

LECTURE DIGEST – AUTO SUMMARIZATION AND KEY POINTS

RECOGNITION

BY

MELISSA YAP CHIA CHEAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2024

REPORT STATUS DECLARATION FORM

Title: LECTURE DIGEST – AUTO SUMMARIZATION AND KEY POINTS RECOGNITION

Academic Session: JAN 2024

I MELISSA YAP CHIA CHEAN
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

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

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

10, LENGKOK RELAU 2,
TAMAN SERI RELAU,
11900 BAYAN LEPAS,
PULAU PINANG

Dr Aun Yichiet
Supervisor's name

Date: 26/4/2024

Date: 26/4/2024

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY OF INFORMATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26/4/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that *Melissa Yap Chia Chean* (ID No: 20ACB01674) has completed this final year project/ dissertation/ thesis* entitled “ LECTURE DIGEST – AUTO SUMMARIZATION AND KEY POINTS RECOGNITION” under the supervision of DR. AUN YICHUET (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology.

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


Yours truly,



(*Melissa Yap Chia Chean*)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**LECTURE DIGEST – AUTO SUMMARIZATION AND KEY POINTS RECOGNITION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : Melissa Yap Chia Chean

Date : 26/4/2024

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Aun Yichiet who has given me this bright opportunity to engage in this project. I am grateful for his guidance and trust in my ability to finish this project. It is my first step to establish a career in Computer Vision and Large Language Model field study. A million thanks to you.

ABSTRACT

Addressing the diverse learning preferences of students who benefit from visual, textual, or auditory cues, we developed Lecture Digest - to automatically summarize lecture videos into multiple types of lecture summary. Lecture Digest provides the option of viewing text summaries, navigating to important sections of the video by clicking chapters, and interacting with chat assistance. In text summarization, we use Google's Speech-to-Text technology to convert spoken words in videos into written text before the transcription feeds into GPT-3.5 to extract important points based on semantic understanding. Meanwhile, in video summarization, we use a custom gesture recognition system based on the MediaPipe framework to recognize hand gestures that can signify important key points. The intuition is that speakers would normally perform some distinguishable hand gesture when emphasizing important points. Then, we mark the timestamp of this landmark into the video for the user to easily skip ahead and rewind to these important segments. Lastly, a chat module acts as a virtual tutor, enabling students to engage deeply with the video content. Users can query the module to pull up information related to the video, review frequently asked questions, and utilize interactive tools like flashcards and quizzes to test their understanding. The experimental results on 60 UTAR students from different faculties showed that all of them are satisfied with the functionalities provided by the Lecture Digest application, proving it to be an effective tool in boosting the student's academic performance.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xi
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	2
1.3 Project Scope and Direction	3
1.4 Contributions	3
1.5 Report Organization	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Speech-to-Text Technology	5
2.1.1 Open-Source Speech-to-Text	5
2.1.2 Paid Speech-to-Text API	6
2.2 Google MediaPipe Framework	9
2.3 Gesture Recognition	10
2.4 Existing Summarization Tools	11
2.4.1 Text Summarization Tools	11
2.4.2 Video Summarization Tools	13
2.4.3 Audio Summarization Tools	16
CHAPTER 3 SYSTEM METHODOLOGY	17
3.1 System Design Diagram	17
3.2 Use Case Diagram	18

3.3	MediaPipe Gesture Recognition Model Architecture	19
3.4	MediaPipe Gesture Recognizer Pipeline	20
3.5	Generative Pre-trained Transformer (GPT) Architecture	21
CHAPTER 4 SYSTEM DESIGN		23
4.1	System Block Diagram	23
4.2	System Components Specifications	24
4.2.1	Video Super-resolution	24
4.2.2	Video Preprocessing	25
4.2.2.1	Audio Extraction	26
4.2.2.2	Audio Normalization	26
4.2.2.3	Store Audio File	26
4.2.2.4	Generate Transcript	26
4.2.3	Text Summary Module	28
4.2.4	Chapter Generation Module	29
4.2.4.1	Train Gesture Recognition Model	29
4.2.4.2	Hand Gesture Recognition	34
4.2.4.3	Integration of Hand Gesture Recognition Process to Generate Video Chapters	34
4.2.5	Chat Module	35
4.2.5.1	Frequently Asked Questions (FAQ) Module	35
4.2.5.2	Question and Answer (QnA) Module	36
4.2.5.3	Flashcard Module	36
4.2.5.4	Quiz Module	36
4.2.6	Rating	36

CHAPTER 5 SYSTEM IMPLEMENTATION	37
5.1 Hardware Setup	37
5.2 Software Setup	37
5.2.1 Platform	37
5.2.2 Storage	38
5.2.3 API	38
5.2.4 Framework	39
5.2.5 Programming Language	39
5.3 Setting and Configuration	40
5.4 System Operation	42
5.4.1 Text Summary Module	45
5.4.2 Chapter Generation Module	47
5.4.3 Chat Module	49
5.4.3.1 Frequently Asked Question (FAQ) Module	49
5.4.3.2 QnA Module	50
5.4.3.3 Flashcard Module	51
5.4.3.4 Quiz Module	52
5.4.4 Rating	52
5.5 Implementation Issues and Challenges	54
5.6 Concluding Remark	54
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	55
6.1 System Testing, Testing Setup and Results	55
6.1.1 Gesture Recognition Model Evaluation	55
6.1.2 User Experience Evaluation	59
6.1.3 Comparison of Text Summary module with other Summary Tool	62
6.1.3.1 Setup	62
6.1.3.2 Comparison of text summary between Lecture Digest and summarizer.tech	64
6.1.3.3 Summary on the comparison of Lecture Digest with other summarizer tools	64
6.2 Project Challenges	65

6.3 Objectives Evaluation	65
6.4 Concluding Remark	66
CHAPTER 7 CONCLUSION AND RECOMMENDATION	67
7.1 Conclusion	67
7.2 Recommendation	68
REFERENCES	69
WEEKLY REPORT	72
POSTER	78
QUESTIONNAIRE	81
PLAGIARISM CHECK RESULT	79

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Key points that are inferred by MediaPipe hand landmark model	9
Figure 2.2	Six gestures useful for making emphasis. Top row: “Box”, “Pyramid” and “Beat” gesture. Botton row: “Pointing”, “Circle” and “Balling” gesture.	11
Figure 2.3	Text summarization with ChatGPT	11
Figure 2.4	Research paper summarization with SciSummary	12
Figure 2.5	Article summarization with Artifact	13
Figure 2.6	YouTube video summarization with YouTube Digest	13
Figure 2.7	TED Talk video summarization with NoteGPT	14
Figure 2.8	Display of summary result in ChatGPT	14
Figure 2.9	YouTube video summarization with Mindgrasp	15
Figure 2.10	Podcast summarization with PodPulse	16
Figure 3.1	System Architecture Diagram	17
Figure 3.2	Use Case Diagram	18
Figure 3.3	MediaPipe Gesture Recognition Model Architecture	19
Figure 3.4	MediaPipe Gesture Recognizer Pipeline	20
Figure 3.5	GPT Architecture	21
Figure 4.1	System Block Diagram	23
Figure 4.2	Lecture Digest Flow	24
Figure 4.3	Original Video	24
Figure 4.4	Enhanced Video	25
Figure 4.5	Video Processing Flow	25
Figure 4.6	Configuration to generate transcription	26
Figure 4.7	Restructure on the transcription output	27
Figure 4.8	Flow of Text Summary Module	28
Figure 4.9	Training of Gesture Recognition Model	29
Figure 4.10	Prepare Dataset	30
Figure 4.11	Load Dataset	31
Figure 4.12	Train model	32

Figure 4.13	Hyperparameter tuning	32
Figure 4.14	Detected gestures in the video	33
Figure 4.15	Export model	33
Figure 4.16	Hand Gesture Recognition Flow	33
Figure 4.17	Generate chapters	34
Figure 4.18	Flow of FAQ Module	35
Figure 4.19	Flow of QnA Module	35
Figure 4.20	Flow of Flashcard Module	36
Figure 4.21	Flow of Quiz Module	36
Figure 4.22	Rating Feature	36
Figure 5.1	Adding MediaPipe dependencies into app-level gradle file	39
Figure 5.2	Creation of Firebase Project	40
Figure 5.3	Adding Firebase dependencies in app-level gradle file	40
Figure 5.4	Enable Cloud Storage and Speech-to-Text API	41
Figure 5.5	Service Account	41
Figure 5.6	Private Key	41
Figure 5.7	Adding Google Cloud Speech API dependency into the app-level gradle file	41
Figure 5.8	Open AI Secret Key	42
Figure 5.9	Welcome Page	42
Figure 5.10	Video upload page	42
Figure 5.11	Choose video file from gallery	43
Figure 5.12	After video is selected	43
Figure 5.13	Video is uploading	44
Figure 5.14	Audio file is synchronized to Google Cloud Storage	44
Figure 5.15	Audio file is synchronized to Google Cloud Storage	44
Figure 5.16	Main Menu Page	45
Figure 5.17	Choosing to view summary in key points (Left), Loading page while video is processing (Center), Loading page after video finish processing (Right)	46
Figure 5.18	Text Summary	46

Figure 5.19	Choosing to view lecture chapters (Left), Loading page while video is processing (Center), Loading page after video finish processing (Right)	47
Figure 5.20	Chapter titles are displayed with the video (Left), Users are navigated to specific part of the video by clicking the chapter titles (Right)	48
Figure 5.21	Chat module	49
Figure 5.22	Frequently Asked Question (FAQ) Module	50
Figure 5.23	QnA Module	50
Figure 5.24	Flashcard Module	51
Figure 5.25	Quiz Module	52
Figure 5.26	Rating Button (Pink)	52
Figure 5.27	Rating Page (Left), Fill in the rating form (Center), After submit rating form (Right)	53
Figure 5.28	Rating is stored into Firebase	53
Figure 6.1	Import libraries	55
Figure 6.2	Visualization functions	55
Figure 6.3	Upload test images	56
Figure 6.4	Test images	57
Figure 6.5	Result of test14.jpg (left), test19.jpg (center), test6.jpg (right)	58
Figure 6.6	Respondent Demographic	59
Figure 6.7	Survey Question 1	59
Figure 6.8	Survey Question 2	60
Figure 6.9	Survey Question 3	60
Figure 6.10	Survey Question 4	61
Figure 6.11	Survey Question 5	61
Figure 6.12	summarize.tech	62
Figure 6.13	Upload video to YouTube Channel	62
Figure 6.14	Add details of video	63
Figure 6.15	Video is uploaded	63
Figure 6.16	Get YouTube link	63
Figure 6.17	Summary from summarizer.tech (Left) and Lecture Digest (Right)	64

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Feature comparison of Paid Speech-to-Text API	6
Table 2.2	Pricing comparison of Paid Speech-to-Text API	7
Table 4.1	Gesture Images	30
Table 5.1	Specifications of laptop	37
Table 6.1	Testing Results	57
Table 6.2	Comparison of Text summary module with other summarizer tools	64

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
FAQ	Frequently Asked Questions
GPT	Generative Pre-trained Transformer
LLM	Large Language Model
QnA	Question and Answer
STT	Speech-to-Text

Chapter 1 Introduction

In this chapter, the problem statement and motivation, project scopes and objectives, contributions, report organization are discussed.

1.1 Problem Statement and Motivation

In today's fast-paced educational environment, students face time constraints when studying and revising their course materials due to the hectic schedules in the semesters. Poor time management and last-minute study habits, compounded by multiple exams clustered together, contribute to anxiety before examination, especially for truant and daydreaming students. The emergence of COVID-19 pandemic has transformed the teaching and learning style into online style [1]. Lecture videos, whether in pre-recorded or live style or not, become an assistance tool for students in studying. According to [2], students prefer to return multiple times to certain points in the video to review the important concept. However, reviewing entire video can be time-consuming.

Hence, students choose to speed up the playback of the lecture videos to shorten the duration of whole video [3]. Nevertheless, it reduces effectiveness in studying, as students' minds are overloaded with large amount of information within a short period, making them not able to digest and absorb the knowledge properly [3]. Therefore, a summarizer tool is brought into play to address the issue above, where it synthesizes summary of the lecture video in different perspectives while preserving the important points at the same time. However, there is lack of summarizer tool which can be used to summarize the lecture videos as most of the existing tools generate summary based on the text or YouTube videos only.

This thesis aims to introduce a user-friendly lecture summarization app that help students to understand the lecture content in fast pace. This project is designed to efficiently condense extensive lecture materials into concise summaries, facilitating better comprehending and revision of important course material. The app offers diverse kinds of summaries, including text summaries, chapters for easy navigation to specific parts of the video, and chatbot functioned as a virtual tutor. This make the app to be a versatile and creative tool to enrich the learning process. Ultimately, this initiative strives to create a productive and engaging educational experience to the individuals involved in learning process.

1.2 Objectives

The objectives of this project are:

1. **To develop a lecture digest – to automatically summarize long lecture videos into text summary and short video summary.**

This project aims to deliver a novel lecture digest application composed of multiple features. The application accepts a lecture video as input and then present users with three options, such as extract key points from the lesson, generate chapters for easy navigation to important section of the video and provide an interactive chat assistance to the users.

2. **To develop a text summarization method using Google Speech-to-Text for transcription and GPT-3.5 Turbo model for summarization**

The application can generate key points based on the video transcription. Specifically, Google Speech-to-Text API will convert the lecture videos into transcription and the transcription will be fed into the Open AI GPT-3.5 turbo model to generate key points. The returned key points will be displayed in bullet points.

3. **To train a custom hand gesture recognition model based on MediaPipe API using transfer learning to recognize important points in the lecture video**

The Lecture Digest app uses MediaPipe Gesture Recognizer API to identify gestures made by lecturer in the video, as the gestures represent the intention of lecturer in emphasizing key points. The timestamp of the corresponding detected gesture is extracted and aligned with the chapters generated by GPT-3.5 Turbo model. When the user clicks on the chapters, the user should be navigated to corresponding part of the video. MediaPipe Model Maker is used to train the gesture recognition model.

4. **To develop a chatbot tutor for user to interact and ask questions based on the lecture video**

This chatbot tutor operates in real-time, offering immediate responses to user inquiries. It provides four types of assistances to users, such as generating frequently asked questions (FAQ), answer to user's questions (QnA), creating flashcards and conducting quiz.

1.3 Project Scope and Direction

The scope of the project is to develop a mobile lecture summarization app designed to automatically condense lengthy lecture videos into text and video summaries. This endeavor aims to generate summaries closely aligned with the core contents of the lectures by integrating various methods, including transcript-based key points recognition, gestures recognition, and chat assistance. Advanced technologies such as computer vision, prompt engineering, Speech-to-Text (STT) and Large Language Model (LLM) are employed to construct the necessary features and models for the final application.

At the end of this project, an Android-based application is delivered. The system accepts various types of lecture videos as input and perform summarization based on the lecture content. The system generates key points from and mark important chapters on the lecture video. Other than that, the system is featured with a chatbot that can answer to user queries, generates frequently asked questions, producing flashcards, and conducting quizzes. The project's success will be evaluated based on the app usability, user experience, together with the response generation efficiency and accuracy.

1.4 Contributions

This lecture digest app hopes to bring wide impact to the educational field by enhancing the learning efficiency in both traditional and online learning environments. The application will provide several contributions to the students.

1. Time saving

The lecture digest application streamlines the learning process by condensing lecture content into key points, helping students grasp essential concepts quickly. Moreover, the generation of video chapters aligned with timestamps highlighting key moments in the video enables students to navigate lengthy recordings efficiently, thus eliminate the time of reviewing the entire video.

2. Personalized learning

This multi-featured application is designed to cater to students' preferences in learning materials as it provides various functionalities such as text summaries, chapter generation, and chat assistance. The students can engage with the content in a way that aligns their personal preferences, paces and needs. Text summaries distill complex

lecture content into key points that facilitate efficient comprehension. The video chapters organize the video into sections, providing students ease navigation to specific topic of interest. Additionally, the virtual tutor in this app provides extra guidance to the students, as well as allowing them to seek clarification and deepen their understanding.

3. Engagement

The application's interactive features, such as conversation assistance and quizzes, create a dynamic learning environment that engages students actively, fostering deeper understanding and retention of the material. By encouraging active participation, the application promotes a more engaging learning experience that can enhance motivation and overall learning outcomes.

4. Improved retention

By engaging in dialogues with chatbots, students actively participate in learning processes. This can help them to enhance memory retention compared to passive methods such as reading textbooks, watching videos without interactive components.

1.5 Report Organization

This report is divided into 7 chapters. Chapter 1 discusses the problem statements and motivation, project scope and objectives, as well as the contribution of the project. In Chapter 2, literature review and previous works by the other authors are presented. Chapter 3 and Chapter 4 discuss about system methodology and system design that is used to develop the application. Chapter 5 presents the system implementation and Chapter 6 discusses about the system evaluation. Lastly, Chapter 7 concludes the project, and recommendations are provided for future improvement.

Chapter 2 Literature Review

2.1 Speech-to-Text Technology

Speech-to-Text involves a computational linguistic process that recognizes and convert spoken language into textual form. This step serves as the initial stage for the summarization task which rely on transcription as its foundation.

2.1.1 Open-Source Speech-to-Text

- **Deep Speech**

Mozilla Deep Speech [4] is an end-to-end speech recognition model that is constructed through the utilization of deep learning techniques. It is excellent in learning and adapting to various languages and dialects. Mozilla Deep Speech demonstrates proficiency not only in processing clear speech, but also in handling the background noise of the input audio. User data undergoes encryption while using this system, ensuring the utmost privacy and security of user information. Hence, it proves to be beneficial for adoption by businesses or individuals who handle with confidential data. However, the model is not generalized, as it performs vary in different environments [5].

- **SpeechBrain**

SpeechBrain [6] is an all-in-one speech toolkit built on PyTorch. It is designed for various speech processing task. SpeechBrain includes the functionalities such as speech recognition, language modeling, audio enhancement and separation, spoken language understanding, multi-microphone processing and etc. It can perform high quality of speech-to-text transcription across various applications, such as dictation, voice search and voice-controlled interface, proving its flexibility in different use cases [4]. However, it poses potential privacy concerns or issues [4].

2.1.2 Paid Speech-to-Text API

Amazon Transcribe [7], Google Speech-to-Text [8], and Microsoft Azure Speech-to-Text [9] are the three leading Speech-to-Text APIs to convert speech into text. They provide nearly similar features to the users, as shown in Table 2.1. Google Speech-to-Text stands out for its robust performance with the ability to handle noises and run locally on the devices as long as the audio data is available in the device.

Table 2.1: Feature comparison of Paid Speech-to-Text API

Feature	Amazon Transcribe	Google Speech-to-Text	Microsoft Azure Speech-to-Text
Real time & Batch transcription	Yes	Yes	Yes
Noise handling	Accuracy of the output will be affected with presence of noise	Able to handle noises from different environment without noise cancellation	Need to create custom model to handle the noise
Formatting	Number transcription, capitalization, punctuation, profanity filter	Profanity filter, punctuation, number transcription, capitalization	Number transcription, capitalization, disfluency removal, punctuation, profanity filter
Multiple language support, Speaker diarization & Customization	Yes	Yes	Yes
Supported input type	Live audio, recorded audio, video	Microphones, audio files	Microphones, audio files, blob storage

Deployment	Cloud-based	<ul style="list-style-type: none"> - Cloud-based or on-premises - Able to run locally on any devices regardless of internet connectivity as long as the voice data resides the device 	Cloud-based or on-premises using containers
------------	-------------	---	---

Based on the Table 2.2, the package offered by Google Speech-to-Text is the most cost-effective, especially for existing users who can access a free tier of service and only incur charges when their usage exceeds certain limits. Furthermore, the fact that Google Speech-to-Text charges users only for successful processing can be advantageous, as it ensures that users are billed only for the results they obtain. In contrast, Amazon Transcribe and Microsoft Azure Speech-to-Text appear to charge users irrespective of whether the processing is successful, which could potentially result in higher costs for users.

