# EDE4002 – Bachelor Thesis
# Final Report

Joey Lim Jun Feng
2101483

# Electronics and Data Engineering

AY2024/2025

04 April 2025

Submitted as part of the requirements for EDE4002 Bachelor Thesis

## Declaration of Authorship

| | |
|---|---|
| **Name of candidate:** Joey Lim Jun Feng | Student ID Number: 2101483 |
| **Degree:** Electronics and Data Engineering | |
| **Project Title:** Leveraging Retrieval-Augmented Generation (RAG) and Large Language Models (LLM) for Chatbot Development | |
| **Academic Supervisors:** Tan See Teck | **Cluster:** Engineering |
| **Industry Supervisor:** *(if applicable)* | **Organization:** *(if applicable)* |

We hereby confirm that:

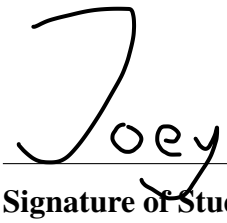
1. This work was done wholly or mainly while in candidature for a degree programme at SIT;
2. Where any part of this project has been previously submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. We have acknowledged the use of all resources in the preparation of this report;

4. The ~~**report contains**~~ **/ does not contain** patentable or confidential information *(strike through whichever does not apply)*;

5. The work was conducted in accordance with the research integrity policy and ethics standards of SIT and that the research data are presented honestly and without prejudice. The SIT Institutional Review Board (IRB) approval number is _____ *(where applicable)*;

6. We have read and understood the University's definition of plagiarism as stated in the SIT Academic Policies Scheme 14: Academic Integrity;

***Plagiarism is the copying, using or passing off another's work as one's own work without giving credit to the author or originator, and also includes self-plagiarism. For example, reusing, wholly or partially, one's previous work in another context without referencing its previous use.***



**Signature of Student**

04/04/2025

**Date**

# Contents

# 1 Abstract

This thesis presents the development of an intelligent chatbot designed to assist students with administrative and academic queries. As universities increasingly rely on digital platforms, automated solutions are essential for providing timely, accurate, and scalable responses. The proposed chatbot leverages LangChain for backend processing and integrates with Chainlit and Telegram as user interfaces. Retrieval-Augmented Generation (RAG) techniques enhance response accuracy by retrieving relevant information from the student handbook and other institutional documents.

A user study was conducted to evaluate the chatbot's effectiveness, usability, and impact on student experience. Participants provided feedback on functionality, accuracy, and ease of use, identifying key areas for improvement, such as response reliability, user guidance, and interface design. These insights informed refinements to the chatbot's workflow and features.

The final implementation delivers an AI-powered assistant capable of providing accurate, context-aware responses, improving accessibility to institutional information for students.

# 2 Introduction

This thesis aims to develop an intelligent chatbot to assist students with both administrative and academic queries. The chatbot is a practical application of Generative Artificial Intelligence (AI), specifically utilizing Large Language Models (LLMs). As illustrated in Fig. 1, the current school website requires users to navigate through multiple sections to locate information for various queries. In contrast, a chatbot will simplify this process by providing immediate and relevant answers, eliminating the need to manually search through numerous web pages. With the increasing reliance on digital platforms, universities require automated solutions that deliver timely, accurate, and scalable responses. This streamlined approach to information access reduces both time and effort, significantly enhancing the overall user experience.

