UTARGPT

By

Loh Jun Xiang

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)

Jun 2024

DECLARATION OF ORIGINALITY

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: UTARGPT_- A university chatbot

Academic Session: _202406

I _LOH JUN XIANG

(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:
_1178, Jalan Seksyen 1/2, 31907,
Kampar

_Dr. Aun Yichiet
Supervisor's name

Date: __11/9/2024___________________ **Date**: ___11/9/2024___________

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/INSTITUTE* OF ___Faculty of Information and Communication Technology____

UNIVERSITI TUNKU ABDUL RAHMAN

Date: _11/9/2024_________________

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ______***Loh Jun Xiang***___________________________ (ID No: __***2002834***__) has completed this final year project entitled "*UTARGPT – A university chatbot* _" under the supervision of __Dr. Aun Yichiet_______ (Supervisor) from the Department of Computer and Communication Technology_, Faculty/Institute* of _Faculty of Information and Communication.

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

Yours truly,

__Loh Jun Xiang__________________

DECLARATION OF ORIGINALITY

I declare that this report entitled "**UTARGPT**" 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 : _Loh Jun Xiang__________________________

Date : _11/9/2024__________________________

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Aun Yichiet who has given me this bright opportunity to engage in an chatbot development project. It is my first step to establish a career in the chatbot development field. A million thanks to you.

To a very special person in my life, Lim Chooy Foon, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

The UTARGPT project introduces a highly innovative solution in the form of a chatbot designed specifically to serve the students and staff of UTAR by centralizing scattered university-related information into a single, accessible point. This significantly simplifies the process of accessing vital university details, which are currently distributed across numerous online platforms, making it difficult for users to find the information they need quickly and efficiently. To streamline this access, we employ a sophisticated web crawler to gather the latest updates and information from the UTAR websites. This web crawler is an essential component of the system as it automates the collection and assembly of data into PDF documents. These documents are subsequently uploaded into a GPT-based knowledge base, which the chatbot utilizes to provide responses to user inquiries. Ensuring the chatbot remains accurate and up-to-date with the latest information is critical. To manage this, a cron job scheduler has been integrated into the system. This scheduler regularly triggers the web crawler to perform data updates, thereby maintaining the chatbot's relevancy and effectiveness in delivering the most current information available to its users. Furthermore, the UTARGPT chatbot is designed to offer a personalized experience for each user. This personalized approach enhances the user experience by making the information not only more accessible but also more applicable to each student's unique situation. In addition to UTARGPT, I will develop OpenAI Assistant chat integrated with RAG, . This parallel development allows the team to compare the performance and user experiences between these two chatbot. Through this comparative analysis, insights gained will be used to continually refine and enhance both platforms. The development of these two different systems ensures a broad evaluation of potential solutions, aiming to optimize the way students and staff interact with university information, ultimately improving the overall efficiency and user satisfaction.

TABLE OF CONTENTS

TITLE PAGE	**i**
DECLARATION OF ORIGINALITY	**ii**
ACKNOWLEDGEMENTS	**iii**
ABSTRACT	**iv**
TABLE OF CONTENTS	**vi**
LIST OF FIGURES	**x**
LIST OF TABLES	**xi**
LIST OF ABBREVIATIONS	**xiii**
CHAPTER 1 INTRODUCTION	**1**
1.1 Introduction	1
1.2 Problem Statement	2-3
1.3 Motivation	4
1.4 Project Scope	5
1.5 Objective	6-8
1.6 Contributions	9-10
1.7 Background Information	11-12
1.8 Report Organization	13
CHAPTER 2 LITERATURE REVIEW	**14**
2.1 Previous Works	
2.1.1 Review of ChatGPT	14-19
2.1.2 Review of HuggingChat	20-22
2.1.3 Review of Bard	23-24
2.1.4 Review of GPT builder	25-27
2.1.5 Enhancing Chatbot Efficiency through RAG	28
2.1.6 Review of method to integrate RAG to LLM	29-31
2.2 Critical remarks of previous work	32-33
CHAPTER 3 System Methodology	**34**
3.1Methodologies	34-35

3.2 System Design	36
3.2.1 Hardware involved	36
3.2.2 Software Involved	36
3.2.3 Technology involved	37-39
3.2.4 System Design Diagram	
3.3 User requirements	40
3.4 timeline	41-42
CHAPTER 4 System Design	**37**
4.1 System Block Diagram	**43**
4.2 Interface Design	**44**
CHAPTER 5 System Implementation	**45**
5.1 Hardware setup	45
5.2 Software setup	46
5.3 Library setup	47-48
5.4 Preliminary work	49-63
5.5 Preliminary Work Result	64-72
5.6 System operation	73-74
5.7 Implementation issues and challenges	75
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	76
6.1 System Testing and Performance Metrics	
6.1.1 Compute similarity index	76-78
6.1.2 A/B testing	79-83
6.2 Project Challenges	84
6.3 Objective Evaluation	85
6.4 Concluding Remark	86
Chapter 7: Conclusion and Recommendation	87
7.1 Conclusion	87
7.2 Recommendation	88
References	89-90
A.1 Weekly Report	A-1
A.2 Poster	A-2

PLAGIARISM CHECK RESULT

CHECK LISTS

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.1.1	ChatGPT launching page	14
Figure 2.1.1.2	Login Page of ChatGPT	15
Figure 2.1.1.3	Sign Up Page of ChatGPT	16
Figure 2.1.1.4	ChatGPT UI	17
Figure 2.1.1.5	ChatGPT sharing chat page	18
Figure 2.1.2.1	HuggingChat UI	20
Figure 2.1.2.2	HuggingChat model types	21
Figure 2.1.3.1	Bard UI	23
Figure 2.1.3.2	Draft generated by Bard	23
Figure 2.1.4.1	Layout of GPT builder	25
Figure 2.1.4.2	Configure page	26
Figure 2.1.4.3	Configure page	26
Figure 2.1.6.1	Workflow of RAG Enhanced Chatbot	29
Figure 2.1.6.2	Code snippet to generate embeddings	29
Figure 2.1.6.3	Code snippet to create, save and load a FAISS store	30
Figure 2.1.6.4	Code snippet to create OpenAI instance	30
Figure 2.1.6.5	Code snippet to classify a user's query into one of the predefined topics	31
Figure 2.1.6.6	Code snippets to define a ChatAgent class that uses a large language model (LLM) to provide personalized	31
	book and movie recommendations in a conversational setting	31
Figure 3.1.1	Agile development methodology	35
Figure 3.2.3.1	UTARGPT System Design Diagram	38
Figure 3.2.3.2	RAG System Design Diagram	39
Figure 3.2.3.3	RAG Chatbot Design Diagram	40
	System Block Diagram	40
Figure 4.1.1	Chatbot interface design	44

Figure 5.4.1 Data that can be downloaded from the websites 51
Figure 5.4.2 Training data that need to be extracted from websites using web scraping 52
Figure 5.4.3 Required packages to perform web scraping 52
Figure 5.4.4 Required library to perform web scraping 53
Figure 5.4.5 Web scraping code snippet 53
Figure 5.4.6 Generate PDF function 53
Figure 5.4.7 Search URL from PDF function 53
Figure 5.4.8 Code snippet to extract URL from the file 54
Figure 5.4.9 Extract image from PDF function 54
Figure 5.4.10 Code snippet to extract image from PDF 54
Figure 5.4.2.1 Academic Calendar Year 2024 Image to text 55
Figure 5.4.3.1 GPT creation page 56
Figure 5.4.3.2 Knowledge section of chatbot 57
Figure 5.4.3.2 Preview section of chatbot 57
Figure 5.4.4.1 API key 58
Figure 5.4.5.1 Remove header function 58
Figure 5.4.5.2 Example of header 58
Figure 5.4.5.3 Code snippet to remove unwanted content 59
Figure 5.4.5.4 Code snippet to remove footer 59
Figure 5.4.5.5 Code snippet to replace the unstructured sentences to structure sentences 59
Figure 5.4.5.6 Example of unstructured sentences
Figure 5.4.6.1 Code snippet Load the data into a list inside a dictionary 60
Figure 5.4.6.2 Code snippet to store the list into variable for data processing purposes 61
Figure 5.4.6.3 Code snippet to split the text into smaller chunk size
Figure 5.4.6.4 Code snippet to create embedder 61
Figure 5.4.6.5 Code snippet to store the embedded files into FAISS vector store
Figure 5.4.6.6 Code snippet to create RAG agent 62
Figure 5.4.6.7 Code snippet to launch the chatbot in Gradio interface UTARGPT share link 63
Figure 5.5.1.1 UTARGPT User Interface 63
Figure 5.5.1.2 UTARGPT refined query and output Code snippet to compute the similarity index
Figure 5.5.1.3 Similarity index of Question: What is the objective of RSS? 64

Figure 6.1.1.1	Similarity index of Question: Core and elective courses	65
Figure 6.1.1.2	Similarity index of Question: Dr Aun Yichiet room number	73
Figure 6.1.1.3	Similarity index of Question: calculation of GPA	73
Figure 6.1.1.4	Similarity index of Question: minimum entry requirement in Foundation	73
Figure 6.1.1.5	Similarity index of Question: courses excluded from calculation of GPA and CGPA	77
Figure 6.1.1.6	Similarity index of Question: CGPA range for Degree with Distinction, Degree with Merit and regular Degree	77
	Similarity index of Question: Who are the HODs of FICT?	77
Figure 6.1.1.7	Number of respondents in groups A and B	78
Figure 6.1.1.8	Percentage breakdown of Question 1	78
Figure 6.1.1.9	Percentage breakdown of Question 2	78
Figure 6.1.2.1	Percentage breakdown of Question 3	78
Figure 6.1.2.2	Percentage breakdown of Question 4	
Figure 6.1.2.3		79
Figure 6.1.2.4	Percentage breakdown of Question 5	79
Figure 6.1.2.5	Percentage breakdown of Question 1	80
Figure 6.1.2.6	Percentage breakdown of Question 2	80
Figure 6.1.2.7		81
Figure 6.1.2.8	Percentage breakdown of Question 3	81
Figure 6.1.2.9	Percentage breakdown of Question 4	82
Figure 6.1.2.10	Percentage breakdown of Question 5	82
Figure 6.1.2.11		83

LIST OF TABLES

Table Number	Title	Page
Table 2.2	Comparison between existing chatbots	32
Table 3.2.1.1	Hardware involved table	37
Table 3.2.2.1	Software involved table	37
Table 3.3.1	User requirement	41
Table 5.1 .1	Hardware setup	47
Table 5.5.1	List of questions	67
Table 5.5.2	Comparison between ChatGPT and UTARGPT	67-72

LIST OF ABBREVIATIONS

AI	Artificial Intellingence
NLP	Natural Language Processing
OCR	Optical Character Recognition
GPT	Generative Pre-trained Transformer
RAG	Retrieval-Augmented Generation

CHAPTER 1

Introduction

1.1 Introduction

In this era of technology, students often rely on the Internet to get the information they need from the school. Every university will have its own website to put all the important announcements to their students. All the students will then find the relevant information from the university websites. This project is mainly about building a centralized chatbot that consists of the information from all sites of the school to allow the students to retrieve the information they are interested in a faster manner.

Chatbot is basically a computer program that can generate human-like conversations to human end-users. Chatbot has many usages such as acting as a chatter to chat with human, answering questions from users and etc. There are three types of chatbots which are ruled-based bots, an AI bot and an application-oriented bot that combined both rule-based and intelligent dialogue systems.[1] Rule-based chatbots are based on predefined rules and the chatbots will answer accordingly. If the users ask a question that is out of scope, the chatbots are unable to respond correctly. AI chatbots are built using machine learning and natural language.[1] It is trained with massive training datasets so it can answer most of the questions asked by the users. [1]

The chatbots built in the project is using GPT builder which is a new tool just launched in 2023 by OpenAI. It is a tool designed to help developers create, test, and deploy custom AI models based on the GPT architecture. Since it is a pre-trained model, the users can easily build the chatbot by having little programming skill, so it allows the users to build a high-quality chatbot within a short period of time. AI chatbots can handle users' questions more efficiently. Unlike the rule-based chatbots, AI chatbots do not have so much restriction to their users. The users can ask any question they want. Besides that, AI chatbots can have a coherent response by referring to the previous question. This feature enhanced the user experience as they did not have to retype the question again when the follow-up question was related to the previous question.

In this project I build two chatbots which are GPT Builder UTARGPT and OpenAI Assistant chatbot with RAG system to compare their performance.

1.2 Problem Statement

1.2.1 Difficult in locating the relevant information

As mentioned above, the university's official websites consist of the university information such as the information about the university, program offered, faculty information and etc. While for the announcement announced by the university will be sent through email to students. For the subjects taken by the students, the announcement by the lecturer to the students will be posted on WBLE. Furthermore, information such as examination timetable, academic calendar and etc is placed in the UTAR portal. There are various types of places that store different information. For the freshman who just comes to UTAR, it will be very hard for them to find for the relevant information as they are still not familiar where the information is allocated. For existing students, they will not have problem in finding the relevant information but the process of searching relevant information will be faster by using a chatbots.

1.2.2 UTAR lacks a sophisticated chatbot system

UTAR is currently lacking a sophisticated chatbot system capable of addressing the diverse and complex information needs of students and staff efficiently. At present, UTAR relies on manual responses from staff through mediums like WhatsApp, phone calls, and emails. This process is not time-effective, especially when urgent queries arise outside working hours, leaving users without timely responses. Furthermore, staff members often need to manually search through UTAR's website to find relevant information, which can lead to delays or inaccurate responses being provided. This not only hinders operational efficiency but also diminishes the user experience for students and staff seeking quick, accurate, and contextually relevant answers to their queries. The implementation of a more advanced AI-powered chatbot integrated with RAG would significantly enhance the university’s capacity to streamline information retrieval, ensuring timely, precise responses and reducing the dependency on manual processes.

1.2.3 Risk of Students Accessing Outdated Information

The university sites contain a lot of outdated information, such as the faculty handbook, which is not updated very often. This can cause students to rely on inaccurate information, affecting them to choose the course, academic pathway and etc. Therefore, if the student accesses all these pieces of information from the websites, they will end up obtaining the incorrect answer. The lack of consistent updates across various sections of the site increases the risk of false information, which may hinder academic development. The implementation of RAG chatbot in the school sites can perfectly prohibited this issue as the chatbot will be trained with the latest information. If the similar pieces of data being found by the chatbot, it has the priority to choose the content from the most recent file.

1.3 Motivation

Motivated by the need for a streamlined information retrieval system at UTAR, I aim to develop a chatbot to assist future students. As a student usually will get frustrated when they are unable to search for the information especially when they need it in urge. The primary goal is to offer a quick and easy way to get university information without sifting through multiple websites. By simply asking the chatbot, students can receive accurate answers instantly, significantly cutting down the time usually spent searching for information. By leveraging the techniques of Natural Language Processing, the chatbot is able to generate human-like text to the students with high precision.

Additionally, having a centralized platform consolidates all university announcements into a single, easily accessible location. This feature eliminates the need for students to visit various websites to gather important information and updates. This is particularly beneficial for new students unfamiliar with the university's information structure. The freshmen usually face difficulty in searching for the correct information from the correct location. By providing a centralized platform which all the school-related information is readily available, the chatbot not only enhances accessibility, but also lowers the learning curve for the new students to join the community. By creating a more friendly and encouraging environment, this strategy helps students adjust to their new academic setting more swiftly and easily.

The chatbot will provide more accurate answers to the users as compared to the other chatbots and the traditional communication methods like manual replies through WhatsApp, phone calls, and email. Unlike human replies, which may be slow and inconsistent, especially outside working hours, leading to delats in providing critical information. In contrast, the chatbot can provide instant responses to the users based on their knowledge. Unlike other chatbots, our chatbot is trained with customized knowledge that specifically focus on the UTAR information and is integrated with RAG system, making it outperform than other chatbot. This make it a more reliable and efficient tool for both students and staff, reducing false information provided to the students and staff.

1.4 Project Scope

In this project, we focus on creating a generative AI chatbot using OpenAI Assistant for UTAR students. Hosted on the Gradio website, students can access the chatbot with the share link. Since two chatbot has been built in this project which are custom chatbot from GPT builder and the OpenAI Assistant chatbot integrated with RAG system, we will compare the performance of these two chatbots and choose the best chatbot to deploy. The initial step is to gather custom datasets to train the model, requiring data collection from various UTAR sources, including the UTAR portal, official websites, FAQs, and handbooks. We employ web scraping for data retrieval. After collecting the data, we preprocess and normalize it to ensure compatibility with the GPT builder's training engine. This stage is crucial as it involves transforming all content, such as extracting text from images using computer vision, into a format the AI can understand. The next phase involves building and fine-tuning the chatbot with these datasets. While GPT builder simplifies the training process, considerable testing and fine-tuning are necessary to refine the chatbot’s accuracy. Up to this step, we have successfully configured the GPT builder custom chatbot. The same data will be used in building the RAG system and OpenAI Assistant chatbot. When building the RAG system, the PDF file need to be preprocessed again such as normalize the data, remove unwanted content. The preprocessed PDF is then tokenized and stored in the vector store created using FAISS. After that, we will need to embed the preprocessed data. Next, we will require to integrate the RAG with the OpenAI Assistant chatbot. The reason of using OpenAI Assistant chatbot instead of GPT builder custom chatbot is that the Assistant chatbot provides the API for integration purposes while GPT builder custom chatbot does not. After being integrated with the OpenAI Assistant chatbot, it will be hosted in the Gradio interface. The user can use the chatbot via the share link provided in Gradio.

1.5 Project Objectives

The development of the chatbots will be based on several objectives. All these objectives will be achieved in this project. I will list out all the objectives and further explain the objectives below:

1.5.1 to develop a customGPT for a context aware UTAR chatbot

Our main objective of this project is to develop a customGPT or custom chatbot that facilitate the process of retrieving the information from different sites by the students. As mentioned above, there are too many sites from the university that allow the students to retrieve useful information. The university is lack of a centralized site to allow the students to get the information. The chatbot will be first trained with the information from those sites, which are the training datasets of our project. After it is successfully fine-tuned, the students can use this chatbot to obtain all the useful information from the university. To make sure the chatbot is convenient and user-friendly, context-aware is important as it can remember the previous interaction and relevant details, making the interaction more personal and customized. For instances, users don't have to repeat themselves when asking follow-up questions, resulting in a smoother, more conversational experience.

1.5.2 To develop a web scrapper tool to collect the data from the UTAR websites

In order to obtain the information from the UTAR websites as the training data for the chatbot to learn, web scrapping was performed to scrap the content of the UTAR websites. To achieve this objective, BeautifulSoap was used to perform the web scraping task. It is a useful library available in Python for users to easily perform web scraping. Basic knowledge of HTML is required if we only want to scrap part of the content from the websites. After we scrapped the content from the websites, we will need to compound all the content from the same websites into one PDF file since the GPT builder only accepts file type input data. To ease this process, a file generate function was written by using Python in the Jupyter Notebook to allow us to reuse the code by passing the list that contains all the content scrapped from the websites and the path of the generated file to be stored as the function parameter. After the files are generated, we will need to perform data normalization as the GPT builder only accepts text-type input data. Therefore, if the file contains the image-type data, we will need to perform text extraction to extract the text from the image and remove the unnecessary image from the file. A function is written to perform the task as well. After the data normalization process, a new PDF file with all the text form data will be generated and the content of the new PDF file can successfully read as the knowledge of the chatbot.

1.5.3 To preprocess, normalize and clean the dataset like segmentation, merging

After we performed web scrapping to obtain the data from the UTAR websites and generate it as a PDF file, we will need to further preprocess the file obtained. Various type of data processing method has been used here, which are data cleaning, segmentation, and data normalization. The images from the file will be extracted out and extract the text inside it since both the GPT builder chatbot and OpenAI Assistant chatbot can only accept text-based input knowledge. We will need to normalize the extracted text. When building the RAG system, extra data preprocessing steps would be required such as data segmentation to sectionize the content of the file, and data cleaning which removes the footer and header of the file to save embedded tokens, All these preprocessing steps are important to improve the quality of the retrieved documents. Some functions has been written in order to make the data-cleaning process simple and easy.

1.5.4 To implement Retrieval-Augmented Generation(RAG) for fast information retrieval from large source

Since there are a lot of sources to be obtained from the university sites and fed to the chatbot as the knowledge, therefore the accuracy and efficiency will be affected as the increase of files fed to the chatbot, and when we reached the maximum file limits, what we have to do is combining the existing file into single file. Therefore, when we train the chatbot, there will be some files consist the knowledge from different files that might not be relevant to each other. Therefore, the implementation of Retrieval-Augmented Generation (RAG) into the chatbot is playing a significant role here, as the RAG integrates a retriever that pulls relevant documents from vast datasets. The RAG will refine the query based on the consumed data and perform a knowledge searching to find for the relevant segment of data and the generator which is our chatbot will use the retrieved information to produce coherent and contextually relevant responses.

1.5.5 To package the chatbot into API for fast deployment

Since the target users of the chatbot are the students and lecturers from the University, therefore it is convenient if the chatbot is directly deployed into the UTAR main websites, so if anyone wants to ask the questions relevant to the University, they can easily get the chatbot from the UTAR websites without needing to installing any software. By encapsulating the chatbot's functionality within an API, developers can seamlessly access its features, ensuring efficient communication and response handling without needing to embed complex code into each project. This approach simplifies deployment, enhances maintainability, and enables faster updates, making the chatbot more adaptable to evolving user needs and environments. The chatbot offers the API function to allow the developers to directly integrate the chatbot into their websites seamlessly.

1.6 Contribution

The development of AI chatbots is inevitable for all the students in UTAR as there are many students who need to access information from UTAR websites every day. This chatbot can facilitate the process of retrieving the information online.

A customize chatbots that is specifically designed for UTAR students to retrieve the internal information from UTAR can greatly reduce the time taken of the process of retrieving the correct information. With this chatbots, the students do not have to browse to all the websites again to get the information they want. Traditionally, student would have to go through numerous websites to obtain the necessary information such as courses, examination schedules and etc. This is not only consuming valuable time but can also be a source of frustration, especially during the critical period like exams. Through the utilization of a chatbot designed specifically for UTAR, students can completely avoid these obstacles. They only required to enter the information they want in question form in the chatbots and the chatbots will generate correct responses to them. This eases the process of getting the correct information for UTAR students.

Besides that, the AI chatbots allowed the students to get relevant information from only one place. UTAR lacks a centralized platform to store all the university-related information. Therefore, the students have to retrieve different information from different websites previously. This is not friendly to the freshmen and also those existing students that are seldom look for the announcements from the university. With AI chatbots, the process of getting different types of information can be eased. This is due to the students do not have to browse to different websites again, but they only required to use the AI chatbots. The AI chatbots act as a center that gathers all the information from all the websites and generates responses to the students accordingly. In the future, the maintenance team can update the latest knowledge in the chatbot knowledge. This is easier to manage by the school IT department as the websites now are managed by different IT departments in the school and the websites from different sites are not developed by the same people. With a centralized platform, the school related information can be managed and update at one site which is a more cost-effective way.

Furthermore, the AI chatbot reduced the risk of students accessing outdated information. Before the invention of the AI chatbot, students needs to manually search for information from websites while the websites contained a lot of outdated information. However, with the invention of the AI chatbot, we can greatly reduce this issue as the AI chatbot is being trained with up-to-date data, and priority to the correct file has been set if the both the files contains the similar content, we will always get the information from the latest file. For instance, both the FICT handbook and the staff directory files contain the information about the HODs of FICT, but the FICT is the outdated one since it never update since 2020. The chatbot will give higher priority to the staff directory file than the FICT handbook as the staff directory file is up-to-date.

Moreover, the chatbot developed in the project is not the typical LLM chatbot, we integrated the LLM chatbot with the RAG system in order to increase the performance of the chatbot. The RAG chatbot provided better output as the RAG will use the query from the users to perform a similarity search first to retrieve the relevant document and perform query refinement on the users' query. Then the refined query and the retrieved document will be sent to the LLM which in this case is the Assistant chatbot, the Assistant chatbot will generate the output based on the retrieved document and refined query.

Last but not least, the chatbot offers the API for fast deployment. By incorporating an API, developers can easily integrate seamlessly into their application or websites. Therefore, this feature allows the chatbot to be embedded in the UTAR main website, making it easily accessible to students, staff, and visitors. It ensures that users can access the chatbot from any device, at any time, without the need for separate software, significantly enhancing its usability and reach across the university's digital ecosystem.

1.7 Background Information

Chatbot is one of the most popular artificial intelligence (AI) technologies nowadays. It can be applied in various field such as education, customer service, IT helpdesk and more. According to BotCore, 80% of businesses required chatbots to operate their business by 2020 [8]. This is due to its key function which provides immediate response to the customer, the customer does not need to wait for a long time for the support agent to come online.

ELIZA is one of the earliest examples of a chatbot invented in 1964 at MIT by Joseph Weizenbaum [9]. It utilized the technique of Natural Language Processing which allow the computer to understand the human language. Soon after, PARRY, a smarter version of ELIZA was invented as it can understand the input from a real person by Psychiatrist Kenneth Colby[8]. The first and most famous chatbot that was being used at that time is known as Alice as it was the most powerful AI chatbot at the time and was awarded the Loebner Prize thrice.

As mentioned above, the AI chatbot is applicable to various field as it can bring significant benefits to the users. The intersection of AI chatbots and educational technology has led to more creative solutions aimed at enhancing the learning process and operational efficiency within the academic institution. The chatbots play an important role which serving as a virtual assistants to clarify and solve question from the students. The UTAR GPT make use of the most recent development, GPT builder which is using the Generative Pre-trained Transformer (GPT) technology, developed by OpenAI, to provide the UTAR student a customized chatbot to retrieve the information from the University.

Before the GPT and related technologies were developed, the developers had to build the NLP chatbot from scratch which is a big challenge and time-consuming as a large amount of time is required to train and fine-tune the chatbot. The chatbot were often unable to handle unexpected questions or stray from their predefined routes since they mostly relied on pre-written scripts. Therefore, this frequently resulted in a frustrating user experience. The invention of the GPT builder has significantly increased the capabilities of these tools. The chatbots can be built within a short period of time with high response accuracy. They can now provide a personalized response to the users.

A few fundamental phrases and concepts are necessary for readers who are not familiar with the technical fundamentals of GPT and chatbot technology to fully understand the UTAR GPT project. Firstly, Natural Language Processing (NLP) refers to the branch of AI concerned with the interaction between computers and humans using natural language. It gives computers the ability to interpret the human language. Besides that, the GPT is a powerful pre-trained model that has been trained with massive datasets before being fine-tuned for a specific task such as responding to queries from the students. The GPT builder is a tool developed by OpenAI specifically to build a chatbot. It used the pre-trained GPT to minimize the training steps and fine-tuning steps of developing a customize chatbot. By uploading the required data as the knowledge of the chatbot, the chatbot will has the high performance and accuracy in answering the questions from the users. Since it is using the pre-trained GPT model from OpenAI, it has good performance in handling unexpected question from the users.

1.9 Report Organization

The details of this project are shown in the following chapters. In Chapter 2, some related existing systems are reviewed in this project. In Chapter 3, the system methodology will be shown. In Chapter 4, the system design of UTARGPT will be shown. In Chapter 5, system implementation CHAPTER 1 27 will be shown. In Chapter 6, system evaluation will be shown. In Chapter 7, the conclusion and the recommendation for further improvement will be shown.

CHAPTER 2

Literature Review

2.1 Previous Works

2.1.1 Review of ChatGPT

ChatGPT (Figure 2.0) is a powerful AI chatbot developed by OpenAI and published in November 2022. It is developed based on the GPT-3.0 architecture. It is using a text-davinci-003 model which is one of the most capable GPT -3 models. ChatGPT is a generative model that has been trained on massive training data from various sources such as online resources, articles and etc. ChatGPT uses deep learning to produce humanlike text through transformer neural networks. The transformer can predict text based on its training data's normal sequence.

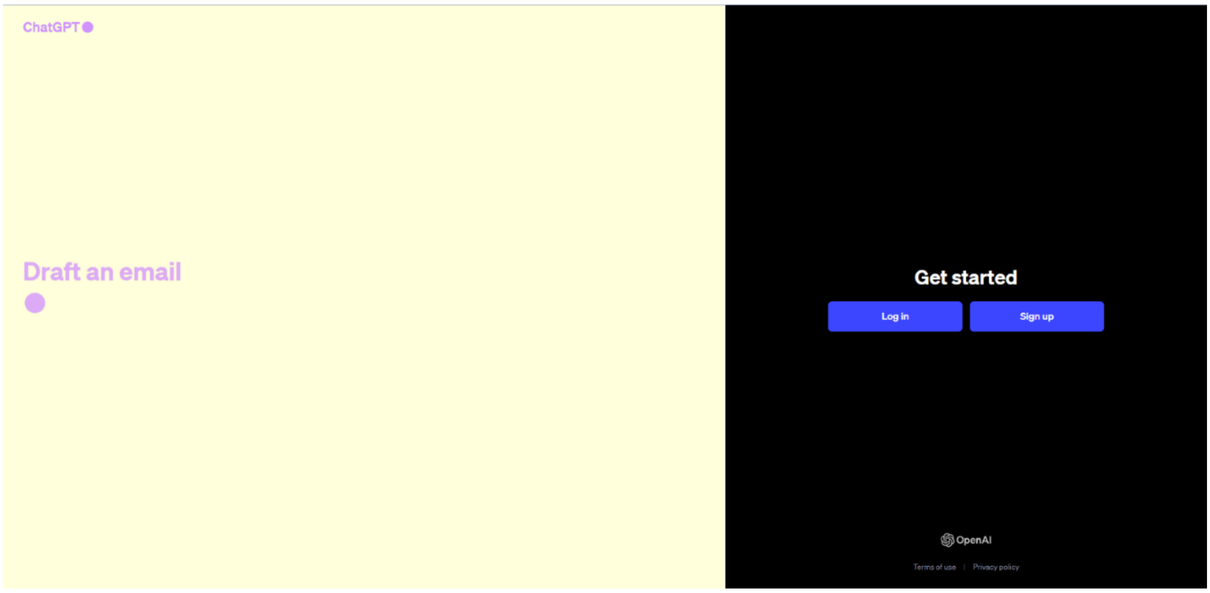
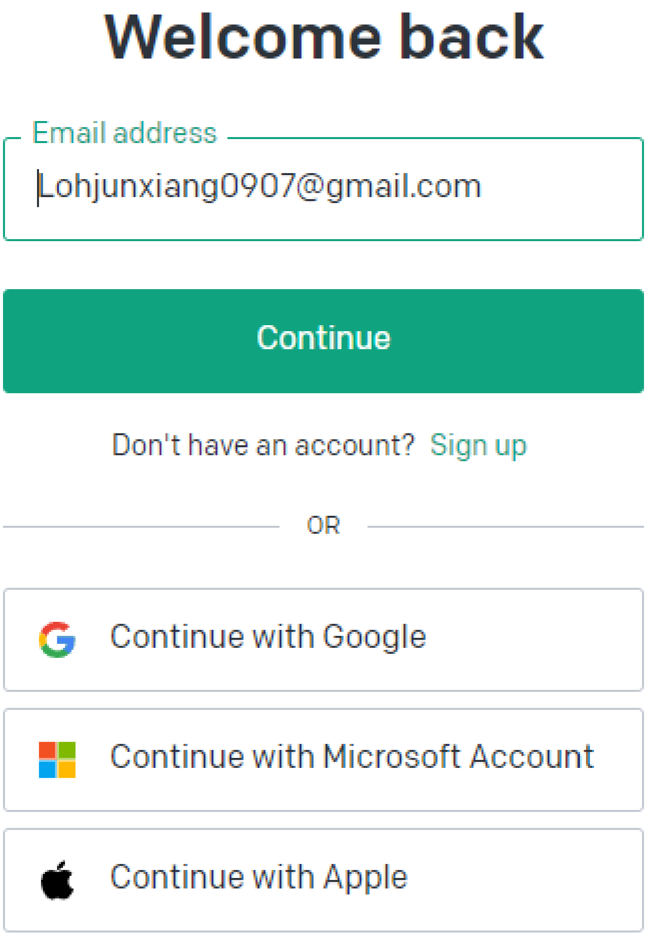


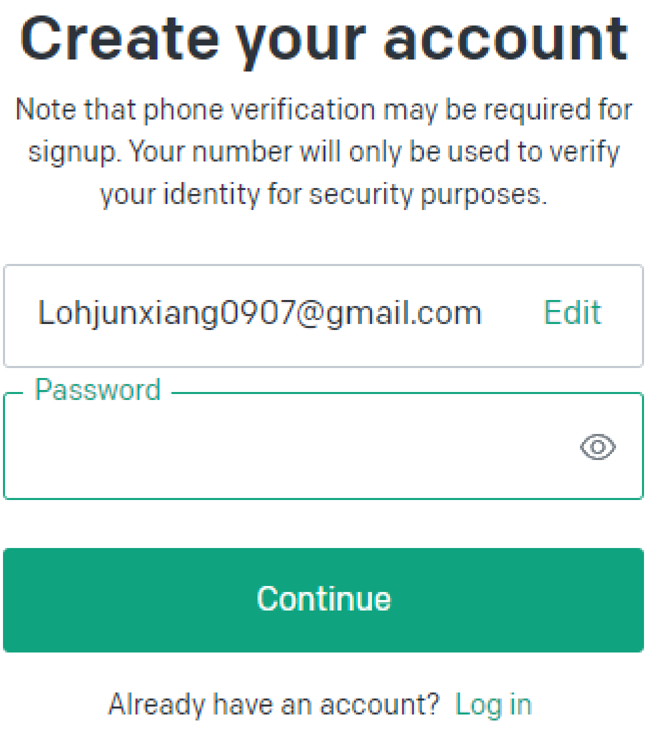
Figure 2.1.1.1 ChatGPT launching page

ChatGPT required the users to sign up their account before they can use it. The login page is shown in Figure 2.0.1. There are a total of 4 methods to sign in which are sign in with Google account, Microsoft account, Apple Account or registered their account in ChatGPT by filling up their email address and password as shown in Figure 2.0.2.



Lohjunxiang0907@gmail.com

Figure 2.1.1.2 Login Page of ChatGPT



Create your account

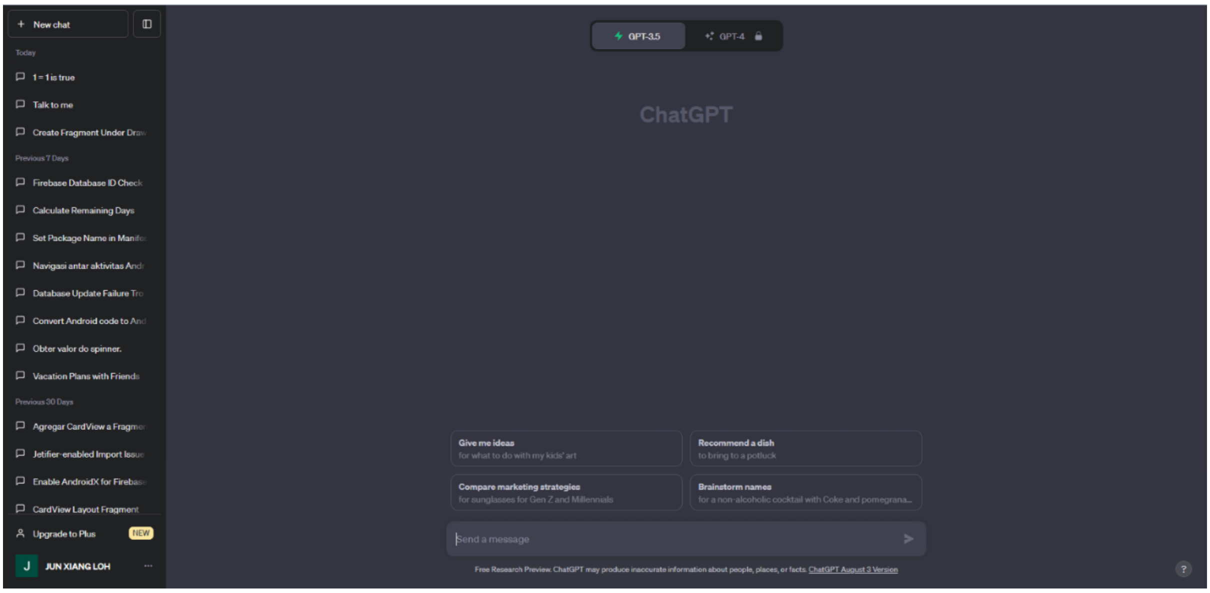
Note that phone verification may be required for signup. Your number will only be used to verify your identity for security purposes.

