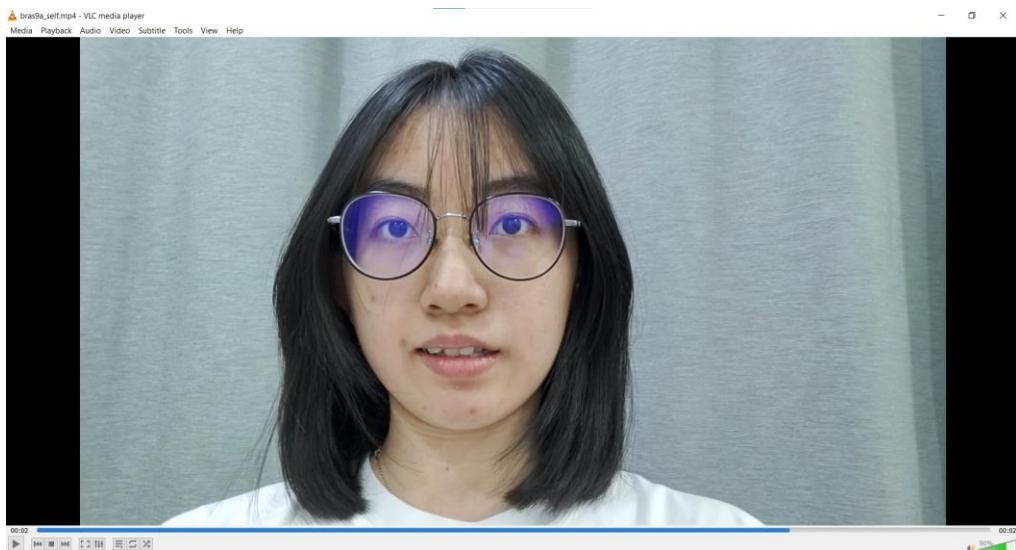


Figure 4.2: Sample input video to test the model. The spoken words are ‘BIN RED AT S NINE AGAIN’.

```
D:\LipCoordNet>python inference.py --input_video lip_images/bras9a_self.mp4  
PLACE RED AT S SIX AGAIN
```

Figure 4.3: Predicted result of the video is ‘PLACE RED AT S SIX AGAIN’.

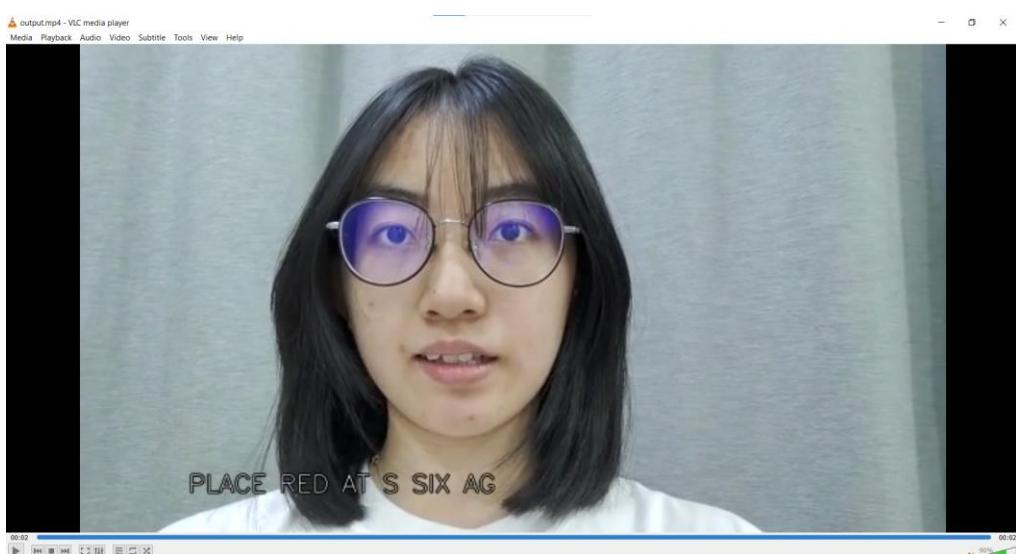


Figure 4.4: Sample output video with the predicted sentence overlayed in the video.

4.6.2 Web Application Low Fidelity Interface Design

A sample of the low-fidelity interface design of the web application is shown in Figures 4.5, 4.6, and 4.7 below. Figure 4.5 shows the home page in which users can select to upload pre-recorded videos from their own devices. After successfully uploading, the video will be displayed on the screen, as shown in Figure 4.6, allowing users to playback and view the uploaded video. When the user presses the ‘Start to Lipread’ button, it will take some time to load and navigate to the lipread page in Figure 4.7. On this page, users can playback the video containing the transcriptions overlayed in the video. Users can also edit transcriptions, download transcriptions into their own devices, or choose to upload another video for lipreading. If the user chooses to upload another video, they will be redirected back to page 2 (Upload Page) as shown in Figure 4.6.

1) Home Page

Web App Name

Web App Name

Some Simple Descriptions

Choose a video file for lipreading:

File Path

Upload file button

after successful upload will display file path

Figure 4.5: Interface design of the home page of the web application.

2. Upload Page

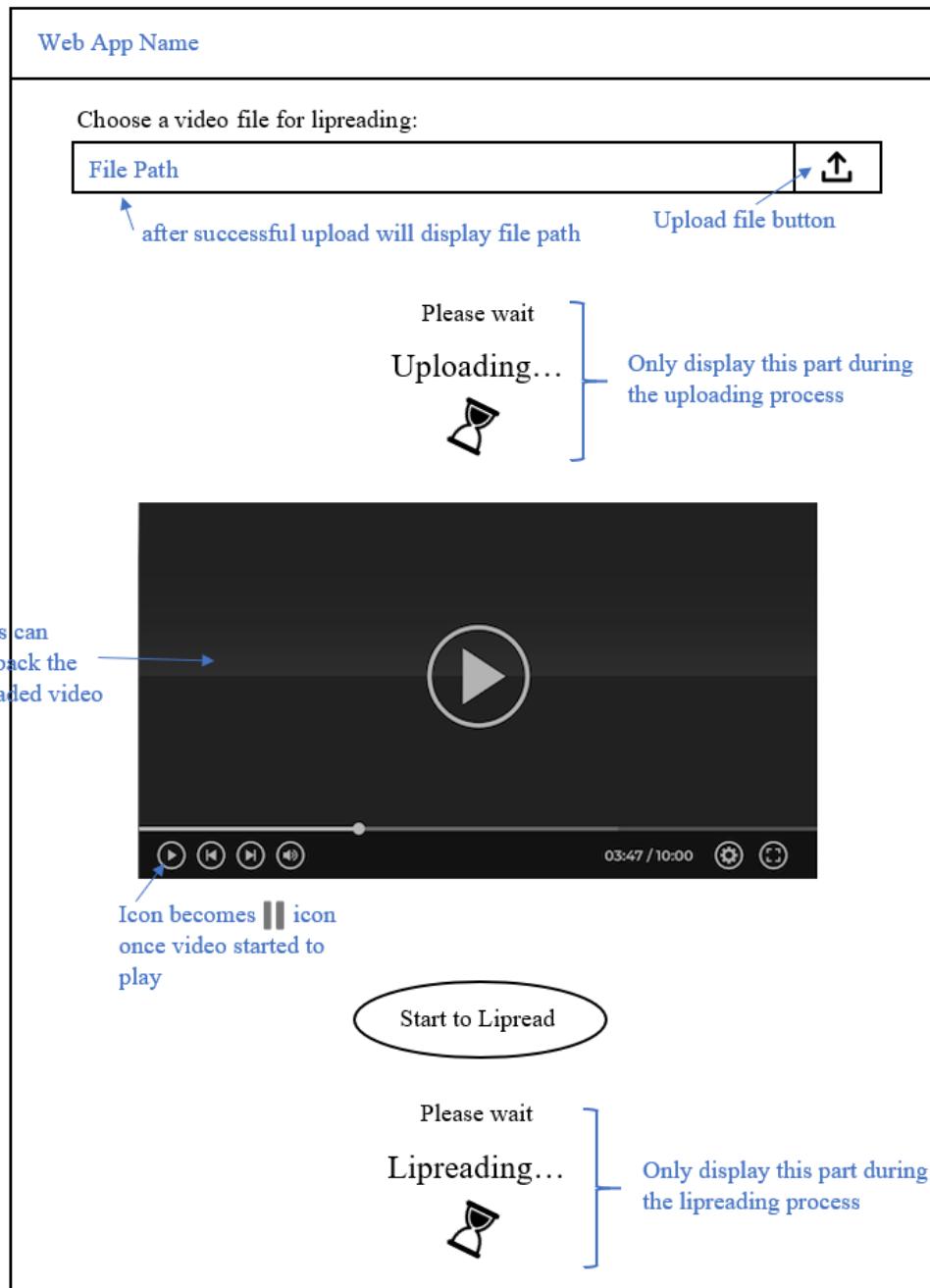


Figure 4.6: Interface design of the upload page of the web application.

3. Lipread Page



Figure 4.7: Interface design of the lipread page of the web application.

4.7 Summary

To sum up, this chapter has included the requirements needed for the lipreading web application by writing the functional requirements and non-functional requirements. Based on the requirements gathered, use case diagram is modelled with the details written in the use case descriptions. After that, the initial prototypes are included in this chapter to prove the feasibility of lipreading in this project.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter discusses about the overview of the system architecture design and database design of the project. The system architecture design includes the ReactJS framework and the Flask framework. The ReactJS framework is the framework used in handling the rendering of the user interface as well as user interactions. The Flask framework is the backend server used to process the requests from frontend and handle the server-side logic. The backend server interacts with the deep learning model and the database used in the project. Additionally, a data dictionary is included to present the database design of the project.

5.2 System Architecture Design

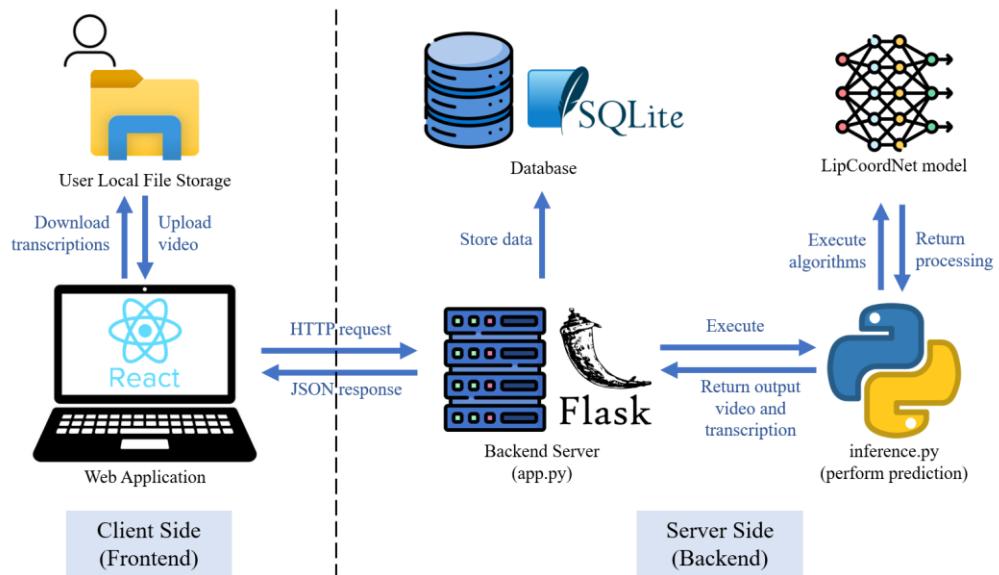


Figure 5.1: The overview of the system architecture design.

Figure 5.1 shows the system architecture design implemented in current project. It consists of a web application developed using ReactJS framework. The development server of React handles the rendering of the user interface in the

user's browser, including the Home Page, Upload Page, and Lipread Page. This frontend development server allows user to upload video from their local file storage to the web application.

Upon the user decided to initiate the lipreading, the HTTP POST request will be sent to the backend server. In this project, the backend server code is developed using Flask, a Python web framework, which serves to build the Rest APIs. The code is written in the app.py file. When the app.py file receives POST request, it will execute the inference.py to perform lipreading prediction. The inference.py will execute the algorithms of the LipCoordNet model and return the output video and transcriptions back to the backend server.

After that, the backend server returns JSON response back to the web application and the response is handled by the React frontend. The output video and transcription can then be viewed on the web application.

Also, the user can send HTTP POST request to the backend server for storing data into the database. The database used in this project is SQLite, which is a C-language library that implements the SQL database engine.

5.2.1 ReactJS architecture

ReactJS is a user interface library that supports the development of the user interface in the web application. The concepts applied by React include the concept of the React elements, JSX, and the React component.

React elements are considered as the smallest building blocks in developing the React application. The element can be created using the React API, 'React.createElement', or by using JSX. The created element can then be rendered to the Browser DOM (tutorialspoint, n.d.).

JSX, with the full name of JavaScript XML, supports the developers to code HTML syntax in React. It is used to design the user interface and can be used to create user interface if it is compiled into the React elements. By using JSX, it makes the development of React applications to be more easier because the developers do not need to use any createElement() and appendChild() methods (W3Schools, n.d.).

React component is the main building block in the development of the application. It is made up of the React elements and JSX. It is either a JavaScript

function or a JavaScript class. Its main responsible is to render the user interface of the web application and perform updates when there is change to the internal state. It also manages and handles the events related to the user interface.

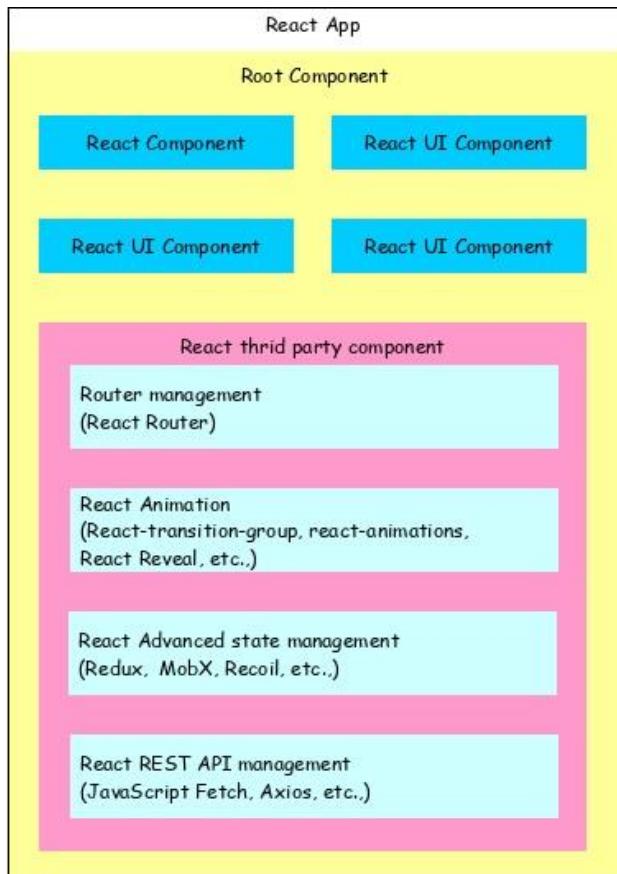


Figure 5.2: The high level architecture of React application.

Source: tutorialspoint (n.d.)

Based on Figure 5.2, the React application is composed by components, each may be nested with the other components, but begins with the root component. The root component is built using different components, with most of them to be the UI component, as well as some third party components. Some examples of the UI components include the forms, buttons, and text inputs.

The third party components have their own specific usage. For instance, this project make use of the ‘useNavigate’ hook which belongs to the React Router routing library under the Router management component to manage the navigation to different web pages. Besides that, this project also uses axios under the React REST API management component to communicate with the

backend. It enables the HTTP requests to be made to the API and processes the data return from the API.