Table 2.2: Pricing comparison of Paid Speech-to-Text API

	Amazon Transcribe	Google Speech-to-Text	Microsoft Azure Speech-to-Text
Pricing policy	Pay-as-you-go	Pay-as-you-go	Pay-as-you-go
New sign up	Free services for 12 months with monthly 60 audio minutes analysis	\$300 credits for access of services, with free monthly 60 minutes audio transcription and analysis	\$200 credit for free access of services within 30 days

CHAPTER 2

Standard pricing	Billed monthly depends on the pricing rates and discounts vary by region, begin with \$0.024 per minute	- \$0.024 and \$0.016 per minutes beyond 60 minutes for the recognition with and without data logging respectively.	<ul style="list-style-type: none"> - \$1 per usage hour - \$2.50 per audio hour for speech translation - \$5 per 1000 transaction for speaker verification - \$10 per 1000 transaction for speaker identification
Pricing factor	Seconds of audio transcribed per month, billed in one second increments, minimum request charge of 15 seconds	Amount of audio successfully processed by the service in every month, measured in second increments	Hours of audio sent to the service, billed in second increments

Comparing open source with paid speech to text models, open-source solutions outperform with the ability of customization and adaptation to specific use case, while paid services is typically used in general-purpose conditions. The current speech-to-text models can be time consuming due to their reliance on huge number of labelled datasets.

2.2 Google MediaPipe Framework

MediaPipe is an open-source framework developed by Google. It provides pre-trained models in detecting and tracking hand landmarks and poses. MediaPipe framework can be used to perform several tasks such as object detection, gesture recognition, image segmentation, hand landmark detection and etc [10]. MediaPipe performs gesture recognition by integrating different models together, which are palm recognizer that analyzes the hand image it captures, a hand landmark model that generate 3D key points of the hand and a gesture recognizer that classify the 3D hand key points into distinct gestures [11]. The hand landmark model infers 21 key points of the hand from a single frame through regressions, which is as shown in Figure 2.2.

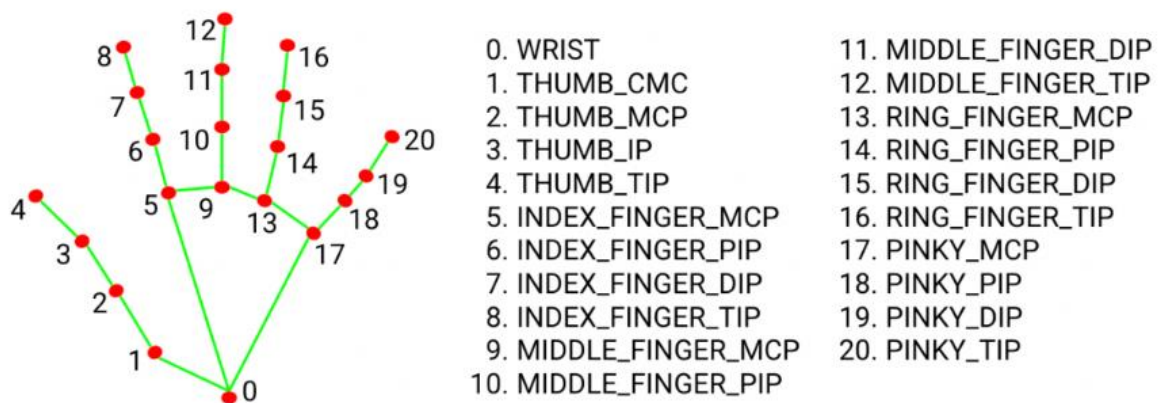


Figure 2.1: Key points that are inferred by MediaPipe hand landmark model

2.3 Gesture Recognition

Based on foreign psychological statistics, individuals rely on 55% of their body language and gestures, 38% on vocals and 7% on verbal language to communicate their thoughts and emotions [12]. In educational videos, educators frequently employ gestures in educational videos to underscore important points or draw attention to significant information visible on the screen [13]. The gestures provide signaling cues that can diminish the processing of irrelevant information during learning, given that the educational materials incorporate visual or auditory cues that emphasize important elements or structure of the content.

There are several hand gestures that is useful in the presentation or speech. The box gesture involves placing palms facing at mid chest level to frame an argument or idea, which is effective at the start of a presentation, during a summary or when introducing the lesson topic [14]. Another two gestures, known as holding a ball and pyramid hands, signals the authority when the speaker is speaking or listening, illustrating understanding and attention [14].

Peng et.al. [15] mentioned that beat gesture is always used by the lecturer during the lesson, as it establishes the speech rhythm and effectively capture learners' attention by emphasizing the verbal information. Additionally, the utilization of deictic gesture notably intensifies the learner's focus on the lecturer, directing their attention towards important visual cues through pointing. Consequently, this leads to enhanced cognitive processing of relevant educational information [13].

Point gesture is often used by the speaker to indicate specific topic on the board or slide, as well as to draw student's attention to a particular topic [16]. Conversely, the spread gesture is independent of lecture content, but it serves as pedagogic commentary, with the extent and duration of spread indicating the difficulty level of the material. In the study of Tian and Bourgeuet [17], they discovered that numbering, pointing, circle and ball gestures are related to teaching as each of them directs attention, explains steps and emphasizes key concepts respectively. The ball gestures were highlighted as a potential clue for video summarization, indicating that the instructor is emphasizing key points.

Nevertheless, interpreting gesture for video summarization can be subjective and context dependent. Different viewers may interpret same gesture differently, as seen in the work of [16], where the formal identifies pointing as the gesture for stressing key points and the latter mentioned that pointing means leading to right direction.



Figure 2.2: Six gestures useful for making emphasis. Top row: “Box”, “Pyramid” and “Beat” gesture. Bottom row: “Pointing”, “Circle” and “Balling” gesture.

2.4 Existing Summarization Tools

2.4.1 Text Summarization Tools

- **ChatGPT**

ChatGPT [18] is a versatile tool capable of summarizing text in various languages while preserving key concepts and vital details. It performs summary work by understanding and responding to user input, where the users need to input keywords in the prompts to instruct ChatGPT to summarize text. As shown in Figure 2.2, user can type “Please summarize the text ...”, followed by the text they want to summarize, optionally with their desired summary length or format. ChatGPT can generate summaries quickly upon receiving the input [19]. Nevertheless, it occasionally struggles to meet user demands and process queries efficiently due to the high volume of daily queries [19]. ChatGPT also face challenge in providing a broad overview of a topic, such as fields summarization. In such case, Wikipedia outperforms ChatGPT as ChatGPT sometimes presents incorrect information as truth [19].

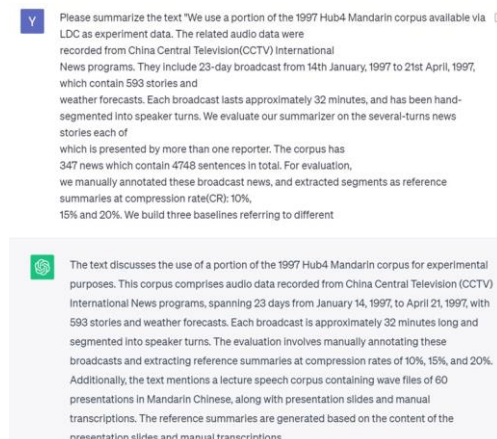


Figure 2.3: Text summarization with ChatGPT

- **SciSummary**

SciSummary [20] is designed specifically for summarizing scientific related contents from research papers or journal articles. It utilizes GPT-3.5 and GPT-4 models to perform summarization task. Users can upload articles, plain text or URL for summarization, additionally able to customize the summary by specifying its length (short, medium, long) and language (English, Chinese, French, German, etc.). SciSummary offers various summarization options, allowing users to choose between generating a synopsis, key points, simplified content, or identifying research opportunities. The summarized results are conveniently stored in a dashboard, and users receive email notifications with the summarized content once the process is complete. A notable feature of SciSummary is its support for bulk summarization, enabling users to upload and summarize multiple files simultaneously [20]. Figure 2.4 shows the summarization result of a research paper by using SciSummary.

The screenshot displays the SciSummary web interface. At the top, the document title is "Human emotions track changes in the acoustic environment.pdf" with a PDF icon and a timestamp "Requested On 2023-09-02 01:02". Below the title, there are navigation tabs: "Article", "Summaries", "Chat", "Metadata & References", and "Notes". The "Article" tab is active, showing a preview of the research paper's first page. The "Summaries" tab is also visible, showing a generated synopsis. The synopsis text is: "This scientific article explores the idea that the emotional qualities of speech and music can also be found in environmental sounds. The authors propose that humans have evolved to be attuned to changes in the frequency spectrum, intensity, and rate of sounds, as these attributes provide important information about the size, proximity, and speed of". The interface includes a "COPY" button and a "Delete" link.

Figure 2.4: Research paper summarization with SciSummary

- **Artifact**

Artifact [21] is a news aggregator that leverages artificial intelligence (AI) to provide personalized recommendations regarding subjects, news sources, and authors based on the reader's preferences. Recently, it introduced a new summarization tool that utilize artificial intelligence to generate summaries, aiding readers to grasp high-level points of an article before delving into the full text. What sets this summarization style apart is its adaptability, allowing users to create simple "Explain like I'm five" summaries, emoji-based summaries, poetic summaries, or summaries tailored to the Gen Z style [22]. However, the Artifact Team mentioned that the AI that perform summarization task might make mistakes, so readers should verify that the summary aligns with the full text from time to time [22]. Figure 2.5 shows the summarization of an article by using Artifact.



Figure 2.5: Article summarization with Artifact

2.4.2 Video Summarization Tools

- **YoutubeDigest**

YoutubeDigest [23] is a browser extension that utilize ChatGPT to summarize YouTube videos, requiring users to have a ChatGPT account as a prerequisite. YouTube Digest offers summaries in various languages, including English, Chinese, German, French, Italian, etc. The summary can be exported as PDF, DOCX or plain text for future reference [23]. Besides that, it provides flexibility in summary format, allowing users to choose between a single paragraph, article, or a set of bullet points, with the option to specify a maximum number of bullet points if desired. However, YoutubeDigest is specifically designed for summarizing YouTube video, meaning that it does not support summarization of videos from other sources. Figure 2.6 shows the YouTube video summarization using YouTube Digest.

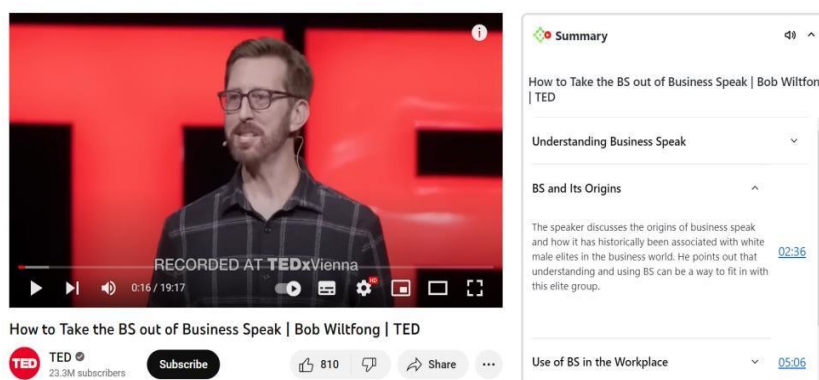


Figure 2.6: YouTube video summarization with YouTube Digest

- **NoteGPT**

NoteGPT [24] is another browser extension that use ChatGPT to summarize and take the notes on YouTube videos. It provides two methods to summarize the YouTube videos, one via an OpenAI API Key, where the API key establishes a connection between NoteGPT and ChatGPT, and the other redirects users to the ChatGPT website, where the transcript from YouTube videos is automatically pasted into the GPT prompt to generate a summary. Users can regenerate summary until they achieved their expected result. However, NoteGPT users need to have a ChatGPT account to use this extension, which is similar to YoutubeDigest [25]. It can be used on various devices but does not support the collaboration with other users [24]. Figure 2.7 shows that NoteGPT is summarizing a TED Talk video, and the summary result will be displayed on ChatGPT, as shown in Figure 2.8.

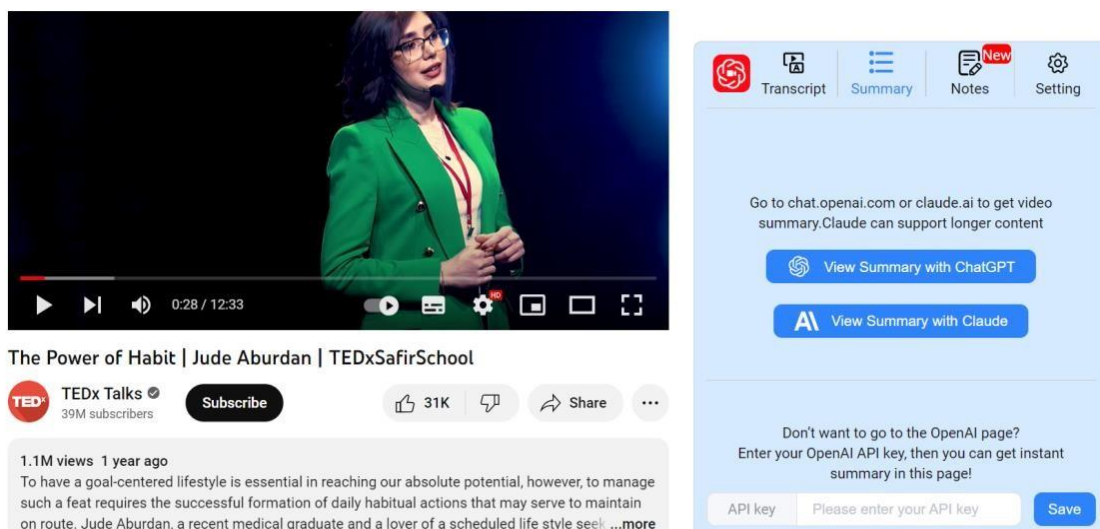


Figure 2.7: TED Talk video summarization with NoteGPT

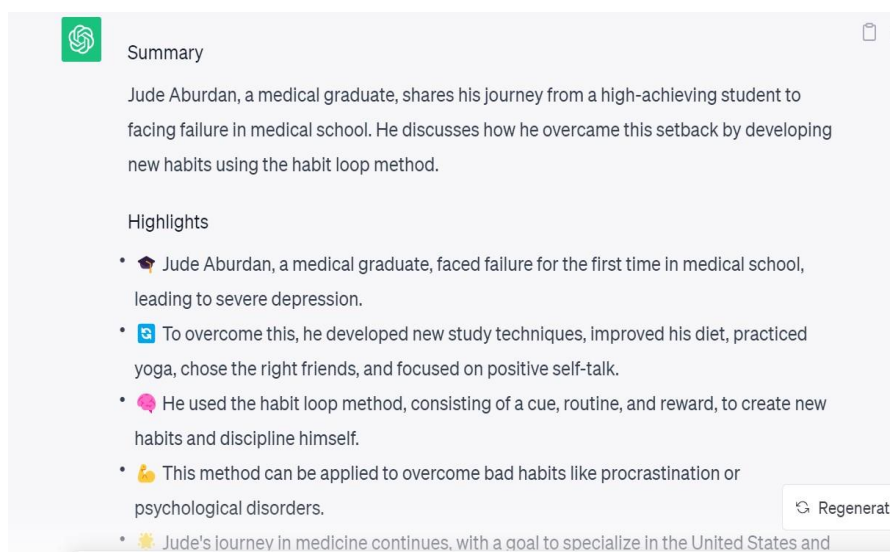


Figure 2.8: Display of summary result in ChatGPT

- **Mindgrasp**

Mingrasp [26] produces text- and audio-based summaries, allows user to download and listen to the generated summaries using the listen feature. Users simply need to upload video files or paste YouTube links, and Mindgrasp will analyse and generate the summary text, as shown in Figure 2.9. Users have the option to read the video transcript and view the notes created by Mindgrasp side by side. Moreover, it is versatile in that it accepts various video types. However, it has language limitations, as it supports only four languages: English, German, Spanish, and French.

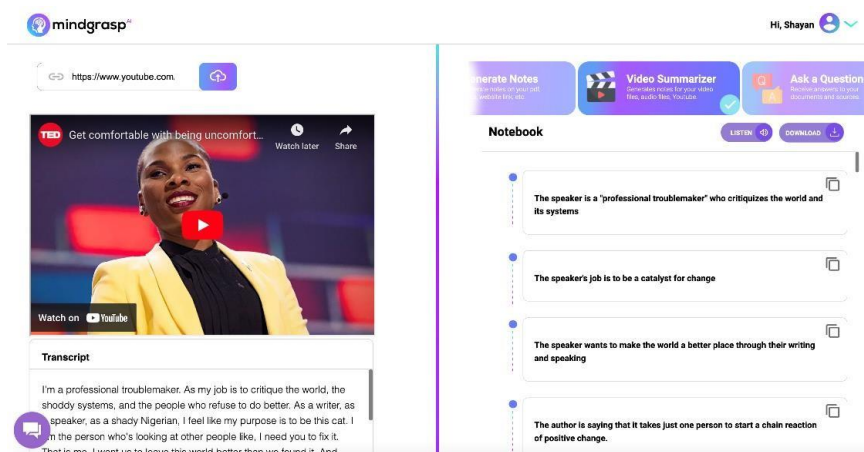


Figure 2.9: YouTube video summarization with Mindgrasp

2.4.3 Audio Summarization Tools

- **PodPulse**

PodPulse [27] is an AI-powered tool designed to condense podcasts into concise and engaging summaries. Users can grasp essential points and valuable knowledge without having to listen to hours of audio content. Users can select podcasts from the app's database or request summaries on-demand by pasting podcast URLs from platforms like Apple Podcasts, Google Podcasts, or YouTube Podcasts. The summaries and takeaways from the podcasts acquired from database are reviewed by human, ensuring the accuracy of the AI-generated results. However, it only supports English podcasts and cannot summarize content in other languages. Figure 2.10 shows the summarization of the PodPulse podcast using PodPulse.

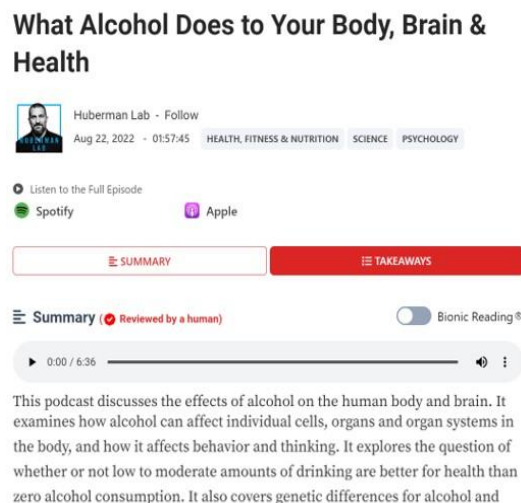


Figure 2.10: Podcast summarization with PodPulse

Chapter 3 System Methodology/Approach

3.1 System Architecture Diagram

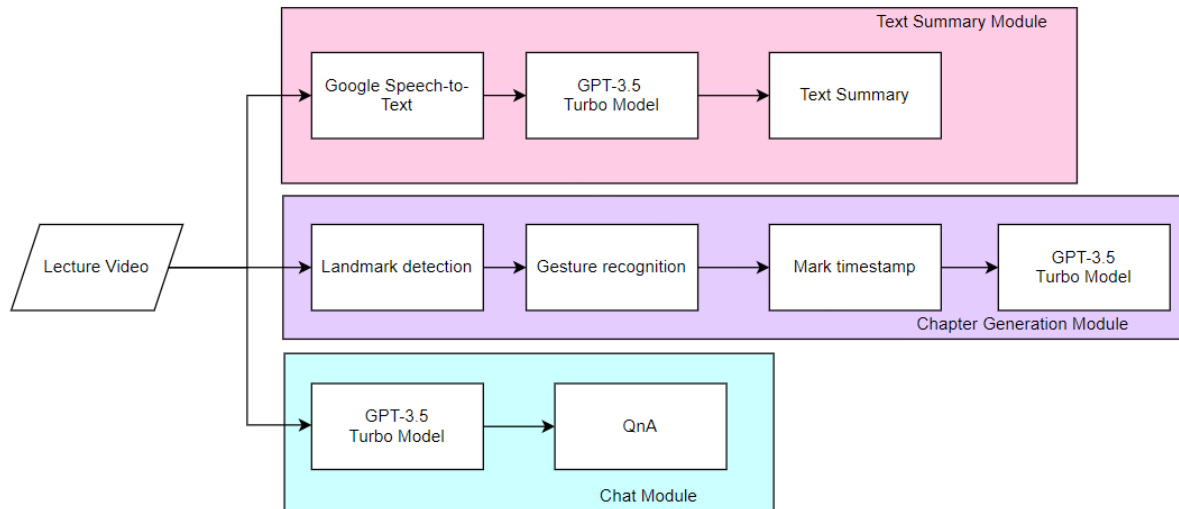


Figure 3.1: System Architecture Diagram

There are three main modules in this Lecture Digest application, each of them functions differently. Text summary module generates the video summary in text. The audio track is extracted from the video and passed to Google Speech-to-Text to produce a transcription. This transcription is subsequently passed to the GPT-3.5 turbo model to generate a text summary. Chapter generation module generates video chapters through gesture recognition by using MediaPipe Gesture Recognizer. The Gesture Recognizer first detects the hand in the video frame and then detects the landmarks. Upon detecting the landmark, it performs gesture classification, marking the corresponding timestamp of the landmark into the video for users to easily skip ahead and rewind to these important segments. The title for important chapters through the GPT-3.5 Turbo model and align with the extracted timestamps. For a chat module which acts as a virtual tutor, the transcription along with the user prompt is passed to the GPT-3.5 model to generate the answer. The chatbot produces different types of assistance, such as generating frequently asked questions, quizzes, flashcards, and QnA session.