Lohjunxiang0907@gmail.com

Password

Figure 2.1.1.3 Sign Up Page of ChatGPT

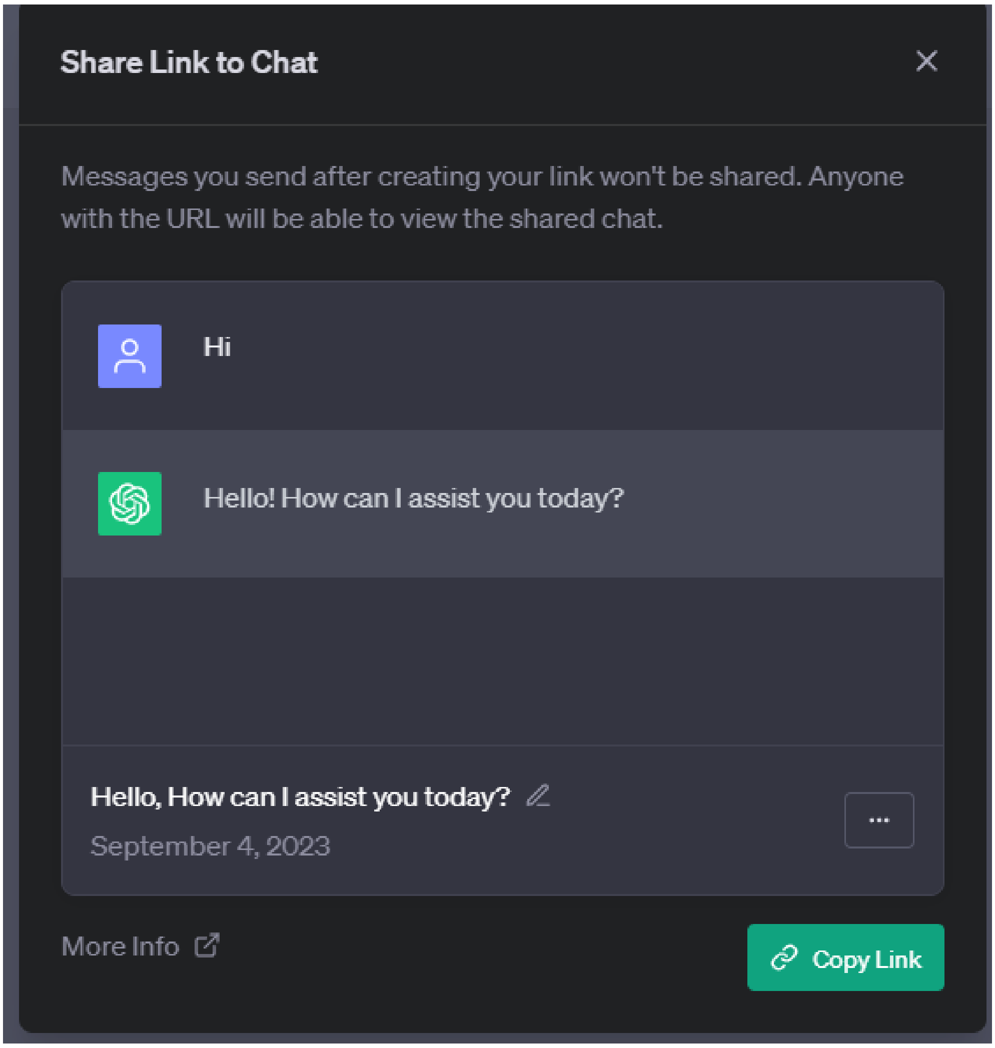
After the users have successfully logged into their account, they will see the page shown in Figure 2.0.3 which they are allowed to create a new chat to categorize their chat for easy revise and enter their questions in the input field. ChatGPT provided subscription plan for their users to unlock extra features which are GPT-4, faster response speed and access to beta features such as plugin and etc. The users are allowed to edit the title of the chat and delete the chat as well.


New chat
Today
1 = 1 is true
Talk to me
Create Fragment Under Draw
Previous 7 Days
Firebase Database ID Check
Calculate Remaining Days
Set Package Name in Manife
Navigasi antar aktivitas Andr
Database Update Failure Tro
Convert Android code to And
Obter valor do spinner.
Vacation Plans with Friends
Previous 30 Days
Agregar CardView a Fragme
Jetifier-enabled Import Issue
Enable AndroidX for Firebase
CardView Layout Fragment
Upgrade to Plus
NEW
J
JUN XIANG LOH
GPT-3.5
GPT-4
ChatGPT
Give me ideas
for what to do with my kids' art
Recommend a dish
to bring to a potluck
Compare marketing strategies
for sunglasses for Gen Z and Millennials
Brainstorm names
for a non-alcoholic cocktail with Coke and pomegrana...
Send a message
?

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure 2.1.1.4 ChatGPT UI

Besides that, the users can share their chat with others by sharing the link to them as shown in Figure 2.0.4. Anyone with the shared URL is able to view the shared chat. ChatGPT also supports many languages such as English, Chinese, Malay and etc. There is no language restriction in ChatGPT so the users from all over the world can use it with their favour language.



Share Link to Chat

Messages you send after creating your link won't be shared. Anyone with the URL will be able to view the shared chat.

Hi

Hello! How can I assist you today?

Hello, How can I assist you today?

September 4, 2023

Figure 2.1.1.5 ChatGPT sharing chat page

Furthermore, ChatGPT provided their API for the users to integrate into their own business or project. To have this feature, the users need to sign up for API access and follow the API ‘s pricing and usage policies.

Pros:

- Clean User Interface
- User-friendly
- Have API to integrate with other applications
- Can respond to a wide range of users' input
- Available all the time
- Can handle a large number of users simultaneously
- Support multiple languages

- Can create coherent responses

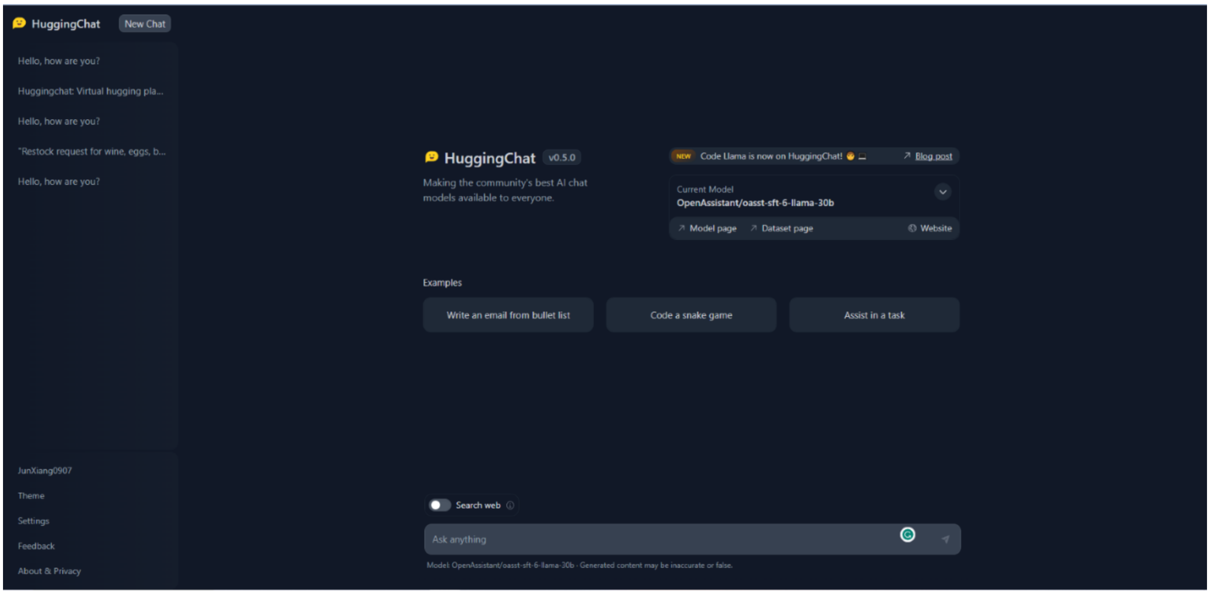
Cons:

- It takes a long times to generate responses when a lot of users are using it
- Knowledge is not updated in real-time
- Required large training datasets
- Integrations of API with other applications can incur high cost

2.1.2 Review of HuggingChat

HuggingChat is an open-source chatbot developed by HuggingFace in April 2023. It is using OpenAssistant LLaMa 30B SFT 6 model and Llama 2 models. HuggingChat provides users with accurate and useful information using a popular Python library known as Transformer. It is a conversational AI model which is designed to involve in conversation and answer questions. HuggingChat is not aimed to give creative answers, but to provide accurate and useful information to the user.

The UI design for HuggingChat is more or less the same as ChatGPT as shown in Figure 2.1 which is clean and easy to use. A chatbot should avoid itself to have bombastic UI, a simple and clean interface is more suitable and comfortable for users. The left side of the page contains the chat history. Users can add a new chat by clicking the "New Chat" button and delete or edit the title by clicking the respective button.



Hello, how are you?

Huggingchat: Virtual hugging pla...

Hello, how are you?

"Restock request for wine, eggs, b...

Hello, how are you?

Making the community's best AI chat models available to everyone.

Code Llama is now on HuggingChat!

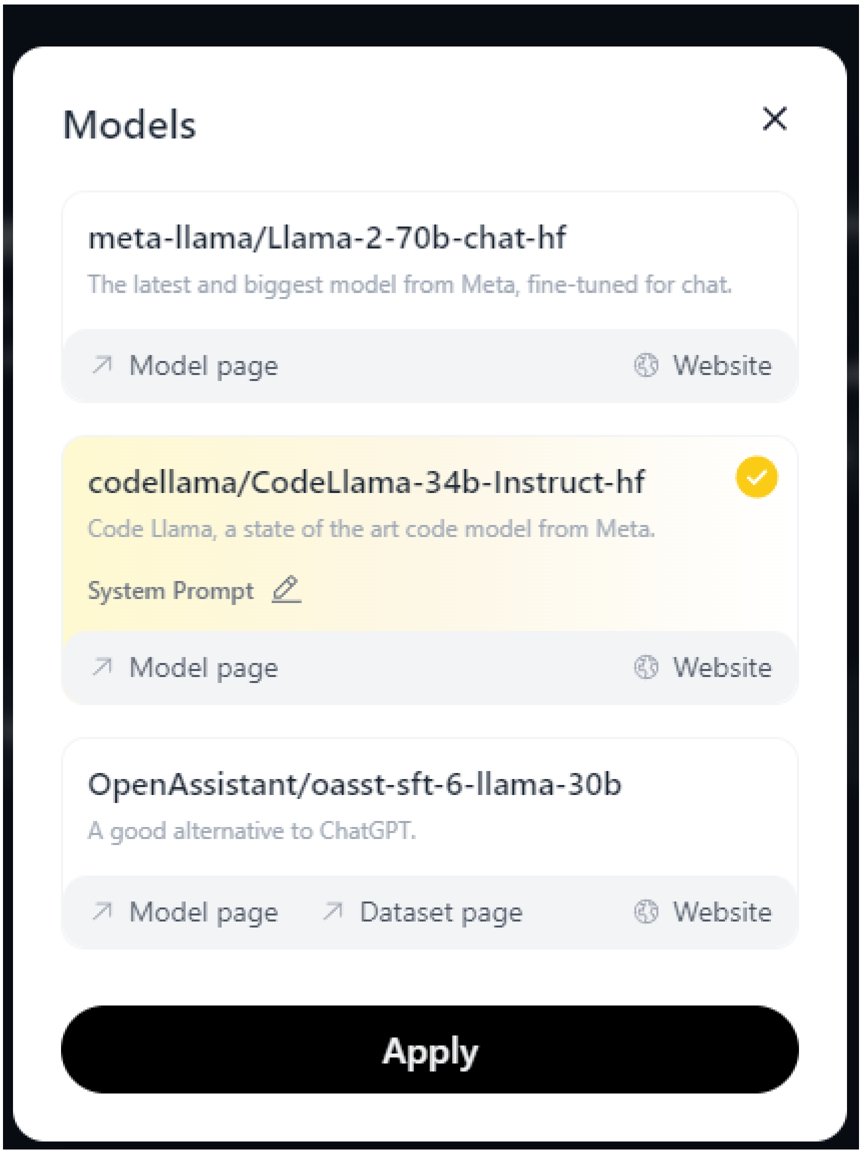
Current Model
OpenAssistant/oasst-sft-6-llama-30b

Ask anything

Model: OpenAssistant/oasst-sft-6-llama-30b · Generated content may be inaccurate or false.

Figure 2.1.2.1 HuggingChat UI

HuggingChat provided extra features for their users to choose from the models. (Figure 2.1.1) They are offering 3 models for different reasons and different purposes. Users can select the models based on their preferences and purposes. Besides that, it provides two response style to the users which are with search web and without search web. The search web functions will gather information from the Internet before generating the response. This increases the accuracy of the response. This feature is useful when the users are searching for information using the chatbot. However, enabled this function will require more time for the response to be generated.


Models

meta-llama/Llama-2-70b-chat-hf
The latest and biggest model from Meta, fine-tuned for chat.

codellama/CodeLlama-34b-Instruct-hf
Code Llama, a state of the art code model from Meta.

OpenAssistant/oasst-sft-6-llama-30b
A good alternative to ChatGPT.

Figure 2.1.2.2 HuggingChat model types

Besides that, HuggingChat provided API for the users to integrate into their own applications. Unlike ChatGPT, the API provided by HuggingChat is free. The users can just use the library “pip install hugface” to integrate the chatbot with their own application.

Pros:

- Clean User Interface
- User-friendly
- Have free API to integrate with other applications

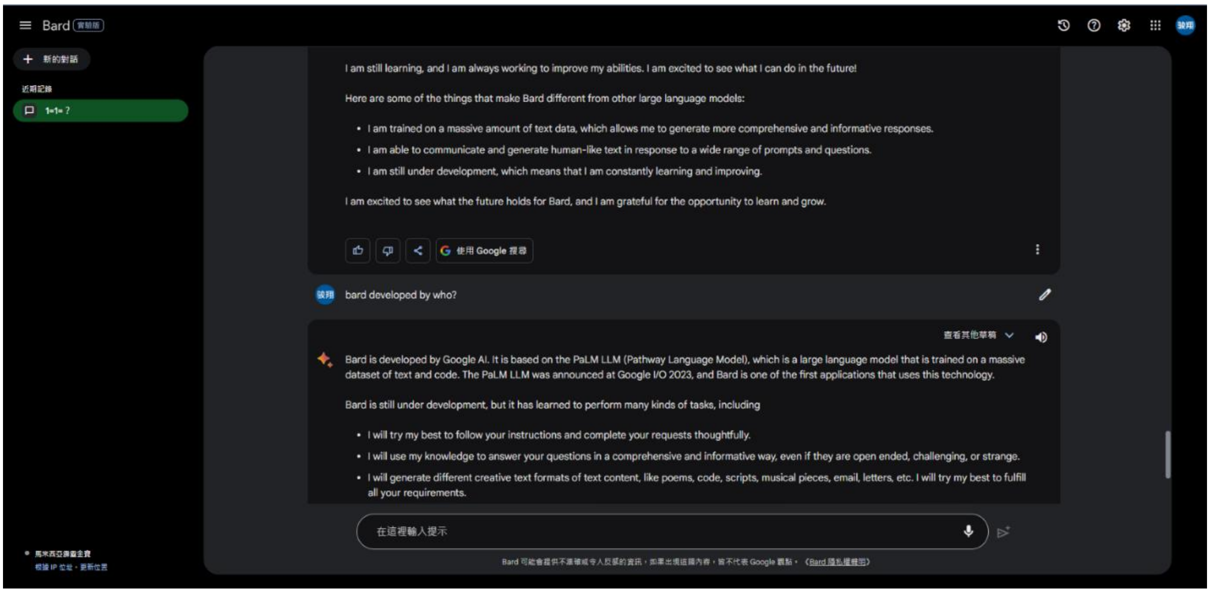
- Can respond to a wide range of users' input
- Available all the time
- Cannot create coherent responses
- Can handle a large number of users simultaneously
- Support multiple languages
- Privacy is protected as the users can choose to share conversations with authors or not
-

Cons:

- Take longer time to generate response especially search web function enabled
- Knowledge is not updated in real-time
- The sidebar cannot be adjusted
- Cannot edit the user input after the response generated

2.1.3 Review of Bard

Bard is a conversational AI chatbot developed by Google AI. It is developed based on the PaLM LLM, which is a large language model that is trained on massive dataset of text and code. It is still under development, therefore the response provided might not as accurate as other chatbots reviewed earlier. The UI design of Bard is shown in Figure 2.2. It provided simple and clean UI for the users.



I am still learning, and I am always working to improve my abilities. I am excited to see what I can do in the future!

Here are some of the things that make Bard different from other large language models:

- I am trained on a massive amount of text data, which allows me to generate more comprehensive and informative responses.
- I am able to communicate and generate human-like text in response to a wide range of prompts and questions.
- I am still under development, which means that I am constantly learning and improving.

I am excited to see what the future holds for Bard, and I am grateful for the opportunity to learn and grow.

bard developed by who?

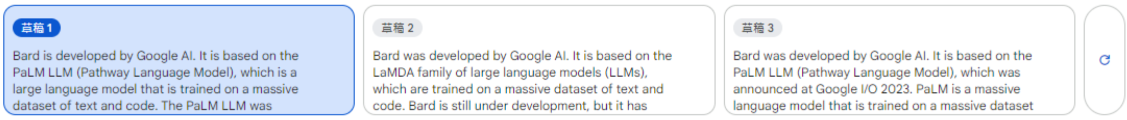
Bard is developed by Google AI. It is based on the PaLM LLM (Pathway Language Model), which is a large language model that is trained on a massive dataset of text and code. The PaLM LLM was announced at Google I/O 2023, and Bard is one of the first applications that uses this technology.

Bard is still under development, but it has learned to perform many kinds of tasks, including

- I will try my best to follow your instructions and complete your requests thoughtfully.
- I will use my knowledge to answer your questions in a comprehensive and informative way, even if they are open ended, challenging, or strange.
- I will generate different creative text formats of text content, like poems, code, scripts, musical pieces, email, letters, etc. I will try my best to fulfill all your requirements.

Figure 2.1.3.1 Bard UI

The sidebar at the left allows users to create a new chat and store the history of the chat. The chat can be deleted and edited by the respective button. The users can choose to pin the chat as well. Besides that, the users can edit their input after the response is generated. The response will regenerate after the user edits their input. If the generated response doesn’t really answer the users, the users can check for other drafts generated. (Figure 2.2.1) Therefore, the users do not need to regenerate again and again.



草稿 1

Bard is developed by Google AI. It is based on the PaLM LLM (Pathway Language Model), which is a large language model that is trained on a massive dataset of text and code. The PaLM LLM was

草稿 2

Bard was developed by Google AI. It is based on the LaMDA family of large language models (LLMs), which are trained on a massive dataset of text and code. Bard is still under development, but it has

草稿 3

Bard was developed by Google AI. It is based on the PaLM LLM (Pathway Language Model), which was announced at Google I/O 2023. PaLM is a massive language model that is trained on a massive dataset

Figure 2.1.3.2 Draft generated by Bard

Besides that, Bard provided another useful feature for the users to input which is the users can choose to input by text or voice. Some users may think that input with voice is more convenient. The generated responses can also be spoken out. Since Bard is still under development stage, it doesn’t provide API for the users but it will be available after the development is mature.

Furthermore, the users are allowed to share their chat section by section with others. Unlike ChatGPT which can share the whole chat history, Bard only allows them to send one question and response at one time. Bard supports multiple languages as well. Then, Bard also will suggest the related topics after the response is generated. The users can search using Google if they are not satisfied with the response generated.

Pros:

- Clean User Interface
- User-friendly
- Can respond to a wide range of users' input
- Available all the time
- Can create coherent responses
- Support multiple languages

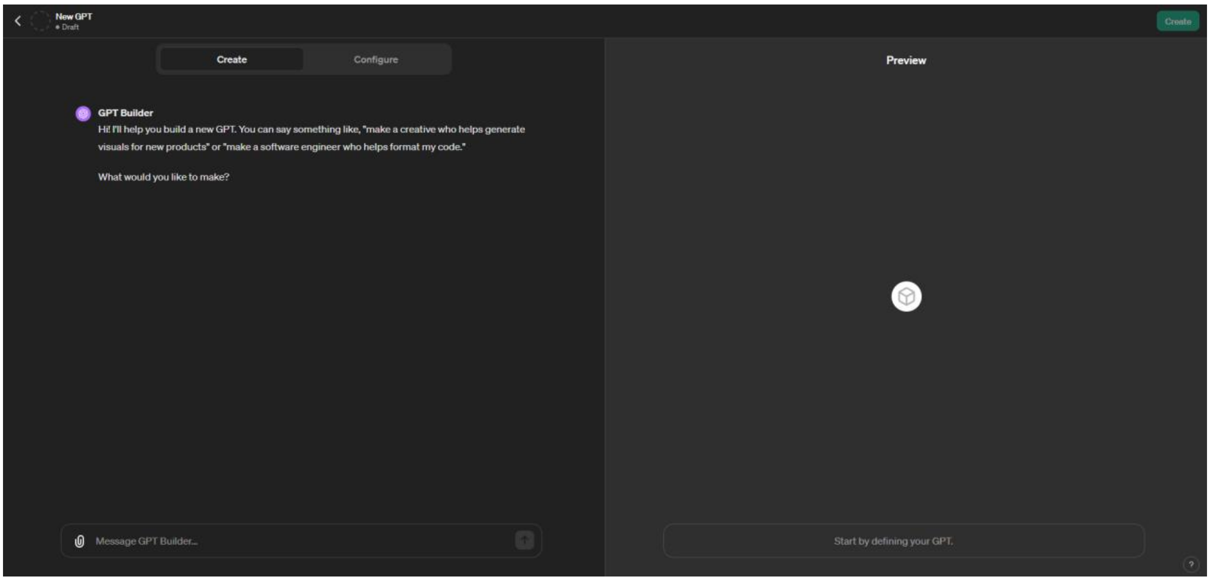
Cons:

- Response may not be accurate as it is still under the development stage
- No API provided

2.1.4 Review of GPT builder

GPT builder is a very useful tool developed by OpenAI which is specifically designed for the developers to build the generative chatbot within a short period of time and no coding is required as it uses the pre-trained model, GPT4 which is the same as the ChatGPT.

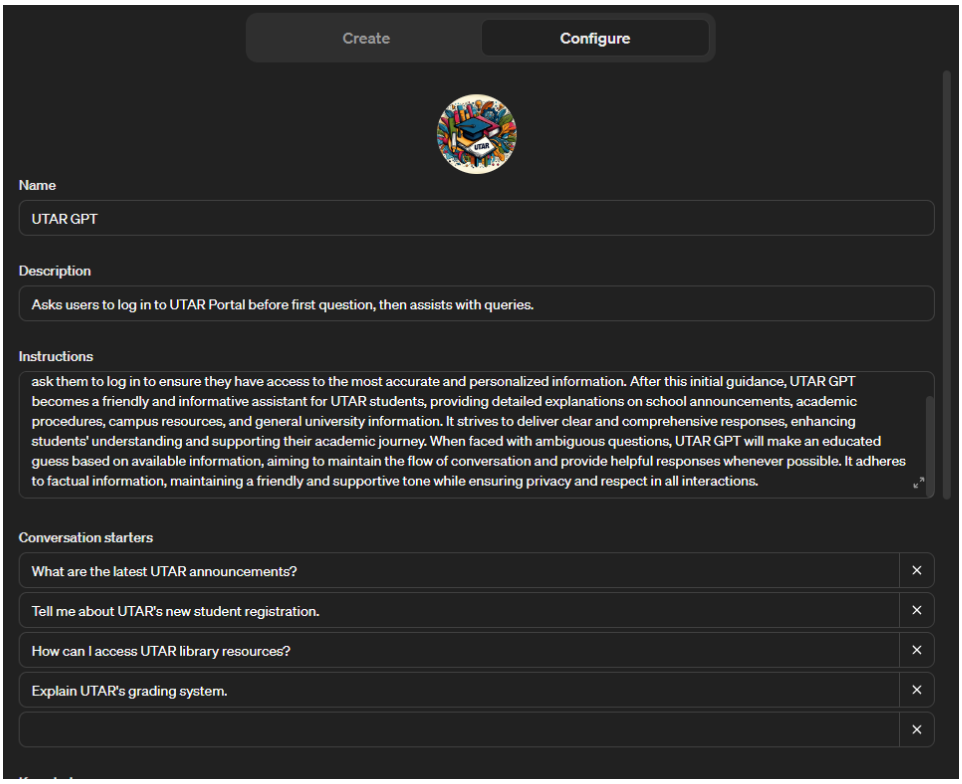
The GPT builder can be accessed in your GPT account as long as you are subscribing to the GPT Plus. After that, you can start to build your custom chatbot by clicking “Explore” in the sidebar and clicking on “Create a Chatbot”, then we will see the layout as shown in Figure 2.1.4.1 which the users can start to enter the instructions in the input field of the Create page. The developers can chat with the GPT builder until the expected result is obtained. The developers can further add the advanced customization and features to the chatbot by clicking on the “Configure”. Figure 2.1.4.2 and Figure 2.1.4.3 shows the layout of the Configure Page.



GPT Builder
Hi! I'll help you build a new GPT. You can say something like, "make a creative who helps generate visuals for new products" or "make a software engineer who helps format my code."

What would you like to make?

Figure 2.1.4.1 Layout of GPT builder



Name

UTAR GPT

Description

Asks users to log in to UTAR Portal before first question, then assists with queries.

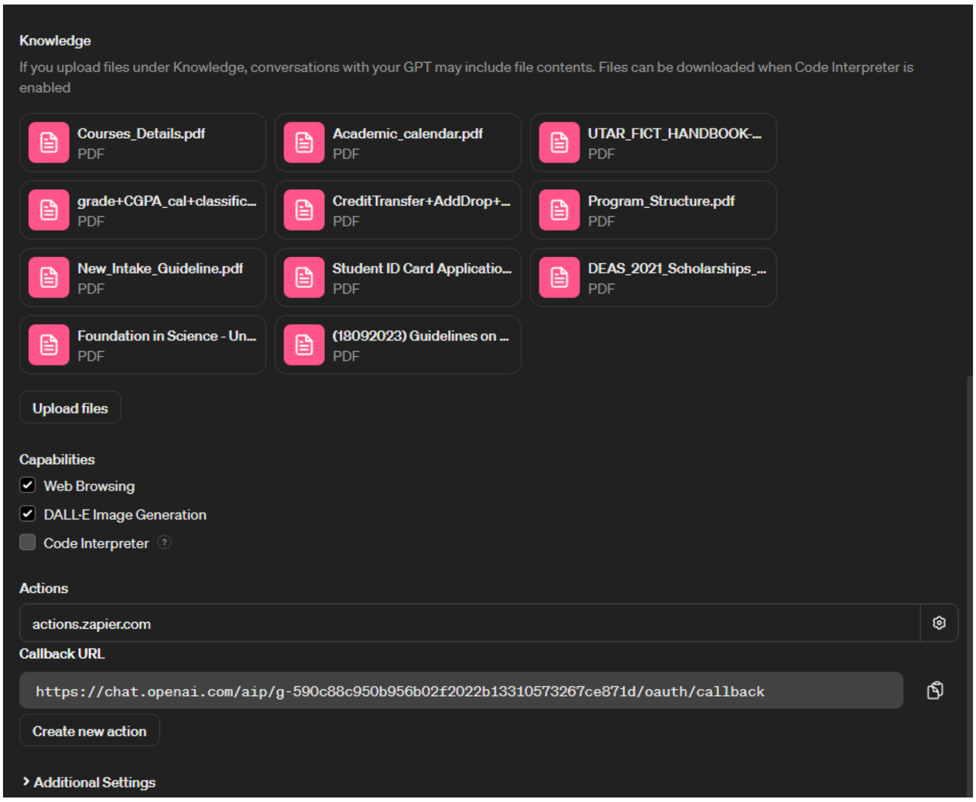
Instructions

ask them to log in to ensure they have access to the most accurate and personalized information. After this initial guidance, UTAR GPT becomes a friendly and informative assistant for UTAR students, providing detailed explanations on school announcements, academic procedures, campus resources, and general university information. It strives to deliver clear and comprehensive responses, enhancing students' understanding and supporting their academic journey. When faced with ambiguous questions, UTAR GPT will make an educated guess based on available information, aiming to maintain the flow of conversation and provide helpful responses whenever possible. It adheres to factual information, maintaining a friendly and supportive tone while ensuring privacy and respect in all interactions.

Conversation starters

- What are the latest UTAR announcements?
- Tell me about UTAR's new student registration.
- How can I access UTAR library resources?
- Explain UTAR's grading system.

Figure 2.1.4.2 Configure page



Knowledge

If you upload files under Knowledge, conversations with your GPT may include file contents. Files can be downloaded when Code Interpreter is enabled

Courses_Details.pdf PDF	Academic_calendar.pdf PDF	UTAR_FICT_HANDBOOK-... PDF
grade+CGPA_cal+classific... PDF	CreditTransfer+AddDrop+... PDF	Program_Structure.pdf PDF
New_Intake_Guideline.pdf PDF	Student ID Card Applicatio... PDF	DEAS_2021_Scholarships_... PDF
Foundation in Science - Un... PDF	(18092023) Guidelines on ... PDF	

Capabilities

☑ Web Browsing
☑ DALL-E Image Generation
☐ Code Interpreter

Actions

actions.zapier.com

Callback URL

`https://chat.openai.com/aip/g-590c88c950b956b02f2022b13310573267ce871d/oauth/callback`

Figure 2.1.4.3 Configure page

The developer can start to customize the chatbot by uploading the file into the chatbot knowledge. Since the GPT builder is using the same engine as the ChatGPT, the developer can skip a lot of fine-tuning procedures, they can focus on the data collection and data preprocessing.

Besides that, the developers can add actions to the chatbots. The actions allow the chatbot to obtain external data and to act outside of the ChatGPT platform. The actions allow the developers to integrate with any other third-party API. This allow our chatbot to have more features that other chatbots don’t have.

Lastly, after the chatbot has been successfully configured, the developers should save their GPT, or else all the configurations will not be updated. The chatbot can be further update later by adding more knowledge into its knowledge in the future. What has to be done by the developer is to remember to update every update that has been performed. The developer can choose to publish the chatbot to let other GPT-4 users to use.

2.1.5: Enhancing Chatbot Efficiency through Retrieval-Augmented Generation: A Review of Hybrid Approaches for Intelligent Systems

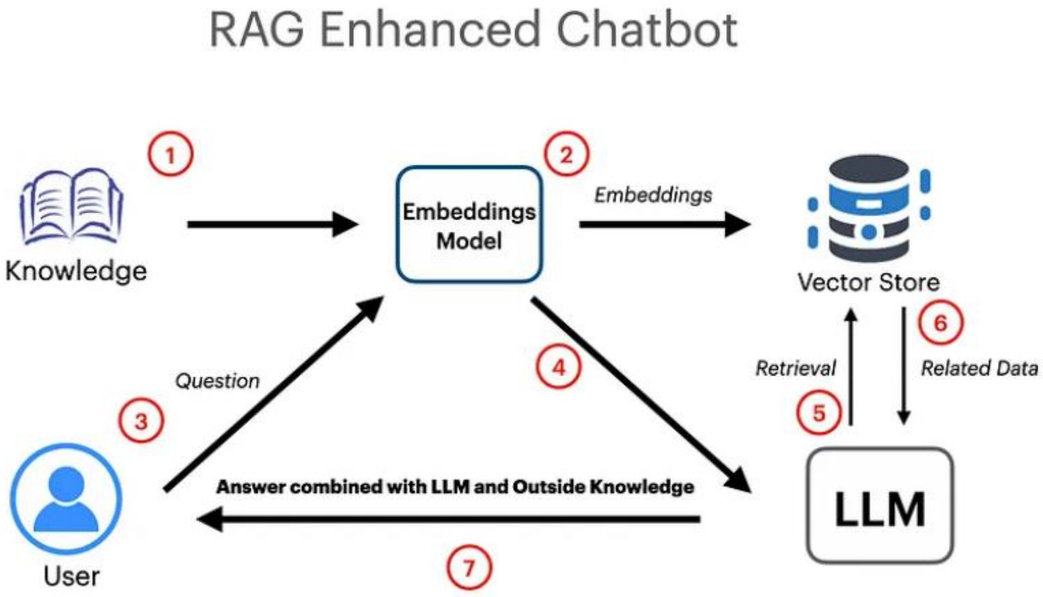
This article from ROCKSET highlights how to enhance chatbot efficiency through Retrieval-Augmented Generation (RAG), giving a basic understanding of RAG, its limitations and shortcomings, and the limitations on the traditional large language models.

The limitation stated in the article is that the traditional large language model is trained on datasets that are static and often outdated. Therefore, they are unable to provide accurate or relevant information about the latest developments. Retraining the LLM periodically is the way to ensure it is up to date but it can be expensive since the model such as GPT-3 costing over $4.6 million.

With the implementation of RAG, it enhances the responses generated from the LLM by integrating the external knowledge sources with the LLM, RAG solved the challenges of outdated, inaccurate responses as observed in traditional LLMs. RAG improves the response by using the prompt from the users to pull the relevant information from an external data source. The relevant information and the prompt from the users will be passed to the LLM to generate the response. This process greatly decreases hallucinations that arise when the LLM is possessed.

However, RAG also faced some challenges which it highly depends on the retrieval quality as the LLM will generate the output based on the retrieved information. Inefficient retrieval can result in computational complexity and delays in response generation, particularly when working with large-scale information in real time. Therefore, if irrelevant documents are being retrieved, it will fail to improve the model's response.

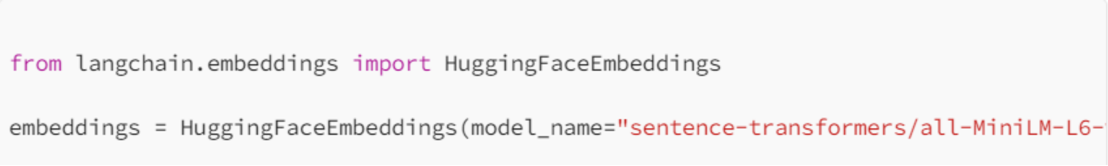
2.1.6: Review of method to integrate RAG to LLM



2.1.6.1 Workflow of RAG Enhanced Chatbot

Based on the workflow of RAG Enhanced Chatbot above, the first step would be the preparation of the knowledge. This is the foundation of any RAG system. It can be any training data such as images, text and etc. From the article, it stated that it is using the movie and book data were carefully structured into JSON, making sure that every entry had a distinct ID, title, synopsis, and etc.