5.2.2 Flask architecture

Flask is a Python microframework which supports the development of web applications quickly and simply. It is based on the Web Server Gateway Interface (WSGI) framework that supports the web servers to send requests to the web applications (Deery, 2023).

In this project, Flask acts as the backend framework to handle the server-side logic, which is coded in the app.py file. It manages the communication between the frontend and the lipreading model. The frontend of the application interacts with the backend via the HTTP GET and POST requests. For instance, when the user made a HTTP POST request to initiate the lipreading process, Flask will process the request and trigger the inference process to begin.

Apart from that, it also interacts with the database via the SQLAlchemy extension. It supports the configuration of the database, the definition and the creation of database table. It also processes the request to add data to the database table upon receiving HTTP POST request.

5.3 Database design

In this project, SQLite is used to organizes the data of the system. It is a serverless and lightweight SQL database engine which is famous for its simplicity (Priy, 2024). The database used in this system is named as ‘lipreadDB’. In this section, a data dictionary is presented to illustrate the database design of the system.

5.3.1 Data dictionary

Data dictionary is used to capture the information related to the data elements in a database. These information provides better understanding to the reader about the meaning, purpose, and usage of the data elements.

Table 5.1: Data dictionary for contact collection.

Attribute	Data type	PK / FK	Description	Nullable	Example values
id	integer	PK	unique identifier of the contact	no	1
name	string	FK	name of the user who sent the message in current contact	no	adam
email	string	FK	email of the user who sent the message in current contact	no	adam@gmail.com
message	string	FK	the message sent in current contact	no	lipreading takes long time

5.4 Conclusion

Overall, this chapter outlines and describes the system architecture and database design of the system in this project. The frontend framework used is ReactJS, the backend framework used is Flask, whereas the database used is SQLite. Some examples of these frameworks components or extensions are described to explain their usage in this project. These designs lays the foundation for the system implementation in next chapter.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Introduction

This chapter describes the implementation of the system including the project setup and the development of the complete web application. In the project setup section, it covers the setup of the ReactJS application, database, and lipreading model. For the web application implementation section, it includes the user interface screenshots, code snippet screenshots, and some feature explanations for the 7 modules in the web application.

6.2 Project Setup

The project setup begins with setting up a React environment for the development of the web application. Since React requires Node.js and npm (Node Package Manager) to run, it can be downloaded from Node.js official website: <https://nodejs.org/en/download/package-manager>. This installation will include the npm. After completed the installation, command prompt is opened at the directory where the project would like to be created. At the command prompt, the command “npx create-react-app my-react-app” is entered by replacing the ‘my-react-app’ with the desired project name which is ‘fyp-web-app-master’. While the command is running, the ‘create-react-app’ is installed and the project is set up in the folder ‘fyp-web-app-master’.

Next, at the command prompt, the command “cd fyp-web-app-mater” is typed to navigate to the project directory. After that, by running the command “npm start” at the command prompt, the React application will be started and opened at the default web browser which is <http://localhost:3000/>.

6.2.1 SQLite Database Setup

Since this project is using SQLite database, it is setup in a Flask application. First of all, the modules ‘Flask’ and ‘SQLAlchemy’ are imported using the code below:

```
from flask import Flask
```

```
from flask_sqlalchemy import SQLAlchemy
```

Next, the Flask application is configured to use the SQLite database with the name of lipreadDB. The SQLAlchemy instance is then created and initialized with the Flask app, as shown as the code below:

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///lipreadDB.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

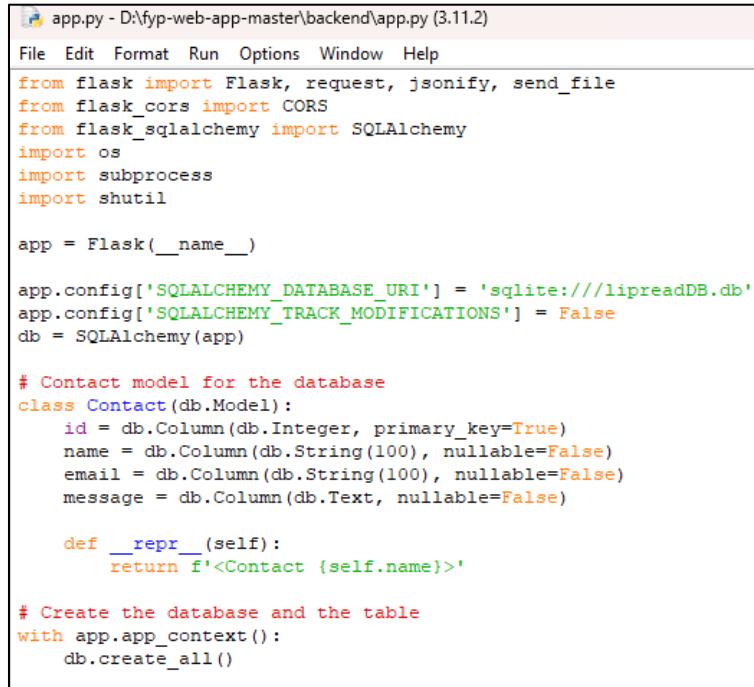
After that, the database model named as ‘Contact’ is defined via the class Contact, with the attributes of ‘id’, ‘name’, ‘email’, and ‘message’.

```
class Contact(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), nullable=False)
    message = db.Column(db.Text, nullable=False)
```

Finally, the database and the ‘Contact’ table are created using the code below:

```
with app.app_context():
    db.create_all()
```

The complete code for setting up the database and ‘Contact’ table can be found in Figure 6.1 below.



```

app.py - D:\fyp-web-app-master\backend\app.py (3.11.2)
File Edit Format Run Options Window Help
from flask import Flask, request, jsonify, send_file
from flask_cors import CORS
from flask_sqlalchemy import SQLAlchemy
import os
import subprocess
import shutil

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///lipreadDB.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# Contact model for the database
class Contact(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), nullable=False)
    message = db.Column(db.Text, nullable=False)

    def __repr__(self):
        return f'Contact {self.name}'

# Create the database and the table
with app.app_context():
    db.create_all()

```

Figure 6.1: The code snippet for defining and creating database and ‘Contact’ table.

As the module runs, it generates a database file (lipreadDB.db), which can be found in the directory ‘backend/instance’ as shown as Figure 6.2 below.

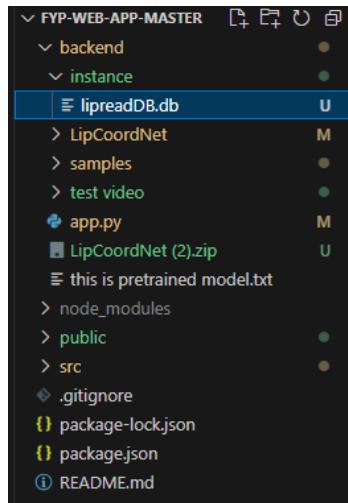


Figure 6.2: Location of the lipreadDB.db file.

By using DB Browser for SQLite, the information related to the ‘Contact’ table can be found, as shown in Figure 6.3 below. With the help of

DB Browser for SQLite, it is easier to visualize the data stored in the database table.

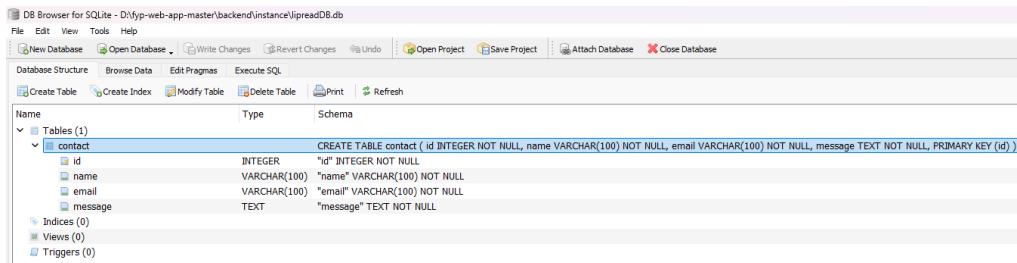


Figure 6.3: The ‘Contact’ table displayed in DB Browser for SQLite.

6.2.2 Lipreading Model Setup

The lipreading model selected to be used in this project is the LipCoordNet model. To apply this model in the project, the first step is to clone the repository to local device. The command used to clone the repository to the desired location of the device is as shown:

```
git clone https://huggingface.co/SilentSpeak/LipCoordNet
```

Since the lipreading model would be applied in the web application, the repository is decided to be cloned in the ‘fyp-web-app-master/backend’ directory.

After cloning the repository, it is navigated to the project directory by using the command ‘cd LipCoordNet’. Within the project directory, all the required dependencies are installed using the command ‘pip install -r requirements.txt’. Figure below shows the required dependencies and their respective versions.

```

certifi==2022.12.7
charset-normalizer==2.1.1
colorama==0.4.6
dlib==19.24.2
editdistance==0.6.2
face-alignment==1.1.1
filelock==3.9.0
fsspec==2023.4.0
idna==3.4
imageio==2.33.1
Jinja2==3.1.2
lazy_loader==0.3
MarkupSafe==2.1.3
mpmath==1.3.0
networkx==3.0
numpy==1.24.1
opencv-python==4.8.1.78
packaging==23.2
Pillow==9.3.0
protobuf==4.25.1
requests==2.28.1
scikit-image==0.22.0
scipy==1.11.4
sympy==1.12
tensorboardX==2.6.2.2
tifffile==2023.12.9
torch==2.1.1
torchaudio==2.1.1
torchvision==0.16.1
tqdm==4.66.1
typing_extensions==4.4.0
urllib3==1.26.13

```

Figure 6.4: The required dependencies and their respective versions to be installed.

6.3 Web Application Implementation

In this section, the implementation of the web application will be described by providing the user interface, the code implementation, and the feature explanation. The implementation will be described based on the modules listed in the Table 6.1 below.

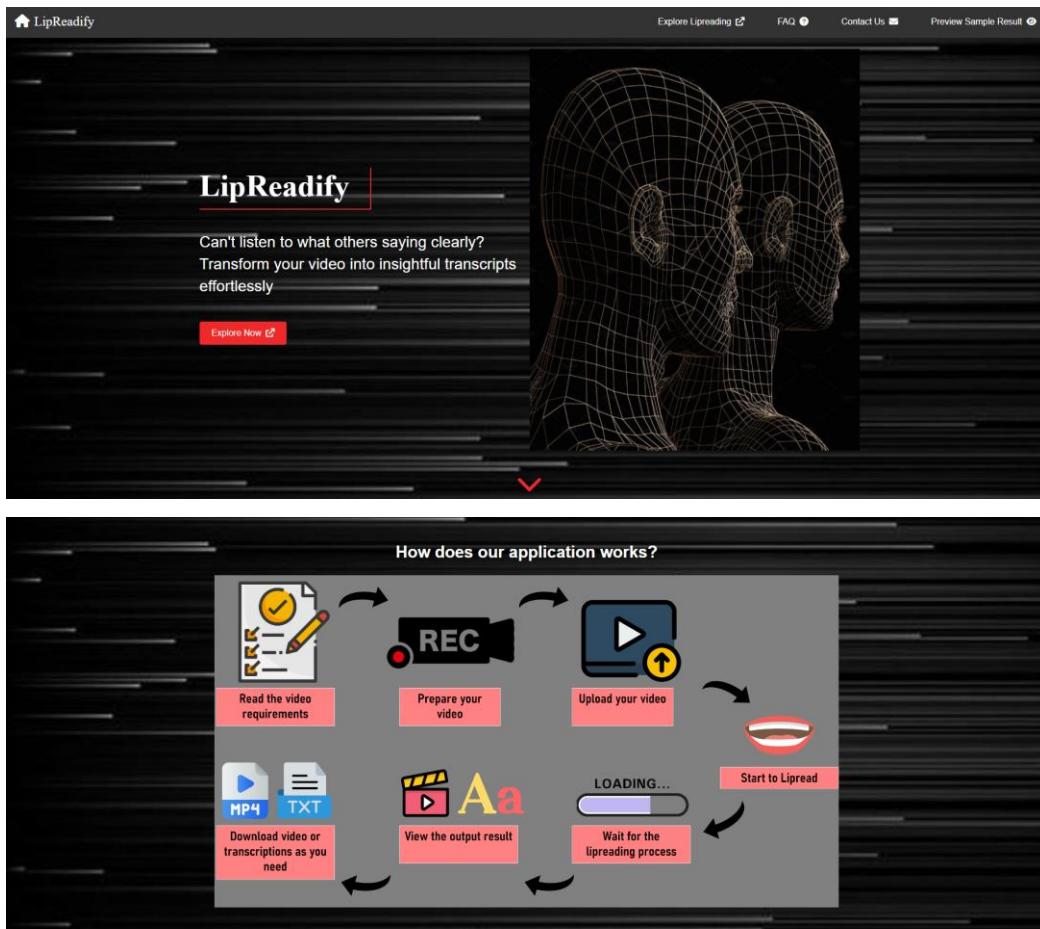
Table 6.1: Modules of the Web Application for Lipreading.

Modules
6.3.1 Video Upload Module
6.3.2 Lipreading Module
6.3.3 Output Video Module
6.3.4 Transcription Module
6.3.5 Preview Sample Module

6.3.6 FAQ Module

6.3.7 Contact Module

At the home page of the web application, users are provided with some descriptions to introduce the usage of the web application as well as some navigation buttons which can navigate to the different modules of the web application.



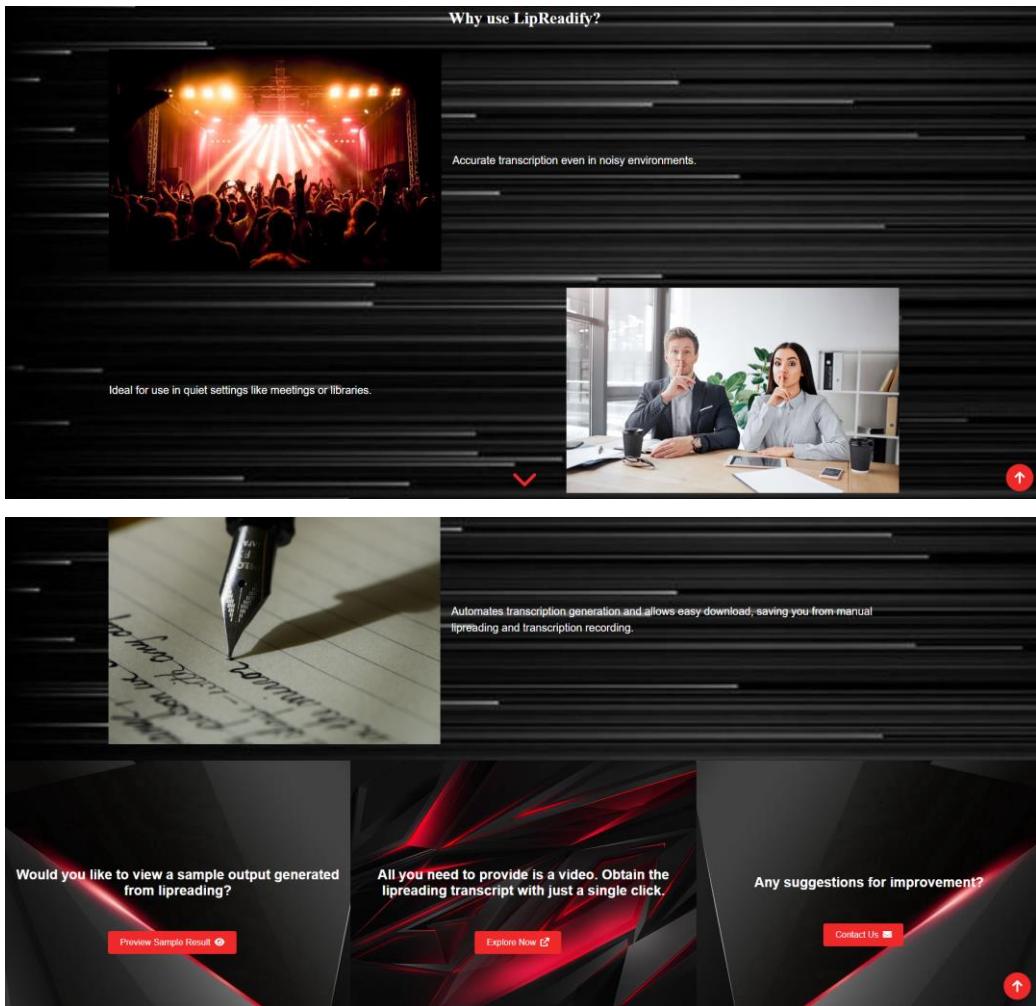


Figure 6.5: Home page of the web application which include some descriptions of the application and some navigation buttons.