3.2 Use Case Diagram

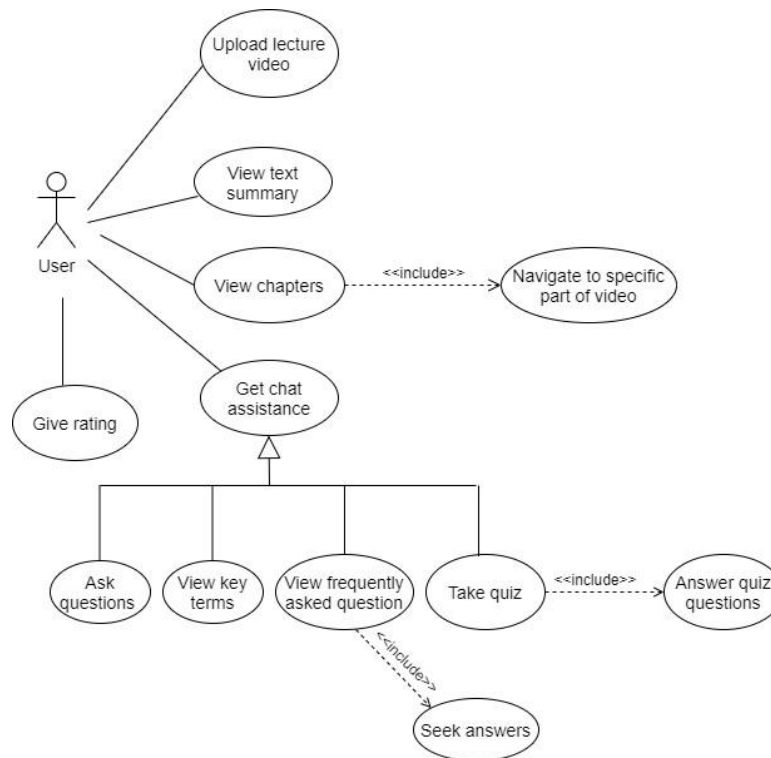


Figure 3.2: Use Case Diagram

Figure 3.2 illustrates the functionalities offered by the Lecture Digest application to users. Users can upload lecture videos to the app and choose to view the video summary in text or video. They can access to specific video chapters that represent important section of the video. Additionally, users can interact with the application by asking questions about the video, viewing key terms, accessing frequently asked questions relevant to exams, and taking quizzes to test their understanding. Users can also provide feedback regarding the quality and usefulness of the application.

3.3 MediaPipe Gesture Recognition Model Architecture

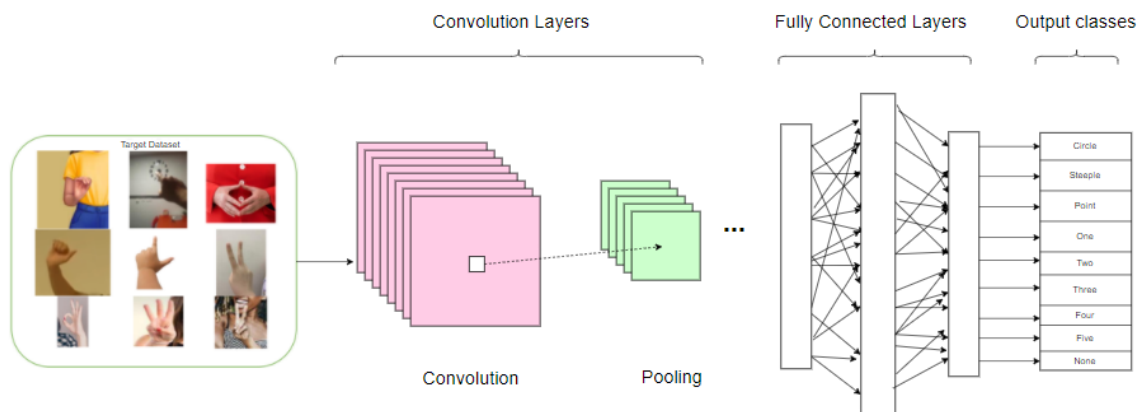


Figure 3.2: MediaPipe Gesture Recognition Model Architecture

MediaPipe Model Maker is a tool to customize pre-existing machine learning models. It uses transfer learning techniques to retrain the existing models with new dataset, thus taking less time to train new model. MediaPipe Model Maker retrain the gesture recognition model by rebuilding the classification layers based on new dataset. The network they used to perform gesture recognition is CNN architecture that consists of five convolutional layers and one fully connected layer. The first convolutional layer consists of 32 Gaussian filters with the dimension of 3x3, followed by 64, 128, 256 and 512 of Gaussian filters with the same dimension (3x3). The fully connected layer in the final stage is a feed forward neural network that performs classifications on the gestures. In this project, new dataset is fed into pretrained gesture recognition model and there will be 9 output classes of gestures (circle, steeple, point, one, two, three, four, five, none) to be recognized. The "none" label which consists of the gestures not classified under any other specific category is required by MediaPipe Model Maker to retrain the gesture recognition model.

3.4 MediaPipe Gesture Recognizer Pipeline

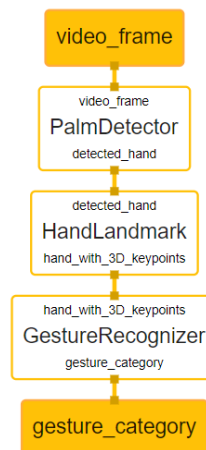


Figure 3.3: MediaPipe Gesture Recognizer Pipeline

MediaPipe Gesture Recognizer API is used to recognize the gesture in the video frame based on the gesture recognition model. Figure 3.3 shows pipeline of Gesture Recognizer. The palm detection model identifies the area containing hands within the entire input image and then crop the image. Subsequently, the hand landmarks detection model detects the hand landmarks on the cropped image. The gesture recognizer then classifies the gesture based on the hand landmarks. During video frame processing, the Gesture Recognizer localizes hand regions in subsequent frames using the bounding box established by the detected hand landmarks in the current frame. Therefore, the time taken to trigger the palm detection model is reduced. The output comprising four key components: hand landmarks in image and world coordinates, handedness (left or right hand), and the gesture category.

3.5 Generative Pre-trained Transformer (GPT) Architecture

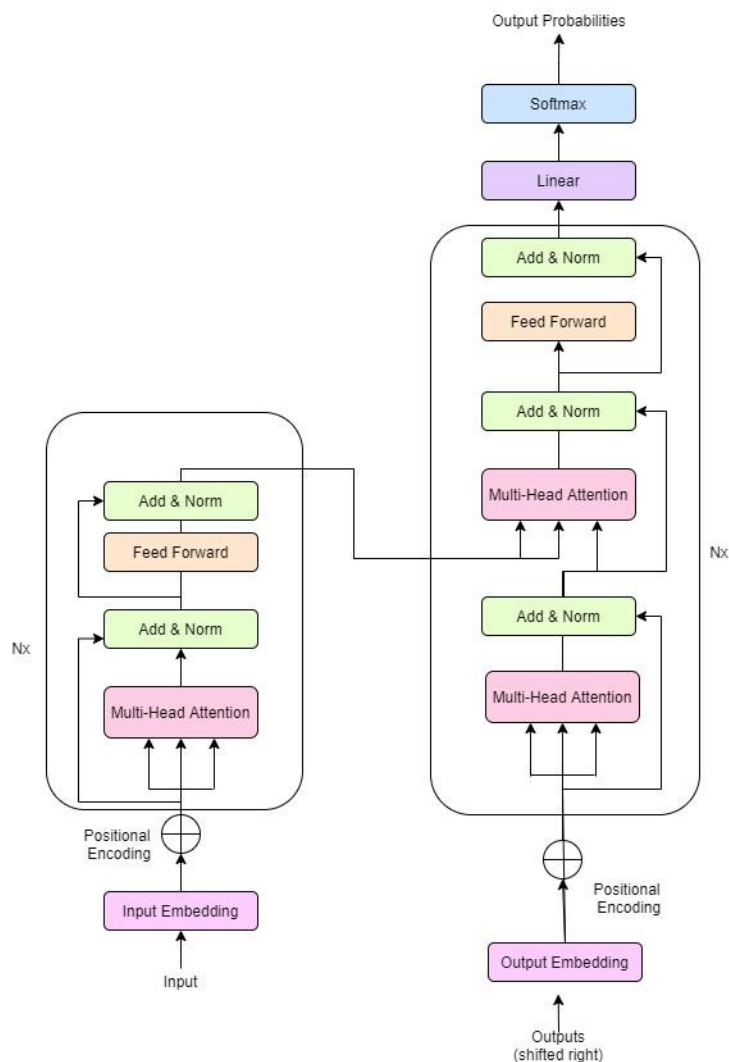


Figure 3.4: GPT Architecture

Figure 3.4 shows the GPT-3.5 Turbo model architecture. GPT-3.5 Turbo model is a multimodal Large Language Model (LLM). It consists of multiple layers of neural networks (recurrent layers, feedforward layers, embedding layers, and attention layers) that work together to process the input text and generate output predictions [28]. The input of GPT is a sequence of N words (tokens), while output is a sequence of prediction for the last word of the input sequence.

The flow of GPT-3.5 is as below:

1. The input text is tokenized into word using byte-level Byte Pair Encoding (BPE) tokenization. Each token is transformed into a high dimensional vector representation through embedding. The embedding process encapsulate the meaning and structure of the input text.

CHAPTER 3

2. As the transformers by nature do not know the orders of the tokens, the positional encoding is added to the input embeddings to provide information about the token's positions.

3. The encoder analyses the input text and create a series of Hidden states to preserve the context and meaning of the text data. The encoder applies self-attention that computes attention scores to weight the importance of different tokens. Multiple head attention which consists of more than one self-attention that are performed simultaneously is used by GPT model to capture different types of relationships. Afterwards, the feed forward neural network is applied to each token separately to capture complex interactions between tokens.

4. The input block is added to its output and the result is normalized after the multi-attention and feed forward process. The decoder produces sequential outputs by focusing on the words generated earlier in the sequence. The outputs are change to numbers through positional encoding.

5. The linear layer is then transforming the output embeddings to original input space. After that the SoftMax function generates probability distribution for each output token in the vocabulary and finally produce output tokens with probabilities.

Chapter 4 System Design

4.1 System Block Diagram

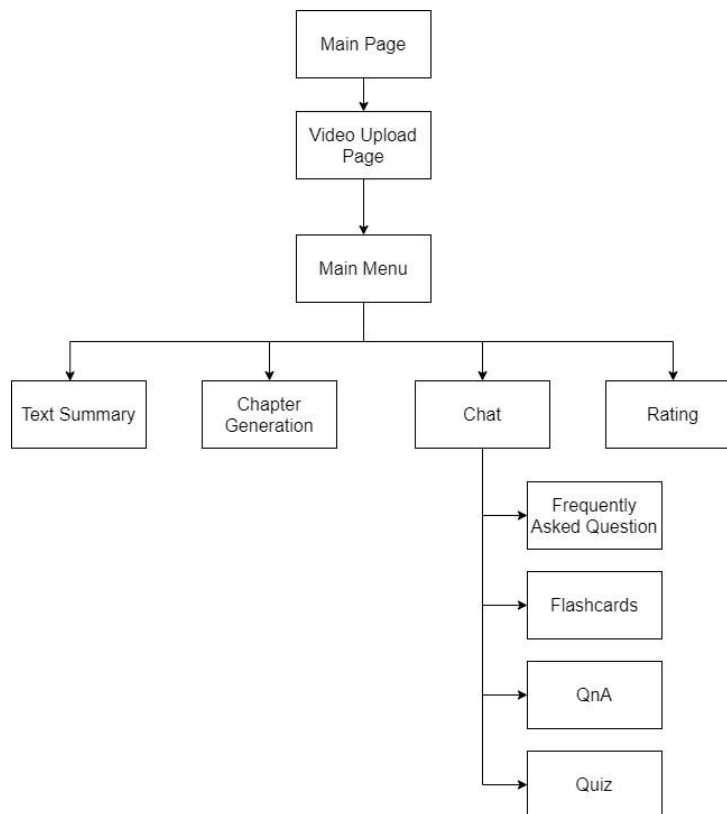


Figure 4.1: System Block Diagram

Lecture Digest consists of three main modules: a text summary module, a chapter generation module, and a chat module, along with an additional rating feature for users to provide feedback.

Text summary module offers lecture summaries in text format. In contrast, chapter generation module provides video chapters that highlights the key moments of the lecture. The chat module comprises 4 submodules: a frequently asked questions module, flashcards module, QnA module, and quiz module. These submodules allow users to access frequently asked exam questions, request important terminologies, inquire about the video, or engage in quizzes respectively.

4.2 System Components Specifications

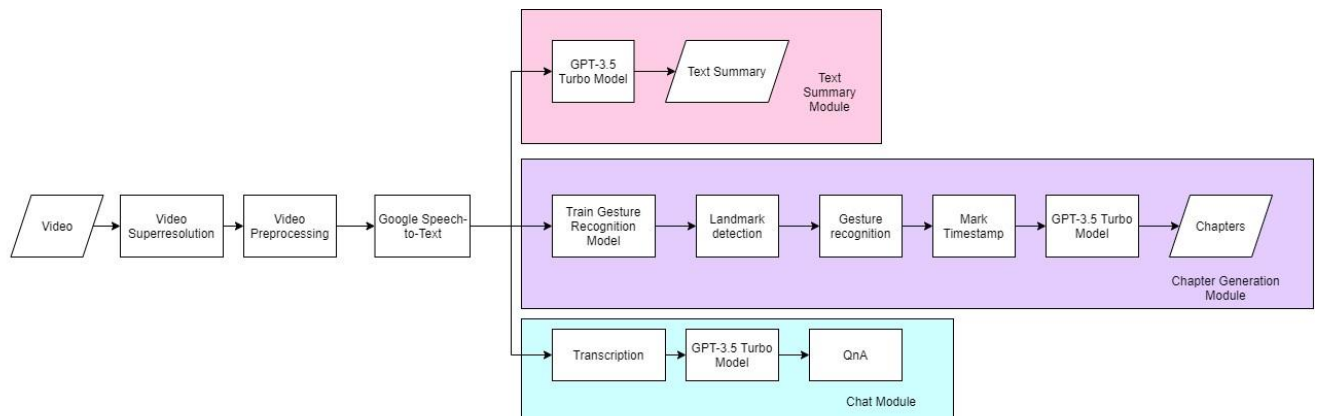


Figure 4.2: Lecture Digest Flow

The application has three main modules, text summary module, chapter generation module and chat module. The flow of each module is discussed in the following section.

4.2.1 Video Super-resolution

As some students might sit very far behind the lecture class, so the video that filmed by them might not be clear. However, the resolution of the video is important to identify the gesture of lecturer in chapter generation module. This stage is optional, as it is exclusively applied to the blurry videos. After receiving the video, the system will check the resolution of the video. If the resolution is 240p and below, TecoGAN API is invoked to upscale the video. The input video will be upscaled by 4x, so the returned video will be clearer. If the video is beyond the resolution threshold, TecoGAN will not be called. After enhancing the video resolution with TecoGAN, the edges of the objects within the video become sharper.



Figure 4.3: Original Video



Figure 4.4: Enhanced Video

4.2.2 Video Preprocessing

This step is used to generate transcription of the video which will be used for different purposes in different modules.

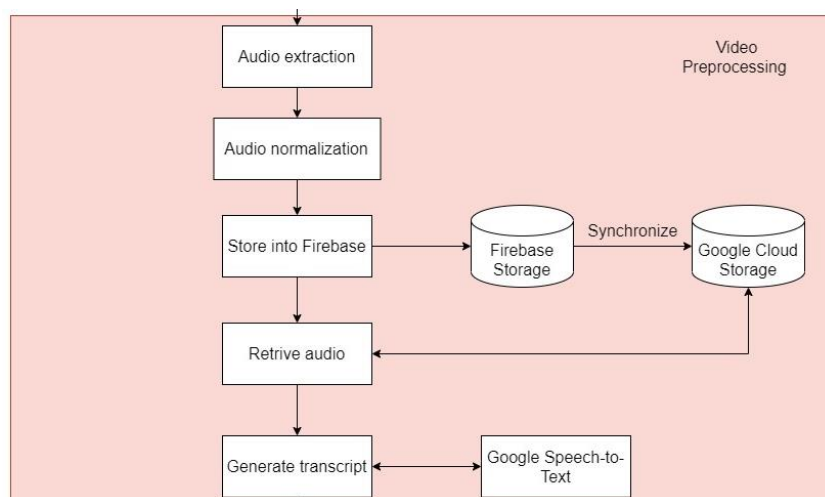


Figure 4.5: Video Preprocessing Flow

4.2.2.1 Audio Extraction

Google Speech-to-Text API accepts audio as input to transcribe the spoken words. Hence, before obtaining the transcription of the lecture video, the audio track needs to be extracted to isolate the audio and visual content. The audio is extracted by using MediaExtractor class that provided by Android Studio, and the output is written into the destination file using MediaMuxer.

4.2.2.2 Audio Normalization

The lecturer's sound that is not consistent throughout the video, resulting in sections that are either too loud or too quiet, which can affect the accuracy of speech recognition for transcription. Therefore, the audio is normalized using AudioTool library so that the volume level is consistent throughout the entire audio file.

4.2.2.3 Store Audio File

The normalized audio file is then uploaded to the Cloud Storage for Firebase. It is synchronized to the Google Cloud Storage bucket due to the nature that Firebase integrates with Google Cloud Storage.

4.2.2.4 Generate Transcript

The audio file is retrieved from Google Cloud Storage and fed into Google Speech-to-Text API to produce the transcription output. The configuration for generating the transcription is shown in Figure 4.6. Video model is set to tell the Google STT that the audio track is came from a video source. The transcription comes along with the timestamp for each spoken word. For the timestamp be included in the transcription, automatic punctuation is applied, and the timestamp for each spoken word is detected.

```
RecognitionConfig config =
    RecognitionConfig.newBuilder()
        .setEncoding(RecognitionConfig.AudioEncoding.MP3)
        .setSampleRateHertz(44100)
        .setLanguageCode("en-US")
        .setEnableWordTimeOffsets(true)
        .setEnableAutomaticPunctuation(true)
        .setUseEnhanced(true)
        .setModel("video")
        .build();
RecognitionAudio audio =
    RecognitionAudio.newBuilder().setUri(gcsUri).build();
```

Figure 4.6: Configuration to generate transcription

The transcription is then restructured such that the timestamps are only included in the first word for each sentence. The timestamps returned is in milliseconds, and it is converted into the format of minute and seconds (mm:ss).

```

BufferedWriter writer = new BufferedWriter(new FileWriter(transcriptFile));
StringBuilder transcriptLine = new StringBuilder("Transcription: ");
for (SpeechRecognitionResult result : results) {
    boolean start=true;

    SpeechRecognitionAlternative alternative =
result.getAlternativesList().get(0);

    for (WordInfo wordInfo : alternative.getWordsList()) {
        if (start){
            transcriptLine.append(String.format( // only put timestamp for
the first word of each sentence
                "(%02d:%02d) %s ",
                wordInfo.getStartTime().getSeconds()/60, //convert to
minutes
                wordInfo.getStartTime().getSeconds()%60, //convert to
seconds
                wordInfo.getWord()));
            start=false;
        }
        else{
            transcriptLine.append(String.format(
                "%s ",
                wordInfo.getWord()));
        }
    }
}
}

```

Figure 4.7: Restructure on the transcription output

4.2.3 Text Summary Module

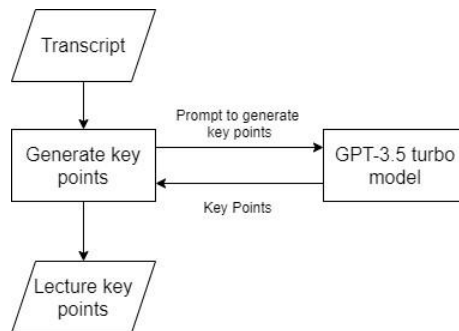


Figure 4.8: Flow of Text Summary Module

This module generates text summary from the video. The video transcription is passed to GPT-3.5 Turbo model to perform summarization. In the prompt request, the GPT model is specifically instructed to condense the transcription into key points as the summary of the lecture video. The resulting key points is presented in bullet points.