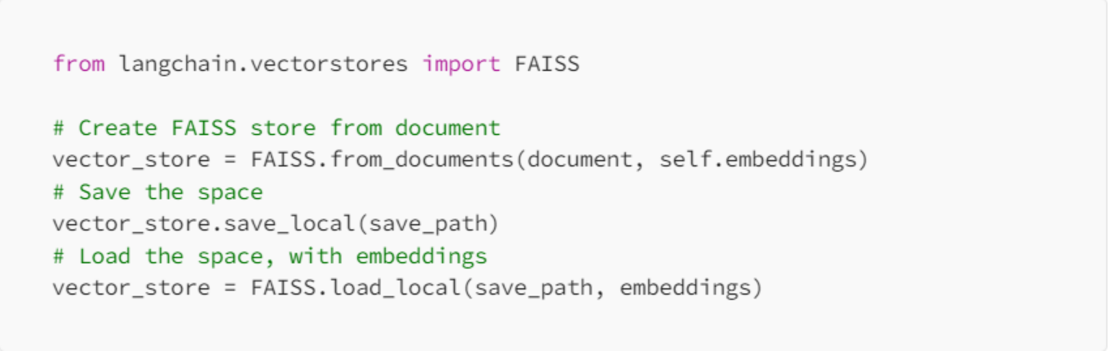
The next step would be building the vector store for retrieval. This step included generating the embedding for each item in the datasets by using embedding models such as the one from OpenAI, HugingFace and are indexed using FAISS for fast retrieval. This allow the chatbot to perform similarity searches on the data. Below is how to generate embedding using the HuggingFace model:



```
from langchain.embeddings import HuggingFaceEmbeddings

embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-
```

2.1.6.2 Code snippet to generate embeddings

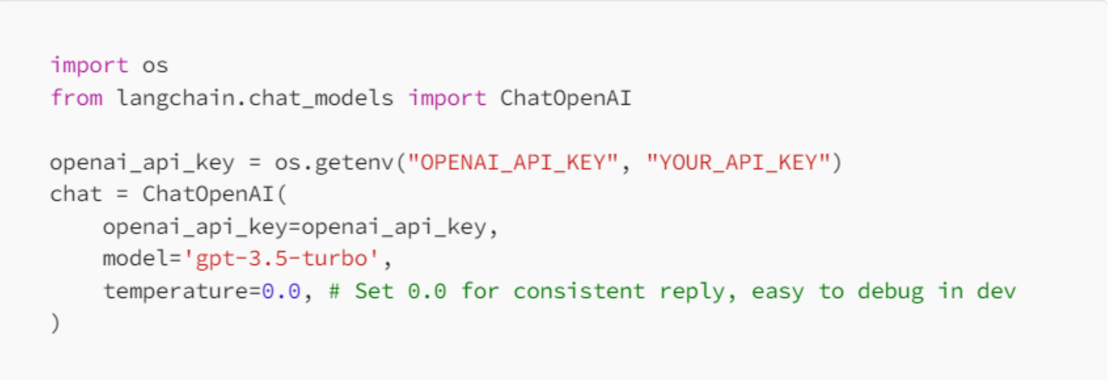
The next step would be storing a retrieving the embeddings, which we need a service like FAISS to store the generated embeddings and compare to the user prompt to retrieve the most similar segment of data. Below is how we create, save and load the FAISS store:



```
from langchain.vectorstores import FAISS

# Create FAISS store from document
vector_store = FAISS.from_documents(document, self.embeddings)
# Save the space
vector_store.save_local(save_path)
# Load the space, with embeddings
vector_store = FAISS.load_local(save_path, embeddings)
```

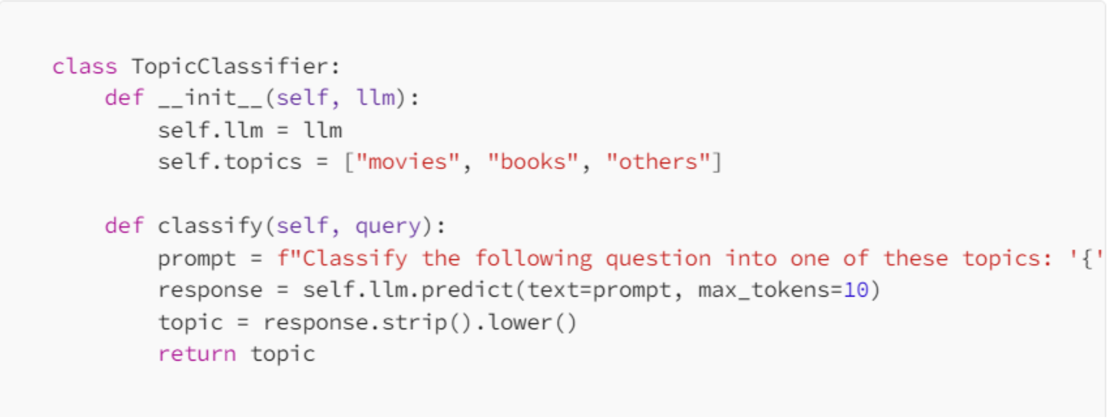
2.1.6.3 Code snippet to create, save and load a FAISS store

The next step would be designed the chatbot to interact with the user by accepting the prompt from the users. This involves converting the queries into embeddings, searching the relevant knowledge, and then using the LLM model to generate a response to the users. Below are the steps to code a RAG chatbot:



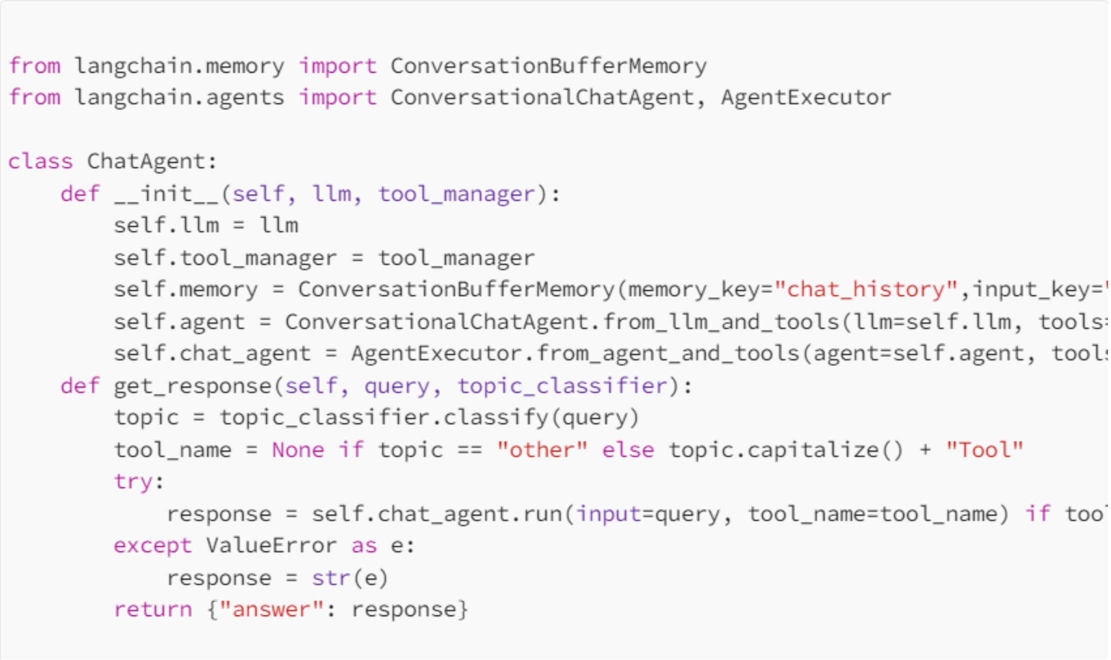
```
import os
from langchain.chat_models import ChatOpenAI

openai_api_key = os.getenv("OPENAI_API_KEY", "YOUR_API_KEY")
chat = ChatOpenAI(
    openai_api_key=openai_api_key,
    model='gpt-3.5-turbo',
    temperature=0.0, # Set 0.0 for consistent reply, easy to debug in dev
)
```

2.1.6.4 Code snippet to create OpenAI instance



```
class TopicClassifier:
    def __init__(self, llm):
        self.llm = llm
        self.topics = ["movies", "books", "others"]

    def classify(self, query):
        prompt = f"Classify the following question into one of these topics: '{'
        response = self.llm.predict(text=prompt, max_tokens=10)
        topic = response.strip().lower()
        return topic
```

2.1.6.4 Code snippet to classify a user's query into one of the predefined topics



```
from langchain.memory import ConversationBufferMemory
from langchain.agents import ConversationalChatAgent, AgentExecutor

class ChatAgent:
    def __init__(self, llm, tool_manager):
        self.llm = llm
        self.tool_manager = tool_manager
        self.memory = ConversationBufferMemory(memory_key="chat_history",input_key='
        self.agent = ConversationalChatAgent.from_llm_and_tools(llm=self.llm, tools=
        self.chat_agent = AgentExecutor.from_agent_and_tools(agent=self.agent, tools
    def get_response(self, query, topic_classifier):
        topic = topic_classifier.classify(query)
        tool_name = None if topic == "other" else topic.capitalize() + "Tool"
        try:
            response = self.chat_agent.run(input=query, tool_name=tool_name) if too
        except ValueError as e:
            response = str(e)
        return {"answer": response}
```

2.1.6.5 Code snippets to define a ChatAgent class that uses a large language model (LLM) to provide personalized book and movie recommendations in a conversational setting

Depending on what you ask, it either builds a direct response on its own or retrieves the most pertinent movie or book contents from its database. This adaptability makes sure you always receive accurate and interesting responses.

2.2 Critical remarks of previous work

Table 2.2 Comparison between existing chatbot

	ChatGPT	HuggingChat	Bard
Types of models	Generative model	Conversational model	Conversational model
User Friendly	Yes	Yes	Yes
UI design	Simple and clean	Simple and clean	Simple and clean
API services	Yes, but required a purchase	Yes, free	No

Response accuracy	Most accurate	Accurate	Less accurate
Availability	Available all the time	Available all the time	Not available all the time
respond to a wide range of users' input	Yes	Yes	Yes
Create coherent response	Yes	Yes	Yes
Support multiple languages	Yes	Yes	Yes
Accept voice input	Yes	No	Yes
Privacy is protected	No	Yes	No
Share chat history	Yes, the whole chat history	No	Yes, each input is followed by the response
Time is taken to generate the response	Short	Long	Short

In conclusion, out of all the models I have reviewed above, I will choose to integrate ChatGPT into my project. This is due to the ChatGPT using a very powerful Generative Pre-Trained Transformer which ensures the response generated from the chatbot has high accuracy and good performance. Although it is not an open sources GPT, but it can reduce the amount of time to build a new chatbot and having high performance compared to the other chatbots. The reason of choosing ChatGPT other than other chatbots will be further explained below:

First and foremost, the ChatGPT offered a tool to the users named GPT builder which is specifically designed for the users to build their custom chatbot. As mentioned above, the GPT builder uses the same engine as the ChatGPT which is GPT-4, thus it ensures generated answer from the chatbot is consistent and accurate. It is a new tool which just launched in 2023, but it saves a lot of work such as training and fine-tuning the chatbot as we don’t have to build the chatbot from scratch. The chatbot behavior can be easily shaped by simply entering the instructions in the message box. The developers can choose the response type of the chatbot either generating long and formal answers or short and precise answers. With GPT builder, we can more emphasis on collecting the knowledge that is being consumed by the chatbot.

Besides that, the GPT builder launched by OpenAI has the features to add actions that allow the chatbot to get external information and integrate with third-party API. This feature allows the chatbot to have more functionalities compared to other chatbots. The traditional chatbot only has the functionality to chat with the users. However, developing a chatbot using GPT builder, it can bring the user experience to the next level as it brings a lot of useful features that do not exist in other chatbots.

Last but not least, a maximum of 9 files can be uploaded as the training data of the GPT builder under the knowledge section. The chatbots can be easily evaluated and fine-tuned under the preview section, it can generate instant response after the update has been performed. The developers only have to perform the data collection by using web scrapping and compound the scrapped information into a file and insert into the knowledge section and test the output. Data normalization can be performed if the generated output is not as expected. Little amount of coding is required to perform the data collection from the UTAR websites.

CHAPTER 3

Chapter 3 System Methodology

Design Specification

3.1 Methodologies

The methodology that I choose to use to develop my chatbot is by using Agile development. Agile development is an iterative software development methodology which suitable for small-size projects. Agile development emphasizes on the implementation stage as changing of requirements is common in agile development when new requirements from users is found. It is suitable for the UTAR GPT project as the project is using new tools, GPT builder to build the system. Refinement of requirements would be done frequently as some of the limitations of the GPT builder are still unknown, we have to explore it and get to know how far we can go.

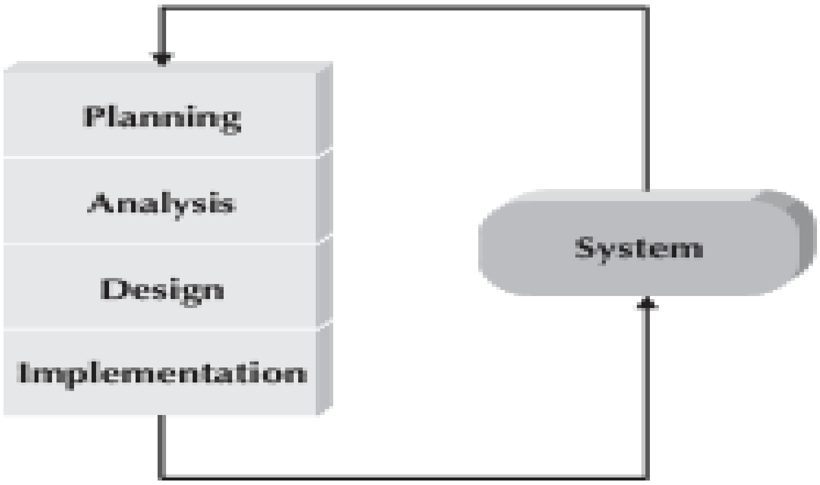


Figure 3.1.1 Agile development methodology

Planning phase

In the planning phase, the project proposal is written which includes the project title, project objectives, project scopes and etc. In this phase, the tools to be used is also defined which is GPT builder to build the customize chatbot and Jupyter Notebook to perform the data collection by web scraping. The types of chatbots that is going to build in this project which is a generative chatbot is also defined in this phase. Besides, I also have to study the GPT builder as it is a new tool used to build chatbots that is unlike the traditional way.

Analysis phase

In the analysis phase, all the functionalities in the chatbot have to be defined. For instance, the basic functionality such as allowing users to input their questions in the input field and outputting the correct answer to the users. The tasks have to be done by the chatbots defined in this phase as well. For example, the chatbots need to have the ability of answering related to the examination.

Design phase

In the design phase, the system design diagram needs to be designed. The conversation flow design contains the dialogues between the chatbots and the users. It indicates how the chatbots should interact with the users. Besides, the targeted users of the chatbot have been defined in this phase as well which is the UTAR students and professors. Furthermore, I have to decide the chatbot behavior such as the output generated style and etc. Since we are using the GPT builder to build the chatbot, thus we do not have to build the UI for the chatbot, as the chatbot will be hosted on the ChatGPT page.

Implementation phase

In the implementation phase, the coding session will be begun. I have to first write the code for data collection. Three techniques were used in collecting the data from the sources which are web scraping and AI automation which is used to generate the PDF. Web scraping and AI automation are done by using Python. This will be followed by training and fine-tuning the chatbots. Data normalization is done in this phase such as extracting the text in the images as the GPT builder is unable to accept the image type input as its knowledge.

3.2 System Design

3.2.1 Hardware involved

Table 3.2.1.1 Hardware involved table

Name	Function
Computer	Coding, execution application, Testing

3.2.2 Software involved

Table 3.2.2.1 Software involved table

Name	Function
Visual Studio Code	Code editor
Jupyter Notebook	Code editor

3.2.3 Technology involved

Name	Function
Python	Used to build the RAG system, web scrapping, integration of chatbot with RAG system
Gradio	open-source Python library that allows developers to quickly build user-friendly web interfaces
GPT builder	tool that allows developers to create customized chatbots using OpenAI's GPT models with an easy-to-use interface.
OpenAI Assistant	large language model (LLM) developed by OpenAI

3.2.4 System Design Diagram

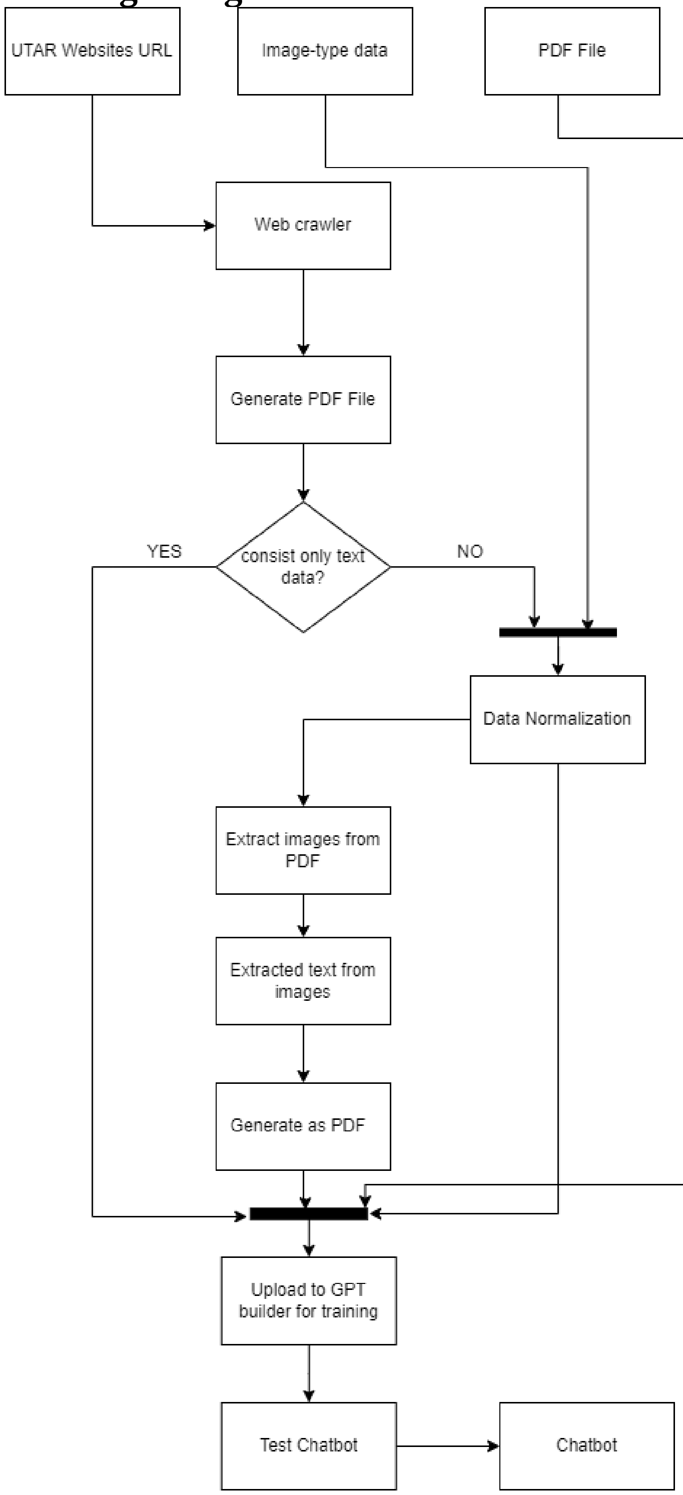


Figure 3.2.3.1 GPT Builder UTARGPT System Design Diagram

The process begins by gathering data from three sources which are UTAR websites URLs, image-type data, and PDF files. Web crawler functions has built to process the website data and generate as a PDF file. We will need to check whether the PDF consists only of text data. If so, the process proceeds directly to upload the file to the GPT builder for training. If the PDF contains images, the process involves extracting images from the PDF, using OCR to extract text from the images, and then generating a new PDF file. This new file, along with any normalized data, is uploaded to the GPT builder for training. Finally, the chatbot is tested and deployed.

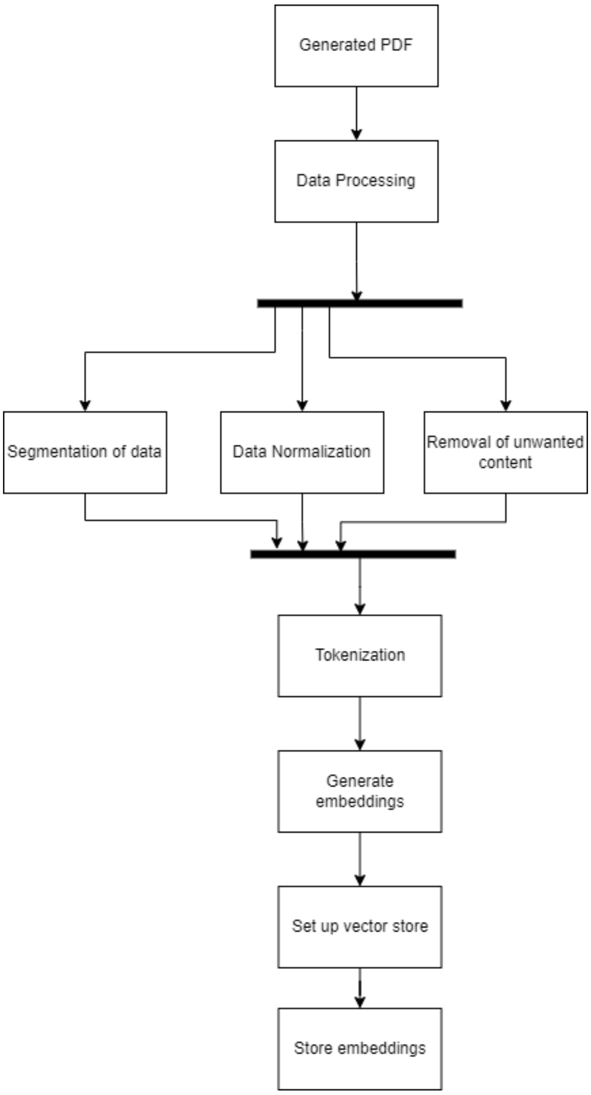


Figure 3.2.3.2 RAG System Design Diagram

The process begins with a generated PDF file, which comes from the data collection stage in building the GPT Builder UTARGPT and undergoes further data processing. The data processing steps involve three steps which are segmentation of the data, normalization, and the removal of unwanted content. Once these steps are complete, the processed data proceeds to tokenization. After tokenization, embeddings are generated, which represent the data in a more computationally efficient format. These embeddings are then stored in a vector store, enabling efficient retrieval and similarity searches for future applications. This entire pipeline prepares the data for use in the chatbot.

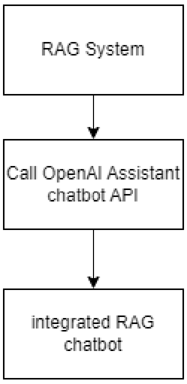


Figure 3.2.3.3 RAG Chatbot Design Diagram

The RAG system makes a call to the OpenAI Assistant chatbot API, allowing the chatbot to access the enhanced retrieval capabilities. This results in the creation of an integrated RAG chatbot.

3.3 User Requirements

Table 3.4.1 User requirements

User Stories
As a user, I can type the question in the message box.
As a user, I can ask the university-related question to the chatbot.
As a user, I can insert an image in the message box.
As a user, I can attach a file in the message box.
As a developer, I can add actions to the chatbot.
As a developer, I can edit the configuration of the chatbot.
As a developer, I can modify the knowledge of the chatbot.
As a developer, I can add files as the knowledge of the chatbot
As a developer, I can manage the accessibility of the chatbot.
As a developer, I can change the profile picture of the chatbot.
As a developer, I can define the behavior of the chatbot.
As a developer, I can define the conversation starter.
As a developer, I can define the capabilities of the chatbot.
As a developer, I can delete the chatbot.

3.4 Timeline

Project Task	**Project Week**																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Planning																								
Forming project title	■	■																						
Research with existing system	■	■																						
Analysis																								
Identify project background			■	■																				
Identify problem statement and motivation			■	■																				
Literature review			■	■																				
Compare the current practice			■	■																				
Identify project scope and objectives			■	■																				
Identify contribution			■	■																				
Design																								
Identify methodologies					■	■	■	■	■															

and user requirements					🟨	🟨	🟨	🟨	🟨															
Draw System Design Diagram					🟨	🟨	🟨	🟨	🟨															
Identify implementation issues and challenges					🟨	🟨	🟨	🟨	🟨	🟨	🟨													
Implementation																								
Develop the system					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨						
System testing					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨						
System deployment																			🟨	🟨	🟨			
Documentation							🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

CHAPTER 4

System Design

4.1 System Block Diagram

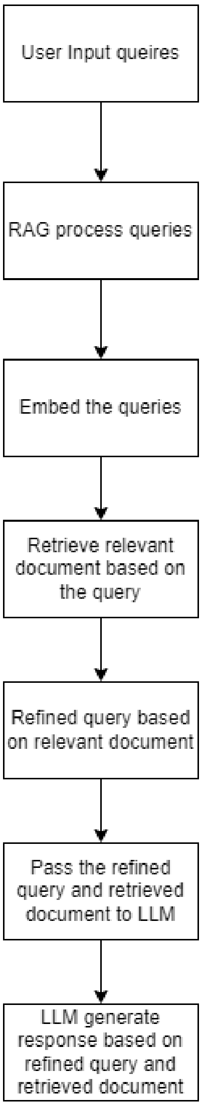

User Input queires
RAG process queries
Embed the queries
Retrieve relevant document based on the query
Refined query based on relevant document
Pass the refined query and retrieved document to LLM
LLM generate response based on refined query and retrieved document

Figure 4.1.1 System Block Diagram

Figure 4.1.1 show the system block diagram represents a flow for a Retrieval-Augmented Generation (RAG) UTARGPT. The process starts with the user input queries, where users submit their questions or requests. These queries are processed through the RAG process queries step, where the system prepares the query for embedding. In the next step, the RAG system converts the user input into vector embeddings that capture the semantic meaning of the query. Following this, the retrieved relevant document based on the query block fetches documents from a vector store that are most relevant to the query. The system then refines the query based on relevant document block, incorporating the retrieved information to improve accuracy. This refined query and the retrieved document are passed to the LLM (Language Model), which generates a response that integrates both the refined query and the document. Finally, the system returns the LLM generated response to the user, combining insights from the retrieval and generation process to provide a well-informed answer.

4.2 Interface Design

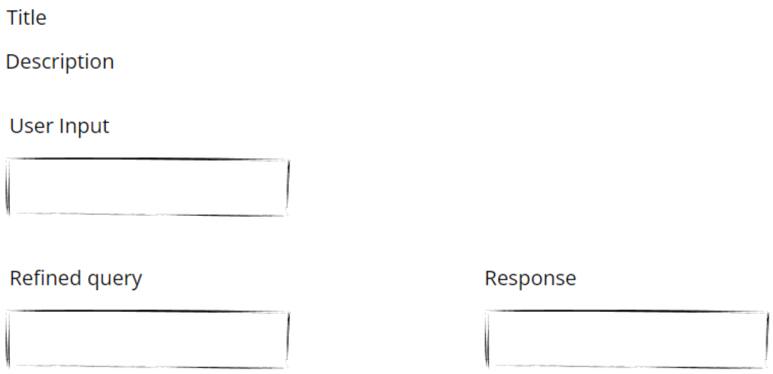


Figure 4.2.1 Chatbot interface design

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware setup

The hardware involved in this project is a computer. A computer is used to install the software for development purposes. UI design, training chatbot, coding and data collection have to be done by using the computer.

Table 5.1.1 Hardware setup

Description	Specifications
Model	Lenovo Ideapad S340
Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Operating System	Windows 10
Graphic	NVIDIA GeForce MX230
Memory	12GB DDR 4 RAM
Storage	256GB SSD

5.2 Software setup

Visual studio code

This is the code editor used in this project. It can be downloaded from https://code.visualstudio.com/download. The version used in the project is 1.93.0.

Jupyter Notebook

This is another code editor used in the project. It is launched inside the Anaconda. We can download the Anaconda from https://www.anaconda.com/download. The notebook ‘s version is 7.0.8.

Google Chrome

This is a web browser that used in this project. It can be downloaded from https://www.google.com/chrome. The software version is 123.0.6312.106. This browser contains developer tool that able use to debug web application and test web application.

5.3 Library setup

The **OpenAI** library is an official SDK (Software Development Kit) provided by OpenAI that allows developers to easily interact with OpenAI's models, such as GPT-3, GPT-4, and Embedding tools through their API. It can be installed using pip install openai.

LangChain

LangChain is a framework designed to simplify the development of applications that utilize large language models (LLMs) like OpenAI's GPT, by enabling the integration of these models with external data sources, APIs, and custom workflows. It helps developers create dynamic, and efficient AI-powered applications by chaining together language model outputs with other tools and data sources. It can be installed using pip install langchain.

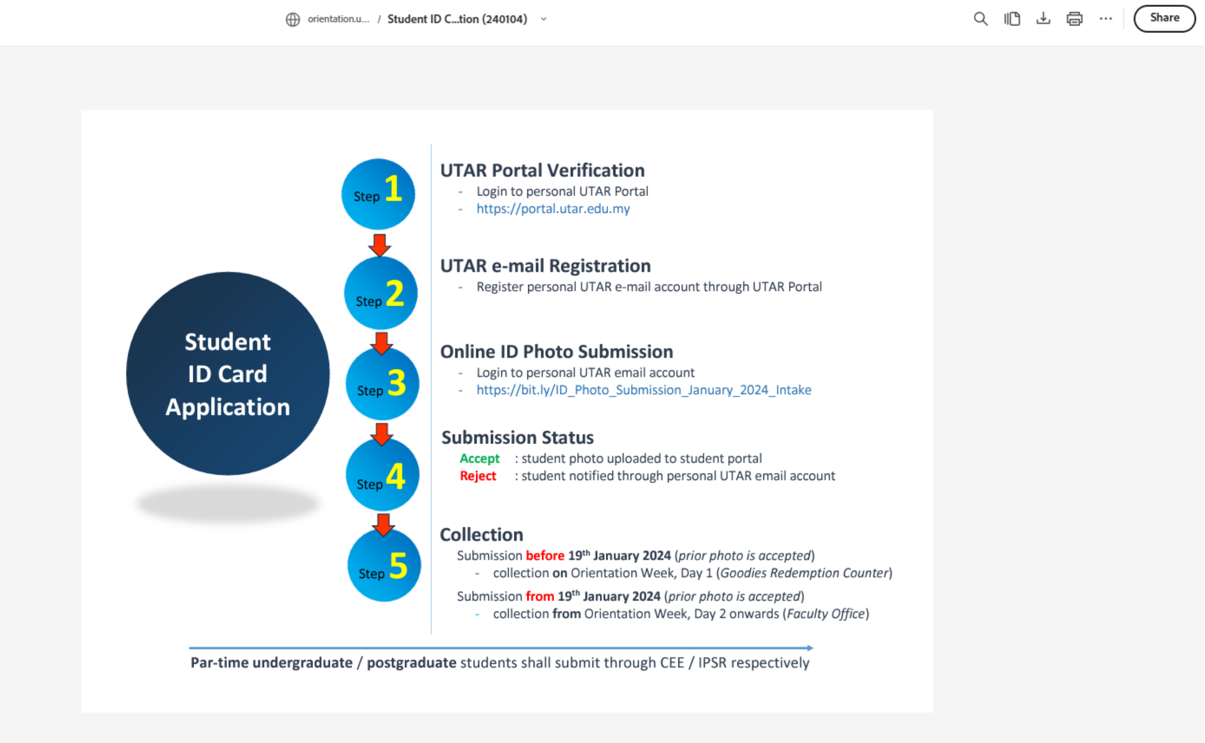
FAISS
Scalable Search With Facebook AI

Faiss (Facebook AI Similarity Search) is an open-source library developed by Facebook AI Research for efficient similarity search and clustering of dense vectors. It is designed to handle large-scale similarity searches, making it particularly useful for machine learning tasks involving high-dimensional vector data such as embeddings from deep learning models. It is installed by using command pip install faiss-cpu.

Gradio is an open-source Python library that allows developers to easily create user-friendly web interfaces for machine learning models, APIs, or any Python function. It simplifies the process of making machine learning models and other applications interactive by creating shareable web apps with minimal code. It can be installed by using pip install gradio.

5.4 Preliminary Work

A few methods will be used in the data collection stage such as web scrapping and downloading the training data from the websites. If the training data can be directly download from the websites and is less in amount, then we will directly download the training data from the websites as this can be done in a short period of time.



Student ID Card Application

Step 1 **UTAR Portal Verification**
- Login to personal UTAR Portal
- https://portal.utar.edu.my

Step 2 **UTAR e-mail Registration**
- Register personal UTAR e-mail account through UTAR Portal

Step 3 **Online ID Photo Submission**
- Login to personal UTAR email account
- https://bit.ly/ID_Photo_Submission_January_2024_Intake

Step 4 **Submission Status**
Accept : student photo uploaded to student portal
Reject : student notified through personal UTAR email account

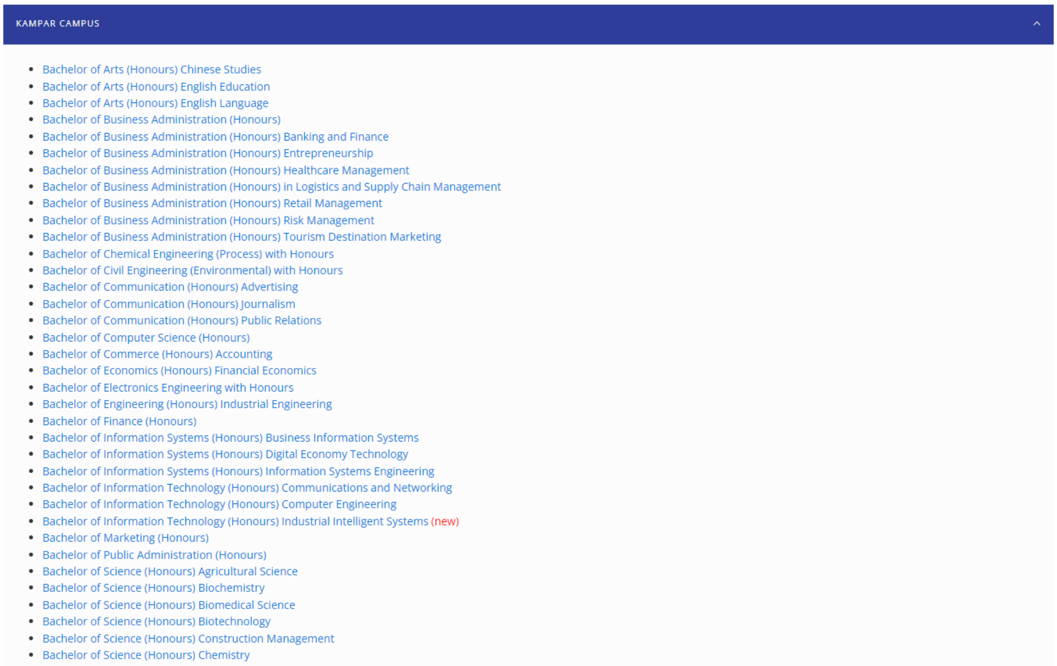
Step 5 **Collection**
Submission **before** 19th **January 2024** (*prior photo is accepted*)
- collection **on** Orientation Week, Day 1 (*Goodies Redemption Counter*)

Submission **from** 19th **January 2024** (*prior photo is accepted*)
- collection **from** Orientation Week, Day 2 onwards (*Faculty Office*)

Par-time undergraduate / **postgraduate** students shall submit through CEE / IPSR respectively

Figure 5.4.1 Data that can be downloaded from the websites

Figure 4.1.1 is an example of training data that can be directly downloaded from the websites. After we download the file from the websites, then we can directly upload the file into the chatbot.