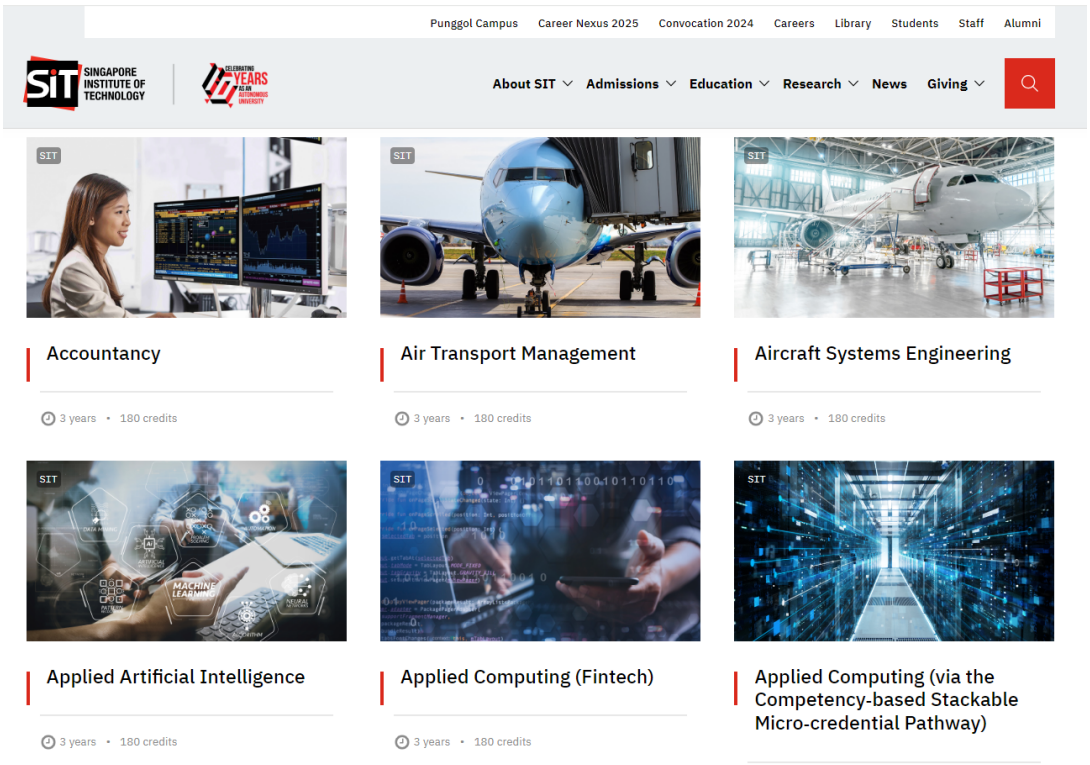



Figure 1: SIT Portal [1]

# 3 Literature Research

## 3.1 Transformer Architecture

A key relevant literature on Large Language Models (LLMs) is a paper titled "Attention Is All You Need," written by researchers at Google Brain [2]. This paper introduces the Transformer model, which consists of two main components: an encoder and a decoder.