4.2.4 Chapter Generation Module

4.2.4.1 Train Gesture Recognition Model

Before identifying gestures from the video frame, the gesture recognition model is trained. The gesture recognition model is trained using the MediaPipe Model Maker. MediaPipe Model Maker uses transfer learning to retrain the existing model with the new data.

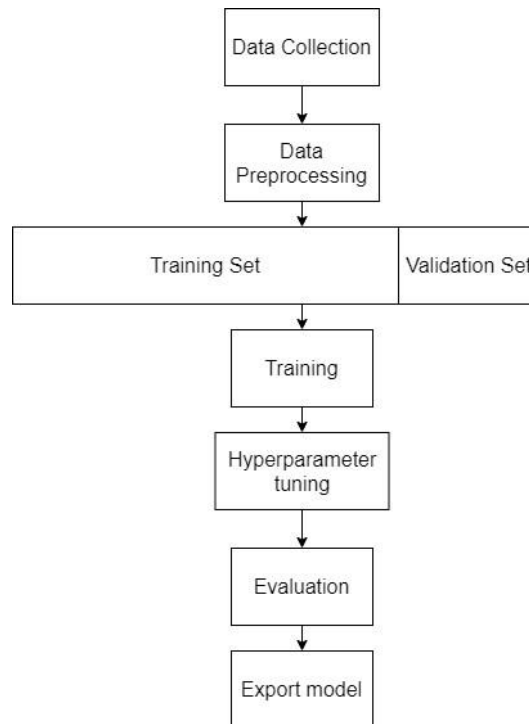


Figure 4.9: Training of Gesture Recognition Model








1. Data Preparation

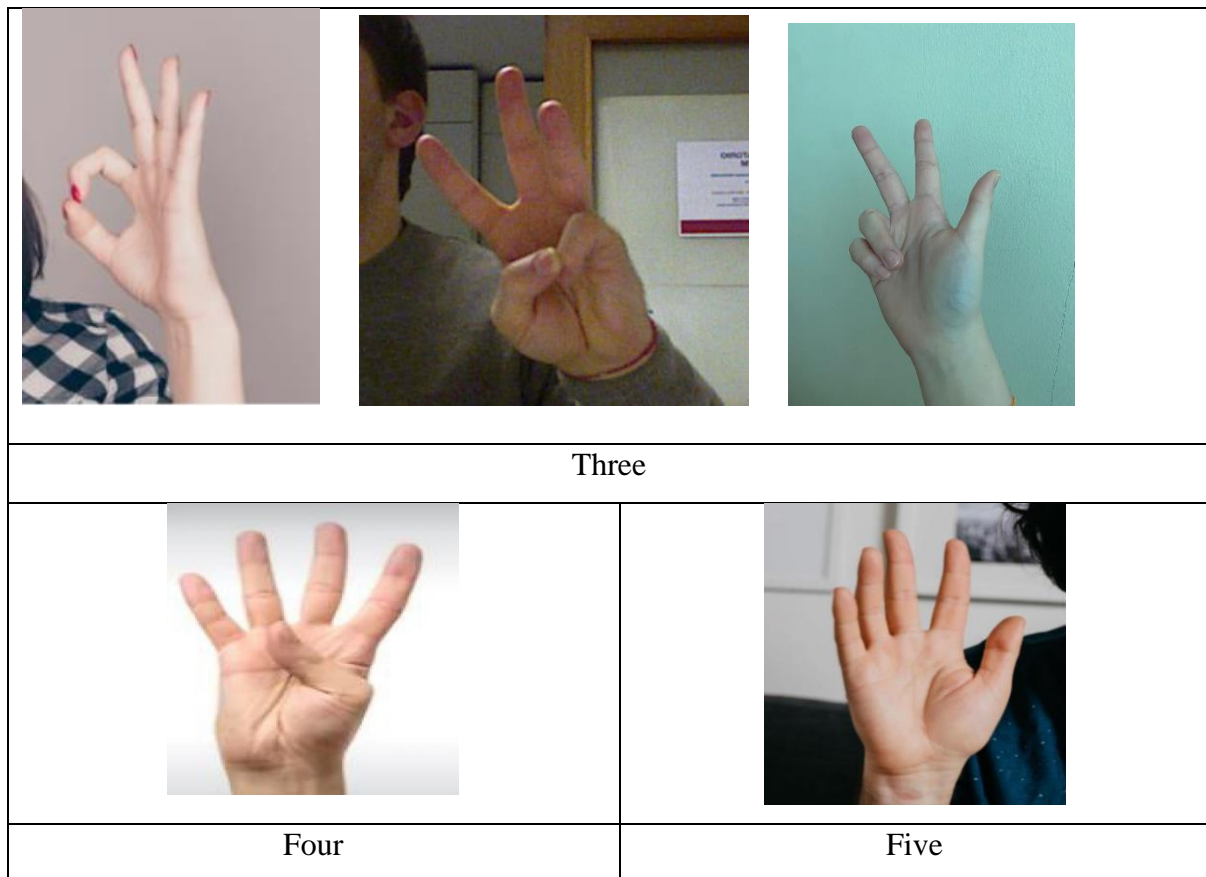
The gesture images are collected and categorized into different folders. There are 9 folders in the dataset, the gestures to be trained are circle, listing (one, two, three, four, five), steeple, point. These gestures are the common gestures that used by lecturers when they are emphasizing key points. Besides that, MediaPipe Model Maker requires a 'none' folder that stores gestures that should not be classified.



Figure 4.10: Prepare Dataset

Table 4.1: Gesture Images

Gestures			
			
Circle	Point	Steeple	
			
One		Two	



2. Load Dataset

The platform to train the gesture model is Google Colab. The dataset is loaded to Google Colab and located at `dataset_path` directory. The hand detection model, MediaPipe Hands is used to discard the images without detected hands. The resulting dataset includes the positions of hand landmarks extracted from each image.

Subsequently, the dataset is divided into 3 parts, 80% is used for training, 10% is used for validation and another 10% is used for testing.

```
data = gesture_recognizer.Dataset.from_folder(
    dirname=dataset_path,
    hparams=gesture_recognizer.HandDataPreprocessingParams()
)
train_data, rest_data = data.split(0.8)
validation_data, test_data = rest_data.split(0.5)
```

Figure 4.11: Load Dataset

3. Train Model

The training and validation data are passed to the pretrained MediaPipe gesture recognizer model. The default parameter (dropout rate = 0.05, learning rate = 0.001, batch size = 2, epoch = 10, learning rate decay = 0.99, gamma = 0.2, shuffle = false) is used to train the model. The accuracy achieved is 0.7981.

```
hparams = gesture_recognizer.HParams(export_dir="exported_model")
options = gesture_recognizer.GestureRecognizerOptions(hparams=hparams)
model = gesture_recognizer.GestureRecognizer.create(
    train_data=train_data,
    validation_data=validation_data,
    options=options
)
```

```
acc = model.evaluate(test_data)
print(f"Test accuracy: {acc}.4f")
```

Test accuracy: 0.7981

Figure 4.12: Train model

4. Hyperparameter Tuning

Subsequently, different combinations of hyperparameter are tried to find the model that can achieved the best accuracy. The best hyperparameter (dropout rate = 0.2, learning rate = 0.01, batch size = 2, epoch = 10, learning rate decay = 0.99, gamma = 0.2, shuffle = true) can achieve the highest accuracy of 0.8372.

```
hparams = gesture_recognizer.HParams(shuffle = True, learning_rate=0.01, export_dir="exported_model_2")
model_options = gesture_recognizer.ModelOptions(dropout_rate=0.2)
options = gesture_recognizer.GestureRecognizerOptions(model_options=model_options, hparams=hparams)
model_2 = gesture_recognizer.GestureRecognizer.create(
    train_data=train_data,
    validation_data=validation_data,
    options=options
)
```

```
acc = model_2.evaluate(test_data)
print(f"Test accuracy: {acc}.4f")
```

Test accuracy: 0.8372

Figure 4.13: Hyperparameter Tuning

5. Testing model

Before export the final model, it is evaluated. The lecture video is taken as the input to test the model and bounding box is drawn on the video frame if gesture is detected. The gesture category is labelled on top of the bounding box.

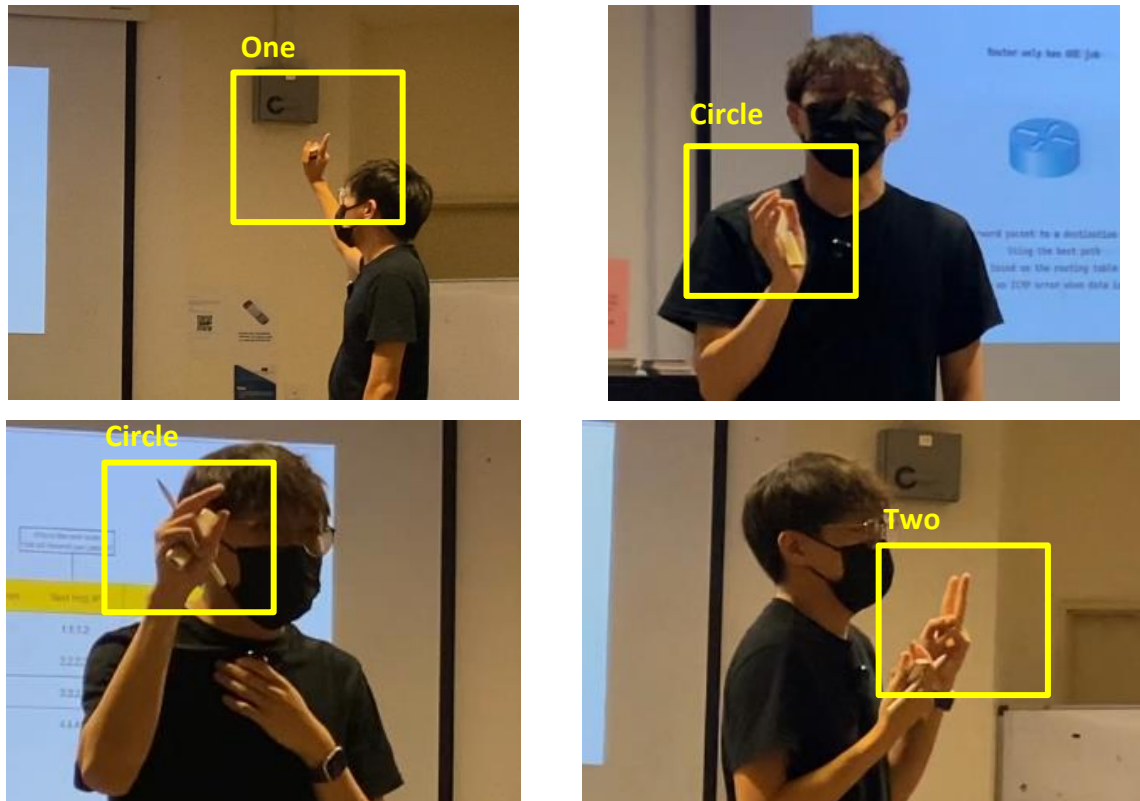


Figure 4.14: Detected gestures in the video

6. Export model

Finally, the fine-tuned model is exported into the local device, which will be loaded to the Android Studio to perform gesture recognition.

```
model_2.export_model()
files.download('exported_model2/gesture_recognizer.task')
```

Figure 4.15: Export Model

4.2.4.2 Hand Gesture Recognition

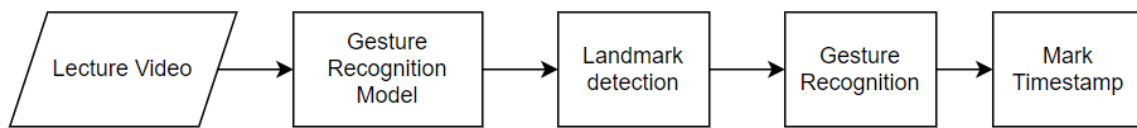


Figure 4.16: Hand Gesture Recognition Flow

The final gesture recognition model is deployed into Android Studio. The gesture recognition is performed using MediaPipe Gesture Recognizer API. The API is invoked to recognize the gesture in the video frame by frame based on the trained gesture model. MediaPipe Gesture Recognizer begins by detecting and localizing the hand landmark, subsequently perform gesture recognition based on the landmark. After the gesture is recognized, the corresponding timestamps is extracted and marked in the video.

4.2.4.3 Integrate the Gesture Recognition Process to Generate Video Chapters

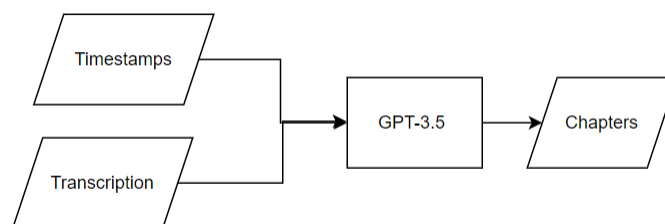


Figure 4.17 Generate chapters

The extracted timestamps, along with the video transcription that has been generated beforehand is passed to the GPT-3.5 turbo model. The GPT model analyses the content surrounding the timestamps and generate chapters that encapsulate the context before and after each significant moment. Finally, the generated chapters are returned and displayed on the app. Users can click on the chapters and navigated to the important moment of the video.

4.2.5 Chat Module

The chat module consists of 4 submodules, which are Frequently Asked Question (FAQ) module, QnA module, flashcard module and quiz module.

4.2.5.1 Frequently Asked Questions (FAQ) Module

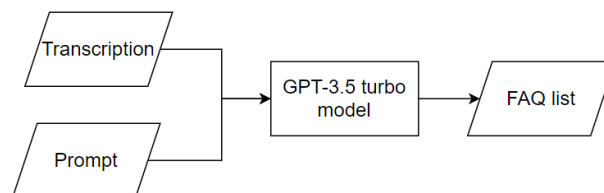


Figure 4.18: Flow of FAQ Module

This module generates list of questions that is commonly asked in the examination. The GPT-3.5 turbo model is instructed to identify significant points within the transcript and formulate relevant questions accordingly. Additionally, it is instructed to respond to user queries related to the generated questions. Upon returning the answers, it generates other new questions to the users.

4.2.5.2 Question and Answer (QnA) Module

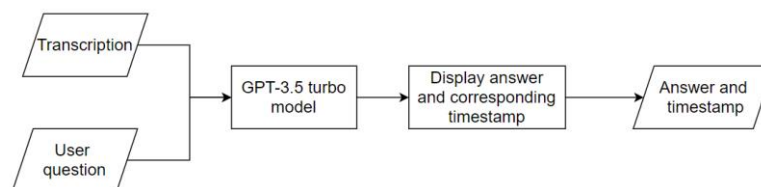


Figure 4.19: Flow of QnA Module

This module serves to address the user queries regarding the lesson content. The user's question and transcription are passed to GPT-3.5 turbo model. The model is instructed to analyze and find the answer from the transcript. Upon processing, the model generates the answer along with timestamps indicating where in the transcript the answer can be found. The timestamp enables users to navigate back to the video content, facilitating fast retrieval of the relevant information.

4.2.5.3 Flashcard Module

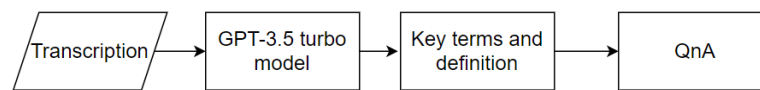


Figure 4.20: Flow of Flashcard Module

This module generates a list of key terms from the lesson by passing the transcription to the GPT-3.5 turbo model. The model is instructed to identify significant terminologies within the lesson content. Subsequently, it provides users with these key terms along with their respective definitions. If users have inquiries regarding any terminologies, their questions are directed to the model, which then returns the corresponding answers back to them.

4.2.5.4 Quiz Module

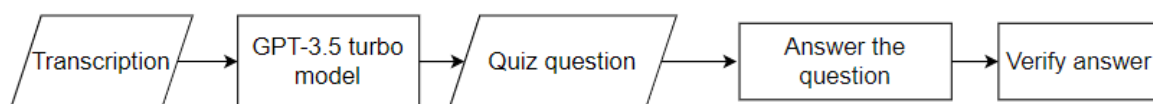


Figure 4.21: Flow of Quiz Module

This module generates quiz questions to assess users' comprehension of the lesson content. It operates by passing the transcript to the GPT-3.5 turbo model, which then formulates questions based on the transcript's contents. After users provide their answers, the model verifies them and returns the quiz results along with a new quiz question to the user.

4.2.6 Rating

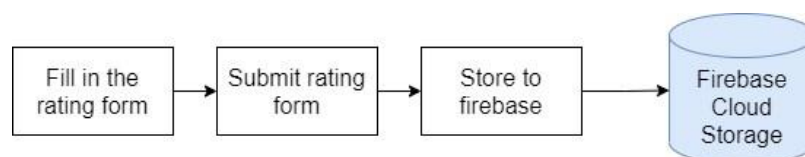


Figure 4.22: Rating Feature

Rating feature is an additional feature in this application. User can give rating to the application. Upon submitting the rating, the rating will store into the firebase.

Chapter 5 System Implementation

5.1 Hardware Setup

This project has used a laptop for the development of Lecture Digest application.

Table 5.1: Specifications of laptop

Description	Specifications
Model	Dell Inspiron 5491 2n1
Processor	Intel(R) Core™ i3-10110U CPU
Operating System	Windows 11
Graphic	Intel(R) UHD Graphics
Memory	12GB RAM
Storage	222GB

5.2 Software Setup

5.2.1 Platform

1. Android Studio

Android Studio is an integrated development environment (IDE) designed specifically to develop Android mobile applications. It supports two types of programming languages: Kotlin and Java. In this project, Java is chosen to develop the mobile application's user interface (UI) and for seamlessly integrating various APIs and models. Java is preferred due to the ease of learning and its extensive support of libraries, frameworks and tools.

2. Google Colab

Google Colab is used to train the gesture recognition model using Mediapipe Model Maker. The trained model is saved into the local device which is then integrated into Android Studio to perform gesture recognition.

5.2.2 Storage

1. Cloud Storage for Firebase

Cloud Storage for Firebase is an infrastructure built upon Google Cloud, facilitating developers to store user-generated content. Firebase is part of Google Cloud Platform, so the Firebase project also exist in Google Cloud Platform. In this project, once the user uploads the lecture video, the normalized audio file is stored into Firebase Cloud Storage, which is used to generate transcription later. The user rating is also saved into Firebase Cloud Storage.

2. Google Cloud Storage

The audio file stored in Firebase Cloud Storage is synchronized to bucket in Google Cloud Storage. During Speech-to-Text process, the Google Speech-to-Text retrieves the audio file stored in Google Cloud Storage.

5.2.2 API

1. Google Speech-to-Text

Google Speech-to-Text API is used to transcribe audio into text. It retrieves the audio file from Google Cloud Storage bucket and executes speech recognition process. It generates and return transcription based on the audio content. The transcription is structured such that every beginning of the sentence consists of its corresponding timestamp.

2. GPT-3.5 Turbo Model

OpenAI GPT-3.5 Turbo model is a Large Language Model that return a text based on the user prompt. This project employs GPT-3.5 Turbo model to generate textual summaries and video chapters. The chatbot is also built based on this model.

3. TecoGAN

In the preprocessing phase, a pre-trained super-resolution API called TecoGAN is used to enhance the video resolution. The TecoGAN API is called to enhance the resolution of blur video.

5.2.3 Framework

1. MediaPipe

MediaPipe is an open-source framework developed by Google for building multimodal machine learning pipelines such as hand tracking, gesture recognition, pose detection and face detection.

In this project, MediaPipe Gesture Recognizer API is used to perform gesture recognition, such that it analyzes the input video frame by frame and identify the frames containing specific trained gestures based on trained model. In order to use MediaPipe Gesture Recognizer, dependencies are added into app-level gradle file, as shown in Figure 5.1.