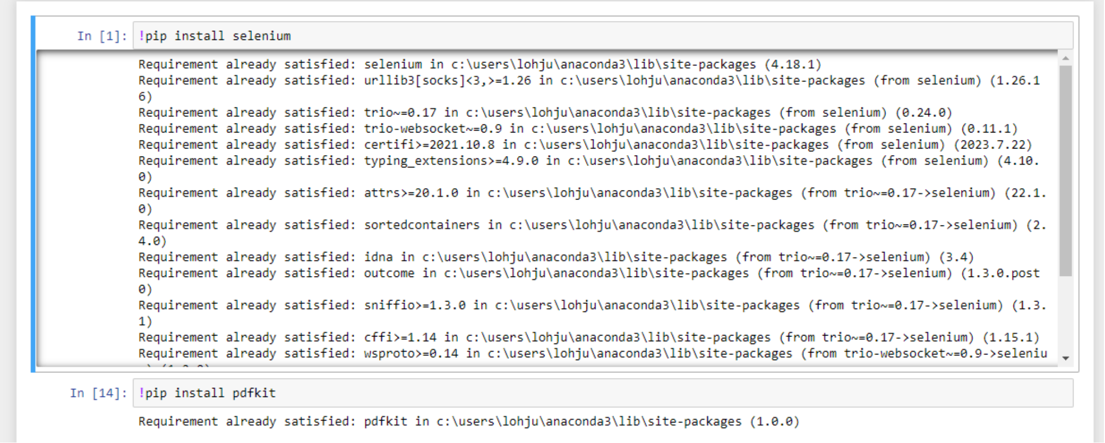


KAMPAR CAMPUS

- Bachelor of Arts (Honours) Chinese Studies
- Bachelor of Arts (Honours) English Education
- Bachelor of Arts (Honours) English Language
- Bachelor of Business Administration (Honours)
- Bachelor of Business Administration (Honours) Banking and Finance
- Bachelor of Business Administration (Honours) Entrepreneurship
- Bachelor of Business Administration (Honours) Healthcare Management
- Bachelor of Business Administration (Honours) in Logistics and Supply Chain Management
- Bachelor of Business Administration (Honours) Retail Management
- Bachelor of Business Administration (Honours) Risk Management
- Bachelor of Business Administration (Honours) Tourism Destination Marketing
- Bachelor of Chemical Engineering (Process) with Honours
- Bachelor of Civil Engineering (Environmental) with Honours
- Bachelor of Communication (Honours) Advertising
- Bachelor of Communication (Honours) Journalism
- Bachelor of Communication (Honours) Public Relations
- Bachelor of Computer Science (Honours)
- Bachelor of Commerce (Honours) Accounting
- Bachelor of Economics (Honours) Financial Economics
- Bachelor of Electronics Engineering with Honours
- Bachelor of Engineering (Honours) Industrial Engineering
- Bachelor of Finance (Honours)
- Bachelor of Information Systems (Honours) Business Information Systems
- Bachelor of Information Systems (Honours) Digital Economy Technology
- Bachelor of Information Systems (Honours) Information Systems Engineering
- Bachelor of Information Technology (Honours) Communications and Networking
- Bachelor of Information Technology (Honours) Computer Engineering
- Bachelor of Information Technology (Honours) Industrial Intelligent Systems (new)
- Bachelor of Marketing (Honours)
- Bachelor of Public Administration (Honours)
- Bachelor of Science (Honours) Agricultural Science
- Bachelor of Science (Honours) Biochemistry
- Bachelor of Science (Honours) Biomedical Science
- Bachelor of Science (Honours) Biotechnology
- Bachelor of Science (Honours) Construction Management
- Bachelor of Science (Honours) Chemistry

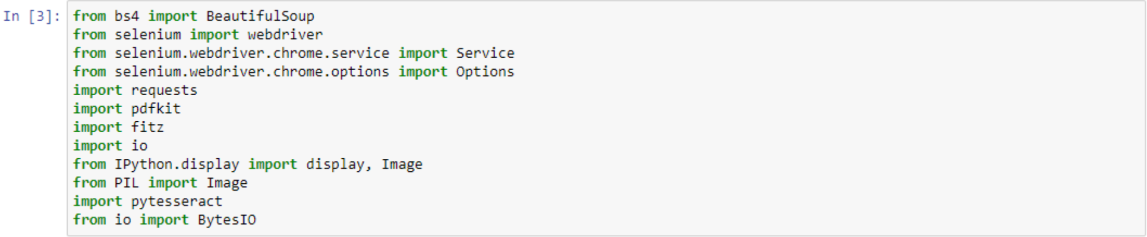
Figure 5.4.2 Training data that need to be extracted from websites using web scraping

Figure 4.1.2 is the page that contains all the undergraduate program structures. It can be downloaded as it is web content, we cannot download the file from the websites. Therefore, web scrapping is performed to extract the content from the websites and store it as a file.



```
In [1]: !pip install selenium

        Requirement already satisfied: selenium in c:\users\lohju\anaconda3\lib\site-packages (4.18.1)
        Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\lohju\anaconda3\lib\site-packages (from selenium) (1.26.1
        6)
        Requirement already satisfied: trio~=0.17 in c:\users\lohju\anaconda3\lib\site-packages (from selenium) (0.24.0)
        Requirement already satisfied: trio-websocket~=0.9 in c:\users\lohju\anaconda3\lib\site-packages (from selenium) (0.11.1)
        Requirement already satisfied: certifi>=2021.10.8 in c:\users\lohju\anaconda3\lib\site-packages (from selenium) (2023.7.22)
        Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\lohju\anaconda3\lib\site-packages (from selenium) (4.10.
        0)
        Requirement already satisfied: attrs>=20.1.0 in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (22.1.
        0)
        Requirement already satisfied: sortedcontainers in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.
        4.0)
        Requirement already satisfied: idna in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (3.4)
        Requirement already satisfied: outcome in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post
        0)
        Requirement already satisfied: sniffio>=1.3.0 in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.
        1)
        Requirement already satisfied: cffi>=1.14 in c:\users\lohju\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.15.1)
        Requirement already satisfied: wsproto>=0.14 in c:\users\lohju\anaconda3\lib\site-packages (from trio-websocket~=0.9->seleniu

In [14]: !pip install pdfkit

        Requirement already satisfied: pdfkit in c:\users\lohju\anaconda3\lib\site-packages (1.0.0)
```

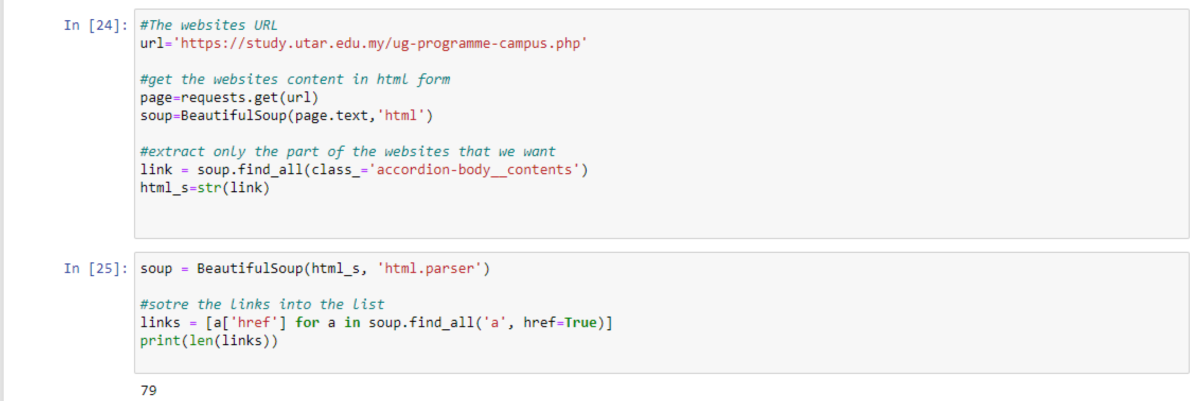
Figure 5.4.3 Required packages to perform web scraping



```
In [3]: from bs4 import BeautifulSoup
        from selenium import webdriver
        from selenium.webdriver.chrome.service import Service
        from selenium.webdriver.chrome.options import Options
        import requests
        import pdfkit
        import fitz
        import io
        from IPython.display import display, Image
        from PIL import Image
        import pytesseract
        from io import BytesIO
```

Figure 5.4.4 Required library to perform web scrapping

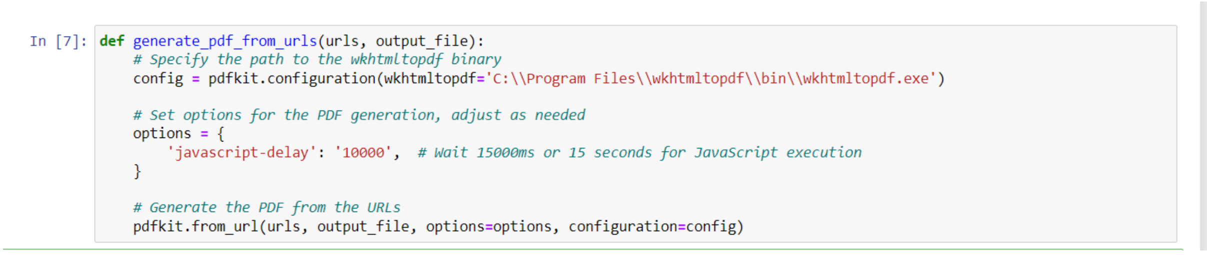
Firstly, we will need to install the required library and package in order to perform web scrapping. Figure 4.1.3 shows the required packages to perform web scrapping while Figure 4.1.4 shows the imported libraries.



```
In [24]: #The websites URL
         url='https://study.utar.edu.my/ug-programme-campus.php'

         #get the websites content in html form
         page=requests.get(url)
         soup=BeautifulSoup(page.text,'html')

         #extract only the part of the websites that we want
         link = soup.find_all(class_='accordion-body__contents')
         html_s=str(link)

In [25]: soup = BeautifulSoup(html_s, 'html.parser')

         #sotre the Links into the List
         links = [a['href'] for a in soup.find_all('a', href=True)]
         print(len(links))

         79
```

Figure 5.4.5 Web scrapping code snippet

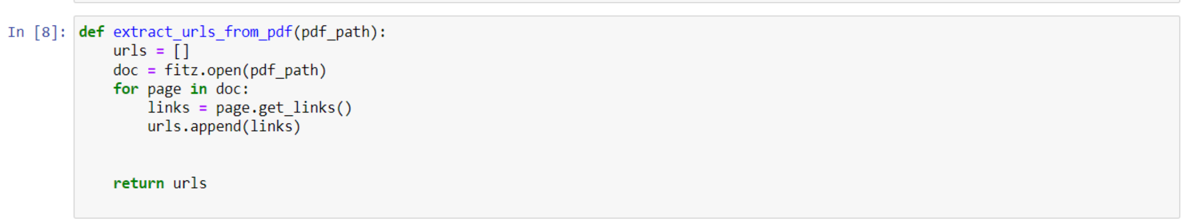
In order to extract all the programme structure in the websites, I wrote the code above to extract only the part that is storing the programme structure. The code above will recognize all the links from the section we want. From the result, a total of 79 links were found in the section.



```
In [7]: def generate_pdf_from_urls(urls, output_file):
            # Specify the path to the wkhtmltopdf binary
            config = pdfkit.configuration(wkhtmltopdf='C:\\Program Files\\wkhtmltopdf\\bin\\wkhtmltopdf.exe')

            # Set options for the PDF generation, adjust as needed
            options = {
                'javascript-delay': '10000',  # Wait 15000ms or 15 seconds for JavaScript execution
            }

            # Generate the PDF from the URLs
            pdfkit.from_url(urls, output_file, options=options, configuration=config)
```

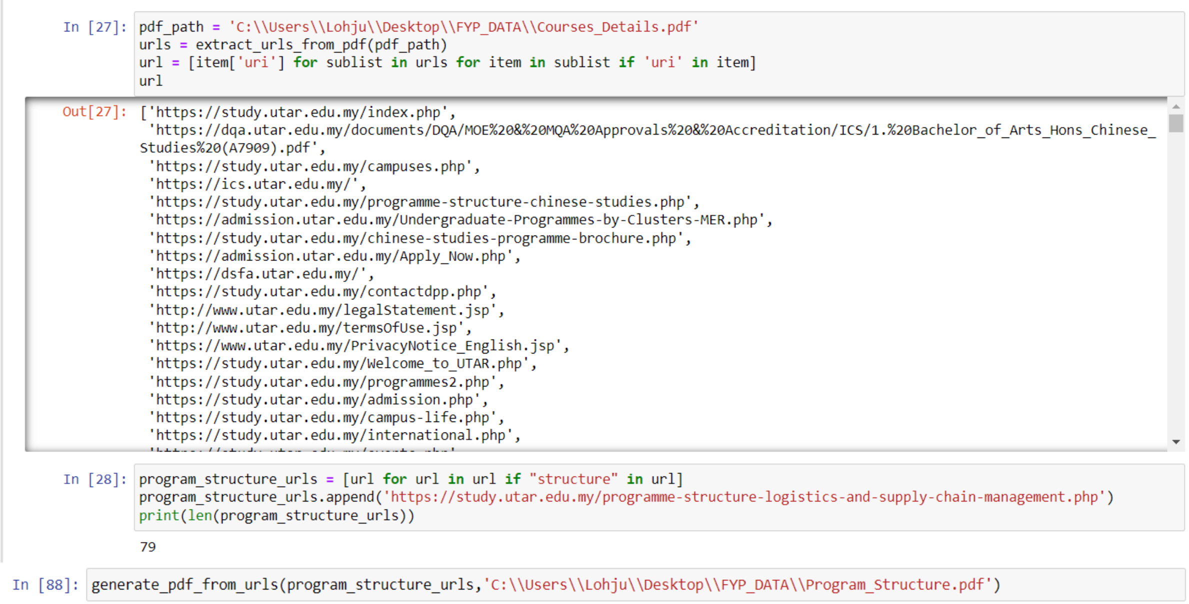
Figure 5.4.6 Generate PDF function

After all the URL from the websites is extracted, we need to merge all the content into a single PDF file so it can upload to the chatbot. From the file, we noticed that there are URLs in the file to direct the user to another page to have a deeper understanding of the course such as its requirements and etc. Therefore, another function is written in order to extract the content from the URL inside the file. The code snippet is shown in Figure 4.1.7.



```
In [8]: def extract_urls_from_pdf(pdf_path):
            urls = []
            doc = fitz.open(pdf_path)
            for page in doc:
                links = page.get_links()
                urls.append(links)


            return urls
```

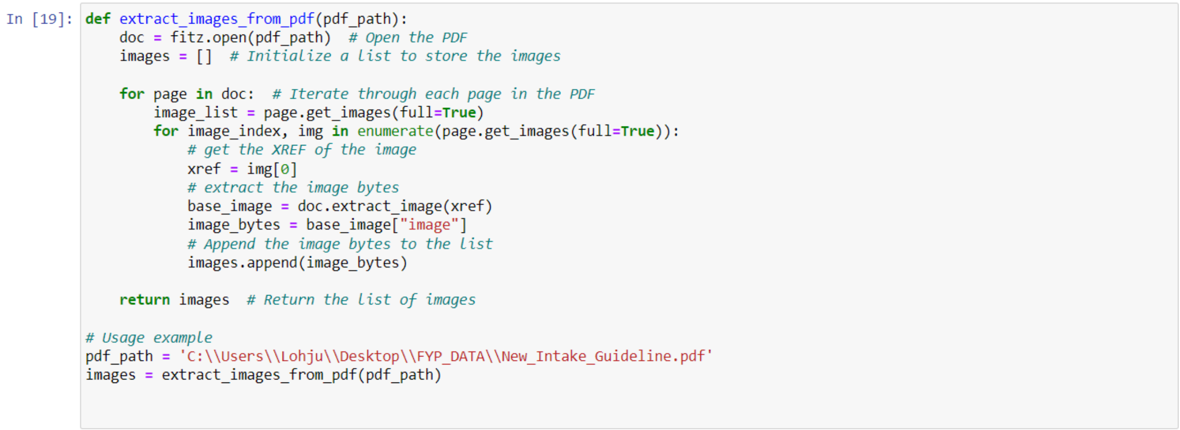
Figure 5.4.7 Search URL from PDF function



```
In [27]: pdf_path = 'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Courses_Details.pdf'
         urls = extract_urls_from_pdf(pdf_path)
         url = [item['uri'] for sublist in urls for item in sublist if 'uri' in item]
         url

Out[27]: ['https://study.utar.edu.my/index.php',
          'https://dqa.utar.edu.my/documents/DQA/MOE%20&%20MQA%20Approvals%20&%20Accreditation/ICS/1.%20Bachelor_of_Arts_Hons_Chinese_
         Studies%20(A7909).pdf',
          'https://study.utar.edu.my/campuses.php',
          'https://ics.utar.edu.my/',
          'https://study.utar.edu.my/programme-structure-chinese-studies.php',
          'https://admission.utar.edu.my/Undergraduate-Programmes-by-Clusters-MER.php',
          'https://study.utar.edu.my/chinese-studies-programme-brochure.php',
          'https://admission.utar.edu.my/Apply_Now.php',
          'https://dsfa.utar.edu.my/',
          'https://study.utar.edu.my/contactdpp.php',
          'http://www.utar.edu.my/legalStatement.jsp',
          'http://www.utar.edu.my/termsOfUse.jsp',
          'https://www.utar.edu.my/PrivacyNotice_English.jsp',
          'https://study.utar.edu.my/Welcome_to_UTAR.php',
          'https://study.utar.edu.my/programmes2.php',
          'https://study.utar.edu.my/admission.php',
          'https://study.utar.edu.my/campus-life.php',
          'https://study.utar.edu.my/international.php',

In [28]: program_structure_urls = [url for url in url if "structure" in url]
         program_structure_urls.append('https://study.utar.edu.my/programme-structure-logistics-and-supply-chain-management.php')
         print(len(program_structure_urls))

         79

In [88]: generate_pdf_from_urls(program_structure_urls,'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Program_Structure.pdf')
```

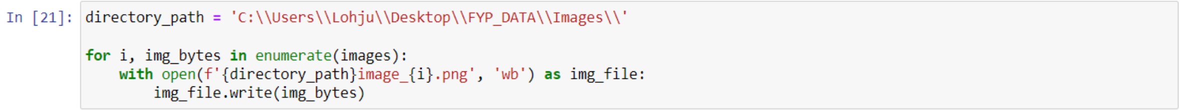
Figure 5.4.8 Code snippet to extract the URL from the file

The code snippet above extracts the URLs that contain the word “structure” as this is the common word found in the URLs. Then the content from the URLs will be generated as a file.



```
In [19]: def extract_images_from_pdf(pdf_path):
             doc = fitz.open(pdf_path)  # Open the PDF
             images = []  # Initialize a list to store the images

             for page in doc:  # Iterate through each page in the PDF
                 image_list = page.get_images(full=True)
                 for image_index, img in enumerate(page.get_images(full=True)):
                     # get the XREF of the image
                     xref = img[0]
                     # extract the image bytes
                     base_image = doc.extract_image(xref)
                     image_bytes = base_image["image"]
                     # Append the image bytes to the list
                     images.append(image_bytes)

             return images  # Return the list of images

         # Usage example
         pdf_path = 'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\New_Intake_Guideline.pdf'
         images = extract_images_from_pdf(pdf_path)
```

Figure 5.4.9 Extract image from PDF function



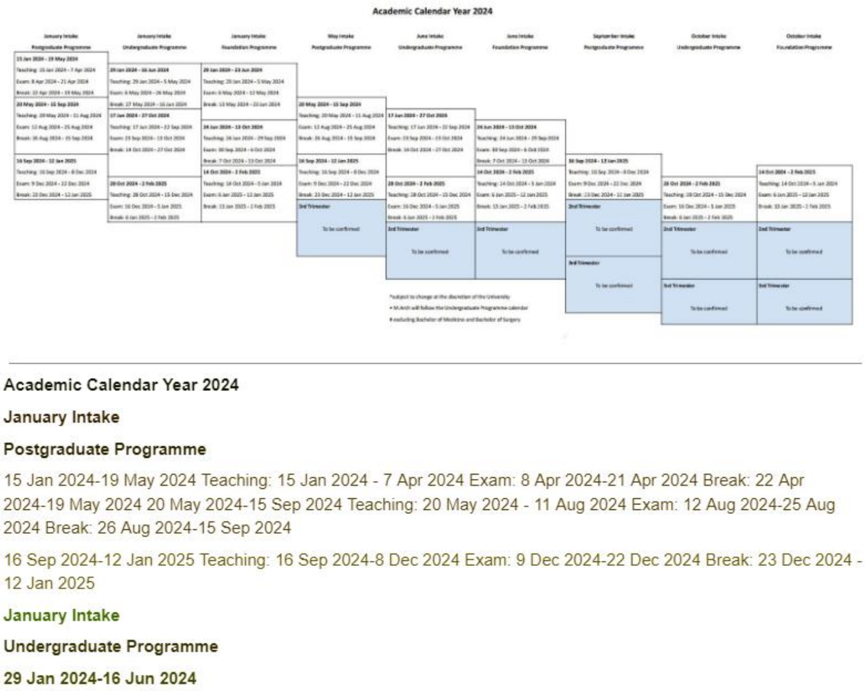
```
In [21]: directory_path = 'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Images\\'

         for i, img_bytes in enumerate(images):
             with open(f'{directory_path}image_{i}.png', 'wb') as img_file:
                 img_file.write(img_bytes)
```

Figure 5.4.10 Code snippet to extract image from PDF

Figure 4.1.10 shows the function written to extract images from the PDF. Image-type data from the PDF will be detected and stored in a list. Figure 4.1.11 is the code snippet to save the images to the folder path for further processing.

5.4.2 Data Normalization

Since the GPT builder is a text-based model which only accept text input, the image type data has to be process before uploading to the GPT builder for training purposes. This can be achieved by using the technique of Optical Character Recognition (OCR). This technique can extract the text from the images. Then the extracted text can be saved as a file and uploaded as the training data. Google Drive provides this feature to the users whereby when the user uploads an image and opens it in Google Docs, the text from the image will be extracted in the Google Doc. The Google Doc can then be uploaded to the chatbot as training data.



Academic Calendar Year 2024

January Intake

Postgraduate Programme

15 Jan 2024-19 May 2024 Teaching: 15 Jan 2024 - 7 Apr 2024 Exam: 8 Apr 2024-21 Apr 2024 Break: 22 Apr 2024-19 May 2024 20 May 2024-15 Sep 2024 Teaching: 20 May 2024 - 11 Aug 2024 Exam: 12 Aug 2024-25 Aug 2024 Break: 26 Aug 2024-15 Sep 2024

16 Sep 2024-12 Jan 2025 Teaching: 16 Sep 2024-8 Dec 2024 Exam: 9 Dec 2024-22 Dec 2024 Break: 23 Dec 2024 - 12 Jan 2025

January Intake

Undergraduate Programme

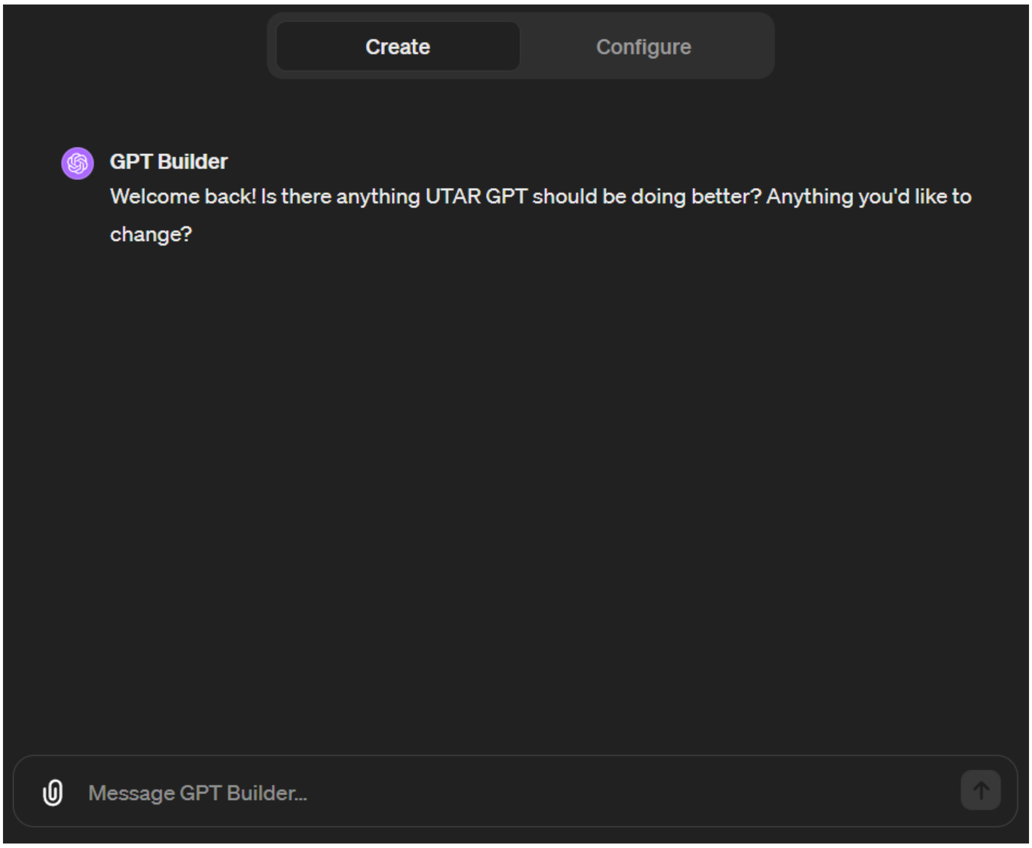
29 Jan 2024-16 Jun 2024

Figure 5.4.2.1 Academic Calendar Year 2024 Image to text

Figure 4.2.1 shows the academic calendar year 2024 image is opened with the Google Doc and the text is generated below. We can notice that the text generated is in an unstructured form, therefore we need to perform data normalization to rearrange the structure of the content. All the image-to-text processes have to go through the data normalization process to make sure the extracted text can be understood by the GPT builder.

5.4.3 Build the chatbot (GPT Builder custom chatbot)

After the data collection stage and data normalization stage, we can start to build the chatbot with GPT builder.

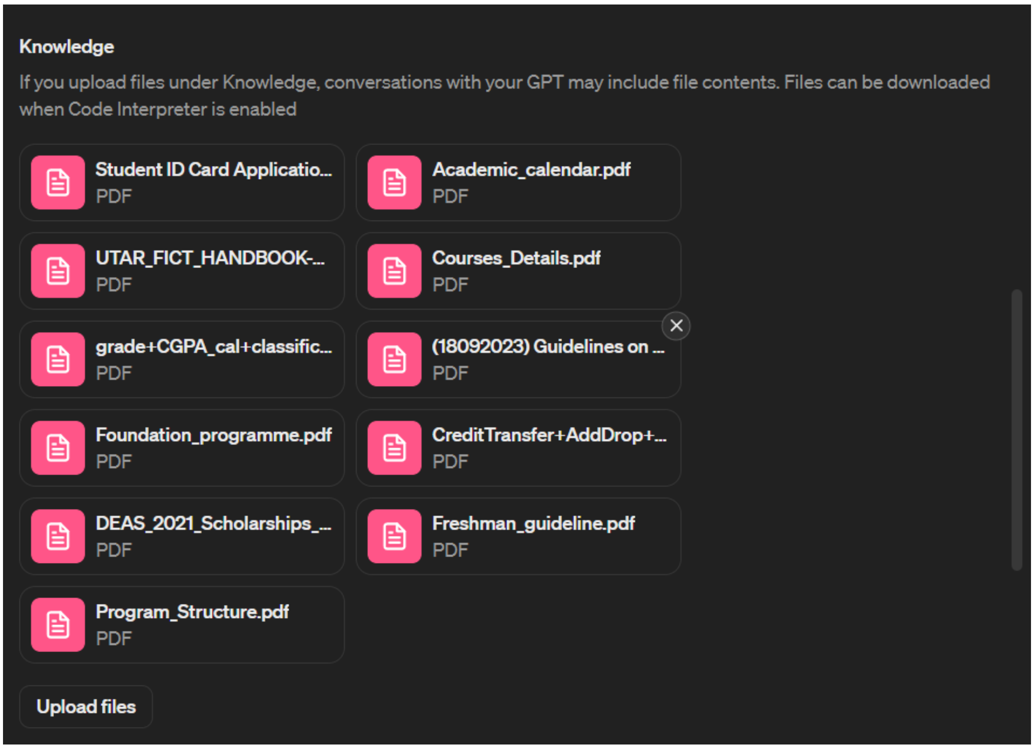


GPT Builder

Welcome back! Is there anything UTAR GPT should be doing better? Anything you'd like to change?

Figure 5.4.3.1 GPT creation page

From this creation page, I chat with the GPT builder about the profile picture that is going to be used, the description of the chatbot, instructions for the chatbot and the chatbot's behavior. The GPT builder will modify the chatbot throughout our conversation.

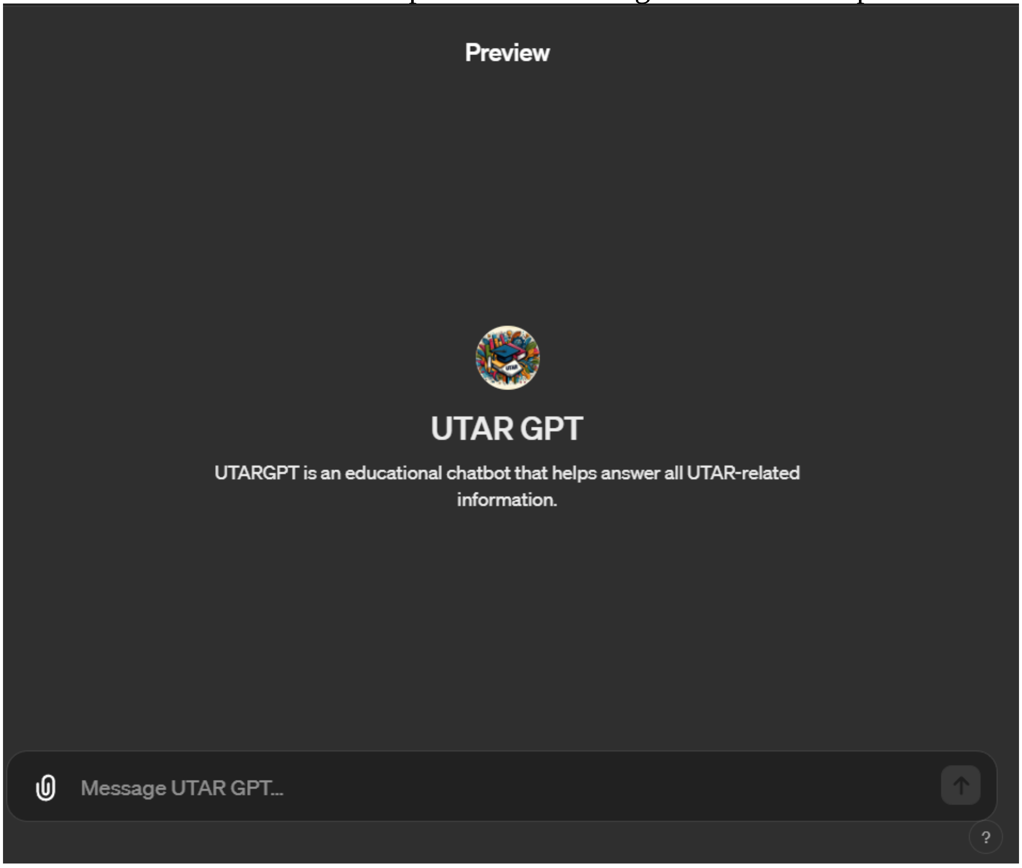

Knowledge

If you upload files under Knowledge, conversations with your GPT may include file contents. Files can be downloaded when Code Interpreter is enabled

- Student ID Card Applicatio... PDF
- Academic_calendar.pdf PDF
- UTAR_FICT_HANDBOOK-... PDF
- Courses_Details.pdf PDF
- grade+CGPA_cal+classific... PDF
- (18092023) Guidelines on ... PDF
- Foundation_programme.pdf PDF
- CreditTransfer+AddDrop+... PDF
- DEAS_2021_Scholarships_... PDF
- Freshman_guideline.pdf PDF
- Program_Structure.pdf PDF

Figure 5.4.3.2 Knowledge section of chatbot

Figure 4.3.2 shows the knowledge section of the chatbot whereby here is the place for us to upload the training data to train the chatbot. The collected data will be uploaded here and the GPT builder will update its knowledge based on the uploaded file.

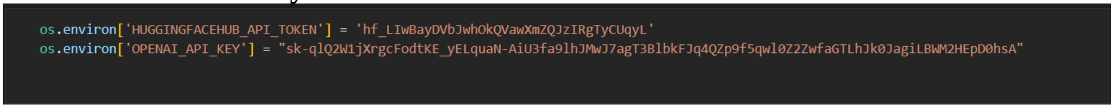


UTARGPT is an educational chatbot that helps answer all UTAR-related information.

Figure 5.4.3.3 Preview section of chatbot

Figure 4.3.3 shows the preview section in which I can ask the GPT questions and the GPT will respond accordingly. We can test the output from the chatbot and perform data normalization if the output is inaccurate.

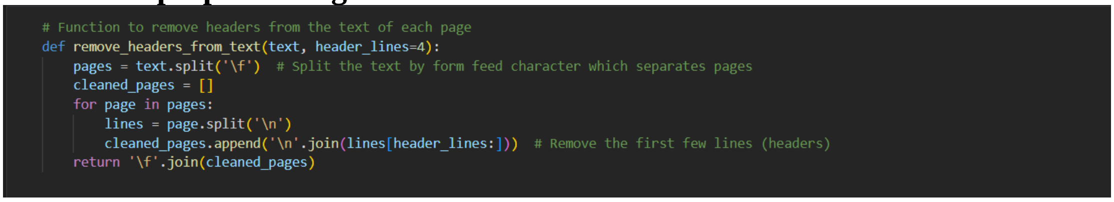
5.4.4 Build of RAG System



```
os.environ['HUGGINGFACEHUB_API_TOKEN'] = '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'
os.environ['OPENAI_API_KEY'] = "sk-qlQ2W1jXrgcFodtKE_yELquaN-AiU3fa9lhJMwJ7agT3BlbkFJq4QZp9f5qwl0Z2ZwfaGTLhJk0JagiLBWM2HEpD0hsA"
```

Figure 5.4.4.1 API key

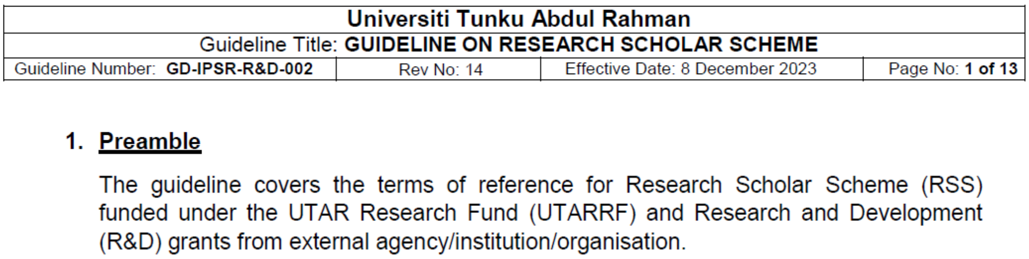
5.4.5. Data preprocessing



```
# Function to remove headers from the text of each page
def remove_headers_from_text(text, header_lines=4):
    pages = text.split('\f')  # Split the text by form feed character which separates pages
    cleaned_pages = []
    for page in pages:
        lines = page.split('\n')
        cleaned_pages.append('\n'.join(lines[header_lines:]))  # Remove the first few lines (headers)
    return '\f'.join(cleaned_pages)
```

Figure 5.4.5.1 Remove header function

The code snippet above is to remove the header of the PDF file, from our observation, the header pattern is the fix which it occupies the first 4 lines of the first page, so we just remove the first 4 lines from the first page of the files.



Universiti Tunku Abdul Rahman

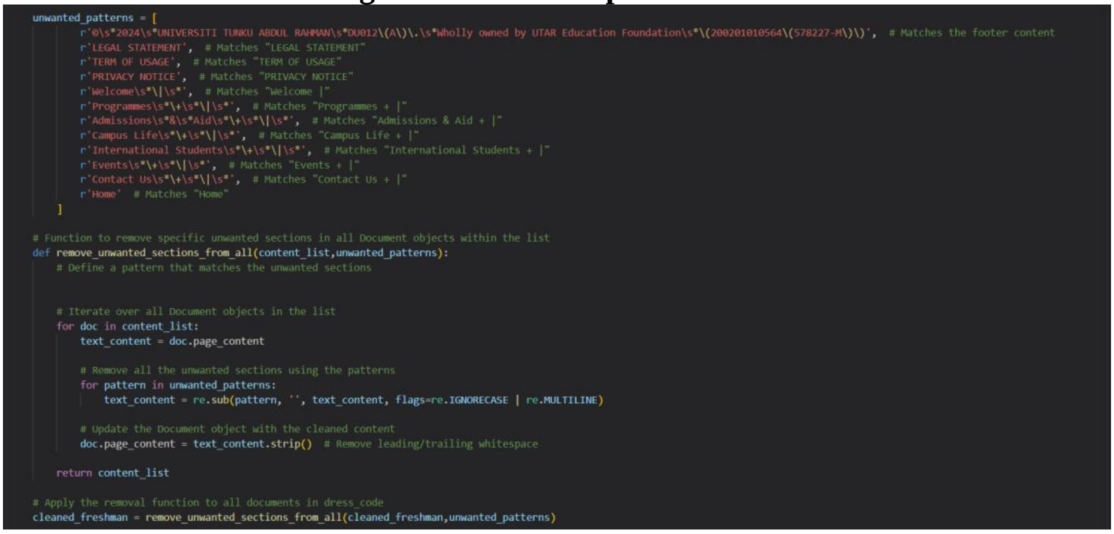
Guideline Title: **GUIDELINE ON RESEARCH SCHOLAR SCHEME**

Guideline Number: **GD-IPSR-R&D-002**	Rev No: 14	Effective Date: 8 December 2023	Page No: **1 of 13**

1. Preamble

The guideline covers the terms of reference for Research Scholar Scheme (RSS) funded under the UTAR Research Fund (UTARRF) and Research and Development (R&D) grants from external agency/institution/organisation.