6.3.1 Video Upload Module

When the users press the 'Explore Now' button at the home page or the 'Explore Lipreading' button at the navigation bar, the users are navigated to the webpage which is in charge of the video upload module. In this module, it allows the users to choose a video file in '.mp4' format to be uploaded for lipreading. It also includes the video requirements that needs to be followed by the users in choosing a suitable video file for lipreading. The requirements should be adhered by the users so that the video upload process and the lipreading process can proceed smoothly, and maximize the accuracy of the lipreading result.

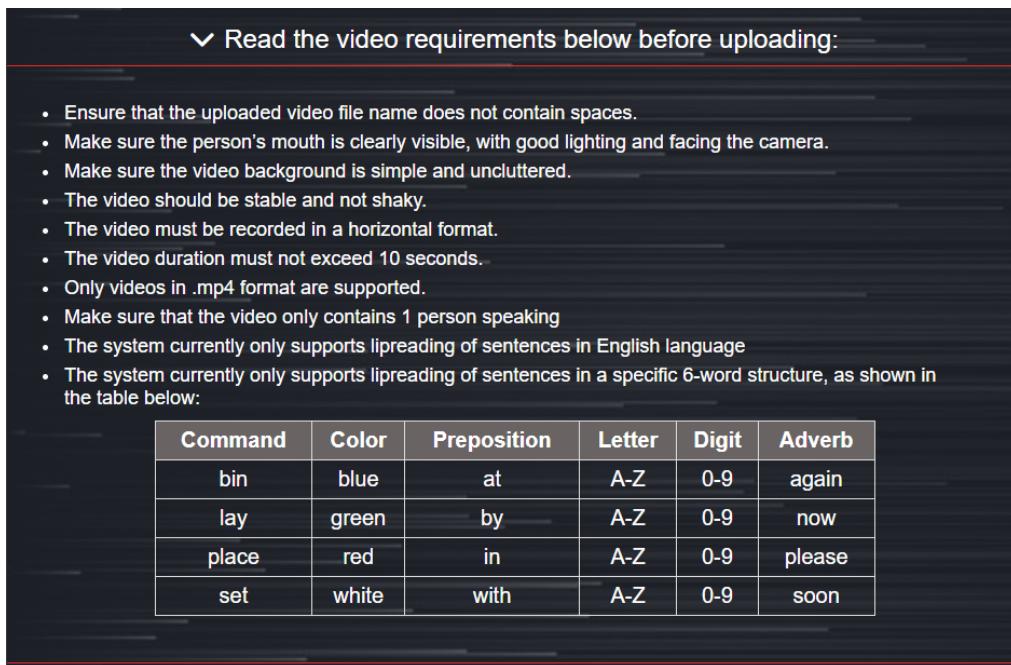


Figure 6.6: The video requirements are listed on the webpage for users to refer before they upload their video to the system.

There is also a sample video that is suitable for lipreading being displayed on the screen for users to take it as a reference.

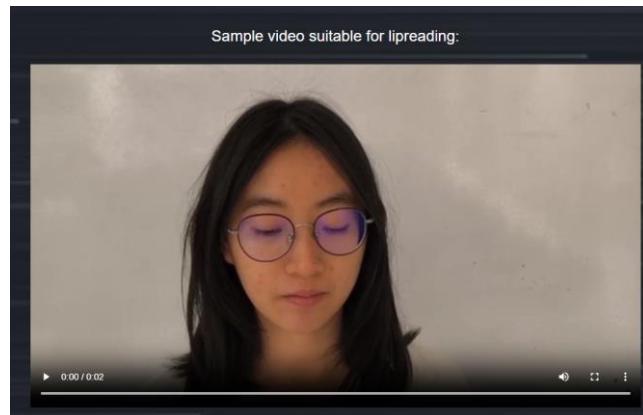


Figure 6.7: The sample video for users to refer before they upload their video to the system.

Whenever the users decided to upload their own video, they can press 'Choose File' button to choose a video file from their own device. The system will display the video file name and the video to the screen. So, the users can preview the uploaded video and make use of the video features including play

video, pause video, full screen, picture in picture, setting video playback speed, controlling video loudness, and download the video.

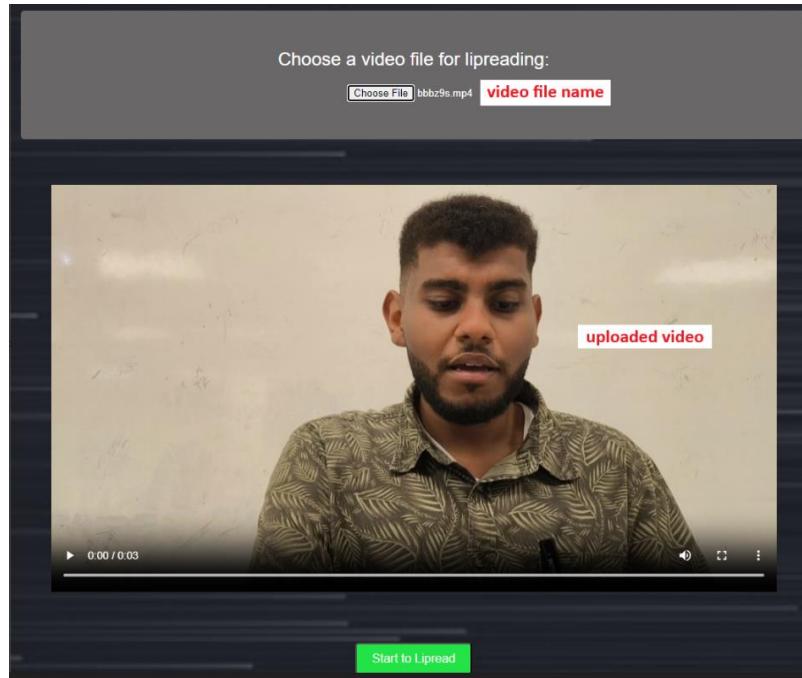


Figure 6.8: The video upload module contains the features for users to choose a video file for lipreading, preview the uploaded video, and start the lipreading process.

```
const handleFileChange = (event) => {
  const file = event.target.files[0];

  if (file) {
    const videoElement = document.createElement('video');
    const url = URL.createObjectURL(file);

    setVideoUrl('');
    setVideoFile(null);

    videoElement.src = url;
    videoElement.onloadedmetadata = () => {
      if (videoElement.duration > 10) {
        setErrorMessage('The video duration exceeds 10 seconds. Please upload a shorter video.');
        setVideoFile(null);
        setVideoUrl('');
      } else {
        setErrorMessage('');
        setVideoUrl(url);
        setVideoFile(file);
      }
    };
  } else {
    setErrorMessage('File selection was cancelled.');
    setVideoFile(null);
    setVideoUrl('');
  }
};
```

```

return (
  <div className="App">
    <header className="Header">
      <Navbar />
    </header>
    <div className="File-selection-container">
      <label className="Description-text">Choose a video file for lipreading:</label>
      <input className="Video-input-icon" type="file" accept="video/*" onChange={handleFileChange} />
    </div>
    {errorMessage && (
      <div className="Error-message">
        <i className="fa fa-exclamation-circle" aria-hidden="true"></i>
        <p>{errorMessage}</p>
      </div>
    )}
    {videoUrl && (
      <div className="Video-container">
        <video controls className="Video-input">
          <source src={videoUrl} type="video/mp4" />
          Your browser does not support the video tag.
        </video>
        <div className="Start-lipread-button">
          <button className="Lipread-button" onClick={handleSubmit}>Start to Lipread</button>
        </div>
      </div>
    )}
  </div>
)

```

Figure 6.9: The code snippet for the user interface shown in Figure 6.8. It includes the logic in handling the file uploaded by the users and displaying the successfully uploaded video to the screen.

6.3.2 Lipreading Module

Once the users decided to start the lipreading process, they can just press the ‘Start to Lipread’ button shown in Figure 6.8. The web application will take around 1 minute time to lipread the video and display the outputs at another web page. Figure 6.10 below shows the ‘handleSubmit’ function which is in charge of handling the frontend logic once the ‘Start to Lipread’ button is pressed.

```

const handleSubmit = async () => {
  setErrorMessage('');
  localStorage.removeItem('outputVideoUrl');
  if (!videoFile) {
    alert("Please upload a video first!");
    return;
  }

  setShowLoading(true);

  const formData = new FormData();
  formData.append('file', videoFile);

  const controller = new AbortController();
  setAbortController(controller);

  try {
    const response = await axios.post('http://localhost:5000/lipread', formData, {
      responseType: 'blob',
      signal: controller.signal,
    });
    const url = window.URL.createObjectURL(new Blob([response.data]));
    localStorage.setItem('outputVideoUrl', url);
    setShowLoading(false);
    navigate('/lipread', { state: { videoUrl } });
  } catch (error) {
    if (axios.isCancel(error)) {
      console.log('Request canceled:', error.message);
    } else {
      console.error('There was an error processing the video:', error);
      setErrorMessage('The video uploaded is not suitable for lipreading. Please upload another video.');
    }
    setShowLoading(false);
  }
};

```

Figure 6.10: The code snippets of the ‘handleSubmit’ function related to the frontend logic in processing the lipreading.

The code segments being framed in red colour shows the logic of sending the video file to the backend endpoint ‘`http://localhost:5000/lipread`’ via a POST request using Axios. If the response is successful, it will return a video file which is then being converted into a URL and passed along to another webpage for display purpose.

In the backend which is the Flask application, the POST request is handled, in which it receives the uploaded video file, runs the inference script to perform lipreading on the uploaded video, and returns the output video as well as the extracted transcription. In the inference script, it will load the LipCoordNet model, process the input video, generate lip coordinates, predict the transcription, and save the output video. The code snippet in the Figure 6.11 below shows the way the ‘`/lipread`’ POST request is handled in the Flask application.

```

from flask import Flask, request, jsonify, send_file
from flask_cors import CORS
from flask_sqlalchemy import SQLAlchemy
import os
import subprocess
import shutil

app = Flask(__name__)
CORS(app, resources={r"/": {"origins": "http://localhost:3000"}})

# Path to the LipCoordNet directory
LIPCOORDNET_DIR = os.path.join(os.path.dirname(__file__), 'LipCoordNet')

# Path to the inference.py script
inference_script = os.path.join(LIPCOORDNET_DIR, 'inference.py')

# Path to the input and output video directories
input_video_dir = os.path.join(LIPCOORDNET_DIR, 'lip_images')
output_video_dir = os.path.join(LIPCOORDNET_DIR, 'output_videos')
transcription = None

```

```

@app.route('/lipread', methods=['POST'])
def lipread():
    global transcription
    samples_folder = r'D:\fyp-web-app-master\backend\samples'
    if os.path.exists(samples_folder) and os.path.isdir(samples_folder):
        shutil.rmtree(samples_folder)

    if 'file' not in request.files:
        return jsonify({"error": "No file part in the request"}), 400

    video_file = request.files['file']
    if video_file.filename == '':
        return jsonify({"error": "No selected file"}), 400
    video_path = os.path.join(input_video_dir, video_file.filename)
    video_file.save(video_path)

    # Run the inference script
    command = f"python {inference_script} --input_video {video_path} --output_path {output_video_dir}"
    print(f"Running command: {command}")
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    print(f"stdout: {result.stdout}")
    print(f"stderr: {result.stderr}")
    if result.returncode != 0:
        return jsonify({"error": "Inference failed", "details": result.stderr}), 500
    lines = result.stdout.strip().splitlines()
    transcription = lines[-1] if lines else "Transcription not found"
    output_video_path = os.path.join(output_video_dir, 'output.mp4')

    return jsonify({
        "output_video_path": "/output_video",
        "transcription": transcription
    })
}

```

Figure 6.11: The code snippet of handling the ‘/lipread’ POST request in Flask application.

Another thing to mention is during the lipreading process, it will take around 1 minute to complete the process. Therefore, a loading overlay is displayed during the lipreading process to inform users that the lipreading process is ongoing.

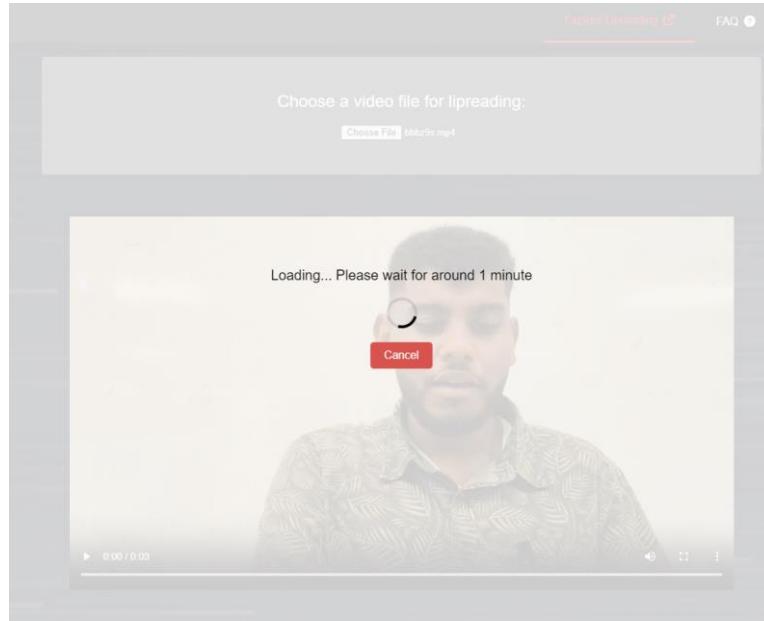


Figure 6.12: Screenshot of the loading overlay.

```
{showLoading && (
  <div className="Loading-overlay">
    <div className="Loading-message">Loading... Please wait for around 1 minute</div>
    <div className="Loading-icon"></div>
    <button className="Cancel-button" onClick={handleCancel}>Cancel</button>
  </div>
)}
```

Figure 6.13: Screenshot of the code related to the loading overlay.

During the loading process, if the users want to cancel the process, they can just click on the cancel button and the loading overlay will be hidden. The logic used to handle the cancel process is shown in the code snippet below.

```
const handleCancel = () => {
  if (abortController) {
    abortController.abort();
    setErrorMessage('Lipreading was canceled.');
    setShowLoading(false);
  }
};
```

Figure 6.14: Code snippet related to the logic of cancel loading overlay.

There are several possibilities that can affect the time taken for lipreading. One of the factors that causes the loading time to be longer is due to

the input video characteristics. The better the video setting, the more data need to be processed, causing the inference time to increase. It is noticed that if the users upload a video captured by their mobile phone directly to the web application for lipreading, it will take more than one minute inference time. However, if the captured video is being modified for some of its video setting before uploading the video for lipreading, the inference time will take less than one minute. Therefore, to improve the inference time of the system, changes have been made to the input video regarding the video size, frame per second (FPS), bitrate, and video encode. Figure below shows the details of the video setting that have been performed on the captured video before uploading to the web application.