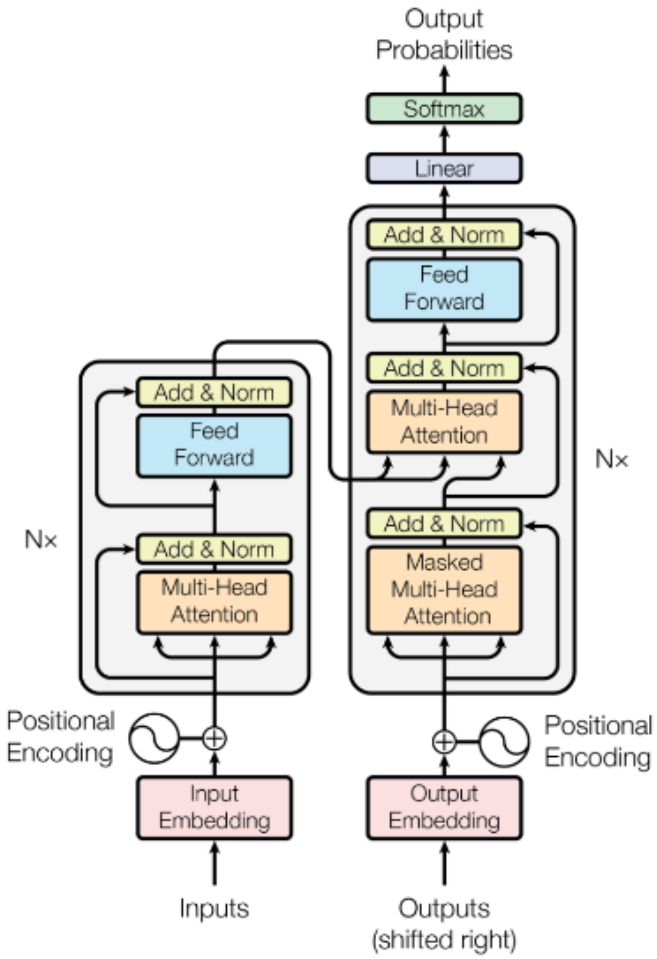



Figure 2: Transformer Model [2]

The encoder processes the input text by embedding sentences and capturing contextual relationships between words through self-attention and positional encoding mechanisms. Sentence embeddings are a fundamental concept in LLMs that enable the model to represent the meaning of sentences numerically. As illustrated in Fig. 3, the process begins when a sentence is first broken into smaller units of text called tokens, determined using a predefined vocabulary. Each token is then converted into a numerical vector through the embedding process. To generate a sentence embedding, the individual token embeddings are combined using a method called pooling. This final sentence embedding serves as a numerical representation of the entire sentence, enabling the LLM to understand the overall meaning of the sentence mathematically.

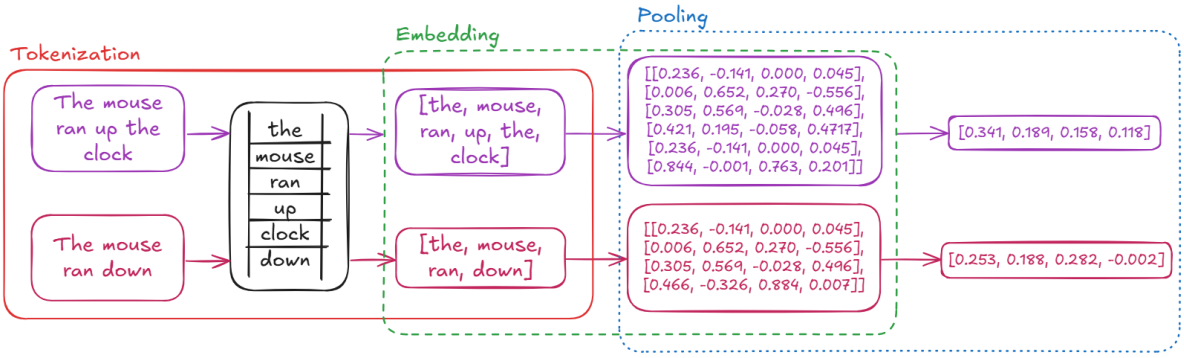



Figure 3: Sentence Embeddings Process

The decoder, on the other hand, uses these embeddings to generate output sequences by predicting the next word in a sequence. It does so by selecting the most probable word from its pre-trained vocabulary, based on patterns and relationships it learned during the training process. The decoder decisions is influenced by both the encoder's output, which provides context from the input sequence, and its own previously generated tokens. This interplay between the encoder and decoder allows the Transformer to excel in tasks such as translation and summarization, where generating coherent and contextually accurate text is essential.

A Generative Pre-trained Transformer (GPT) is a specific implementation of Large Language Models (LLMs) that uses a decoder-only architecture. For this chatbot, an OpenAI GPT model is used as the LLM. Examples of GPT-based models include ChatGPT from OpenAI and LLaMA from Meta (formerly Facebook).