On the other hand, MediaPipe Model Maker is a tool for fast building and training of a new machine learning model through transfer learning. It operates across a range of model types, such as object detection, gesture recognition, and classification for images, text, or audio data. In this project, it is used to customize the gesture recognition model.

```
//mediapipe
implementation ("com.google.mediapipe:tasks-vision:latest.release")

implementation("com.google.mediapipe:tasks-vision:latest.release") {
    exclude(group = "com.google.protobuf", module = "protobuf-javalite")
}
```

Figure 5.1: Adding MediaPipe dependencies into app-level gradle file

5.2.4 Programming Language

1. Java

Java is used to develop the mobile application in Android Studio.

2. Python

Python is used to train the gesture recognition model in Google Colab.

5.3 Setting and Configurations

Prior to the development of lecture digest application, some software needs to be set up.

1. Android Studio:

IDE Version: Android Studio Giraffe | 2022.3.1 Patch 3

Android Virtual Device version: Pixel 3a XL API 29

Android Studio is the IDE to create user interface, integrate with API and test and debug the application. A new project should be created to write the code and a virtual device needs to setup to simulate the working flow of the application.

2. Firebase

Firebase is a mobile development platform provided by Google. To store the files into Firebase Cloud Storage, a Firebase account and project are needed. Firstly, a Firebase project is created. The lecture digest app is connected to the Firebase project. A bucket needs to be created to store the normalized audio files. In this project, the Firebase project named Lecture Digest and the bucket name lecture-digest-97df2.appspot.com. A folder named lecture/ is created to store normalized audio files, while lecture_digest_rating/ stores the user rating. Subsequently, the cloud storage dependencies are added into the Android app by configuring the dependencies in the app level gradle file.

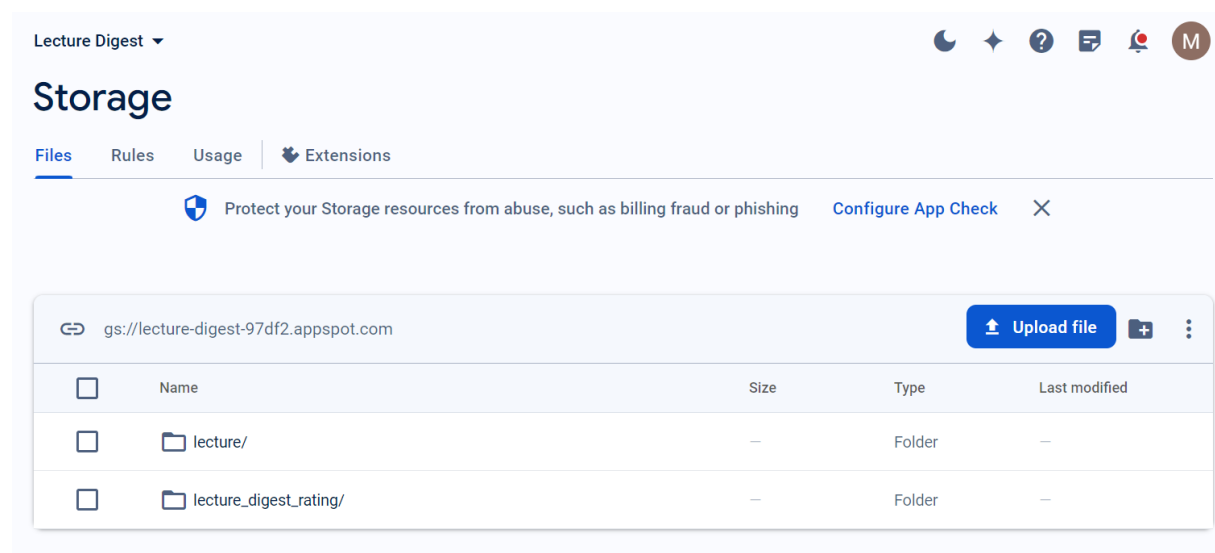


Figure 5.2: Creation of Firebase Project

```
implementation("com.google.firebase:firebase-storage:20.3.0")
implementation(platform("com.google.firebase:firebase-bom:32.6.0"))
implementation("com.google.firebase:firebase-analytics")
```

Figure 5.3: Adding Firebase dependencies in app-level gradle file

3. Google Cloud Platform

Google Cloud Platform is a cloud computing platform provided by Google. In this project, Google Cloud Storage and Google Speech-to-Text API are used for file storage and speech recognition respectively. To assess these two services, a Google Cloud account needs to be created. Firebase shares the same Cloud Storage with Google Cloud, so the project that created in Firebase is also existed concurrently in Google Cloud.

Name	↓ Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
Cloud Speech-to-Text API	380	0	171	257
Artifact Registry API	16	25	27	104
Cloud Firestore API	7	0	76	865
Cloud Storage for Firebase API	7	0	415	865

Figure 5.4: Enable Cloud Storage and Speech-to-Text API

Cloud Storage for Firebase API and Cloud Speech-to-Text API should be enabled to use their services. To enable Google STT in this project, a service account is created, followed by the generation of private key for that service account. After adding a private key, the service account credential key is downloaded as JSON file.

Service Accounts		Manage service accounts	
<input type="checkbox"/>	Email	Name ↑	Actions
<input type="checkbox"/>	lecture-digest@lecture-digest-97df2.iam.gserviceaccount.com	lecture digest	

Figure 5.5: Service Account

Type	Status	Key	Key creation date	Key expiration date	
	Active	69a6a9d8897a0c15fe1f4dde9fd2f6930a1deb13	Apr 3, 2024	Jan 1, 10000	

Figure 5.6: Private Key

```
implementation ("com.google.cloud:google-cloud-speech:4.25.0")
```

Figure 5.7: Adding Google Cloud Speech API dependency into the app-level gradle file

4. Open AI

GPT-3.5 Turbo model is used to generate the key points from the lecture video transcription. To use this model, it is necessary to create an OpenAI account. The new user will be granted \$5.00 in free credits. After setting up the account, a secret key must be created, and it will not be visible again after generation. Hence, the secret key should be

copied and stored into a text file for future access to OpenAI services.

NAME	SECRET KEY	TRACKING ⓘ	CREATED	LAST USED ⓘ	PERMISSIONS
lecture-digest	sk-...GLKo	Enabled	Apr 3, 2024	Apr 22, 2024	All

Figure 5.8: Open AI Secret Key

5.4 System Operation

Users are presented with a welcome page when launching the application. They need to click ‘Start learning’ button to begin their learning journey.



Figure 5.9: Welcome Page

Upon clicking the button, user is directed to an upload page. User needs to click the plus button to choose the video file from the local device. The supported video format is mp4 and avi.

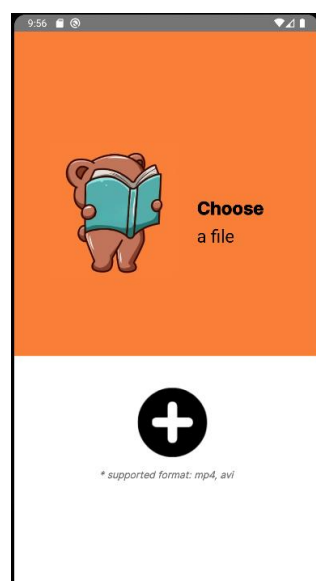


Figure 5.10: Video upload page

The gallery will be opened, and the user can pick the video file from the directory.

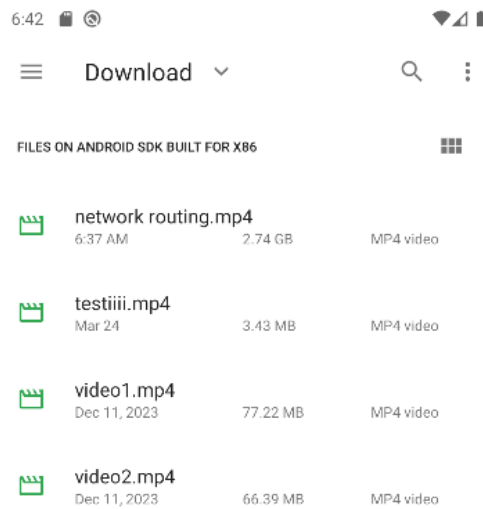


Figure 5.11: Choose video file from gallery

After selected the desired video file, the video name is appeared on the screen and upload button is displayed. User needs to click the button to upload the video.

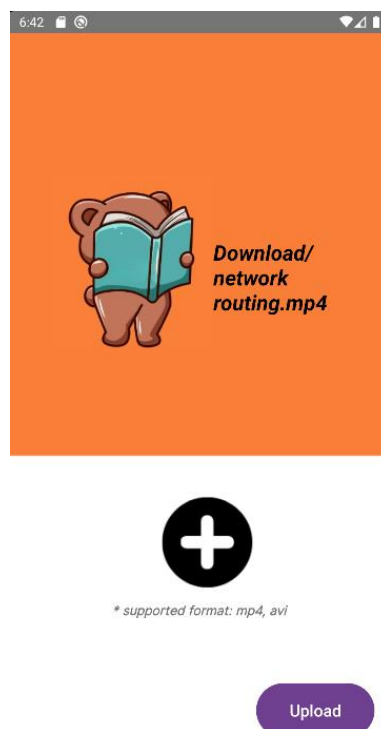


Figure 5.12: After video is selected

After clicking the upload button, a loading page is appeared.

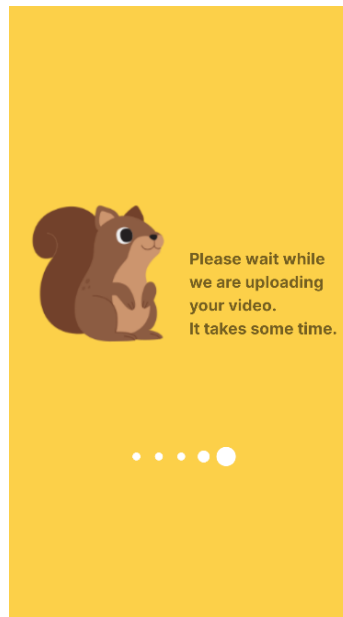


Figure 5.13: Video is uploading

At the background, the app extracts the audio track from the video, subsequently normalizes the audio file. After finish normalization, the audio file is stored into Firebase Cloud Storage. A copy is synchronized to Google Cloud Storage.

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	normalizedAudio_1713778513087	1.19 MB	audio/mp3	Apr 22, 2024
<input type="checkbox"/>	normalizedAudio_1713780672748	32.85 MB	audio/mp3	Apr 22, 2024
<input type="checkbox"/>	normalizedAudio_1713787308780	257.98 KB	audio/mp3	Apr 22, 2024

Figure 5.14: Audio file is stored in Firebase Cloud Storage

lecture-digest-97df2.appspot.com

Location: us (multiple regions in United States) | Storage class: Standard | Public access: Subject to object ACLs | Protection: Soft Delete

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE | OBSERVABILITY | INVENTORY REPORTS

Buckets > lecture-digest-97df2.appspot.com > lecture

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | TRANSFER DATA | MANAGE HOLDS | EDIT RETENTION | DOWNLOAD | DELETE

Filter by name prefix only | Filter: Filter objects and folders | Show: Live objects only

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified
<input type="checkbox"/>	normalizedAudio_1713778513087	1.2 MB	audio/mp3	Apr 22, 2024, 5:35:34 PM	Standard	Apr 22, 2024, 5:35:34 PM
<input type="checkbox"/>	normalizedAudio_1713780672748	32.9 MB	audio/mp3	Apr 22, 2024, 6:17:40 PM	Standard	Apr 22, 2024, 6:17:40 PM
<input type="checkbox"/>	normalizedAudio_1713787308780	258 KB	audio/mp3	Apr 22, 2024, 8:02:15 PM	Standard	Apr 22, 2024, 8:02:15 PM

Figure 5.15: Audio file is synchronized to Google Cloud Storage

After that, Google Speech-to-Text API is called to generate transcript from the audio file. It fetches the file from Google Cloud Storage and process the audio file. The transcript generated is served for different purposes in different modules.

Once the transcription is generated, user is navigated to the main menu page. There are three options for user to choose to understand the lecture content, which are text summary, chapter generation and chat. The pink round button allows users to enter rating page, while the green round button allows users to upload another lecture video.

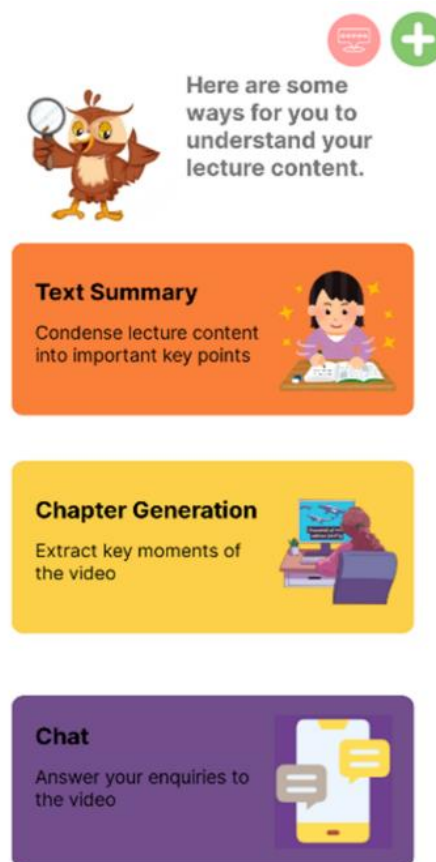


Figure 5.16: Main Menu Page

4.4.1 Text Summary Module

When the user wants to view the summary in text, they need to click on the Text Summary cell. After clicking, a loading page will be appeared. At the background, the GPT-3.5 Turbo model is called. The transcription together with the prompt are passed to it. After finish processing, the result is returned, and the loading bar is disappeared. The next arrow button is appeared, user needs to click this to view the text summary.

CHAPTER 5

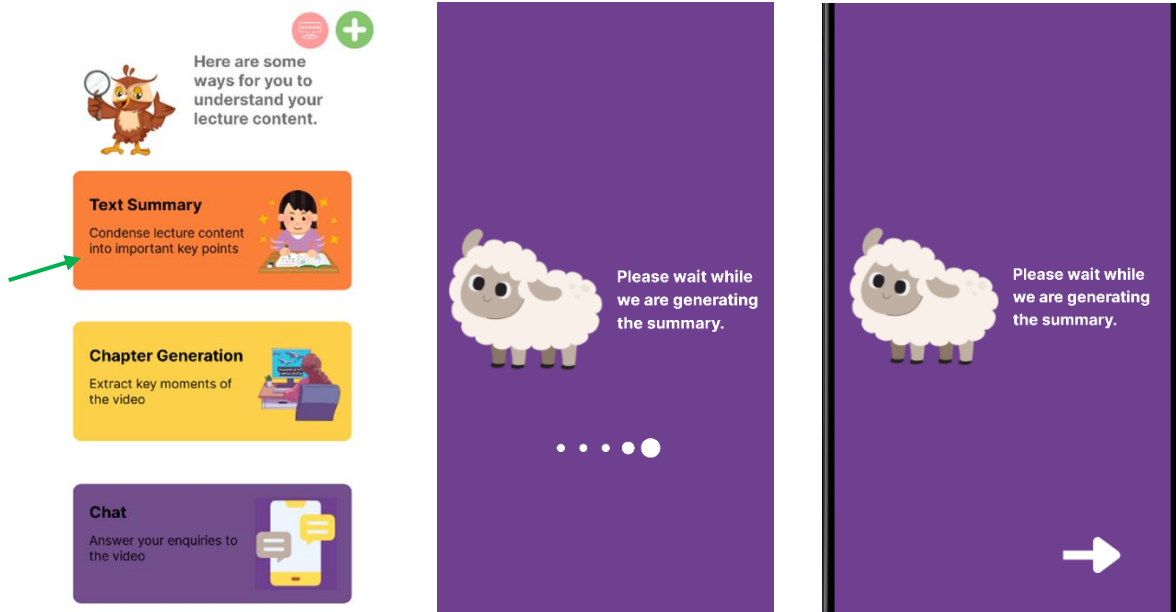


Figure 5.17: Choosing to view summary in key points (Left), Loading page while video is processing (Center), Loading page after video finish processing (Right)

The text summary is displayed in the form of bullet point to the users. The lecture video is displayed along so that the user can refer to the recording while looking at the text summary at the same time. Users can scroll through the page to read the remaining summary.

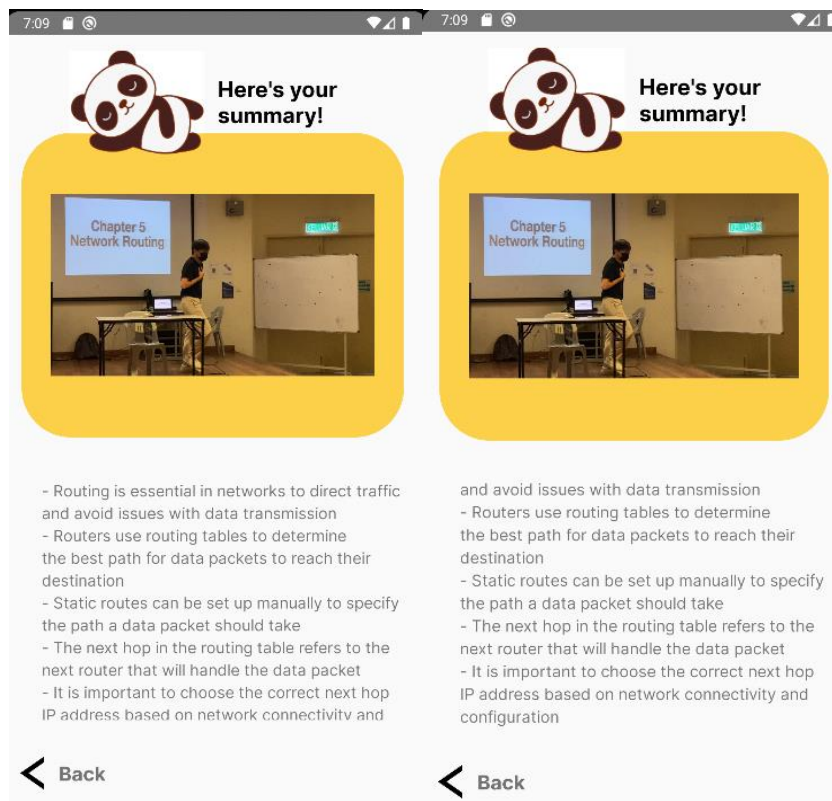


Figure 5.18: Text summary

5.4.2 Chapter Generation Module

This module generates the subchapters of the lecture video to the users. User should press the Chapter Generation cell to view the subchapters of the lecture video. A loading page is appeared after they click that cell. At the background, the MediaPipe Gesture Recognizer API is called to perform gesture recognition for the video frame. Once finish processing, the loading bar is disappeared, and the next button is displayed. Users have to click the next button to view the subchapters of the video.

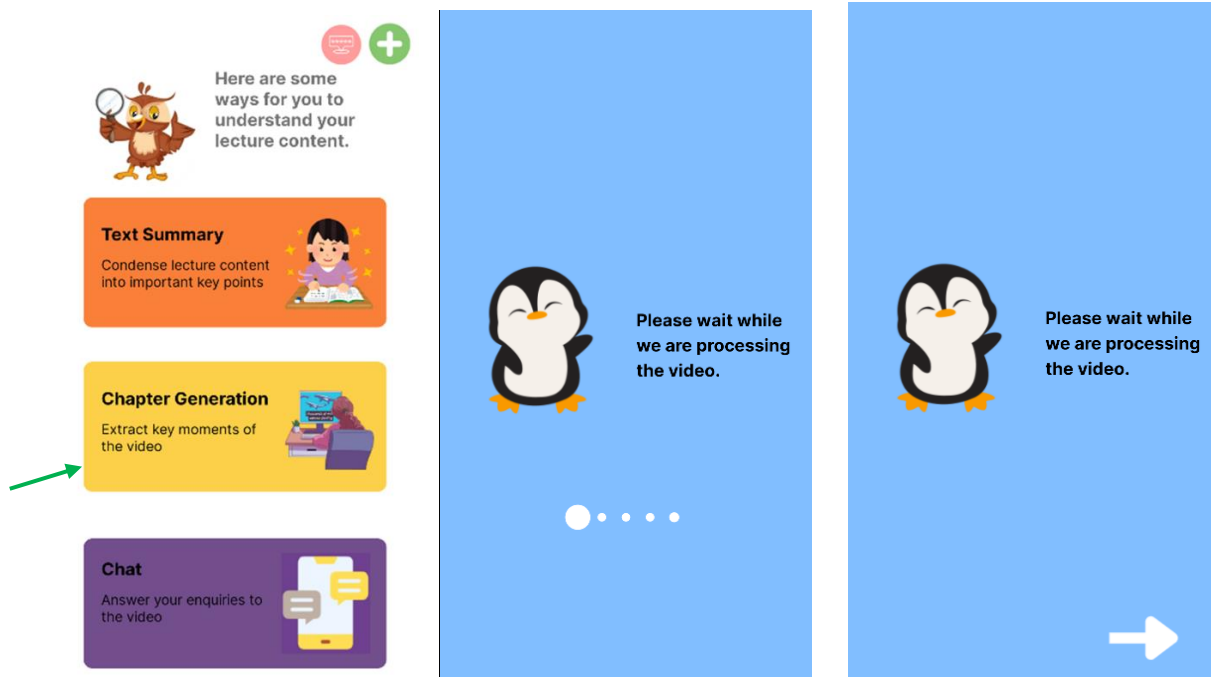


Figure 5.19: Choosing to view lecture chapters (Left), Loading page while video is processing (Center), Loading page after video finish processing (Right)

CHAPTER 5

The video and chapters are displayed to the users. They can click on chapter's title if they are just interested in specific part of the video. They are navigated to the corresponding part of the video where the timestamp matches.

