MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL

By

CHUA FENG ZHAO

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)

JUNE 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Mobile Tour Guide Application for Kampar Tourism Using an Interactive Chatbot with Generative Pre-trained Language Model

Academic Session: June 2024

I CHUA FENG ZHAO

(CAPITAL LETTER)

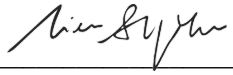
declare that I allow this Final Year Project Report to be kept in

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:

No 38, Leboh Bangau, Taman Bekerley,

41150, Klang, Selangor

Prof. Ts. Liew Soung Yue
Supervisor's name

Date: 8/9/2024

Date: 12/9/2024

FACULTY/INSTITUTE* OF INFORMATION & COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 8/9/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ***Chua Feng Zhao*** (ID No: ***22ACB00604***) has completed this final year project/ dissertation/ thesis* entitled "*Mobile Tour Guide Application for Kampar Tourism Using an Interactive Chatbot with Generative Pre-trained Language Model*" under the supervision of Prof. Ts. Liew Soung Yue (Supervisor) from the Department of _____, Faculty/Institute* of Information & Communication Technology , and _____ (Co-Supervisor)* from the Department of _____, Faculty/Institute* of _____.

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

Yours truly,

Chua Feng Zhao

DECLARATION OF ORIGINALITY

I declare that this report entitled "**MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL**" 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 : Chua Feng Zhao

Date : 8/9/2024

ACKNOWLEDGEMENTS

I would like to express my biggest thanks to my supervisor, Professor Liew Soung Yue, for giving me this opportunity to work on this project, for being supportive of this project and for helping the development of the project by providing suggestions and resources when I needed them.

ABSTRACT

Tourism in small, remote areas is way more uncommon than in large cities due to their hard-to-access information, making planning for a trip difficult and time consuming as tourists will have to scour the entire Internet to find places and attractions which they may or may not prefer as second-hand opinion can also be difficult to come by due to the area's remote nature. Tour guides are trained to resolve these types of issues but can also introduce new problems for the tourists. This project aims to develop a chatbot for a mobile application which allows users to immediately request an area or attraction in one of these small, remote areas that they would like to go to vacation for, and immediately obtain an answer through its 'command and answer' model. Previous work has been done on tourism chatbots, however these chatbots are usually not language adaptive due to the primitive language model they used. Not only that, these chatbots, when providing suggestions, usually miss out other factors such as user's preferences. The chatbot this project will be developing will consider user's request alongside user's preferences, for example, whether they prefer attractions that prefer being physically active or just sightseeing. In addition, the chatbot is also made to adapt to additional information the user could provide, such as the time and weather in Kampar and the user's current coordinate. With the advent and popularity of ChatGPT, this advanced language model will be used in order for it to grab context from user's messages to accurately determine user's request with little misinterpretation, and to grab context from user reviews for tourism areas to evaluate how well should it be suggested. Web scraping is performed to feed much more specific Kampar tourism knowledge into a language model's database which will allow it to provide more comprehensive tourism answers. The chatbot also aims to act like a tour guide and provide information regarding an attraction or site whenever users request for it. The chatbot also provides extra functions which can improve the user's traveling experience such as providing directions, updating user's traveling preferences in real-time and providing user's the ability to make reviews of the places in Kampar.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
RAG	Retrieval-Augmented Generation
LLM	Large Language Model

Chapter 1

Introduction

According to the Cambridge dictionary, the definition of being a tourist is to be “someone who visits a place for pleasure and interest, usually while on holiday” [1]. Almost everyone travels at least occasionally. Malaysia is no stranger to the art of tourism, in fact, according to data collected back in 2019, tourism is one of the largest growing industries in Malaysia as it contributes roughly 6.7% of Malaysia’s gross domestic product (GDP) [2]. Malaysia’s tourism sector thrives due to many factors and one of the major ones is the diverse and exotic cultures presented in Malaysia. With many different cultures, each with their own history, beliefs, artistic expression, food, etcetera, Malaysia’s wide cultural variety gives it an edge in the tourism sector and with the major lockdown period that occurred back in 2020 starting to subside, the growth of Malaysia’s tourism sector doesn’t seem it will stop soon.

1.1 Problem Statement and Motivation

However, while the major and important cities, such as KL, the heart of Malaysia, are well documented in various resources, such as the Internet, which makes it easy for potential tourists to look up recommended places and attractions that suits their wants and plan their trip in advance or to look places up on the spot during their vacation, some remote areas of Malaysia are not so well documented, making information relating to these areas both harder to access and spread very thin all across multiple sources. One of these places is the quaint town of Kampar, located in Perak in Malaysia. The hard-to-access information of certain smaller areas like this can lead to some tourists not wanting to explore the area as they do not want to explore a place they are not familiar with without a guide or recommendation nor do they want to waste time combing through website after website to search for that information. This can cause tourists to miss out on great experiences, as usually it is through these smaller towns where great experiences of Malaysia’s culture can be had, not to mention the niche attractions that can only be had in these small towns. For the tourists who want to explore these smaller towns even without extensive information on them, the hard-to-access information can sour their experiences as not only tourists needs to waste time finding information they might not be able to find, but they can also often get lost or participate in events or attraction that they will not

know they do not find interesting until it is too late. Overall, this can negatively affect Malaysia's tourism industry.

One way to combat this problem is to have a tour guide, who will have extensive and up to date knowledge on the tourism area, not to mention it is easy to get recommendations that cater towards the tourists' wants and preferences as tourists can just ask them and not have to scour the aera or the Internet to do so. However, this presents its own set of problems. For one, hiring a tour guide costs money, which some tourists cannot afford to spend more in addition to their vacation. Not only that, but tour guides also usually have a set schedule. This limits the freedom of tourists who may want to explore the area at their own pace. Finally, tour guides also just add another person to the traveling group, which may not be ideal for some tourists who just want to experience the vacation by themselves, their family or group of friends without any outside parties.

Technology has been so integrated into our daily lives that it is impossible to think of a future where we could live without it. Tourism has also utilized technology to ease the process of planning and travelling for tourists. For example, the Internet is now used as an information repository for tourist attractions. Not only that, but tourists can also use the Internet to help book into hotels they will stay at during their trip. A chatbot is a technology that uses a Language Model to simulate a conversation between a system and a user. A chatbot can act like a tour guide but without the hassle of introducing factors such as cost or a restrictive schedule. An example chatbot that is extremely popular at the moment is ChatGPT, where it uses a Language Based Model to communicate with users and understand them. A chatbot using a powerful language model can even provide detailed explanation of areas they are in that suits the user's level of understanding. This could be the next step the tourism sector, especially for these less information areas need to improve.

According to a whiteboard written by WBB [3], users will typically abandon processes that requires them to navigate multiple layers of information, such as searching for recommendations or getting great vacation deals from a separate application that they need to download separately. Thus, travel apps can help users save time on searching for recommendations for their vacation as they work by a 'command and answer' model. Instead of having to search up for recommendations, tourists can just request for recommendations and have a system present them to the user, skipping the time needed to search.

Not only that, chatbots can be incorporated with algorithms or machine learning in order for them to cater to user's preferences, which can further enhance the user's travelling experience.

In this project, a mobile tour guide application for Kampar tourism using an interactive chatbot will be developed. With the recent development of the powerful AI language model of ChatGPT, the technology allows developers to exploit the powerful model behind it to develop very useful chatbot for applications in different areas, such as tourism, which will be the focus of the application.

1.2 Objectives

The main objective of this project is to develop a recommendation chatbot for the purposes of tourism in small towns with time consuming and hard to access information. In this case, this project will focus on Kampar in Malaysia. The chatbot will be implemented into a Kampar tourism mobile application. This is done in order to solve the problem of tourists not wanting to explore these small towns due to the hard-to-access information, which makes it hard for them to plan for a trip or even find sites or locations they want to visit to on-the-spot, not to mention the likelihood of encountering an event that does not appeal to them until they have already signed up for it.

This chatbot aims to allow tourists to request for suggestions of where to go in Kampar for various purposes, such as searching for lodging, a place to eat, or a leisure attraction and immediately obtain a or a few suggestions that tourists can select from immediately, instead of having to search for recommendation themselves and wasting their time. Using the 'command and answer' model [3], tourist can efficiently and seamlessly obtain recommendations that is catered towards their wants. Furthermore, the chatbot also aims to, just like a tour guide, be able to provide additional information on places, foods or cultures whenever users want quickly without needing to switch to an Internet browser. The chatbot also aims to provide suggestions even when users have little information provided on where they want to go. This makes the chatbot cater to the tourist who take the journey as it comes or tourists who have no idea where they want to go.

The scope of the project is wide to achieve this objective. Each of the project scopes also addresses issues and limitations observed from previous works and provides its own innovation and contributions.

1.3 Project Scope and Direction

This project will focus on creating a chatbot that utilizes an advanced large language model. In this project, GPT-4o mini will be used as it is currently popular and relatively simple to obtain and use. The aim is to build a chatbot that can converse with users with little misinterpretation of their requests and also to provide a more human-like chatbot experiences compared to what is already experimented with.

Furthermore, the project aims to conduct data collection in order to obtain existing tourist data to as the chatbot's knowledge base. Web scraping, which is the process of extracting information from websites, will be conducted for this project in order for data from existing tourism website, for example TripAdvisor, which is a popular tourism website. Data processing will also be conducted to process and clean the data once collected.

In addition, the project aims to retrieve user preferences through the application in order to further understand where users want to go and to provide a much more suitable and personalized response for the user based on their preferences. These user preferences can also help for users who do not know where to go as they can just ask a simple question on where they should go and the chatbot can take in their preferences as a guideline on what the chatbot should recommend to the user.

Moreover, the project aims to allow users to improve the output of the chatbot by allowing users to choose to send certain additional information, such as weather, time and current location, which can further improve the decision-making process of the chatbot and provide better suggestions.

The project also aims to add additional features to improve user's traveling experience, such as updating user's preferences on the spot, providing directions to places and ability to send reviews.

1.4 Contributions

The contributions that this project brings to tourism chatbot is the use of a large language model to develop the chatbot. Doing this, the chatbot is able to better understand user's request and able to pick up on emotional context from existing user reviews which can help in the decision-making process of the chatbot. It also allows for more comprehensive answers that provides more context and information for users.

Furthermore, this project also brings in a more in-depth user profiling of their traveling preferences, which adds more dimensions to the personalization that the chatbot is able to make for its user's requests. In addition, the chatbot is also able to provide information for users who provide little to no information as well.

In addition, the project also incorporates extra information to improve the suggestions of the chatbot and users can allow the chatbot to update their travelling preferences in real-time so that the chatbot will always be up to date on the user's current tastes, even if they change.

1.5 Report Organization

In Chapter 2 of the report, literatures on previously worked on tourism chatbots will be reviewed in order to determine what has already been done in the realm of tourism chatbots and what can be done to improve upon them. In Chapter 3 of the report will go over the methodology of the system, including describing how the chatbot understands user's request and make recommendations, the architecture of the system and the use cases of the mobile application. In Chapter 4, the design of the system, ranging from the web scraping flow, the design used to process the data collected, and the standard flow of using the mobile application of the project proposed will be described. In Chapter 5, The process of developing and implementing the system is listed. Chapter 6 will show the various test conducted to evaluate the effectiveness of the system alongside evaluating the system in achieving the objectives of the project. Finally, Chapter 7 will conclude the report.

Chapter 2
Literature Review

2.1 The Development of AI Chatbot for Kampar Tourism Mobile Application

2.1.1 Review of Proposed Method

This chatbot, that was developed by Ngui Miew Yiong [4] aims to create a chatbot specifically made for small towns with difficult to access information, such as Kampar in Perak Malaysia.

The chatbot for this proposed method uses the IBM cloud services to create the chatbot. The IBM cloud service contains a service called the Watson Assistant, which is a pretrained AI chatbot cloud service. In Watson Assistant, developers can code in intents where an intent is a set of contexts, user queries and responses to those queries. Using this, developers can allow users to interact with the system using a language-based model, converting user's language-based queries into a language-based response based on the intents developers code in.

The chatbot allows users to interact with the chatbot, either through text or voice. Through the intents that are incorporated into the chatbot, the chatbot can determine whether the user is just interacting with the chatbot normally as a chatting partner. Figure 2.1.1 shows the chatbot responding to non-tourism related queries. When the chatbot detects certain keywords that relate to tourism and users asking for recommendations, such as 'direction' and 'café', the chatbot will access the database to grab the required information, in this case, directions to a café, and present it to the user. Users can then give the area a star rating from 1 to 5 stars and can even request more similar places to the one suggested if the one suggested isn't to the user's liking. Users can also have the chatbot route them to their destination. Figure 2.1.1 shows the recommendation screen of the chatbot.

Figure 2.1.1: Proposed Chatbot Responding to Non-tourism Text [4]

Figure 2.1.2: Recommendation Module of Chatbot [4]

The chatbot also contains an admin page where admins can view user's favorite places that the chatbot suggested, the user's favorite tags and can visualize the data through bar charts.

2.1.2 Strengths of Proposed Methods

The chatbot is able to respond to queries or text not related to tourism with language that resembles more human than AI or machine. This can enhance the conversational experience for the user as it enhances the illusion that the user is talking to someone rather than a robot or a machine. It gives the users the feeling of talking to a tour guide.

The chatbot is also able to provide analytical information about users. This information can be used for both creating personalized marketing for certain users from the third parties who can access this information, which can further grow the tourism industry but also as a way to obtain feedback from the application and improve the system should the user's experience does not meet expectations. For example, admins can find out whether a user that searches 'café' often gets recommended more restaurants instead and can readjust the application accordingly.

2.1.3 Critique of Proposed Method

The recommending module of the chatbot is very primitive as it only context the chatbot tries to obtain is the type of places the user wants to go to. For example, a café or a restaurant. It does not consider other external and internal factors, such as the user's budget, the user's current location, the weather, the time and day, the user's previous activity and etcetera. The chatbot only recommends a random location of that type upon request. This makes it so the chatbot does not personalize its suggestion according to the user and some users will not be inclined to use the application as its suggestion may not align with the user's interests.

2.2 The Application of AGNES Algorithm to Optimize Knowledge Base for Tourism Chatbot

2.2.1 Review of Proposed Method

Sometimes, tourists are only on vacation for a short amount of time. In this scenario, it is imperative that they do not waste time trying to figure out what attractions, events or location of interests they should visit, since they only have so less time to do so. This article written by A. V. D. Sano., et al [5] aims to implement the AGNES (Agglomerative Nesting) algorithm in order for the tourism chatbot to determine which group of tourist attractions can visit optimally under a short time frame.

According to the article, the AGNES algorithm is a hierarchical clustering algorithm. In which when ran, will create a dendogram. A dendogram is a tree-like diagram which contains data points or clusters in a hierarchical structure. It is a visual tool which helps us view how similar data points or objects group together based on their similarity or dissimilarity. A dendrogram is read from top to bottom, in which the clusters formed can be seen, allowing the data points or clusters that are most similar to each other to be viewed. The lower the dendrogram goes, the clusters get smaller but also grow more abundant.

The methodology of this article is to implement this algorithm into a chatbot, so that it is able to determine the best set of attractions tourists can visit in their limited time. To do so, the article first describes how data is collected. First, 39 popular sites all over Malang City, Malang Regency and Batu City in Indonesia are selected. Then, their geolocation, which is based on the location's latitude and longitude, is found, and recorded. Finally, the distance between every two locations is determined using a latitude longitude converter software. Table 2.1.1 shows how this data is stored.

Table 2.1.1: Table Showing Data Storage Method for Tourism Sites and their Distances [5]

No	Geolocation		Tourism Sites						
	Latitude	*Longitude*		1	2	3	4	5	6
1	-7,878576	112,520494	Museum Angkut	0	2,4941	0,6757	1,4796	1,398	4,9185
2	-7,896281	112,53442	Batu Night Spectaculer (BNS)	2,4941	0	1,8211	1,0422	1,0963	2,4324
3	-7,883645	112,523885	Jatim Park I	0,6757	1,8211	0	0,8285	0,725	4,2497
4	-7,887984	112,530006	Jatim Park II: Museum Satwa dan Batu Secret Zoo	1,4796	1,0422	0,8285	0	0,2135	3,4438
5	-7,8886	112,528169	Eco Green Park	1,398	1,0963	0,725	0,2135	0	3,5251
6	-7,911904	112,549898	Predator Fun Park	4,9185	2,4324	4,2497	3,4438	3,5251	0

Once the data is collected and stored, the data is mined using the AGNES algorithm. The following two parameters are used. The first criterion is the linkage criterion, where an

average-link is used. The second criterion is the measurement type, where the Euclidean distance equations are used.

The chatbot uses Google's DialogFlow framework as its knowledge base. DialogFlow is a chatbot framework that allows developers to create a conversation flow through inputting intents. Which are a set of contexts, user queries, events, actions, parameters and responses. Through these intents, the chatbot will be able to determine the intent and contexts of user's messages and queries, by cross-referencing them with the intents that has been programmed into the chatbot framework, allowing the chatbot to answer user's questions appropriately. Figure 2.1.1 shows the console of DialogFlow and how intents are programmed into the framework.

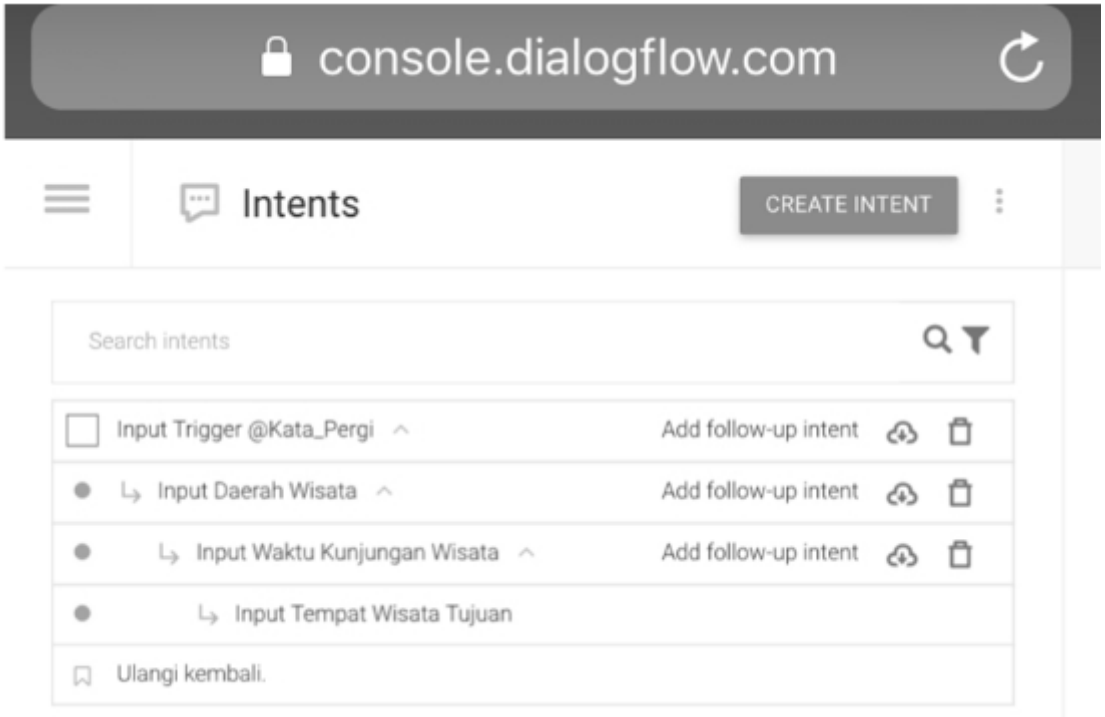


Figure 2.2.1: Console of DialogFlow [5]

Figure 2.1.2 shows the results of the chatbot using the AGNES algorithm. The 39 sites are clustered into clusters, with each cluster being on different levels. With the higher levels meaning that they are optimized for lower day trips as for every node with two levels of depth from the bottom can be grouped as one-day tours, fourth depth as two-day tours, sixth depth as three-day tours, and so on. Figure 2.1.3 shows the chatbot responding to the user appropriately. The user requests for sites that can be optimally visited in a one-day tour and the chatbot returns one of few sites that existed in the two depth clusters.

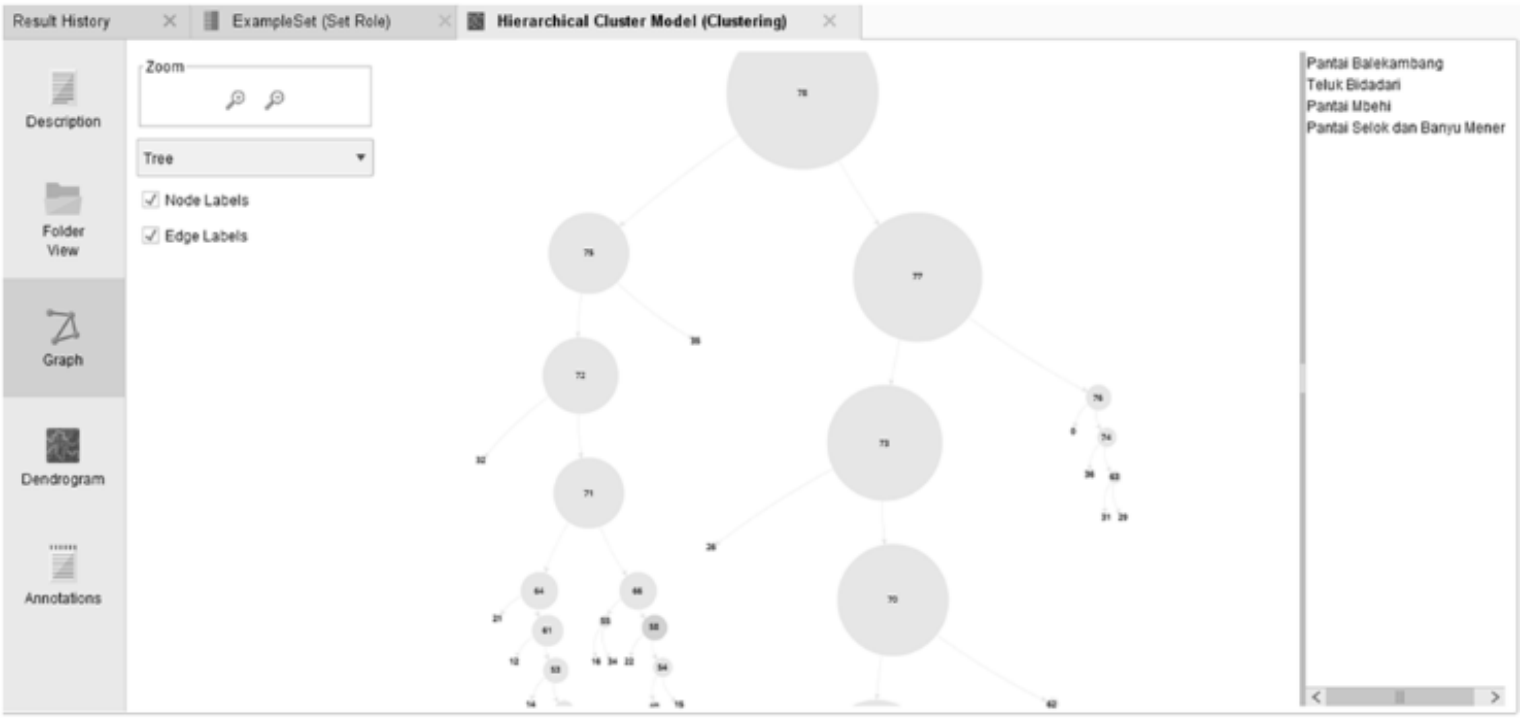


Figure 2.2.2: Clusters from the AGNES algorithm [5]

Figure 2.2.3: Results of Chatbot [5]

2.2.2 Strengths of Proposed Method

One of the strengths of this proposed method is that it keeps in mind the distances between every location when clustering the sites through the AGNES algorithm, it keeps in mind the distances between every location when making its decision. This makes it so that the tourists do not have to waste too much of their very limited time from traveling from one site to another, which makes the flow of visitations better. Another strength of this method is that it provides a more concrete answers compared to a search engine. While a search engine can provide information such as the time frame that will be used for certain events, it does not provide answers that fits the user's time constraints.

2.2.3 Critique of Proposed Method

Firstly, it doesn't take into account of other factors that might hinder a tourist's trip to these areas, such as weather forecasts. An outdoor event will not be suitable if the weather forecasts rain for the day.

Not only that, the chatbot is not language-adaptive due to the framework used to build the chatbot in. DialogFlow is not a sophisticated language model as it can only generate responses based on what is given to it as intents. This means the chatbot will sometimes not pick up on the user's request as their way of phrasing their wants does not match the chatbot intents. For example, the chatbot may not understand that 'grabbing a bite' likely means the user wants to go to a restaurant as that is not coded into its intent.

Furthermore, the chatbot does not take in user's location into account, which can very well mean it is not optimized for the trip the chatbot provides. For example, certain sites or attractions may be more suitable to users who are driving and living in a hotel that is closer to them rather than sites that are further away.

Finally, the sites selected are seemingly random based on the clusters that are created from the AGNES algorithm and do not cater to tourist's interest. For example, a tourist may prefer outdoor events but the chatbot may suggest a museum instead.

2.3 A Context-Aware Chatbot for Tourist Destinations

2.3.1 Review of Proposed Method

This article written by F. Clarizia, et al. [6], aims to implement the context dimension tree to represent user context as they chat with the chatbot to provide suggestions on where they should go. They also aim to create a chatbot that can provide a compelling tour guide-like experience by having the chatbot provide story-like responses to historical queries. According to the article, context is described as the combination of places, information or identity of people, their activities, information on their devices and environmental factors such as weather conditions and time parameters such as day of the week.

According to the article, context is important for user suggestions because this information can be used to detect and react to a user situation. This allows systems that utilizes context based learning to adapt its presented content to better tailor its data to suit the users preferences and needs, increase the accuracy of information retrieval, discover services, make user interaction implicit and create intelligent environments. For example, a system with user context information can provide content based on different user interests and profiles. Furthermore, it can detect user environment to derive information and provide appropriate information based on that location.

According to the article, context can be represented through a Context Dimension Tree, which is what the proposed system of this article uses to track user context through their text. The proposed system also uses natural language processing (NLP) to have machines understand and relate to human beings. It processes sentences not by single word, but entire sentences. By doing this, the deeper meaning of the words said to the chatbot can be extracted.

Figure 2.3.1 shows the proposed architecture for the proposed system. It consists of the back-office or the back end of the system, which contains the knowledge base which is where all the data regarding user data, context information and tourism data that the chatbot will retrieve from and store to as it operates, and the front-end which contains the visible working components of the chatbot. There are two important components of the chatbot in the front-end side of the system.

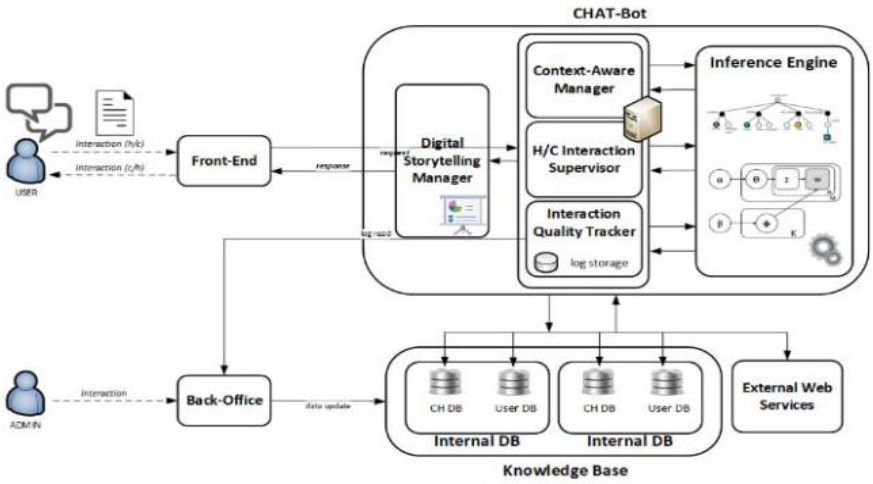


Figure 2.3.1: Proposed System Architecture [6]

The first part of the front-end side of the proposed system is the digital storytelling manager. This module controls the guidance of users in order to personalize the journey of each user in a creative and unique way. Every user will have a different experience and will ask different questions about an attraction or an exhibit they are in. Just like a tour guide, the chatbot will provide answers tailored to the user using the digital storytelling manager, so every user's experience with the exhibit will be uniquely theirs. The digital storytelling manager is made so it can adapt to a lot of situations in order to meet its functioning criteria.

The second part of the front-end side of the proposed system is the context-aware manager. As described earlier, the chatbot uses a context dimension tree to determine the possible contexts from user conversation, allowing extraction of personalized information and use them to provide custom-tailored data from private or local sources. These can be used to maintain user profiles, or review data of the location they are in. The Latent Dirichlet Allocation model is used as a base for the context dimension tree. The Latent Dirichlet Allocation is a model commonly used for topic modeling. It does this by mapping a probability distribution on how likely a set of words are related to each other. In this proposed system's case, it wants to explain the correlation between keywords and topics. The chatbot can then use this information to determine the context of a sentence. For example, the chatbot can determine the probability that 'restaurant' and 'directions' are correlated in the sentence, and from there, it can extrapolate that the user would like to do something relating to directions to a restaurant. Sentences users typed are stored in smaller sentences and stored in a Mixed Graph of Terms (mGT) as clusters. Bayesian filters for keywords aid in determining the probability of the correlation of keywords to each other in a sentence by determining whether term belongs to a concept node.

Figure 2.3.2 shows the proposed system in use. The chatbot is able to determine user context by reading where they are geographically, accurately determine where the user is, and provide them a suitable hotel based on that location. It is also able to determine, from user's text, that they want to purchase a ticket and provide the correct directions for users to do so. It also provides more information regarding the attraction when prompted. When exiting the system, the system will grab user feedback in order to be used to further improve the system in the future.

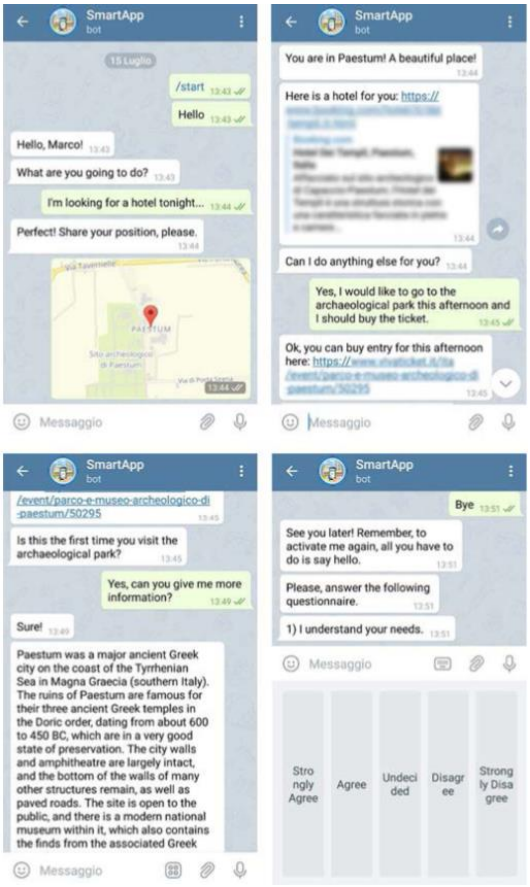


Figure 2.3.2: Proposed System Chatbot [6]

Feedback was taken from the users based on 5 criteria: recommendation, conversation, presentation, usability and future developments. Overall, user reviews were positive, which shows that the system is able to accurately cater to multiple different needs from various types of users which can be seen in the confusion matrix shown in Table 2.3.1, which shows a high accuracy along with a high precision and recall when it comes to the chatbot's ability to determine user's context. However, the aspect of the system that received the least amount of positive feedback is on the chatbot's ability to converse.