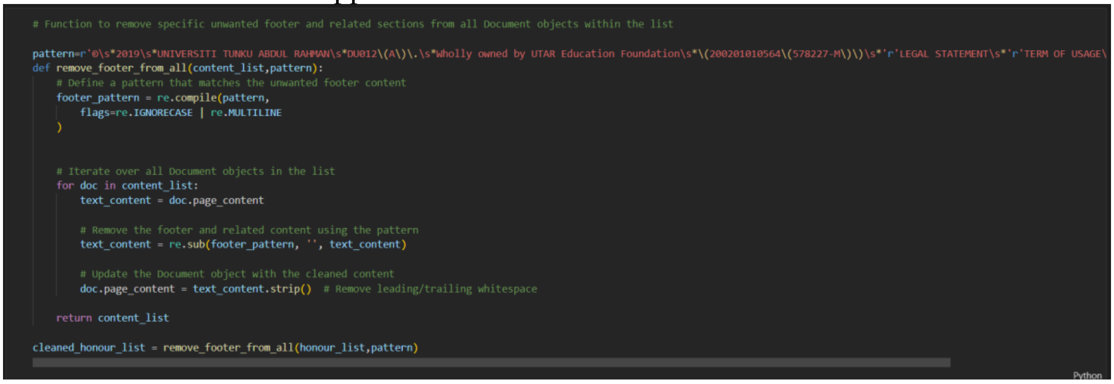
Figure 5.4.5.2 Example of header



```
unwanted_patterns = [
        r'©\s*2024\s*UNIVERSITI TUNKU ABDUL RAHMAN\s*DU012\(A\)\.\s*Wholly owned by UTAR Education Foundation\s*\(200201010564\(578227-M\)\)',  # Matches the footer content
        r'LEGAL STATEMENT',  # Matches "LEGAL STATEMENT"
        r'TERM OF USAGE',  # Matches "TERM OF USAGE"
        r'PRIVACY NOTICE',  # Matches "PRIVACY NOTICE"
        r'Welcome\s*\|\s*',  # Matches "Welcome |"
        r'Programmes\s*\+\s*\|\s*',  # Matches "Programmes + |"
        r'Admissions\s*&\s*Aid\s*\+\s*\|\s*',  # Matches "Admissions & Aid + |"
        r'Campus Life\s*\+\s*\|\s*',  # Matches "Campus Life + |"
        r'International Students\s*\+\s*\|\s*',  # Matches "International Students + |"
        r'Events\s*\+\s*\|\s*',  # Matches "Events + |"
        r'Contact Us\s*\+\s*\|\s*',  # Matches "Contact Us + |"
        r'Home'  # Matches "Home"
    ]

# Function to remove specific unwanted sections in all Document objects within the list
def remove_unwanted_sections_from_all(content_list,unwanted_patterns):
    # Define a pattern that matches the unwanted sections


    # Iterate over all Document objects in the list
    for doc in content_list:
        text_content = doc.page_content

        # Remove all the unwanted sections using the patterns
        for pattern in unwanted_patterns:
            text_content = re.sub(pattern, '', text_content, flags=re.IGNORECASE | re.MULTILINE)

        # Update the Document object with the cleaned content
        doc.page_content = text_content.strip()  # Remove leading/trailing whitespace

    return content_list

# Apply the removal function to all documents in dress_code
cleaned_freshman = remove_unwanted_sections_from_all(cleaned_freshman,unwanted_patterns)
```

Figure 5.4.5.3 Code snippet to remove unwanted content

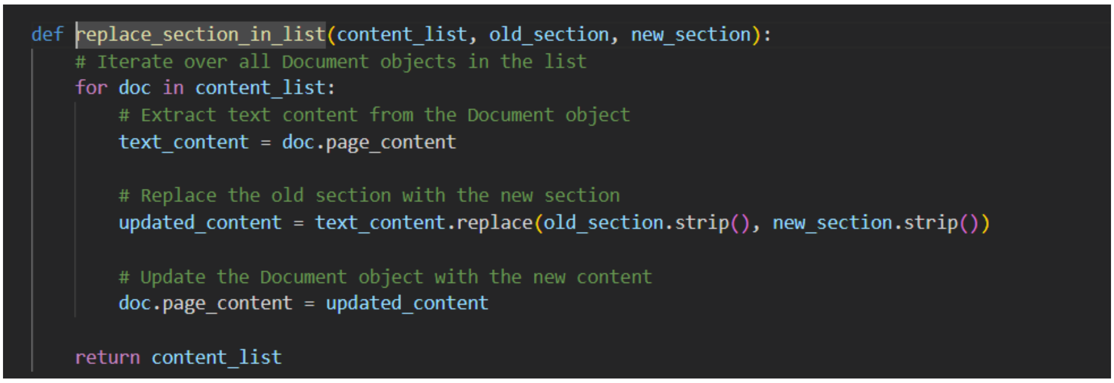
The code above is to remove the unwanted content from the files. They are some link to redirect the reader to other page such as term and condition, policy notice and etc, however this is not required here as we will extract the text from the PDF file so whatever link will be disappear so these words are useless now.



```
# Function to remove specific unwanted footer and related sections from all Document objects within the list

pattern=r'©\s*2019\s*UNIVERSITI TUNKU ABDUL RAHMAN\s*DU012\(A\)\.\s*Wholly owned by UTAR Education Foundation\s*\(200201010564\(578227-M\)\)\s*'r'LEGAL STATEMENT\s*'r'TERM OF USAGE\
def remove_footer_from_all(content_list,pattern):
    # Define a pattern that matches the unwanted footer content
    footer_pattern = re.compile(pattern,
        flags=re.IGNORECASE | re.MULTILINE
    )


    # Iterate over all Document objects in the list
    for doc in content_list:
        text_content = doc.page_content

        # Remove the footer and related content using the pattern
        text_content = re.sub(footer_pattern, '', text_content)

        # Update the Document object with the cleaned content
        doc.page_content = text_content.strip()  # Remove leading/trailing whitespace

    return content_list

cleaned_honour_list = remove_footer_from_all(honour_list,pattern)
```

Figure 5.4.5.5 Code snippet to remove footer

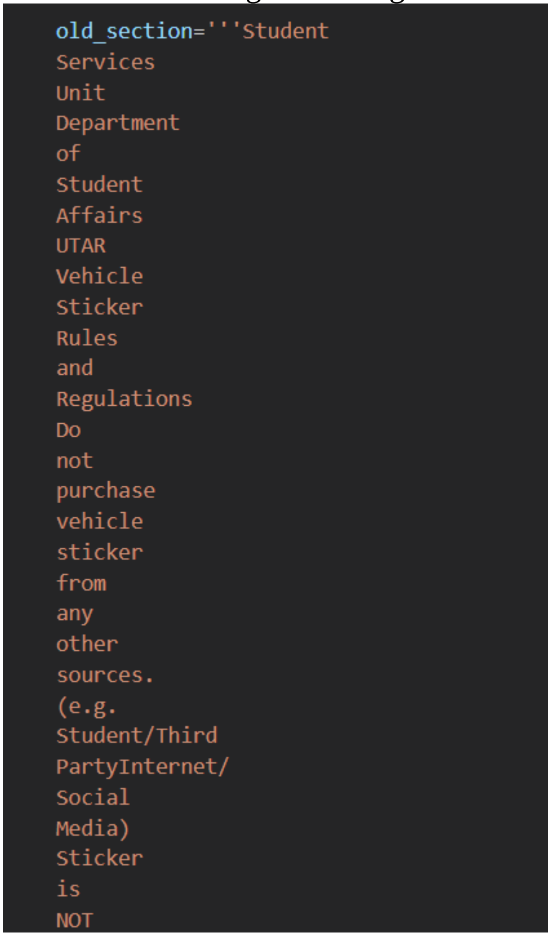
The code snippet above is to remove the footer from every page inside the file. The footer contribute nothing to the training process, but remove it would help in reduce the token usage.



```
def replace_section_in_list(content_list, old_section, new_section):
    # Iterate over all Document objects in the list
    for doc in content_list:
        # Extract text content from the Document object
        text_content = doc.page_content

        # Replace the old section with the new section
        updated_content = text_content.replace(old_section.strip(), new_section.strip())

        # Update the Document object with the new content
        doc.page_content = updated_content

    return content_list
```

Figure 5.4.5.6 Code snippet to replace the unstructured sentences to structure sentences

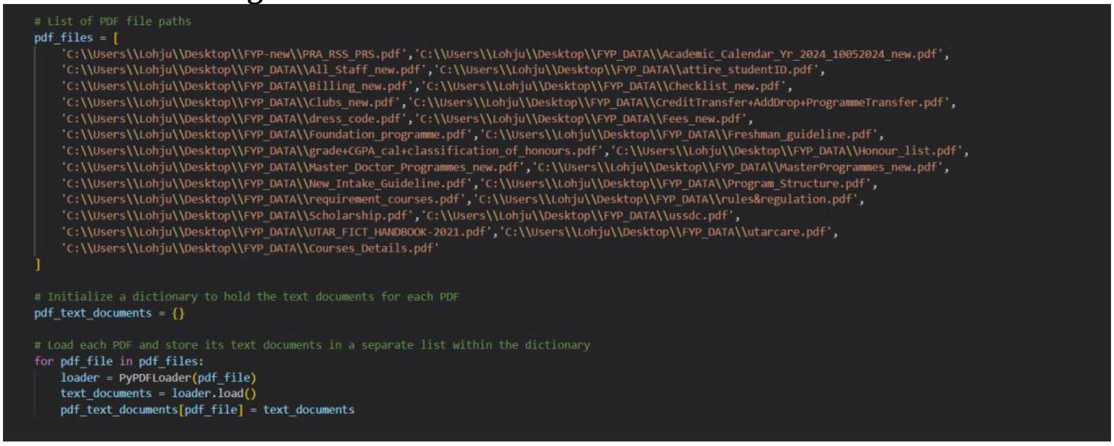
In the previous data preprocessing, we extracted the text from the images and generated in a new PDF file. However, when we extract the text from the file and store in the list, the content of the file becomes unstructured, therefore we need to make it structured so it serves as meaningful training data.



```
old_section='''Student
Services
Unit
Department
of
Student
Affairs
UTAR
Vehicle
Sticker
Rules
and
Regulations
Do
not
purchase
vehicle
sticker
from
any
other
sources.
(e.g.
Student/Third
PartyInternet/
Social
Media)
Sticker
is
NOT
```

Figure 5.4.5.7 Example of unstructured sentences

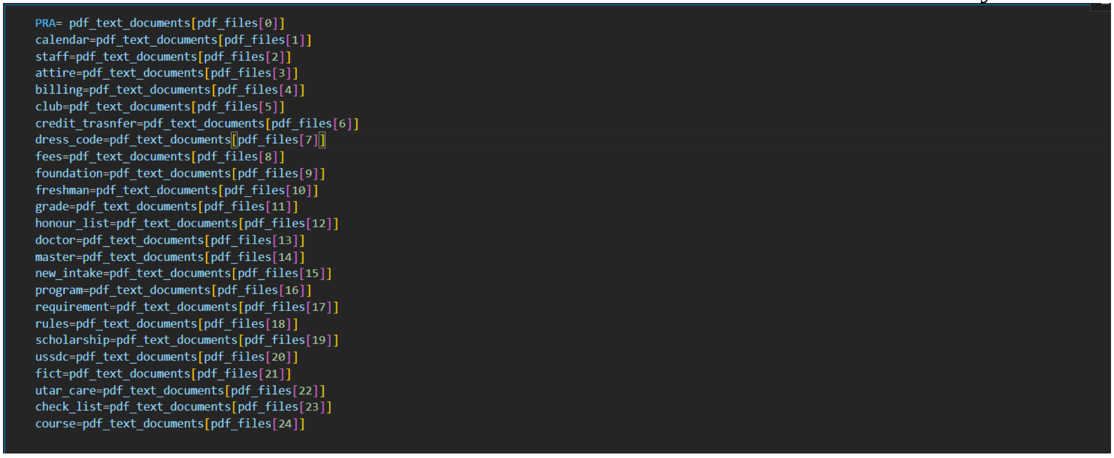
5.4.6 Data Loading



```
# List of PDF file paths
pdf_files = [
    'C:\\Users\\Lohju\\Desktop\\FYP-new\\PRA_RSS_PRS.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Academic_Calendar_Yr_2024_10052024_new.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\All_Staff_new.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\attire_studentID.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Billing_new.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Checklist_new.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Clubs_new.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\CreditTransfer+AddDrop+ProgrammeTransfer.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\dress_code.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Fees_new.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Foundation_programme.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Freshman_guideline.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\grade+CGPA_cal+classification_of_honours.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Honour_list.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Master_Doctor_Programmes_new.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\MasterProgrammes_new.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\New_Intake_Guideline.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Program_Structure.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\requirement_courses.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\rules&regulation.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Scholarship.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\ussdc.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\UTAR_FICT_HANDBOOK-2021.pdf','C:\\Users\\Lohju\\Desktop\\FYP_DATA\\utarcare.pdf',
    'C:\\Users\\Lohju\\Desktop\\FYP_DATA\\Courses_Details.pdf'
]

# Initialize a dictionary to hold the text documents for each PDF
pdf_text_documents = {}

# Load each PDF and store its text documents in a separate list within the dictionary
for pdf_file in pdf_files:
    loader = PyPDFLoader(pdf_file)
    text_documents = loader.load()
    pdf_text_documents[pdf_file] = text_documents
```

Figure 5.4.6.1 Code snippet Load the data into a list inside a dictionary

The code snippet above is to load the file from its sources into the list named pdf_files. Then we extract the text from each file and store it inside a list in a dictionary.

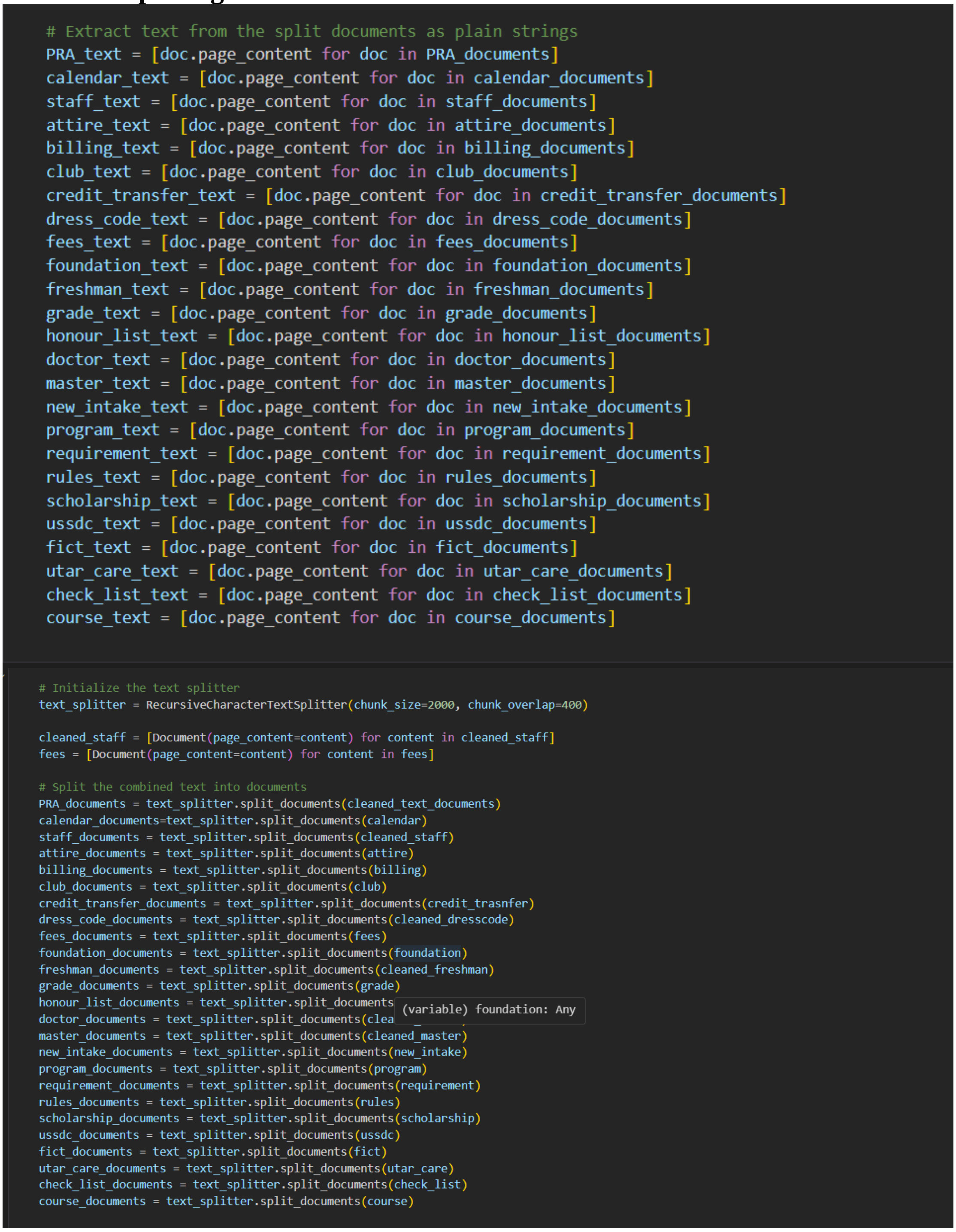


```
PRA= pdf_text_documents[pdf_files[0]]
calendar=pdf_text_documents[pdf_files[1]]
staff=pdf_text_documents[pdf_files[2]]
attire=pdf_text_documents[pdf_files[3]]
billing=pdf_text_documents[pdf_files[4]]
club=pdf_text_documents[pdf_files[5]]
credit_trasnfer=pdf_text_documents[pdf_files[6]]
dress_code=pdf_text_documents[pdf_files[7]]
fees=pdf_text_documents[pdf_files[8]]
foundation=pdf_text_documents[pdf_files[9]]
freshman=pdf_text_documents[pdf_files[10]]
grade=pdf_text_documents[pdf_files[11]]
honour_list=pdf_text_documents[pdf_files[12]]
doctor=pdf_text_documents[pdf_files[13]]
master=pdf_text_documents[pdf_files[14]]
new_intake=pdf_text_documents[pdf_files[15]]
program=pdf_text_documents[pdf_files[16]]
requirement=pdf_text_documents[pdf_files[17]]
rules=pdf_text_documents[pdf_files[18]]
scholarship=pdf_text_documents[pdf_files[19]]
ussdc=pdf_text_documents[pdf_files[20]]
fict=pdf_text_documents[pdf_files[21]]
utar_care=pdf_text_documents[pdf_files[22]]
check_list=pdf_text_documents[pdf_files[23]]
course=pdf_text_documents[pdf_files[24]]
```

Figure 5.4.6.2 Code snippet to store the list into variable for data processing purposes

The code snippet above is to store the content from the list in the dictionary into variable so that each file can be pre-processed separately as every file required different data preprocess method.

4.4.3 Data splitting

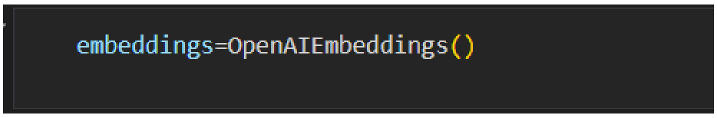


```
# Extract text from the split documents as plain strings
PRA_text = [doc.page_content for doc in PRA_documents]
calendar_text = [doc.page_content for doc in calendar_documents]
staff_text = [doc.page_content for doc in staff_documents]
attire_text = [doc.page_content for doc in attire_documents]
billing_text = [doc.page_content for doc in billing_documents]
club_text = [doc.page_content for doc in club_documents]
credit_transfer_text = [doc.page_content for doc in credit_transfer_documents]
dress_code_text = [doc.page_content for doc in dress_code_documents]
fees_text = [doc.page_content for doc in fees_documents]
foundation_text = [doc.page_content for doc in foundation_documents]
freshman_text = [doc.page_content for doc in freshman_documents]
grade_text = [doc.page_content for doc in grade_documents]
honour_list_text = [doc.page_content for doc in honour_list_documents]
doctor_text = [doc.page_content for doc in doctor_documents]
master_text = [doc.page_content for doc in master_documents]
new_intake_text = [doc.page_content for doc in new_intake_documents]
program_text = [doc.page_content for doc in program_documents]
requirement_text = [doc.page_content for doc in requirement_documents]
rules_text = [doc.page_content for doc in rules_documents]
scholarship_text = [doc.page_content for doc in scholarship_documents]
ussdc_text = [doc.page_content for doc in ussdc_documents]
fict_text = [doc.page_content for doc in fict_documents]
utar_care_text = [doc.page_content for doc in utar_care_documents]
check_list_text = [doc.page_content for doc in check_list_documents]
course_text = [doc.page_content for doc in course_documents]
```

```
# Initialize the text splitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=2000, chunk_overlap=400)

cleaned_staff = [Document(page_content=content) for content in cleaned_staff]
fees = [Document(page_content=content) for content in fees]

# Split the combined text into documents
PRA_documents = text_splitter.split_documents(cleaned_text_documents)
calendar_documents=text_splitter.split_documents(calendar)
staff_documents = text_splitter.split_documents(cleaned_staff)
attire_documents = text_splitter.split_documents(attire)
billing_documents = text_splitter.split_documents(billing)
club_documents = text_splitter.split_documents(club)
credit_transfer_documents = text_splitter.split_documents(credit_trasnfer)
dress_code_documents = text_splitter.split_documents(cleaned_dresscode)
fees_documents = text_splitter.split_documents(fees)
foundation_documents = text_splitter.split_documents(foundation)
freshman_documents = text_splitter.split_documents(cleaned_freshman)
grade_documents = text_splitter.split_documents(grade)
honour_list_documents = text_splitter.split_documents (variable) foundation: Any
doctor_documents = text_splitter.split_documents(clea
master_documents = text_splitter.split_documents(cleaned_master)
new_intake_documents = text_splitter.split_documents(new_intake)
program_documents = text_splitter.split_documents(program)
requirement_documents = text_splitter.split_documents(requirement)
rules_documents = text_splitter.split_documents(rules)
scholarship_documents = text_splitter.split_documents(scholarship)
ussdc_documents = text_splitter.split_documents(ussdc)
fict_documents = text_splitter.split_documents(fict)
utar_care_documents = text_splitter.split_documents(utar_care)
check_list_documents = text_splitter.split_documents(check_list)
course_documents = text_splitter.split_documents(course)
```

Figure 5.4.6.3 Code snippet to split the text into smaller chunk size

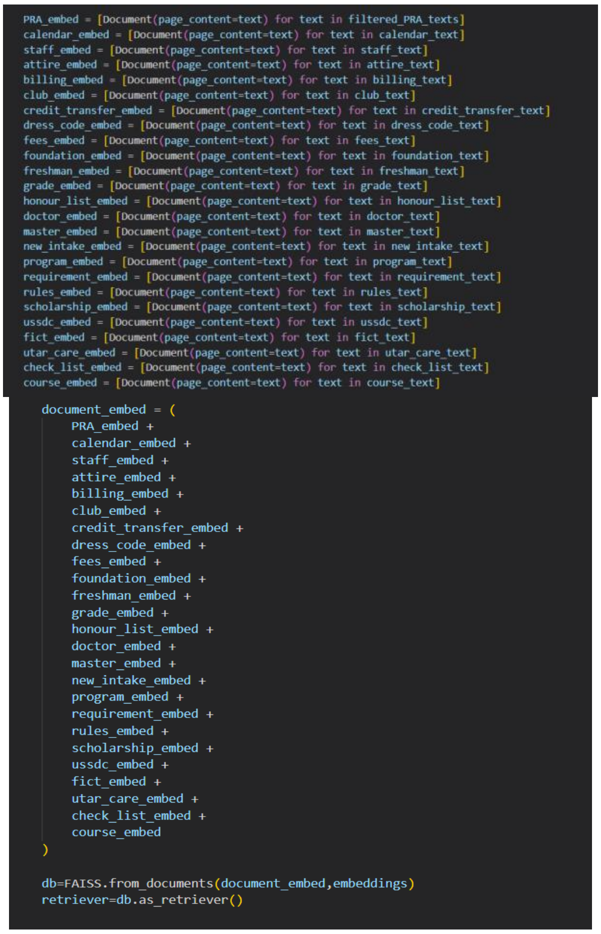
The code above is for splitting the data into smaller chunk size is important for efficient processing, especially in tasks like information retrieval or embedding generation. Smaller chunks allow for more details, and specific information retrieval, which helps when responding to user queries.



```
embeddings=OpenAIEmbeddings()
```

Figure 5.4.6.4 Code snippet to create embedder

The code above is to create the embedder to embed the training data and users queries.

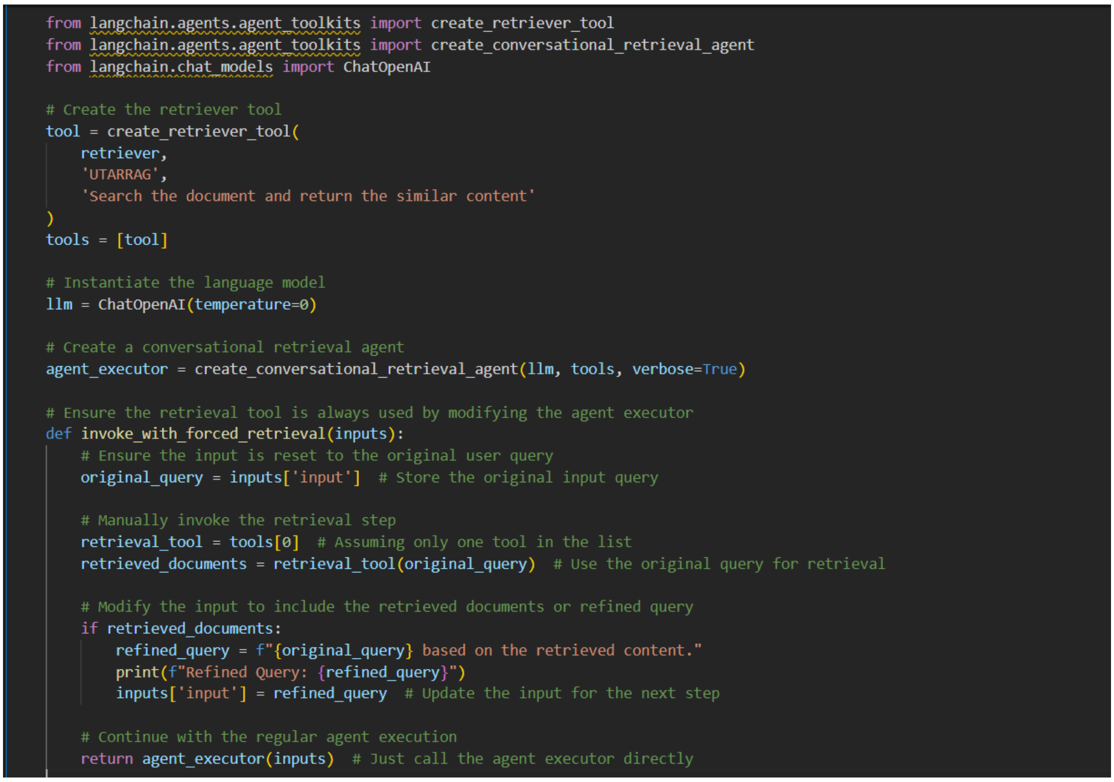


```
PRA_embed = [Document(page_content=text) for text in filtered_PRA_texts]
calendar_embed = [Document(page_content=text) for text in calendar_text]
staff_embed = [Document(page_content=text) for text in staff_text]
attire_embed = [Document(page_content=text) for text in attire_text]
billing_embed = [Document(page_content=text) for text in billing_text]
club_embed = [Document(page_content=text) for text in club_text]
credit_transfer_embed = [Document(page_content=text) for text in credit_transfer_text]
dress_code_embed = [Document(page_content=text) for text in dress_code_text]
fees_embed = [Document(page_content=text) for text in fees_text]
foundation_embed = [Document(page_content=text) for text in foundation_text]
freshman_embed = [Document(page_content=text) for text in freshman_text]
grade_embed = [Document(page_content=text) for text in grade_text]
honour_list_embed = [Document(page_content=text) for text in honour_list_text]
doctor_embed = [Document(page_content=text) for text in doctor_text]
master_embed = [Document(page_content=text) for text in master_text]
new_intake_embed = [Document(page_content=text) for text in new_intake_text]
program_embed = [Document(page_content=text) for text in program_text]
requirement_embed = [Document(page_content=text) for text in requirement_text]
rules_embed = [Document(page_content=text) for text in rules_text]
scholarship_embed = [Document(page_content=text) for text in scholarship_text]
ussdc_embed = [Document(page_content=text) for text in ussdc_text]
fict_embed = [Document(page_content=text) for text in fict_text]
utar_care_embed = [Document(page_content=text) for text in utar_care_text]
check_list_embed = [Document(page_content=text) for text in check_list_text]
course_embed = [Document(page_content=text) for text in course_text]
```

```
document_embed = (
    PRA_embed +
    calendar_embed +
    staff_embed +
    attire_embed +
    billing_embed +
    club_embed +
    credit_transfer_embed +
    dress_code_embed +
    fees_embed +
    foundation_embed +
    freshman_embed +
    grade_embed +
    honour_list_embed +
    doctor_embed +
    master_embed +
    new_intake_embed +
    program_embed +
    requirement_embed +
    rules_embed +
    scholarship_embed +
    ussdc_embed +
    fict_embed +
    utar_care_embed +
    check_list_embed +
    course_embed
)

db=FAISS.from_documents(document_embed,embeddings)
retriever=db.as_retriever()
```

Figure 5.4.6.5 Code snippet to store the embedded files into FAISS vector store

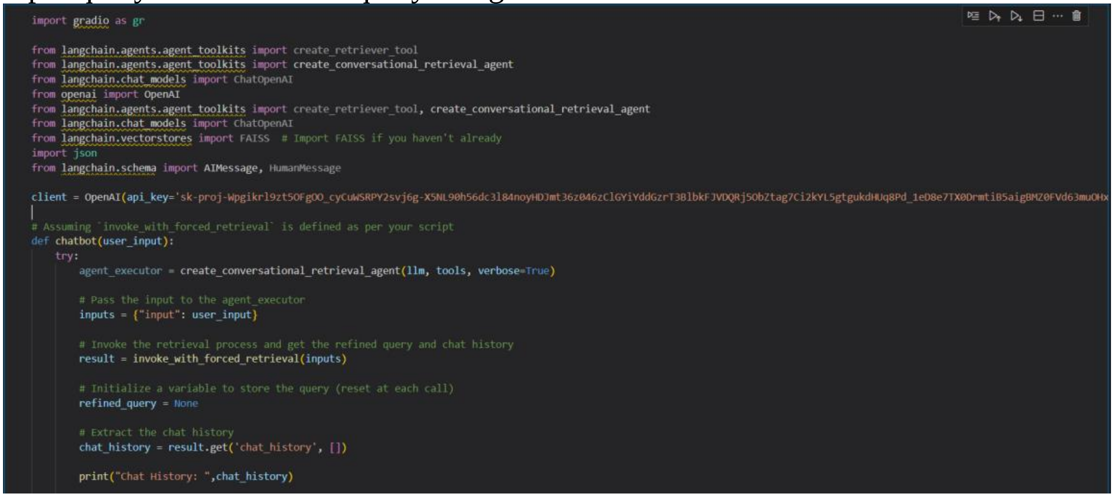
The embeddings is being combined into a single list. Then, it creates a FAISS index to store and retrieve the document embeddings, and set up a retriever for querying the embedded data efficiently.