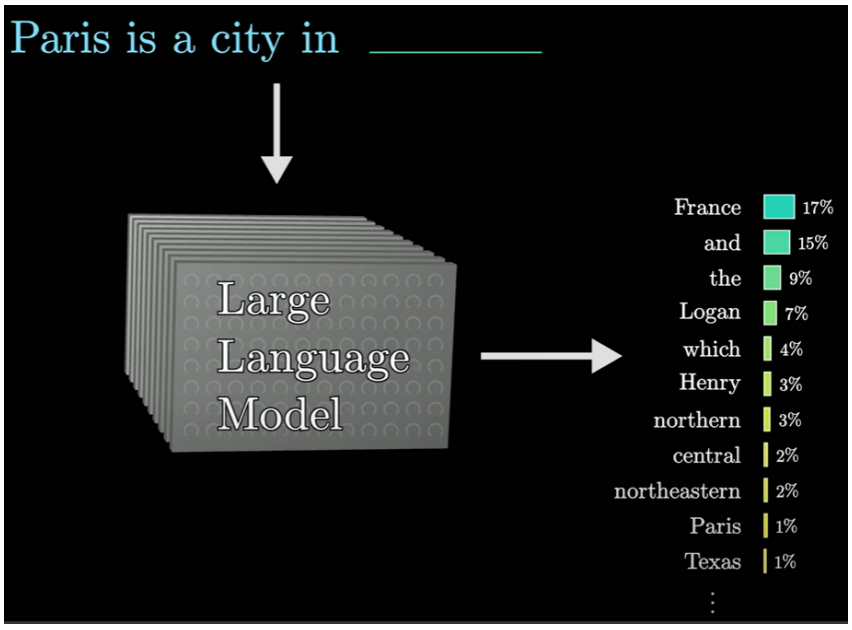



Figure 4: Text Generation [3]

## 3.2 "Hallucinations" in LLMs

A significant challenge encountered by chatbots is the phenomenon of "hallucinations". Since a GPT model is trained on large amounts of data, it occasionally generates information that is not grounded in input data or real-world knowledge, leading to outputs that may be misleading or entirely incorrect [4]. To address this issue, various methods have been explored to mitigate hallucinations and enhance the reliability of chatbot responses.

- **Retrieval-Augmented Generation (RAG)**: This method combines the generative abilities of LLMs with a retrieval system that pulls relevant information from a vector database. By grounding responses in factual data, RAG significantly reduces hallucinations and improves the accuracy of the model's outputs [5].
- **Mixture of Experts (MoE)**: The Mixture of Experts method utilizes multiple specialized expert models, where each model focuses on a specific aspect of language understanding. These expert models collaborate to generate a response, resulting in higher performance and efficiency compared to a single-expert model [4].
- **Fine-Tuning**: Fine-tuning involves training a pretrained model on a task-specific dataset to optimize its performance for a particular application. The pretrained model serves as the base, and additional domain-specific data is used to further train it. Unlike the previous two methods, fine-tuning does not incorporate a retrieval mechanism but rather adapts the pre-trained model to create a specialized version [6].

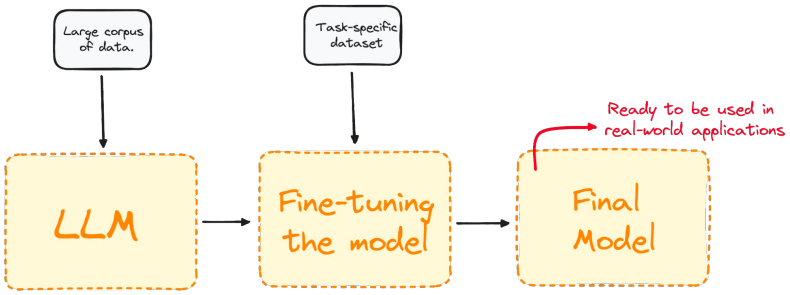



Figure 5: Visualizing the Fine-Tuning Process [6]

In this paper, the Retrieval-Augmented Generation (RAG) method will be used, and a more in-depth explanation will be provided in the following section.