Table 2.3.1: Confusion Matrix of Proposed Chatbot [6]

	Service	Indoor	Outdoor	Tourist	Citizen	Holiday	Freetime	Morning	Afternoon	Night
Service	9905	205	176	186	163	181	190	170	182	183
Indoor	57	4141	56	45	38	36	75	74	58	50
Outdoor	48	80	6362	71	77	56	54	61	57	90
Tourist	57	54	67	4113	78	46	65	43	56	73
Citizen	73	58	57	67	6411	74	73	63	68	53
Holiday	204	167	192	191	157	7633	154	164	158	184
Freetime	211	187	222	224	189	232	9621	236	193	206
Morning	121	136	130	138	101	116	98	5881	111	103
Afternoon	46	41	67	76	66	80	47	76	4035	83
Night	99	74	62	101	86	100	56	63	93	6223
Overall Accuracy : 87,13%										

2.3.2 Strengths of Proposed Method

One of the strengths of this chatbot is that it is able to interface with private services, such as help send links to book for hotels or certain events. This reduces the hassle from tourists as they can just refer to the chatbot to book at the hotel or events the chatbot already recommended so that tourist don't need to search the Internet separately for them to book. This saves users some time.

Moreover, the chatbot is able to fully adapt to user's needs. When users request for a hotel booking, the chatbot is able to recognize that the user's needs a hotel booked near them and requests for geographical location before providing a suitable hotel for them to book at. It is then able to adapt to the user's change of topic on the fly as well.

Another strength of this chatbot is that it follows up on its answers. For example, the chatbot was able to inquire whether the user has been to a certain area or place before after suggesting it to the user. This is advantageous because it allows the chatbot to immediately glean more information from users, improving user's data and allowing the chatbot to obtain more context information in order to provide a more customized user experience.

2.3.3 Critique of Proposed Method

One of the flaws of this system is that it doesn't automatically record the location from the device, instead, requests users to share their location, which can be a hassle for users. This is especially true if users are currently outside, unsure of where to go or what direction to head towards. In that scenario, users will expect an immediate response in order to get a move on to their next destination. This adds an extra step into the process which wastes tourist's time.

Furthermore, it is already established by the article that one of the system's least positive aspect is its ability to converse. This could be due to the primitive language model used due to its lack of dialect and grammar training. This can lead to the chatbot misinterpreting user's request, grabbing the wrong context from their text and providing the wrong solution to the user's problem.

Finally, the chatbot only provides information based on what the user already wants and doesn't provide a lot of information if the user is unsure of what they want. For instance, when the user wanted to look for a hotel, the chatbot only suggested one random hotel from its knowledge base. Some tourists just do not actually know what they want during their vacation as they plan for their vacation, or sometimes tourists are forced to change their plans at the last minute and have no idea where they want to go from there. The lack of information given when the user doesn't provide what they want could pose a problem as users may not like the suggestion given. A chatbot that can try to suggest multiple places when users do not know where to go can improve user's experience as users can maximize their vacation time having fun instead of spending time figuring out where they should go.

2.4 Chatbot-based Tourist Recommendations Using Model-Based Reasoning

2.4.1 Review of Proposed Method

While a chatbot can help users make recommendations for their tourist needs, a chatbot might struggle to help make the recommendations for the tourist if the tourist cannot restrict the number of recommendations that would be provided from a prompt. For example, a user that just asks that they want to go eat with no further context will generate a lot of suggestions, which does not help users to find the best option to suit their needs. Likewise, a user that asks for too many restrictions may not get a match at all, which in the end will result with no matches, which also doesn't help users at all. This article, written by I. Nica., O. A. Tazl, and F. Wotawa. [7] aims to use model-based reasoning to address both issues in a tourism chatbot.

Figure 2.4.1 shows a mock-up of the proposed system, where the tourism chatbot will respond to users' tourism queries and help recommend places for users, at the same time, adapt when there are too many suggestions due to lack of restrictions or no suggestions can be made due to too many restrictions.

```
Bot: Hello John, how can I help you today?
John: I would like to plan a trip.
Bot: Great! Are you interested in a specific region
       or would you like to plan the trip by activities?
John: I am interested in a specific region.
Bot: Please click on map to pick the point of interest
       or type in the place that you are interested in.
 (John clicks a point on the map near the city Berlin)
Bot: OK, I will search for relevant sights near Berlin.
(Shows a list of relevant sights in Berlin.)
John: Show me the Brandenburger Tor.
(The bot shows several information about the Brandenburger Tor.)
John: Looks nice! Find a low-priced hotel near this sight.
Bot: I am sorry, but there are no hotels of this category near
     Brandenburger Tor. Which requirement is not that important
     to you? Price or distance?
John: Distance.
Bot: OK! There are 9 hotels available for you. You should
     incorporate another filter. How about parking? Will
     you have a car?
John: No, I do not need a parking facility.
Bot: OK! Now there are 4 hotels left. Here is the list: Hotel 7,
     Hotel 10, Hotel 15, Hotel 18.
John: Cool! Is Hotel 15 available from 19.08.2018 to 25.08.2018
      for two people?
Bot: Yes! Should I book it?
John: Yes.
Bot: Hotel booked!
```

Figure 2.4.1: Mock-up of proposed system [7]

In order to achieve this result, an item set must be created for a relevant tourist item, for example, a hotel. Table 2.4.1 shows hotels labeled through their distance to other sites or attractions, the hotel category, whether parking space is given and the price range of the hotel.

For this proposed system, whenever users make a request, in case, a request to find or book a hotel, the chatbot will first detect if the users have any of these restrictions. The system will then try to limit and suggest hotels that fit the user's criteria. For example, a user requesting a 3-star hotel with parking space. The chatbot will recommend Hotel 3, Hotel 4, Hotel 9, Hotel 16 and Hotel 19 to the user. While this provides a suitable list of items for users to choose from, sometimes, the chatbot may recommend too much or too little to the user.

Table 2.4.1: Set of Hotel Information [7]

name	distance	category	parking	price
Hotel 1	med	5-stars	false	med
Hotel 2	med	5-stars	true	high
Hotel 3	med	3-stars	true	low
Hotel 4	long	3-stars	true	low
Hotel 5	med	5-stars	false	high
Hotel 6	short	4-stars	true	high
Hotel 7	long	2-stars	false	low
Hotel 8	med	5-stars	false	high
Hotel 9	long	3-stars	true	low
Hotel 10	long	2-stars	false	low
Hotel 11	short	4-stars	true	high
Hotel 12	short	3-stars	false	med
Hotel 13	med	5-stars	true	high
Hotel 14	short	3-stars	false	med
Hotel 15	long	3-stars	false	low
Hotel 16	long	3-stars	true	med
Hotel 17	med	3-stars	true	low
Hotel 18	med	2-stars	false	low
Hotel 19	long	2-stars	true	low
Hotel 20	long	4-stars	true	med

To fix the issue of the chatbot providing too many recommendations, the concept of entropy is used to limit the suggestions. The entropy theory used is Shannon's information entropy. The proposed formula to calculate the entropy is described as: H(X) = $-\sum_i P(x_i) log P(x_i)$ where H(X) is the entropy for attribute X and x_i being the probability that attribute X takes value x_i. The entropy will be calculated for every attribute that is stated by the user, afterwards, the attribute with the largest entropy value and request the user to further restrict it, which reduces the amount of recommendations given.

To fix the issue of the chatbot providing too less recommendation, some sort of reasoning method, for example consistency-based reasoning, is proposed. When users enter their criteria when searching for a site or attraction that is too restrictive, the proposed chatbot will first determine the minimal set of inconsistent requirements compared to the dataset. For example, 4 attributes are given, and the inconsistent requirements are the requirements that conflict with each other compared to the dataset. Thus, if attributes 2 and 3 are conflicting, then the chatbot will prompt the user to choose 1 of these 2 attributes to relax their requirements on so that the chatbot is able to recommend more sites or locations for the tourist to choose from.

These two rectifying modules combine to create the EntRecom Algorithm. This algorithm is made up of three parts, the recommendation module which handles recommending sites or attractions to tourists and calls the other two parts of the algorithm, the Inconsistent Requirement Algorithm, which is used to limit a large amount of recommendation and the GetBestEntrAttribute, which is used to find the entropy over the large set of criteria in order to increase the limited number of recommendations. This leads to a system which displays just the right number of suggestions for users when users prompt it for recommendations for a location they want to go to.

2.4.2 Strengths of Proposed Method

The proposed method provides tourists with many recommendations whenever a user request something from the chatbot. This is advantageous as tourists may have other, smaller minor criteria for a site or attraction that they may not type into the chatbot. Giving users multiple attractions can have a higher chance to provide a suggestion that a tourist will fully enjoy.

At the same time, the chatbot also does not provide too many options. Too many options can sour tourists' experience as they will have to spend more time, energy and effort into eliminating the options that they do not want. This wastes time that tourist can use to head to a desired location and enjoy their tour or vacation.

The chatbot also excels in helping tourists that do not know what criteria they are looking for in a site or attraction as the chatbot, seeing the lack of criteria users provide, will be able to prompt and give users criteria they can filter out.

2.4.3 Critique of Proposed Method

The proposed method assumes that the user knows where they would want to go next. For example, the proposed method assumes that the user already knows they want to go to a hotel or a restaurant. For smaller towns, information regarding attractions or events is difficult to come by, even on the Internet. Users might not know what events they want to go to. The algorithm proposed in this article will not suffice in handling situations like this due to how the data needs to be stored and how the EntRecom algorithm works. This is especially true for events that last for only a few days, or yearly events.

2.5 Personalised A.I. Chatbot for Kampar Tourism Mobile Application

2.5.1 Review of Proposed Method

This chatbot, that was developed by Low Zhi Yuan [8], similar to [4], aims to create a chatbot specifically made for small towns with difficult to access information.

Just like [4], this chatbot uses Watson Assistant as the knowledge base for the chatbot. Thus, the design for [8] is similar to [4], however, this chatbot has many key differences from [4].

The chatbot contains a recommender module, which recommends locations to users based on their preferences. These preferences are taken from user's profile and will adapt to the user's profile. For this proposed system, user's preference is obtained through the app itself, where users can take a short questionnaire which users can select their preferred lodging, attractions, food, etc. This builds a user profile of the user which will be used in the recommendation step. Users are rewarded vouchers at the end of the questionnaire. Figure 2.5.1 shows how users' preferences are taken through the "My Favourite" module.

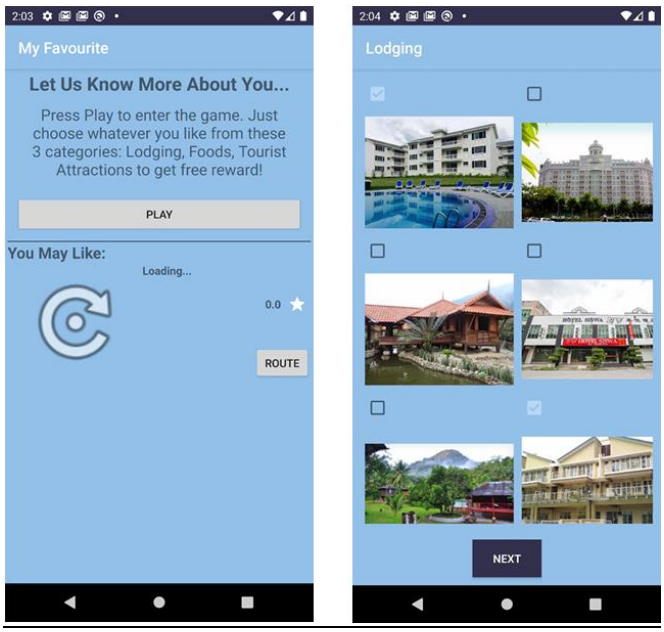


Figure 2.5.1: "My Favourite" Module to Grab User Preferences [8]

In addition, the chatbot obtains user preferences by inquiring whether the suggested location is to the tourist's liking. Users can provide feedback on whether the location suggested is to their preference (true) or not to their preferences (false). This way, the chatbot is able to constantly learn user's preferences through context-based learning.

Information regarding restaurants, lodging and attractions is stored in the database of the system. Each location is labeled based on what type of restaurant, lodging or attraction they are through one-hot encoding. Figure 2.5.2 shows how restaurants are stored in the database of the system.

location_id	name	Local foods	Café	Fast foods
0	Chan Siew Heng Claypot Chicken Rice	1	0	0
1	Restaurant Jia Kampar	1	0	0
2	Spade's Burger Kampar	0	0	1
3	Chicken Bun Restaurant Yau Kee	1	0	0
4	Villa 21 Café	0	1	0
5	Vegan Life	0	1	0
6	East Ocean Seafood Restaurant	1	0	0
7	Restaurant Fook Kee	1	0	0
8	Kam Ling Restaurant	1	0	0
9	Chill Out Bistro	0	1	0
10	Ah Boy Noodle	0	1	0
11	Medan Selera Kampar	1	0	0
12	Dai Ga Jay	1	0	0
13	Mala Hotpot	0	1	0
14	Mosel Western & Fusion Cuisine	0	1	0

Figure 2.5.2: Storing of Restaurant Information [8]

In order to generate appropriate recommendations for users, the user's preference information from their user profile, which was obtained prior will be used together with the lodging, restaurant and attraction information. First, the locations that the users have already been to are grabbed. Then, their one-hot encoded values are multiplied by the user's rating of the location, which is obtained through the user after their visit, and then it is summed up to obtain a weight value for each type of location. Afterwards, each value of every location is multiplied by the weight previously obtained and finally, the final weight of the locations is obtained by taking the rating value for each location divided by the sum of the user's profile to obtained a normalized weighted user preference for recommendation. Locations with higher weight values have a higher probability of being preferred by the user and thus will be recommended to the user when they make a query for a restaurant, lodging or attraction. Users can then have the chatbot route them to the suggested location or request the chatbot to provide them with another suggestion, which the chatbot will suggest a location with the second highest weight. Figure 2.5.3 and Figure 2.5.4 shows how the weights are calculated for each restaurant. Figure 2.5.5 shows the recommender module, which is similar in design to [4].

name	Local foods	Café	Fast foods
Chan Siew Heng Claypot Chicken Rice	1x5=5	0x5=0	0x5=0
Vegan Life	0x3=0	1x3=3	0x3=0
Bak Kut The Kam Heong Food	1x2=2	0x2=0	0x2=0
Ming Kei Chee Cheong Fun	1x5=5	0x5=0	0x5=0
Gok Jie Restaurant	1x4=4	0x4=0	0x4=0
TOTAL	16	3	0

Figure 2.5.3: Weight calculated from user ratings [8]

location_id	name	Local food	Café	Fast foods	TOTAL
0	Chan Siew Heng Claypot Chicken Rice	1x16=16	0x3=0	0x0=0	16
1	Restaurant Jia Kampar	1x16=16	0x3=0	0x0=0	16
2	Spade's Burger Kampar	0x16=0	0x3=0	1x0=0	0
3	Chicken Bun Restaurant Yau Kee	1x16=16	0x3=0	0x0=0	16
4	Villa 21 Café	0x16=0	1x3=3	0x0=0	3
5	Vegan Life	0x16=0	1x3=3	0x0=0	3
6	East Ocean Seafood Restaurant	1x16=16	0x3=0	0x0=0	16
7	Restaurant Fook Kee	1x16=16	0x3=0	0x0=0	16
8	Kam Ling Restaurant	1x16=16	0x3=0	0x0=0	16
9	Chill Out Bistro	0x16=0	1x3=3	0x0=0	3
10	Ah Boy Noodle	0x16=0	1x3=3	0x0=0	3
11	Medan Selera Kampar	1x16=16	0x3=0	0x0=0	16
12	Dai Ga Jay	1x16=16	0x3=0	0x0=0	16
13	Mala Hotpot	0x16=0	1x3=3	0x0=0	3
14	Mosel Western & Fusion Cuisine	0x16=0	1x3=3	0x0=0	3

Figure 2.5.4: Weight of each Restaurant [8]

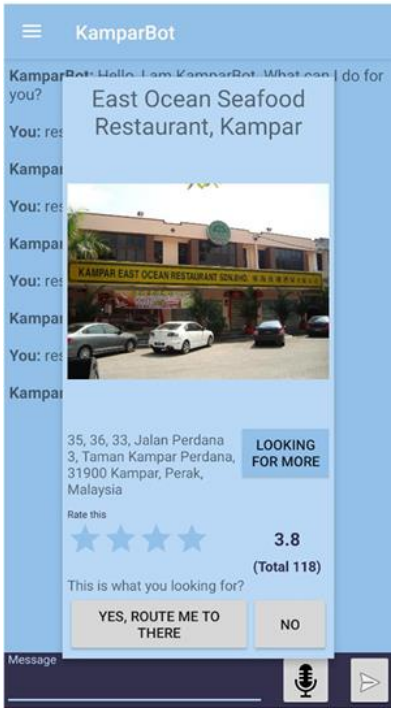


Figure 2.5.5: Recommender System of Proposed Method [8]

2.5.2 Strengths of Proposed Method

The proposed method is able to recommend locations to users not just based on each individual location, but also through the type of location. For example, the chatbot will be more inclined to recommend users that prefer local foods restaurants that serve local foods due to the algorithm the chatbot is built on. This allows the chatbot to better cater to each individual user and increase user's satisfaction with the chatbot.

In addition, the chatbot provides a voucher reward for users after taking the questionnaire that asks users of their preferred locations. This encourages users to provide user data for the chatbot which can use this to generate more personalized suggestions.

2.5.3 Critique of Proposed Method

In this proposed method, due to how user ratings influence the weights of each location type, the chatbot will heavily prioritize location types that users have already been to before and leaving little room for location types that have not even been visited before. This is bad for tourists that wants to try new things. This is also hindered by the fact that users can't specify which location type they want suggested. For example, if a user has only been to one café and leaves a rating, all remaining suggestions will prioritize cafes as cafes will have the highest weight by default due to only cafes having ratings.

Chapter 3

System Methodology/Approach

3.1 Methodology of the System

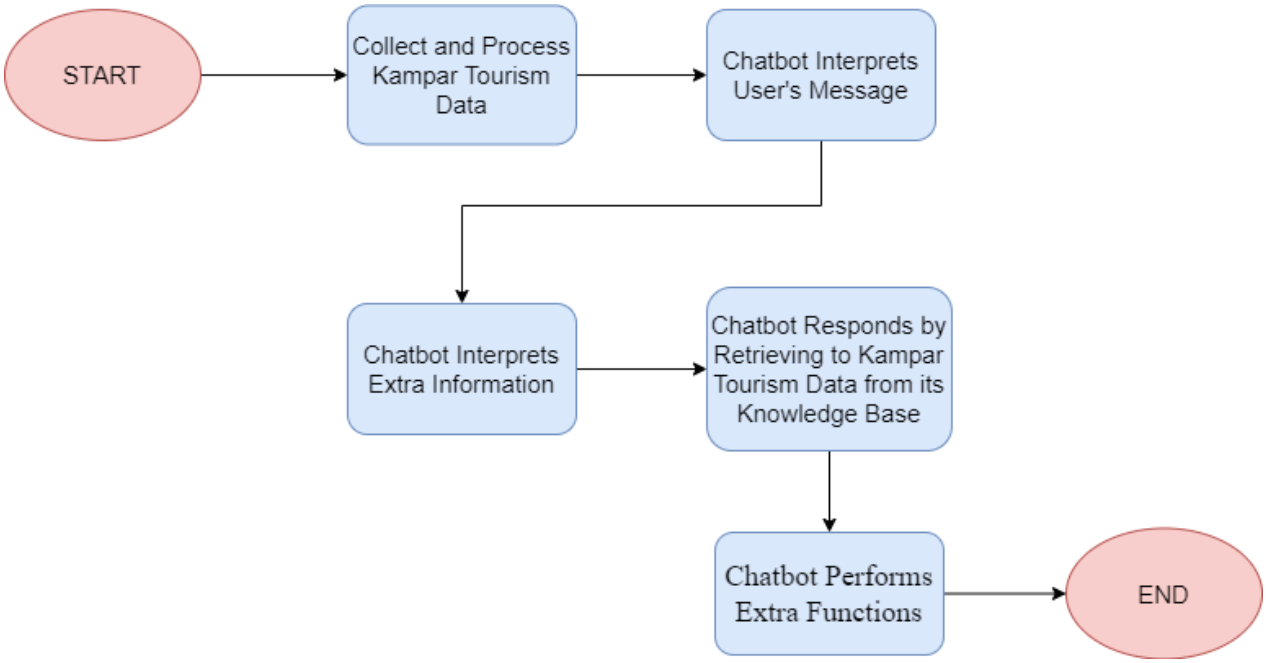


Figure 3.1.1: Methodology of the System

Building the system first starts with the collection of Kampar tourism data and tourists' reviews. This is collected so the chatbot has knowledge of the tourism data specifically for Kampar tourism and the areas around Kampar, like Ipoh, in which it can take reference and respond to queries with. Data collection will be done through web scraping of the tourism website, TripAdvisor using the Python libraries, Selenium and BeautifulSoup. Data collected is split into three types, hotels, restaurants and attractions as those are the three main locations that tourists will go during their travels.

The data collected will then be processed to fill in missing information, to remove unwanted and redundant information, and to supplement extra information not already provided by TripAdvisor. Data processing is done both manually and with Python, whenever applicable.

Once the data is collected and processed, the chatbot is then built. The chatbot uses OpenAI Assistant as the basis of the chatbot and GPT-4o mini for the chatbot's Large Language Model.

The assistant is trained to understand that it is a tourism chatbot and it should respond to users as if it is a tour guide situated and specialising in Kampar.

The assistant works by first receiving user's input. The GPT-4o mini Large Language Model provided by OpenAI will then interpret the user's input to determine what the user wants. In addition, the chatbot is also trained to understand and interpret additional information that can be given by the user and how to respond to them. Such as the user's travelling preferences which can influence what attractions, restaurants or hotels users prefer to go to; and environmental factors such as the current weather in Kampar, time in Kampar, and user's location in Kampar. The chatbot is made to account for these factors when providing suggestions for users, allowing the chatbot to be personalized for every user as well as the user's environment.

After interpreting both the user's enquiries and additional information, the chatbot will refer to its knowledge base in order to provide the most optimal answer for users. The knowledge base of the chatbot is then built using the data collected and processed previously. The data is then vectorized through OpenAI's algorithm and stored as a vector, in which the chatbot will perform appropriate retrieval through a variation of RAG in order to answer user's queries. Thus, based on the context of the user's enquiry and additional information, appropriate data will be retrieved to answer the enquiry.

Finally, the chatbot is also made to perform additional functions after responding to the message should the user requests for it. For example, updating the user's travelling preferences in real time to match the chat history and directing users to a specific location.

3.2 Chat Suggestion and Retrieval Methodology

The tourism chatbot will be built using a Large Language Model, more specifically, OpenAI's Large Language Model, GPT-4o mini. A language model allows a deeper understanding of user's sentences through a probability distribution over words that will most likely be the most appropriate [9].

However, while it can understand and respond to general enquiries very well, it cannot respond well to specific enquiries, such as what time does a restaurant close. This is because, while a Large Language Model like GPT-4o mini has some the tourism information, it is not trained on much more specific information during its creation. As such, the model will simply provide

either simple, not specific, false or ambiguous information [10]. In addition, since the GPT model is not updated often, the tourism data provided by the language model alone will most likely be outdated, especially since the tourism landscape is always changing with the shutting down and reopening of restaurants, attractions and hotels. A user's traveling experience can be soured if they were to be given this outdated data. This makes large language models unreliable for cases such as this. The system developed in this project also faces this issue.

In order to solve this problem, the model needs access to this specific knowledge that it was not trained on during its development. Thus, the system developed in this project utilizes Retrieval-Augmented Generation to retrieve and understand Kampar tourism information and help provide much more specific and precise suggestions and information of places from Kampar for users.

Retrieval-Augmented Generation involves augmenting the response given by an LLM by retrieving external information provided by a developer [11]. In this case, this system aims to augment the OpenAI LLM's responses by retrieving Kampar tourism data, which will be obtained through web scraping. This external data acts as the knowledge base for the OpenAI LLM and is stored in a vector database through vectorization. OpenAI Assistant uses its own variation of RAG, and this system uses this methodology to provide Kampar tourism chatbot functions to users.

In this system, vectorization involves feeding Kampar tourism data to the OpenAI Assistant. OpenAI uses a technique called embeddings to store and embed the files and perform knowledge retrieval [12]. Figure 3.2.1 shows the process of embedding, where every word in a file is converted into a vector. The vectors will then be stored in a vector database as an index, which tells the OpenAI algorithm where each word is in a file.

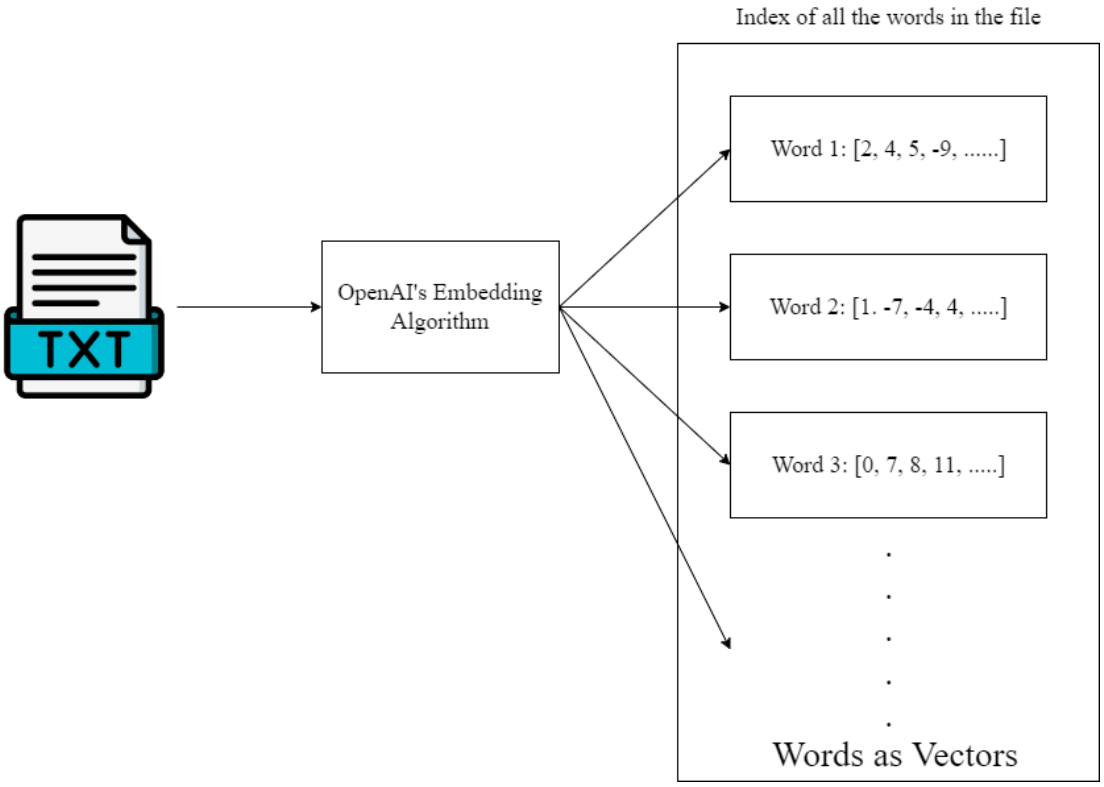


Figure 3.2.1: OpenAI Converting Words from a File to Vectors

For the retrieval process of OpenAI Assistant's Retrieval-Augmented Generation, when OpenAI receives the user's message and traveling preferences, the assistant will convert the message and preferences into vectors as well. OpenAI will then refer to the vector data containing embeddings of the tourism data and tourists' reviews that was fed into it and will retrieve the strings of words whose vector's distance is close to the vector of the message and preferences, as that suggest relatedness [13]. This allows the model to grab accurate suggestions from the tourism data that was fed. This also allows the model to predict the user's emotional context based on their word choice used. The OpenAI's large language model, GPT-4o mini, will then formulate a response based on what was grabbed. Figure 3.2.2 shows the algorithm in the context of grabbing suggestions.

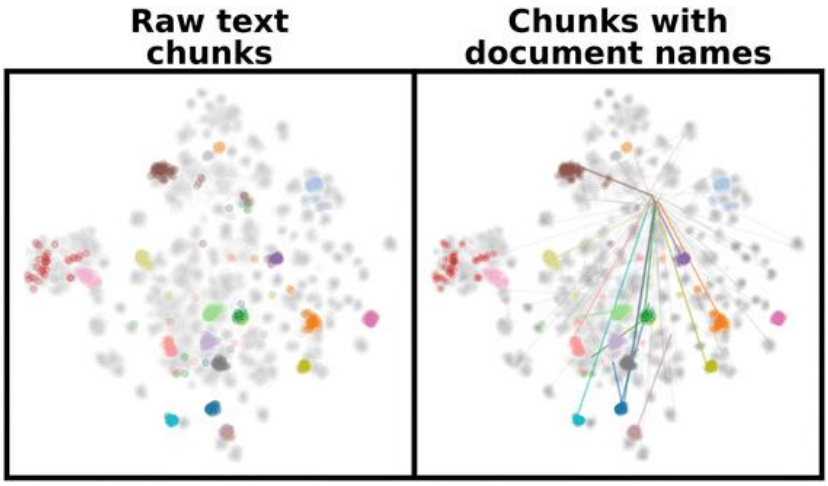


Figure 3.2.2: Visualization of OpenAI Embeddings and Suggesting Algorithms [20]

The assistant will provide 3 suggestions per inquiry. This is so that users have other options should they find the initial suggestion is not suitable for them due to certain circumstances. The chatbot will also provide certain traveling information for the user. For example, user may ask what a certain cuisine is, or more information about an attraction. The assistant will then refer to the associated scrapped file and its own general knowledge base to answer the user's questions. Figure 3.2.3 shows the overall process of knowledge retrieval, embedding and providing suggestion from user's prompt.