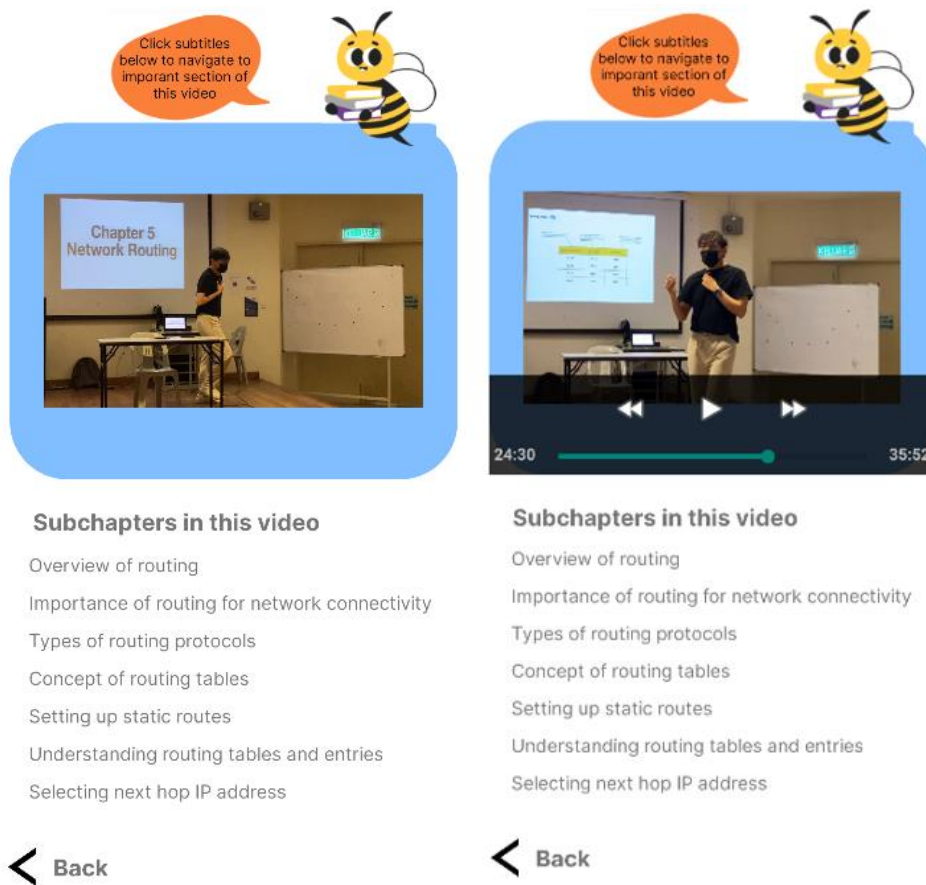


Figure 5.20: Chapter titles are displayed with the video (Left), Users are navigated to specific part of the video by clicking the chapter title (Right)

5.4.3 Chat Module

This is the module where the user can interact with Lecture Digest chatbot. Figure 5.19 shows the first page of the chat module. 4 conversation starters are provided to the users, which are Frequently Asked Question (FAQ), QnA Session, Flashcards and Quiz.

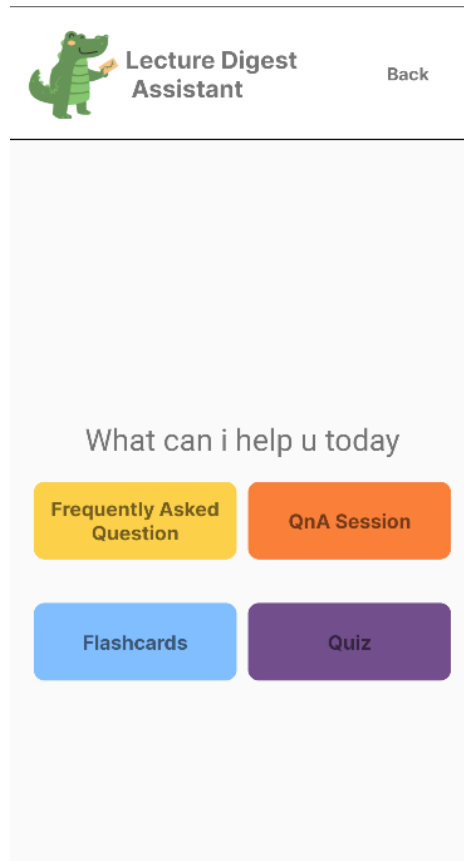


Figure 5.21: Chat module

5.4.3.1 Frequently Asked Questions (FAQ) Module

This module generates list of questions that are frequently asked in the examinations. Users will receive list of questions from Lecture Digest assistant, and then they can simply type the question number to get the corresponding answer. Upon returning the answer, Lecture Digest assistant will generate other questions to the user.

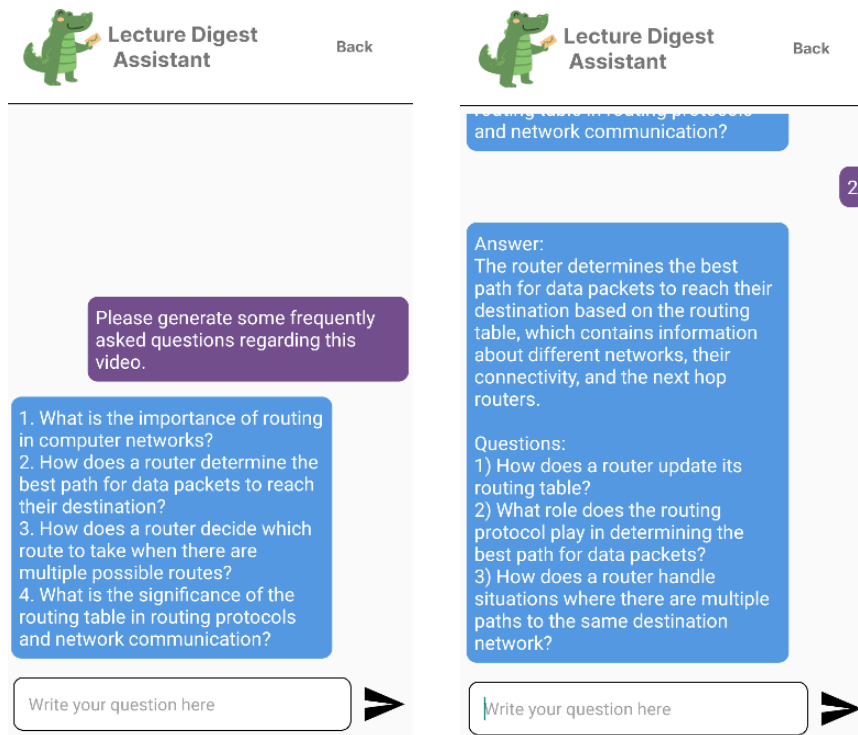


Figure 5.22: Frequently Asked Question (FAQ) Module

5.4.3.2 QnA Module

This module allows user to ask any questions related to the video. Lecture Digest assistant will answer the user question and return the video timestamp that contains the answer to the users. Therefore, the user can listen back to the recording themselves based on the timestamp received.

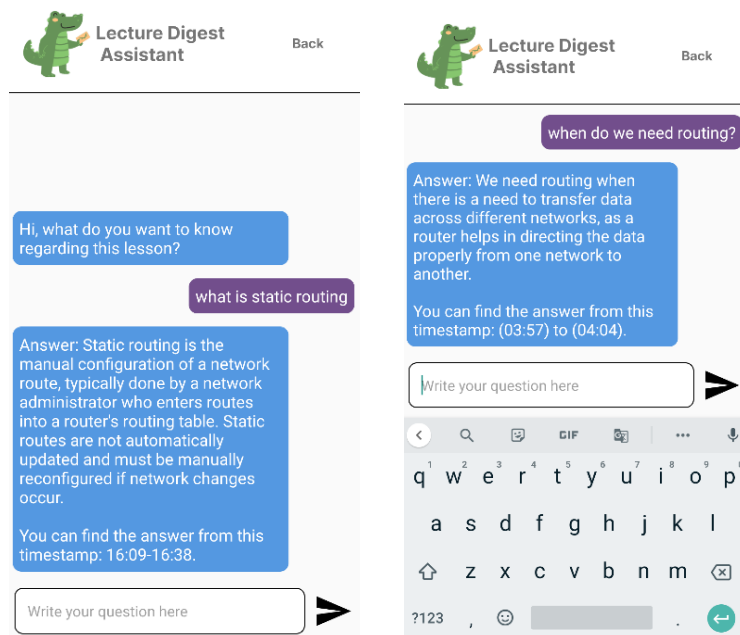


Figure 5.23: QnA Module

5.4.3.3 Flashcard Module

This module provides users a list of key terms extracted from the lesson content. The purpose of having this session is to let users quickly grasp and memorize new and important vocabulary introduced in the lesson. The key terms along with their corresponding definitions will be given to the users. User can choose to ask the questions related to the terms if they are not understand. The Lecture Digest assistant will provide the explanation to the users as well.

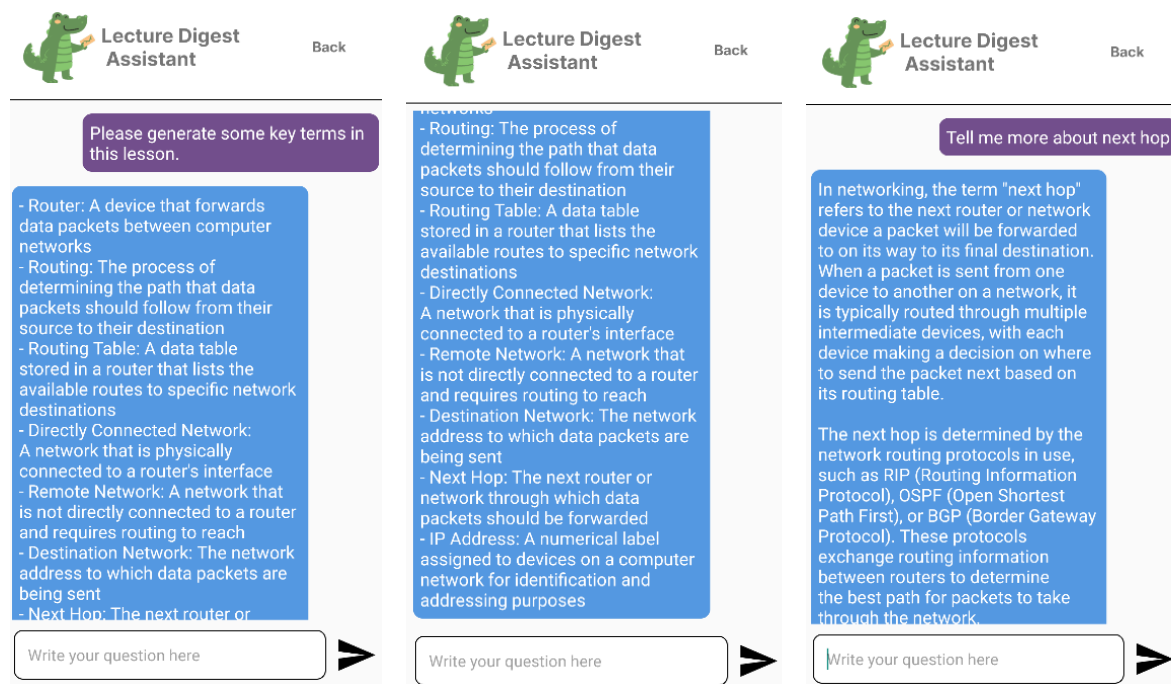


Figure 5.24: Flashcard Module

5.4.3.4 Quiz Module

This module allows users to test on their understanding on the lecture content. Lecture Digest assistant will provide a quiz question to the users. It will check the answer of the users and explain to them if they answer wrongly. They will ask whether the users want to continue answering or not. If yes, then the assistant will give other quiz question to the users.

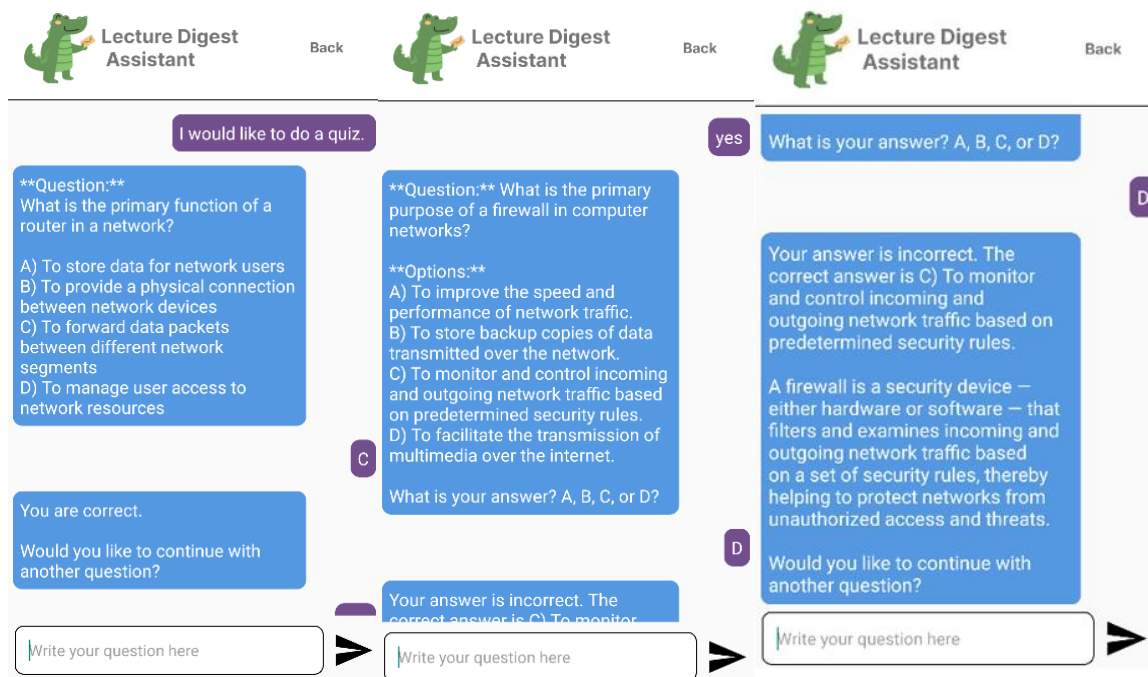


Figure 5.25: Quiz Module

5.4.4 Rating

Users are allowed to rate on the application. They need to click on the pink button to enter the rating page.

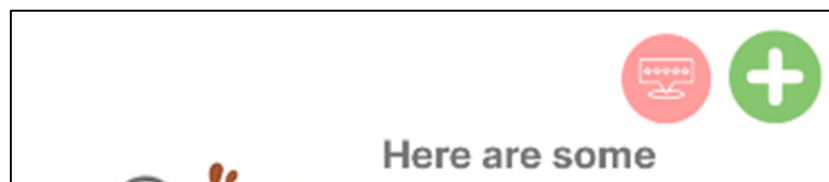


Figure 5.26: Rating Button (Pink)

They can just rate by giving 1-5 stars. After that, they should click submit button to submit the form. The rating form is stored into the Firebase Cloud Storage.

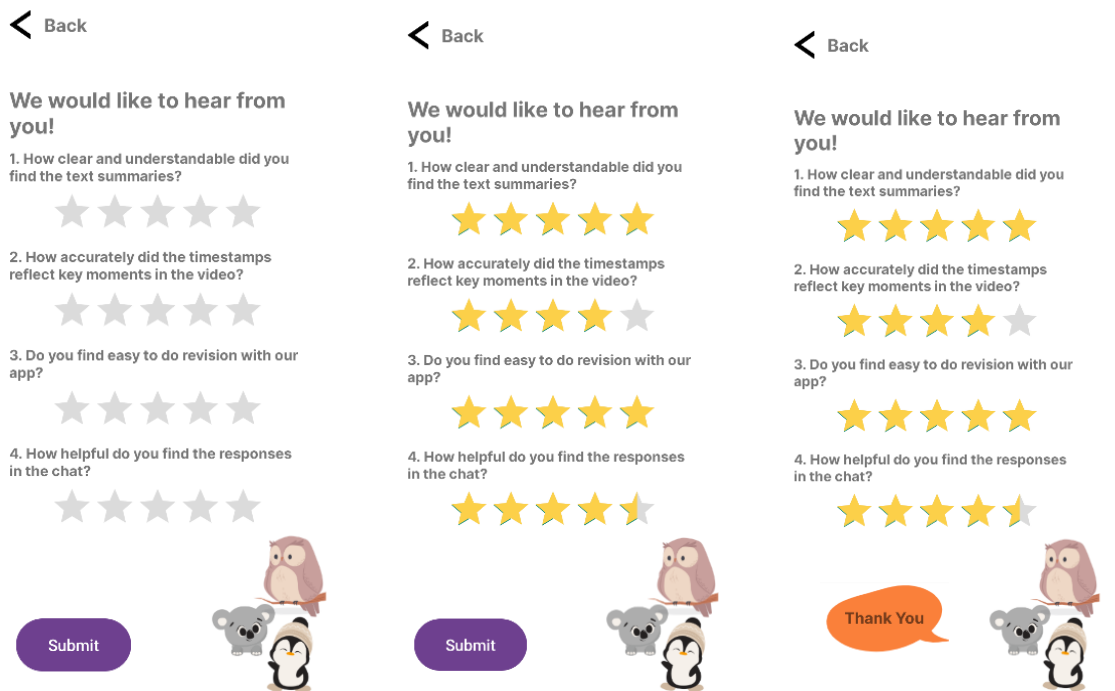
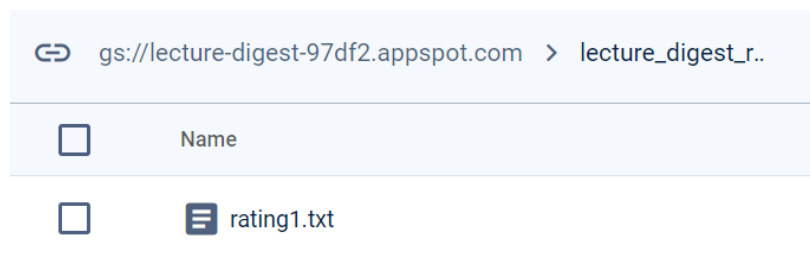


Figure 5.27: Rating Page (Left), Fill in the rating form (Center), After submit rating form (Right)



```
1. How clear and understandable did you find the text summaries? : 5.0
2. How accurately did the timestamps reflect key moments in the video? : 4.0
3. Do you find easy to do revision with our app? : 5.0
4. How helpful do you find the responses in the chat? : 4.5
```

Figure 5.28: Rating is stored into Firebase

5.5 Implementation Issues and Challenges

1. High Memory Usage

As the lecture video comes with big size, processing job on the video is quite heavy. The stage of extracting audio track from video, normalizing audio volume, uploading audio file to Firebase and generating transcription require significant amount of memory. Consequently, the application sometimes experienced slowdown and sluggishness when executing.

2. Large Video Size

The lecture videos often span hours, resulting in large file sizes that consume large amount of storage space on the laptop.