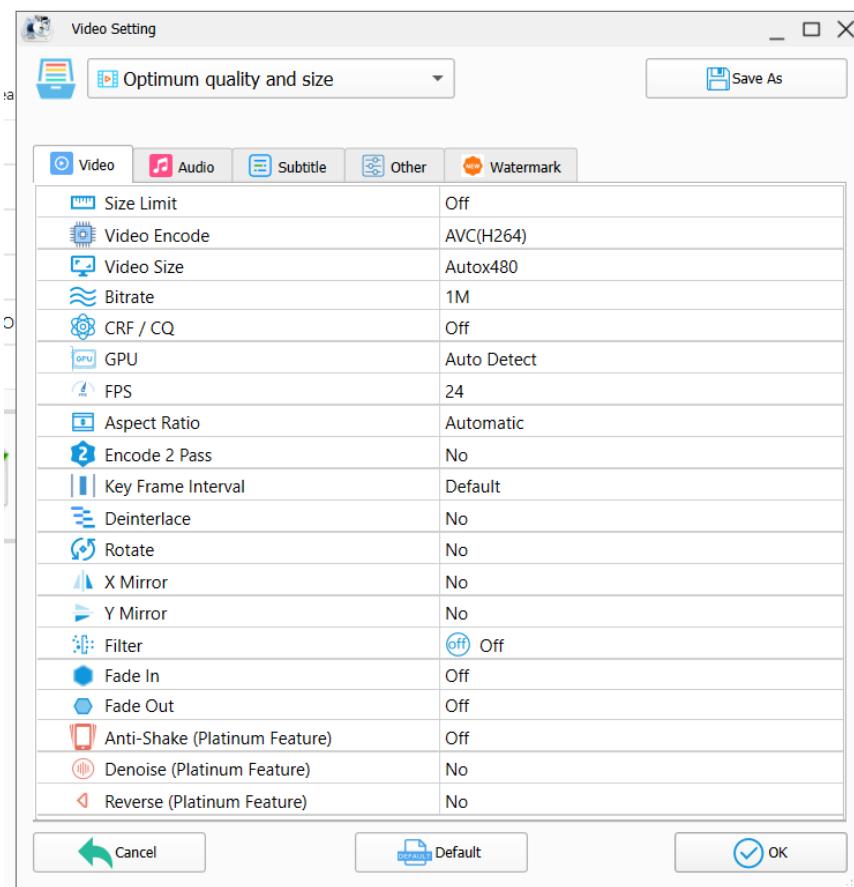


Figure 6.15: Details of the video setting that have been performed on the captured video before uploading to the web application.

Another factor that can affect the inference time is the hardware constraints. The GPU used in this project is NVIDIA GeForce GTX 1650 Ti. Although the usage of GPU has accelerated the inference speed, it is still considered a mid-range GPU. During the inference process, the GPU utilization has reached 100% even though it only utilizes 1.2 GB out of the total 4 GB memory. If there is a more powerful GPU, it can improve the utilization of the memory. Also, the GTX 1650 Ti belongs to the Turing architecture, which lacks some advanced features that can be found in higher-end GPUs like the RTX series. The example of RTX series GPU is Tensor Cores which are specifically designed to accelerate the AI workloads (Rao, 2024). Therefore, it would be better to have a more powerful GPU with better AI computational capabilities to further reduce the inference time.

6.3.3 Output Video Module

Once the loading process of the lipreading has completed, it will navigate to the lipreading output webpage. In this webpage, it displays the original video uploaded by the users, the output video of lipreading that consists of transcriptions overlaid on it, and the transcriptions displayed in textarea. It also provides features for users to download the transcriptions in text file (.txt) format and supports the users to upload another video for lipreading. The complete user interface of this webpage is shown as figure below.

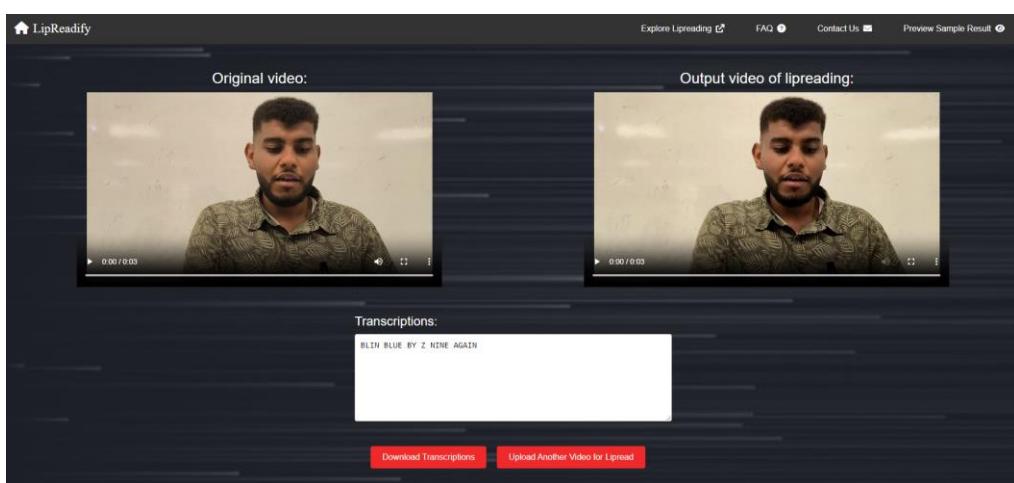


Figure 6.16: The complete user interface for the lipreading output page.

Since the main focus of the output video module is about the output video, therefore the code snippet related to the logic in handling and displaying the output video is included in the figure below. The `videoUrl` variable is used to represent the URL of the output video. It is fetched from the Flask backend at the URL ‘`http://localhost:5000/output_video`’. The browser will then load the video file from this URL.

```

src > pages > js Lipread.js > Lipread
  1 import React, { useState, useEffect } from 'react';
  2 import { useLocation, useNavigate } from 'react-router-dom';
  3 import axios from 'axios';
  4 import Navbar from '../components/Navbar';
  5
  6 function Lipread() {
  7   const location = useLocation();
  8   const navigate = useNavigate();
  9   const initialVideoUrl = location.state?.videoUrl || '';
 10   const [uploadVideoUrl, setUploadVideoUrl] = useState(initialVideoUrl);
 11   const [paragraphText, setParagraphText] = useState(
 12     ''
 13   );
 14   const videoUrl = "http://localhost:5000/output_video";
 15 }

<div className="Video-container-lipread">
  <p className="Video-label">Output video of lipreading:</p>
  <video controls className="Lipread-Video">
    <source src={videoUrl} type="video/mp4" />
    Your browser does not support the video tag.
  </video>
</div>

```

Figure 6.17: Frontend code in obtaining and displaying the output video.

In the backend route, when it receives client requests, the server will respond the client by sending the output video file (`output.mp4`) so that the video file can be played in the client’s browser.

```

@app.route('/output_video', methods=['GET'])
def get_output_video():
    output_video_path = os.path.join(output_video_dir, 'output.mp4')

    return send_file(output_video_path, mimetype='video/mp4')

```

Figure 6.18: Backend code in responding client request with the output video file (`output.mp4`).



Figure 6.19: Screenshot of the output video file rendered in the browser which contains the transcriptions being overlayed on the video.

6.3.4 Transcription Module

For the transcription module, the system will display the output transcription that is generated by the lipreading process at the textarea of the lipreading output webpage. The fetchTranscription function sends a GET request to the Flask backend which is running on ‘<http://localhost:5000/transcription>’ to fetch the transcription data. It then updates the paragraphText state with the fetched transcription text.

```
function Lipread() {
  const location = useLocation();
  const navigate = useNavigate();
  const initialVideoUrl = location.state?.videoUrl || '';
  const [uploadVideoUrl, setUploadVideoUrl] = useState(initialVideoUrl);
  const [paragraphText, setParagraphText] = useState(
    ''
  );
  const videoUrl = "http://localhost:5000/output_video";

  useEffect(() => {
    const fetchTranscription = async () => {
      try {
        const response = await axios.get('http://localhost:5000/transcription');
        const transcription = response.data.transcription;
        setParagraphText(transcription);
      } catch (error) {
        console.error('Error fetching transcription:', error);
      }
    };

    fetchTranscription();
  }, []);
}
```

Figure 6.20: Code snippet related to displaying of fetched transcription text at the textarea.

In the Flask application, when the /transcription endpoint is accessed via a GET request, it will obtain the value of the transcription global variable and return back to the frontend if the transcription is available. The value of the transcription global variable is actually assigned during the /lipread endpoint receives a POST request, which is during the time the users press the ‘Start to Lipread’ button.

```

@app.route('/lipread', methods=['POST'])
def lipread():
    global transcription
    samples_folder = r'D:\fyp-web-app-master\backend\samples'
    if os.path.exists(samples_folder) and os.path.isdir(samples_folder):
        shutil.rmtree(samples_folder)

    if 'file' not in request.files:
        return jsonify({"error": "No file part in the request"}), 400

    video_file = request.files['file']
    if video_file.filename == '':
        return jsonify({"error": "No selected file"}), 400
    video_path = os.path.join(input_video_dir, video_file.filename)
    video_file.save(video_path)

    # Run the inference script
    command = f'python {inference_script} --input_video {video_path} --output_path {output_video_dir}'
    print(f"Running command: {command}")
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    print(f"stdout: {result.stdout}")
    print(f"stderr: {result.stderr}")
    if result.returncode != 0:
        return jsonify({"error": "Inference failed", "details": result.stderr}), 500
    # Extract only the final complete line of transcription
    lines = result.stdout.splitlines()
    transcription = lines[-1] if lines else "Transcription not found"
    output_video_path = os.path.join(output_video_dir, 'output.mp4')

    return jsonify({
        "output_video_path": "/output_video",
        "transcription": transcription
    })

@app.route('/transcription', methods=['GET'])
def get_transcription():
    global transcription
    if transcription:
        return jsonify({"transcription": transcription})
    else:
        return jsonify({"error": "Transcription not found"}), 404

```

Figure 6.21: Flask application code related to the accessing of /transcription endpoint via a GET request.

Figure below shows a screenshot of the output transcription being displayed at the textarea. This is the outcome of the lipreading performed by the system. As shown in the figure, the value of the predicted transcription is ‘BLIN BLUE BY Z NINE AGAIN’.

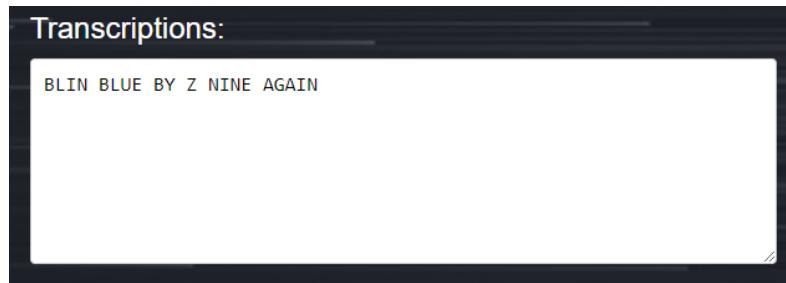


Figure 6.22: Screenshot of the transcription displayed at the textarea.

However, it is assumed that the predicted transcription is not 100% same as the actual transcription. For example, the actual transcription of this video should be ‘BIN BLUE BY Z NINE SOON’. In this case, the users can modify the transcription displayed at the textarea to their desired text. The code used to handle this logic is displayed in the figure below.

```
const handleChange = (event) => {
  setParagraphText(event.target.value);
};
```

Figure 6.23: Code snippet used to handle the logic of modifying the transcriptions at the textarea.

Upon modification completed, the users can download the modified transcription to their own device. As the users click on the ‘Download Transcriptions’ button indicated in the figure below, a transcriptions.txt text file is downloaded to their own device.

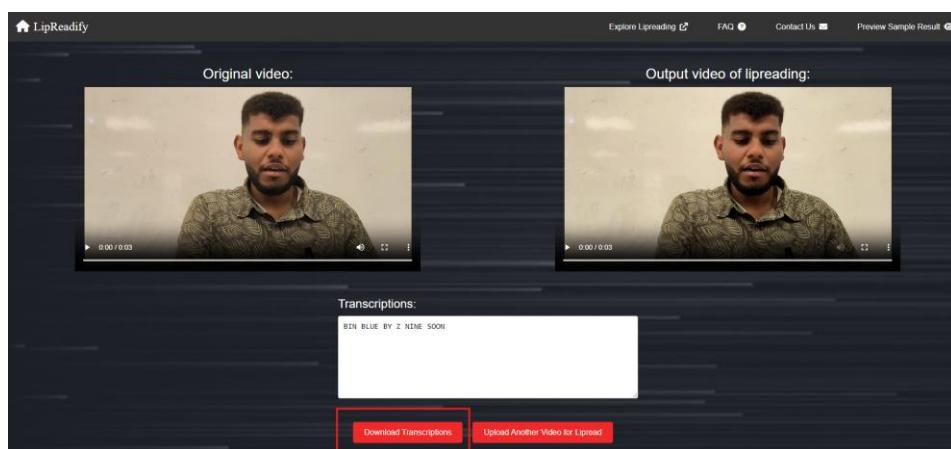


Figure 6.24: Screenshot of the lipreading output webpage to indicate the ‘Download Transcriptions’ button.

```
const handleDownloadTranscriptions = () => {
  const blob = new Blob([paragraphText], { type: 'text/plain' });
  let a = document.createElement('a');
  a.style.display = 'none';
  document.body.appendChild(a);
  let url = window.URL.createObjectURL(blob);
  a.href = url;
  a.download = 'transcriptions.txt';
  a.click();
  window.URL.revokeObjectURL(url);
  document.body.removeChild(a);
};
```

Figure 6.25: Code snippet related to the logic in handling download transcriptions.

The users can then open the downloaded text file to view the content. With this download transcription feature, the users can have a copy of the lipreading ouput on their own device, avoiding the task to manually record down the output of the lipreading.

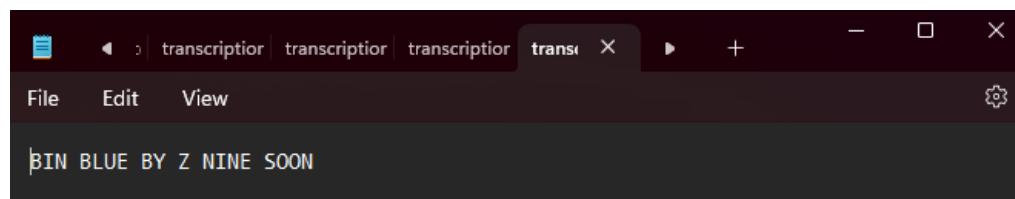


Figure 6.26: The download text file content, which is same as what has been modified at the textarea of the web application.

6.3.5 Preview Sample Module

The purpose of having the preview sample module is to allow the users to preview the sample output of the web application after completed the lipreading process. This ensures that the users who wonder about the output can have a look to improve their understanding on the usage of the system.

This module consists of some images and description texts to explain the content of the images. It also consists of a sample output video that allows the users to playback and view the sample output video.