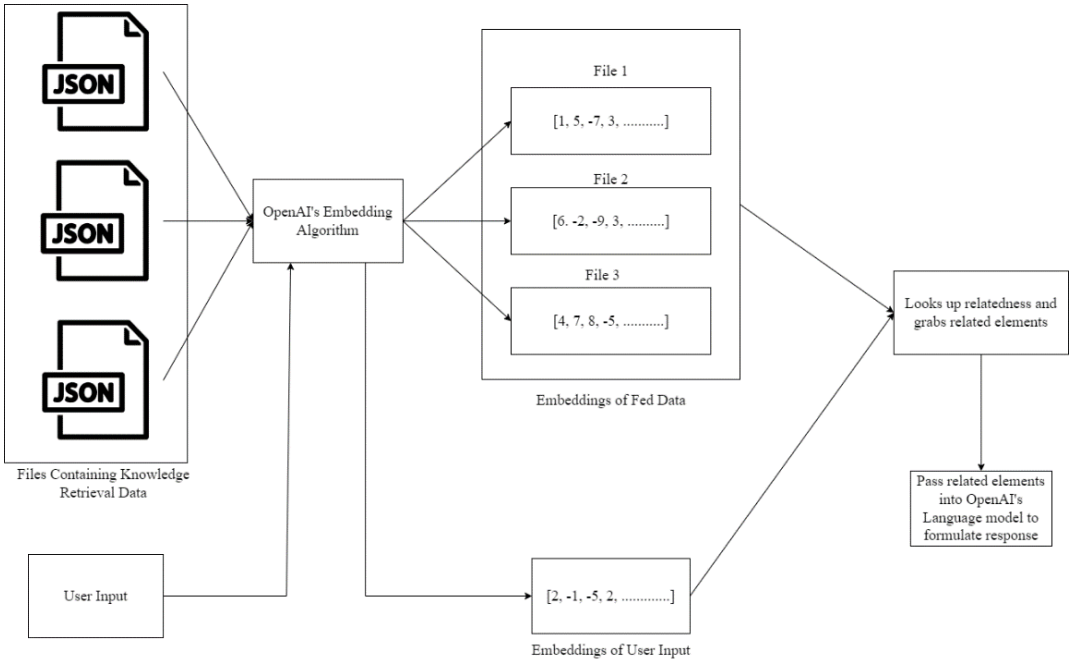


Figure 3.2.3: Overall Process of Knowledge Retrieval, Embedding and Providing Suggestions

3.3 System Architecture

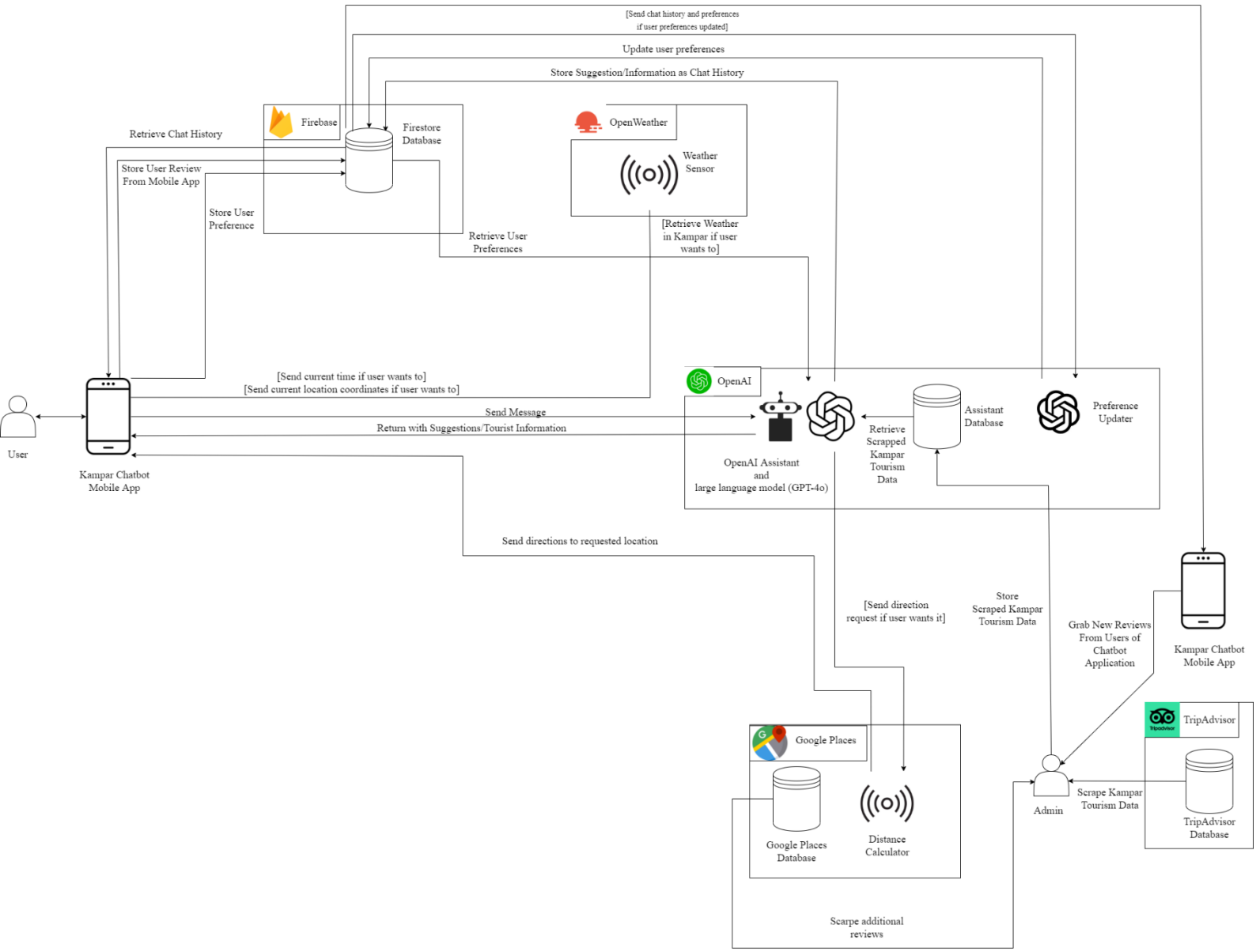


Figure 3.3.1: System Architecture

As seen in Figure 3.3.1, a total of five services or websites are used in this system's architecture. The data needed by the chatbot in order for it to understand what tourist destinations it can recommend; their details and their tourists' reviews are obtained from the TripAdvisor database through web scraping. This scrapped data will be obtained by the admin.

The scrapped data will then be processed by the admin. During this process, additional tourists review will be obtained as a larger number of reviews can increase the accuracy of the Kampar

chatbot. Thus, additional reviews for the hotels, restaurants and attractions scrapped from TripAdvisor will be supplemented with reviews from Google using Google Places API.

The scrapped and processed data will be fed into the OpenAI Assistant that is developed and trained to act like a tour guide which will provide suggestions to users. The fed data is then stored into a vector database by the assistant to be read through vectorization and retrieved from the vector database whenever a user makes a request through sending a message.

The Firestore Database located in Firebase is used to store the user's chat history and the user's traveling preferences. The user's chat history and traveling preferences will be stored and retrieved based on their unique user id that is provided by Firebase whenever they register for an account. Thus, a user can only access their chat history and traveling preferences are stored separately per person. The stored traveling preferences will be read by the OpenAI Assistant whenever it needs to make a traveling suggestion for the user. Storing travelling preferences is important as it gives the chatbot the personalization component. It allows tourists with different taste to get different responses which aligns with their preferences. For example, a user which prefers seafood and a user which prefers pastries can have their chatbot to be personalized in a way to provide different restaurants based on those preferences. The assistant will use the preferences to further personalize the response given to the user.

Whenever the user sends a message, it will be sent through the OpenAI API and will be passed into the OpenAI Assistant. The OpenAI Assistant will take the user's message, along with their traveling preferences from the Firestore Database in Firebase and the scraped Kampar tourism information it has been fed with in order to formulate an appropriate, and personalized response using the OpenAI's large language model, which is GPT-4o. For example, if the user asks for a restaurant and they have stated they wanted local food in their traveling preferences, the OpenAI Assistant will only retrieve Kampar's restaurant information from its database in order to find a suitable one for the user.

In addition, users are also given the option to send the current weather in Kampar, the current time and their current location to the OpenAI Assistant whenever they send a message. This is to further personalize the responses from the chatbot to fit the current weather, time and user's current location. For example, should it be a rainy afternoon, the OpenAI Assistant will not recommend hiking as it is an inappropriate activity under that environmental condition. It is natural for users to, when they do not have a specific place they want to go to, to be suggested places that are close to them instead of places far away as it saves them them, Thus, the OpenAI

Assistant will take the user's location into account should the user sends it by recommending locations much closer to the user's current location. The weather data is obtained through OpenWeather through its API whenever the user sends a message and is willing to send Kampar's current weather data. While the current time and user's location is obtained through the user's device itself whenever the user sends a message and is willing to send their time and location.

Furthermore, users could ask the chatbot for directions to a location. When this happens, the OpenAI Assistant will forward the request to Google Maps using Google Places API, in which it will calculate the travel path the user needs to take and sends it back to the user.

Moreover, users have the option to update their traveling preferences after every message. This is to allow the chatbot to adapt to the user's changing needs during their vacation, as preferences can change over time due to several factors. For example, a user might gain a preference to a certain type of restaurant during their travels which they might not have a preference to before, or the unique culture and traditions makes the user prefer a place or activity only in the travelling location. If users allow their traveling preferences to update after every message, after the chatbot returns a response to the user, the user's current chat history is retrieved from the Firestore Database and sent to another chatbot made to update the user's preferences. The chatbot will then update the preferences based on the user's chat history, taking into account what the user had asked and what the chatbot had recommended to the user. The updated preferences are then sent back to the Firestore Database to be stored.

Users will also be able to send reviews of Kampar locations in the application itself. This is done so that user's reviews can further be integrated into the chatbot's knowledge base, allowing the users to continuously improve the application by filling the chatbot's knowledge base with more reviews the more it is used. However, currently the reviews are not submitted to the knowledge base due to time constraints.

Finally, the admin of the application will be able to re-scrape tourism data automatically upon logging into the app. This allows them to update the chatbot's knowledge base periodically without needing to enter into the OpenAI console. During this process, all the user reviews that are sent through the application will be gathered. Web scraping is also supposed to be re-conducted as well; however, it is not implemented currently due to time constraints. As stated previously, the re-scraped reviews from both the Internet and the user reviews from the

application will not upload itself to the knowledge for this version of the application due to time constraints.

3.4 Use Case Diagram and Description

The following will detail the use case diagram and description of the chatbot application itself, including explanation of basic functions of the chatbot such as account creation and chat history deletion. Figure 3.4.1 shows the Use Case Diagram for the application.

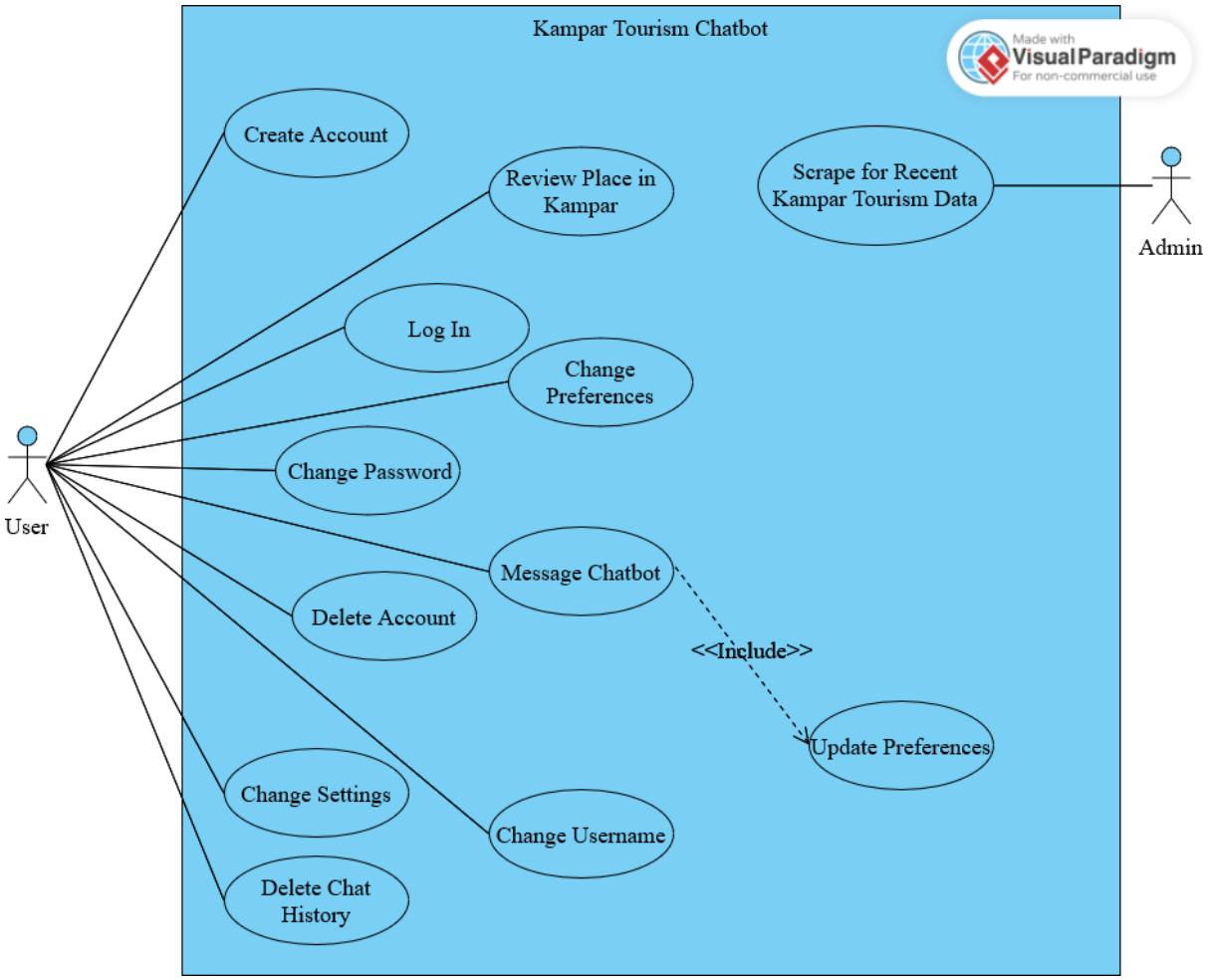


Figure 3.4.1: Use Case Diagram

Table 3.4.1: Use Case Description for Create Account

Use Case ID	UC001	Version	1.0
Use Case	Create Account		

Purpose	To create an account for the Kampar Tourism Chatbot so that user's data such as their travelling preferences and chat history can be saved individually	
Actor	User	
Trigger	User selects 'Register' in sign in menu	
Precondition	User is in sign in menu	
Scenario name	Step	Action
Main Flow	1	User selects 'Register' in sign in menu
	2	System prompts user for email
	3	System prompts user for password
	4	System prompts user to confirm password
	5	User selects 'Register'
	6	Account is created a registered in Firebase.
	7	Firestore Database opens database for new user to store chat history and preferences
	8	User is brought to main menu
Alternate Flow – Not a valid email	2.1	System detects input in email column is not a valid email
	2.2	System displays 'Provide a valid email' and prompts User to input a valid email
	2.3	Back to Main Flow 2
Alternate Flow – Confirm password does not match initial password	4.1	System detects confirm password does not match with initial password
	4.2	System displays 'Passwords do not match' and prompts User to input a valid password
	4.3	Back to Main Flow 4
Rules		-
Author		Chua Feng Zhao

Table 3.4.2: Use Case Description for Log In

Use Case ID	UC002	Version	1.0
Use Case	Log In		
Purpose	To log in to the account created for the Kampar Tourism Chatbot so Users can access their chat history and personalized preferences		

Actor	User	
Trigger	-	
Precondition	User is in sign in menu	
Scenario name	Step	Action
Main Flow	1	User enters email
	2	User enters password
	3	User selects ‘Sign In’
	4	System verifies with Firebase Authentication to see if email and password combination is correct
	5	System brings user to main menu
Alternate Flow – Not a valid email	1.1	System detects input in email column is not a valid email
	1.2	System displays ‘Provide a valid email’ and prompts User to input a valid email
	1.3	Back to Main Flow 1
Alternate Flow – Email and password combination does not match account stored in Firebase	4.1	System detects email and password combination does not match the accounts in Firebase
	4.2	System displays ‘The supplied auth credential is incorrect, malformed or has expired’ and prompts User to input a valid password
	4.3	Back to Main Flow 4
Rules		-
Author		Chua Feng Zhao

Table 3.4.3: Use Case Description for Change Password

Use Case ID	UC003	Version	1.0
Use Case	Change Password		
Purpose	To allow users that forgot their password to change it		
Actor	User		
Trigger	-		
Precondition	User is in sign in menu		

Scenario name	Step	Action
Main Flow	1	User selects ‘Forgotten Password?’
	2	System prompts User to enter email of their account
	3	User selects ‘Reset Password’
	4	System sends email to address User entered in which users can reset their password with
	5	User selects ‘Go Back’
Rules		-
Author		Chua Feng Zhao

Table 3.4.4: Use Case Description for Delete Account

Use Case ID	UC004	Version	1.0
Use Case	Delete Account		
Purpose	To delete the current account		
Actor	User		
Trigger	-		
Precondition	User is in main menu		
Scenario name	Step	Action	
Main Flow	1	User selects ‘Account’	
	2	User selects ‘Delete Account’	
	3	Account is removed from Firebase	
	5	System brings user to sign in menu	
Rules		-	
Author		Chua Feng Zhao	

Table 3.4.5: Use Case Description for Change Settings

Use Case ID	UC005	Version	1.0
Use Case	Change Settings		
Purpose	To change the various settings of the application, such as what additional information to be sent to the chatbot		

	when users message it and whether the chatbot will update user preference in real-time	
Actor	User	
Trigger	-	
Precondition	User is in the main menu	
Scenario name	Step	Action
Main Flow	1	User selects ‘Settings’
	2	User toggles settings they want to enable or disable
	3	Enabled/disabled settings are saved
Rules		-
Author		Chua Feng Zhao

Table 3.4.6: Use Case Description for Delete Chat History

Use Case ID	UC007	Version	1.0
Use Case	Delete Chat History		
Purpose	To delete the user’s chat history with the chatbot		
Actor	User		
Trigger	-		
Precondition	User is in the main menu		
Scenario name	Step	Action	
Main Flow	1	User selects ‘Settings’	
	2	User selects ‘Delete Chat History’	
	3	User selects ‘YES’	
	4	Firebase deletes all chat history associated with the user	
Alternate Flow – User selects ‘NO’	3.1	User selects ‘NO’	
	3.2	System returns to settings menu with no changes to their chat history	
Rules		-	
Author		Chua Feng Zhao	

Table 3.4.7: Use Case Description for Change Preferences

Use Case ID	UC007	Version	1.0
Use Case	Change Preferences		
Purpose	To allow users to change their travelling preferences for personalization purposes		
Actor	User		
Trigger	-		
Precondition	User is in the main menu		
Scenario name	Step	Action	
Main Flow	1	User selects ‘Edit Preferences’	
	2	User chooses to change the preferences shown or to leave the preferences empty	
	3	User selects ‘Submit’	
	4	System updates preferences into Firebase, with the system providing Firebase with “User does not want to share this” or “User does not have a preference” if user left section empty.	
Rules	-		
Author	Chua Feng Zhao		

Table 3.4.8: Use Case Description for Change Username

Use Case ID	UC008	Version	1.0
Use Case	Change Username		
Purpose	To allow users to change the name shown in main menu		
Actor	User		
Trigger	-		
Precondition	User is in the main menu		
Scenario name	Step	Action	
Main Flow	1	User selects ‘Account’	
	2	User types their desired name in ‘Name’	
	3	User clicks on tick	

	4	Firebase Authentication updates username and it is displayed in main menu
Rules		-
Author		Chua Feng Zhao

Table 3.4.9: Use Case Description for Message Chatbot

Use Case ID	UC009	Version	1.0
Use Case	Message Chatbot		
Purpose	To message the chatbot in order to obtain suggestions for the user's travel in Kampar		
Actor	User		
Trigger	-		
Precondition	User is in the main menu		
Scenario name	Step	Action	
Main Flow	1	User selects 'Explore Kampar Today!'	
	2	User types their desired inquiry	
	3	System grabs user's travelling preferences from user's Firebase account database	
	4	Language model interprets message, preferences and, if any, additional information sent	
	5	OpenAI Assistant retrieves relevant information from knowledge base based on inquiry	
	6	User's and chatbot's message are stored in the user's Firebase database	
Sub Flow – User allows current weather to be sent	3a.1	System obtains Kampar's current weather from OpenWeather and sends to chatbot as additional information	
	3a.2	Back to main flow 4	
Sub Flow – User allows current time to be sent	3b.1	System obtains current time from user's device and sends to chatbot as additional information	
	3b.2	Back to main flow 4	
Sub Flow – User allows current coordinates to be sent	3c.1	System obtains current coordinates from user's device and sends to chatbot as additional information	
	3c.2	Back to main flow 4	
Sub Flow – User is asking for suggestions	5a.1	System returns 3 suggestions with explanation for users	

	5a.2	Back to main flow 6
Sub Flow – User is asking for information	5b.1	System returns information of the place, culture or cuisine user asked for
	5b.2	Back to main flow 6
Sub Flow – User is asking for direction and allows chatbot to provide directions	5c.1	System returns message which links to Google Maps through Google Places
	5c.2	User's and chatbot's message are stored in the user's Firebase database
	5c.3	User clicks on message
	5c.4	System passes request of destination, alongside user's location, to Google Places
	5c.5	Google Places generates map with path from user's location to destination location
Alternate Flow – User's message failed to send	2.1	The system returns error message
Alternate Flow – User is asking for directions but does not allow chatbot to provide directions	5d.1	System returns "Sorry! I can't guide you to this destination at this time. Please check your settings and try again!"
	5d.2	Back to main flow 6
Rules		-
Author		Chua Feng Zhao

Table 3.4.10: Use Case Description for Update Preference

Use Case ID	UC010	Version	1.0
Use Case	Update Preferences		
Purpose	To allow the system to update the user's preference based on their chat history and old preferences if the user allows it		
Actor	User		
Trigger	-		
Precondition	User messaged the chatbot		
Scenario name	Step	Action	
Main Flow	1	System grabs chat history and user's old preferences from their Firebase database	
	2	System sends the chat history and user's old preferences to the preference updater chatbot through OpenAI API	
	3	OpenAI language model analyzes both chat history and old preference	

	4	Preference updater chatbot returns updated preferences based on the chat history and old preference
	5	System updates user's new travelling preferences in their Firebase database
Rules		-
Author		Chua Feng Zhao

Table 3.4.11: Use Case Description for Review Place in Kampar

Use Case ID	UC011	Version	1.0
Use Case	Review Place in Kampar		
Purpose	To allow users to review places in Kampar.		
Actor	User		
Trigger	-		
Precondition	User is in main menu		
Scenario name	Step	Action	
Main Flow	1	User selects 'Send a Review'	
	2	User selects what type of locations they want to review	
	3	User selects the location they want to review	
	4	User provides Review Title	
	5	User provides Review Text	
	6	User provides Review Score, from 1 - 5	
	7	User selects 'Submit'	
	8	Review is stored in Firestore Database	
Sub Flow – Hotels are selected	2a.1	System provides list of hotels to provide review for	
	2a.2	Back to main flow 3	
Sub Flow – Restaurants are selected	2b.1	System provides list of restaurants to provide review for	
	2b.2	Back to main flow 3	
Sub Flow – Attractions are selected	2c.1	System provides list of attractions to provide review for	
	2c.2	Back to main flow 3	
Rules		-	
Author		Chua Feng Zhao	

Table 3.4.12: Use Case Description for Scrape for Recent Kampar Data

Use Case ID	UC012	Version	1.0
Use Case	Scrape for Recent Kampar Data		
Purpose	To update the chatbot's knowledge base with the most recent Kampar travelling information		
Actor	Admin		
Trigger	-		
Precondition	Admin logs in		
Scenario name	Step	Action	
Main Flow	1	System prompts Admin if they want to scrape Kampar tourism data	
	2	Admin selects 'YES'	
	3	System queries Firestore Database for all reviews of Kampar Locations made by users	
	4	System places reviews in appropriate places in the correct file	
Alternate Flow – Admin selects 'NO'	2.1	Admin selects 'NO'	
	2.2	System returns to main menu with no action done	
Rules		-	
Author		Chua Feng Zhao	

Chapter 4
System Design

4.1 Web Scraping Design

As mentioned in Chapter 3, in order for the chatbot to recommend the restaurants, hotels and attractions to users following their requirements and specifications, the chatbot must first know the various restaurants, hotels and attractions of Kampar and the areas surrounding Kampar, such as Ipoh. Including what they offer in their services, and what other users think of these places so that the chatbot can use this information to formulate recommendations for users. Thus, tourism and traveling data of the restaurants, hotels and attractions are acquired in order to move forward with the project. Web scraping is conducted in order to obtain the data in a quick and efficient manner.

In this project, the data is scrapped from TripAdvisor, a popular tourism website that contains information such as tourist's reviews, description of the tourist destinations and tourist's ratings of the locations. Web scraping is conducted using Python. This is because there are multiple useful premade libraries that are useful for web scraping for Python. This project uses two of these libraries, BeautifulSoup and Selenium.

Both BeautifulSoup and Selenium are web scraping libraries for Python, however BeautifulSoup is simpler but resource efficient and Selenium allows for website interaction, such as webpage navigation through simulating clicks [14]. Figure 4.1.1 shows how both BeautifulSoup and Selenium are used in scraping data from TripAdvisor.

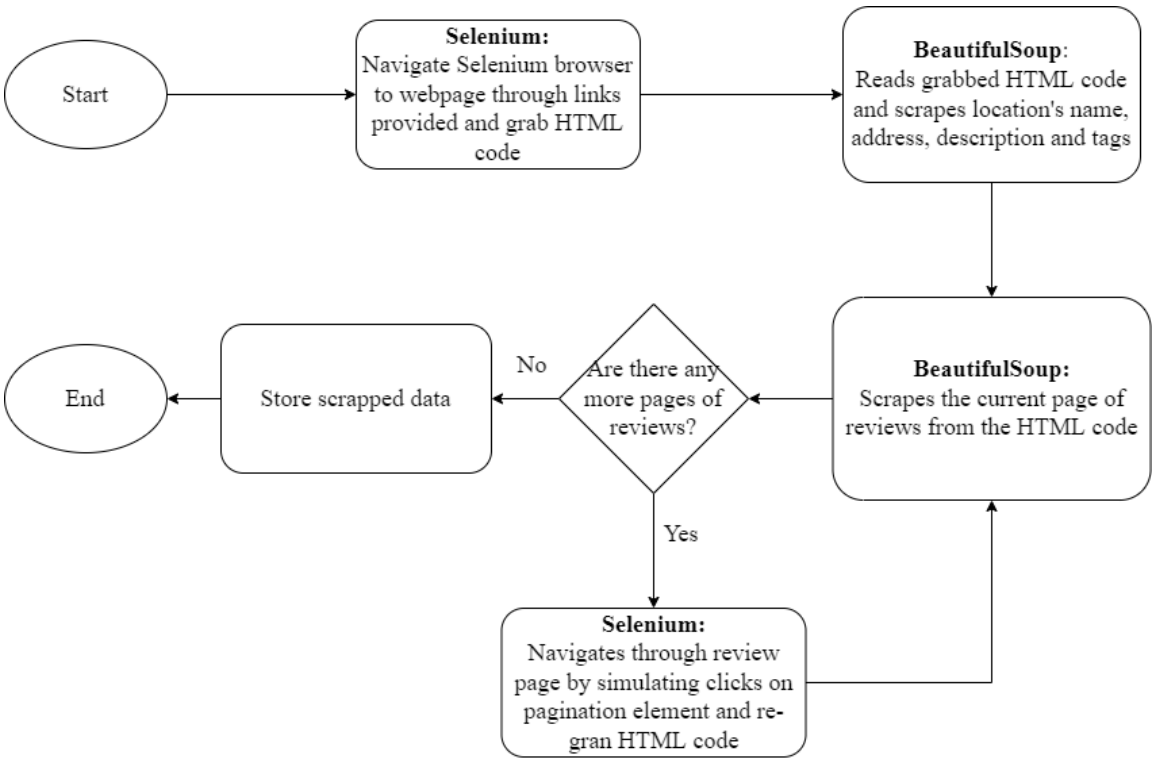


Figure 4.1.1 Web Scraping Design

Selenium is initially used because it is able to simulate and automate web browsing functions [15]. Selenium is used to automate the navigation of a web browser to the TripAdvisor page containing the restaurant, hotel or attraction from Kampar that we want to scrape, Selenium will then be used to grab the webpage's raw HTML code to be read by BeautifulSoup for scraping. This allows for some browser settings to be implemented to reduce the chance of being caught by TripAdvisor's bot detection system, which will stop the scraping if it detects the program as a bot. Some of these settings include implementing a user agent and to implement delays during the navigation process to simulate a real user.

BeautifulSoup is then used to scrape the location's name, address, descriptions and tags. BeautifulSoup will read the raw HTML code taken by Selenium and will search for specific HTML tags and classes that will store the tourism information that we want to collect. Since BeautifulSoup is more resource efficient, it is chosen over selenium. In addition, switching web crawler methods can further reduce the chance of being caught by TripAdvisor's bot detection system as the selenium browser will be idle before TripAdvisor can detect its activity. The website structure for the restaurants, hotels and attractions in TripAdvisor are different from

one another. Thus, three separate applications with varying code are written to accommodate to the difference in structure in the three types of tourist locations in TripAdvisor.

BeautifulSoup is also used to scrape the tourists' reviews of the location. In TripAdvisor, only 10 or 15 reviews can exist in one page. In order to see the remaining reviews, users will have to navigate through the pagination bar. This can be seen in Figure 4.1.2.

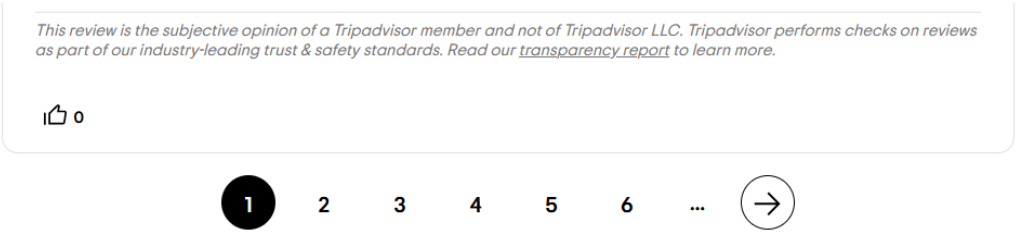


Figure 4.1.2: TripAdvisor Review Pagination [16]

Selenium is used to navigate through the pages of reviews as it can automate website functions. This is done by having the selenium browser simulate a click on the HTML element that is responsible for navigating the review pagination. Selenium will then re-grab the HTML code to gran the new page of reviews to allow BeautifulSoup to scrape the new reviews. This is done until all pages of reviews are scrapped.

4.2 Data Processing Design

Once the data has been scrapped, the data needs to be processed before it can be vectorized to be used as the knowledge base of the chatbot. This is because some data may be missing, inconsistent or redundant which is not desirable to be used as information in which the chatbot can retrieve. In this project, data processing is done using two methods, manually and through a Python program which connects to the Google Places API. Figure 4.2.1 shows the design used to process the data scrapped.

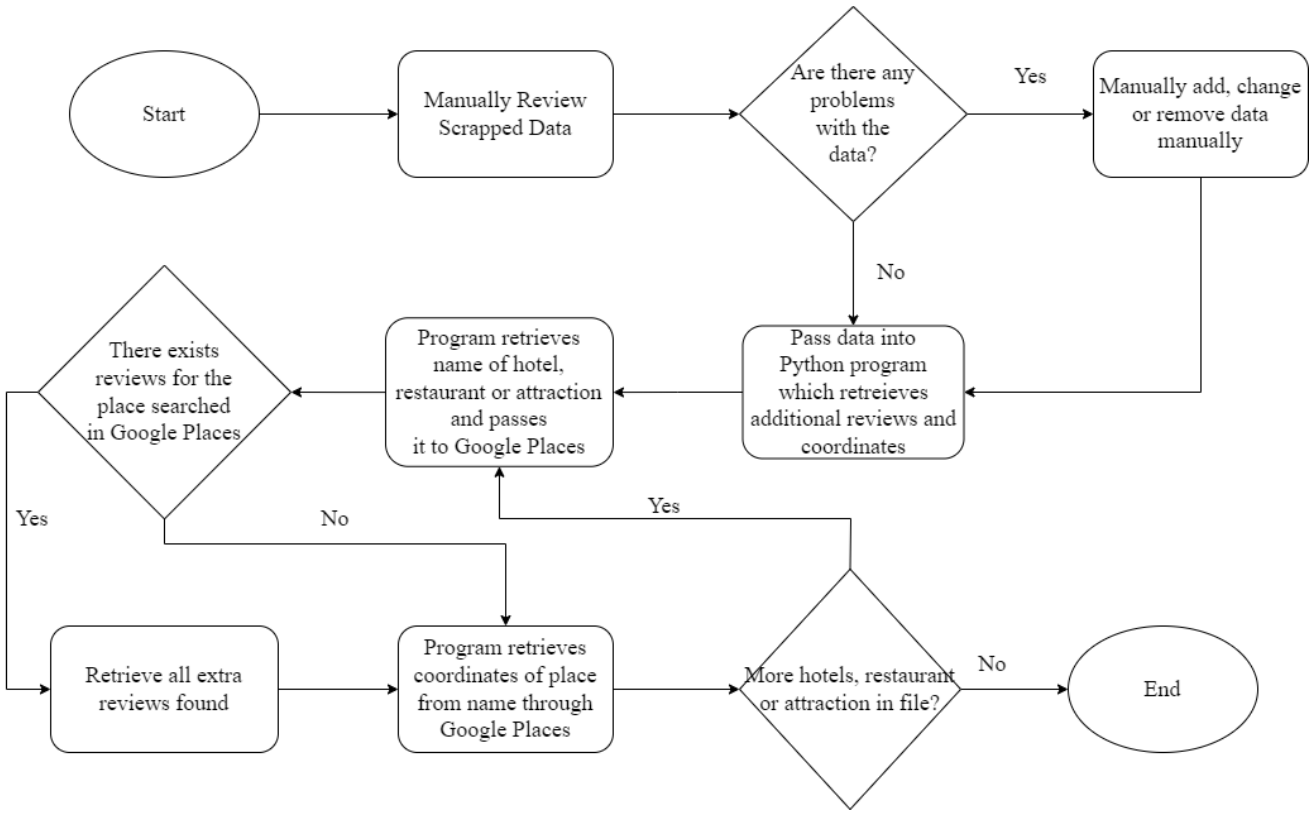


Figure 4.2.1: Data Processing Design

The data scrapped is first manually reviewed for any problems that could exist, such as missing data, inconsistent data and so on. Should there be any problems encountered, the problems are manually dealt with. This process is done manually as a automated method could be unreliable due to how scattered information could be and because web scraping proves to be an unreliable method of automation as companies are using stronger methods to safeguard the safety of their websites and data.