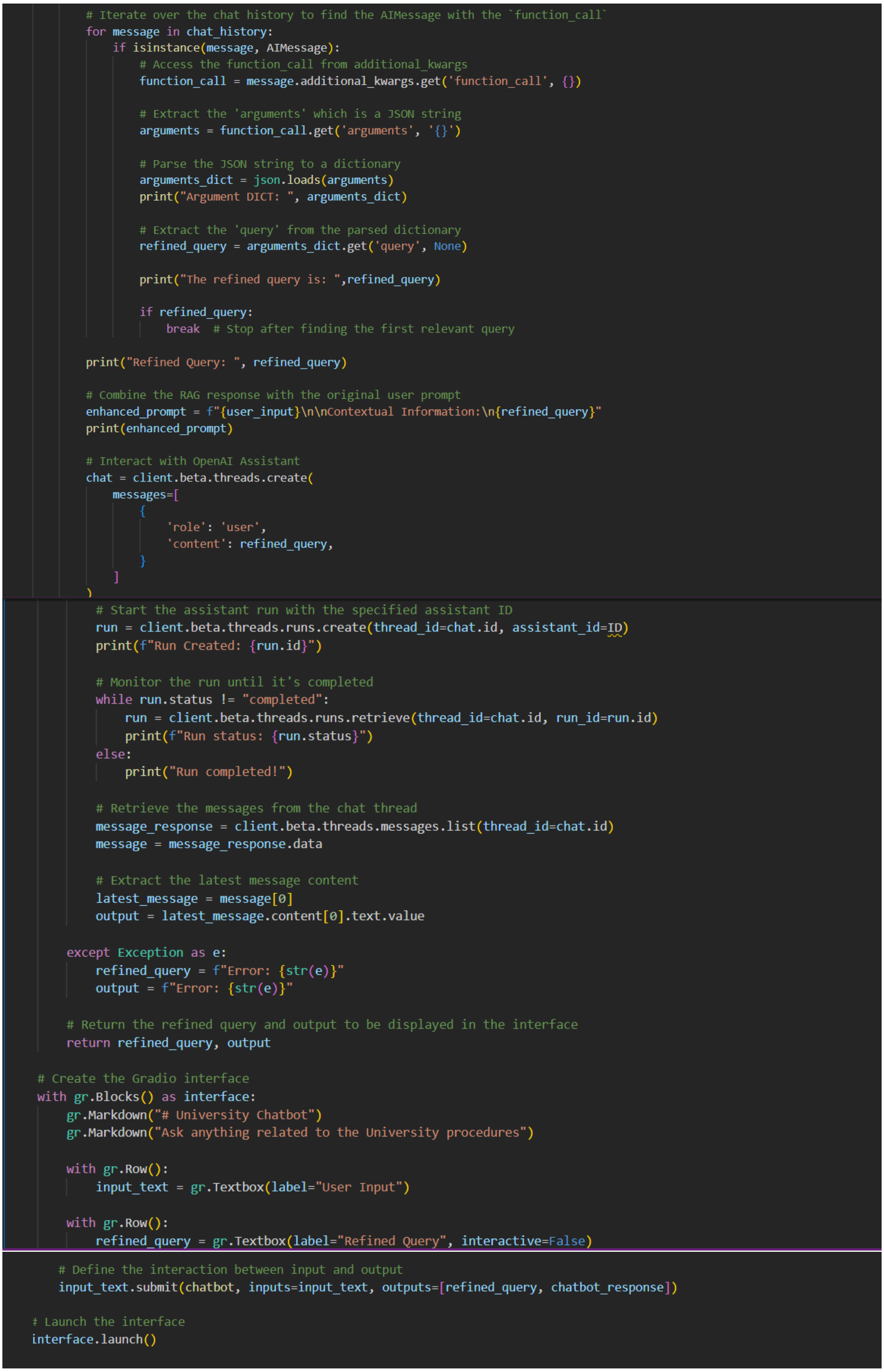
```
from langchain.agents.agent_toolkits import create_retriever_tool
from langchain.agents.agent_toolkits import create_conversational_retrieval_agent
from langchain.chat_models import ChatOpenAI

# Create the retriever tool
tool = create_retriever_tool(
    retriever,
    'UTARRAG',
    'Search the document and return the similar content'
)
tools = [tool]

# Instantiate the language model
llm = ChatOpenAI(temperature=0)

# Create a conversational retrieval agent
agent_executor = create_conversational_retrieval_agent(llm, tools, verbose=True)

# Ensure the retrieval tool is always used by modifying the agent executor
def invoke_with_forced_retrieval(inputs):
    # Ensure the input is reset to the original user query
    original_query = inputs['input']  # Store the original input query

    # Manually invoke the retrieval step
    retrieval_tool = tools[0]  # Assuming only one tool in the list
    retrieved_documents = retrieval_tool(original_query)  # Use the original query for retrieval

    # Modify the input to include the retrieved documents or refined query
    if retrieved_documents:
        refined_query = f"{original_query} based on the retrieved content."
        print(f"Refined Query: {refined_query}")
        inputs['input'] = refined_query  # Update the input for the next step

    # Continue with the regular agent execution
    return agent_executor(inputs)  # Just call the agent executor directly
```

Figure 5.4.6.6 Code snippet to create RAG agent

The code snippet above is to build a conversational retrieval agent which equipped with both LLM and the retrieval tool to handle conversations and retrieve relevant documents. It forced the retrieval tool to always fetch relevant documents based on the input query and refines the query using retrieved content.



```
import gradio as gr

from langchain.agents.agent_toolkits import create_retriever_tool
from langchain.agents.agent_toolkits import create_conversational_retrieval_agent
from langchain.chat_models import ChatOpenAI
from openai import OpenAI
from langchain.agents.agent_toolkits import create_retriever_tool, create_conversational_retrieval_agent
from langchain.chat_models import ChatOpenAI
from langchain.vectorstores import FAISS  # Import FAISS if you haven't already
import json
from langchain.schema import AIMessage, HumanMessage

client = OpenAI(api_key='sk-proj-Wpgikrl9zt5OFgOO_cyCuWSRPY2svj6g-X5NL90h56dc3l84noyHDJmt36z046zClGYiYddGzrT3BlbkFJVDQRj5ObZtag7Ci2kYL5gtgukdHUq8Pd_1eD8e7TX0DrmtiB5aigBMZ0FVd63muOHx

# Assuming `invoke_with_forced_retrieval` is defined as per your script
def chatbot(user_input):
    try:
        agent_executor = create_conversational_retrieval_agent(llm, tools, verbose=True)

        # Pass the input to the agent_executor
        inputs = {"input": user_input}

        # Invoke the retrieval process and get the refined query and chat history
        result = invoke_with_forced_retrieval(inputs)

        # Initialize a variable to store the query (reset at each call)
        refined_query = None

        # Extract the chat history
        chat_history = result.get('chat_history', [])

        print("Chat History: ",chat_history)
```



```
        # Iterate over the chat history to find the AIMessage with the `function_call`
        for message in chat_history:
            if isinstance(message, AIMessage):
                # Access the function_call from additional_kwargs
                function_call = message.additional_kwargs.get('function_call', {})

                # Extract the 'arguments' which is a JSON string
                arguments = function_call.get('arguments', '{}')

                # Parse the JSON string to a dictionary
                arguments_dict = json.loads(arguments)
                print("Argument DICT: ", arguments_dict)

                # Extract the 'query' from the parsed dictionary
                refined_query = arguments_dict.get('query', None)

                print("The refined query is: ",refined_query)

                if refined_query:
                    break  # Stop after finding the first relevant query

        print("Refined Query: ", refined_query)

        # Combine the RAG response with the original user prompt
        enhanced_prompt = f"{user_input}\n\nContextual Information:\n{refined_query}"
        print(enhanced_prompt)

        # Interact with OpenAI Assistant
        chat = client.beta.threads.create(
            messages=[
                {
                    'role': 'user',
                    'content': refined_query,
                }
            ]
        )
        # Start the assistant run with the specified assistant ID
        run = client.beta.threads.runs.create(thread_id=chat.id, assistant_id=ID)
        print(f"Run Created: {run.id}")

        # Monitor the run until it's completed
        while run.status != "completed":
            run = client.beta.threads.runs.retrieve(thread_id=chat.id, run_id=run.id)
            print(f"Run status: {run.status}")
        else:
            print("Run completed!")

        # Retrieve the messages from the chat thread
        message_response = client.beta.threads.messages.list(thread_id=chat.id)
        message = message_response.data

        # Extract the latest message content
        latest_message = message[0]
        output = latest_message.content[0].text.value

    except Exception as e:
        refined_query = f"Error: {str(e)}"
        output = f"Error: {str(e)}"

    # Return the refined query and output to be displayed in the interface
    return refined_query, output

# Create the Gradio interface
with gr.Blocks() as interface:
    gr.Markdown("# University Chatbot")
    gr.Markdown("Ask anything related to the University procedures")

    with gr.Row():
        input_text = gr.Textbox(label="User Input")

    with gr.Row():
        refined_query = gr.Textbox(label="Refined Query", interactive=False)
    # Define the interaction between input and output
    input_text.submit(chatbot, inputs=input_text, outputs=[refined_query, chatbot_response])

# Launch the interface
interface.launch()
```

Figure 5.4.6.7 Code snippet to launch the chatbot in Gradio interface

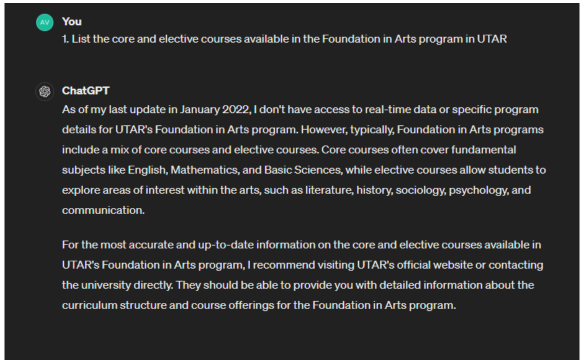
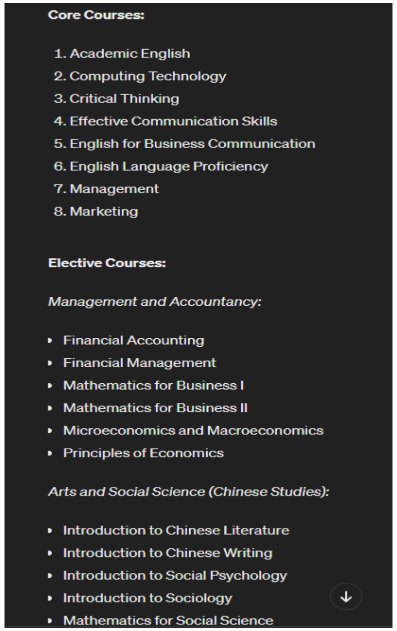
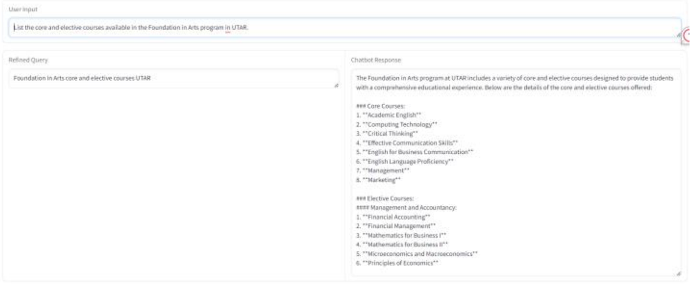
The code snippet here is to deploy the RAG chatbot in Gradio interface which accept the input from the users, and it will display the refined query and use the refined query to generate the output.

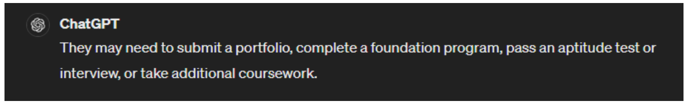
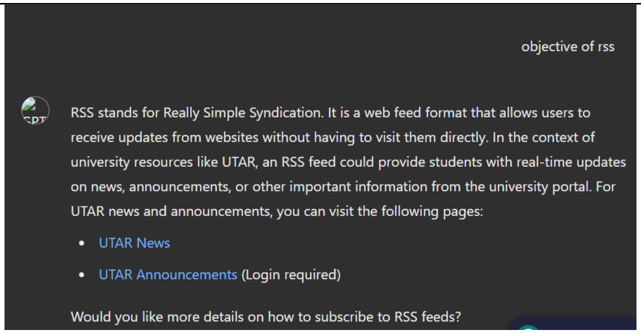
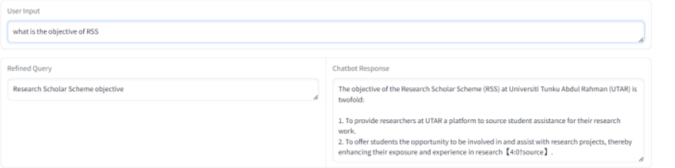
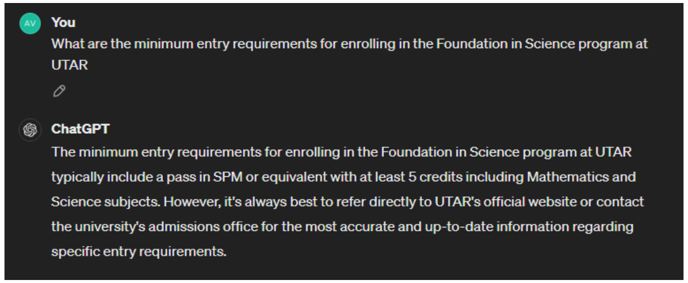
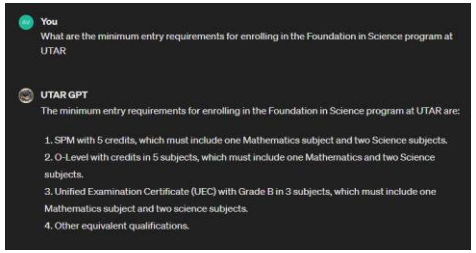
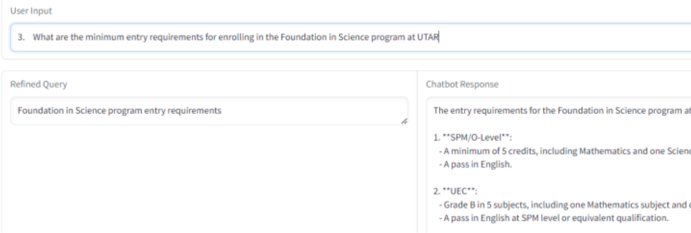
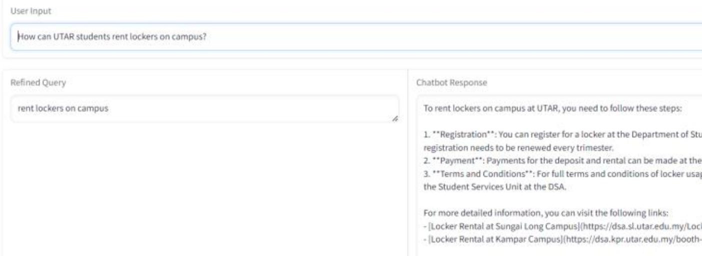
5.5 Preliminary Work Result

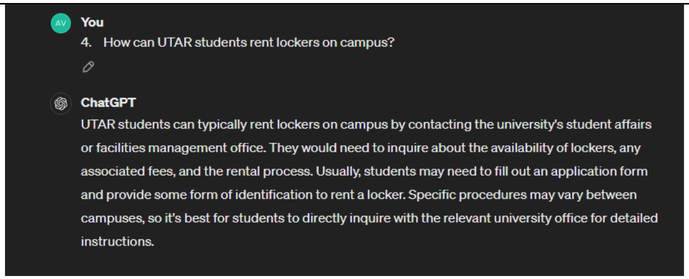
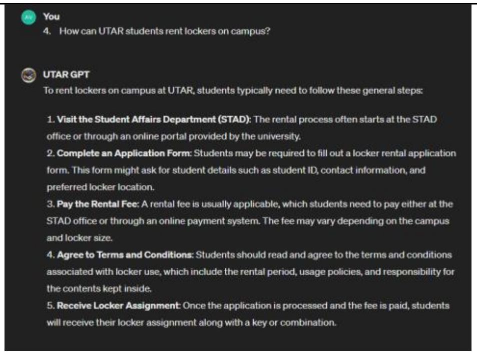
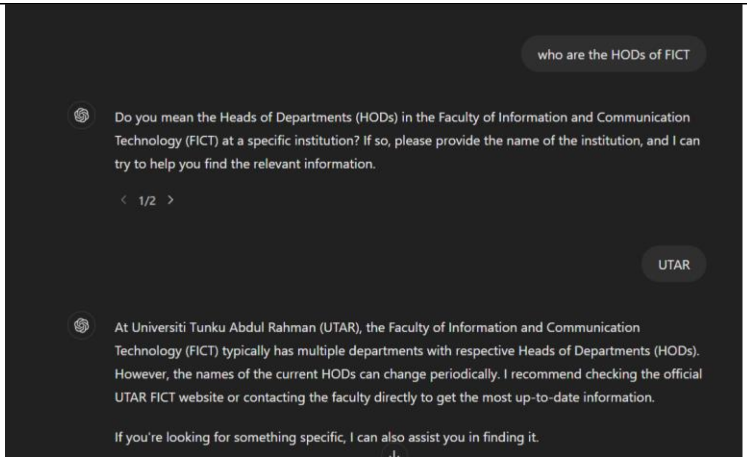
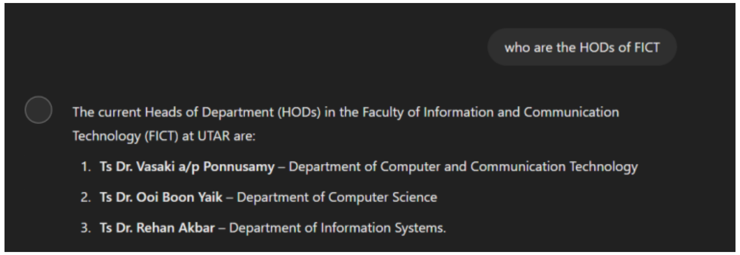
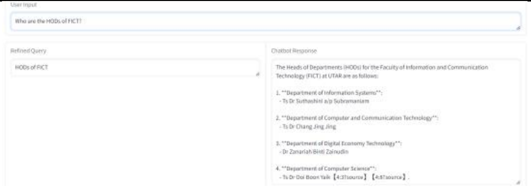
Table 5.5.1 List of questions

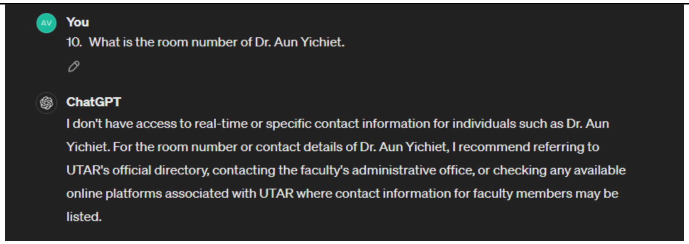
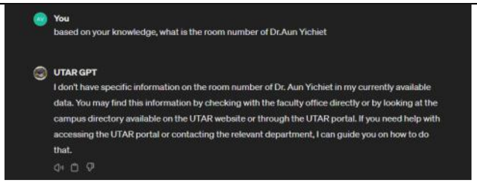
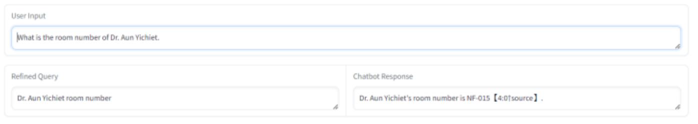
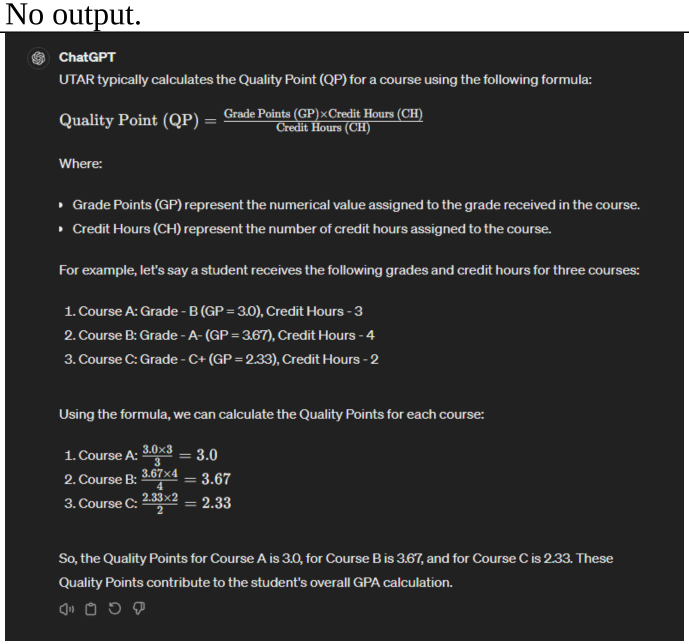
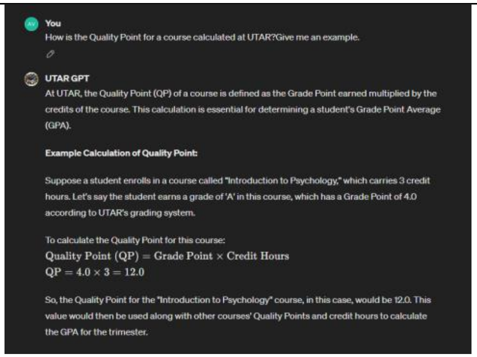
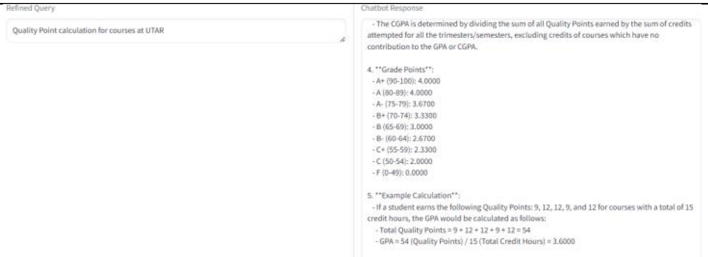
Question List
1. List the core and elective courses available in the Foundation in Arts program in UTAR. 2. Objective of RSS 3. What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR 4. How can UTAR students rent lockers on campus? 5. Who are the HODs of FICT? 6. What is the room number of Dr. Aun Yichiet. 7. How is the Quality Point for a course calculated at UTAR? Give me an example. 8. How does UTAR define the calculation of Grade Point Average (GPA) for a trimester? 9. What are the CGPA ranges for graduating with a Degree with Distinction, a Degree with Merit, and a regular Degree at UTAR? 10. Which types of courses are excluded from the calculation of GPA and CGPA at UTAR?

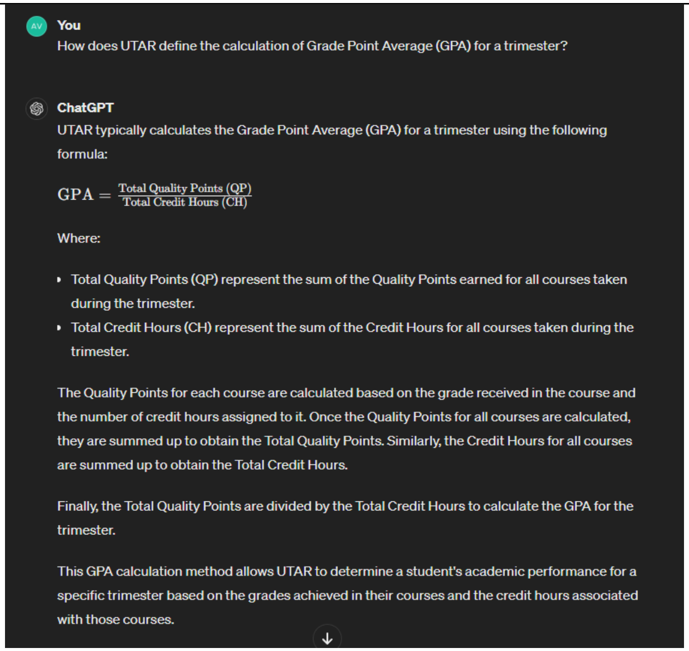
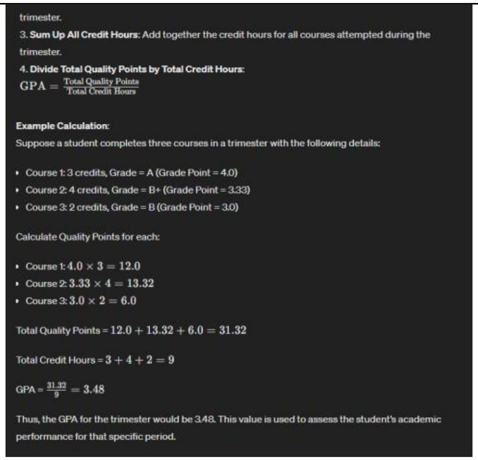
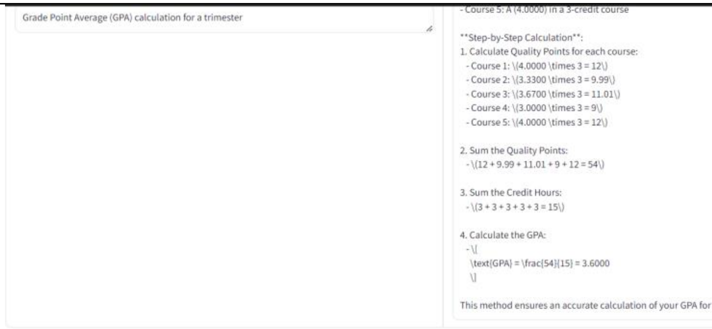
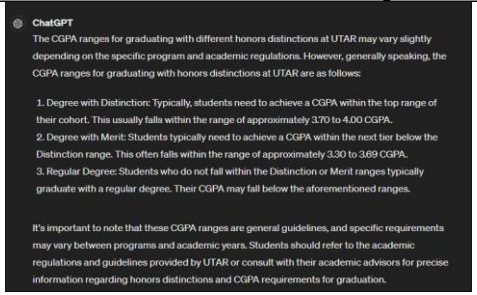
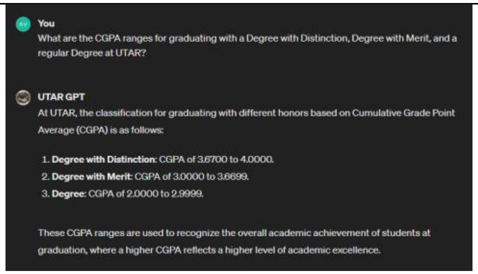
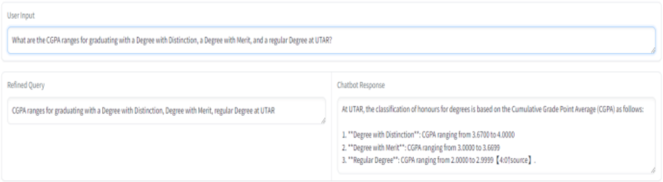
Table 5.5.2 Comparison between ChatGPT and UTARGPT

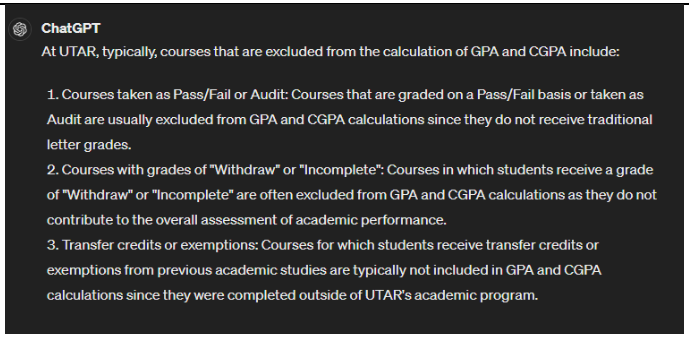
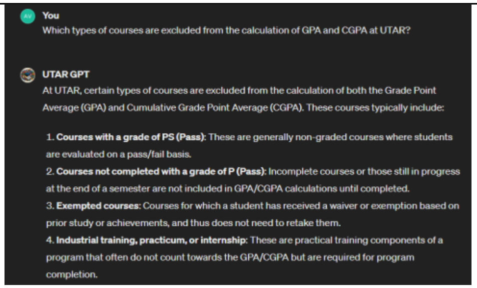
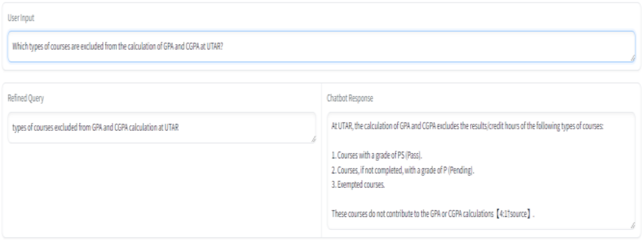
ChatGPT	GPT Builder UTARGPT	Assistant UTARGPT
 **You** 1. List the core and elective courses available in the Foundation in Arts program in UTAR **ChatGPT** As of my last update in January 2022, I don't have access to real-time data or specific program details for UTAR's Foundation in Arts program. However, typically, Foundation in Arts programs include a mix of core courses and elective courses. Core courses often cover fundamental subjects like English, Mathematics, and Basic Sciences, while elective courses allow students to explore areas of interest within the arts, such as literature, history, sociology, psychology, and communication. For the most accurate and up-to-date information on the core and elective courses available in UTAR's Foundation in Arts program, I recommend visiting UTAR's official website or contacting the university directly. They should be able to provide you with detailed information about the curriculum structure and course offerings for the Foundation in Arts program. Can't provide correct output.	 **Core Courses:** 1. Academic English 2. Computing Technology 3. Critical Thinking 4. Effective Communication Skills 5. English for Business Communication 6. English Language Proficiency 7. Management 8. Marketing **Elective Courses:** *Management and Accountancy:* - Financial Accounting - Financial Management - Mathematics for Business I - Mathematics for Business II - Microeconomics and Macroeconomics - Principles of Economics *Arts and Social Science (Chinese Studies):* - Introduction to Chinese Literature - Introduction to Chinese Writing - Introduction to Social Psychology - Introduction to Sociology - Mathematics for Social Science Correct output is given.	 User Input: List the core and elective courses available in the Foundation in Arts program in UTAR. Refined Query: Foundation in Arts core and elective courses UTAR Chatbot Response: The Foundation in Arts program at UTAR includes a variety of core and elective courses designed to provide students with a comprehensive educational experience. Below are the details of the core and elective courses offered: ### Core Courses: 1. **Academic English** 2. **Computing Technology** 3. **Critical Thinking** 4. **Effective Communication Skills** 5. **English for Business Communication** 6. **English Language Proficiency** 7. **Management** 8. **Marketing** ### Elective Courses: #### Management and Accountancy: 1. **Financial Accounting** 2. **Financial Management** 3. **Mathematics for Business I** 4. **Mathematics for Business II** 5. **Microeconomics and Macroeconomics** 6. **Principles of Economics**

 **ChatGPT** They may need to submit a portfolio, complete a foundation program, pass an aptitude test or interview, or take additional coursework. Wrong output.	 objective of rss RSS stands for Really Simple Syndication. It is a web feed format that allows users to receive updates from websites without having to visit them directly. In the context of university resources like UTAR, an RSS feed could provide students with real-time updates on news, announcements, or other important information from the university portal. For UTAR news and announcements, you can visit the following pages: • UTAR News • UTAR Announcements (Login required) Would you like more details on how to subscribe to RSS feeds? Wrong output.	 User Input what is the objective of RSS Refined Query Research Scholar Scheme objective Chatbot Response The objective of the Research Scholar Scheme (RSS) at Universiti Tunku Abdul Rahman (UTAR) is twofold: 1. To provide researchers at UTAR a platform to source student assistance for their research work. 2. To offer students the opportunity to be involved in and assist with research projects, thereby enhancing their exposure and experience in research【4:0†source】. Correct output.
 **You** What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR **ChatGPT** The minimum entry requirements for enrolling in the Foundation in Science program at UTAR typically include a pass in SPM or equivalent with at least 5 credits including Mathematics and Science subjects. However, it's always best to refer directly to UTAR's official website or contact the university's admissions office for the most accurate and up-to-date information regarding specific entry requirements. Only provide SPM requirement	 **You** What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR **UTAR GPT** The minimum entry requirements for enrolling in the Foundation in Science program at UTAR are: 1. SPM with 5 credits, which must include one Mathematics subject and two Science subjects. 2. O-Level with credits in 5 subjects, which must include one Mathematics and two Science subjects. 3. Unified Examination Certificate (UEC) with Grade B in 3 subjects, which must include one Mathematics subject and two science subjects. 4. Other equivalent qualifications. Provide correct answer.	 User Input 3. What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR Refined Query Foundation in Science program entry requirements Chatbot Response The entry requirements for the Foundation in Science program at 1. **SPM/O-Level**: - A minimum of 5 credits, including Mathematics and one Scienc - A pass in English. 2. **UEC**: - Grade B in 5 subjects, including one Mathematics subject and - A pass in English at SPM level or equivalent qualification. Provide correct answer.
		 User Input How can UTAR students rent lockers on campus? Refined Query rent lockers on campus Chatbot Response To rent lockers on campus at UTAR, you need to follow these steps: 1. **Registration**: You can register for a locker at the Department of Stu registration needs to be renewed every trimester. 2. **Payment**: Payments for the deposit and rental can be made at the 3. **Terms and Conditions**: For full terms and conditions of locker usag the Student Services Unit at the DSA. For more detailed information, you can visit the following links: - [Locker Rental at Sungai Long Campus](https://dsa.sl.utar.edu.my/Loc - [Locker Rental at Kampar Campus](https://dsa.kpr.utar.edu.my/booth-

 **You** 4. How can UTAR students rent lockers on campus? **ChatGPT** UTAR students can typically rent lockers on campus by contacting the university's student affairs or facilities management office. They would need to inquire about the availability of lockers, any associated fees, and the rental process. Usually, students may need to fill out an application form and provide some form of identification to rent a locker. Specific procedures may vary between campuses, so it's best for students to directly inquire with the relevant university office for detailed instructions. Can't provide answer	 **You** 4. How can UTAR students rent lockers on campus? **UTAR GPT** To rent lockers on campus at UTAR, students typically need to follow these general steps: 1. **Visit the Student Affairs Department (STAD):** The rental process often starts at the STAD office or through an online portal provided by the university. 2. **Complete an Application Form**: Students may be required to fill out a locker rental application form. This form might ask for student details such as student ID, contact information, and preferred locker location. 3. **Pay the Rental Fee:** A rental fee is usually applicable, which students need to pay either at the STAD office or through an online payment system. The fee may vary depending on the campus and locker size. 4. **Agree to Terms and Conditions:** Students should read and agree to the terms and conditions associated with locker use, which include the rental period, usage policies, and responsibility for the contents kept inside. 5. **Receive Locker Assignment**: Once the application is processed and the fee is paid, students will receive their locker assignment along with a key or combination. Correct output	Correct output
 who are the HODs of FICT Do you mean the Heads of Departments (HODs) in the Faculty of Information and Communication Technology (FICT) at a specific institution? If so, please provide the name of the institution, and I can try to help you find the relevant information. 1/2 UTAR At Universiti Tunku Abdul Rahman (UTAR), the Faculty of Information and Communication Technology (FICT) typically has multiple departments with respective Heads of Departments (HODs). However, the names of the current HODs can change periodically. I recommend checking the official UTAR FICT website or contacting the faculty directly to get the most up-to-date information. If you're looking for something specific, I can also assist you in finding it. Wrong answer	 who are the HODs of FICT The current Heads of Department (HODs) in the Faculty of Information and Communication Technology (FICT) at UTAR are: 1. **Ts Dr. Vasaki a/p Ponnusamy** – Department of Computer and Communication Technology 2. **Ts Dr. Ooi Boon Yaik** – Department of Computer Science 3. **Ts Dr. Rehan Akbar** – Department of Information Systems. Outdated info.	 User Input Who are the HODs of FICT? Refined Query HODs of FICT Chatbot Response The Heads of Departments (HODs) for the Faculty of Information and Communication Technology (FICT) at UTAR are as follows: 1. **Department of Information Systems**: - Ts Dr Suthashini a/p Subramaniam 2. **Department of Computer and Communication Technology**: - Ts Dr Chang Jing Jing 3. **Department of Digital Economy Technology**: - Dr Zanariah Binti Zainudin 4. **Department of Computer Science**: - Ts Dr Ooi Boon Yaik 【4:3†source】 【4:8†source】. Correct answer

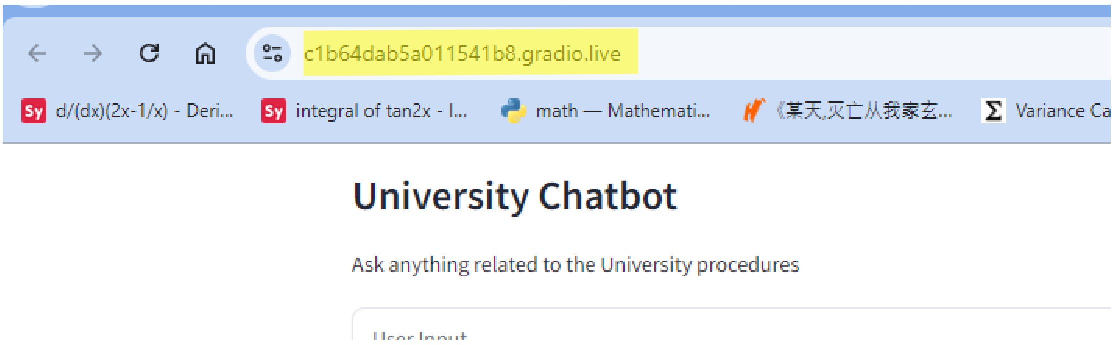
 **You** 10. What is the room number of Dr. Aun Yichiet. **ChatGPT** I don't have access to real-time or specific contact information for individuals such as Dr. Aun Yichiet. For the room number or contact details of Dr. Aun Yichiet, I recommend referring to UTAR's official directory, contacting the faculty's administrative office, or checking any available online platforms associated with UTAR where contact information for faculty members may be listed. No output.	 **You** based on your knowledge, what is the room number of Dr.Aun Yichiet **UTAR GPT** I don't have specific information on the room number of Dr. Aun Yichiet in my currently available data. You may find this information by checking with the faculty office directly or by looking at the campus directory available on the UTAR website or through the UTAR portal. If you need help with accessing the UTAR portal or contacting the relevant department, I can guide you on how to do that. No output, but provide guidance to student to get the information.	 User Input What is the room number of Dr. Aun Yichiet. Refined Query Dr. Aun Yichiet room number Chatbot Response Dr. Aun Yichiet's room number is NF-015【4:0†source】. Correct output
 **ChatGPT** UTAR typically calculates the Quality Point (QP) for a course using the following formula: $\text{Quality Point (QP)} = \frac{\text{Grade Points (GP)} \times \text{Credit Hours (CH)}}{\text{Credit Hours (CH)}}$ Where: • Grade Points (GP) represent the numerical value assigned to the grade received in the course. • Credit Hours (CH) represent the number of credit hours assigned to the course. For example, let's say a student receives the following grades and credit hours for three courses: 1. Course A: Grade - B (GP = 3.0), Credit Hours - 3 2. Course B: Grade - A- (GP = 3.67), Credit Hours - 4 3. Course C: Grade - C+ (GP = 2.33), Credit Hours - 2 Using the formula, we can calculate the Quality Points for each course: 1. Course A: $\frac{3.0\times3}{3} = 3.0$ 2. Course B: $\frac{3.67\times4}{4} = 3.67$ 3. Course C: $\frac{2.33\times2}{2} = 2.33$ So, the Quality Points for Course A is 3.0, for Course B is 3.67, and for Course C is 2.33. These Quality Points contribute to the student's overall GPA calculation. Incorrect answer, it shows the calculation for GPA.	 **You** How is the Quality Point for a course calculated at UTAR?Give me an example. **UTAR GPT** At UTAR, the Quality Point (QP) of a course is defined as the Grade Point earned multiplied by the credits of the course. This calculation is essential for determining a student's Grade Point Average (GPA). **Example Calculation of Quality Point:** Suppose a student enrolls in a course called "Introduction to Psychology," which carries 3 credit hours. Let's say the student earns a grade of 'A' in this course, which has a Grade Point of 4.0 according to UTAR's grading system. To calculate the Quality Point for this course: $\text{Quality Point (QP)} = \text{Grade Point} \times \text{Credit Hours}$ $\text{QP} = 4.0 \times 3 = 12.0$ So, the Quality Point for the "Introduction to Psychology" course, in this case, would be 12.0. This value would then be used along with other courses' Quality Points and credit hours to calculate the GPA for the trimester. Correct answer.	 Refined Query Quality Point calculation for courses at UTAR Chatbot Response - The CGPA is determined by dividing the sum of all Quality Points earned by the sum of credits attempted for all the trimesters/semesters, excluding credits of courses which have no contribution to the GPA or CGPA. 4. **Grade Points**: - A+ (90-100): 4.0000 - A (80-89): 4.0000 - A- (75-79): 3.6700 - B+ (70-74): 3.3300 - B (65-69): 3.0000 - B- (60-64): 2.6700 - C+ (55-59): 2.3300 - C (50-54): 2.0000 - F (0-49): 0.0000 5. **Example Calculation**: - If a student earns the following Quality Points: 9, 12, 12, 9, and 12 for courses with a total of 15 credit hours, the GPA would be calculated as follows: - Total Quality Points = 9 + 12 + 12 + 9 + 12 = 54 - GPA = 54 (Quality Points) / 15 (Total Credit Hours) = 3.6000 Correct output.