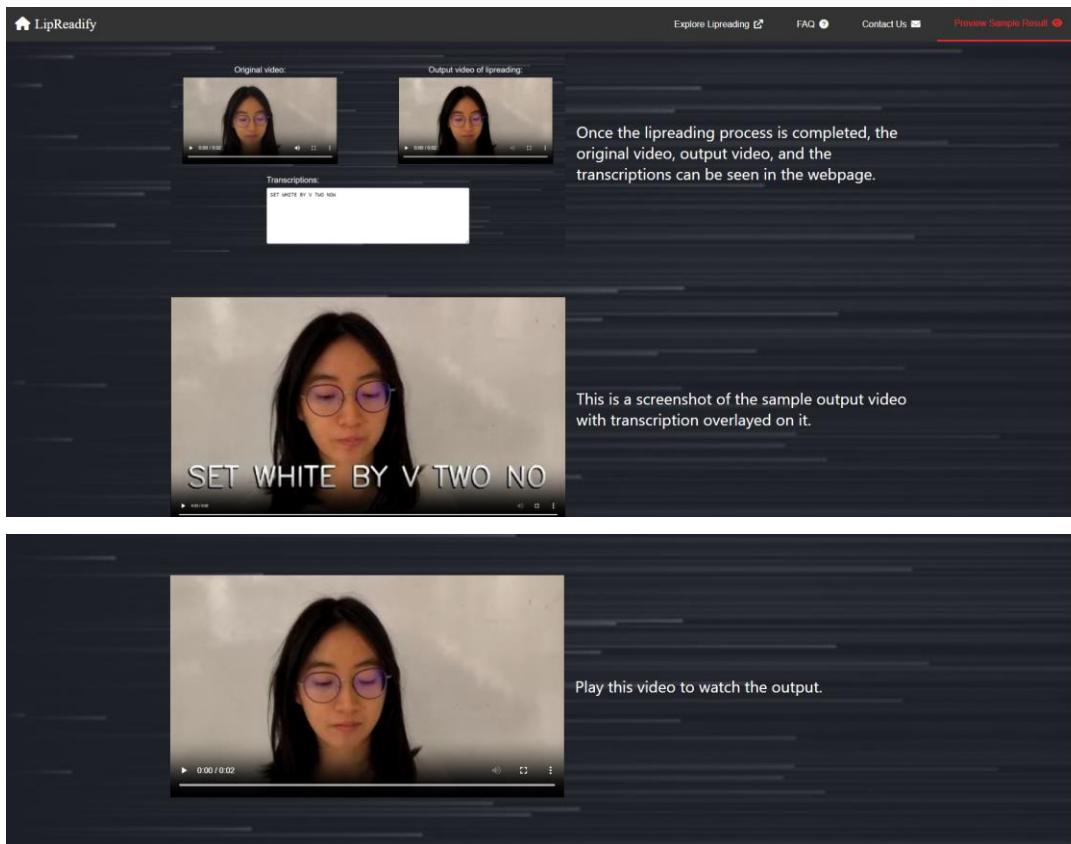


Figure 6.27: The screenshots of the web application for the preview sample module.

Below shows some screenshot of the code snippet used in developing the web application for the preview sample module.

```

src > pages > Preview.js > ...
1 import React from 'react';
2 import Navbar from '../components/Navbar';
3
4 function Preview() {
5   return (
6     <div className="App">
7       <header className="Header">
8         <Navbar />
9         <main className="Preview-content">
10           <div className="Preview-item">
11             
12             <div className="Preview-description">
13               <p>Once the lipreading process is completed, the original video, output video, and the transcriptions can be seen in the webpage.</p>
14             </div>
15           </div>
16
17           <div className="Preview-item">
18             <div className="Preview-description">
19               <p>This is a screenshot of the sample output video with transcription overlayed on it.</p>
20             </div>
21             
22           </div>
23
24           <div className="Preview-item">
25             <video controls className="Preview-video">
26               <source src="/image/preview_vid.mp4" type="video/mp4" />
27               Your browser does not support the video tag.
28             </video>
29             <div className="Preview-description">
30               <p>Play this video to watch the output.</p>
31             </div>
32           </div>
33         </main>
34       </header>
35     </div>
36   );
37 }
38 export default Preview;

```

Figure 6.28: The code snippet used to develop preview sample module.

6.3.6 FAQ Module

In the FAQ module, the users are provided with a list of questions related to the web application and their respective answers. By clicking on the default chevron right icon next to the questions, the answer of the question will be displayed. Similarly, by clicking on the chevron down icon, the answers will be hidden. So, based on the questions and answers provided in this module, users can have a better understanding about the web application, and some of their questions or enquiries can be answered.

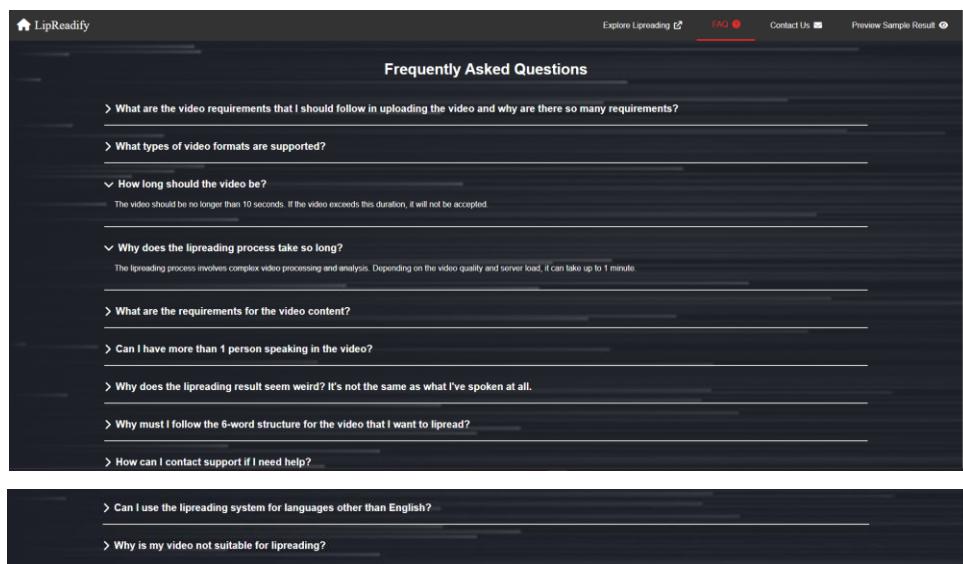


Figure 6.29: Screenshots of the web application FAQ module.

Since the structure of the questions and answers divisions are similar to each other, the code snippet below only shows the division for the first question and answer.

```

src > pages > JS Faq.js > Faq
1 import React, { useState } from 'react';
2 import Navbar from '../components/Navbar';
3
4 function Faq() {
5   const [expandedQuestions, setExpandedQuestions] = useState([]);
6
7   const toggleQuestion = (index) => {
8     setExpandedQuestions(prevState =>
9       prevState.includes(index)
10      ? prevState.filter(qIndex => qIndex !== index)
11      : [...prevState, index]
12    );
13  };
14
15  return (
16    <div className="App">
17      <header className="Header">
18        <Navbar />
19      </header>
20      <div className="Faq-container">
21        <h1>Frequently Asked Questions</h1>
22        <div className="Faq-question">
23          <i onClick={() => toggleQuestion(12)}>
24            <i className={fa $expandedQuestions.includes(12) ? 'fa-chevron-down' : 'fa-chevron-right'}></i>
25            What are the video requirements that I should follow in uploading the video and why are there so many requirements?
26          </i>
27          {expandedQuestions.includes(12) && (
28            <p className="Faq-answer">
29              To ensure accurate lipreading results, your video should follow these requirements:
30              <ul>{/*ul that lists the video requirements */}...
31            </ul>
32            These requirements help maintain the quality and accuracy of the lipreading process. Adhering to them allows the system to better analyze and process the video content.
33          )}
34        </div>
35
36        <div className="Faq-question">
37          <i onClick={() => toggleQuestion(0)}>
38            <i className={fa $expandedQuestions.includes(0) ? 'fa-chevron-down' : 'fa-chevron-right'}></i>
39            What types of video formats are supported?
40          </i>
41        </div>
42      </div>
43    </div>
44  );
45}
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```

Figure 6.30: Code snippet for developing FAQ module with the first question and answer division being shown.

6.3.7 Contact Module

In the contact module, users are allowed to provide suggestions to help improve the web application. Figure 6.31 below shows the screenshot of the web application contact module. The users are required to provide their name, valid email address, and suggestions in order to submit their suggestions to the backend successfully.

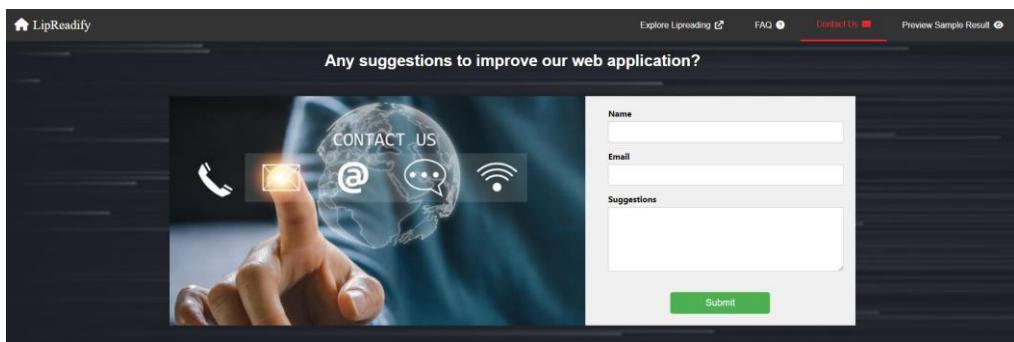


Figure 6.31: Screenshot of the web application contact module.

There are some input validations being performed and error message will be displayed if the users do not enter name, email, suggestions, or do not enter a valid email address with ‘@’ icon included.

Name

Email

Please fill out this field.

yeelin@gmail.com

Suggestions

This is a suggestion

Submit

Figure 6.32: Sample of error message being shown when a user does not enter the name credential and presses the submit button.

Name

Yee Lin

Email

yeelin@gmail.com

Suggestions

It would be good to have a chatbot

Submit

Figure 6.33: Sample of valid user input which is ready to be submitted.

Figure 6.34 below shows the screenshots of the codes used to develop the contact module. Upon the user presses the submit button, the handleSubmit function will handle the form submission by sending a POST request to the server at 'http://localhost:5000/api/contact'. The data being sent to the server include the name, email, and suggestions message.

```
src > pages > JS Contact.js > [o] Contact
1  import React, { useState } from 'react';
2  import axios from 'axios';
3  import Navbar from '../components/Navbar';
4
5  const Contact = () => {
6    const [formData, setFormData] = useState({
7      name: '',
8      email: '',
9      suggestions: ''
10 });
11
12  const [successMessage, setSuccessMessage] = useState('');
13  const [failureMessage, setFailureMessage] = useState('');
14
15  const handleChange = (e) => {
16    const { name, value } = e.target;
17    setFormData({
18      ...formData,
19      [name]: value
20    });
21    setSuccessMessage('');
22    setFailureMessage('');
23  };
24
25  const handleSubmit = async (e) => {
26    e.preventDefault();
27    try {
28      // Post the form data to the server
29      const response = await axios.post('http://localhost:5000/api/contact', formData, {
30        headers: {
31          'Content-Type': 'application/json',
32        }
33      });
34
35      // Check if response status is in the range of 2xx
36      if (response.status >= 200 && response.status < 300) {
37        setFormData({ name: '', email: '', suggestions: '' });
38        setSuccessMessage('Your suggestions has been received');
39        setFailureMessage('');
40      } else {

```

```

41     |         console.log('Failed to submit form');
42     |         setFailureMessage('Failed to submit the form. Please try again.');
43     |         setSuccessMessage('');
44   }
45 } catch (error) {
46   console.error('Error:', error);
47   console.log('Failed to submit form');
48   setFailureMessage('Failed to submit the form. Please try again.');
49   setSuccessMessage('');
50 }
51 );
52
53 return (
54   <div className="App">
55     <header className="Contact-header">
56       <Navbar />
57       <h1 className="contact-title">Any suggestions to improve our web application?</h1>
58       <div className="contact-container">
59         <div className="contact-left">
60           
61         </div>
62
63         <div className="contact-right">
64           <form className="contact-form" onSubmit={handleSubmit}>
65             <div className="form-group">
66               <label htmlFor="name">Name</label>
67               <input
68                 type="text"
69                 id="name"
70                 name="name"
71                 value={formData.name}
72                 onChange={handleChange}
73                 required
74               />
75             </div>
76             <div className="form-group">
77               <label htmlFor="email">Email</label>
78
79               <input
80                 type="email"
81                 id="email"
82                 name="email"
83                 value={formData.email}
84                 onChange={handleChange}
85                 required
86               />
87             </div>
88             <div className="form-group">
89               <label htmlFor="suggestions">Suggestions</label>
90               <textarea
91                 id="suggestions"
92                 name="suggestions"
93                 value={formData.suggestions}
94                 onChange={handleChange}
95                 required
96               ></textarea>
97             </div>
98             <button type="submit" className="submit-button">Submit</button>
99           </form>
100           {successMessage && <p className="success-message">{successMessage}</p>}
101           {failureMessage && <p className="failure-message">{failureMessage}</p>}
102         </div>
103       </header>
104     </div>
105   ];
106 };
107
108 export default Contact;
109

```

Figure 6.34: Code snippet of the contact module.

The backend logic of handling this form submission is to save it to the database, which is handled by Flask. It receives the data in JSON format, extracts the necessary fields, and save them as a new record in the database. Once the saving is completed, the server will respond the 201 status code with a confirmation message of ‘Contact saved successfully’.

```
@app.route('/api/contact', methods=['POST'])
def add_contact():
    data = request.get_json()

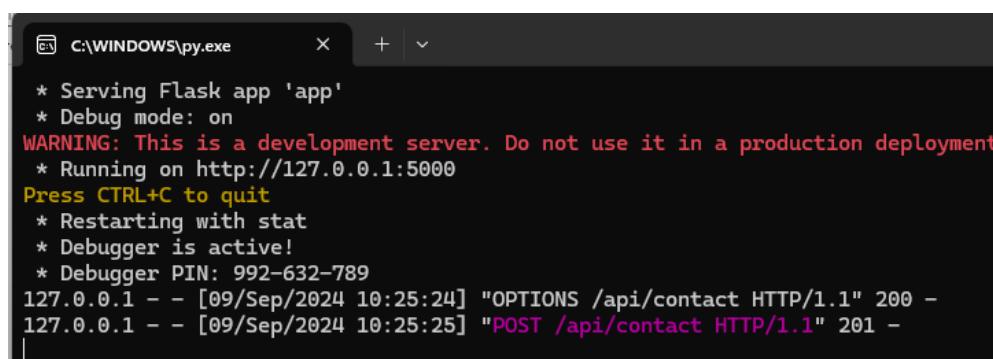
    name = data.get('name')
    email = data.get('email')
    message = data.get('suggestions')

    new_contact = Contact(name=name, email=email, message=message)
    db.session.add(new_contact)
    db.session.commit()

    return jsonify({'message': 'Contact saved successfully'}), 201
```

Figure 6.35: Code snippet of the backend logic of handling the contact form submission.

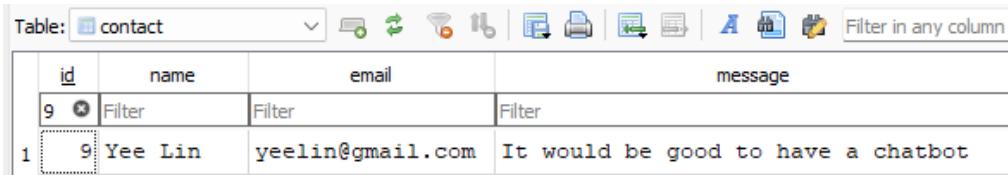
By observing the output of the Flask application below, it can be noticed that the POST request was successfully made to the /api/contact endpoint, and a new record is made in the contact table of the database, supporting with the 201 status code.