Once the issues in the scrapped data have been resolved, or if there are no problems in the data, the data will then be passed into a program which is connected to the Google Places API. This is done so that additional reviews for the hotels, restaurants and attractions can be obtained from Google. These additional reviews will be able to increase the accuracy of the chatbot as it increases the pool of reviews that it can retrieve from. The program does this by looping through each hotel, restaurant and attraction and searching their names through Google using the Google Places API. If additional reviews are found, then the reviews are all retrieved and stored. Once this is done, the location's coordinates are also grabbed using Google Places API.

The coordinates of the place are obtained so that the retrieval algorithm can refer and retrieve coordinate values of the locations instead of the address itself, which is a more reliable way to track distance between places as the chatbot cannot understand addresses by itself [17]. This process is repeated until every hotel, restaurant and attractions are processed.

4.3 Mobile Chatbot Application Design

Figures 4.3.1 and 4.3.2 show the system design diagram for the system flow of the mobile application that will house the Kampar tourism chatbot.

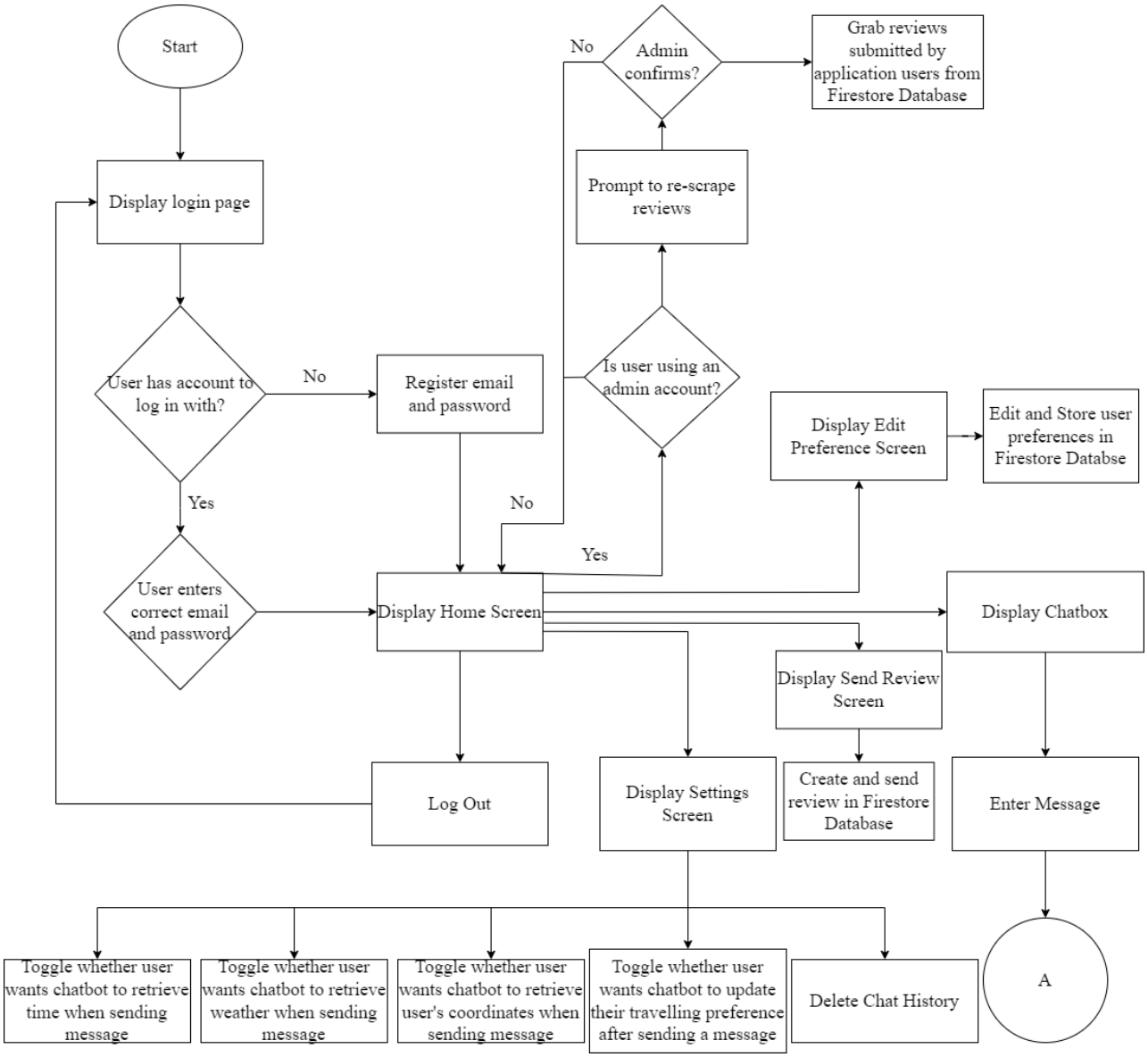


Figure 4.3.1: Mobile Chatbot Application Design (1)

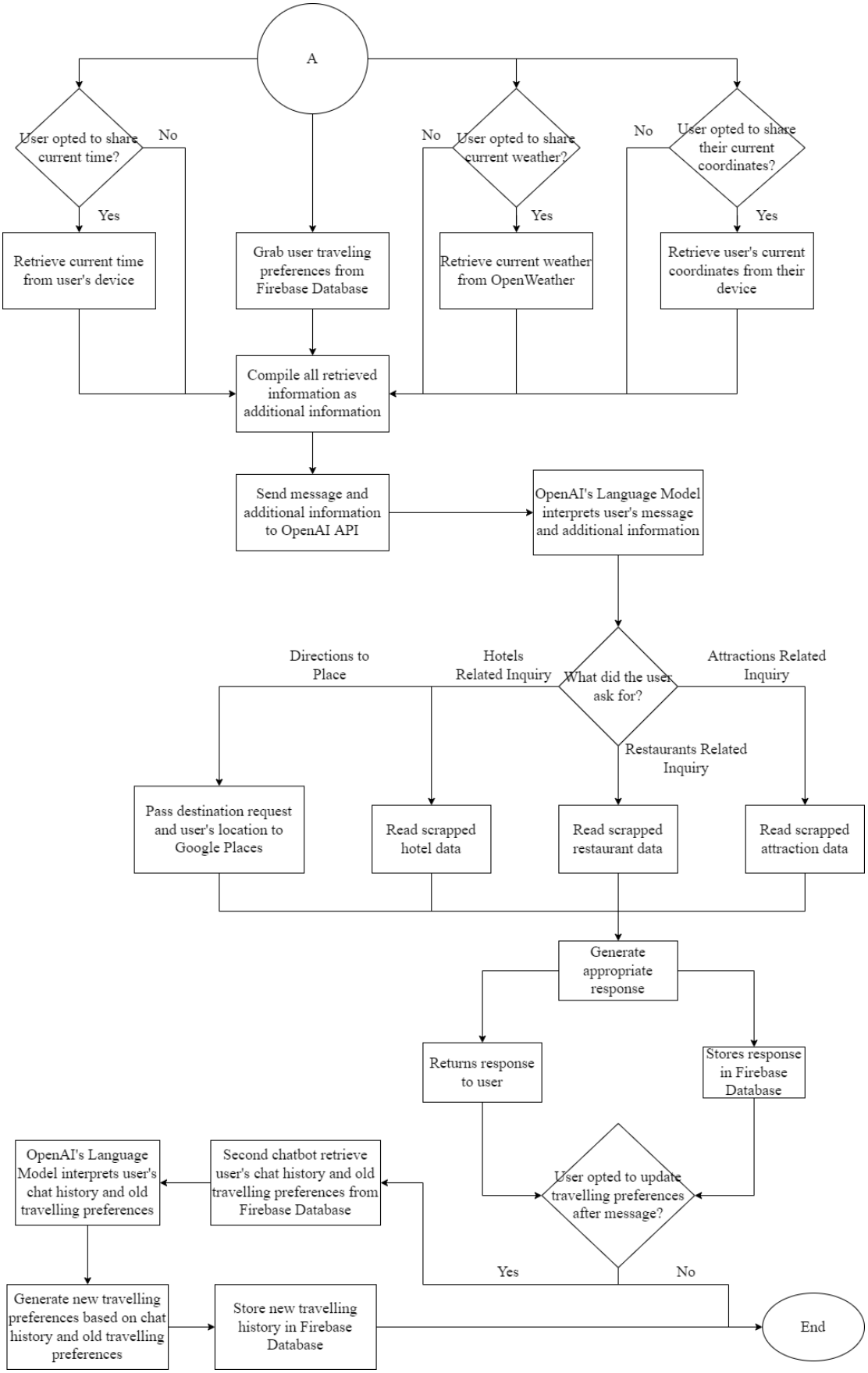


Figure 4.3.2: Mobile Chatbot Application Design (2)

Figure 4.2.3 shows the logo used for the Kampar Tourism Chatbot application.

Figure 4.2.3: Kampar Tourism Chatbot Logo

The mobile application will begin by asking the user for their login information. Should the user not have an account, they can register one by verifying their email. Users will have the create an account because this is the method the chatbot will use to personalize its suggestion outputs. User preferences and chat history will be stored separately per account. The authentication process and data storage are handled by Firebase's Authentication and Firestore Database respectively. Figure 4.2.4 shows the login page of the application.

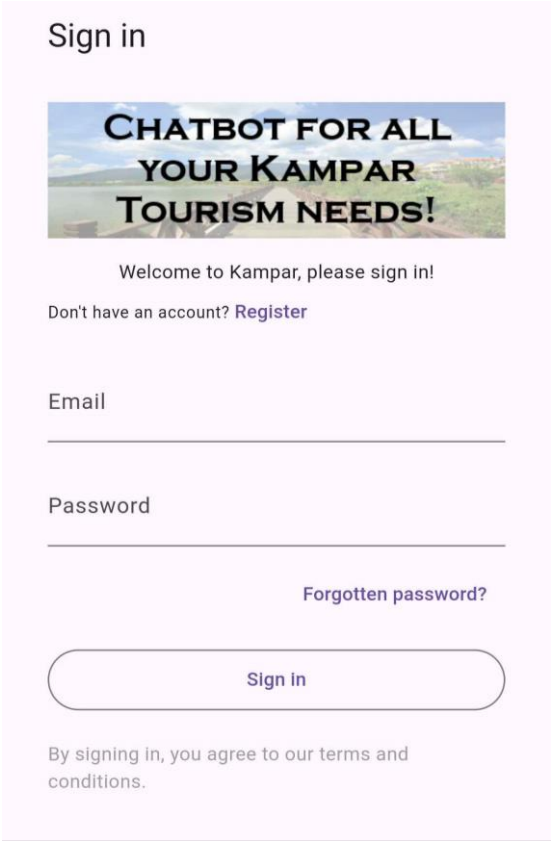


Figure 4.2.4: Application Sign In Page

Once users have successfully log into their account, users will be presented with a few options. Figure 4.2.5 shows the application's home menu.

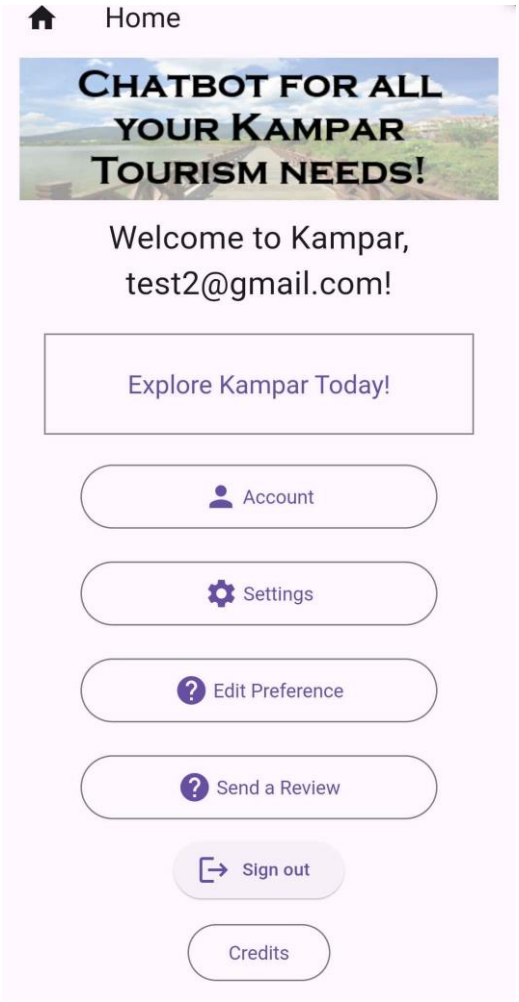


Figure 4.2.5: Application Home Screen

If the user is using the admin account, the user will be prompted to re-scrape tourism information in order to update the tourism information in the OpenAI Assistant's vector database. This is done to keep the information stored in the chatbot to be up to date. For example, restaurants could shut down and if the tourism information is not updated, the chatbot could suggest to users places that do not exist anymore which can sour users' experiences with the chatbot. Figure 4.2.6 shows the prompt for the admin. When 'Yes' is clicked, the reviews submitted by users through this application will be grabbed and stored. However, re-scraping

of websites like TripAdvisor and the updating of knowledge base is not done for this version of the application. This can be work in which can be improved in the future.

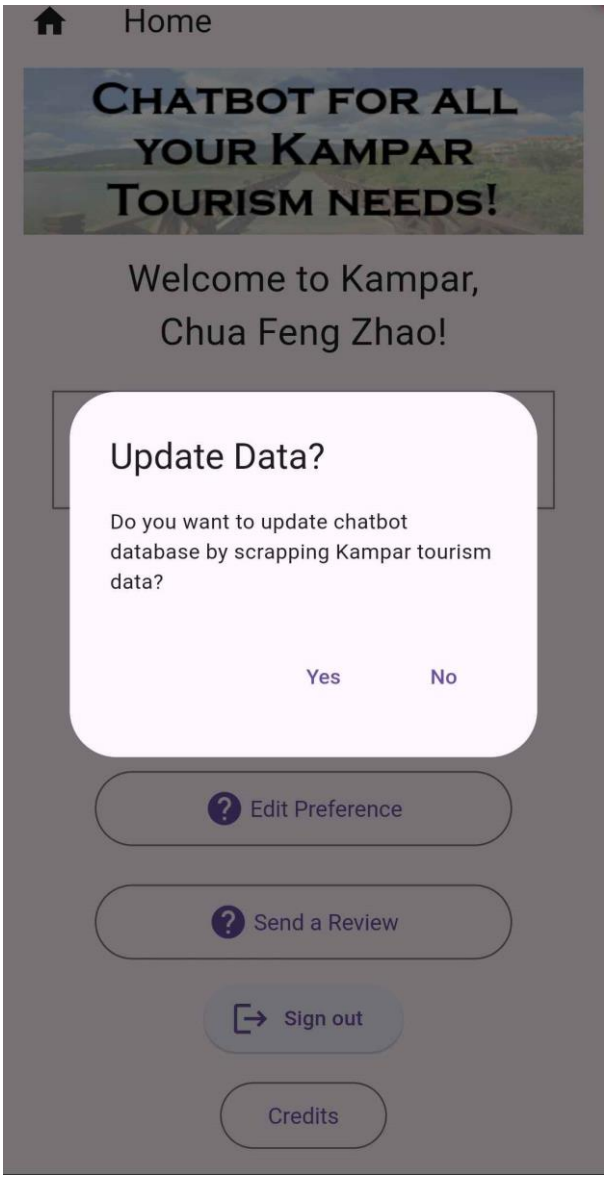


Figure 4.2.6: Admin Automatic Web Scraping

Users can enter into the application's settings, where users can delete their current chat history with the chatbot. By deleting the chat history, users will wipe all current context chatbot will have about the user. This makes the chatbot think the user is a new one and will have no previous knowledge of the user to base its suggestions from. In addition, users can also change what additional information they want to be sent whenever they message the chatbot such as the current weather, time and their coordinates. Users are recommended to enable these when

they are already in Kampar. Enabling these can allow users to enhance the personalization of the bot. Finally, the chatbot also allows users to toggle whether their travelling preferences will be updated after every message. Figure 4.2.7 shows the settings screen of the application.

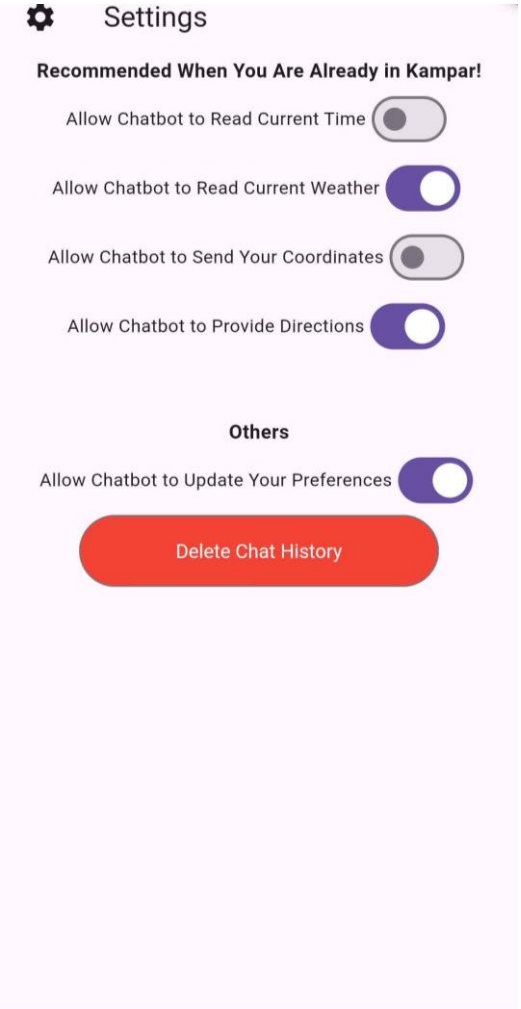


Figure 4.2.7: Setting Screen of Application

Users can enter the 'Edit Preferences' screen. The application will prompt the user to enter their traveling preferences. These include who the user is traveling with, their budget situation, the types of restaurants they prefer, the types of attractions they prefer and the types of hotels they prefer. Users can also choose not to enter in certain information too if they want to keep it hidden or if they have no preferences in that field in which case it will be stored as 'having no preference' or 'no information available'. These preferences will be stored in the Firestore Database, in which the chatbot will later grab to personalize the suggestions it will provide for the user. Figure 4.2.8 shows the 'Edit Preferences' screen.

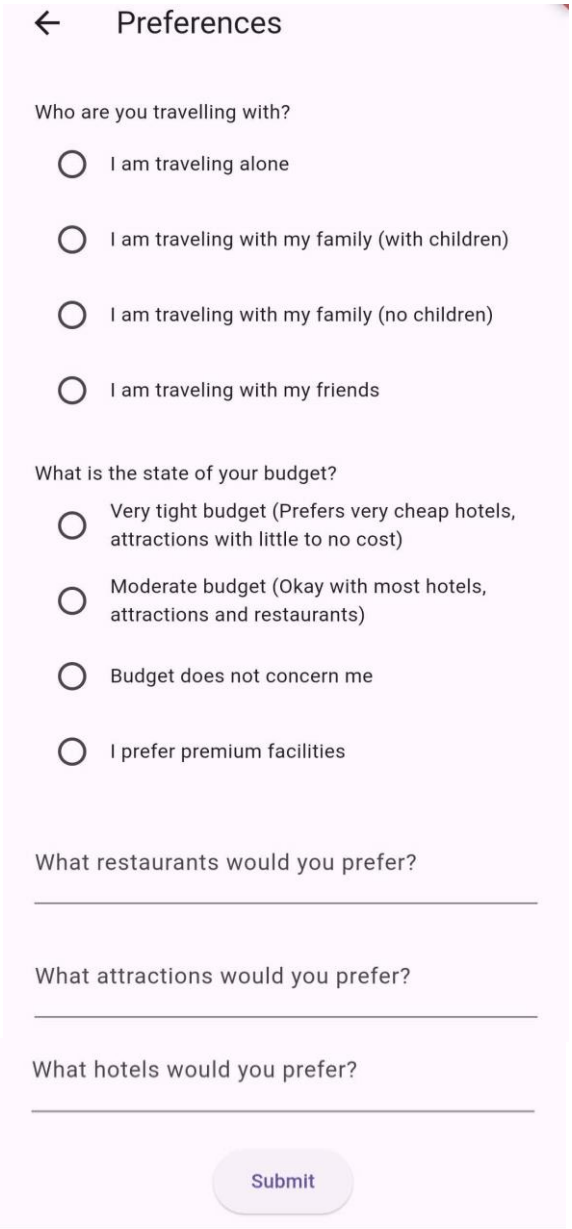


Figure 4.2.8: 'Edit Preferences' Screen

Users can also make reviews for the various places in Kampar. To do so, they will have to click on the 'Send a Review' button on the main menu. After that, the application will ask users what they want to review, hotels, restaurants or attractions. Depending on what they choose, the selection of places they can review will be different. Afterwards, users can provide a title for their review, the review itself and rating from 1 to 5. Once users click on submit, the review will be stored in Firestore Database to be grabbed by the admin whenever they initiate the automatic scraping and updating of Kampar tourism data. Figure 4.2.9 shows the Send a Review page of the chatbot application.

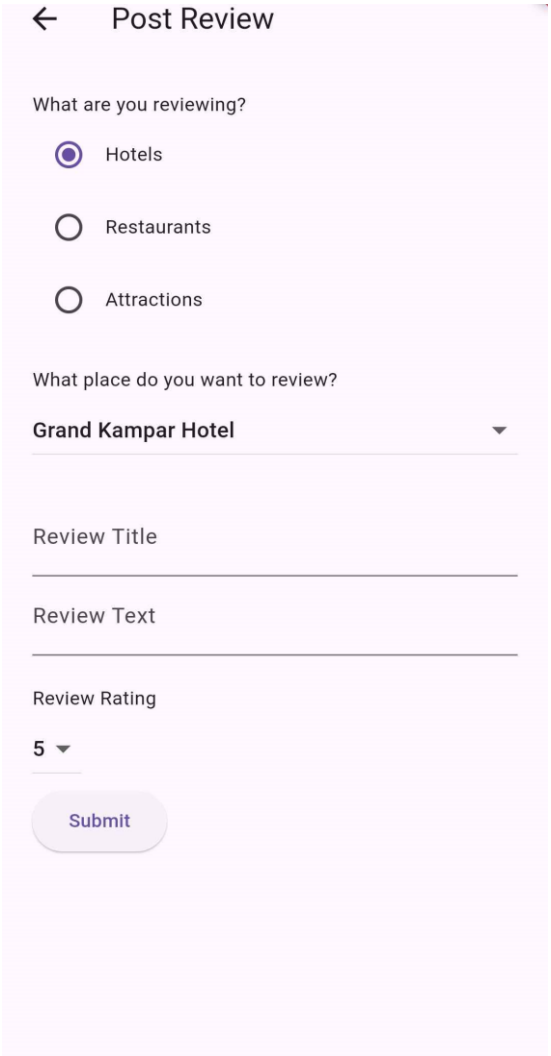


Figure 4.2.9: 'Send a Review' Screen

Users can enter the main chat window, where they will be able to send messages to the chatbot in order to obtain suggestions. The chatbot uses the OpenAI language model to understand the user's request. A language model allows a deeper understanding of user's sentences through a probability distribution over words that will most likely be the most appropriate [9]. Based in the user's input, the OpenAI language model will determine what the user wants from hotel-based information, restaurant-based information, or attractions-based information. Once the model has detected what type of tourist destination the user is looking for, it will pass that request to the OpenAI Assistant. The OpenAI Assistant, which is a relatively new service which allows the creation of AI-based assistants [18]. These assistants allow users to follow up from their previous conversation history with the chatbot and allows the feeding and reading of custom or private data. The assistant was trained to act like a tour guide and will provide

tourism suggestions for users based on their inquiry, traveling preferences and additional information they elected to send by toggling them in the setting page such as the current weather, time and the user's location. The chatbot can also direct the user to a location of their request by communicating with the Google Places API, in this scenario the chatbot will pass the destination request to the Google Places API and it will return a google maps showing the location from the user's location to the destination. Finally, to address the fact that user's travelling preferences could change or vary while travelling compared to their initial preferences due to change in taste or preference in culture specificity, if the user has enable the function to update their travelling preferences after they message, the user's chat history and their old preferences will be sent to another OpenAI Assistant built to read the user's chat history and old preferences as a reference to build the user's new travelling preferences. Figure 4.2.10 shows an empty chat screen of the application.

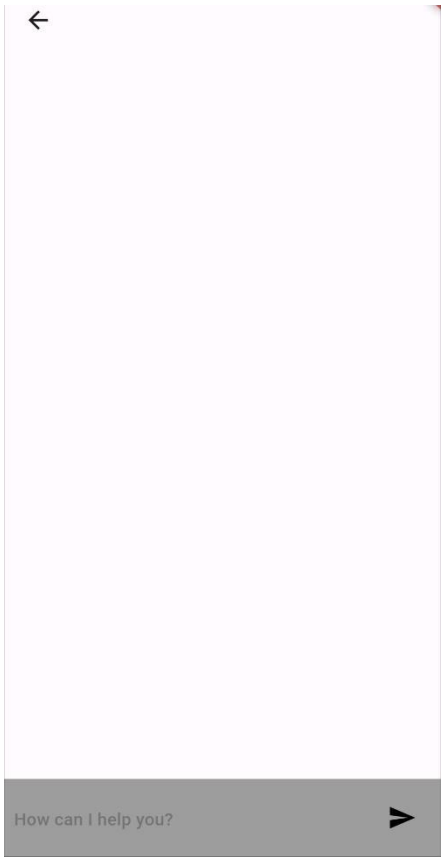


Figure 4.2.10: Empty Chat Screen

Chapter 5
System Implementation

5.1 Hardware Setup

This project uses two main hardware devices. A laptop is used in order to research for the project, grab and process the Kampar tourism data for the project and run the programs necessary for the development of the Kampar tourism mobile chatbot application. Table 5.1.1 shows the hardware specifications of the laptop used in the project.

Table 5.1.1: Laptop Specification

Description	Specifications
Model	Acer Nitro AN515-54
Processor	Intel Core i5-8300H @ 2.30GHz, 2304Mhz, 4 Cores
Operating System	Windows 10
Graphic	NVIDIA GeForce GTX 1050 3GB DDR5
Memory	8GB DDR4 RAM
Storage	1 512GB SATA SSD & 1 1TB SATA SSD

In addition, an android mobile phone is used to deploy, test and troubleshoot the Kampar tourism mobile chatbot application. Table 5.1.2 shows the hardware specifications of the android mobile phone used in the project.

Table 5.1.2: Android Phone Specification

Description	Specifications
Model	OPPO A92
Processor	Qualcomm Snapdragon 655 2.02GHz
Operating System	Android 11
Memory	8GB RAM
Storage	128GB

5.2 Software Setup

Multiple software and services were used to develop the Kampar Tourism Chatbot.

5.2.1 Software

5.2.1.1 Android Studio

The Kampar mobile app will be developed in Android Studio, which is the official integrated development environment (IDE) for the Android OS, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. Android Studio comes with a code editor where the code for the Kampar Tourism Mobile app can be developed in.

5.2.1.2 Visual Studio Code

Another integrated development environment (IDE) will be used during this project. This IDE will be used to run the Python code that will grab the information that is necessary for the project. It is also used to process the information already scrapped.

5.2.2 Programming Languages

5.2.2.1 Flutter and Dart

Both Flutter and Dart are programming languages in which the mobile application will be developed in. Flutter is an open-source framework, UI software development kit. It will be used to allow the Kampar mobile application to be cross platform from a single codebase. It is a plugin for Android Studio to implement the UI elements provided in the Flutter open-source framework for the front-end development of the Kampar mobile application which contains the chatbot. Dart is an object-oriented programming language that is used to mainly develop mobile and web applications.

5.2.2.2 Python

Python is the programming language that will be used for data collection in the form of web scraping. Python is selected for this purpose due to its access to useful web scraping API such as BeautifulSoup and Selenium.

5.2.3 Services

5.2.3.1 Firebase

Firebase will be used both to integrate the mobile application in the Firebase backend cloud computing services. Firebase offers an authentication service, which allows user accounts to be created, managed and accessed in a secure manner without needing to develop and manage the service from scratch, which saves time on development. Firebase also provides a NoSQL database, known as the Firestore Database, which allows the tourism data, user data and user preferences data to be stored to be used by the mobile application. It is a cloud-hosted NoSQL database that allows the storing and synchronizing data in real time, which is suitable for the dynamically changing user data and user preferences.

5.2.3.2 OpenAI API

OpenAI is an advanced AI language model that the project will be using. The chatbot will be using this API in order process user's messages and obtain the context within the message and return an appropriate response. For example, if a user would say they want to go to a café, the OpenAI language model will determine from the message that the user wants a suggestion to go to a café, and so the model will return a café suggestion. The OpenAI API is also used to extract the context and meaning of user reviews, the model will be used to determine whether the review is positive or negative based on the language used in the review.

5.2.3.3 BeautifulSoup and Selenium

Both BeautifulSoup and Selenium are Python libraries that are used for web scrapping. They are set up and used to scrape Kampar tourism data from TripAdvisor. BeautifulSoup can only

be used to extract elements from HTML code while Selenium allows a program to interact with the webpage as it searches for the elements in the website code.

5.2.3.4 OpenWeather API

OpenWeather API is used to grab the weather in Kampar at the time of the user's message to the chatbot in order to further improve the suggestions provided by the chatbot.

5.2.3.5 Google Places API

Google Places API is an API offered by Google which allows developers to access information provided by Google in regards to their travelling services. In this project, Google Places API is set-up for two purposes. It is first used to scrape more tourists information on places in Kampar beyond the reviews offered by TripAdvisor. In addition, it is used to offer directions to users who wishes to go to a specific place in Kampar.

5.3 Setting and Configuration

5.3.1 Data Acquisition

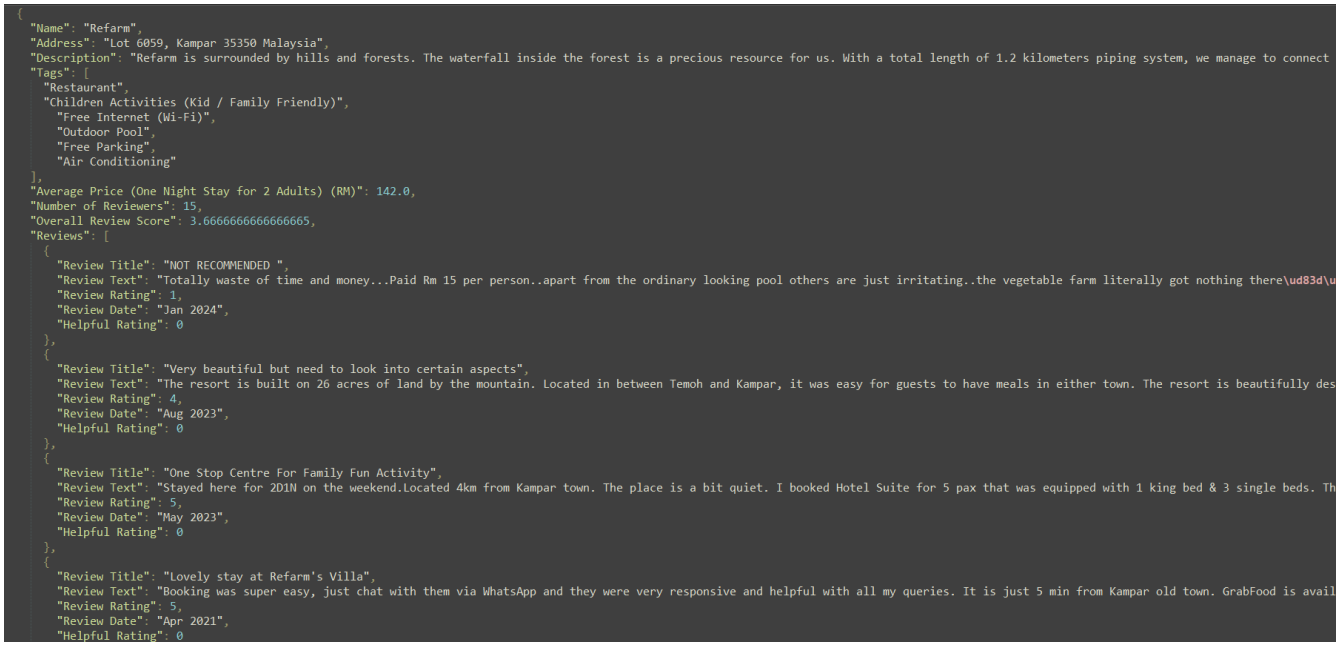
Kampar tourism data and the tourism data of the area surrounding Kampar, like Ipoh, is first scraped from TripAdvisor using both Python libraries, BeautifulSoup and Selenium with Selenium used for navigating the TripAdvisor website and the pages of reviews and BeautifulSoup used to scrap the data from the webpages.