 **You** How does UTAR define the calculation of Grade Point Average (GPA) for a trimester? **ChatGPT** UTAR typically calculates the Grade Point Average (GPA) for a trimester using the following formula: $\text{GPA} = \frac{\text{Total Quality Points (QP)}}{\text{Total Credit Hours (CH)}}$ Where: - Total Quality Points (QP) represent the sum of the Quality Points earned for all courses taken during the trimester. - Total Credit Hours (CH) represent the sum of the Credit Hours for all courses taken during the trimester. The Quality Points for each course are calculated based on the grade received in the course and the number of credit hours assigned to it. Once the Quality Points for all courses are calculated, they are summed up to obtain the Total Quality Points. Similarly, the Credit Hours for all courses are summed up to obtain the Total Credit Hours. Finally, the Total Quality Points are divided by the Total Credit Hours to calculate the GPA for the trimester. This GPA calculation method allows UTAR to determine a student's academic performance for a specific trimester based on the grades achieved in their courses and the credit hours associated with those courses. Correct answer without example.	 trimester. 3. **Sum Up All Credit Hours:** Add together the credit hours for all courses attempted during the trimester. 4. **Divide Total Quality Points by Total Credit Hours:** $\text{GPA} = \frac{\text{Total Quality Points}}{\text{Total Credit Hours}}$ **Example Calculation:** Suppose a student completes three courses in a trimester with the following details: - Course 1: 3 credits, Grade = A (Grade Point = 4.0) - Course 2: 4 credits, Grade = B+ (Grade Point = 3.33) - Course 3: 2 credits, Grade = B (Grade Point = 3.0) Calculate Quality Points for each: - Course 1: $4.0 \times 3 = 12.0$ - Course 2: $3.33 \times 4 = 13.32$ - Course 3: $3.0 \times 2 = 6.0$ Total Quality Points = $12.0 + 13.32 + 6.0 = 31.32$ Total Credit Hours = $3 + 4 + 2 = 9$ GPA = $\frac{31.32}{9} = 3.48$ Thus, the GPA for the trimester would be 3.48. This value is used to assess the student's academic performance for that specific period. Correct answer with example, student can more easier to understand.	 Grade Point Average (GPA) calculation for a trimester - Course 5: A (4.0000) in a 3-credit course **Step-by-Step Calculation**: 1. Calculate Quality Points for each course: - Course 1: \(4.0000 \times 3 = 12\) - Course 2: \(3.3300 \times 3 = 9.99\) - Course 3: \(3.6700 \times 3 = 11.01\) - Course 4: \(3.0000 \times 3 = 9\) - Course 5: \(4.0000 \times 3 = 12\) 2. Sum the Quality Points: - \(12 + 9.99 + 11.01 + 9 + 12 = 54\) 3. Sum the Credit Hours: - \(3 + 3 + 3 + 3 + 3 = 15\) 4. Calculate the GPA: - \[\text{GPA} = \frac{54}{15} = 3.6000 \] This method ensures an accurate calculation of your GPA for Correct output.
 **ChatGPT** The CGPA ranges for graduating with different honors distinctions at UTAR may vary slightly depending on the specific program and academic regulations. However, generally speaking, the CGPA ranges for graduating with honors distinctions at UTAR are as follows: 1. Degree with Distinction: Typically, students need to achieve a CGPA within the top range of their cohort. This usually falls within the range of approximately 3.70 to 4.00 CGPA. 2. Degree with Merit: Students typically need to achieve a CGPA within the next tier below the Distinction range. This often falls within the range of approximately 3.30 to 3.69 CGPA. 3. Regular Degree: Students who do not fall within the Distinction or Merit ranges typically graduate with a regular degree. Their CGPA may fall below the aforementioned ranges. It's important to note that these CGPA ranges are general guidelines, and specific requirements may vary between programs and academic years. Students should refer to the academic regulations and guidelines provided by UTAR or consult with their academic advisors for precise information regarding honors distinctions and CGPA requirements for graduation. Incorrect answer, degree with distinction should from 3.67-4.00.	 **You** What are the CGPA ranges for graduating with a Degree with Distinction, Degree with Merit, and a regular Degree at UTAR? **UTAR GPT** At UTAR, the classification for graduating with different honors based on Cumulative Grade Point Average (CGPA) is as follows: 1. **Degree with Distinction:** CGPA of 3.6700 to 4.0000. 2. **Degree with Merit:** CGPA of 3.0000 to 3.6699. 3. **Degree:** CGPA of 2.0000 to 2.9999. These CGPA ranges are used to recognize the overall academic achievement of students at graduation, where a higher CGPA reflects a higher level of academic excellence. Correct answer.	 User Input What are the CGPA ranges for graduating with a Degree with Distinction, a Degree with Merit, and a regular Degree at UTAR? Refined Query CGPA ranges for graduating with a Degree with Distinction, Degree with Merit, regular Degree at UTAR Chatbot Response At UTAR, the classification of honours for degrees is based on the Cumulative Grade Point Average (CGPA) as follows: 1. **Degree with Distinction**: CGPA ranging from 3.6700 to 4.0000 2. **Degree with Merit**: CGPA ranging from 3.0000 to 3.6699 3. **Regular Degree**: CGPA ranging from 2.0000 to 2.9999【4:0†source】. Correct output.

 **ChatGPT** At UTAR, typically, courses that are excluded from the calculation of GPA and CGPA include: 1. Courses taken as Pass/Fail or Audit: Courses that are graded on a Pass/Fail basis or taken as Audit are usually excluded from GPA and CGPA calculations since they do not receive traditional letter grades. 2. Courses with grades of "Withdraw" or "Incomplete": Courses in which students receive a grade of "Withdraw" or "Incomplete" are often excluded from GPA and CGPA calculations as they do not contribute to the overall assessment of academic performance. 3. Transfer credits or exemptions: Courses for which students receive transfer credits or exemptions from previous academic studies are typically not included in GPA and CGPA calculations since they were completed outside of UTAR's academic program. Correct answer.	 **You** Which types of courses are excluded from the calculation of GPA and CGPA at UTAR? **UTAR GPT** At UTAR, certain types of courses are excluded from the calculation of both the Grade Point Average (GPA) and Cumulative Grade Point Average (CGPA). These courses typically include: 1. **Courses with a grade of PS (Pass):** These are generally non-graded courses where students are evaluated on a pass/fail basis. 2. **Courses not completed with a grade of P (Pass):** Incomplete courses or those still in progress at the end of a semester are not included in GPA/CGPA calculations until completed. 3. **Exempted courses:** Courses for which a student has received a waiver or exemption based on prior study or achievements, and thus does not need to retake them. 4. **Industrial training, practicum, or internship:** These are practical training components of a program that often do not count towards the GPA/CGPA but are required for program completion. Correct answer.	 User Input Which types of courses are excluded from the calculation of GPA and CGPA at UTAR? Refined Query types of courses excluded from GPA and CGPA calculation at UTAR Chatbot Response At UTAR, the calculation of GPA and CGPA excludes the results/credit hours of the following types of courses: 1. Courses with a grade of PS (Pass). 2. Courses, if not completed, with a grade of P (Pending). 3. Exempted courses. These courses do not contribute to the GPA or CGPA calculations 【4:1†source】. Correct answer.

5.5 System Operation (with screenshot)

5.5.1 Use the Assistant UTARGPT



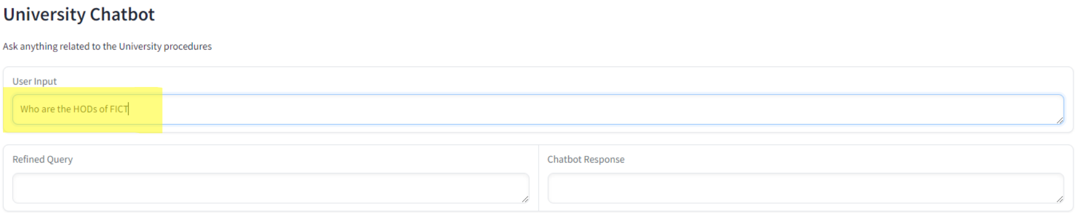
c1b64dab5a011541b8.gradio.live

University Chatbot

Ask anything related to the University procedures

Figure 5.5.1.1 UTARGPT share link

The Assistant UTARGPT is accessed by using the share link above. Anyone with the share link can access the chatbot by pasting the share link in their web browser.

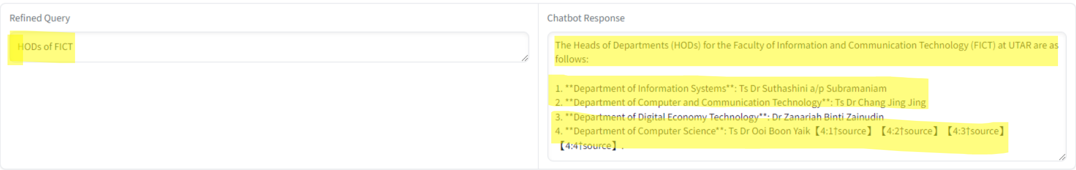

University Chatbot

Ask anything related to the University procedures

User Input: Who are the HODs of FICT

Figure 5.5.1.2 UTARGPT User Interface

After we have enter the share link, we will see the user interface as shown as above. The users can enter their queries in the chat box and press “Enter” and the refined query will be generated.


Refined Query: HODs of FICT

Chatbot Response

The Heads of Departments (HODs) for the Faculty of Information and Communication Technology (FICT) at UTAR are as follows:

1. **Department of Information Systems**: Ts Dr Suthashini a/p Subramaniam
2. **Department of Computer and Communication Technology**: Ts Dr Chang Jing Jing
3. **Department of Digital Economy Technology**: Dr Zanariah Binti Zainudin
4. **Department of Computer Science**: Ts Dr Ooi Boon Yaik 【4:1†source】 【4:2†source】 【4:3†source】 【4:4†source】 .

Figure 5.5.1.3 UTARGPT refined query and output

After the users pressed “Enter”, the refined output will be generated on the left inside the refined query column while the output will be generated in the right column.

5.6 Implementation issues and challenges

In this project, several implementation issues and challenges were faced during the implementation phase. Since the GPT builder is a new tool just launched in 2023, many of its functionalities is still under exploration. The implementation issues and challenges will be further explained below:

First and foremost, the chatbot can’t read the data from the websites in real-time. This issue is not significant when dealing with static data which the data will not be changed for a long period of time. However, the issue becomes more pronounced with the dynamic input data from the UTAR websites which will change very frequently such as the daily announcements from the university. For these kinds of data, we will need to schedule regular maintenance to update the knowledge of the chatbot in order to make sure our chatbot is up to date. This could impose a burden to the developer team as the daily announcement might require the developer team to perform the update every day. It requires continuous attention from the developers and could lead to rising operating expenses. This strategy is required to keep the chatbot working, but it also emphasizes the need for a more automated solution that can handle the dynamic input data more effectively. This can improve the system’s overall sustainability and cost-effectiveness. However, GPT builder doesn’t offer this feature up to this point.

Furthermore, the GPT builder can’t read and process the image input data. Although the ChatGPT has to ability to analyze the image input from the message box, this feature is not available in the GPT builder as the GPT-4 is a text-based model, the ChatGPT can read the images is due to it has been integrated with the image processing component. Inability in study the image type data can affect the performance of the chatbot as some of the input data is in image form. To solve this problem, the technique of computer vision has been applied in the data collection stage. Text from the images will be extracted and generated into a file. Thus, the chatbot will then has the ability to understand the information that is being delivered from the images. However, this approach is only applicable to the images that contain the text. Some important images such as the floor plan of the school is unable to be study by the chatbot as it doesn’t contain any text.

Besides that, the GPT builder only can upload a maximum of 25 files as its knowledge to train the chatbot. It might not be sufficient for us the upload all the information from the UTAR websites if we want to categorize the information into different file. To solve this problem, we need to merge all the information from the UTAR websites into a single file or a maximum of 25 files. However, this will lead to a problem in maintaining the chatbot as the training data is not documented, we will take a long time in figure out the content that we were looking for to modify or to perform data normalization.

Moreover, the GPT builder provides less accurate answers as it consumes more files especially when dealing with the merge files. These files are the combination of several files together to solve the problem of insufficient storage for uploading the files. By merging the file, we can provide more files as knowledge to the chatbot. However, this reduced the accuracy of the chatbot. The chatbot loses its accuracy to the queries related to the content in the merge file. For instance, inside the merged file, “PRA_RSS_PRS.pdf” has clearly stated the “ Objective of RSS” but when the query is being asked to the chatbot, it provides incorrect information to the users.

Last but not least, the chatbot does not offer API for the developers to integrate it seamlessly into their own applications and websites. This limitation not only restricts its potential use in various platforms, where developers might want to leverage its capabilities within their customized workflows or products but it also hinders the possibility of improving the performance of the chatbot. For instance, without the API access, the chatbot cannot be integrated with the advanced system like RAG system, which could significantly improve the performance of the chatbot to generate more accurate outputs. Allowing the support for APIs would provide for more flexibility and enhancement opportunities.

However, the above issues have been solved by using the OpenAI Assistant to build a chatbot. The OpenAI Assistant chatbot offer the API call functions to the developers to integrate the chatbot into their application and websites. By having the API calling function, we can integrate the RAG system with the OpenAI Assistant chatbot to improve its performance and accuracy of the response.

5.7 Concluding remark

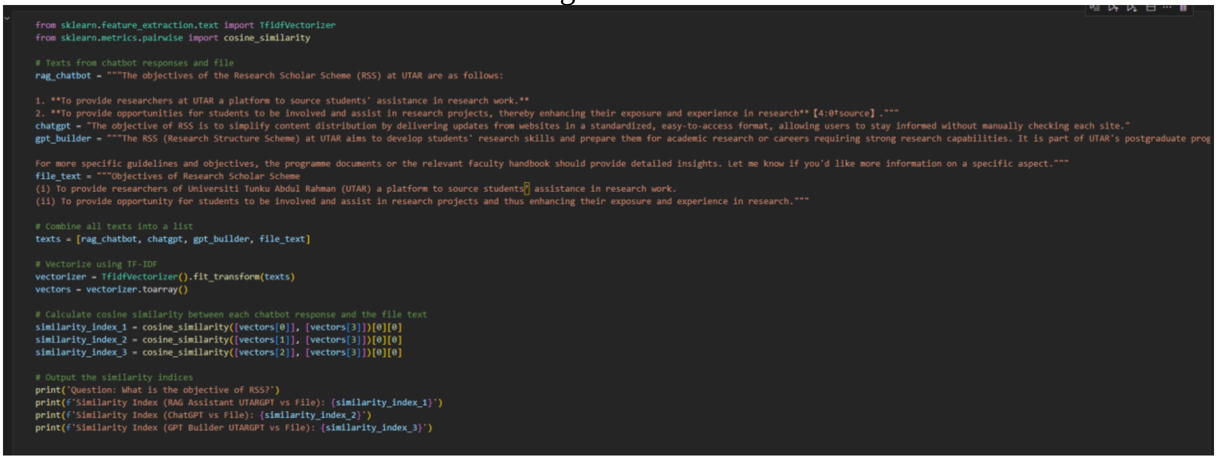
In conclusion, there are some requirements needed to be setup for this project in order to ensure that the system is able to execute as expected. Preliminary works has been done to build the RAG System, GPT Builder UTARGPT and Assistant UTARGPT. However, there are some implementation issues and challenges exist in this project. Firstly, the chatbot cannot read the data from websites in real time, the GPT builder only can train with text-based data, so additional work is required to trained the GPT builder with image data, the GPT builder only can accept up to maximum of 25 files as its knowledge which lead to merging of file is required but this action cause another issue whereby the accuracy drop for merge file, and the GPT builder chatbot does not offer API for integration. However, all these issues has been improved in the OpenAI Assistant RAG chatbot.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

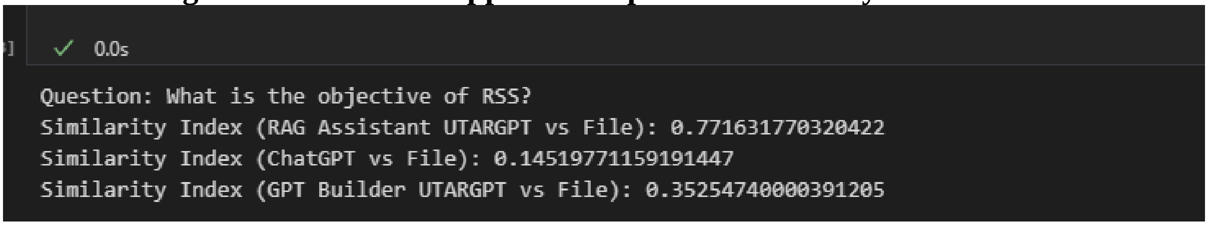
6.1.1 Compute similarity index

Compute similarity index is a technique to compute the similarity between two files. If the similarity index is high, it indicates that both files are having similar content. This evaluation method is useful to detect how similar the answer from the UTARGPT with the training data. This evaluation method is done by using the python code. We utilize the cosine similarity from the sklearn library to compute the similarity between the answer from the UTARGPT and the training file.



```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Texts from chatbot responses and file
rag_chatbot = """The objectives of the Research Scholar Scheme (RSS) at UTAR are as follows:

1. **To provide researchers at UTAR a platform to source students' assistance in research work.**
2. **To provide opportunities for students to be involved and assist in research projects, thereby enhancing their exposure and experience in research** [4:0†source] ."""
chatgpt = "The objective of RSS is to simplify content distribution by delivering updates from websites in a standardized, easy-to-access format, allowing users to stay informed without manually checking each site."
gpt_builder = """The RSS (Research Structure Scheme) at UTAR aims to develop students' research skills and prepare them for academic research or careers requiring strong research capabilities. It is part of UTAR's postgraduate prog

For more specific guidelines and objectives, the programme documents or the relevant faculty handbook should provide detailed insights. Let me know if you'd like more information on a specific aspect."""
file_text = """Objectives of Research Scholar Scheme
(i) To provide researchers of Universiti Tunku Abdul Rahman (UTAR) a platform to source students’ assistance in research work.
(ii) To provide opportunity for students to be involved and assist in research projects and thus enhancing their exposure and experience in research."""

# Combine all texts into a list
texts = [rag_chatbot, chatgpt, gpt_builder, file_text]

# Vectorize using TF-IDF
vectorizer = TfidfVectorizer().fit_transform(texts)
vectors = vectorizer.toarray()

# Calculate cosine similarity between each chatbot response and the file text
similarity_index_1 = cosine_similarity([vectors[0]], [vectors[3]])[0][0]
similarity_index_2 = cosine_similarity([vectors[1]], [vectors[3]])[0][0]
similarity_index_3 = cosine_similarity([vectors[2]], [vectors[3]])[0][0]

# Output the similarity indices
print('Question: What is the objective of RSS?')
print(f'Similarity Index (RAG Assistant UTARGPT vs File): {similarity_index_1}')
print(f'Similarity Index (ChatGPT vs File): {similarity_index_2}')
print(f'Similarity Index (GPT Builder UTARGPT vs File): {similarity_index_3}')
```

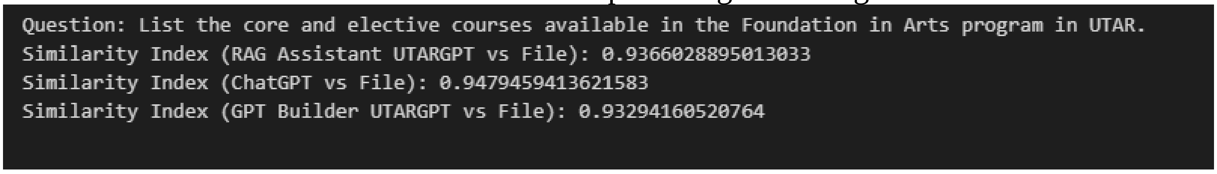
Figure 6.1.1.1 Code snippet to compute the similarity index



```
Question: What is the objective of RSS?
Similarity Index (RAG Assistant UTARGPT vs File): 0.771631770320422
Similarity Index (ChatGPT vs File): 0.14519771159191447
Similarity Index (GPT Builder UTARGPT vs File): 0.35254740000391205
```

Figure 6.1.1.2 Similarity index of Question: What is the objective of RSS?

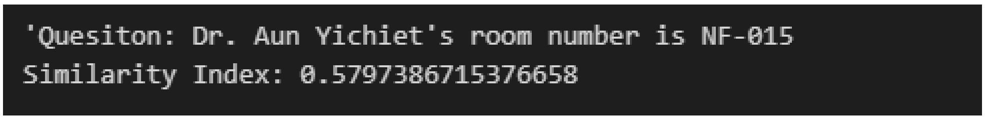
From above, we can notice that the similarity is high for Assistant UTARGPT while low for the other since the other two chatbot is providing the wrong answer.



```
Question: List the core and elective courses available in the Foundation in Arts program in UTAR.
Similarity Index (RAG Assistant UTARGPT vs File): 0.9366028895013033
Similarity Index (ChatGPT vs File): 0.9479459413621583
Similarity Index (GPT Builder UTARGPT vs File): 0.93294160520764
```

Figure 6.1.1.3 Similarity index of Question: Core and elective courses

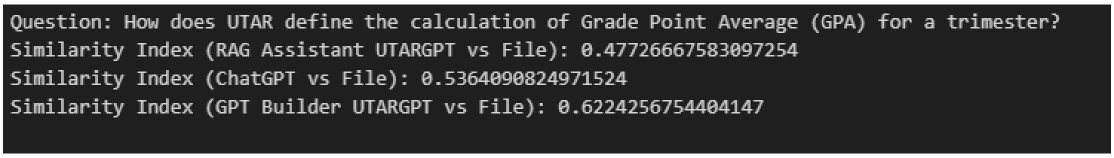
All three chatbots provide a high similarity index since all the answers are correct.



```
'Quesiton: Dr. Aun Yichiet's room number is NF-015
Similarity Index: 0.5797386715376658
```

Figure 6.1.1.4 Similarity index of Question: Dr Aun Yichiet room number

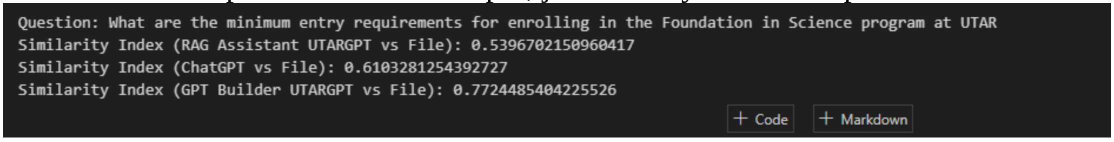
Since the other chatbot can’t provide the room number, the code snippet only computes the similarity index of the RAG chatbot. The answer generated is correct but due to the way of the answer is provided, it results in moderate similarity index.



```
Question: How does UTAR define the calculation of Grade Point Average (GPA) for a trimester?
Similarity Index (RAG Assistant UTARGPT vs File): 0.47726667583097254
Similarity Index (ChatGPT vs File): 0.5364090824971524
Similarity Index (GPT Builder UTARGPT vs File): 0.6224256754404147
```

Figure 6.1.1.5 Similarity index of Question: calculation of GPA

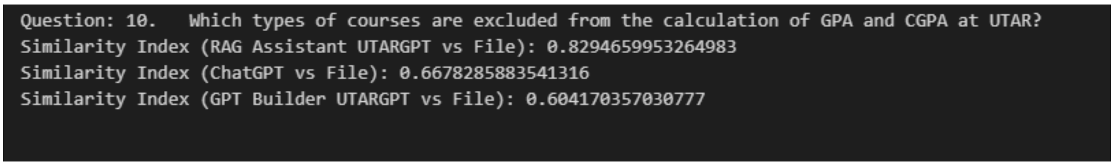
All three chatbot provided correct output, just the way of their response is different.



```
Question: What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR
Similarity Index (RAG Assistant UTARGPT vs File): 0.5396702150960417
Similarity Index (ChatGPT vs File): 0.6103281254392727
Similarity Index (GPT Builder UTARGPT vs File): 0.7724485404225526
```

Figure 6.1.1.6 Similarity index of Question: minimum entry requirement in Foundation

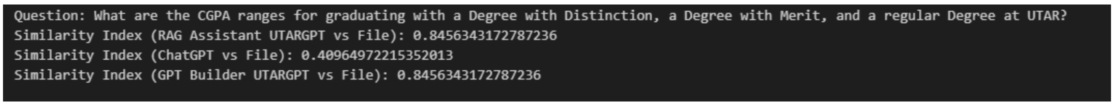
All three chatbots provided correct output, just the way of their response is different while GPT Builder UTARGPT has the highest index.



```
Question: 10.   Which types of courses are excluded from the calculation of GPA and CGPA at UTAR?
Similarity Index (RAG Assistant UTARGPT vs File): 0.8294659953264983
Similarity Index (ChatGPT vs File): 0.6678285883541316
Similarity Index (GPT Builder UTARGPT vs File): 0.604170357030777
```

Figure 6.1.1.7 Similarity index of Question: courses excluded from calculation of GPA and CGPA

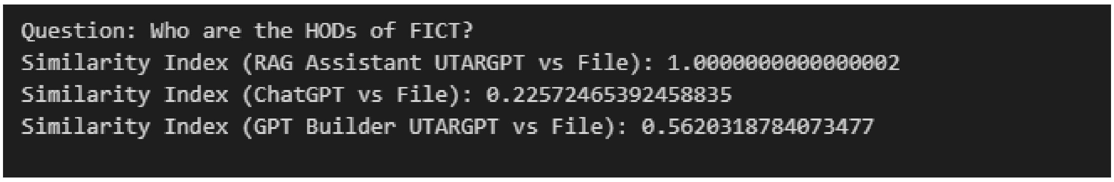
All three chatbots provided correct output, just the way of their response is different while Assistant UTARGPT has the highest index.



```
Question: What are the CGPA ranges for graduating with a Degree with Distinction, a Degree with Merit, and a regular Degree at UTAR?
Similarity Index (RAG Assistant UTARGPT vs File): 0.8456343172787236
Similarity Index (ChatGPT vs File): 0.40964972215352013
Similarity Index (GPT Builder UTARGPT vs File): 0.8456343172787236
```

Figure 6.1.1.8 Similarity index of Question: CGPA range for Degree with Distinction, Degree with Merit and regular Degree

Both Assistant UTARGPT and GPT Builder UTARGPT provided high similar index while ChatGPT provided low similarity index. The UTARGPT provided the correct answer while ChatGPT did not.



```
Question: Who are the HODs of FICT?
Similarity Index (RAG Assistant UTARGPT vs File): 1.0000000000000002
Similarity Index (ChatGPT vs File): 0.22572465392458835
Similarity Index (GPT Builder UTARGPT vs File): 0.5620318784073477
```

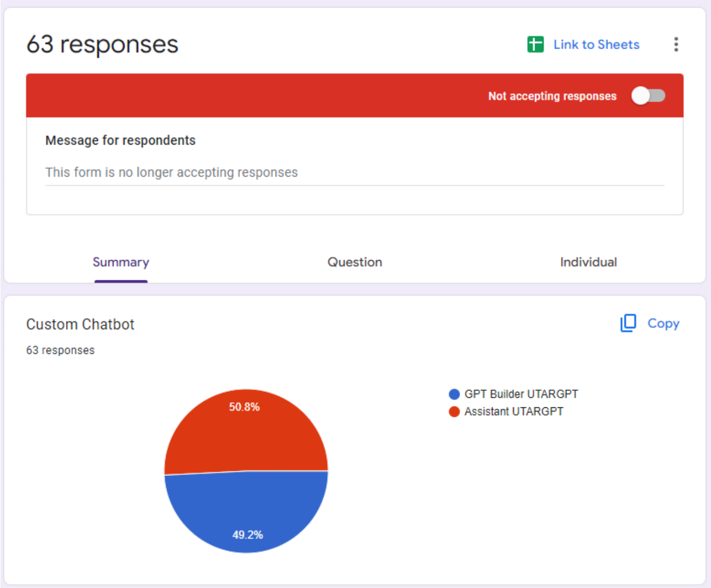
Figure 6.1.1.9 Similarity index of Question: Who are the HODs of FICT?

The Assistant UTARGPT provided an extremely high similarity index while the other did not. It is the only chatbot that provides the correct answer.

6.1.2 A/B testing

A/B testing also known as split testing is a method used to compare two versions of something to determine which one provides the better performance based on a set of metrics. In our case, the respondents are split into 2 group A and B. Group A is using the GPT Builder UTARGPT while Group B is using Assistant UTARGPT to answer the same sets of questions.

A total of 63 respondents have been involved in this survey where 31 of them are in group A (GPT Builder UTARGPT) and the remaining 32 are in group B (Assistant UTARGPT).



63 responses

Not accepting responses

Message for respondents

This form is no longer accepting responses

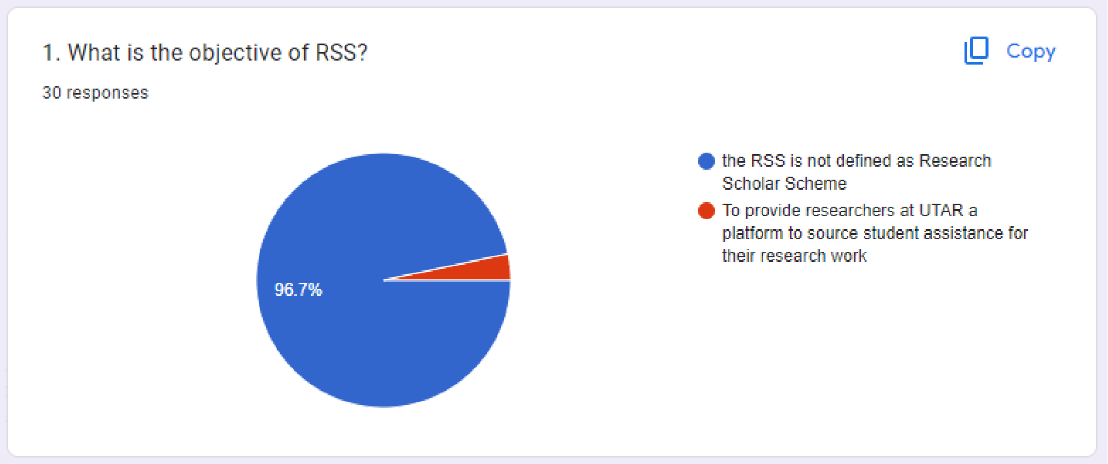
Custom Chatbot

63 responses

● GPT Builder UTARGPT
● Assistant UTARGPT

Figure 6.1.2.1 Number of respondents in groups A and B

Group A:



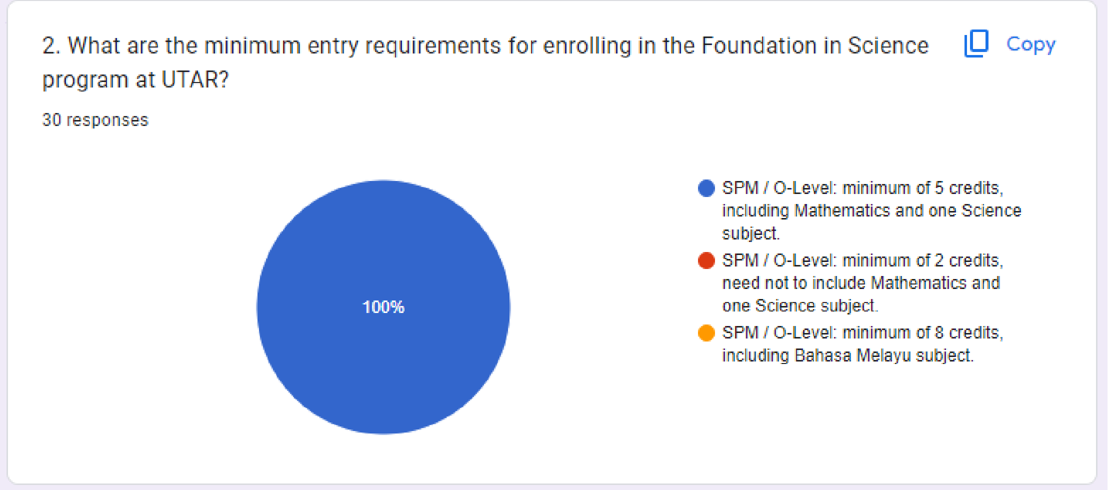
1. What is the objective of RSS?

30 responses

● the RSS is not defined as Research Scholar Scheme
● To provide researchers at UTAR a platform to source student assistance for their research work

Figure 6.1.2.2 Percentage breakdown of Question 1

The correct answer should be “to provide researchers at UTAR a platform to source student assistance for their research work”. The GPT Builder UTARGPT provided the wrong answer most of the time. It is only a 3.3% chances for it to provide the correct answer.



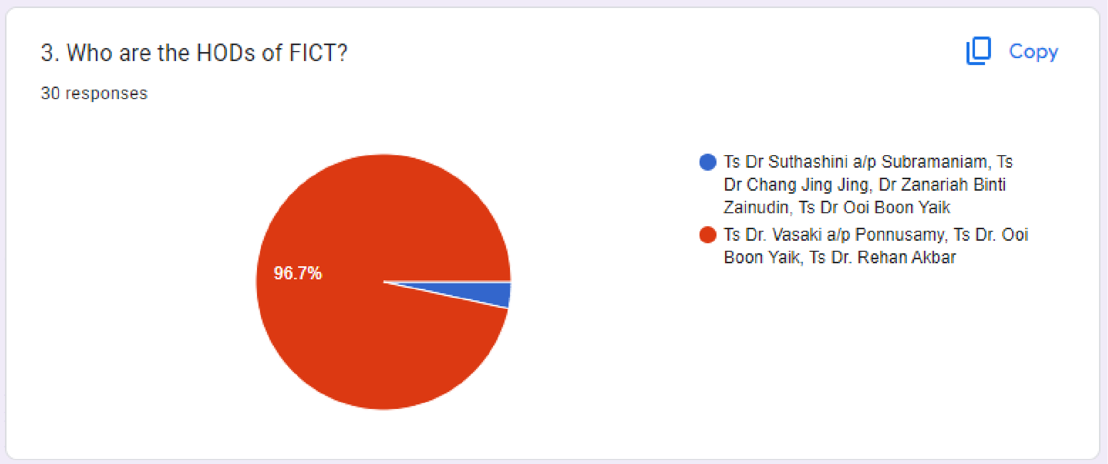
2. What are the minimum entry requirements for enrolling in the Foundation in Science program at UTAR?

30 responses

● SPM / O-Level: minimum of 5 credits, including Mathematics and one Science subject.
● SPM / O-Level: minimum of 2 credits, need not to include Mathematics and one Science subject.
● SPM / O-Level: minimum of 8 credits, including Bahasa Melayu subject.

Figure 6.1.2.3 Percentage breakdown of Question 2

The GPT builder UTARGPT able to provide correct answer for all the respondents.



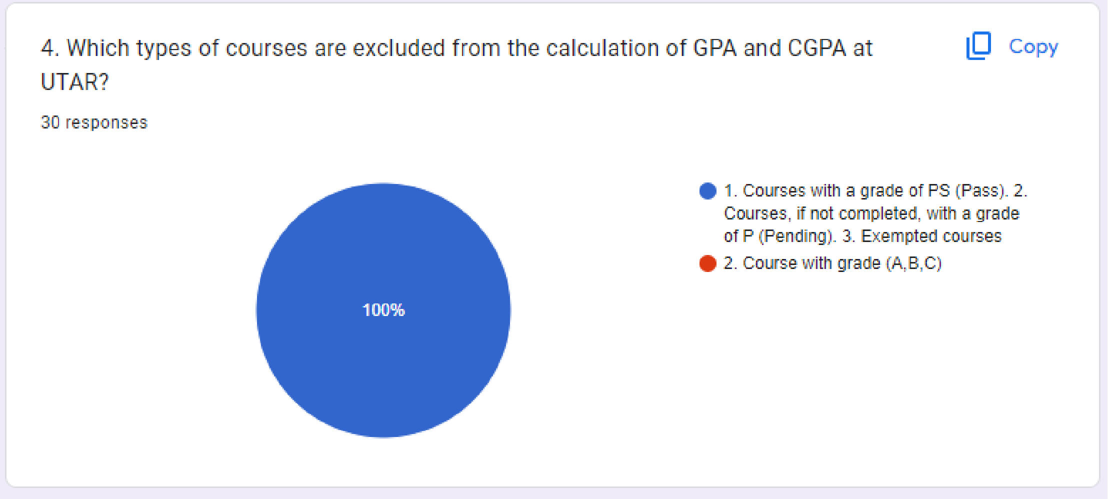
3. Who are the HODs of FICT?