```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 992-632-789
127.0.0.1 - - [09/Sep/2024 10:25:24] "OPTIONS /api/contact HTTP/1.1" 200 -
127.0.0.1 - - [09/Sep/2024 10:25:25] "POST /api/contact HTTP/1.1" 201 -
```

Figure 6.36: The output of the Flask application to show that the POST request was made successfully.

At the DB Browser for SQLite application, the contact table of the database is opened. In the table, it is noticed that the data of the contact form submitted previously have been successfully recorded in the database, as shown as figure below.



The screenshot shows a database table named 'contact' with four columns: 'id', 'name', 'email', and 'message'. The table contains two rows of data:

	id	name	email	message
1	9	Yee Lin	yeelin@gmail.com	It would be good to have a chatbot

Figure 6.37: The name, email, and message data submitted in the web application are successfully stored in the database.

6.4 Conclusion

This chapter has provided a complete overview of the implementation of the system starting from the setting up of the application, database, and model until the complete development of the web application. The features developed in the web application are following strictly to the functional requirements mentioned in Chapter 4. The features are categorized into seven different modules, each in charge of their specific roles or usages.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

In this chapter, the unit testing and system usability testing of the web application are conducted. The functionalities of the system are being tested using the unit testing. For the system usability testing, it is evaluated by using the System Usability Scale (SUS), by having five respondents to try out the test scenarios and fill in the user satisfaction survey form. Based on their responses, the SUS score is calculated to determine the grade and adjective rating of the web application.

7.2 Unit Testing

In this project, the unit testing is performed to test every functions of each modules in the web application manually so that the requirements specification is fulfilled. The details of each test cases are included in the tables below. There are total of 47 test cases being prepared and executed.

7.2.1 Video Upload Module

Table 7.1: Unit testing of video upload module.

Upload Video for Lipreading					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC01	Choose a valid video file	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 4. Click on start to lipread button 5. Wait for the loading to complete	Video file	Able to view the output page which contains the original video, output video, and transcriptions	Pass
TC02	Choose a video file with .mpg format for lipreading	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file .mpg format	Video file with .mpg format	Display validation error message	Pass

TC03	Choose a video file which contains spacing in the naming	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file which contains spacing in the naming	Video file which contains spacing in the naming	Display validation error message	Pass
TC04	Choose a video file with video duration more than 10 seconds	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file with video duration more than 10 seconds	Video file with video duration more than 10 seconds	Display validation error message	Pass
TC05	Choose a video file recorded in vertical format	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file recorded in vertical format 4. Wait for loading	Video file recorded in vertical format	Display validation error message	Pass
TC06	Choose a video file that is recorded at a cluttered background	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file that is recorded at a cluttered background 4. Wait for the loading	Video file that is recorded at a cluttered background	Display validation error message	Pass

TC07	Choose a video file that is recorded in shaky condition	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file that is recorded in shaky condition 4. Wait for the loading	Video file that is recorded in shaky condition	Display validation error message	Pass
TC08	Choose a video file that does not contains a person's mouth facing the camera	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file that that does not contains a person's mouth facing the camera 4. Wait for the loading	Video file that that does not contains a person's mouth facing the camera	Display validation error message	Pass
TC09	Choose a video file that contains 2 person speaking in the video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file that contains 2 person speaking in the video 4. Wait for the loading	Video file that contains 2 person speaking in the video	Display validation error message	Pass
View Video Requirements					

Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC10	Display the video requirements in the explore lipreading page	1. Navigate to explore lipreading page 2. View the video requirements in the page	-	The video requirements in the page can be seen	Pass
TC11	Hide the video requirements	1. Navigate to explore lipreading page 2. Click on the chevron down icon	-	The video requirements are hidden	Pass
TC12	Expand the hidden video requirements	1. Navigate to explore lipreading page 2. Click on the chevron down icon 3. Click on the chevron right icon	-	The hidden video requirements are expanded	Pass
Playback Sample Video					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC13	Play the video	1. Navigate to explore lipreading page 2. Click on the play button of the sample video	-	The video plays	Pass

TC14	Pause the video	1. Navigate to explore lipreading page 2. Click on the play button of the sample video 3. Click on the pause button of the sample video	-	The video pauses	Pass
TC15	Set the video into full screen mode	1. Navigate to explore lipreading page 2. Click on the full screen button of the sample video	-	The video is displayed in full screen mode	Pass
TC16	Set the video into picture in picture mode	1. Navigate to explore lipreading page 2. Click on the ellipsis icon at the sample video 3. Click on the picture in picture button	-	The video is displayed in picture in picture mode	Pass
TC17	Change the video playback speed to 0.5	1. Navigate to explore lipreading page 2. Click on the ellipsis icon at the sample video 3. Click on the playback speed button 4. Click on the 0.5 button	-	The video plays in the speed of 0.5	Pass
Playback Uploaded Video					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC18	Play the video	1. Navigate to explore lipreading page 2. Click choose file button	Video file	The video plays	Pass

		3. Select a valid video file 4. Click on the play button of the uploaded video			
TC19	Pause the video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 4. Click on the play button of the uploaded video 5. Click on the pause button of the uploaded video	Video file	The video pauses	Pass
TC20	Set the video into full screen mode	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 4. Click on the full screen button of the uploaded video	Video file	The video is displayed in full screen mode	Pass
TC21	Set the video into picture in picture mode	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 4. Click on the ellipsis icon at the uploaded video 5. Click on the picture in picture button	Video file	The video is displayed in picture in picture mode	Pass

TC22	Change the video playback speed to 0.5	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 4. Click on the ellipsis icon at the uploaded video 5. Click on the playback speed button 6. Click on the 0.5 button 	Video file	The video plays in the speed of 0.5	Pass
------	--	---	------------	-------------------------------------	------

7.2.2 Lipreading Module

Table 7.2: Unit testing of lipreading module.

Loading Process					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC23	Cancel loading	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a valid video file 	Video file	Display message at explore lipreading page to	Pass

		4. Click on start to lipread button 5. Click on the cancel button		notify that lipreading was cancelled	
--	--	--	--	--------------------------------------	--

7.2.3 Output Video Module

Table 7.3: Unit testing of output video module.

View Output Video					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC24	Display the output video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete	Video file	The output video can be seen	Pass
TC25	Display the caption in the output video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file	Video file	The output video contains the caption same as what	Pass

		4. Click start to lipread button 5. Wait for the loading to complete 6. Play the output video 7. View the caption shown in the video		shown in the transcription textbox at the beginning	
Playback Output Video					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC26	Play output video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Click on the play button of the output video	Video file	The video plays	Pass
TC27	Pause output video	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button	Video file	The video pauses	Pass

		5. Wait for the loading to complete 6. Click on the play button of the output video 7. Click on the pause button of the output video			
TC28	Set output video into full screen mode	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Click on the full screen button of the output video	Video file	The video is displayed in full screen mode	
TC29	Set output video into picture in picture mode	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Click on the ellipsis icon at the output video 7. Click on the picture in picture button	Video file	The video is displayed in picture in picture mode	Pass

TC30	Change output video playback speed to 0.5	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Click on the ellipsis icon at the output video 7. Click on the playback speed button 8. Click on the 0.5 button 	Video file	The video plays in the speed of 0.5	Pass
------	---	--	------------	-------------------------------------	------

7.2.4 Transcription Module

Table 7.4: Unit testing of transcription module.

View Transcription					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status

TC31	Display transcription in the text area	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 	Video file	The predicted transcription can be seen in the text area	Pass
Edit Transcription					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC32	Modify the original transcription in the text area to a new transcription	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Move the mouse cursor to the transcriptions text area 7. Delete the original transcription and replace it with the new transcription 	<ul style="list-style-type: none"> • Video file • New transcription: <p>“This is a new transcription”</p>	The predicted transcription is able to be edited	Pass
Download Transcription					

Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC33	Download transcription that contains original transcription	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Click on the download transcriptions button 7. Open the downloaded text file and see 	Video file	The downloaded text file contains the transcription content same as what written in the web application	Pass
TC34	Download transcription that contains modified transcription	<ol style="list-style-type: none"> 1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Delete the original transcription at the transcriptions text area and replace it with the new transcription 7. Click on the download transcriptions button 	<ul style="list-style-type: none"> • Video file • New transcription: “This is a new transcription” 	The downloaded text file contains the transcription content same as the new transcription written in the web application	Pass

		8. Open the downloaded text file and see			
TC35	Download transcription that is empty	1. Navigate to explore lipreading page 2. Click choose file button 3. Select a video file 4. Click start to lipread button 5. Wait for the loading to complete 6. Remove the original transcription at the transcriptions text area 7. Click on the download transcriptions button 8. Open the downloaded text file and see	Video file	The downloaded text file is empty	Pass

7.2.5 Preview Sample Module

Table 7.5: Unit testing of preview sample module.

View Sample Result					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status

TC36	View the content in Preview Sample Result page	1. Navigate to the preview sample result page 2. View the content in the page	-	All images, texts, and video in this page can be seen	Pass
TC37	Play the video	1. Navigate to the preview sample result page 2. Click on the play button of the video	-	The video plays	Pass
TC38	Pause the video	1. Navigate to the preview sample result page 2. Click on the play button of the video 3. Click on the pause button of the video	-	The video pauses	Pass
TC39	Set the video into full screen mode	1. Navigate to the preview sample result page 2. Click on the full screen button of the video	-	The video is displayed in full screen mode	Pass
TC40	Set the video into picture in picture mode	1. Navigate to the preview sample result page 2. Click on the ellipsis icon at the video 3. Click on the picture in picture button	-	The video is displayed in picture in picture mode	Pass
TC41	Change the video playback speed to 0.5	1. Navigate to the preview sample result page 2. Click on the ellipsis icon at the video 3. Click on the playback speed button 4. Click on the 0.5 button	-	The video plays in the speed of 0.5	Pass

7.2.6 FAQ Module

Table 7.6: Unit testing of FAQ module.

View Frequently Asked Questions					
Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC42	View the answers for every questions	1. Navigate to the preview sample result page 2. Click on every questions one by one	-	The answers for all questions are displayed	Pass

7.2.7 Contact Module

Table 7.7: Unit testing of contact module.

Submit Suggestions

Test Case ID	Test Case Description	Test Execution Steps	Test Data	Expected Result	Test Status
TC43	Provide valid input	1. Navigate to the contact page 2. Fill in name, valid email, and suggestions 3. Click submit button	<ul style="list-style-type: none"> • name • email • suggestions 	<ul style="list-style-type: none"> • Show success message • Database contains the submitted data 	Pass
TC44	Provide empty name input	1. Navigate to the contact page 2. Fill in valid email and suggestions 3. Click submit button	<ul style="list-style-type: none"> • email • suggestions 	Display validation error message	Pass
TC45	Provide empty email	1. Navigate to the contact page 2. Fill in name and suggestions 3. Click submit button	<ul style="list-style-type: none"> • name • suggestions 	Display validation error message	Pass
TC46	Provide invalid email	1. Navigate to the contact page 2. Fill in name, invalid email and suggestions 3. Click submit button	<ul style="list-style-type: none"> • name • invalid email • suggestions 	Display validation error message	Pass

TC47	Provide empty suggestions	1. Navigate to the contact page 2. Click submit button	-	Display validation error message	Pass
------	---------------------------	---	---	----------------------------------	------

7.3 System Usability Testing

The system usability testing of this project is conducted using the System Usability Scale (SUS) to evaluate and analyze the usability of the web application. The reason of using SUS in evaluating the system usability is because of its low cost assessment and reliability in measuring the usability of the system with some simple questions and calculations only (Brooke, 1995). Although the SUS seems simple, there are more than hundreds of studies that validates its usefulness in measuring the effectiveness, efficiency, and ease of use of the system (UIUX Trend, n.d.).

In this project, the system usability testing is conducted using two methods, the face-to-face testing method and the online testing method. For the face-to-face testing, the respondent is invited to try out the web application physically. Figure 7.1 below shows the respondent, Tee Szen Yong is trying out the web application based on the test scenario.



Figure 7.1: The respondent, Tee Szen Yong is conducting the usability testing physically.

For the respondents who are unable to attend the usability testing physically, the testing is conducted via online using the AnyDesk tool which support remote control of the screen. Therefore, the respondents can control the screen to try out the web application from their remote comfort location.

Before the system usability testing is conducted, the respondents are briefed with the background of the web application so that they understand the

usage of the application. They are also given the informed consent form shown in Appendix A to understand their involvement in the testing and sign it. Next, each test scenarios are read to the respondents for them to complete the task by trying out the web application. After the respondents have completed all test scenarios, they are given a User Satisfaction Survey to answer ten rating-scale questions and three short answer questions. The User Satisfaction Survey template is shown in the Table 7.8 below.

Table 7.8: User Satisfaction Survey template that consists of ten rating-scale questions and three short answer questions.

Source: Brooke (1995)

Participant No.: Name:					
Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)
1. I think I would like to use this system frequently.					
2. I found the system unnecessarily complex.					
3. I thought the system was easy to use.					
4. I think that I would need the support of a technical person to be able to use this system.					
5. I found the various functions in this system were well integrated.					

6. I thought there was too much inconsistency in this system.					
7. I would imagine that most people would learn to use this system very quickly.					
8. I found the system very cumbersome to use.					
9. I felt very confident using the system.					
10. I needed to learn a lot of things before I could get going with this system.					

1. What do you like best about the system?

2. What do you like least about the system?

3. Do you have any suggestions for improving the current system?

7.3.1 Test Scenarios

Table 7.9 below shows the test scenarios prepared for the respondents. Each of the scenarios contain a simple task which require the respondents to try out the web application to figure out the answer.

Table 7.9: Usability Testing Scenarios.

Scenario 1 – Video requirements

You've heard that the LipReadify web application can perform lipreading just by uploading video onto it. However, you are not sure about the requirements of the video that it supports. Where would you look for this information?

Scenario 2 – Preview sample result

Imagine that this is your first time using this LipReadify web application. You wonder what would the lipreading output looks like, hence you wish to preview the sample result of the lipreading. Where would you look for the sample result?

Scenario 3 – Perform lipreading on the uploaded video