Hotels, restaurants, and attractions data are scrapped and stored separately using their own web scraping code due to the different HTML structure of the hotel, restaurant, and attraction pages in TripAdvisor and also due to the slightly different information that is to be scrapped between the restaurants, hotels and attractions. The information that are scrapped during this process include the name, address, tags which describes the types of services the tourist destination offers, the descriptions provided by the hotel or attraction owners, the average price for a one-night stay for two adults for the hotels, the estimated duration of the attraction and information of the tourists' reviews of the locations.

Regarding the tourist review information, the number of reviewers for the location is scrapped. This shows the popularity of a location as the more people visit the location, the more reviews it will receive. The overall rating for the location is also scrapped. This indicated the quality of the location; a higher rating means a better quality. The overall rating ranges from 0 to 5 and it is calculated by summing up each reviewer's rating and dividing it by the total number of reviewers. For every review of the location, the review's title, text, rating which also ranges from 0 to 5, the date of the review and the helpful rating the review received. The review's date shows the relevancy of the review, the newer the review, the higher likely it is up-to-date and relevant. The helpful rating represents the reliability of the review.

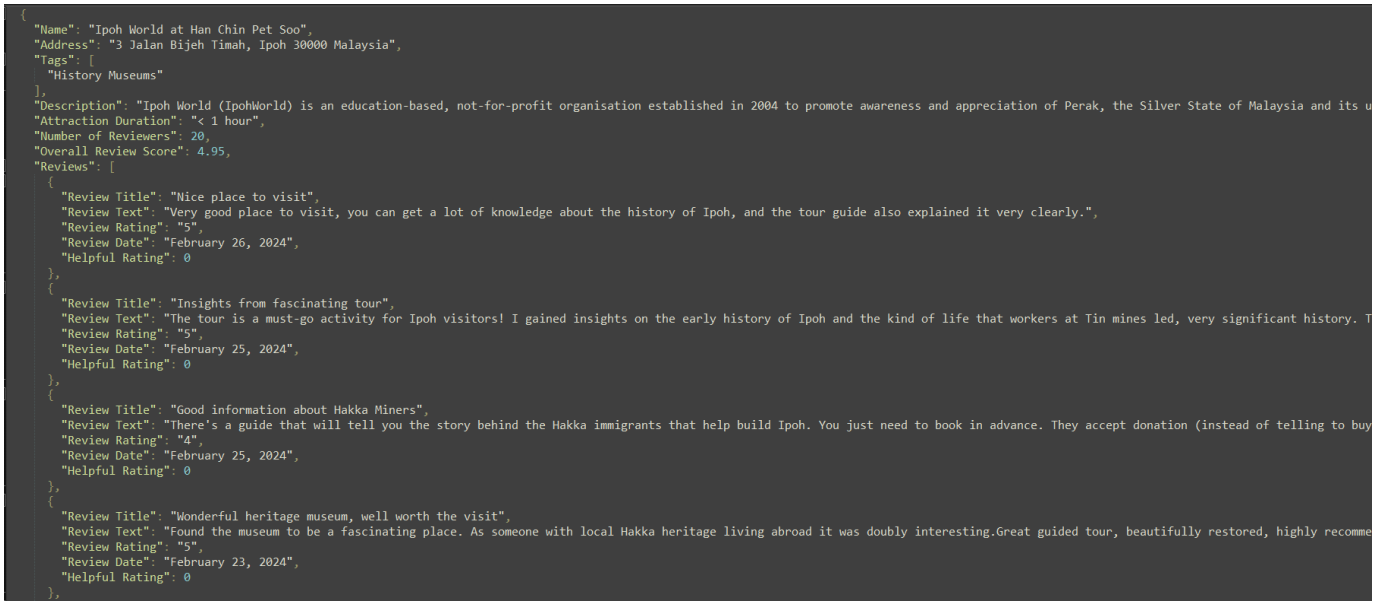
The scrapped data is then stored into JSON files. Three JSON files are created, one for restaurants, one for hotels and one for attractions. Figures 5.3.1.1, 5.3.1.2 and 5.3.1.3 shows how the scrapped data is stored in the JSON file for restaurants, hotels and attractions respectively.

Figure 5.3.1.1: JSON Structure of Kampar's Restaurant Information



```
{
  "Name": "Refarm",
  "Address": "Lot 6059, Kampar 35350 Malaysia",
  "Description": "Refarm is surrounded by hills and forests. The waterfall inside the forest is a precious resource for us. With a total length of 1.2 kilometers piping system, we manage to connect
  "Tags": [
    "Restaurant",
    "Children Activities (Kid / Family Friendly)",
    "Free Internet (Wi-Fi)",
    "Outdoor Pool",
    "Free Parking",
    "Air Conditioning"
  ],
  "Average Price (One Night Stay for 2 Adults) (RM)": 142.0,
  "Number of Reviewers": 15,
  "Overall Review Score": 3.6666666666666665,
  "Reviews": [
    {
      "Review Title": "NOT RECOMMENDED ",
      "Review Text": "Totally waste of time and money...Paid Rm 15 per person..apart from the ordinary looking pool others are just irritating..the vegetable farm literally got nothing there\ud83d\u
      "Review Rating": 1,
      "Review Date": "Jan 2024",
      "Helpful Rating": 0
    },
    {
      "Review Title": "Very beautiful but need to look into certain aspects",
      "Review Text": "The resort is built on 26 acres of land by the mountain. Located in between Temoh and Kampar, it was easy for guests to have meals in either town. The resort is beautifully des
      "Review Rating": 4,
      "Review Date": "Aug 2023",
      "Helpful Rating": 0
    },
    {
      "Review Title": "One Stop Centre For Family Fun Activity",
      "Review Text": "Stayed here for 2D1N on the weekend.Located 4km from Kampar town. The place is a bit quiet. I booked Hotel Suite for 5 pax that was equipped with 1 king bed & 3 single beds. Th
      "Review Rating": 5,
      "Review Date": "May 2023",
      "Helpful Rating": 0
    },
    {
      "Review Title": "Lovely stay at Refarm's Villa",
      "Review Text": "Booking was super easy, just chat with them via WhatsApp and they were very responsive and helpful with all my queries. It is just 5 min from Kampar old town. GrabFood is avail
      "Review Rating": 5,
      "Review Date": "Apr 2021",
      "Helpful Rating": 0
```


Figure 5.3.1.2: JSON Structure of Kampar's Hotel Information



```
{
  "Name": "Ipoh World at Han Chin Pet Soo",
  "Address": "3 Jalan Bijeh Timah, Ipoh 30000 Malaysia",
  "Tags": [
    "History Museums"
  ],
  "Description": "Ipoh World (IpohWorld) is an education-based, not-for-profit organisation established in 2004 to promote awareness and appreciation of Perak, the Silver State of Malaysia and its u
  "Attraction Duration": "< 1 hour",
  "Number of Reviewers": 20,
  "Overall Review Score": 4.95,
  "Reviews": [
    {
      "Review Title": "Nice place to visit",
      "Review Text": "Very good place to visit, you can get a lot of knowledge about the history of Ipoh, and the tour guide also explained it very clearly.",
      "Review Rating": "5",
      "Review Date": "February 26, 2024",
      "Helpful Rating": 0
    },
    {
      "Review Title": "Insights from fascinating tour",
      "Review Text": "The tour is a must-go activity for Ipoh visitors! I gained insights on the early history of Ipoh and the kind of life that workers at Tin mines led, very significant history. T
      "Review Rating": "5",
      "Review Date": "February 25, 2024",
      "Helpful Rating": 0
    },
    {
      "Review Title": "Good information about Hakka Miners",
      "Review Text": "There's a guide that will tell you the story behind the Hakka immigrants that help build Ipoh. You just need to book in advance. They accept donation (instead of telling to buy
      "Review Rating": "4",
      "Review Date": "February 25, 2024",
      "Helpful Rating": 0
    },
    {
      "Review Title": "Wonderful heritage museum, well worth the visit",
      "Review Text": "Found the museum to be a fascinating place. As someone with local Hakka heritage living abroad it was doubly interesting.Great guided tour, beautifully restored, highly recomme
      "Review Rating": "5",
      "Review Date": "February 23, 2024",
      "Helpful Rating": 0
    },
```


Figure 5.3.1.3: JSON Structure of Kampar's Attraction Information

5.3.2 Data Processing

While some information in TripAdvisor is provided by the restaurant, hotel or attraction owners, most information in TripAdvisor is provided by travellers who already have explored the location. Thus, TripAdvisor sometimes may not have sufficient information for a certain location. For example, the webpage may not display the attraction duration for a certain

attraction due to not having enough information provided by tourists. In this scenario, the missing data must be processed. Figure 4.2.2.1 shows the attraction duration in the attraction 'Kinta Tin Mining Museum' be labelled as 'N/A' as it is not displayed in TripAdvisor's website.

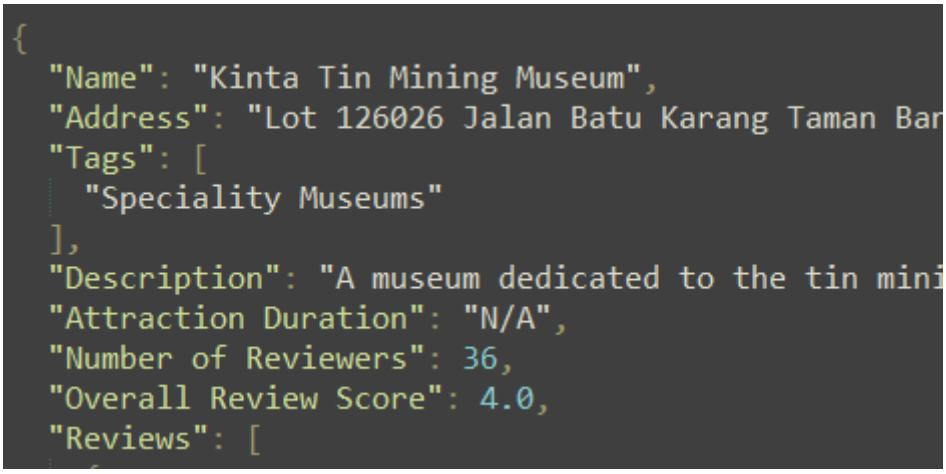


Figure 5.3.2.1: Processing Method for Lack of Attraction Duration

More egregiously, some hotels, restaurant or attractions will not have any user review information. To preprocess this, review information is attempted to be manually scrapped from other websites, such as Agoda.com. If no reviews can be manually scrapped due to its scarcity, the hotel, restaurant or attraction is dropped from the dataset. Figure 5.3.2.2 shows the hotel 'Khazanah Hijau Homestay' having no review information scrapped due to the lack of information present in TripAdvisor.

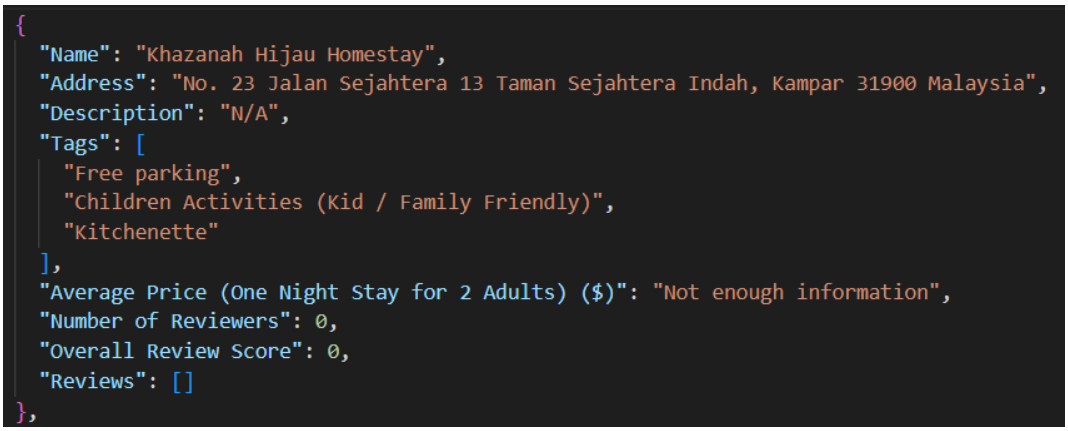


Figure 5.3.2.2: Lack of Review Data

Once the data has been manually reviewed and resolved, the data is passed onto a Python program that communicates with Google Places API. This is so that the Google reviews of the restaurants, attractions and hotels in Kampar can be easily retrieved. To do so, each name of the restaurant, attraction or hotel are read, then it is sent through Google Places API to retrieve the review. Due to the slight differences in review structure that exists in Google and TripAdvisor, slight changes needed to be made to the reviews before being stored. For example, there are no review titles for the Google reviews as Google does not allow users to have a title for their reviews, thus they are processed to list 'N/A'. In addition, the program will also grab the coordinate location of the area as well in addition to the reviews. This is done because the chatbot cannot understand addresses as OpenAI's Language Model does not have context over geographical locations through an address. Thus, the location needs to be in a form which is both universal and understandable to a chatbot. Coordinates is a good choice as chatbots are able to perform calculations with coordinates to determine distance, thus the chatbots do not need to understand geographical locations. Figures 5.3.2.3 and 5.3.2.4 shows the JSON data before and after it has been processed.

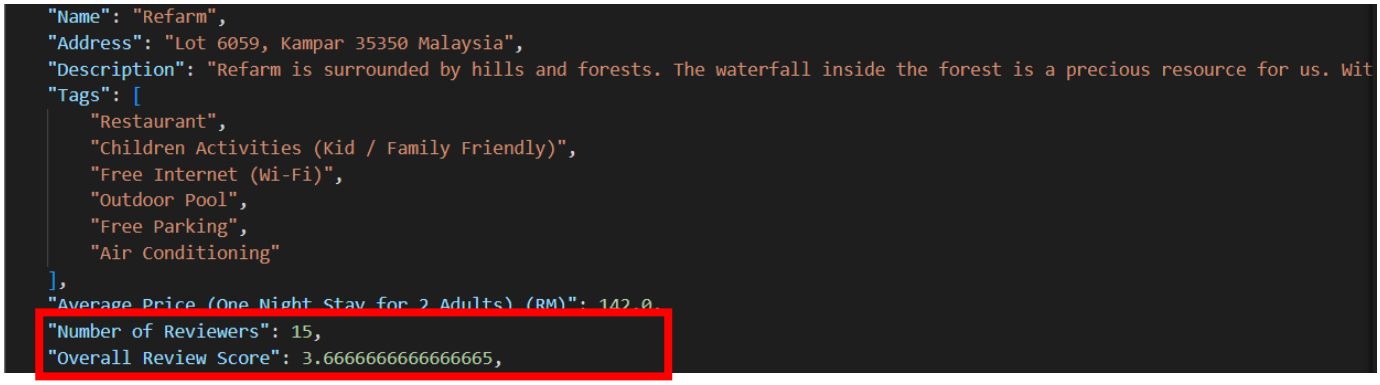


Figure 5.3.2.3: JSON Data Before Processing

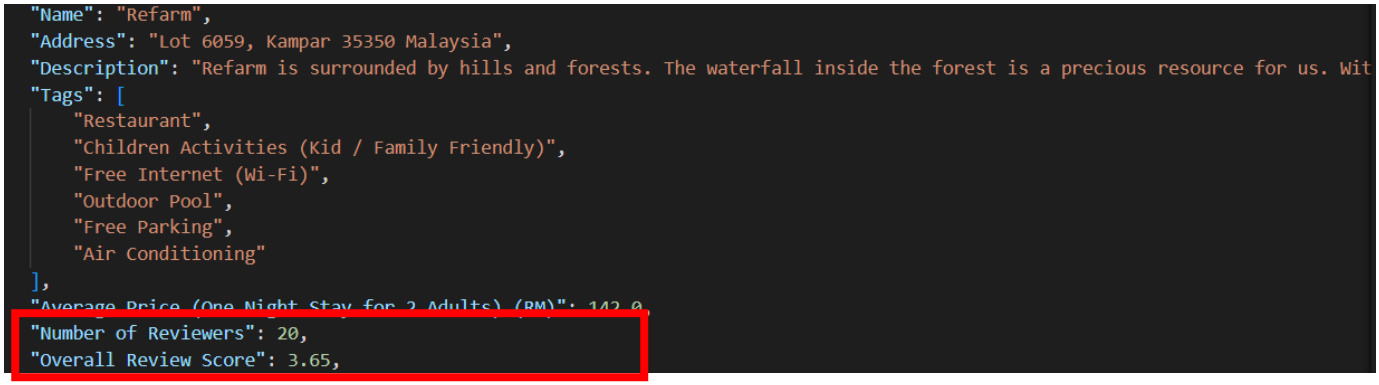


Figure 5.3.2.4: JSON Data After Processing

5.3.3 Firebase Settings

As mentioned, Firebase is used to manage user's account and act as storage for both user's chat history and their travelling preferences. Using Firebase Authentication, users can sign up to an account using an email and password. Upon sign up, users will be provided with a User ID that is hidden to the user. Figure 5.3.3.1 shows the accounts registered to the application as seen in Firebase Console.

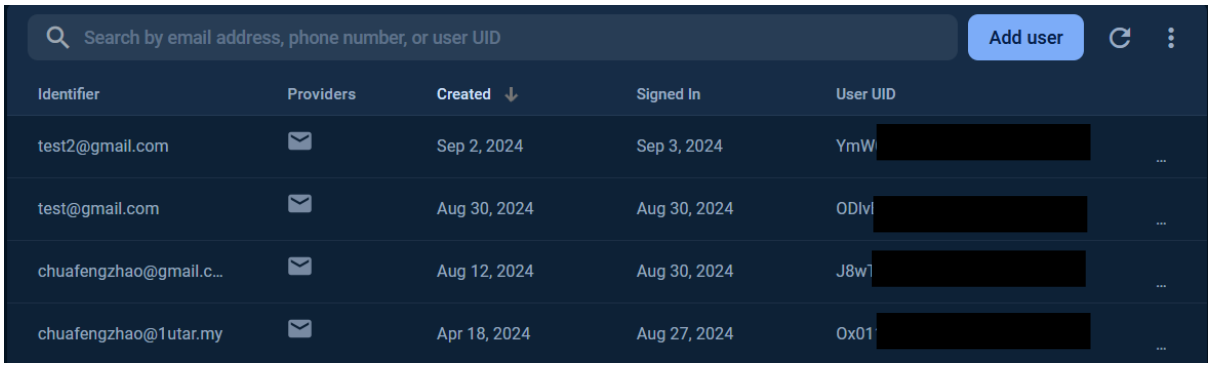


Figure 5.3.3.1: Accounts Stored in Firebase Authentication

Using Firestore Database, user's chat history, travelling preferences and the various user's reviews of the places in Kamparare stored. Each user's chat history and travelling preferences are stored based on their unique User ID, thus, no users will ever have conflicts with the storage of their data. Figure 5.3.3.2 shows how data is stored in Firebase, where the User ID stores the user's chat history and 'thread&profiling' stores the user's travelling preferences. In addition, user reviews are all stored in 'reviews'

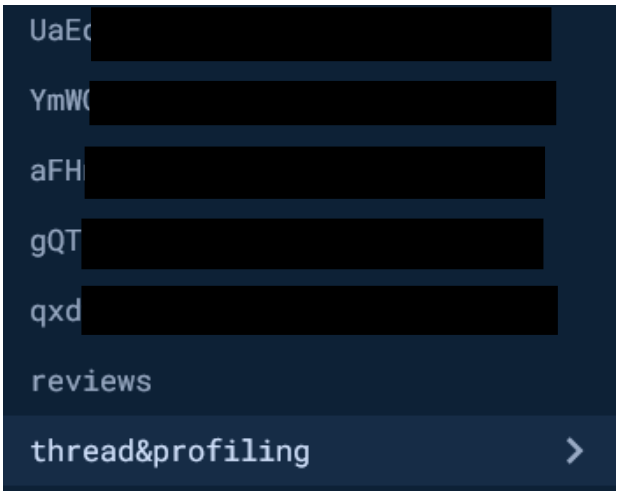


Figure 5.3.3.2: Data Stored in Firestore Database

5.3.4 Instructing OpenAI Assistant and Feeding Data

Once the tourism data has been obtained and processed, the OpenAI assistant is then built using the using the assistant builder provided by OpenAI. First, the three JSON files containing Kampar's restaurant, hotel, and attractions data are first fed into the assistant, in which it will embed them and store them for knowledge retrieval in order to improve comprehension of Kampar tourism data of the GPT-4o mini language model used. Figure 5.3.4.1 shows the files vectorized in the OpenAI Assistant console.

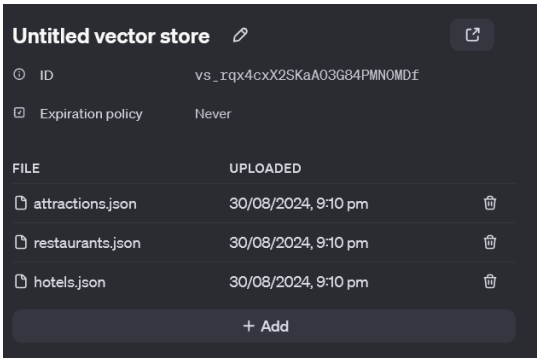


Figure 5.3.4.1: Kampar Tourism Data Stored as Vectors in Assistant Database

During the creation of the OpenAI Assistant, it is required to provide it and train it with instructions on what it should and should not do when making suggestions to users. It is specified to the chatbot that it is a tour guide-like assistant that will provide hotel, restaurant and attraction suggestions to users based on their inquiries. To make the chatbot provide as accurate and ideal output as possible, some rules were defined into the assistant's protocol.

First, it is specified that, should the user ask a question unrelated to tourism, the chatbot should kindly and politely tell the user that it cannot answer that question as it will be outside the chatbot's expertise and intended role in the application. This is done to prevent unintended use cases for the chatbot application.

Then, some rules regarding providing suggestions were specified. The chatbot was made to provide the name and general location of the provided hotel, attraction, or restaurant. The chatbot was also told to provide 3 suggestions when providing them. This is because users may not want to choose the first provided suggestion because they might not like a specific aspect of the suggestion that was not accounted for by the chatbot.

The chatbot is also instructed to only provide a short explanation for its provided suggestions. The justification for this is that users will not want to read a lot when all they need are suggestions for where they should go for their vacation. Thus, providing a short explanation will lessen the hassle provided to the user. Users can request for further explanation in order to receive a long explanation from the chatbot should they want it.

The chatbot is also tasked to not tell users to search for more information about certain locations online, if possible. This is because the chatbot asking users to do this will defeat the purpose of the chatbot itself.

The chatbot is also instructed to always take certain user disabilities to account, such as with allergies. Furthermore, the chatbot is tasked with taking into account user's travelling preferences that will be read from the Firestore Database. This is to provide the personalization component to the chatbot where different users' tastes can be recognized and taken into account when making the suggestions. Travelling preferences are separated into 5 categories, these include the people that the user is travelling with, the user's budget constraints and what hotel, restaurants and attractions would the user prefer when visiting.

In addition, the chatbot is also instructed to take into account the current weather conditions, current time and the user's current coordinates should the users decide to pass that information as additional information when providing suggestions for users. This is done because different attractions or restaurants are suited for different weather and time conditions. For example, going hiking is not suggested on a rainy day but an indoor restaurant can be a place users can go on the rainy day to relax. The chatbot is also tasked to retrieve the coordinates of the locations and reference them with the user's coordinates should they be provided in order to try to provide locations that are nearer to the user. Furthermore, the chatbot is also tasked to, if the user asks for attractions suggestions during nighttime, the chatbot will suggest attractions for the following morning as most attractions will be closed at night. Figure 5.2.4.2 shows the assistant as seen in the OpenAI console.

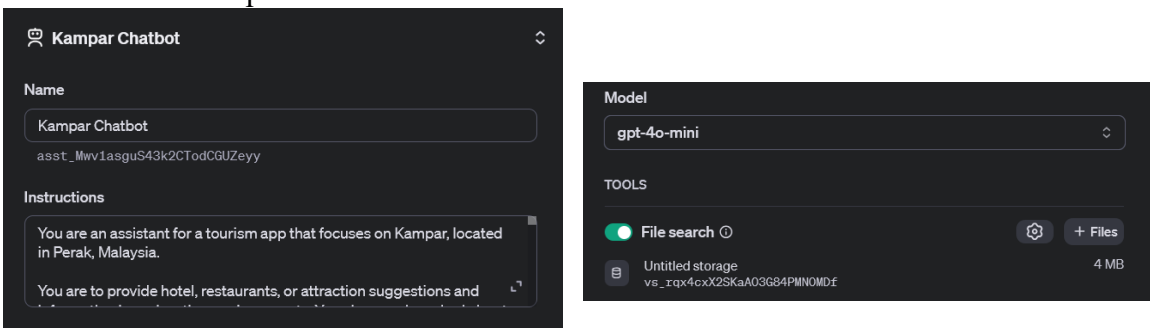


Figure 5.3.4.2: Assistant as Seen in OpenAI's Console

Moreover, the chatbot has a parameter called temperature. Temperature ranges from 0 to 2. The higher the value, the more random the term selection of the language model [19]. The temperature of the OpenAI assistant is set to 1.15 as it is important to give users a variable of randomness into the response as multiple locations could match the user's query and the randomness can help with variable responses for the same question, but at the same time, it is also important to not make the responses too random as the suggestions still need to line up with the user's query.

Finally, the chatbot is ran in a simulation to confirm that it is able to retrieve data from the files embedded into its vector database. Figure 5.3.4.3 shows the simulator retrieving data in response to user's query.

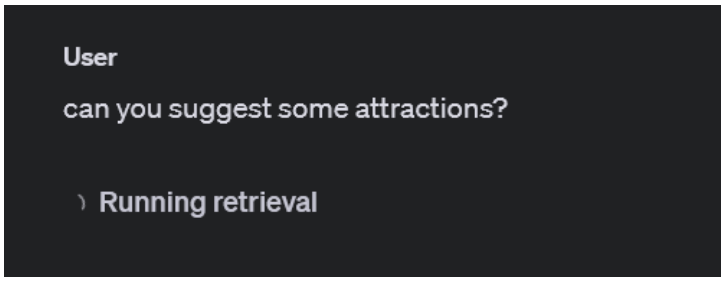


Figure 5.3.4.3 Chatbot Running Retrieval

5.3.5 Instructing OpenAI Assistant with Additional Functions

In addition to the main function of the OpenAI Assistant, two more additional functions are implemented into the assistant. Firstly, the main assistant utilizes a feature of OpenAI assistant which allows the language model to invoke functions should the language model recognize that the user's query is appropriate for the invocation of the function. For this project, the main assistant to instructed to invoke a function called get_direction, which connects to Google Places API to find a direction to a destination whenever the user request for directions. Using RAG, the chatbot will be able to extract the destination location from the user's query. The destination location is then passed into the function, where it is used as a string variable to send to Google Places through its API to search for its location. The main assistant is given some terms it should look for to invoke the function such as 'can you bring me to…'. Figure shows the settings for the conditions to invoke the function in the assistant's console.

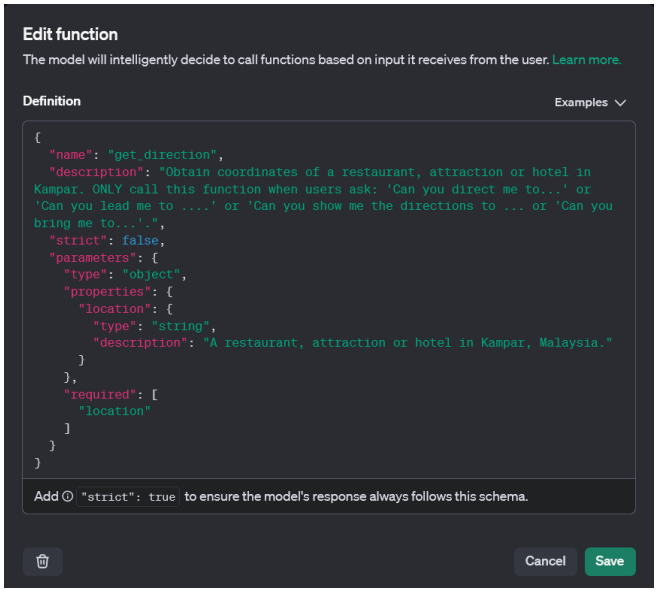


Figure 5.3.5.1: Settings of OpenAI Assistant for Invoking get_direction Function

Furthermore, a smaller OpenAI assistant is made in order to update the user's travelling preferences should they want the chatbot to help them proactively update their preferences which can help the main chatbot to suggest them places most suitable to them. This assistant is instructed to update the user's preferences based on the user's chat history and their old travelling preferences. Just like the main assistant, this assistant is given a few protocols to follow. Firstly, the assistant must return the exact same format as the old travelling preferences. It is instructed to not elaborate on the changes made as users will not be able to see their updated preferences anyways. Figure 5.3.5.2 shows the assistant as seen in OpenAI's console.

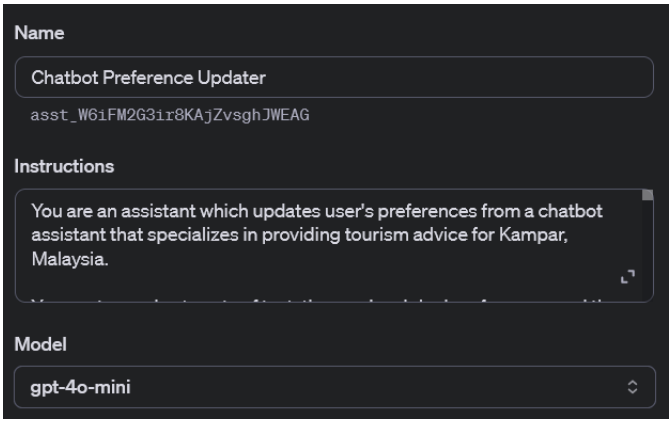


Figure 5.3.5.2 Small Assistant as Seen in OpenAI's Console

The assistant is instructed to reference the old preferences and the chat history with a ratio of 4:1. The assistant is also instructed to include what users may be considering if a user decides to ask questions of a specific area, this is because the user may be considering the area if they bothered to ask about it.

Moreover, the assistant is tasked to not include many preferences into the user's preferences and to remove some old preferences should there be too many. Finally, the assistant is tasked to not include any redundant preferences.

In addition, the chatbot is also made to not change the user's preferences if the latest question the user ask was for the chatbot to provide suggestions. This is because suggestions do not indicate that the user has have a change of preference in any way as the chatbot could still provide places that do not agree with the user. The assistant is made to look for terms such as 'suggest' and 'recommend' when determining this protocol.

5.3.6 Testing and Validation

During the creation and development of the OpenAI assistant, the assistant's algorithm has been tested constantly to reduce any unwanted output from the chatbot. During this process, a problem that was found is that the chatbot, when using the more advanced GPT-4 language model, preferred to provide longer explanations and answers, which is not optimal for users. Thus, the initial GPT-4 model is switched to the much lighter GPT-4o mini model, which tends to prefer shorter outputs.