30 responses

● Ts Dr Suthashini a/p Subramaniam, Ts Dr Chang Jing Jing, Dr Zanariah Binti Zainudin, Ts Dr Ooi Boon Yaik
● Ts Dr. Vasaki a/p Ponnusamy, Ts Dr. Ooi Boon Yaik, Ts Dr. Rehan Akbar

Figure 6.1.2.4 Percentage breakdown of Question 3

The GPT builder UTARGPT provided the outdated HODs most of the time.



4. Which types of courses are excluded from the calculation of GPA and CGPA at UTAR?

30 responses

● 1. Courses with a grade of PS (Pass). 2. Courses, if not completed, with a grade of P (Pending). 3. Exempted courses
● 2. Course with grade (A,B,C)

Figure 6.1.2.5 Percentage breakdown of Question 4

GPT builder UTARGPT able to provide the correct answer all the time.

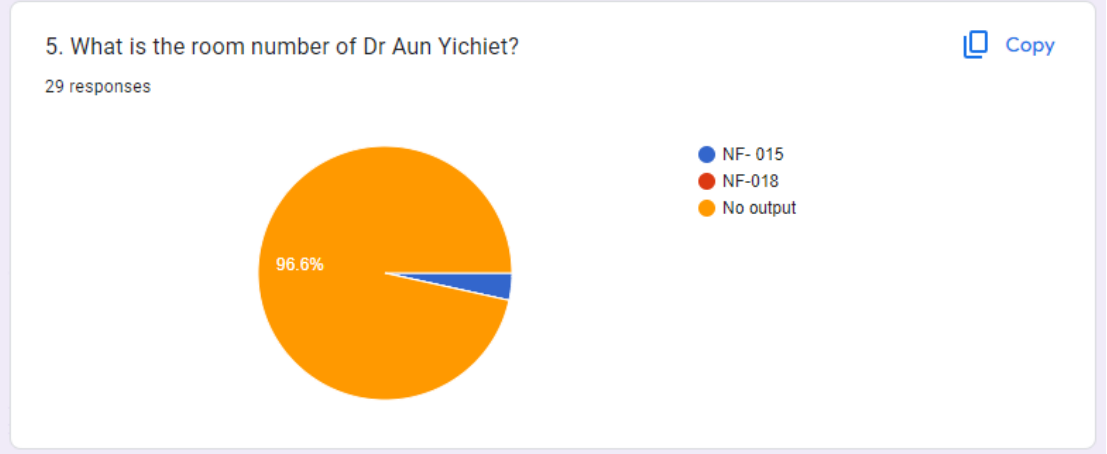


Figure 6.1.2.6 Percentage breakdown of Question 5

GPT builder UTARGPT is unable to provide the room number most of the time.

Group B:

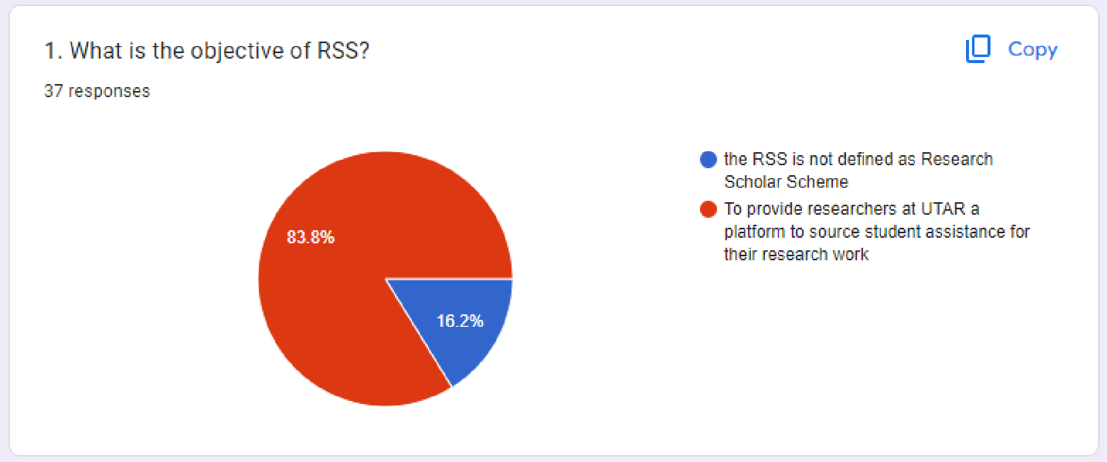


Figure 6.1.2.7 Percentage breakdown of Question 1

Assistant UTARGPT is able to provide correct answers most of the time. (83.8%)

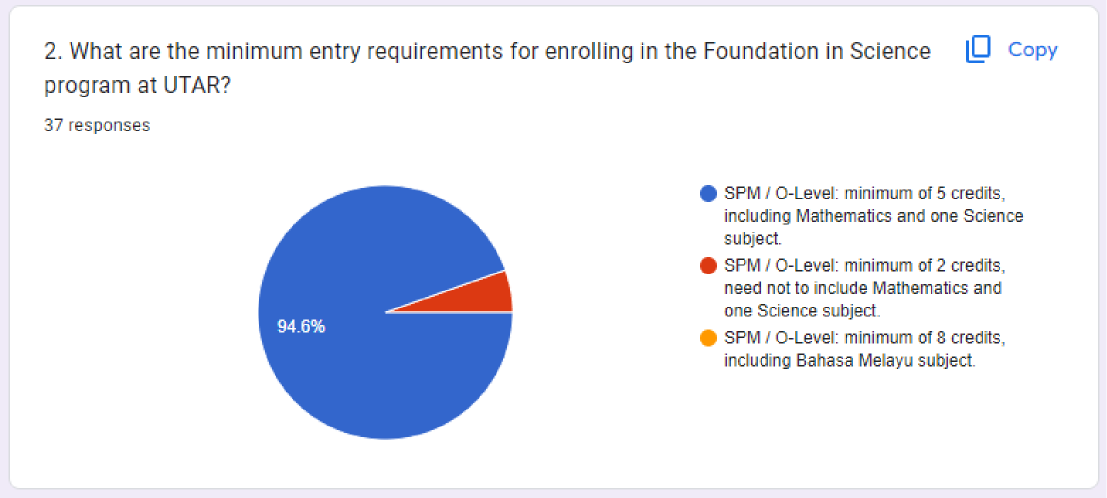
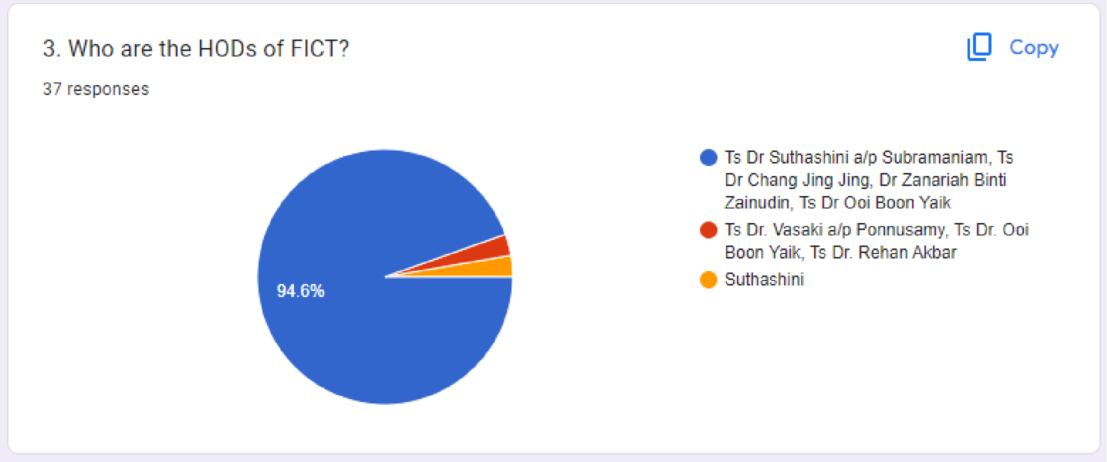


Figure 6.1.2.8 Percentage breakdown of Question 2

Assistant UTARGPT is able to provide correct answers most of the time. (94.6%)



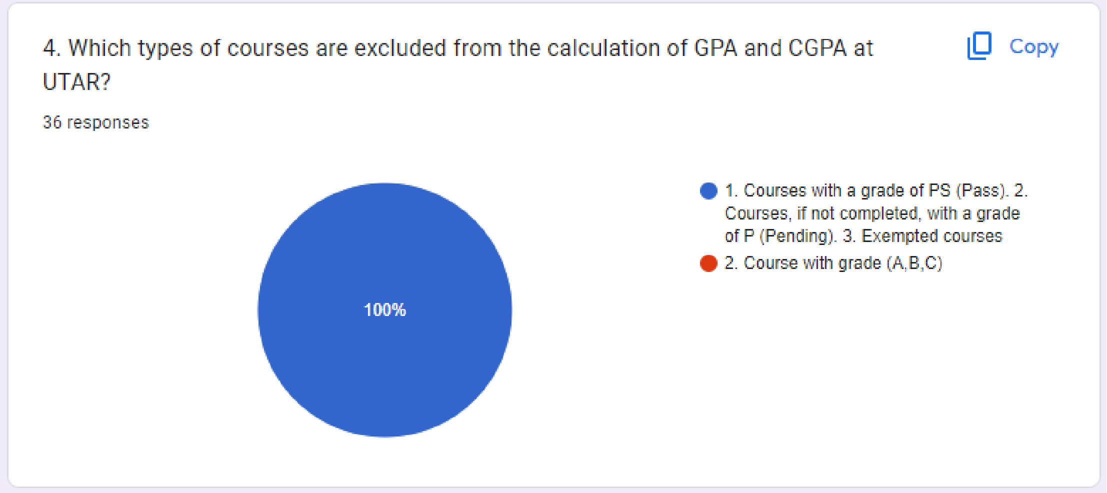
3. Who are the HODs of FICT?

37 responses

Copy
94.6%
● Ts Dr Suthashini a/p Subramaniam, Ts Dr Chang Jing Jing, Dr Zanariah Binti Zainudin, Ts Dr Ooi Boon Yaik
● Ts Dr. Vasaki a/p Ponnusamy, Ts Dr. Ooi Boon Yaik, Ts Dr. Rehan Akbar
● Suthashini

Figure 6.1.2.9 Percentage breakdown of Question 3

Assistant UTARGPT is able to provide correct answers most of the time. (94.6%)



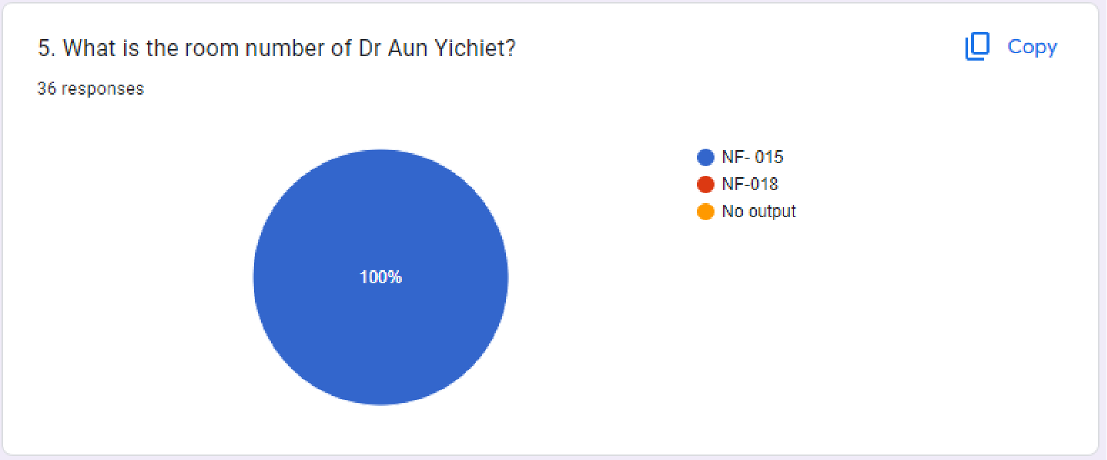
4. Which types of courses are excluded from the calculation of GPA and CGPA at UTAR?

36 responses

Copy
100%
● 1. Courses with a grade of PS (Pass). 2. Courses, if not completed, with a grade of P (Pending). 3. Exempted courses
● 2. Course with grade (A,B,C)

Figure 6.1.2.10 Percentage breakdown of Question 4

Assistant UTARGPT is able to provide correct answers all of the time.



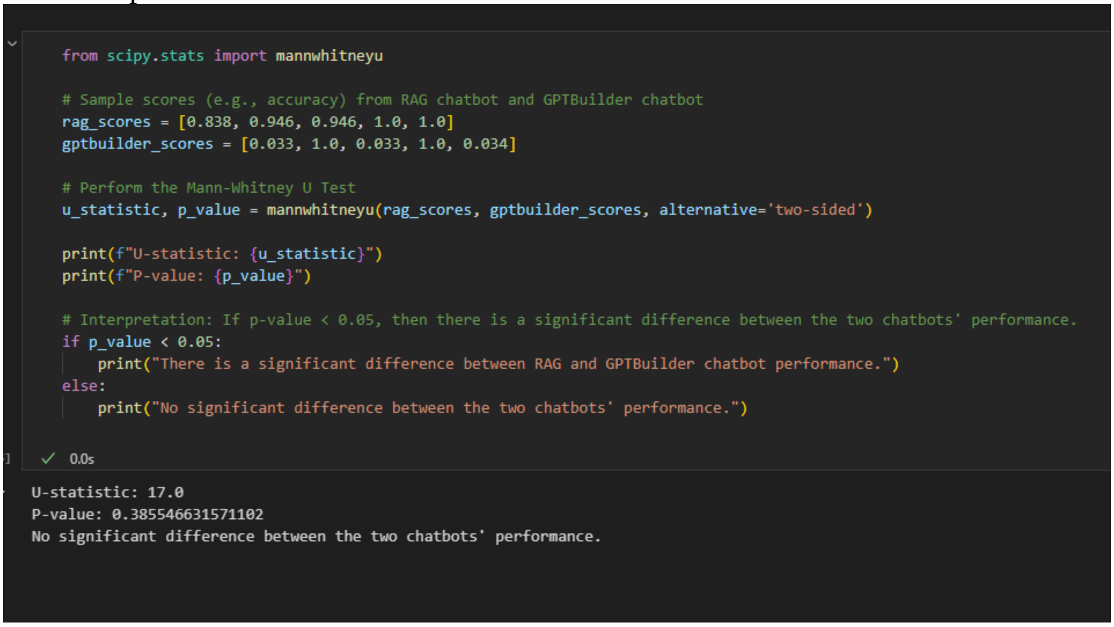
5. What is the room number of Dr Aun Yichiet?

36 responses

Figure 6.1.2.11 Percentage breakdown of Question 5

Assistant UTARGPT is able to provide correct answers all of the time.

Based on the response from the respondents, we can conclude that Assistant UTARGPT is overall provide better and more accurate answer than GPT Builder UTARGPT.



```python
from scipy.stats import mannwhitneyu

# Sample scores (e.g., accuracy) from RAG chatbot and GPTBuilder chatbot
rag_scores = [0.838, 0.946, 0.946, 1.0, 1.0]
gptbuilder_scores = [0.033, 1.0, 0.033, 1.0, 0.034]

# Perform the Mann-Whitney U Test
u_statistic, p_value = mannwhitneyu(rag_scores, gptbuilder_scores, alternative='two-sided')

print(f"U-statistic: {u_statistic}")
print(f"P-value: {p_value}")

# Interpretation: If p-value < 0.05, then there is a significant difference between the two chatbots' performance.
if p_value < 0.05:
    print("There is a significant difference between RAG and GPTBuilder chatbot performance.")
else:
    print("No significant difference between the two chatbots' performance.")
```

```
U-statistic: 17.0
P-value: 0.385546631571102
No significant difference between the two chatbots' performance.
```

Figure 6.1.2.12: Code snippet to compute p-value

Although we can conclude that the Assistant UTARGPT is having better performance than the GPT Builder UTARGPT, but we need to see how well the Assistant UTARGPT is better than the GPT Builder UTARGPT. This can be done by compuiting the p-value to see whether there are having significant difference between them. From the computed p-value, we can see that although Assistant UTARGPT is having better performance, but the different is not significant.

6.2 Project Challenges

In the development of the educational chatbot, there are some challenges faced by me throughout the development stage. First and foremost, extra time and effort are required for me to learn new skills such as web scrapping, building a RAG system, integrating the RAG system with LLM and etc. Since the web scrapping is a technique that exist for quite a long time, I can find a lot of references video to learn on how to perform web scrap easily. Therefore, this is not hard for me to learn it within a short period of time. However, for the RAG system, it is a new technology that it just invented in 2023, therefore it results in fewer references and tutorials that I can refer to on the Internet. This caused me to have a hard time to build a RAG system from scratch. I spent long time in developing a better performance RAG system that can retrieve accurate documents from the vector store.

Besides that, the unclear documentation of the OpenAI about the GPT builder cause unnecessary work has been done. At the early stage of the project, I plan to build the RAG system to integrate with the GPT builder UTARGPT in order to enhance the response from the GPT builder UTARGPT. However, this cannot be done as the GPT builder UTARGPT does not provide API to integrate into other system. Initially, I thought that the GPT builder UTARGPT is able to integrate into other system as the documentation did not clearly state. Therefore, I proceed to use the ChatOpenAI() to instantiate the language model to integrate with the RAG system, but the response is worse than the GPT builder UTARGPT. Therefore, I proceed to find another more powerful chatbot to integrate with the RAG system and compare their performance. However, the Assistant UTARGPT is not an instance of OpenAI, but it is the object of OpenAI, so I can’t directly integrate it with the RAG system, as the invoke agent of the RAG system only accepts LLM instance as its parameter. Therefore, the main purpose of the RAG system besides retrieving the relevant document, it mainly serves to refine the query, to ensure the query from the user is always understandable by the chatbot.

Despite using the same code snippet to generate responses, the behavior of the response generation becomes inconsistent when deployed using the Gradio interface. In particular, the response times noticeably increase, leading to delays that were not present during local testing or in other environments. Additionally, the refined queries that are being generated by the code become inconsistent for no apparent reason, even though the same logic is being applied.

6.3 Objective Evaluation

In this project, it consists of a total of 5 objectives that need to be achieved in our project.

The first objective is to build a customGPT for a context-aware UTAR chatbot. This is our main objective of the project and it is achieved by using OpenAI Assistant and integrating it with the RAG system we built from scratch to outperform than other chatbot. It used the training data web scrapped from the websites to train the chatbot to make it a custom GPT for UTAR students and the nature of OpenAI Assistant to provide context-aware responses to the users.

The second objective is to build a web scrapper tool to collect the data from UTAR websites. To achieve this objective, some commonly used functions has been written in order to facilitate the process of web scrapping and make the future maintenance and upgrading of the chatbot easily. With all these functions written, the developers can perform web scrapping to collect the data for future improvement more easily.

The third objective is to preprocess, normalize and clean the dataset like segmentation merging. To achieve this objective, the same method has been used. Some commonly used functions have been written to clean the unnecessary data to save the tokens and etc. By having these functions, whenever there is new knowledge inserted, the developers can directly clean the data and serve as the training data.

The fourth objective is to implement the Retrieval- Augmented Generation (RAG) for fast information retrieval from large sources. Implementation of RAG can enhance the performance of the chatbot significantly. It improves the performance of the chatbot by retrieving relevant documents and refine the query and pass to the chatbot for response generation. The generated response is better than the previous chatbot without integrated with RAG system.

The fifth objective is to package the chatbot into an API for fast deployment. To achieve this, the chatbot offer an API for seamlessly integration into other system. Unlike GPT Builder UTARGPT, it does not offer the API for integration, therefore it possesses some limitations during the development and deployment phase. With the API function, it added a lot of possibilities for us to enhance the chatbot capabilities and increase user conveniency in accessing the chatbot.

6.4 Concluding Remark

In conclusion, this project has passed two testing which are similarity index computation and A/B testing. Both of these tests has shown that the Assistant UTARGPT is having better performance than GPT Builder UTARGPT. However, the A/B test score has shown that the Assistant UTARGPT does not outperform than the GPT Builder UTARGPT a lot. During the development phase, I faced some challenges which are extra time and effort in learning new knowledge, unclear documentation from OpenAI and response time become inconsistent when deployed in Gradio. With he extra effort being paid to the project, all the objectives of the project has been achieved.

Chapter 7 Conclusion And Recommendation

7.1 Conclusion

In conclusion, Assistant UTARGPT is chosen to centralize the information from the UTAR, allow the student from UTAR to access the information at one site. Before the UTARGPT existed, several problems were faced by the student in accessing the information which are the student don’t know where to get the information and find hard to search the information. The university is lack of a centralized place to store all the information from the university. Since I am a student, I used to get frustrated when searching for relevant information. Therefore, this motivated me to build the UTAR GPT to facilitate the process of retrieving the information as student need a faster and easier way to retrieve the information, the students need a centralized platform to access the information from the school and the student can keep track of the information whether it has been updated by the lecturer. The novel idea of this project is to build a generative chatbot as a centralized platform to answer university-related questions from users. The chatbot is built by using a new tool, OpenAI Assistant and integrated with RAG system, we built from scratch. Web scraping was performed to collect the data from the websites in order to train the chatbot. In order to make sure the chatbot can understand the training data well, data normalization is required to increase the response accuracy of the chatbot. After the preprocessing of data, I started to build the RAG system and integrated it with the OpenAI Assistant UTARGPT. The RAG will refine the query based on the retrieved document and pass to the UTARGPT for response generation. This project emphasizes guiding the new-intake student and the FICT student. The chatbot can be deployed by the staff in UTAR later into their official websites for easy access. All the objectives stated in the project has been achieved.

7.2 Recommendation

Although the developed chatbot is able to deploy and be used now, it still contains some improvements that able to be made. Firstly, the developed chatbot can be integrated by the university staff into their official websites for easy access. With the integration, student can use it any anytime. The current chatbot is only allowed to access via a share link. Next, the developer can add more knowledge to the chatbots as I can only access to limited resources from the university. Therefore, the current project only focus on FICT student and new intake students which I can access the data easily. The staff of the university have the permission to use all these files as the knowledge to make the chatbot more powerful and useable for all the student in the university. Besides that, the Assistant UTARGPT can be launched on a interface that can provide faster response time. The response time of the current chatbot is longer than other chatbot, which decrease its efficiency when generating the response to the users. By using another interface that is having powerful server and backend, we can improve the time taken for generating the response.

REFERENCES

[1] P. Marine, “What are the different types of chatbots?,” *Sinch Engage*, Feb. 07, 2023. https://engage.sinch.com/blog/what-are-the-different-types-of-chatbots/ (accessed Oct. 15, 2023).

[2] A. Kumar, “OpenAI GPT-3 Models List: Explained with Examples,” *Data Analytics*, Jun. 03, 2023. https://vitalflux.com/openai-gpt-3-models-list-examples/ (accessed Oct. 16, 2023).

[3] S. Parmar, “Step-by-Step Guide: How to Integrate ChatGPT into Your Own Application or Website,” *Medium*, Jul. 06, 2023. https://shantun.medium.com/step-by-step-guide-how-to-integrate-chatgpt-into-your-own-application-or-website-2ca908f9daf5 (accessed Mar. 01, 2024).

[4] T. Keary, “AI Hallucination,” *Techopedia*, Jun. 30, 2023. https://www.techopedia.com/definition/ai-hallucination (accessed Feb. 10, 2024).

[5] BoredGeekSociety, “GPT-3 Just Got Better at resolving complex prompts with Davinci-003,” *Medium*, Dec. 20, 2022. https://medium.com/@boredgeeksociety/gpt-3-just-got-better-at-resolving-complex-prompts-with-davinci-003-708393805afe (accessed Dec. 12, 2023).

[6] “How to train a chatbot,” *labelbox-guides.ghost.io*. https://labelbox.com/guides/how-to-train-a-chatbot/ (accessed Jan. 05, 2023).

[7] J. Lau, “How to build your own custom ChatGPT with OpenAI’s GPT builder,” *zapier.com*, Nov. 14, 2023. https://zapier.com/blog/custom-chatgpt/ (accessed Feb. 02, 2024).

[8] R. Reddy, “Chatbots: The Past, Present, And Future,” *BotCore*, Dec. 19, 2018. https://botcore.ai/blog/chatbots-the-past-present-and-future/ (accessed Apr. 12, 2024).

[9] B. Tarnoff, “Weizenbaum’s nightmares: how the inventor of the first chatbot turned against AI,” *The Guardian*, Jul. 25, 2023. Accessed: Apr. 12, 2024. [Online]. Available: https://www.theguardian.com/technology/2023/jul/25/joseph-weizenbaum-inventor-eliza-chatbot-turned-against-artificial-intelligence-ai#:~:text=In%201966%2C%20an%20MIT

[10] D. Alan, W. Barbara Haley, and T. David, “SYSTEMS ANALYSIS & DESIGN An Object-Oriented Approach with UML D E N N I S W I X O M T E G A R D E N,” 2015.

[11] A. Khare, “How to Build a Chatbot Using Retrieval Augmented Generation (RAG),” *Rockset.com*, May 21, 2024. https://rockset.com/blog/how-to-build-a-chatbot-using-retrieval-augmented-generation-rag/ (accessed Sep. 11, 2024).

[12] B. Huang, “Chatbot with LLM and RAG in Action - Bill Huang - Medium,” *Medium*, Feb. 28, 2024. https://medium.com/@yingbiao/chatbot-with-llm-and-rag-in-action-575382df4323 (accessed Sep. 11, 2024).

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4T1	**Study week no.: 2**
Student Name & ID:Loh Jun Xiang 2002834	
Supervisor: Dr Aun Yichiet	
Project Title: UTARGPT	

1. WORK DONE

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

- View and refresh memories on what I did on FYP1.

2. WORK TO BE DONE

- Figure out the tool to enhance the chatbot
- Enhance the GPT Builder UTARGPT.

3. PROBLEMS ENCOUNTERED

- No problem

4. SELF EVALUATION OF THE PROGRESS

- Self-assigned tasks are completed within the expected timeframe.

Supervisor's signature

Student's signature

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4T1	**Study week no.: 5**
Student Name & ID:Loh Jun Xiang 2002834	
Supervisor: Dr Aun Yichiet	
Project Title: UTARGPT	

1. WORK DONE

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

- Enhance GPT Builder

2. WORK TO BE DONE

- Build a chatbot using OpenAI Assistant
- Build RAG System
- Perform data processing

3. PROBLEMS ENCOUNTERED

- Lack of knowledge about building the RAG system

4. SELF EVALUATION OF THE PROGRESS

- Self-assigned tasks are completed within the expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4T1	**Study week no.: 8**
Student Name & ID:Loh Jun Xiang 2002834	
Supervisor: Dr Aun Yichiet	
Project Title: UTARGPT	

1. WORK DONE

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

- Done building the RAG system

2. WORK TO BE DONE

- Build OpenAI Assistant UTARGPT
- Perform testing on both UTARGPT
-

3. PROBLEMS ENCOUNTERED

- Lacking knowledge of OpenAI Assistant UTARGPT

4. SELF EVALUATION OF THE PROGRESS

- Self-assigned tasks are completed within the expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4T1	**Study week no.: 10**
Student Name & ID:Loh Jun Xiang 2002834	
Supervisor: Dr Aun Yichiet	
Project Title: UTARGPT	

1. WORK DONE

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

- Done chatbot testing
- Done building Assistant chatbot

2. WORK TO BE DONE

- Report

3. PROBLEMS ENCOUNTERED

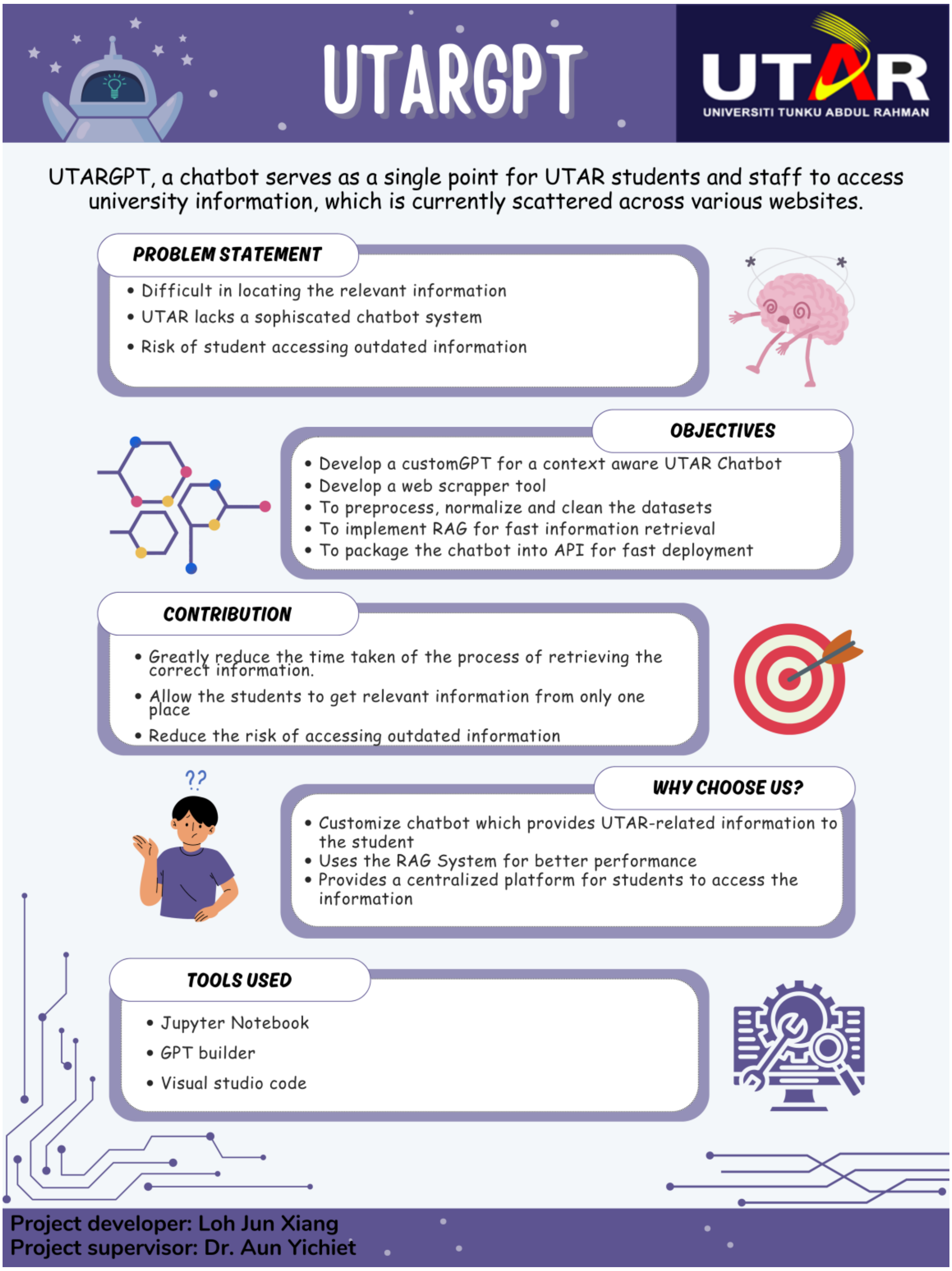
–

4. SELF EVALUATION OF THE PROGRESS

- Self-assigned tasks are completed within the expected timeframe.

Supervisor's signature　　　　　　　　Student's signature

POSTER



UTARGPT

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

UTARGPT, a chatbot serves as a single point for UTAR students and staff to access university information, which is currently scattered across various websites.

PROBLEM STATEMENT

- Difficult in locating the relevant information
- UTAR lacks a sophiscated chatbot system
- Risk of student accessing outdated information

OBJECTIVES

- Develop a customGPT for a context aware UTAR Chatbot
- Develop a web scrapper tool
- To preprocess, normalize and clean the datasets
- To implement RAG for fast information retrieval
- To package the chatbot into API for fast deployment

CONTRIBUTION

- Greatly reduce the time taken of the process of retrieving the correct information.
- Allow the students to get relevant information from only one place
- Reduce the risk of accessing outdated information

WHY CHOOSE US?

- Customize chatbot which provides UTAR-related information to the student
- Uses the RAG System for better performance
- Provides a centralized platform for students to access the information

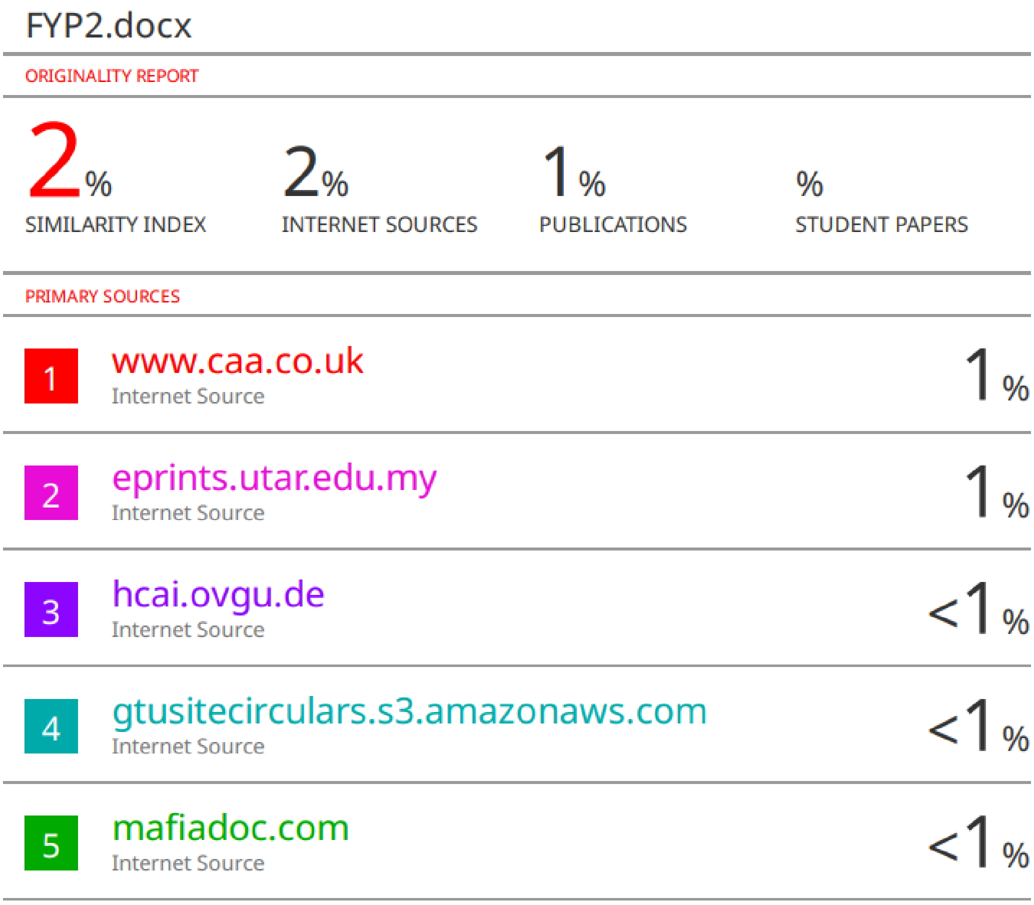
TOOLS USED

- Jupyter Notebook
- GPT builder
- Visual studio code

Project developer: Loh Jun Xiang
Project supervisor: Dr. Aun Yichiet

APPENDIX

PLAGIARISM CHECK RESULT



FYP2.docx

ORIGINALITY REPORT

2%	2%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.caa.co.uk Internet Source	1%
2	eprints.utar.edu.my Internet Source	1%
3	hcai.ovgu.de Internet Source	<1%
4	gtusitecirculars.s3.amazonaws.com Internet Source	<1%
5	mafiadoc.com Internet Source	<1%

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:	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Loh Jun Xiang
ID Number(s)	2002834
Programme / Course	Bachelor of Computer Science
Title of Final Year Project	UTARGPT

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: __2__ **% Similarity by source** Internet Sources: 2 % Publications: 1 % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: 0	

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

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

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: __11/9/2024_________________________	Date: _____11/9/2024_____________________

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	2002834
Student Name	Loh Jun Xiang
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, Loh Jun Xiang, have checked and confirmed all the items listed in the table are included in my report.

APPENDIX

_________ _______________

(Signature of Student)

Date: 13/9/2024