Imagine that you would like to try out the lipreading feature of the web application. You followed all the video requirements in preparing this video and named it as ‘video.mp4’. Now, you would like to upload this video to the web application and start the lipreading to see the output. How would you do?

Scenario 4 – Edit and download transcription

Continue with scenario 3, the system has completed the lipreading process and it displays the output to the screen. Imagine that the transcriptions displayed on the screen is ‘BLIN BLUEZ BY Z NINE AGAIN’. However, you think that there are typo mistakes at the first 2 words because it should be ‘BIN BLUE BY Z NINE AGAIN’. You want to save the modified transcription in your own device. How would you do?

Scenario 5 – Contact support

Imagine that you are a user named Brandon. Your email is brandon@gmail.com. You think that the output video of the lipreading is better to include the original audio. Therefore, you wish to provide this suggestion to support team of this web application. How would you do?

Scenario 6 – Look for answer from Frequently Asked Question (FAQ)

Imagine that you wonder why does the lipreading process of the web application takes so long. You wish to look for the answer for this question. How would you do to obtain your answer?

For scenario 3, it requires the respondents to take a video of their own so that they can try out the web application using their own video. Figure 7.2 below shows some screenshots of the video that have been taken by the respondents.

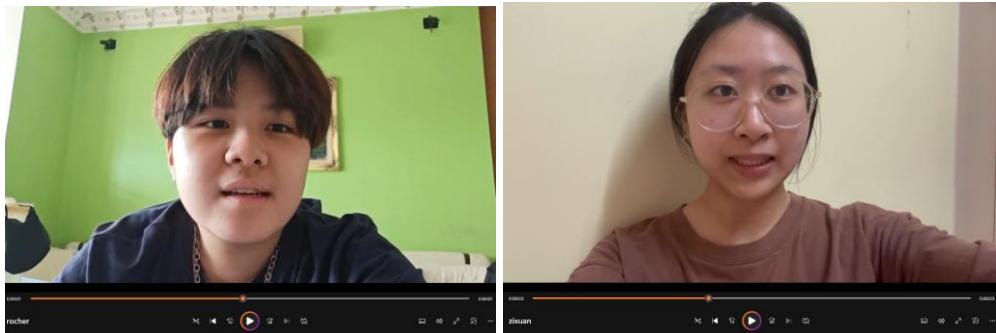


Figure 7.2: Screenshots of the video taken by the respondents during the usability testing.

7.3.2 Test Results of System Usability Testing

Five respondents have been selected to conduct the system usability testing by trying out the seven scenarios written in section 7.3.1 (Nielsen, 2000). After trying out the scenarios, they are required to fill in the user satisfaction survey form in which their responses can be found in the Appendix B.

Based on their responses, the average SUS score is calculated to determine the grade and the adjective rating of the system. The calculation steps to obtain the total score and the SUS score for each respondents is listed as following:

- i. For odd-numbered questions (question 1, 3, 5, 7, 9), subtract one from the usability score.
- ii. For even-numbered questions (question 2, 4, 6, 8, 10), subtract the usability score from five.

- iii. Sum up the usability scores that have been converted in step i and ii. This will be the total score.
- iv. Multiply the total score with 2.5. This will contribute to the SUS score.

After obtaining the five SUS score for the five respondents, the average SUS score can be calculated by adding up the five SUS scores together and divide it by the total number of respondents, which is five in this case.

With the average SUS score being calculated, evaluation of the score can be made using the adjective ratings approach. The grade and adjective rating of the web application can be determined by referring to the Table 7.10 below. It shows a general guideline in interpreting the average SUS score. The score 68 is the average score which falls on the 50th percentile within the scale of 0 to 100. It should be take note that the SUS score is representing a total number falls within the scale of 0 to 100, but not representing a percentage.

Table 7.10: General guideline in interpreting the average SUS score.

Source: UIUX Trend (n.d.)

Average SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Therefore, based on the user satisfaction survey result performed by the five respondents, the usability scores are obtained, and the total score as well as SUS score have been calculated. By calculating the average SUS score, it is identified that the web application in this project obtains an average SUS score of 87.5, falling in the grade A with adjective rating of excellent. This indicates that the web application has high usability with an overall of great effectiveness, efficiency, and satisfaction to the users.

Table 7.11: Summary of the user satisfaction survey results which include the total score and SUS score for each respondents. Based on the SUS scores, the average SUS score, grade, and adjective rating are obtained.

Participants number	Usability score for each question										Total Score	SUS Score
	1	2	3	4	5	6	7	8	9	10		
1	4	1	5	2	5	2	5	1	5	2	36	90
2	5	1	5	2	5	2	5	2	5	1	37	92.5
3	4	1	5	1	4	2	5	4	4	1	33	82.5
4	4	3	4	1	5	1	5	3	4	1	33	82.5
5	5	1	4	2	5	2	5	1	4	1	36	90
Average SUS Score												87.5
Grade												A
Adjective Rating												Excellent

Apart from the ten rating scale questions asked in the user satisfaction survey form, there are also three open-ended questions provided for the respondents to understand their likes and dislikes on the system and their suggestions to improve the system. This allows the respondents to write their actual feelings and thoughts about the web application other than the closed-questions that only allow them to rate the system.

The tables below shows the comments provided by the respondents on the question of ‘What do you like best about the system?’, ‘What do you like least about the system?’, and ‘Do you have any suggestions for improving the current system?’.

Table 7.12: Answers by the respondents about the most liked features.

Question: “What do you like best about the system?”	
Respondent Number	Answer

1	The FAQ section, it perfect answered the question I ponder about this web application
2	From my point of view, I like best with the uploading video feature as it is well designed and organized at the place which is easy to be seen, and I'm able to know what should I do for the video uploading tab.
3	The user-friendly interface
4	Easy to use.
5	It can transcript the lipreading within a short time.

Table 7.13: Answers by the respondents about the most disliked features.

Question: "What do you like least about the system?"	
Respondent Number	Answer
1	None, all good
2	From my point of view, I like least with the layout and design, it can be enhanced by formatting the layout and it can be designed in more fancy way.
3	N/A
4	It takes more time to load the video.
5	The transcript content is limited.

Table 7.14: Answers by the respondents about the suggestions to improve the system.

Question: "Do you have any suggestions for improving the current system?"	
Respondent Number	Answer
1	Would prefer to have a video demonstration on how to use the web application for simple and straight forward understanding
2	From my point of view, the system can be enhanced by embedding AI chat box for user's enquiries in order for user to find out the answer easier and quickly.

3	The time taken to show the output
4	Maybe can add a live chat feature to allow user to communicate directly to support team.
5	The transcript content can be varied other than the six-word structure.

As an overall, from the answers by the respondents, it can be identified that the respondents think that the web application is easy to use and has a user-friendly interface. However, they are less satisfied with the time taken to perform lipreading and the transcription content that the web application supports is limited. Besides that, the respondents suggest to include a chat feature in the web application so that they can communicate with either the AI chatbot or the human support team when they faced any questions in using the web application. They also prefer to have a video demonstration of the way to use the application instead of the texts and images provided by the web application. This is because video is able to keep the viewers engaged, grab their attention effectively, and able to convey the messages in a straightforward way.

7.4 Conclusion

In conclusion, the unit testing and system usability testing have been performed to ensure that the system fulfilled the requirement specifications as well as to analyze the usability of the system. 47 test cases are executed to test out seven different modules of the system. For the system usability testing, it is conducted using System Usability Scale (SUS) and its final result shows that the system achieved a grade A with the adjective rating of Excellent.

CHAPTER 8

CONCLUSION AND RECOMMENDATION

8.1 Conclusion

In conclusion, all objectives listed in Chapter 1 has been successfully achieved.

The objectives include:

- a. To study an existing deep learning model, LipCoordNet for lipreading based on its suitability with this project.
- b. To implement the deep learning model, LipCoordNet in a web application.
- c. To evaluate the usability of the web application using System Usability Scale (SUS) after completed the development of the application.

The first objective is achieved by performing research on 22 papers and websites to study the existing lipreading models. Based on the study, the most popular lipreading model that receives the highest number of citations is the LipNet model. Based on further study, the LipCoordNet model, which is an advanced version of the LipNet model that integrates the lip landmark coordinates is found to have better performance than the LipNet model. Also, by considering the practical perspective, the LipCoordNet model is more compatible with the web application. Therefore, the existing deep learning model has been studied based on its suitability with the project.

The second objective is achieved by implementing the LipCoordNet model in the web application. After completing the development of the frontend of the web application, the Flask application is used as the backend server to handle the HTTP GET and POST requests related to the lipreading feature and to display the output on the screen. With the help of the backend Flask application, the LipCoordNet model has been implemented in the web application and is used to support the inferencing of the uploaded video.

The third objective is achieved by performing the system usability test via the System Usability Scale (SUS) after completing the web application development. Based on the results by the respondents, the web application obtains an average SUS score of 87.5, leading to a grade A with the adjective

rating of excellent. There are a total of 5 respondents participated in the system usability test. The small number of participants involved in the testing has caused the usability test of the web application may not fully address the usability issues arising in the application, therefore more number of respondents is suggested to convey more optimal results in the usability of the web application.

Also, this project has successfully addressed the two problems mentioned in the problem statement section in Section 1.3. The first problem is that the external environment has impeded the listening of the human voice during a communication. In this project, the external environments are specified as the loud background noise environment and the silent environment. Since the lipreading feature applied in this project does not depend on the audio but depends on the lip movements of the speaker instead, therefore the two specified bad acoustic conditions do not affect the lipreading performance in this project. The web application can support the users to upload videos with the speaker speaking in a loud background noise environment or speaking unclearly. It will be able to generate the inference result based on the uploaded video despite the video is recorded in these acoustic conditions. Therefore, it can encounter problems caused by the external environment under these two acoustic conditions.

The second problem is about the manual lipreading and manual recording in human-based lipreading. The manual lipreading is a professional skill that is not easily mastered by everyone. Also, the manual lipreading and recording processes are very tedious, exhausting, and time-consuming. This project has successfully addressed this problem by having a web application which implements the LipCoordNet model to perform the lipreading process. People can visualize the output of the lipreading and download the copy of the output on their device for future reference. Therefore, the lipreading and transcription generation are all done automatically, eliminating the problems aroused by manual lipreading and manual recording.

8.2 Limitations and recommendations for future works

Throughout the development and training of the system in this project, several limitations are identified and listed in the table below. For each of the limitations, recommendations are provided, hoping to enhance the system's functionalities, performance and usability in future iterations.

Table 8.1: Limitations and recommendations of the system.

No.	Limitations	Recommendations
1	The lipreading process in the web application takes long time (around 1 minute).	<ul style="list-style-type: none"> • A more powerful GPU with higher memory, better AI computational capabilities, and higher throughput can be applied to reduce the inference time.
2	The word error rate (WER) and character error rate (CER) of the Asian speakers are lower than the Native speakers.	<ul style="list-style-type: none"> • Model training is suggested to be done by using the dataset of Asian speakers to improve the WER and CER of Asian speakers. • A group of developers or researchers is recommended to work on the data collection and processing of the data together because it is a large workload to process the videos into frames, obtain the lip landmark coordinates file, and obtain the alignments file.
3	The system currently only supports the lipreading of the sentences in the specific 6-word structure.	<ul style="list-style-type: none"> • Train the model using dataset that contains more diversified sentence structures and vocabulary such as BBC LRS2 and LRW dataset so that the web application can support lipreading of more different sentences in the video uploaded by the users.

4	The system currently only supports lipreading of the sentences in English language.	<ul style="list-style-type: none"> • Train the model with sentences in multiple languages so that the system can recognize the lip movements for different languages.
5	The system is not able to lipread video in scenarios where the speaker is moving while speaking, not facing the camera while speaking, or speaking in a cluttered background.	<ul style="list-style-type: none"> • Implement a robust lip detection and tracking algorithm so that the mouth region of interest (ROI) can be continuously adjusted and detected even though the speaker is moving while speaking. • Implement background subtraction technique such as the Gaussian Mixture Models (GMM) to ignore the background and focus on the speaker's lips.
6	The web application currently only supports upload of the video in .mp4 format.	<ul style="list-style-type: none"> • Expand the supported file formats by using tools such as FFmpeg to convert non-MP4 file formats to .mp4 file format.
7	The web application currently only supports lipreading of video that contains a single speaker.	<ul style="list-style-type: none"> • Enhance the lipreading model so that it is able to differentiate between multiple speakers and able to transcribe overlapping speech spoken by multiple speakers at the same time.
8	The web application currently only supports the lipreading of one video per upload.	<ul style="list-style-type: none"> • Enhance the system to support batch processing of multiple videos.
9	Based on the feedback from the system usability testing respondents, they hope to	<ul style="list-style-type: none"> • Develop a live chat feature to allow AI chatbot to respond to the user's questions immediately with some

	<p>include a chat feature in the web application for them to ask questions when they encounter problems.</p>	<p>default answers. The chat feature should also allow the users to communicate with the support team.</p>
--	--	--

REFERENCES

- 121 Captions. (n.d.). *What is forensic lip reading and how can it help you?* Retrieved April 16, 2024, from <https://www.121captions.com/what-is-forensic-lip-reading/professional-lip-reading-services/#:~:text=Forensic%20lip%20reading%20is%20a,particular%20of%20acts%20of%20a%20case>.
- Abrar, M. A., Nafiul Islam, A. N. M., Hassan, M. M., Islam, M. T., Shahnaz, C., & Fattah, S. A. (2019). Deep lip reading-a deep learning based lip-reading software for the hearing impaired. *IEEE Region 10 Humanitarian Technology Conference, R10-HTC, 2019-November.* <https://doi.org/10.1109/R10-HTC47129.2019.9042439>
- AI & Insights. (2023, June 27). *Convolutional Neural Networks (CNNs) in Computer Vision.* <https://medium.com/@AIandInsights/convolutional-neural-networks-cnns-in-computer-vision-10573d0f5b00>
- AltexSoft. (n.d.). *Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices.*
- Amazon Web Services. (n.d.). *What is Machine Learning?* Retrieved March 31, 2024, from <https://aws.amazon.com/what-is/machine-learning/>
- Asri, S. A., Sunaya, I. G. A. M., Rudiaستari, E., & Setiawan, W. (2018). Web Based Information System for Job Training Activities Using Personal Extreme Programming (PXP). *Journal of Physics: Conference Series*, 953(1). <https://doi.org/10.1088/1742-6596/953/1/012092>
- Assael, Y., Shillingford, B., Whiteson, S., & Freitas, N. (2016, November 5). *LIPNET: END-TO-END SENTENCE-LEVEL LIPREADING.* <https://youtube.com/playlist?list=PLXkuFIFnXUAPIrXKgtIpctv2NuSo7xw3k>
- Bangor, A., Kortum, P., & Miller, J. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Stud.*, 4(3), 114–123.
- Better Health Channel. (n.d.). *Hearing loss - lipreading.* Retrieved April 2, 2024, from <https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/hearing-loss-lipreading>

- Bowden, R., Cox, S., Harvey, R., Lan, Y., Ong, E.-J., Owen, G., & Theobald, B.-J. (n.d.). *Recent developments in automated lip-reading*.
- Britannica Kids. (n.d.). *communication*. Retrieved April 2, 2024, from <https://kids.britannica.com/students/article/communication/273754#:~:text=Communication%20serves%20five%20major%20purposes,in%20a%20form%20of%20communication>.
- Brooke, J. (1995). SUS - A quick and dirty usability scale. *Usability Eval. Ind.*, 189. <https://www.researchgate.net/publication/228593520>
- Condliffe, J. (2016, November 21). *AI Has Beaten Humans at Lip-reading*. <https://www.technologyreview.com/2016/11/21/69566/ai-has-beaten-humans-at-lip-reading/>
- Cooke, M., Barker, J., Cunningham, S., & Shao, X. (2006). An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America*, 120(5), 2421–2424. <https://doi.org/10.1121/1.2229005>
- Deery, M. (2023, August 30). *What Is Flask and How Do Developers Use It? A Quick Guide*. <https://careerfoundry.com/en/blog/web-development/what-is-flask/>
- Drug Discovery Pro. (n.d.). *Pros and Cons of Machine Learning and Deep Learning*. Retrieved March 31, 2024, from <https://drugdiscoverypro.com/pros-and-cons-of-machine-learning-and-deep-learning/>
- Feng, D., Yang, S., Shan, S., & Chen, X. (2020). *Learn an Effective Lip Reading Model without Pains*. <http://arxiv.org/abs/2011.07557>
- Fenghour, S., Chen, D., Guo, K., & Xiao, P. (2020). Lip Reading Sentences Using Deep Learning with only Visual Cues. *IEEE Access*, 8, 215516–215530. <https://doi.org/10.1109/ACCESS.2020.3040906>