Any problems with the output are resolve by redefining the instructions for the OpenAI assistant and retesting the output.

5.3.7 Developing Chatbot Application UI

A basic user interface is then developed in order to allow users to communicate with the OpenAI assistant to obtain travelling suggestions through a user-friendly mobile application. The user interface also allows users to personalize their chatbot experience by saving their travelling preferences for the chatbot to take reference to whenever it needs to provide travelling suggestions.

The user interface is also developed to allow users to toggle the additional information they might be willing to send, such as weather, time and their current location for more personalized travelling suggestions.

The user will also be able to send a review through the application's UI, it is also developed so that the process will be as seamless as possible for the user.

Finally, the user interface is also able to allow users to delete their chat history should they so choose, and toggle additional settings not related to the additional information, such as enabling the chatbot to be able to provide directions to the user.

5.4 System Operations

In this section, the process of usage of the application for a typical user will be shown, alongside screenshots of the operation itself.

5.4.1 Failing to Login into the Application

Figure 5.4.1.1 shows the error screen if the user inputs the incorrect email and password combination.

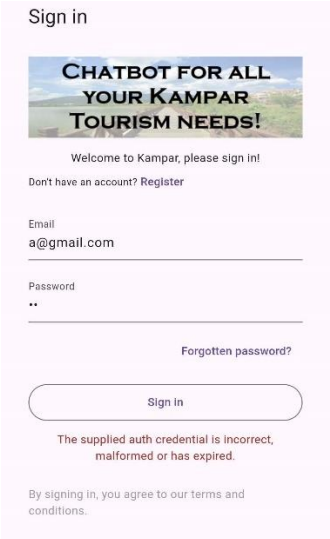


Figure 5.4.1.1: Error During Log In

5.4.2 Successfully Logging into the Application and Home Screen

Figures 5.4.2.1 and 5.4.2.2 show a successful login and the home screen.

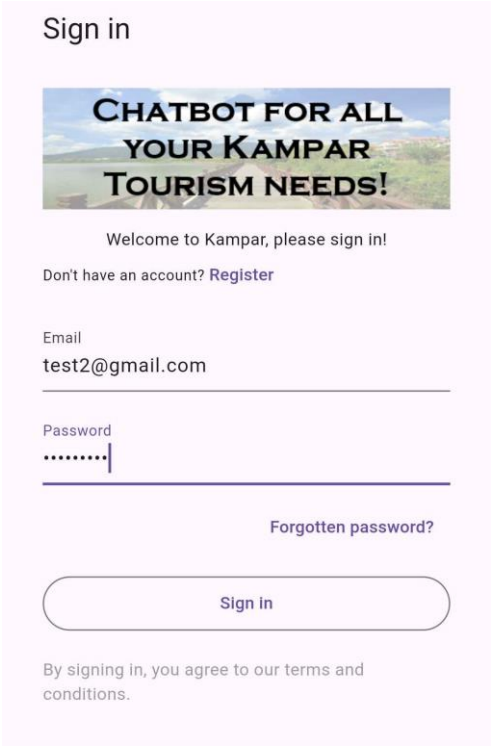


Figure 5.4.2.1: Logging In with Correct Credentials

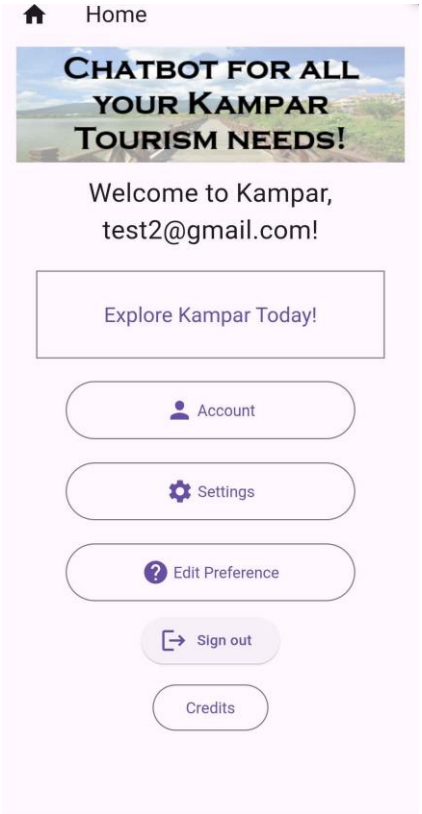


Figure 5.4.2.2: Reaching Main Menu

5.4.3 Editing Travelling Preferences

Figures 5.4.3.1, 5.4.3.2 and 5.4.3.3 show how users can edit their travelling preferences, how they are stored in Firestore Database and the difference between preferences that are filled and preferences that are not.

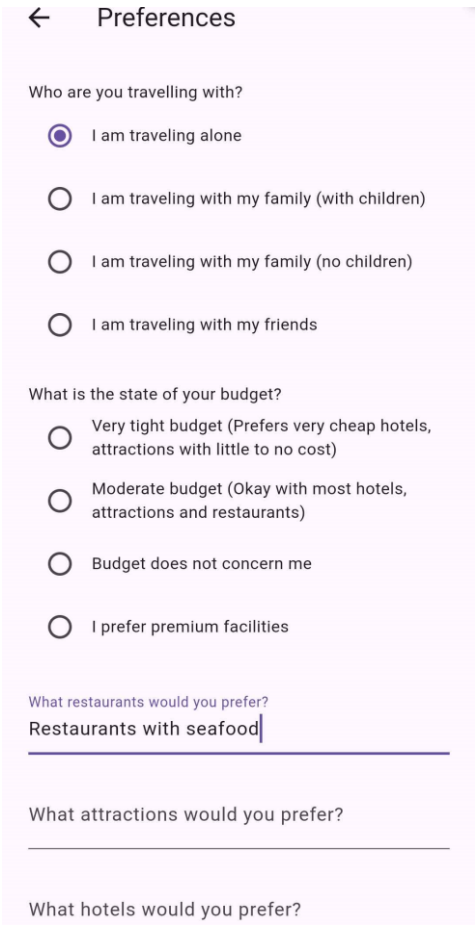


Figure 5.4.3.1: Changing Traveling Preferences

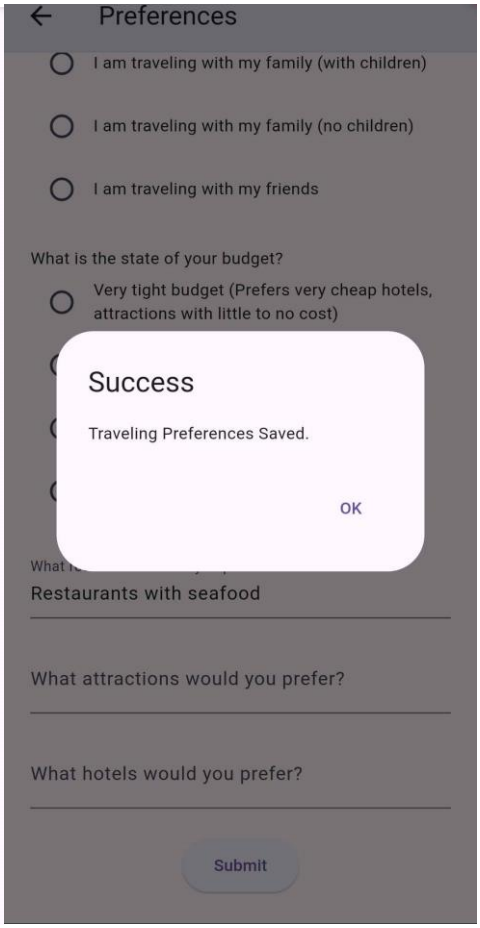


Figure 5.4.3.2: Saving Travelling Preferences

Figure 5.4.3.3: Traveling Preferences as Saved in Firestore Database

5.4.4 Changing Some Settings

Figure 5.4.4.1 shows how settings can be changed in the application. Settings are stored locally in the user's system.

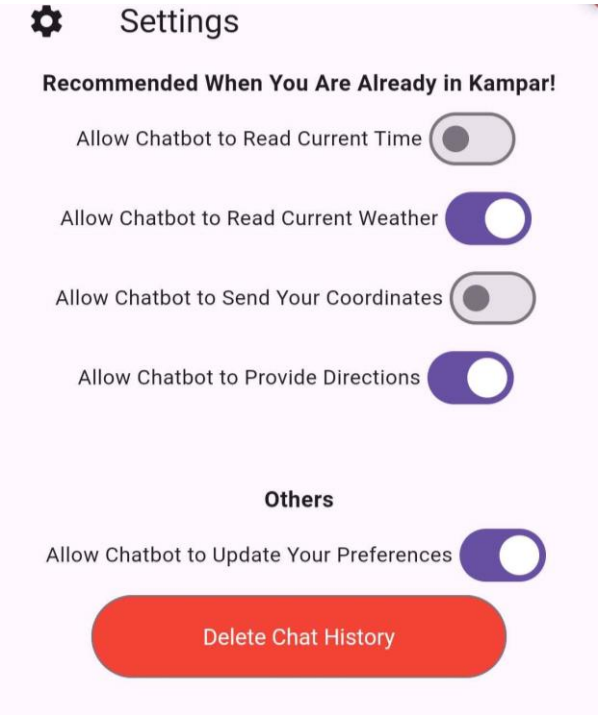


Figure 5.4.4.1: Changing Settings

5.4.5 Sending a Review of a Place in Kampar

Figure 5.4.5.1 shows how a review is sent through the application. Figure 5.4.5.2 shows how the review is stored in Firestore Database.

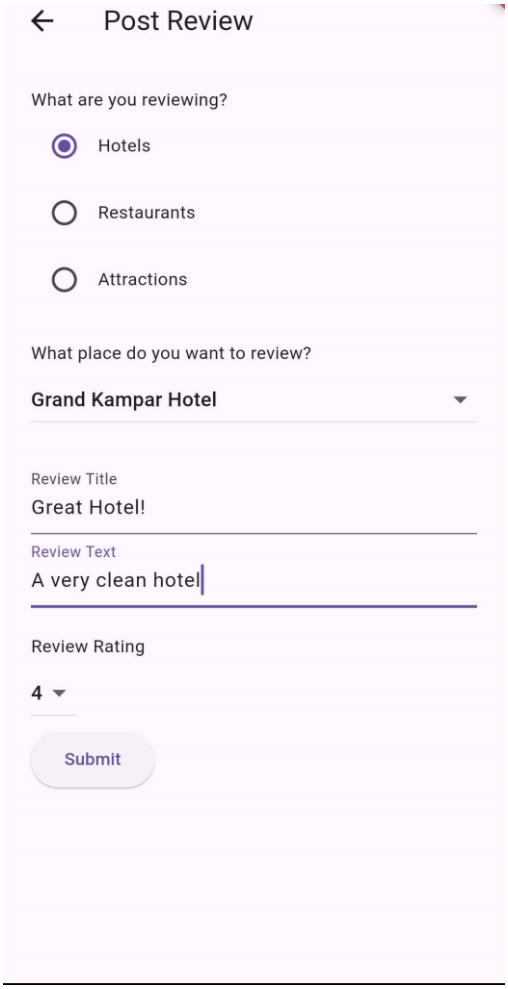


Figure 5.4.5.1: Sending a Review

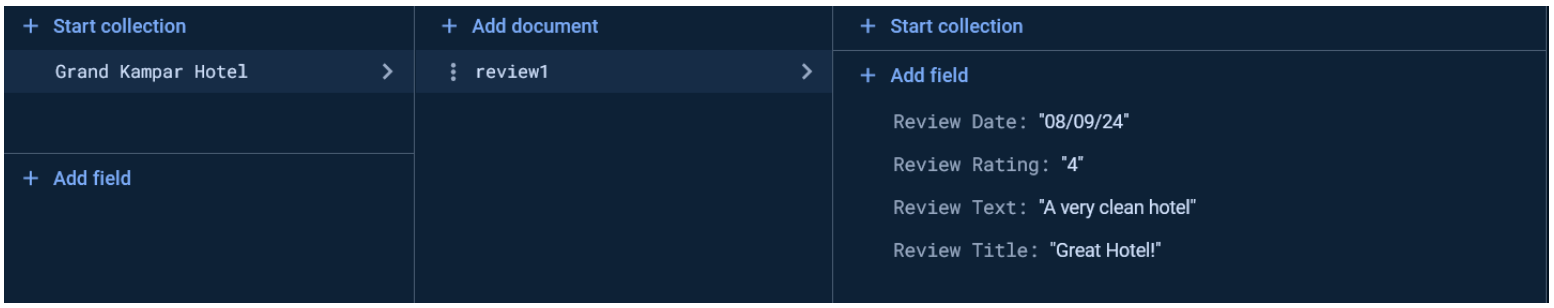


Figure 5.4.5.2: Review Being Stored in Firestore Database

5.4.6 Sending a Message to Chatbot and Chatbot Reply

Figures 5.4.6.1, 5.4.6.2, 5.4.6.3 and 5.4.6.4 show a message being sent and the chatbot's reply, as well as how OpenAI receives the message through the Android Studio console and how it is stored in Firestore Database.

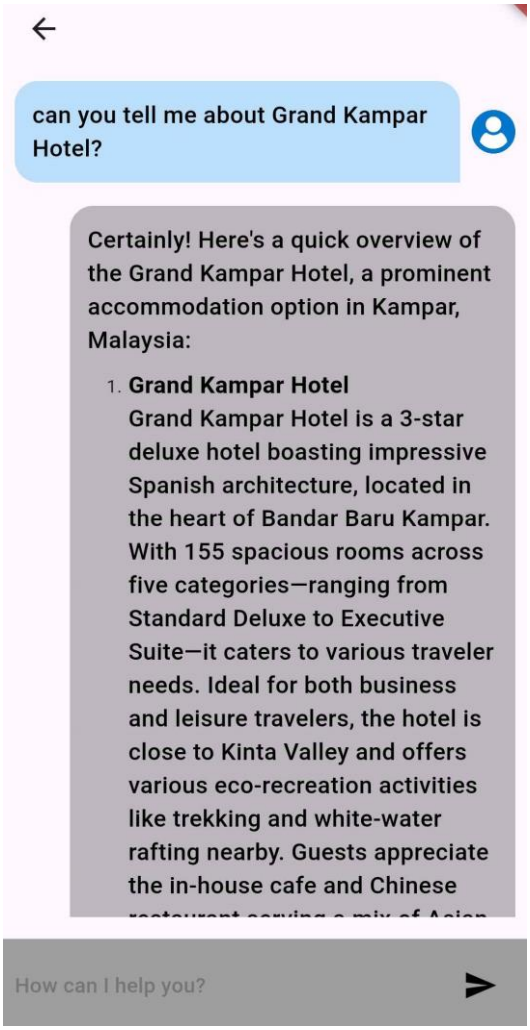


Figure 5.4.6.1: Sending a Message to Chatbot

```
[OpenAI] starting request https://api.openai.com/v1/threads/thread_668NQJAvDzI1CbQthyQVeR7O/messages
[OpenAI] request body :{role: user, content: Can you tell me about Grand Kampar Hotel?}
[OpenAI] ============= success ==================
[OpenAI] starting request https://api.openai.com/v1/threads/thread_668NQJAvDzI1CbQthyQVeR7O/runs
[OpenAI] request body :{assistant_id: asst_Mwv1asguS43k2CTodCGUZeyy, additional_instructions: Traveling Group: The user is traveling alone
Budget Constraint: The user elects to not share their budget preferences
Restaurant Preference: Restaurants with seafood
Attraction Preference: This user did not specify their attraction preferences
Hotel Preference: This user did not specify their hotel preferences
```

Figure 5.4.6.2: Console Output of Message Being Sent to Chatbot

Figure 5.4.6.3: Firestore Database Storing Chat History (1)

Figure 5.4.6.4: Firestore Database Storing Chat History (2)

5.4.7 Chatbot Providing Directions

Figures 5.4.7.1 and 5.4.7.2 shows how the chatbot invokes the Google Places API to send directions to the user. Once users click on the message, they will be shown a Google Maps with directions to their desired location.

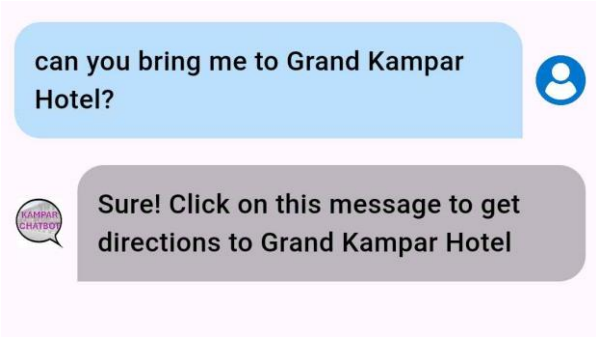


Figure 5.4.7.1: Requesting for Directions

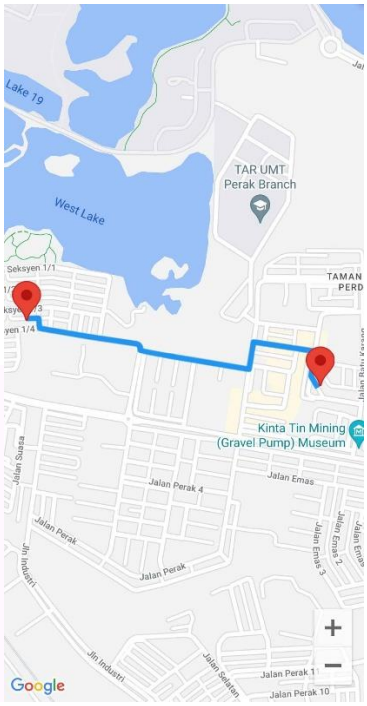


Figure 5.4.7.2: Google Maps Showing Direction to Destination

5.4.8 Chatbot Updating User Preference

Figure 5.4.8.1 shows how the user's travelling preferences have changed after the messages in Figures 5.4.6.1 and 5.4.7.1.

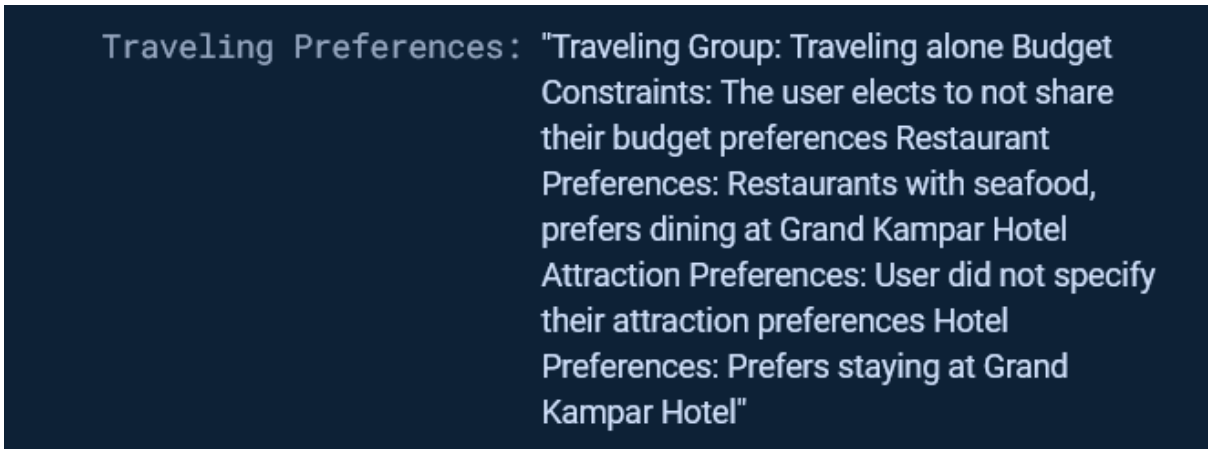


Figure 5.4.8.1: Updated Preference after Chat Messages from Figures 5.4.6.1 and 5.4.7.1

5.4.9 Deleting Chat History

Figures 5.4.9.1 and 5.4.9.2 shows the process of deleting the chat history and the result of doing so being an empty chat screen.

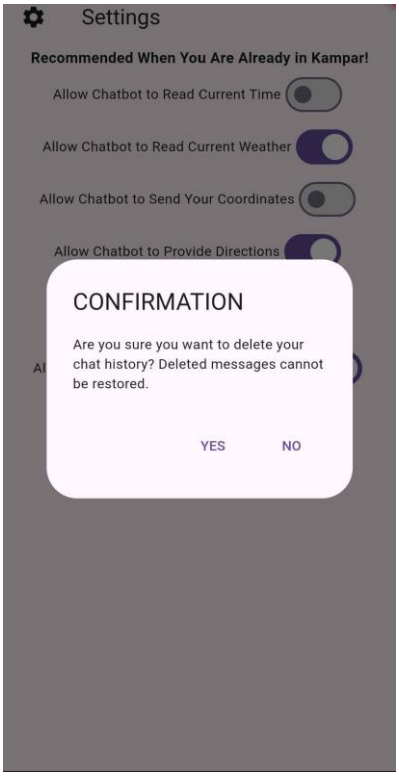


Figure 5.4.9.1: Confirming Deleting Chat History

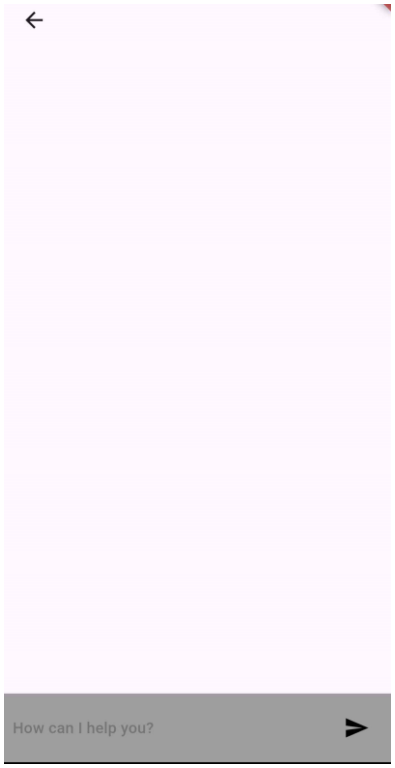


Figure 5.4.9.2: Empty Chat Screen After Deletion

5.4.10 Automatic Scraping of Kampar Tourism Data

Figure 5.4.10.1 shows how an admin is prompted to re-scrape Kampar tourism data in order to keep the chatbot knowledge base up to date. The implementation of this function is not fully complete, as such, this will only take the reviews sent by the users through this chatbot application from the Firestore Database and will only store the data locally and not sent to the knowledge base of the chatbot. Figure 5.4.10.2 shows the tourism data to be updated with the reviews made by the users of the application. The review stored can be seen in Figure 5.4.5.1.

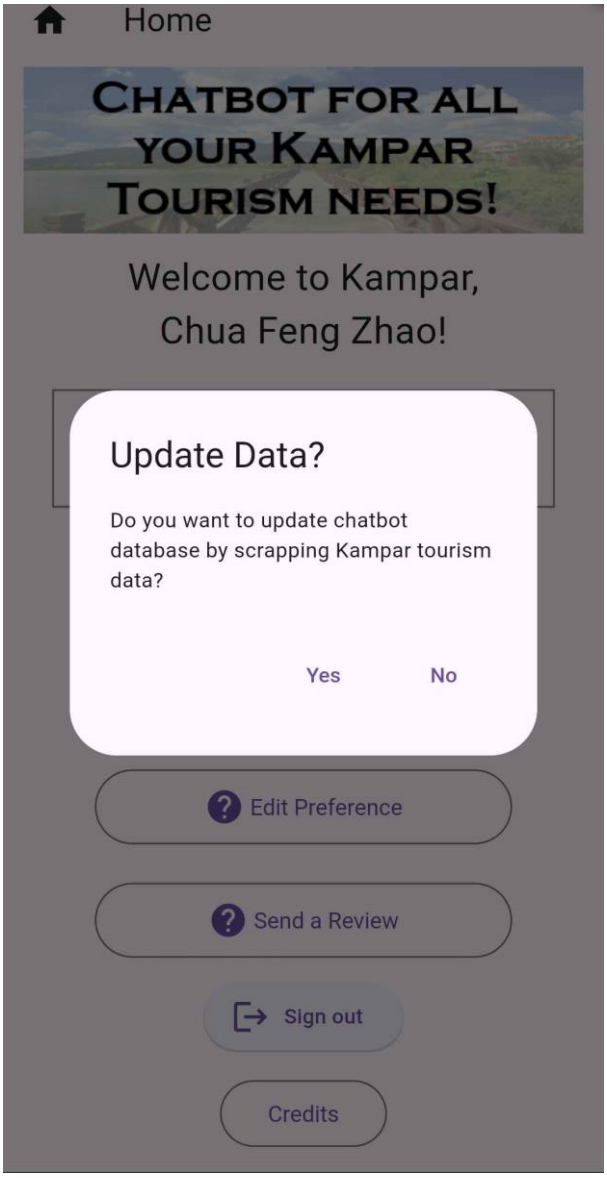


Figure 5.4.10.1: Admin Prompted to Re-scrape and Kampar Tourism Data


```
    {
        "Review Title": "N/A",
        "Review Text": "Stratergic location, good service . Reasonable price cafe food. Quiet and spacious rooms. Recommend to stay
        "Review Rating": 4,
        "Review Date": "5 months ago",
        "Helpful Rating": 0
    },
    {
        "Review Title": "Great Hotel!",
        "Review Text": "A very clean hotel",
        "Review Rating": "4",
        "Review Date": "08/09/24",
        "Helpful Rating": 0
    }
],
```


Figure 5.4.10.2: Review from Application Stored Locally

5.5 Implementation Issues and Challenges

OpenAI Assistant is a relatively new technological development for developers, as such, implementing it into the project provided a lot of issues and challenges as there are not many reports are guides when it comes to its implementation into developer's projects.

In addition, a Flutter SDK was used to communicate with the OpenAI API. However, since OpenAI is new and is still receiving constant updates, there are times when a new feature has been developed for the chatbot, but the SDK has not updated yet, leaving the project unable to proceed until the SDK fully updated.

Moreover, sometimes communication with the API can be unstable due to the traffic to this extremely popular API. Thus, sometimes messages can't be sent fully, or retrieval of the Kampar tourism files can fail with no fault of the user. These usually come with an error code of 500, which according to OpenAI's website, means it is an error on their end [21].

Furthermore, since suggesting tourists with recommendations has many nuances and rules, many rules and protocols are implemented into the chatbot. However, it is difficult to make OpenAI acknowledge all the rules and protocols. Wording needs to be precise and changes in the protocol can make the chatbot act different than desired. In addition, small changes in the assistant's protocol can change how the chatbot provides information entirely, thus, it was hard to try to amend errors as other errors that were once thought fixed could re-emerge.

Finally, the process of web scraping Kampar tourism information from TripAdvisor proved to be difficult as TripAdvisor had a strong bot detection system which meant the web scrapper could not scrap the website continuously. Thus, scraping took intervals where it took several hours just to finish the scrapping process.

5.6 Concluding Remarks

Overall, the system is able to perform all the basic functionalities that is expected of a chatbot and at the same time provide a chatbot with multiple setting options to change the additional information being sent and the capability to change the user's travelling preferences which allows the chatbot to be unique and stand out.

In the next chapter, the results and output of the chatbot will be evaluated.

Chapter 6
System Evaluation and Discussion

6.1 System Testing and Performance Metrics

To test the system, the system's functionalities outside of the chatbot's functionality itself, such as logging in and deleting the chat history will be tested against the expected outcome of the functions. If the tested outcome matches the expected outcome, the test succeeds.

After that, the chatbot's functionalities will be tested. The same process applies where the tested outcome is tested with the initial outcome. Remarks and explanation will be given in order to clarify and justify certain traits that should be noted for the chatbot and to point out any flaws or limitations that may exist in the chatbot's functionalities. Screenshots will also be provided for the chatbot functionalities to support any remarks that may be given.

6.2 Testing Setup and Results

The test is all done in the android Mobile Phone that is stated in Chapter 5.

6.2.1 System Functionalities Outside Chatbot

Table 6.2.1.1: Testing of System Functionalities Outside Chatbot

Test Case	Description	Tested Outcome	Expected Outcome	Evaluation
Register Account	User makes new account by inputting an email and a password	An account is made and is stored in Firebase	An account is made and is stored in Firebase	Pass
Failing to Login	User provides an email and password not	User is prevented from logging in and seeing home page	User is prevented from logging in and seeing home page	Pass

	present in Firebase			
Successfully Logging In	User provides an email and password that exists in Firebase	User successfully logs into the app and sees home page	User successfully logs into the app and sees home page	Pass
Deleting Account	User deletes their account from Firebase Authentication	User's account no longer exist in Firebase Authentication	User's account no longer exist in Firebase Authentication	Pass
Changing Password	User changes their password due to forgetting their old password	An email is sent to the user's email account with a link to change their password	An email is sent to the user's email account with a link to change their password	Pass
Changing Name	User enters 'Account' section of application and changes their name. The name is then visible in the home screen instead of their email.	User's name displays in home page instead of email	User's name displays in home page instead of email	Pass
Communicating with Chatbot	User enters message into chatbot and receives a reply	User's message appears on screen as a text bubble. A few seconds later, the chatbot replies with their	User's message appears on screen as a text bubble. A few seconds later, the chatbot replies with their	Pass

		message appearing as a text bubble	message appearing as a text bubble	
Deleting Chat History	User enters 'Settings' menu and clicks on the red 'Delete Chat History' button. The chat screen will be empty.	User's chat screen is empty	User's chat screen is empty	Pass
Editing Preferences	User enters preferences they wish to share, leaving preferences they don't want to share empty. Preferences shared should be saved in Firestore Database and preferences not shared should be saved as "User does not want to share this" or "User does not have a preference"	Firestore Database stores preferences users want to share appropriately, and stores preferences users do not want to share as "User does not want to share this" or "User does not have a preference"	Firestore Database stores preferences users want to share appropriately, and stores preferences users do not want to share as "User does not want to share this" or "User does not have a preference"	Pass
Making Review	Users makes review of specified place	The review made is stored in	The review made is stored in	Pass

		Firestore Database	Firestore Database	
Prompt for automatic web scraping	If the user is the admin account, upon logging in, the admin will be prompted if they want to scrape and update the tourism data stored in the OpenAI Assistant knowledge base and also obtaining the reviews made from the users of the application	The update tourism data prompt only appears if the user is the admin and upon login and reviews made from this application is grabbed and stored	The update tourism data prompt only appears if the user is the admin and upon login and reviews made from this application is grabbed and stored	Pass

6.2.2 Chatbot Functionalities

Firstly, the chatbot's main purpose, which is to provide suggestions, is tested. Figures 6.2.2.1, 6.2.2.2 and 6.2.2.3 shows the test that is done to show the chatbot is able to differentiate the suggestions for hotels, restaurants and attractions. This also shows the chatbot is able to recognize more foreign terms, such as "grabbing a bite". This shows that the chatbot is able to act more human-like, which helps users ease into the use of the chatbot as a virtual tour guide. The results show the chatbot providing three suggestions, which helps gives users options as users may not prefer the first suggestion given. This also shows that the chatbot is ablet to provide suggestions even when users do not give much information for the chatbot. One flaw that can be taken from this is that sometimes it takes a lot more from user's reviews, such as naming the attraction with the slang used in the reviews instead of the actual named scraped.