3. Long Processing Time

Large video file also leads to long processing time. It takes a significant amount of time to upload the video to Firebase and generate the transcription, which slow down the project implementation.

4. Dependencies on API

The generation of text summaries, subchapters, and the implementation of chat functions within the application rely heavily on the GPT-3.5 Turbo model. This large language model plays a crucial role in ensuring the accuracy and effectiveness of the application. Moreover, the transcriptions generated by Google Speech-to-Text also hold significant importance in the accuracy of the GPT model's results, as the model analyzes these transcriptions to generate its outputs. Any disruptions or issues with the API calls can adversely affect the application's performance. Network connectivity problems can disrupt API calls, which leads to failures in generating results.

5.6 Concluding Remark

In summary, this chapters provides a comprehensive overview of the development and operation of the system. Furthermore, the implementation issues and challenges are also discussed. Despite the challenges faced during the development process, the application has been successfully realized, and its objectives are accomplished.

Chapter 6 System Evaluation and Discussion

6.1 System Testing, Testing Setup and Results

6.1.1 Gesture Recognition Model Evaluation

The gesture recognition model is evaluated before deploying into Android Studio. The evaluation is done in Google Colab.

6.1.1.1 Gesture Recognition Evaluation Setup

1. Import libraries

Firstly, import mediapipe and matplotlib libraries. The landmark_pb2 module is imported to draw hand landmarks, while vision module is imported to perform gesture recognition task.

```
import math
import mediapipe as mp
from mediapipe.framework.formats import landmark_pb2
from mediapipe.tasks import python
from mediapipe.tasks.python import vision
```

Figure 6.1: Import libraries

2. Define visualization functions

If the gesture is detected, the gesture category is displayed, along with the accuracy score.

The hand landmarks is drawn on the image.

```
plt.rcParams.update({
    'axes.spines.top': False,
    'axes.spines.right': False,
    'axes.spines.left': False,
    'axes.spines.bottom': False,
    'xtick.labelbottom': False,
    'xtick.bottom': False,
    'ytick.labeleft': False,
    'ytick.left': False,
    'xtick.labeltop': False,
    'xtick.top': False,
    'ytick.labelright': False,
    'ytick.right': False
})

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

def display_one_image(image, title, subplot, titlesize=16):
    """Displays one image along with the predicted category name and score."""
    plt.subplot(*subplot)
    plt.imshow(image)
    if len(title) > 0:
        plt.title(title, fontsize=int(titlesize), color='black', fontdict={'verticalalignment': 'center'}, pad=int(titlesize/1.5))
    return (subplot[0], subplot[1], subplot[2]+1)
```

```

def display_batch_of_images_with_gestures_and_hand_landmarks(images, results):
    """Displays a batch of images with the gesture category and its score along with the hand landmarks."""
    # Images and labels.
    images = [image.numpy_view() for image in images]
    gestures = [top_gesture for (top_gesture, _) in results]
    multi_hand_landmarks_list = [multi_hand_landmarks for (_, multi_hand_landmarks) in results]

    # Auto-squaring - drop data that does not fit into square or square-ish rectangle.
    rows = int(math.sqrt(len(images)))
    cols = len(images) // rows

    # Size and spacing.
    FIGSIZE = 13.0
    SPACING = 0.1
    subplot=(rows,cols, 1)
    if rows < cols:
        plt.figure(figsize=(FIGSIZE,FIGSIZE/cols*rows))
    else:
        plt.figure(figsize=(FIGSIZE/rows*cols,FIGSIZE))

    # Display gestures and hand landmarks.
    for i, (image, gestures) in enumerate(zip(images[:rows*cols], gestures[:rows*cols])):
        title = f"{gestures.category_name} ({{gestures.score:.2f}})"
        dynamic_titlesize = FIGSIZE*SPACING/max(rows,cols) * 40 + 3
        annotated_image = image.copy()

        for hand_landmarks in multi_hand_landmarks_list[i]:
            hand_landmarks_proto = landmark_pb2.NormalizedLandmarkList()
            hand_landmarks_proto.landmark.extend([
                landmark_pb2.NormalizedLandmark(x=landmark.x, y=landmark.y, z=landmark.z) for landmark in hand_landmarks
            ])

            mp_drawing.draw_landmarks(
                annotated_image,
                hand_landmarks_proto,
                mp_hands.HAND_CONNECTIONS,
                mp_drawing_styles.get_default_hand_landmarks_style(),
                mp_drawing_styles.get_default_hand_connections_style())

        subplot = display_one_image(annotated_image, title, subplot, titlesize=dynamic_titlesize)

plt.tight_layout()
plt.subplots_adjust(wspace=SPACING, hspace=SPACING)
plt.show()

```

Figure 6.2: Visualization functions

3. Upload test images

Images to be tested are uploaded to Google Colab.

```

from google.colab import files
uploaded = files.upload()

for filename in uploaded:
    content = uploaded[filename]
    with open(filename, 'wb') as f:
        f.write(content)
IMAGE_FILENAMES = list(uploaded.keys())

print('Uploaded files:', IMAGE_FILENAMES)

```

Figure 6.3: Upload test images

4. Testing

Gesture recognition is conducted in this stage. Before performing testing, the trained gesture recognition model named 'gesture_recognizer.task' is uploaded to Google Colab. The GestureRecognizer object will perform classification based on this model. After recognition, the results are displayed.

```
# Create an GestureRecognizer object.
base_options = python.BaseOptions(model_asset_path='gesture_recognizer.task')
options = vision.GestureRecognizerOptions(base_options=base_options)
recognizer = vision.GestureRecognizer.create_from_options(options)

images = []
results = []
for image_file_name in IMAGE_FILENAMES:

    # Load the input image.
    image = mp.Image.create_from_file(image_file_name)

    # Recognize gestures in the input image.
    recognition_result = recognizer.recognize(image)

    # Process the result and visualize
    images.append(image)
    top_gesture = recognition_result.gestures[0][0]
    hand_landmarks = recognition_result.hand_landmarks
    results.append((top_gesture, hand_landmarks))

display_batch_of_images_with_gestures_and_hand_landmarks(images, results)
```

Figure 6.4: Test images

There are total 20 images being tested. Table 6.1 shows the testing results.

Table 6.1: Testing Results

Gesture	Test Image	Detected?	Correct?	Accuracy score
Circle	test1.jpg	Yes	Yes	0.88
	test2.jpg	Yes	Yes	0.71
	test3.jpg	Yes	Yes	0.90
Pointing	test4.jpg	Yes	Yes	0.64
	test5.jpg	No	-	-
	test6.jpg	Yes	Yes	0.70
Steeple	test7.jpg	Yes	Yes	0.87
	test8.jpg	Yes	Yes	0.56
	test9.jpg	Yes	No	0.76
One	test10.jpg	Yes	Yes	0.76
	test11.jpg	Yes	Yes	0.88
	test12.jpg	Yes	Yes	0.63
	test13.jpg	Yes	Yes	0.55

Two	test14.jpg	Yes	Yes	0.82
	test15.jpg	Yes	Yes	0.79
	test16.jpg	Yes	Yes	0.76
	test17.jpg	Yes	Yes	0.54
Three	test18.jpg	Yes	Yes	0.50
	test19.jpg	Yes	Yes	0.63
	test20.jpg	Yes	Yes	0.55
	test21.jpg	Yes	Yes	0.76
Four	test22.jpg	No	-	-
	test23.jpg	Yes	Yes	0.88
Five	test24.jpg	Yes	Yes	0.67
	test25.jpg	Yes	Yes	0.77



Figure 6.5: Result of test14.jpg (left), test19.jpg (center), test6.jpg (right)

Out of 20 images, 18 images have gestures being detected, and only 1 image has been misclassified. In overall, the gesture recognition model demonstrates robustness in accurately identifying various types of gestures, making it well-suited for integration into the Lecture Digest application.

6.1.2 User Experience Evaluation

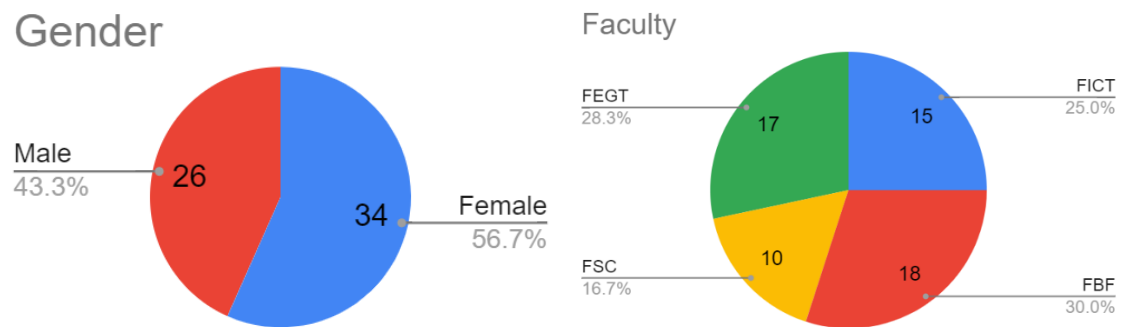


Figure 6.6: Demographic of Respondents

Survey was conducted to gain insights on the user satisfaction on the services provided by Lecture Digest. 60 UTAR students are being surveyed, 26 of them are male while 34 of them are female. They came from different faculties in UTAR, which are (Faculty of Engineering and Green Technology) FEGT, Faculty of Science (FSc), Faculty of Information Technology and Communication (FICT) and Faculty of Business and Finance (FBF).

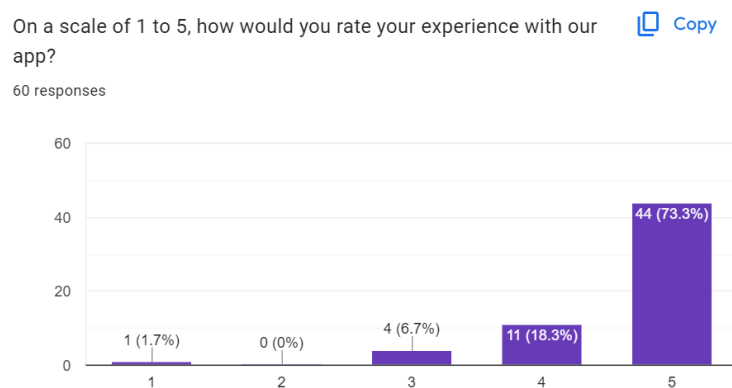


Figure 6.7: Survey Question 1

Survey question one analyses the user experience with Lecture Digest. 55 out of 60 of respondents are satisfied about our application, with 11 of them rated scale of 4 and 44 of them reated scale of 5.

On a scale of 1 to 5, how likely are you to recommend our application to your friend? [Copy](#)

60 responses

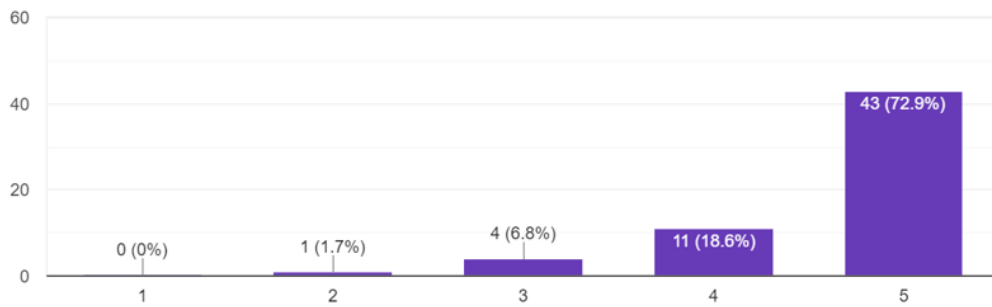


Figure 6.8 Survey Question 2

Survey questions 2 is about the willingness of respondents in recommending our app to other people. 91.5% of respondents are willing to recommend it other users and 6.8% holds neutral opinion.

On a scale of 1 to 5, how satisfied are you with the quality the text summary? [Copy](#)

60 responses

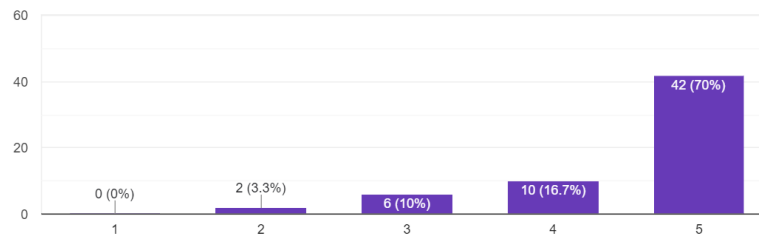


Figure 6.9: Survey Question 3

Survey question 3 assess the users satisfactory on text summary module. Most users are satisfied with the summary they obtained, with 70% of them rated 5 and 16.7% of them rated 4.

On a scale of 1 to 5, do you agree that navigating to crucial parts of the video help by clicking the video chapters help to reduce revision time?



60 responses

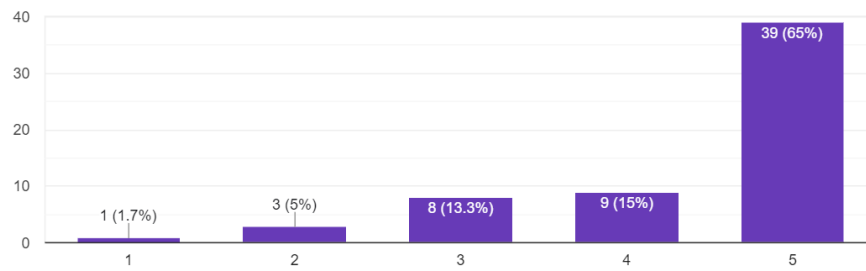


Figure 6.10: Survey Question 4

Survey questions 4 assess user's satisfactory on chapter generation module. 65% of users rate for 5, indicating that they are very agree that navigating to specific part of the video helps them to reduce revision time. 13.3% and 15% of respondents rate for the scale of 3 and 4 respectively.

On a scale of 1 to 5, how satisfied are you with the service provided by our chat assistance?



60 responses

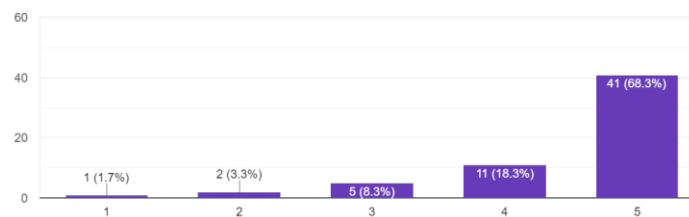


Figure 6.11: Survey Question 5

Survey questions 5 assess students' satisfactory on the functionality of chat module, majority respondents are satisfied with its services and responses, with more than half of the respondents rate for 4 and 5 (18.3% and 68.3% respectively).

Overall, most users who have tried our Lecture Digest application report a positive experience with it.

6.1.3 Comparison of Text Summary Module with other Summary Tools

As most summary tool existed online are just able to generate text summary tool from the video, so the result of text summary module in Lecture Digest is compared with other summarization tools.

Summary tool: summarize.tech

summarize.tech only summarizes the video on YouTube, so the video file is uploaded to the YouTube channel for comparison purpose.

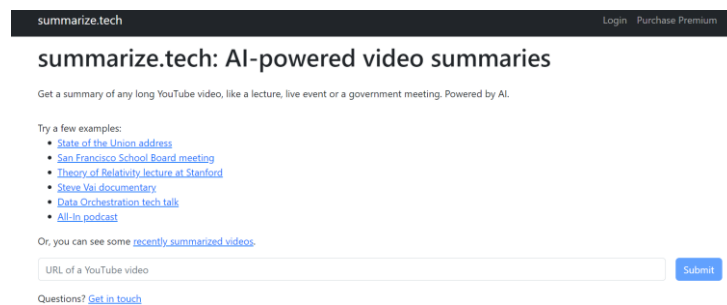


Figure 6.12: summarize.tech

6.1.3.1 Setup

1. Press “CREATE” button and click “Upload videos”. After that, click “SELECT FILES” to upload the video from the local device.

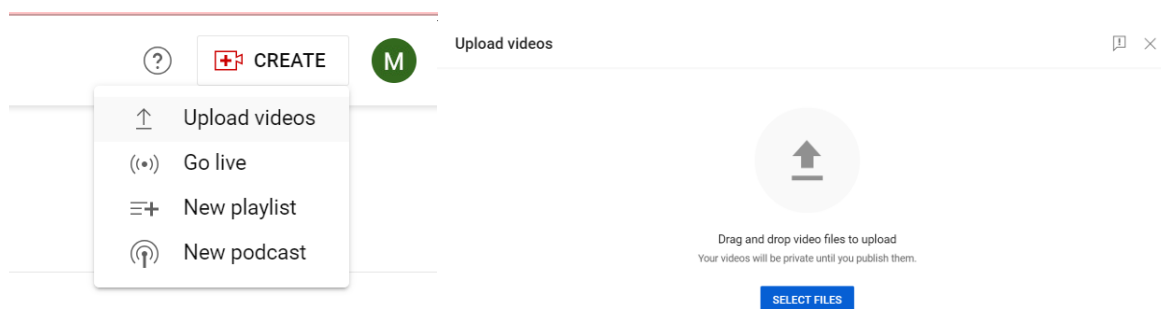


Figure 6.13: Upload video to YouTube channel

2. After select the files from device, add the details of the videos.

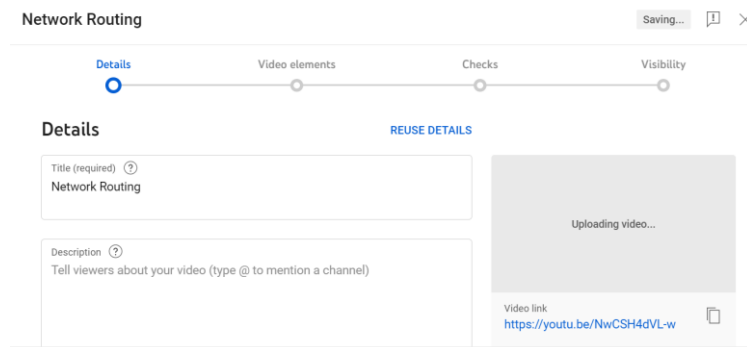


Figure 6.14: Add details of video

3. Wait for the videos to upload and process. After finish uploading, copy the link of YouTube link of the video by clicking “Get shareable link”. The link is pasted to the summarizer.tech.

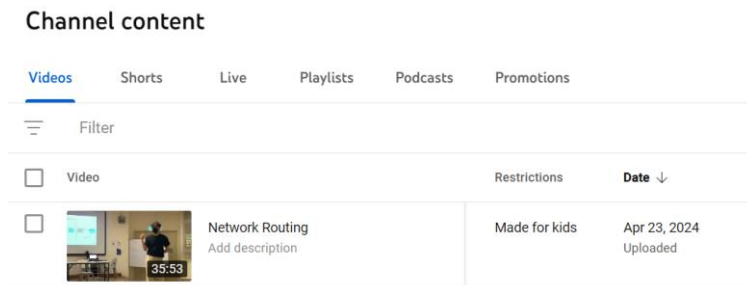


Figure 6.15: Video is uploaded

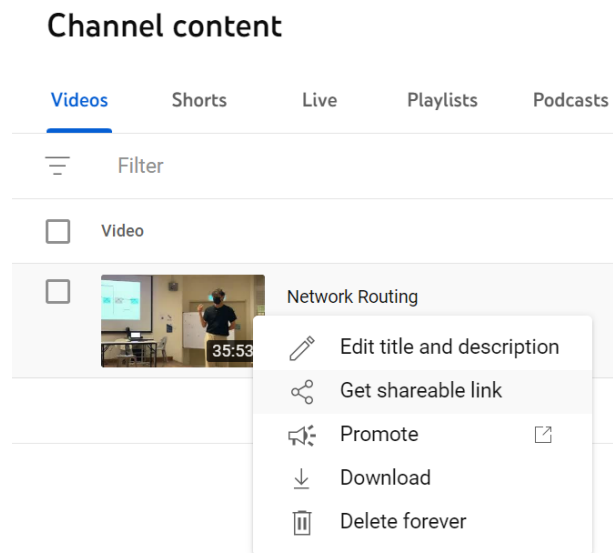


Figure 6.16: Get YouTube link

6.1.3.2 Comparison of text summary between Lecture Digest and summarizer.tech

Summary of [network routing](#)

This is an AI generated summary. There may be inaccuracies. - The green links below are Amazon affiliate links where summarize.tech may earn a commission.