- Furbush, J. (2021, July 27). *Understand these 5 key deep learning classification metrics for better application success*. <https://www.cognex.com/en-my/blogs/deep-learning/understanding-deep-learning-metrics#:~:text=The%20most%20commonly%20used%20metric,message%20in%20a%20single%20number>.

- GeeksforGeeks. (n.d.). *Decision Tree*. Retrieved March 31, 2024, from <https://www.geeksforgeeks.org/decision-tree/>
- Gillioz, A., Casas, J., Mugellini, E., & Khaled, O. A. (2020). Overview of the Transformer-based Models for NLP Tasks. *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020*, 179–183. <https://doi.org/10.15439/2020F20>
- Google Cloud. (n.d.). *Supervised Learning*. Retrieved March 31, 2024, from <https://cloud.google.com/discover/what-is-supervised-learning#section-3>
- Google for Developers. (n.d.). *Classification: Accuracy*. Retrieved March 24, 2024, from <https://developers.google.com/machine-learning/crash-course/classification/accuracy>
- Hao, M., Mamut, M., Yadikar, N., Aysa, A., & Ubul, K. (2020). A survey of research on lipreading technology. In *IEEE Access* (Vol. 8, pp. 204518–204544). Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/ACCESS.2020.3036865>
- HuggingFace. (n.d.-a). *Enhanced GRID Corpus with Lip Landmark Coordinates*. Retrieved April 12, 2024, from <https://huggingface.co/datasets/SilentSpeak/EGCLLC>
- HuggingFace. (n.d.-b). *LipCoordNet: Enhanced Lip Reading with Landmark Coordinates*. Retrieved April 12, 2024, from <https://huggingface.co/SilentSpeak/LipCoordNet>
- IBM. (n.d.-a). *What are recurrent neural networks?* Retrieved March 31, 2024, from <https://www.ibm.com/topics/recurrent-neural-networks>
- IBM. (n.d.-b). *What is supervised learning?* Retrieved March 31, 2024, from <https://www.ibm.com/topics/supervised-learning>
- IBM. (2023, December 27). *What are support vector machines (SVMs)?* <https://www.ibm.com/topics/support-vector-machine>
- Ilieva, S., Dzhurov, Y., & Krasteva, I. (2009). *Personal Extreme Programming—An Agile Process for Autonomous Developers*. <https://www.researchgate.net/publication/229046039>
- Javatpoint. (n.d.). *K-Means Clustering Algorithm*. Retrieved March 31, 2024, from <https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning>

- Kansal, S. (2021, June 17). *Machine Learning: How to Build Scalable Machine Learning Models*. <https://www.codementor.io/blog/scalable-ml-models-6rvtf8dsd>
- Khalifa, A. (2018, August 28). *Concentration Fatigue: What Is It & How It Affects Deaf And Hard Of Hearing People?* <https://hearmeoutcc.com/concentration-fatigue-affects-deaf-people/>
- Koumparoulis, A., & Potamianos, G. (2019). MobilipNet: Resource-efficient deep learning based lipreading. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2019-September*, 2763–2767. <https://doi.org/10.21437/Interspeech.2019-2618>
- LinkedIn. (2024, January 23). *What are some tips for applying deep learning to unstructured data?* <https://www.linkedin.com/advice/1/what-some-tips-applying-deep-learning-unstructured-ngcec>
- Lu, Y., & Li, H. (2019). Automatic lip-reading system based on deep convolutional neural network and attention-based long short-term memory. *Applied Sciences (Switzerland)*, 9(8). <https://doi.org/10.3390/app9081599>
- Lu, Y., Tian, H., Cheng, J., Zhu, F., Liu, B., Wei, S., Ji, L., & Wang, Z. L. (2022). Decoding lip language using triboelectric sensors with deep learning. *Nature Communications*, 13(1). <https://doi.org/10.1038/s41467-022-29083-0>
- Madhavan, S., & Jones, M. T. (2017, September 7). *Deep learning architectures*. <https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/>
- Margam, D. K., Aralikatti, R., Sharma, T., Thanda, A., K, P. A., Roy, S., & Venkatesan, S. M. (2019). *LipReading with 3D-2D-CNN BLSTM-HMM and word-CTC models*. <http://arxiv.org/abs/1906.12170>
- Martinez B, Ma P, Petridis S, & Pantic M. (2020). *Lipreading using Temporal Convolutional Networks*. IEEE.
- May, P., Martinezy, B., Petridis, S., & Pantic, M. (2021a). Towards practical lipreading with distilled and efficient models. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*,

- 2021-June, 7608–7612.
<https://doi.org/10.1109/ICASSP39728.2021.9415063>
- May, P., Martinezy, B., Petridis, S., & Pantic, M. (2021b). Towards practical lipreading with distilled and efficient models. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2021-June,* 7608–7612.
<https://doi.org/10.1109/ICASSP39728.2021.9415063>
- Mesbah, A., Berrahou, A., Hammouchi, H., Berbia, H., Qjidaa, H., & Daoudi, M. (2019). Lip reading with Hahn Convolutional Neural Networks. *Image and Vision Computing, 88,* 76–83.
<https://doi.org/10.1016/j.imavis.2019.04.010>
- Nanos, G. (2024, March 18). *Machine Learning: Flexible and Inflexible Models.*
<https://www.baeldung.com/cs/ml-flexible-and-inflexible-models>
- Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B, 4,* 51–62. <https://doi.org/10.20544/horizons.b.04.1.17.p05>
- Nemani, P., Krishna, G. S., Ramisetty, N., Sai, B. D. S., & Kumar, S. (2023). Deep Learning-Based Holistic Speaker Independent Visual Speech Recognition. *IEEE Transactions on Artificial Intelligence, 4(6),* 1705–1713. <https://doi.org/10.1109/TAI.2022.3220190>
- Nielsen, J. (2000, March 18). *Why You Only Need to Test with 5 Users.*
<https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>
- Nordquist, R. (2020, August 25). *The Basic Elements of the Communication Process.* <https://www.thoughtco.com/what-is-communication-process-1689767>
- Petridis S, Li Z, & Pantic M. (2017). *End-To-End Visual Speech Recognition With LSTMs.* 6567.
- Pingchuan, M., Brais, M., Yujiang, W., Stavros, P., Jie, S., & Maja, P. (n.d.). *Lipreading using Temporal Convolutional Networks.* Retrieved April 2, 2024, from
https://github.com/mpc001/Lipreading_using_Temporal_Convolutional_Networks

- Piper, K. (2019, April 4). *The AI breakthrough that won the “Nobel Prize of computing.”* <https://www.vox.com/future-perfect/2019/4/4/18294978/ai-turing-award-neural-networks#:~:text=Last%20week%2C%20the%20%241%20million%20Turing%20Award%20%20E2%80%94,intelligence%3A%20Yann%20LeCun%2C%20Geoffrey%20Hinton%2C%20and%20Yoshua%20Bengio>.
- Priy, S. (2024). *Introduction to SQLite.* <https://www.geeksforgeeks.org/introduction-to-sqlite/>
- Questions and Answers in MRI. (n.d.). *Types of Deep Neural Networks.* Retrieved March 31, 2024, from <https://mriquestions.com/deep-network-types.html>
- Rao, R. (2024, July 25). *Tensor Cores vs CUDA Cores: The Powerhouses of GPU Computing from Nvidia.* <https://www.wevolver.com/article/tensor-cores-vs-cuda-cores>
- Saitoh, T., & Kubokawa, M. (2019). *LiP25w: Word-level Lip Reading Web Application for Smart Device.* <http://www.slab.ces.kyutech.ac.jp/SSSD/>
- Salakhutdinov, R., Tenenbaum, J. B., & Torralba, A. (2013). Learning with hierarchical-deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1958–1971. <https://doi.org/10.1109/TPAMI.2012.269>
- Sauro, J. (2018, September 19). *5 Ways to Interpret a SUS Score.* <https://measuringu.com/interpret-sus-score/>
- Sheng, C., Zhu, X., Xu, H., Pietikainen, M., & Liu, L. (2022). Adaptive Semantic-Spatio-Temporal Graph Convolutional Network for Lip Reading. *IEEE Transactions on Multimedia*, 24, 3545–3557. <https://doi.org/10.1109/TMM.2021.3102433>
- Shillingford, B., Assael, Y., Hoffman, M. W., Paine, T., Hughes, C., Prabhu, U., Liao, H., Sak, H., Rao, K., Bennett, L., Mulville, M., Coppin, B., Laurie, B., Senior, A., & de Freitas, N. (2018). *Large-Scale Visual Speech Recognition.* <http://arxiv.org/abs/1807.05162>
- Stackademic. (2023, December 29). *The Difference Between Back Propagation and Forward Propagation in Deep Learning.*

- <https://blog.stackademic.com/the-difference-between-back-propagation-and-forward-propagation-in-deep-learning-2b2248e6d00c>
- Tsourounis, D., Kastaniotis, D., & Fotopoulos, S. (2021). Lip reading by alternating between spatiotemporal and spatial convolutions. *Journal of Imaging*, 7(5). <https://doi.org/10.3390/jimaging7050091>
- tutorialspoint. (n.d.). *ReactJS - Architecture*. Retrieved August 31, 2024, from https://www.tutorialspoint.com/reactjs/reactjs_architecture.htm
- UIUX Trend. (n.d.). *Measuring and Interpreting System Usability Scale (SUS)*. Retrieved September 4, 2024, from <https://uiuxtrend.com/measuring-system-usability-scale-sus/>
- University of Wisconsin-Milwaukee. (n.d.). *Online Forced Aligner*. Retrieved September 10, 2024, from <https://web.uwm.edu/forced-aligner/usability.gov>.
- (n.d.). *System Usability Scale (SUS)*. Retrieved March 19, 2024, from <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>
- W3Schools. (n.d.). *React JSX*. Retrieved August 31, 2024, from https://www.w3schools.com/react/react_jsx.asp
- Wand, M., Koutník, J., & Schmidhuber, J. (2016). Lipreading with long short-term memory. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016-May, 6115–6119. <https://doi.org/10.1109/ICASSP.2016.7472852>
- Weng, X., & Kitani, K. (2019). *Learning Spatio-Temporal Features with Two-Stream Deep 3D CNNs for Lipreading*. <http://arxiv.org/abs/1905.02540>
- Xing, G., Han, L., Zheng, Y., & Zhao, M. (2023). Application of deep learning in Mandarin Chinese lip-reading recognition. *Eurasip Journal on Wireless Communications and Networking*, 2023(1). <https://doi.org/10.1186/s13638-023-02283-y>
- Xu, K., Li, D., Cassimatis, N., & Wang, X. (2018). LCANet: End-to-end lipreading with cascaded attention-CTC. *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, 548–555. <https://doi.org/10.1109/FG.2018.00088>
- Yargic, A., & Dogan, M. (2013). A lip reading application on MS Kinect camera. *2013 IEEE International Symposium on Innovations in Intelligent Systems*

and Applications, IEEE INISTA 2013.
<https://doi.org/10.1109/INISTA.2013.6577656>

APPENDICES

Appendix A: Informed consent form

Consent & Recording Release Form

I agree to participate in the study conducted and recorded by Lau Yee Lin for her Final Year Project purpose.

I understand and consent to the use and release of the recording by Lau Yee Lin for her Final Year Project purpose. I understand that the information and recording is for research purposes only and that my name and image will not be used for any other purpose. I relinquish any rights to the recording and understand the recording may be copied and used by Lau Yee Lin without further permission.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date: 3 September 2024

Please write your name: Tee Szen Yong

Please sign your name: _____


Thank you!

We appreciate your participation

Consent & Recording Release Form

I agree to participate in the study conducted and recorded by Lau Yee Lin for her Final Year Project purpose.

I understand and consent to the use and release of the recording by Lau Yee Lin for her Final Year Project purpose. I understand that the information and recording is for research purposes only and that my name and image will not be used for any other purpose. I relinquish any rights to the recording and understand the recording may be copied and used by Lau Yee Lin without further permission.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date: 3 September 2024

Please write your name: Ting Hee



Please sign your name: _____

Thank you!

We appreciate your participation

Consent & Recording Release Form

I agree to participate in the study conducted and recorded by Lau Yee Lin for her Final Year Project purpose.

I understand and consent to the use and release of the recording by Lau Yee Lin for her Final Year Project purpose. I understand that the information and recording is for research purposes only and that my name and image will not be used for any other purpose. I relinquish any rights to the recording and understand the recording may be copied and used by Lau Yee Lin without further permission.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date: 3 September 2024

Please write your name: Tan Rocher



Please sign your name: _____

Thank you!

We appreciate your participation

Consent & Recording Release Form

I agree to participate in the study conducted and recorded by Lau Yee Lin for her Final Year Project purpose.

I understand and consent to the use and release of the recording by Lau Yee Lin for her Final Year Project purpose. I understand that the information and recording is for research purposes only and that my name and image will not be used for any other purpose. I relinquish any rights to the recording and understand the recording may be copied and used by Lau Yee Lin without further permission.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date: 3 September 2024

Please write your name: Hew Zi Xuan

Please sign your name: _____ 

Thank you!

We appreciate your participation

Consent & Recording Release Form

I agree to participate in the study conducted and recorded by Lau Yee Lin for her Final Year Project purpose.

I understand and consent to the use and release of the recording by Lau Yee Lin for her Final Year Project purpose. I understand that the information and recording is for research purposes only and that my name and image will not be used for any other purpose. I relinquish any rights to the recording and understand the recording may be copied and used by Lau Yee Lin without further permission.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date: 3 September 2024

Please write your name: Wong Khai Xin

Please sign your name: 

Thank you!

We appreciate your participation

Appendix B: Usability test responses

Participant No.: 1						
Name: Tee Szen Yong						
Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)	
1. I think I would like to use this system frequently.				✓		
2. I found the system unnecessarily complex.	✓					
3. I thought the system was easy to use.					✓	
4. I think that I would need the support of a technical person to be able to use this system.		✓				
5. I found the various functions in this system were well integrated.					✓	
6. I thought there was too much inconsistency in this system.		✓				
7. I would imagine that most people would learn to use this system very quickly.					✓	
8. I found the system very cumbersome	✓					

to use.					
9. I felt very confident using the system.					✓
10. I needed to learn a lot of things before I could get going with this system.		✓			

1. What do you like best about the system?

The FAQ section, it perfect answered the question I ponder about this web application

2. What do you like least about the system?

None, all good

3. Do you have any suggestions for improving the current system?

Would prefer to have a video demonstration on how to use the web application for simple and straight forward understanding

Participant No.: 2					
Name:	Ting Hee				
Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)
1. I think I would like to use this system frequently.					✓
2. I found the system unnecessarily complex.	✓				
3. I thought the system was easy to use.					✓
4. I think that I would need the support of a technical person to		✓			

be able to use this system.				
5. I found the various functions in this system were well integrated.				✓
6. I thought there was too much inconsistency in this system.		✓		
7. I would imagine that most people would learn to use this system very quickly.				✓
8. I found the system very cumbersome to use.		✓		
9. I felt very confident using the system.				✓
10. I needed to learn a lot of things before I could get going with this system.	✓			

1. What do you like best about the system?

From my point of view, I like best with the uploading video feature as it is well designed and organized at the place which is easy to be seen, and I'm able to know what should I do for the video uploading tab.

2. What do you like least about the system?

From my point of view, I like least with the layout and design, it can be enhanced by formatting the layout and it can be designed in more fancy way.

3. Do you have any suggestions for improving the current system?

From my point of view, the system can be enhanced by embedding AI chat box for user's enquiries in order for user to find out the answer easier and quickly.

Participant No.: 3					
Name: Tan Rocher					
Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)
1. I think I would like to use this system frequently.				✓	
2. I found the system unnecessarily complex.	✓				
3. I thought the system was easy to use.					✓
4. I think that I would need the support of a technical person to be able to use this system.	✓				
5. I found the various functions in this system were well integrated.				✓	
6. I thought there was too much inconsistency in this system.		✓			
7. I would imagine that most people would learn to use this system very quickly.					✓
8. I found the system very cumbersome to use.				✓	

9. I felt very confident using the system.				✓	
10. I needed to learn a lot of things before I could get going with this system.	✓				

1. What do you like best about the system?

The user-friendly interface

2. What do you like least about the system?

N/A

3. Do you have any suggestions for improving the current system?

The time taken to show the output

Participant No.: 4					
Name: Hew Zi Xuan					
Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)
1. I think I would like to use this system frequently.				✓	
2. I found the system unnecessarily complex.			✓		
3. I thought the system was easy to use.				✓	
4. I think that I would need the support of a technical person to be able to use this system.	✓				

5. I found the various functions in this system were well integrated.					✓
6. I thought there was too much inconsistency in this system.	✓				
7. I would imagine that most people would learn to use this system very quickly.					✓
8. I found the system very cumbersome to use.			✓		
9. I felt very confident using the system.				✓	
10. I needed to learn a lot of things before I could get going with this system.	✓				

1. What do you like best about the system?

Easy to use.

2. What do you like least about the system?

It takes more time to load the video.

3. Do you have any suggestions for improving the current system?

Maybe can add a live chat feature to allow user to communicate directly to support team.

Participant No.: 5

Name: Wong Khai Xin

Question	Strongly Disagree (1)	(2)	Neutral (3)	(4)	Strongly Agree (5)
1. I think I would like to use this system frequently.					✓
2. I found the system unnecessarily complex.	✓				
3. I thought the system was easy to use.				✓	
4. I think that I would need the support of a technical person to be able to use this system.		✓			
5. I found the various functions in this system were well integrated.					✓
6. I thought there was too much inconsistency in this system.		✓			
7. I would imagine that most people would learn to use this system very quickly.					✓
8. I found the system very cumbersome to use.	✓				
9. I felt very confident using the system.				✓	

10. I needed to learn a lot of things before I could get going with this system.	✓				
--	---	--	--	--	--

1. What do you like best about the system?

It can transcript the lipreading within a short time.

2. What do you like least about the system?

The transcript content is limited.

3. Do you have any suggestions for improving the current system?

The transcript content can be varied other than the six-word structure.