Figure 6.2.2.1: Testing Hotel Suggestions

Figure 6.2.2.2: Testing Restaurant Suggestions

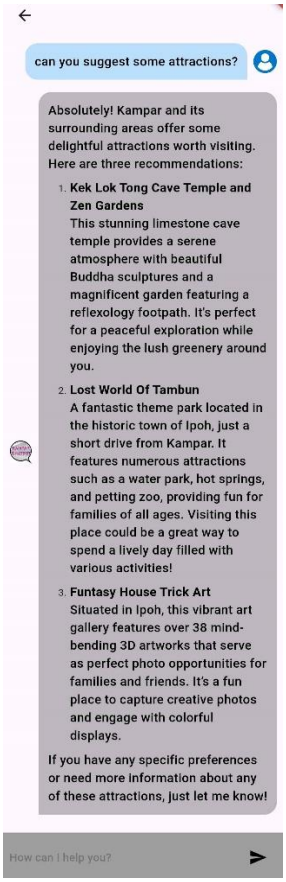


Table 6.2.2.1: Testing Chatbot Functionalities (1)

Figure 6.2.2.3: Testing Attraction Suggestions

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking for suggestions	Chatbot provides three suggestions of either hotels, restaurants or attractions depending on user's query	Chatbot provides three suggestions of either hotels, restaurants or attractions depending on user's query	Pass	Sometimes takes a lot more from user's reviews as facts

Then, the chatbot is then tested to see if it can follow up on information. Users may want more information on a specific location or attraction after it is suggested. Thus, it is imperative that the chatbot is able to provide extra information for users once it has given its suggestions. Figure 6.2.2.4 shows the user following up on the chatbot's hotel suggestion by asking for more information on Grand Kampar Hotel, in which the chatbot provides the information.

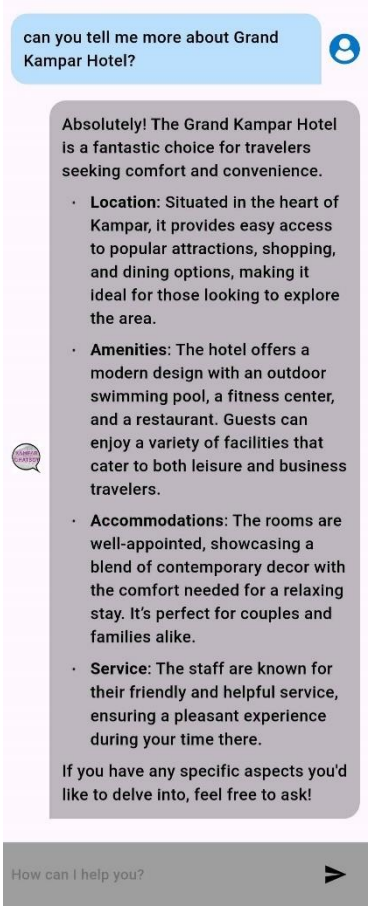


Figure 6.2.2.4: Testing Follow-Up Information

Table 6.2.2.2: Testing Chatbot Functionalities (2)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking for follow-up information	Chatbot provides more information on a	Chatbot provides more information on a	Pass	-

	location that the user requests for	location that the user requests for		

In addition, the chatbot is also tested for its ability to understand the user's emotional context. For example, if the user dislike a particular attraction, the chatbot should be able to recognize that and provide recommendations that are not similar to the attraction that the user dislikes. Figure 6.2.2.5 shows the chatbot responding to the user's dislike of Kinta Mining Museum with attractions that are more active and fun. While the chatbot is able to recognize user's negative and positive emotions, one limitation found is that the chatbot sometimes misinterprets the reason why users might dislike the place if the user does not provide extra context for the chatbot.

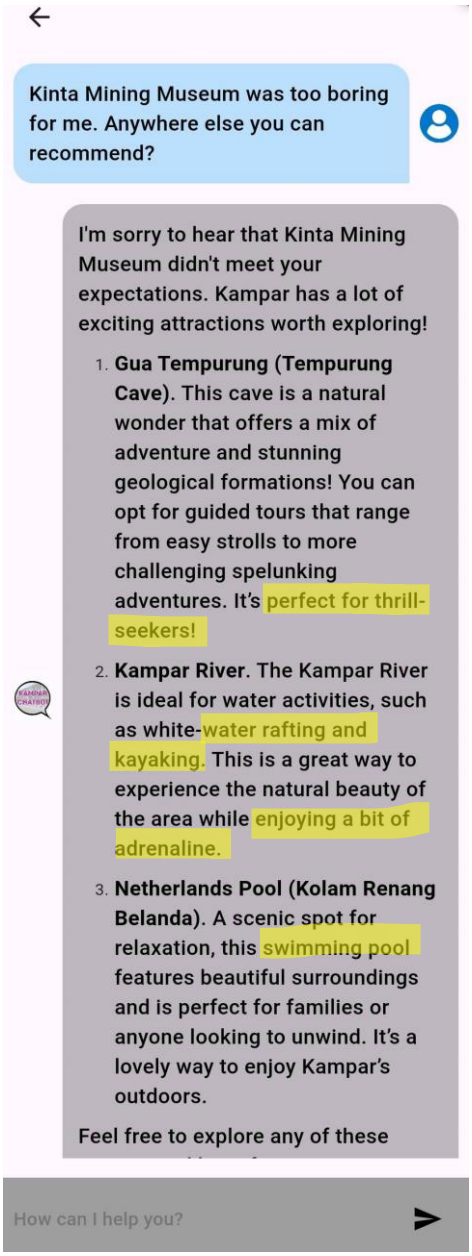


Figure 6.2.2.5: Testing Emotional Context

Table 6.2.2.3: Testing Chatbot Functionalities (3)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking questions with emotional context	Chatbot considers emotional context when providing response	Chatbot considers emotional context when providing response	Pass	User may need to provide more context for the chatbot to fully understand emotional context

The chatbot is then tested for its ability to provide information on certain local cuisine and cultures. While food items like 'Nasi Lemak' is common knowledge to local Malaysians, travellers and tourists may not have an understanding as to what that is. With the chatbot acting as a local tour guide, it is important that it can also provide this local information. Figure 6.2.2.6 shows the test done to show the chatbot's capabilities on providing local information. Nasi Lemak is used in this example.

Figure 6.2.2.6: Testing Requesting Information About Local Cuisine or Culture

Table 6.2.2.4: Testing Chatbot Functionalities (4)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking about local tradition or cuisine	Chatbot provides information on the local cuisine or tradition stated	Chatbot provides information on the local cuisine or tradition stated	Pass	-

Moreover, the chatbot is also tested for its ability to provide directions for users. As it is meant to be a virtual tour guide, it is also important that it can help guide users to the locations they want to visit. It is also important for the chatbot to not provide directions to a completely different direction. Unfortunately, due to the limitations of OpenAI Assistants, users must be navigated to a separate page to view the map and direction rather than it being able to be displayed on the chat itself. The chatbot is also only able to direct users to one location at a time. Figures 6.2.2.7 and 6.2.2.8 that the chatbot is able to guide users to the location they asked for.

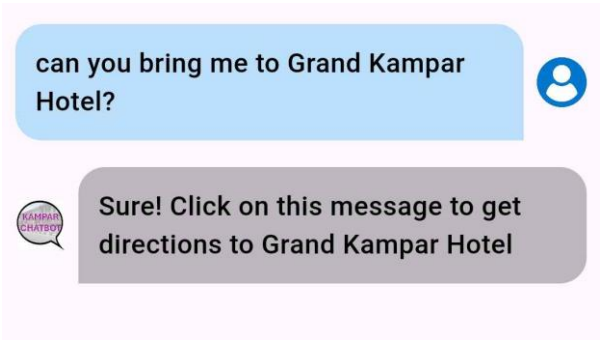


Figure 6.2.2.7: Testing Requesting for Directions

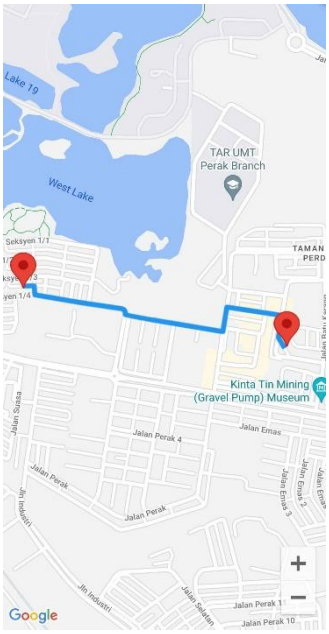


Figure 6.2.2.8: Google Maps Showing Direction to Destination

Table 6.2.2.5: Testing Chatbot Functionalities (5)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking for direction to a specific place	Chatbot provides a Google Maps page with directions to user's specified location	Chatbot provides a Google Maps page with directions to user's specified location	Pass	Users need to click on the message to access the Google Maps. Users can only be directed to one location at a time.

Then, the chatbot is tested on its ability to understand user's preferences. This is also important as it gives the chatbot the personalization component. It allows tourists with different taste to get different responses which aligns with their preferences. During testing, 5 separate preferences are tested. Figures 6.2.2.9 and 6.2.2.10 show the test for the user travelling with their family, Figures 6.2.2.11 and 6.2.2.12 shows the test for the chatbot to understand the user's strict budget constraints and Figures 6.2.2.13-6.2.2.18 shows the test for the chatbot to understand the user's hotel (the user wants hotels with fun activities to do), restaurant (the user wants restaurant with seafood) and attraction (the user wants attractions that shows the history of Kampar) preferences. Through the testing, it is found that the chatbot is not as proficient and financial constraints. This could be due to the lack of information when it comes to monetary information.

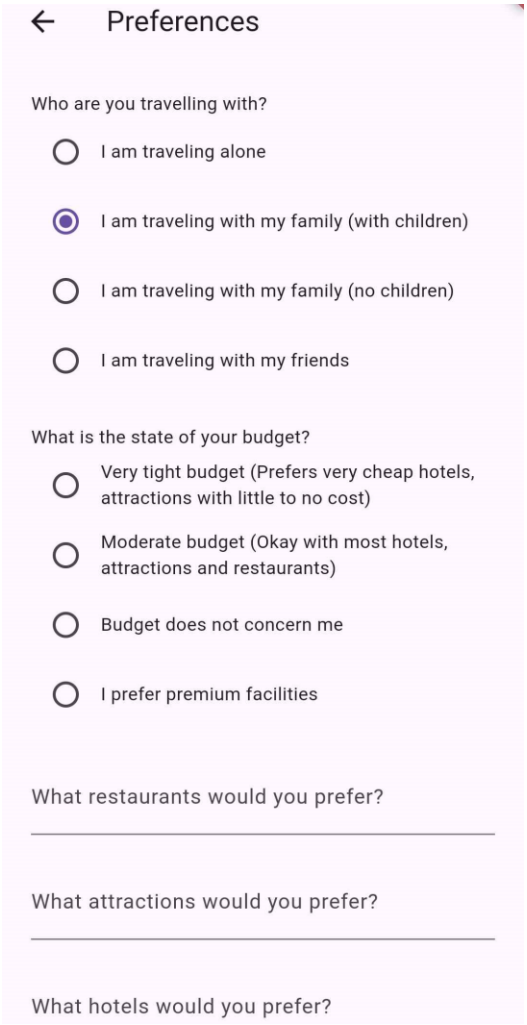


Figure 6.2.2.9: Setting Traveling Group Preference

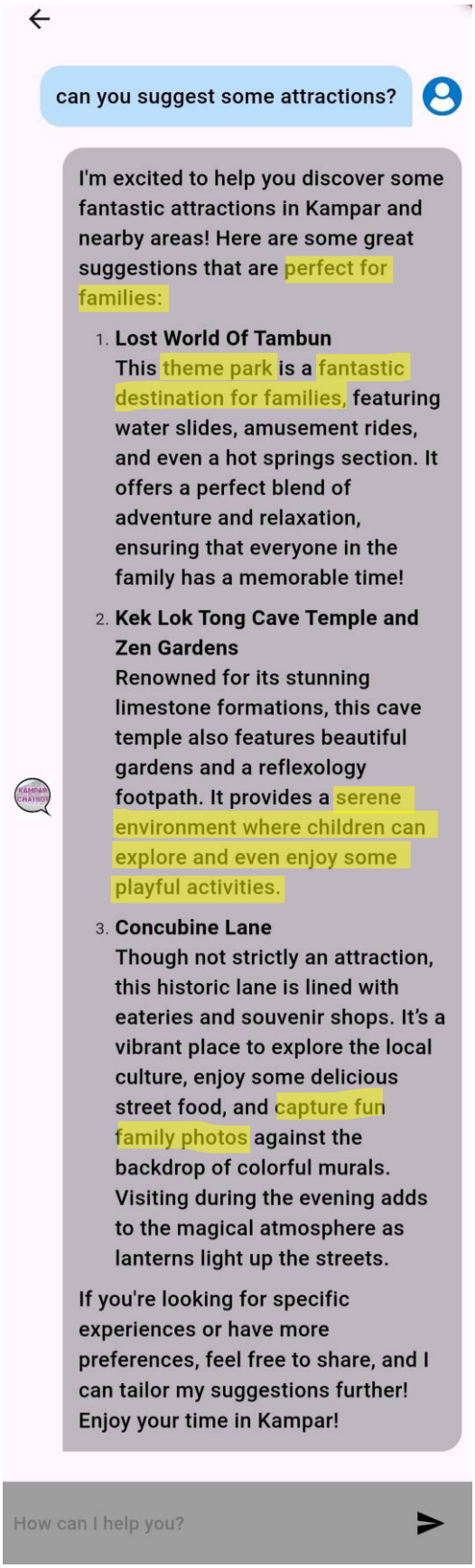


Figure 6.2.2.10: Testing Suggestion with Traveling Group Preference

Figure 6.2.2.11: Setting Budget Constraint Preference

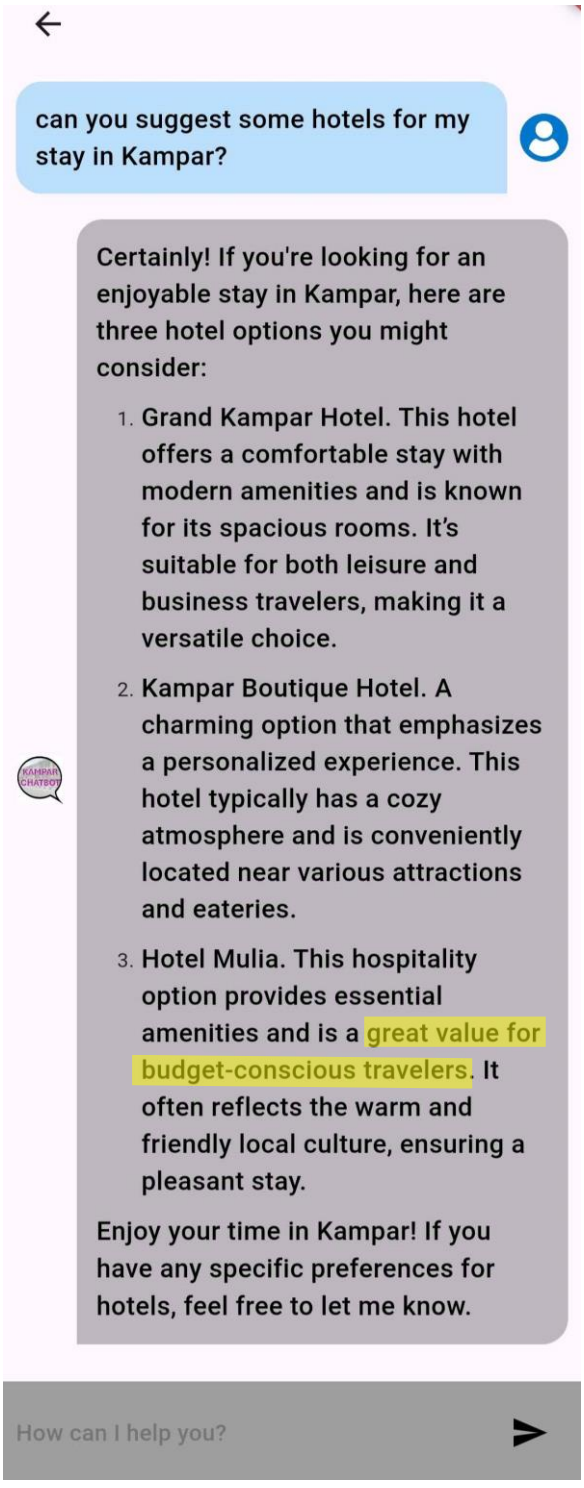


Figure 6.2.2.12: Testing Suggestion with Budget Constraint Preference

Figure 6.2.2.13: Setting Restaurant Preference

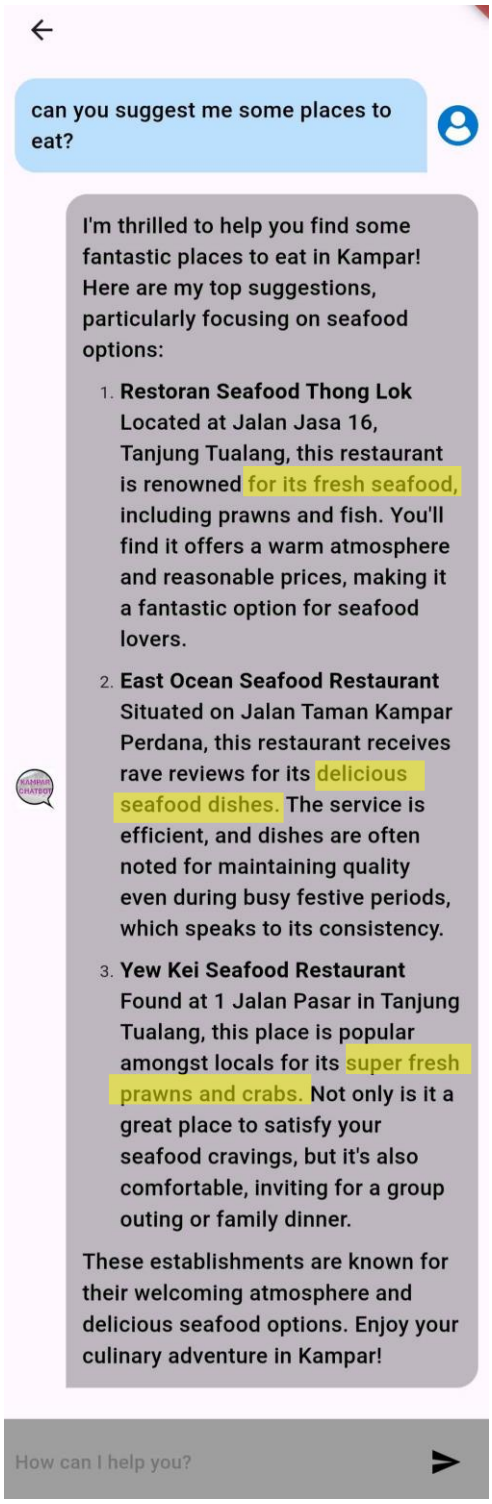


Figure 6.2.2.14: Testing Suggestion with Restaurant Preference

Figure 6.2.2.15: Setting Attraction Preference

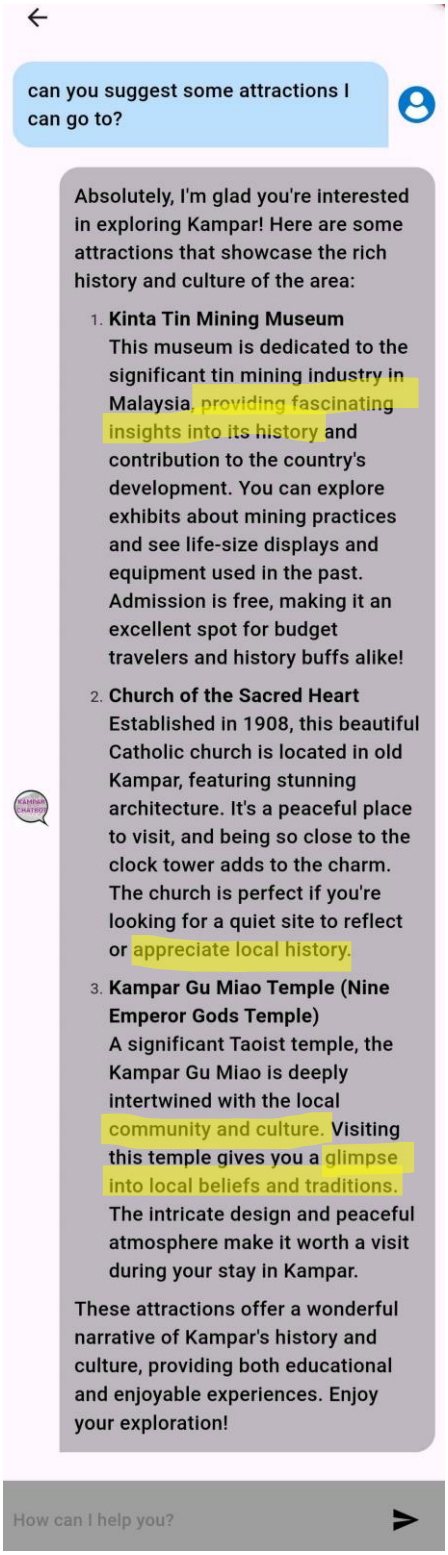


Figure 6.2.2.16: Testing Suggestion with Attraction Preference

Figure 6.2.2.17: Setting Hotel Preference

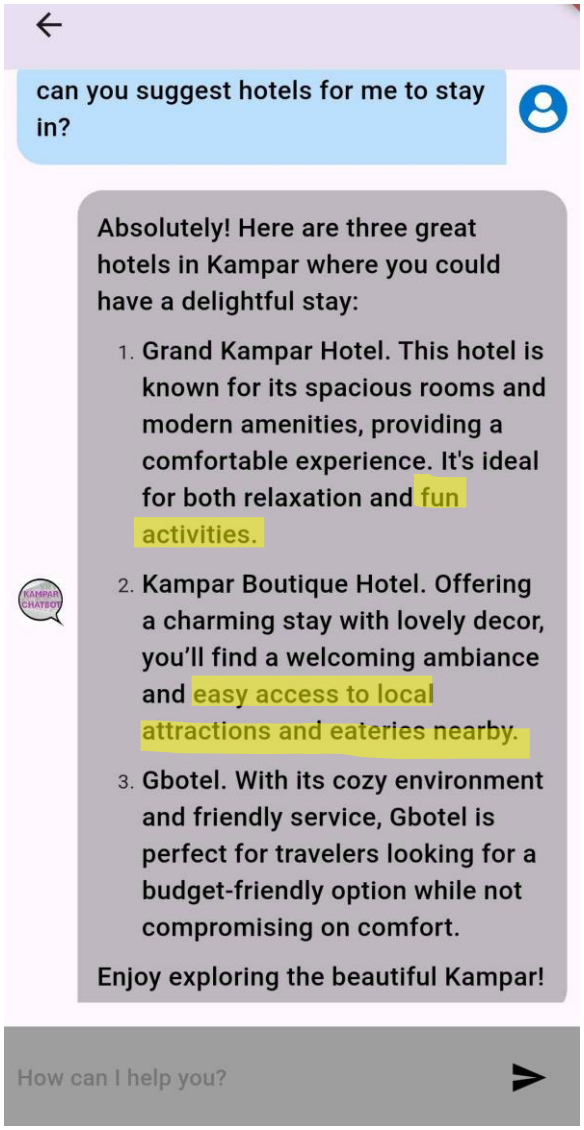


Figure 6.2.2.18: Testing Suggestion with Hotel Preference

Table 6.2.2.6: Testing Chatbot Functionalities (6)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking suggestions with travelling preferences	Chatbot provides information based on the preferences	Chatbot provides information based on the preferences	Pass	Chatbot struggles a little bit providing suggestion based on

	given appropriately	given appropriately		financial constraint due to the lack of financial information during the data collection phase.

Next, the chatbot is tested for the additional information users can choose to send by toggling them in the settings. These are recommended for users who are already in Kampar and looking for suggestions on the spot as these factors can influence what is an optimal place to go. For example, for the first test in this category, the chatbot is tested for its understanding of providing suggestions with the current time and weather in Kampar in mind. The chatbot is seen to provide suggestions for attractions that fit an overcast afternoon, as seen in Figure 6.2.2.19.

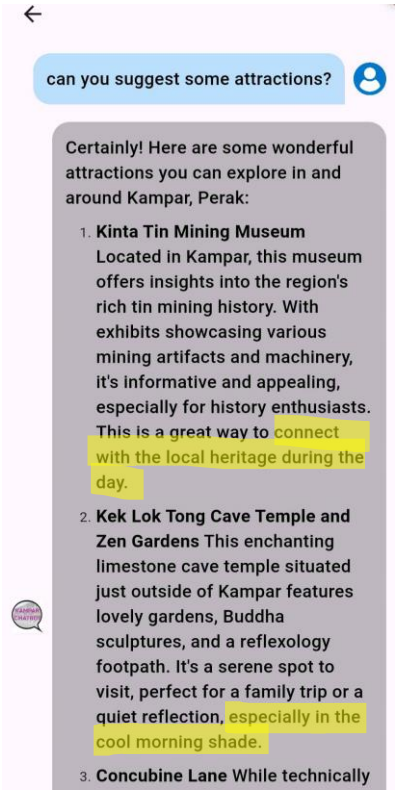


Figure 6.2.2.19: Testing suggested with Respect to Weather and Time

Table 6.2.2.7: Testing Chatbot Functionalities (7)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking questions with Kampar's current time and weather passed as additional information	Chatbot provides suggestions based on the time and weather passed	Chatbot provides suggestions based on the time and weather passed	Pass	-

The chatbot is also tested for providing information based on the user's current coordinates in Kampar. Figures 6.2.2.20-6.2.2.23 shows the users asking for locations close by and the chatbot providing locations that are close by the user, alongside Google Maps images that prove they are close to the location in which the chatbot is tested at with travelling distance being not more than 8 minutes by car. The results show that the chatbot can only respond with locations close to the user only if the user requests it, which inconveniences the user somewhat as they have to specify it themselves to the chatbot.

Figure 6.2.2.20: Testing Suggestions in Relation to Current Coordinates

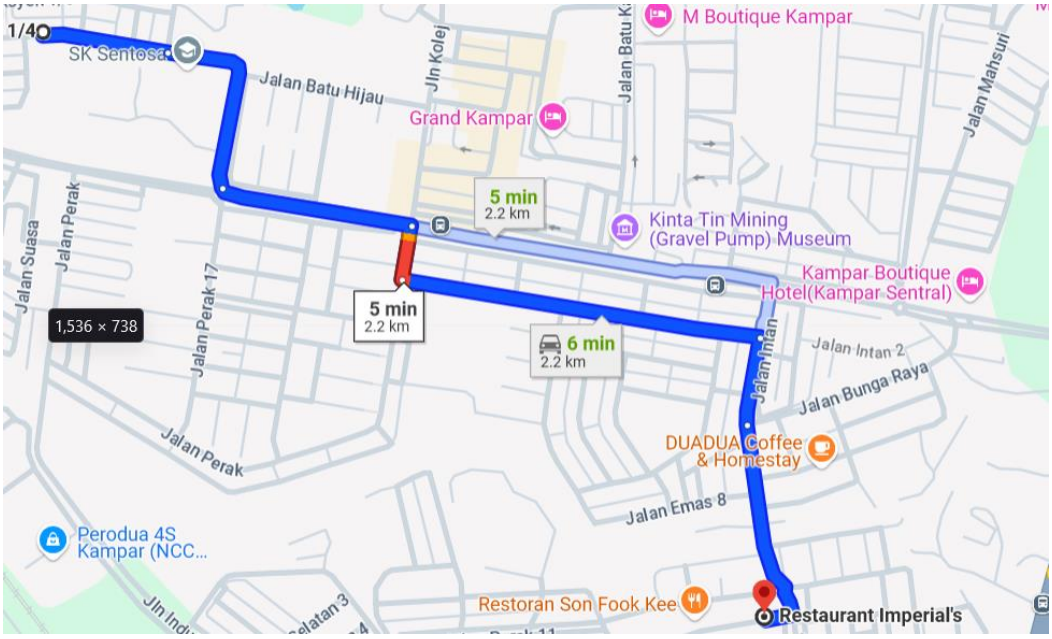


Figure 6.2.2.21: Map to Restoran Imperial's

Figure 6.2.2.22: Map to East Ocean Seafood Restaurant

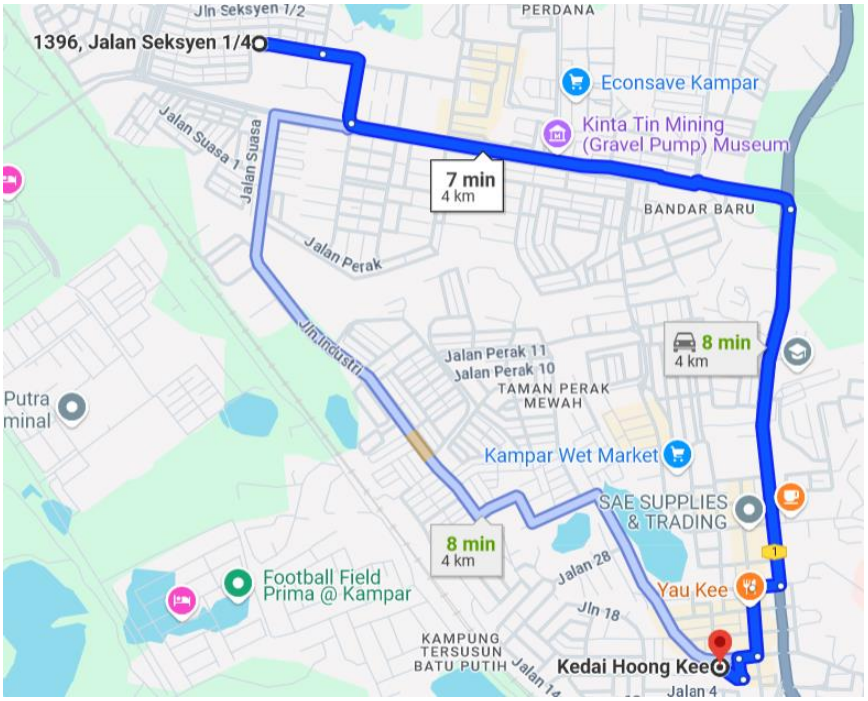


Figure 6.2.2.23: Map to Kedai Hoong Kee

Table 6.2.2.8: Testing Chatbot Functionalities (8)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Asking questions with user's coordinates passed as additional information	Chatbot provides suggestions based on closeness of location to user's coordinates	Chatbot provides suggestions based on closeness of location to user's coordinates	Pass	Users need to specify they want places that are close for the chatbot to respond with the information wanted.

Finally, the chatbot is also tested for its ability to update user's preferences as they communicate with the chatbot. Figures 6.2.2.24, 6.2.2.25 and 6.2.2.26 shows the user's preferences before and after the user asks for a restaurant that does not align with the user's original restaurant preferences. The results show that the chatbot is able to identify that the user would occasionally go for local cuisine in addition to their original preference for seafood. However, this functionality is not perfect. Should users enable this functionality, the chatbot is inclined to always produce a response and update the user's preferences. While during the implementation of the system, the preference updater chatbot is made to sometimes not update the user's preferences whenever there is little changes, these occurrences are rare. This leads to user's preferences sometimes being updated to having false positive information as due to the nature of OpenAI, it leans heavily onto statements being true rather than not. This means that the AI prefers to say a user prefers a place or attraction just because they have asked about it before, which can sour the user's further suggestions the longer their chat history goes, as each time the user's sends a message, the entire chat history and the user's old preference is sent as reference for the preference updater chatbot.