[Summarize another video](#) · [Purchase summarize.tech Premium](#)

00:00:00 - 00:35:00

In the "network routing" YouTube video, the speaker explains that routing is necessary for connecting different networks and is compared to Google Maps guiding traffic between sources and destinations. Routing is only required when there are two or more networks that need to communicate, and routers determine the best path for sending packets based on their [routing tables](#). The speaker covers the concept of routing precedence, static routing, and the importance of understanding routing tables for exams. When determining the next hop router for network addresses, the speaker emphasizes being systematic and knowing network addresses, with directly connected networks having no next hop router. The correct IP address for a router is chosen based on the same network as the next hop router, and the speaker clarifies the use of IP addresses for internal and external networks.

- Routing is essential in networks to direct traffic and avoid issues with data transmission
- Routers use routing tables to determine the best path for data packets to reach their destination
- Static routes can be set up manually to specify the path a data packet should take
- The next hop in the routing table refers to the next router that will handle the data packet
- It is important to choose the correct next hop IP address based on network connectivity and configuration

Figure 6.17: Summary from summarizer.tech (Left) and Lecture Digest (Right)

As depicted in Figure 6.23, the summary generated by summarizer.tech presented in paragraphs, whereas Lecture Digest provides summaries in bullet points. However, offering key points enhances readability, particularly for lengthy summaries.

The summary generated by summarizer.tech tends to be longer and more generalized compared to Lecture Digest that is more straight to the point. For instance, summarizer.tech often uses general terms such as "the speakers cover the concept of " or "the speaker clarifies," whereas Lecture Digest directly presents key points from the lesson. This direct approach focus on key details and important points, making the summary clear and precise. It eliminates unnecessary details, thus can help the users to grasp main ideas quickly.

6.1.3.3 Summary on the comparison of Lecture Digest with other summarizer tools

Besides summarizer.tech, we also compare the text summary results with another 4 summarizer tools. Lecture Digest application outperforms these tools in terms of accuracy, length and efficiency. Table below shows the comparison summary.

Table 6.2: Comparison of Text summary module with other summarizer tools

Summarizer Tool/App	Comments
summarizer.tech	Summary is lengthy and general.
Descript	Summary result is too general.
Sider AI	Take long processing time to generate result.
Notta	Summary result is irrelevant to the lecture content.
NoteGPT	Summary result is too lengthy.

6.2 Project Challenges

1. API restrictions

The functionality of the Lecture Digest application heavily relies on the GPT-3.5 model for generating results. Occasionally, the model may produce incorrect results, which is unavoidable. Additionally, there are constraints regarding the frequency and size of requests sent to the GPT-3.5 model. Specifically, only a maximum of three requests can be made within a one-minute period, and the token limit for each request should not exceed 16,385 tokens. Therefore, when utilizing the chat functions to demonstrate the application's capabilities, it is important to ensure that requests are not sent too rapidly within one minute and that the instructions provided are concise and within the token limit, while still being clear and precise.

2. Duplicancy issue on the class dependencies

Initially, the user rating is planned to store in Cloud Firestore, a database designed for NoSQL data storage. However, when attempting to use both Cloud Firestore and Google Speech To Text API together, a duplicate class error is occurred. As a result, the user ratings are stored in cloud storage as text files instead storing in Cloud Firestore.

6.3 Objectives Evaluation

This project has successfully accomplish all stated objectives.

1. To develop a lecture digest – to automatically summarize long lecture videos into text summary and short video summary.

In this project, the Lecture Digest application, which comprises three modules: text summary, chapter generation, and chat module have been successfully developed. Users have the flexibility to choose any of these modes to comprehend lecture content. Each of these features contributes to enhancing the effectiveness of revision from different perspectives.

2. To develop a text summarization method using Google Speech-to-Text for transcription and GPT-3.5 Turbo model for summarization

Users can view the text summary of the video. The video transcription is produced using Google Speech-to-Text and then processed by the GPT-3.5 model to produce the lesson summary in text. The summary is displayed in point form.

3. To train a custom hand gesture recognition model based on MediaPipe API using transfer learning to recognize important points in the lecture videos

The Lecture Digest application can generate the video chapters which reveal the key moments in the lecture through gesture recognition. Users can navigate to the important part of the lecture by clicking the video chapters. The gesture recognition model is trained using MediaPipe Model Maker and passed to MediaPipe Gesture Recognizer API to perform gesture recognition.

4. To develop a chatbot tutor for user to interact and ask questions based on the lecture video

Users can engage with the chatbot tutor in Lecture Digest. The chatbot can generate frequently asked questions, answer the questions of the users, provide key terms of the lecture and conduct quiz to the users. This interactive feature enhances user experience and facilitates a deeper understanding of the lecture content.

6.4 Concluding Remark

This chapter discusses the testing conducted on the Lecture Digest application to evaluate its performance. Compared with other text summarizer tools, it stands out for its precise summaries. Additionally, it receives high user rating from users, indicating a high level of satisfaction with the user experience on this application. The gesture recognition accuracy has also been assessed, and it performs well in classifying gestures with minimal misclassifications. All objectives outlined in Chapter 1 have been achieved, demonstrating the successful development of this project. Overall, the Lecture Digest application provides high-quality functionalities, which enhance the user experience and usability.

Chapter 7 Conclusion and Recommendation

7.1 Conclusion

In conclusion, a user-friendly lecture digest application has been successfully realized. This application offers various interactive features aimed at aiding students in their studies by catering to diverse learning styles and preferences. It proves beneficial for university students by effectively condensing lengthy and tedious lecture content into multiple digestible formats. Through this application, lecture content can be summarized into concise key points, and important sections within videos can be pinpointed to save students time spent on extensive review. Students can enter their queries to the chatbot and receive answer from it, access to frequently asked questions about the lesson, utilizing flashcards for better memorization and involving in quizzes to test their understanding.

The Lecture Digest application has been successfully developed by utilizing various technologies such as GPT-3.5 Turbo model, Google Speech-to-Text and MediaPipe. All predefined objective and scope have been met. The application has demonstrated its novelty in providing learning and study assistance from different perspectives, thus effectively improving its overall usability. In future, this application will enhance the study experience of the students, and thus improving their academic performance.

7.2 Recommendation

There are several improvements that can be added to this project to improve its usability in the future.

Firstly, the gesture recognition model can be improved. A larger dataset should be created to enhance training accuracy. Additionally, data augmentation techniques can be employed to generate more complex and noisy images, allowing the model to adapt to various qualities of images and free from overfitting issue.

Secondly, a translation feature can be integrated into the application to translate outputs into the user's desired language. This feature would enable users to comprehend content more effectively, especially when they encounter unfamiliar vocabulary or complex sentences.

Thirdly, it is advisable to invest in a more powerful GPT model to generate more accurate results. For instance, utilizing the GPT-4 turbo model, which is currently the state-of-the-art Large Language Model, would yield more precise and professional results.

REFERENCES

- [1] T. Wangchen *et al.*, “EDUZONE – a educational video summarizer and digital human assistant for effective learning,” *2022 7th International Conference on Information Technology Research (ICITR)*, 2022. doi:10.1109/icitr57877.2022.9993155
- [2] J. Kim *et al.*, “Understanding in-video dropouts and Interaction Peaks in online lecture videos,” *Proceedings of the first ACM conference on Learning @ scale conference*, 2014. doi:10.1145/2556325.2566237
- [3] C.-Y. Mo, C. Wang, J. Dai, and P. Jin, “Video playback speed influence on learning effect from the perspective of personalized adaptive learning: A study based on cognitive load theory,” *Frontiers in Psychology*, vol. 13, 2022. doi:10.3389/fpsyg.2022.839982
- [4] Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., et al. (2014). Deep speech: scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567. doi: 10.48550/arXiv.1412.5567
- [5] A. Ferraro, A. Galli, V. La Gatta, and M. Postiglione, “Benchmarking open source and paid services for speech to text: An analysis of quality and input variety,” *Frontiers in Big Data*, vol. 6, Sep. 2023. doi:10.3389/fdata.2023.1210559
- [6] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., et al. (2021). SpeechBrain: a general-purpose speech toolkit. arXiv preprint arXiv:2106.04624. doi: 10.48550/arXiv.2106.04624
- [7] G. H. Kranz, “Transcribe,” Amazon, <https://aws.amazon.com/transcribe/> (accessed Sep. 9, 2023).
- [8] “Speech-to-text: Automatic speech recognition | google cloud,” Google, <https://cloud.google.com/speech-to-text/> (accessed Sep. 9, 2023).

REFERENCES

- [9] “Speech to text – audio to text translation: Microsoft azure,” Speech to Text – Audio to Text Translation | Microsoft Azure, <https://azure.microsoft.com/en-us/products/ai-services/speech-to-text> (accessed Sep. 9, 2023).
- [10] “MediaPipe,” *Google Developers*. <https://developers.google.com/mediapipe>
- [11] “On-Device, Real-Time Hand Tracking with MediaPipe,” *research.google*.
<https://research.google/blog/on-device-real-time-hand-tracking-with-mediapipe/>
- [12] J. Yin, “Body language classification and communicative context,” *Proceedings of the International Conference on Education, Language, Art and Intercultural Communication*, 2014. doi:10.2991/icelaic-14.2014.105
- [13] M. Beege *et al.*, “Investigating the effects of beat and deictic gestures of a lecturer in educational videos,” *Computers & Education*, vol. 156, p. 103955, 2020. doi:10.1016/j.compedu.2020.103955
- [14] R. Ravinal, “7 important hand gestures,” Toastmasters International,
<https://www.toastmasters.org/magazine/magazine-issues/2021/june/7-hand-gestures>
(accessed Nov. 23, 2023).
- [15] Z. Peng, Z. Yang, J. Xiahou, and T. Xie, “Recognizing teachers’ hand gestures for effective non-verbal interaction,” *Applied Sciences*, vol. 12, no. 22, p. 11717, 2022. doi:10.3390/app122211717
- [16] J. R. Zhang, K. Guo, C. Herwana, and J. R. Kender, “Annotation and taxonomy of gestures in lecture videos,” 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, 2010. doi:10.1109/cvprw.2010.5543253
- [17] Y. Tian and M.-L. Bourguet, “Lecturers’ hand gestures as clues to detect pedagogical significance in video lectures,” *Proceedings of the European Conference on Cognitive Ergonomics*, 2016. doi:10.1145/2970930.2970933
- [18] “Chatgpt,” ChatGPT, <https://openai.com/chatgpt> (accessed Sep. 9, 2023).

REFERENCES

- [19] J. G. Meyer et al., “CHATGPT and large language models in academia: Opportunities and challenges,” *BioData Mining*, vol. 16, no. 1, 2023. doi:10.1186/s13040-023-00339-9
- [20] “Use ai to summarize scientific articles in seconds,” *SciSummary*, <https://scisummary.com/> (accessed Sep. 9, 2023).
- [21] *Artifact*, <https://artifact.news/> (accessed Sep. 9, 2023).
- [22] *Artifact Team*, “Summaries tool, now on Artifact,” *Medium*, <https://medium.com/artifact-news/summaries-tool-now-on-artifact-8eee439049fc> (accessed Sep. 9, 2023).
- [23] “Generate summaries for YouTube videos using chatgpt: Chrome extension,” *YouTubeDigest*, <https://www.youtubedigest.app/> (accessed Sep. 9, 2023).
- [24] “Ai note taking tool for clever learning,” *NoteGPT*, <https://notegpt.io/> (accessed Sep. 9, 2023).
- [25] H. B. U. Haq, M. Asif, M. B. Ahmad, “Video Summarization Techniques: A Review,” *International Journal of Scientific Technology Research*, vol. 9(11), pp. 146-153, 2020.
- [26] “Video summarizer,” *Mindgrasp*, <https://mindgrasp.ai/video-summarizer/> (accessed Sep. 9, 2023).
- [27] “Transforming podcasts into bite-sized brilliance,” *PodPulse*, <https://podpulse.ai/> (accessed Sep. 9, 2023).
- [28] “The GPT-3 Architecture, on a Napkin,” *dugas.ch*.
https://dugas.ch/artificial_curiosity/GPT_architecture.html

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 2
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Revise the work done in FYP 1

2. WORK TO BE DONE

- Collect gesture dataset
- Preprocess dataset
- Find technologies for gesture recognition

3. PROBLEMS ENCOUNTERED

No

4. SELF EVALUATION OF THE PROGRESS

- OK



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 4
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Collect and preprocess dataset, found MediaPipe to perform gesture recognition

2. WORK TO BE DONE

- Design gesture recognition pipeline
- Train gesture recognition model

3. PROBLEMS ENCOUNTERED

- Low accuracy on training model

4. SELF EVALUATION OF THE PROGRESS

- Need to fine tune hyperparameters of the model



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 6
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Design gesture recognition pipeline, finish training gesture recognition model

2. WORK TO BE DONE

- Integrate gesture recognition model into application
- Design chapter generation module on application

3. PROBLEMS ENCOUNTERED

- Class dependency issues

4. SELF EVALUATION OF THE PROGRESS

- Look for open source to solve the class dependency issue



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 8
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Finish integrating gesture recognition model into application

2. WORK TO BE DONE

- Continue design chapter generation module
- Design pipeline for chat module

3. PROBLEMS ENCOUNTERED

No

4. SELF EVALUATION OF THE PROGRESS

- OK



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 10
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Finish developing chat generation module

2. WORK TO BE DONE

- Continue develop chat module
- Integration of all modules

3. PROBLEMS ENCOUNTERED

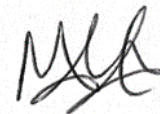
- Result from GPT-model is inconsistent

4. SELF EVALUATION OF THE PROGRESS

- Perform prompt engineering to get more accurate answer



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Sem 3	Study week no.: 12
Student Name & ID: Melissa Yap Chia Chean 20ACB01674	
Supervisor: Dr Aun Yichiet	
Project Title: Lecture Digest – Auto Summarization and Key Points Recognition	

1. WORK DONE

- Done developing the whole application

2. WORK TO BE DONE

- Report writing
- System testing

3. PROBLEMS ENCOUNTERED

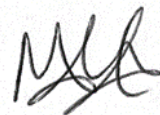
No

4. SELF EVALUATION OF THE PROGRESS

- OK



Supervisor's signature



Student's signature

POSTER



Lecture Digest

Auto-Summarization and Key Points Recognition

Poor time management among students often results in time constraints when revising. Here's how our application can facilitate a productive and engaging study and learning experience



Objectives

- Deliver a multifeatured lecture digest mobile application
- Perform transcript-based key points recognition
- Generate video subchapters by employing gesture recognition
- Create an interactive chat module capable of delivering real-time responses



Features

- Text Summary
- Video Chapters
- Chatbot Tutor



Methods

- Gesture Recognition
- Computer Vision
- Large Language Model (LLM)



Technologies

- Google Speech-to-Text
- GPT-3.5 Turbo Model
- MediaPipe



Contributions

- Time saving
- Personalized learning
- Engagement
- Improved retention



Conclusion

- This application aids students in comprehending lecture content through various means, thereby enhancing their study experience and improving their performance in examinations.

Developer: Melissa Yap Chia Chean

Supervisor: Dr Aun Yichiet

QUESTIONNAIRE

Lecture Digest Survey

I am Melissa Yap Chia Chean, a final-year student at Universiti Tunku Abdul Rahman pursuing a Bachelor of Computer Science degree. You are invited to participate in a survey regarding the use of the Lecture Digest app. Your insights and feedback are invaluable to help us improve the app and make it more effective for users like you.

Thank you in advance for your participation. Your input is highly appreciated.

1. Gender

Mark only one oval.

- Male
 Female

2. Faculty

Mark only one oval.

- Faculty of Business and Finance
 Faculty of Information and Communication Technology
 Faculty of Science
 Center of Foundation
 Faculty of Engineering and Green Technology

3. On a scale of 1 to 5, how would you rate your experience with our app?

Mark only one oval.

1 2 3 4 5

APPENDICES

4. On a scale of 1 to 5, how likely are you to recommend our application to your friend?

Mark only one oval.

1 2 3 4 5

5. On a scale of 1 to 5, how satisfied are you with the quality the text summary?

Mark only one oval.

1 2 3 4 5

6. On a scale of 1 to 5, do you agree that navigating to crucial parts of the video help by clicking the video chapters help to reduce revision time?

Mark only one oval.

1 2 3 4 5

7. On a scale of 1 to 5, how satisfied are you with the service provided by our chat assistance?

Mark only one oval.

1 2 3 4 5

PLAGIARISM CHECK RESULT

Lecture Digest - Auto Summarization and Key Points Recognition.pdf			
ORIGINALITY REPORT			
8%	5%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	eprints.utar.edu.my Internet Source	2%	
2	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%	
3	link.springer.com Internet Source	<1%	
4	Submitted to University of Sousse Student Paper	<1%	
5	Mengting Xiao, Zhiquan Feng, Xiaohui Yang, Tao Xu, Qingbei Guo. "Multimodal interaction design and application in augmented reality for chemical experiment", Virtual Reality & Intelligent Hardware, 2020 Publication	<1%	
6	picovoice.ai Internet Source	<1%	
7	Mohammad Hasanzadeh Mofrad, Daniel Mosse. "Speech recognition and voice separation for the internet of things",	<1%	

Proceedings of the 8th International
Conference on the Internet of Things - IOT
'18, 2018

Publication

8	Submitted to Ton Duc Thang University Student Paper	<1 %
9	d197for5662m48.cloudfront.net Internet Source	<1 %
10	www.analyticsvidhya.com Internet Source	<1 %
11	Giovanni Spitale, Gerold Schneider, Federico Germani, Nikola Biller-Andorno. "Exploring the role of AI in classifying, analyzing, and generating case reports on assisted suicide cases: feasibility and ethical implications", Frontiers in Artificial Intelligence, 2023 Publication	<1 %
12	Antonino Ferraro, Antonio Galli, Valerio La Gatta, Marco Postiglione. "Benchmarking open source and paid services for speech to text: an analysis of quality and input variety", Frontiers in Big Data, 2023 Publication	<1 %
13	Ritu Gala, Revathi Vijayaraghavan, Valmik Nikam, Arvind Kiwelekar. "Real-Time Cognitive Evaluation of Online Learners through Automatically Generated Questions", 2021	<1 %

	IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), 2021 Publication	
14	5wwwwww.easychair.org Internet Source	<1 %
15	Mohaimenul Azam Khan Raiaan, Md. Saddam Hossain Mukta, Kaniz Fatema, Nur Mohammad Fahad et al. "A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges", IEEE Access, 2024 Publication	<1 %
16	stevecondylios.r-universe.dev Internet Source	<1 %
17	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
18	Submitted to University of Bedfordshire Student Paper	<1 %
19	Submitted to University of Nottingham Student Paper	<1 %
20	www.geeksforgeeks.org Internet Source	<1 %
21	Submitted to Universiti Teknologi Petronas Student Paper	<1 %

APPENDICES

22	Submitted to Universiti Tenaga Nasional Student Paper	<1 %
23	www.mdpi.com Internet Source	<1 %
24	Submitted to Queen's University of Belfast Student Paper	<1 %
25	Submitted to Liverpool John Moores University Student Paper	<1 %
26	blogpost10986.pointblog.net Internet Source	<1 %
27	hdl.handle.net Internet Source	<1 %
28	www.waveshare.com Internet Source	<1 %
29	Kianoush Haratiannejadi, Neshat Elhami Fard, Rastko R. Selmic. "Smart Glove and Hand Gesture-based Control Interface For Multi-rotor Aerial Vehicles", 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019 Publication	<1 %
30	Yrjo Lappalainen, Nikesh Narayanan. "Aisha: A Custom AI Library Chatbot Using the ChatGPT API", Journal of Web Librarianship, 2023 Publication	<1 %
31	docplayer.net Internet Source	<1 %



**FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

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

Full Name(s) of Candidate(s)	Melissa Yap Chia Chean
ID Number(s)	20ACB01674
Programme / Course	Bachelor of Computer Science
Title of Final Year Project	Lecture Digest – Auto Summarization and Key Points Recognition

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 8 </u> % Similarity by source Internet Sources: <u> 5 </u> % Publications: <u> 3 </u> % Student Papers: <u> 3 </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

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

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

Signature of Supervisor

Name: Dr Aun Yichiet

Date: 26/4/2024

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB01674
Student Name	Melissa Yap Chia Chean
Supervisor Name	Dr Aun Yichiet

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

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

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

(Signature of Student)

Date: 26/4/2024