Figure 6.2.2.24: Preferences Before Messaging Chatbot

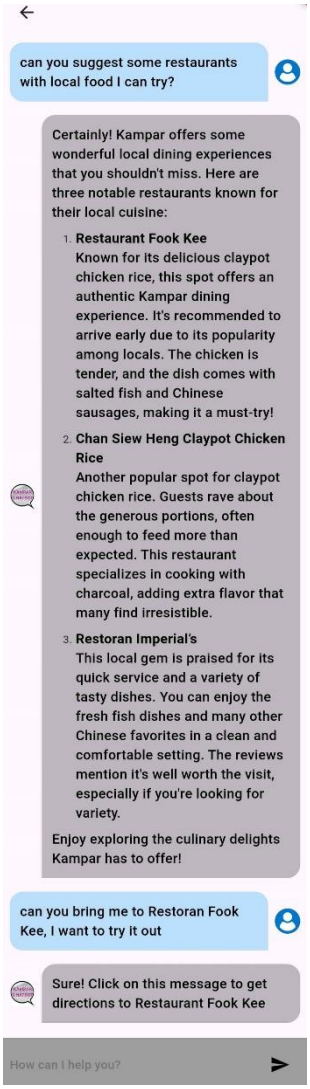


Figure 6.2.2.25: Messaging Chatbot with Different Request from Preference

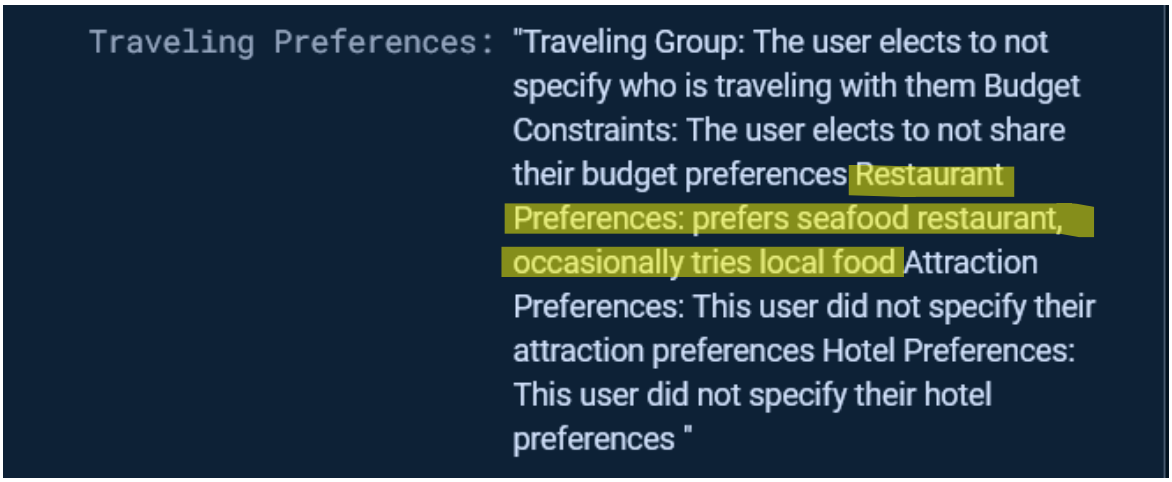


Figure 6.2.2.26: Preferences After Messaging Chatbot

Table 6.2.2.9: Testing Chatbot Functionalities (9)

Test Case	**Tested Outcome**	**Expected Outcome**	**Evaluation**	**Limitations**
Updating user's preferences based on chat history	Chatbot updates user's preferences based on its conversation with its user. Will not update preferences if asked for suggestion.	Chatbot updates user's preferences based on its conversation with its user. Will not update preferences if asked for suggestion.	Pass	Chatbot heavily leans into statements being true, which leads to it citing false positives such as users preferring a place just because the user asks about it in the chat history.

6.3 Project Challenges

The first major challenges encountered in this project is the lack of certain information. Especially monetary information. Since price information is always fluctuating, it was hard to grab this information to be stored in the OpenAI Assistant's vector database. This makes the chatbot not as informed when it comes to the monetary information and thus, it cannot help users with a lot of monetary based suggestions.

Furthermore, another major challenge encountered in this project is the inconsistency with the chatbot. Due to the random, probability-based nature of a large language model, which generates words based on the probability of words coming next during generation, the chatbot's responses vary and it is hard to get it to be consistent. For example, the output of the chatbot may sometimes have bolded titles and italicized descriptions, and sometimes do not. This sometimes also affects the suggestions it gives to the users as well such as sometimes not mentioning certain preferences in the description of the suggestion even though the location does fit with the user's preferences. Even with protocols in place which trains the chatbot to behave a certain way, it will always have a variance to its answering method, which is hard to change. Especially since, as mentioned in Chapter 5, one small change to the protocol of the OpenAI Assistant can largely affect how it answers certain queries, so changing the protocol to reign in the issues of undesirable variance is difficult.

6.4 Objectives Evaluation

The chatbot succeeds in achieving the project's objectives, which is to provide a chatbot which provides immediate travelling suggestions for the users travelling in Kampar, especially travellers who are travelling on the spot. The chatbot also achieves the ability to be personalized to each traveller with its ability to change and even update preferences, allowing for unique responses for each user. This cuts the time needed from the user to manually search for hard-to-find information that is scattered across the Internet just to find the place they might like to go to.

The chatbot is also able to provide additional information on places and information on cultures and local cuisines, which supports the chatbot's 'support and answer' model it aims for.

In addition, the chatbot is able to act less like a robot assistant and more like a tour guide as it understands foreign terms and context, such as 'grabbing a bite', understanding what a user

may dislike based on their query and can uniquely identify requests such as requesting for direction versus requesting for suggestions or information.

The chatbot is also able to provide more detailed information as compared to previous chatbots due to the language model being able to construct and extract much more complex contexts, especially from the Kampar tourism data stored in the OpenAI assistant's knowledge base.

6.5 Concluding Remark

Overall, the project passes many of the tests that was planned for it. As such, the project can be considered a success as the chatbot is not only able to provide suggestions for users, but it also does so while being able to consider multiple factors such as travelling preferences, weather, time and the user's current coordinates. In addition, the chatbot is also able to perform extra functions which can improve the user's travelling experience such as the ability to provide suggestions and updating their travelling in real time as they message the chatbot.

The Kampar tourism data scraped also proved to be extremely useful to the output of the chatbot as it is able to allow the chatbot to provide much more detailed and specific explanations for users which is grabbed from the user reviews and details of the places stored in the scraped data. In addition, the scraped data also ensures that the data provided is much more up to date compared to the information provided by the chatbot without embeddings, as OpenAI does not update the GPT knowledge base often. Figures 6.5.1, 6.5.2, 6.5.3 and 6.5.4 shows the difference between the chatbot with and without the Kampar tourism data embedded for the use of RAG, with Figure 6.5.1 and 6.5.3 being before the embedding of Kampar tourism information and only providing information based on GPT-4o mini's limited knowledge of Kampar tourism, and Figure 6.5.2 and 6.5.4 being after the embedding of Kampar tourism information. These examples are taken from the OpenAI simulator for the chatbot.

Figure 6.5.1: Chatbot Output Before Embedding Kampar Tourism Data (1)

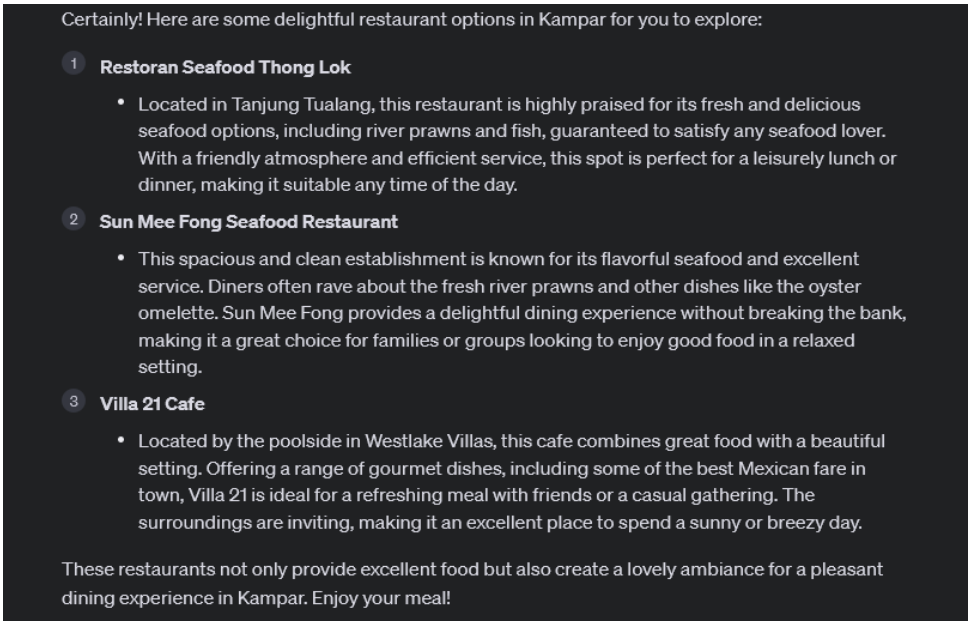


Figure 6.5.2: Chatbot Output After Embedding Kampar Tourism Data (1)

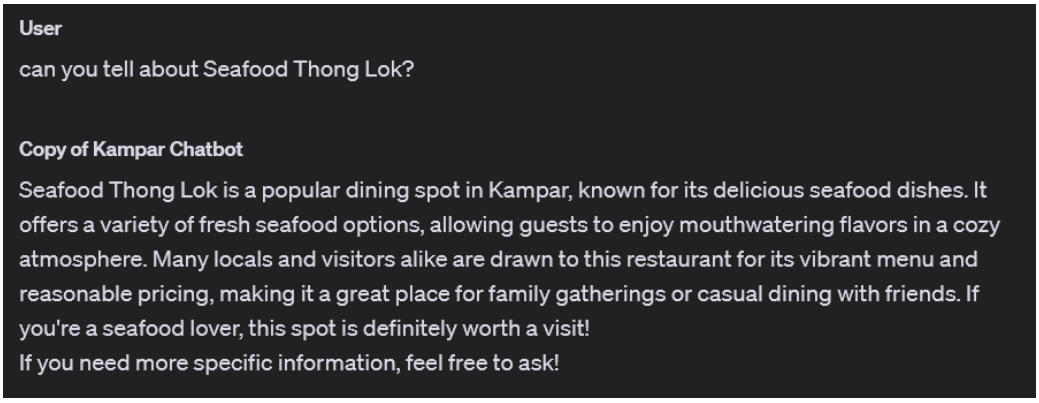


Figure 6.5.3: Chatbot Output Before Embedding Kampar Tourism Data (2)

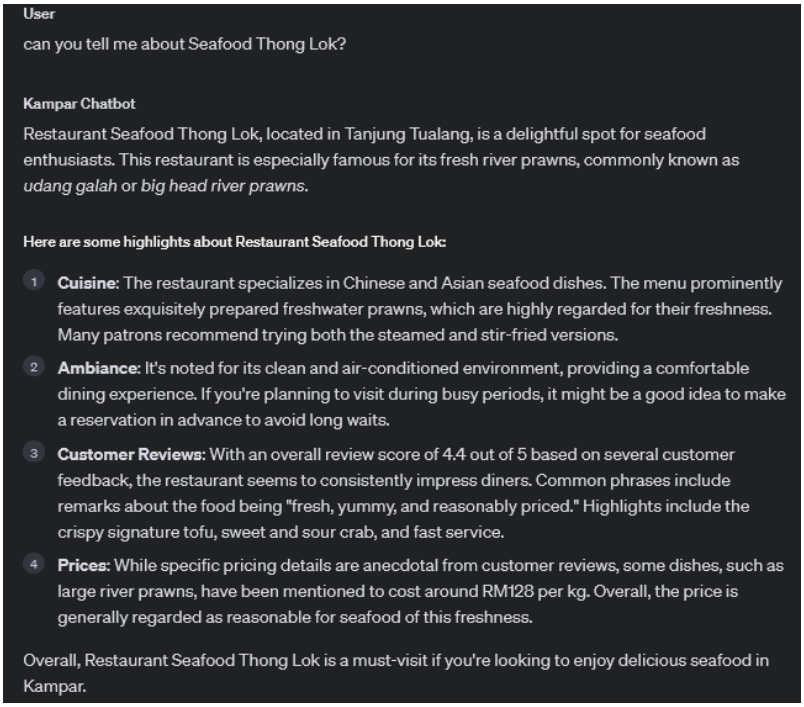


Figure 6.5.4: Chatbot Output After Embedding Kampar Tourism Data (2)

However, the chatbot is not without its flaws. For instance, the chatbot faces issues when it comes to suggesting hotels or providing information of the price of places, this is due to the lack of good source of information of prices, in addition to the ever-fluctuating prices of hotels and restaurants.

Furthermore, the chatbot will not be able to provide certain niche information for similar reasons, such as time an attraction or restaurants opens for, and whether or not certain attractions have any entry fee, as this information is also very scarce on the internet.

The chatbot is also unable to help users book for hotels. Which does hassle users as they will have to book the hotels by themselves through the hotel's website or manually in-person. The chatbot also struggles when it comes to hotels a lot more compared to restaurants and attractions. This is likely due to the similarities when it comes to the reviews in hotels. This is also heavily affected by the fact there are not many hotels with websites in Kampar.

In spite of these flaws, the chatbot serves to provide improvements in many ways compared to previously developed chatbots with its deep language understanding and much deeper user profiling and personalization.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

Tourism is an important part to a country's economy. Unfortunately, a lack of availability to information of a certain traveling location can turn potential tourists off from traveling to that location as they don't want to deal with the hassle of searching for that information.

Kampar is one of these locations where information of it is scattered thin throughout the internet, with no main central location where users can quickly access all of this information without hassle.

This project was developed in hopes to provide a 'command and answer' model for users who wants to travel to Kampar, where users can obtain both information and suggestions on places to go in Kampar by just asking questions rather than having them to search the places out themselves.

The chatbot was developed using the OpenAI model using their Assistant module in order to assist in storing and retrieving information of Kampar tourism that is scrapped. The OpenAI Assistant uses a concept known as RAG in order to store and retrieve foreign data to assist the language model in providing Kampar tourism suggestion and information for users. The data is stored using a process called vectorization through embedding and stored in a vector database. Using the large language model that is provided by OpenAI, GPT-4o mini, the chatbot was able to interpret user's request and provide an appropriate response to the user by comparing the vectors of the user's message and the Kampar tourism data stored in the Assistant's vector database. The responses provided by the chatbot with the vectorization and is seen to be much more comprehensive and contains much more information.

The chatbot, through the language model used, has a deep language understanding, being both able to understand user's request very well and also provide complex and comprehensive answers.

Users can also indicate their traveling preferences in which the chatbot can take into account when forming its suggestions and responses so that users can obtain a more personalized response.

Users are also able to send additional information such as current weather and time in Kampar and the user's current location in order to improve the suggestions provided by the chatbot.

Moreover, the chatbot is able to perform additional functions to improve the tourist's experience with the application, namely providing directions and updating tourist's travelling preferences.

This shows using a large language model with RAG is a huge next step forward for tourism chatbots.

7.2 Recommendation

This project uses the RAG algorithm provided by OpenAI, which is built into the OpenAI Assistant. While this meant easier implementation as both the knowledge base and Assistant were part of one service, it comes at a cost of flexibility when it comes to the retrieval of data from the knowledge base. Thus, one recommendation that could be given to improve this project in the future is to build the knowledge base of the chatbot from scratch instead of using the one provided by OpenAI. This involves opening a vector database and vectorizing the Kampar tourism data manually then linking the chatbot to the custom vector database. The benefits of this is more flexibility of the data and the retrieval process that is done by the chatbot to the tourism data. This can improve the quality and efficiency of the chatbot.

Furthermore, while testing is done on the application, it has not been tested with the actual intended audience of the application, foreigners due to the lack of resources and the difficulty of finding foreigners in Kampar who are willing to test the application Thus, for future development of the application, the application can be given to actual foreigners to test the application. This way, more meaningful feedback can be given as the actual intended audience of the application will be testing.

Moreover, while the application does allow users to submit reviews, the reviews are only stored locally and will not be seen by other users. One other improvement is to make the reviews public to everyone so that users can have another avenue in which they can obtain their recommendations from. The chatbot could also look into these reviews to further improve its output.

REFERENCES

[1] “Meaning of tourist in English,” dictionary.cambridge.org. https://dictionary.cambridge.org/dictionary/english/tourist (accessed Sept. 07, 2023)

[2] Statista Research Department, “Travel and tourism in Malaysia – statistics & facts” statista.com, Nov. 19, 2022. https://www.statista.com/topics/5741/travel-and-tourism-in-malaysia/#topicOverview (accessed Sept 07, 2023)

[3] WBB, “How Chatbots are Revolutionising the Travel & Tourism Industry” (accessed Sept 07, 2023)

[4] N. Miew Yiong., “The Development of AI Chatbot for Kampar Tourism”, 2020

[5] A. V. Dian Sano, T. Daud Imanuel, M. Intanadias Calista, H. Nindito and A. Raharto Condrobimo, "The Application of AGNES Algorithm to Optimize Knowledge Base for Tourism Chatbot," 2018 International Conference on Information Management and Technology (ICIMTech), Jakarta, Indonesia, 2018, pp. 65-68, doi: 10.1109/ICIMTech.2018.8528174.

[6] F. Clarizia, F. Colace, M. De Santo, M. Lombardi, F. Pascale and D. Santaniello, "A Context-Aware Chatbot for Tourist Destinations," 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Sorrento, Italy, 2019, pp. 348-354, doi: 10.1109/SITIS.2019.00063.

[7] Nica, I., Tazl, O.A., & Wotawa, F. “Chatbot-based Tourist Recommendations Using Model-based Reasoning,” 2019 ConfWS. Available: https://ceur-ws.org/Vol-2220/05_CONFWS18_paper_31.pdf

[8] L. Zhi Yuan., “Personalised A.I. Chatbot for Kampar Tourism Mobile Application”, 2021

[9] “Language Models, Explained: How GPT and Other Models Work” altexsoft.com, Jan 18, 2023. https://www.altexsoft.com/blog/language-models-gpt/

[10] “What is RAG (Retrieval-Augmented Generation)?” aws.amazon.com. https://aws.amazon.com/what-is/retrieval-augmented-generation/ (Accessed Sept 1, 2024)

[11] “What is Retrieval-Augmented Generation (RAG)?” cloud.google.com. https://cloud.google.com/use-cases/retrieval-augmented-generation (Accessed Sept 1, 2024)

[12] “File Search”, platform.openai.com. https://platform.openai.com/docs/assistants/tools/file-search?context=streaming (accessed Apr 14, 2024)

[13] “Embeddings”, platform.openai.com. https://platform.openai.com/docs/guides/embeddings/what-are-embeddings (accessed Apr 14, 2024)

[14] G., Pornaras, “Selenium vs. Beautiful Soup: A Full Comparison” blazemeter.com, Oct. 7, 2021. https://www.blazemeter.com/blog/selenium-vs-beautiful-soup-python#selenium-vs-beautifulsoup

[15] Danashree, “A complete guide to Web Scraping with Selenium & Python in 2024” nanonets.com, Mar 20, 2023. https://nanonets.com/blog/web-scraping-with-selenium/

[16] TripAdvisor, “Grand Kampar Hotel” https://www.tripadvisor.com/Hotel_Review-g3522152-d1393215-Reviews-Grand_Kampar_Hotel-Kampar_Kampar_District_Perak.html (accessed Apr 14, 2024)

[17] O., Dan, “Detecting addresses with Watson Assistant” medium.com, Jul. 1, 2021. https://medium.com/ibm-watson/detecting-addresses-with-watson-assistant-a3d9028f2a1b

[18] “Assistant API”, platform.openai.com. https://platform.openai.com/docs/assistants/overview?context=with-streaming (accessed Apr 14, 2024)

[19] E., David, “What Temperature Means in Natural Language Processing and AI” thenewstack.io, Jan. 9, 2024. https://thenewstack.io/what-temperature-means-in-natural-language-processing-and-ai/

[20] K. G. Yager, “Domain-specific chatbots for science using embeddings,” Digital discovery, vol. 2, no. 6, pp. 1850–1861, Jan. 2023, doi: https://doi.org/10.1039/d3dd00112a.

[21] “Error Codes” platform.openai.com. https://platform.openai.com/docs/guides/error-codes/api-errors (Accessed Sept. 1, 2024)

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T2, Y3	**Study week no.: 6**
Student Name & ID: Chua Feng Zhao | 22ACB00604	
Supervisor: Prof. Ts. Liew Soung Yue	
Project Title: MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL	

1. WORK DONE The application is made to allow users to direct them to their desired location
2. WORK TO BE DONE Make the chatbot recognize weather and time as additional information in order to improve the output of the chatbot
3. PROBLEMS ENCOUNTERED The chatbot has a rough time understanding the weather and time information at present
4. SELF EVALUATION OF THE PROGRESS The project progress is satisfactory, however, should probably focus on functions that improve the chatbot output

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T2, Y3	**Study week no.: 7**
Student Name & ID: Chua Feng Zhao | 22ACB00604	
Supervisor: Prof. Ts. Liew Soung Yue	
Project Title: MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL	

1. WORK DONE

The chatbot is now able to consistently recognize weather and time information when suggesting output

2. WORK TO BE DONE

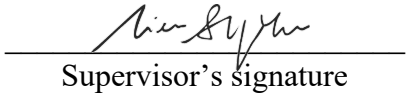
Look into how the RAG function could be improved

3. PROBLEMS ENCOUNTERED

Running out of ideas that can be done to improve the output of the chatbot

4. SELF EVALUATION OF THE PROGRESS

The project progress is satisfactory, however disappointed that not many other ideas can be thought of to improve the chatbot output



Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T2, Y3	**Study week no.: 8**
Student Name & ID: Chua Feng Zhao | 22ACB00604	
Supervisor: Prof. Ts. Liew Soung Yue	
Project Title: MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL	

1. WORK DONE

A custom vector database is looked into to improve RAG retrieval of chatbot, but was ultimately dropped due to time constraints.
The chatbot is now able to update the user's traveling preference in real-time

2. WORK TO BE DONE

Fine-tune the updating of user's traveling preferences

3. PROBLEMS ENCOUNTERED

Only testing the updating preference function with short chat history is feasible

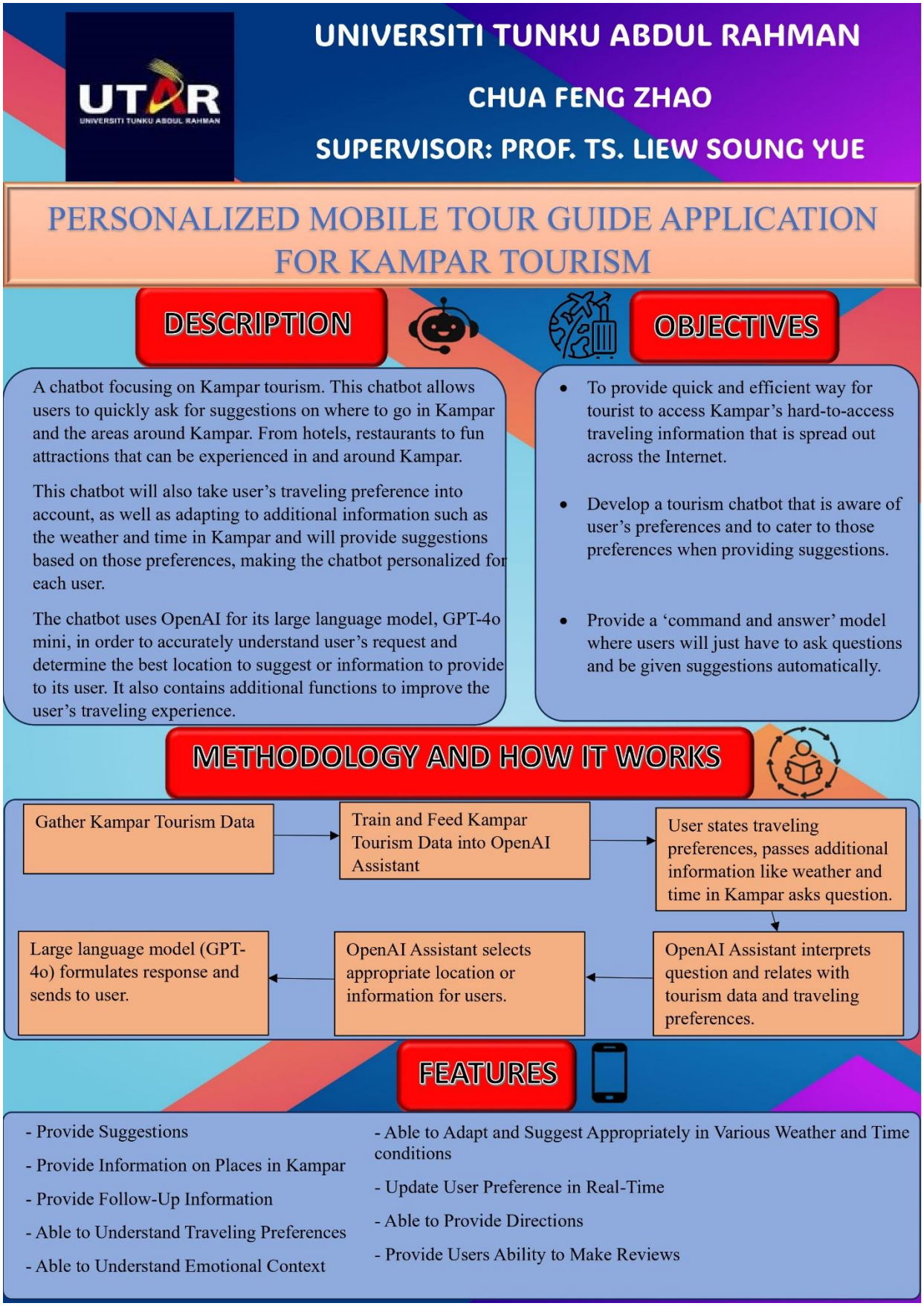
4. SELF EVALUATION OF THE PROGRESS

Satisfied that the chatbot has one more major feature that can improve the chatbot's output

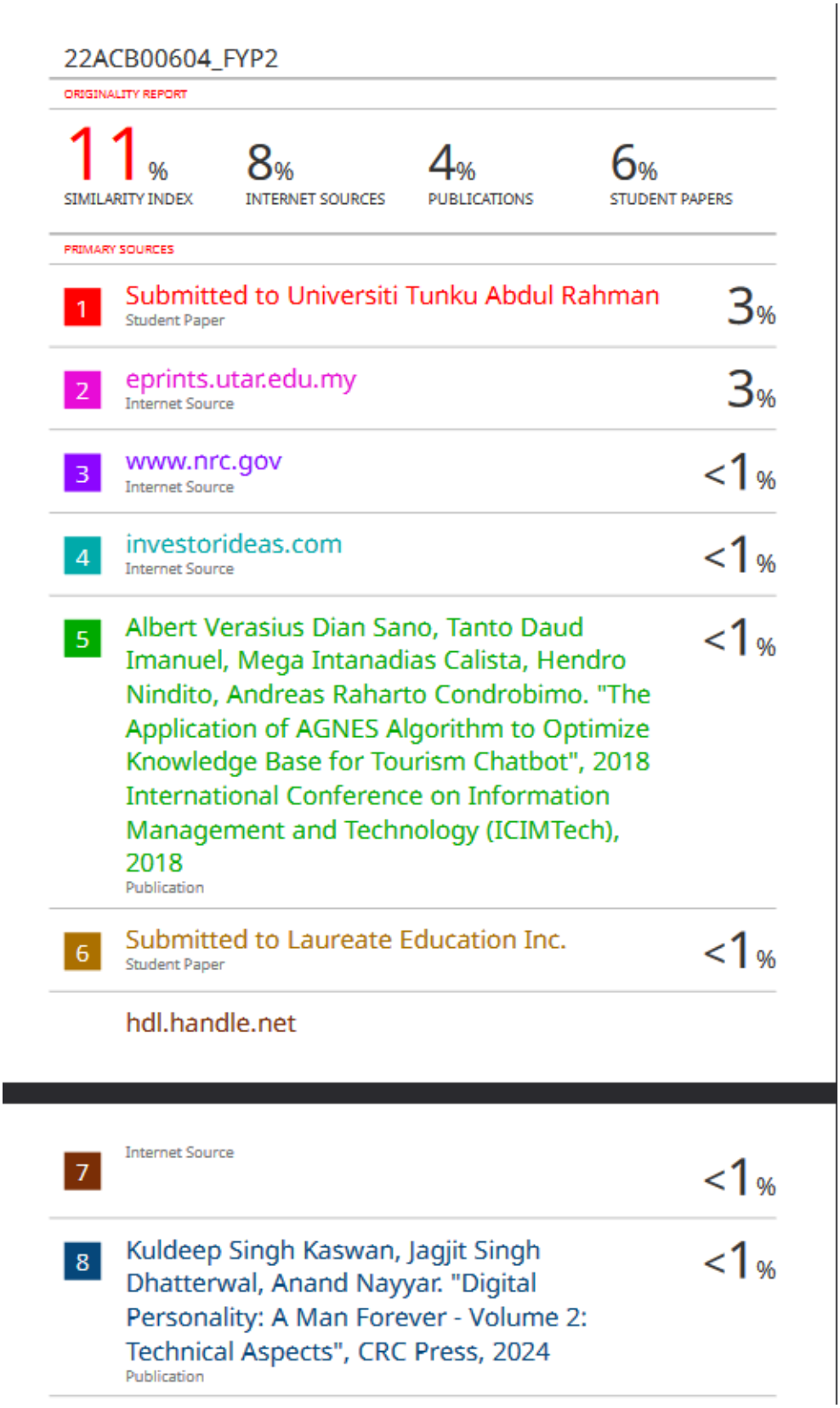
Supervisor's signature

Student's signature

POSTER


UNIVERSITI TUNKU ABDUL RAHMAN
CHUA FENG ZHAO
SUPERVISOR: PROF. TS. LIEW SOUNG YUE
PERSONALIZED MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM
DESCRIPTION
A chatbot focusing on Kampar tourism. This chatbot allows users to quickly ask for suggestions on where to go in Kampar and the areas around Kampar. From hotels, restaurants to fun attractions that can be experienced in and around Kampar.
This chatbot will also take user's traveling preference into account, as well as adapting to additional information such as the weather and time in Kampar and will provide suggestions based on those preferences, making the chatbot personalized for each user.
The chatbot uses OpenAI for its large language model, GPT-4o mini, in order to accurately understand user's request and determine the best location to suggest or information to provide to its user. It also contains additional functions to improve the user's traveling experience.
OBJECTIVES
To provide quick and efficient way for tourist to access Kampar's hard-to-access traveling information that is spread out across the Internet.
Develop a tourism chatbot that is aware of user's preferences and to cater to those preferences when providing suggestions.
Provide a 'command and answer' model where users will just have to ask questions and be given suggestions automatically.
METHODOLOGY AND HOW IT WORKS
Gather Kampar Tourism Data
Train and Feed Kampar Tourism Data into OpenAI Assistant
User states traveling preferences, passes additional information like weather and time in Kampar asks question.
OpenAI Assistant interprets question and relates with tourism data and traveling preferences.
OpenAI Assistant selects appropriate location or information for users.
Large language model (GPT-4o) formulates response and sends to user.
FEATURES
- Provide Suggestions
- Provide Information on Places in Kampar
- Provide Follow-Up Information
- Able to Understand Traveling Preferences
- Able to Understand Emotional Context
- Able to Adapt and Suggest Appropriately in Various Weather and Time conditions
- Update User Preference in Real-Time
- Able to Provide Directions
- Provide Users Ability to Make Reviews

PLAGIARISM CHECK RESULTS



22ACB00604_FYP2

ORIGINALITY REPORT

11%	8%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	3%
2	eprints.utar.edu.my Internet Source	3%
3	www.nrc.gov Internet Source	<1%
4	investorideas.com Internet Source	<1%
5	Albert Verasius Dian Sano, Tanto Daud Imanuel, Mega Intanadias Calista, Hendro Nindito, Andreas Raharto Condrobimo. "The Application of AGNES Algorithm to Optimize Knowledge Base for Tourism Chatbot", 2018 International Conference on Information Management and Technology (ICIMTech), 2018 Publication	<1%
6	Submitted to Laureate Education Inc. Student Paper	<1%
	hdl.handle.net	
7	Internet Source	<1%
8	Kuldeep Singh Kaswan, Jagjit Singh Dhatterwal, Anand Nayyar. "Digital Personality: A Man Forever - Volume 2: Technical Aspects", CRC Press, 2024 Publication	<1%

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

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chua Feng Zhao
ID Number(s)	22ACB00604
Programme / Course	CS
Title of Final Year Project	MOBILE TOUR GUIDE APPLICATION FOR KAMPAR TOURISM USING AN INTERACTIVE CHATBOT WITH GENERATIVE PRE-TRAINED LANGUAGE MODEL

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 11 % **Similarity by source** Internet Sources: 8 % Publications: 4 % Student Papers: 6 %	Within the required range.
Number of individual sources listed of more than 3% similarity: 0	Within the required range.
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: Liew Soung Yue

Date: 12/9/2024

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	22ACB00604
Student Name	Chua Feng Zhao
Supervisor Name	Prof. Ts. Liew Soung Yue

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

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

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

_____ _____

(Signature of Student)
Date: 8/9/2024