## 3.3 Detailed Explanation on RAG

As previously explained in the section on encoders, sentences can be represented as vectors. These vectors encode the semantic meaning of sentences within a high-dimensional space. To visualize them on a 2D plane, dimensionality reduction techniques such as PCA (Principal Component Analysis) can be applied. The core idea of RAG is to embed the user's query as a vector and use the cosine similarity formula (shown in Fig. 6) to identify the contents most relevant to the query. The cosine similarity value ranges between -1 and +1: -1 refers to the vectors point in opposite directions, indicating complete dissimilarity, 0 refers to the vectors being orthogonal (90° angle), indicating no similarity and +1 refers to the vectors are perfectly aligned (same direction), meaning they are highly similar.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 6: Cosine Similarity Formula

Where **A** & **B** are vectors, $A_i$ is an individual value in vector **A**, $B_i$ is an individual value in vector **B** and $n$ is the number of elements in the vectors.

Once the cosine similarity between the query vector and document vectors are computed, the vectors that are determined to be the most similar are retrieved as context. As illustrated in Fig. 7, Document 1 and Document 2 are the vectors most similar to the query vector. Therefore, only these two documents are retrieved and provided to the LLM, enabling it to generate a more accurate and contextually relevant response for the user.

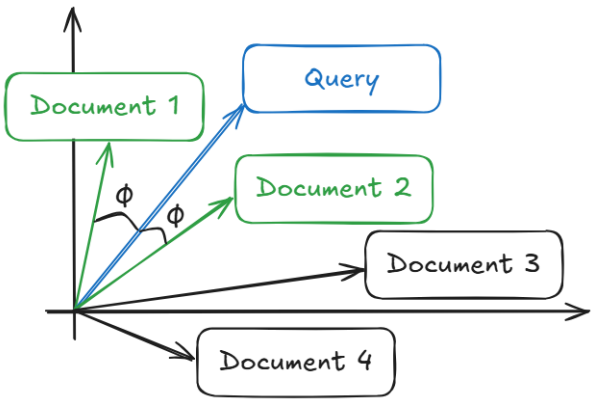



Figure 7: Vector Similarity in RAG

# 4 Problem Statement and Proposed Solution

## 4.1 Problem Statement

The primary problem addressed in this thesis is the difficulty students face when navigating the school portal for administrative and academic queries. The current school portal requires users to manually search through multiple sections of the webpage to find relevant information. This process can lead to confusion and delays in locating critical resources such as course information, scholarship details, or academic schedules.

Additionally, the availability of the school help desk is often limited by operating hours and staff constraints, meaning that students may not always receive timely assistance when it is needed. During off-hours or peak periods, students must rely solely on the website, which does not always provide efficient or direct answers to their queries.

These challenges result in a poor user experience, increased administrative workload, and frustration among students. Therefore, the lack of an automated, intelligent system to assist students in retrieving relevant information from the portal presents a significant issue that this thesis aims to resolve.

Research has demonstrated that chatbots are particularly valued for their ability to provide instant responses, 24/7, reducing the need for users to manually navigate through complex websites or wait for helpdesk support. A qualitative study has also indicated that chatbots can save businesses 30% on customer support costs, while improving the overall user experience by reducing response times by 80% [7].

## 4.2 Proposed Solution

The solution to the problem of inefficient information retrieval in the school portal is to implement a conversational chatbot. This chatbot will utilize a response agent capable of dynamically determining how to handle various user queries by leveraging tools such as retrieving data from a local database or performing Google searches restricted to SIT's website. The chatbot will provide relevant content to effectively address each query and will be accessible via a webpage or a Telegram bot.

# 5 Methodology

## 5.1 Overview

The diagram below (Fig. 8) provides an overview of the chatbot's infrastructure. Chainlit and Telegram serve as the user interfaces, facilitating interactions with users. Langchain acts as the orchestration tool and backend of the chatbot while Redis is employed as the database. A detailed explanation of each component and its usage will be provided in the subsequent sections of this report.


Figure 8: Overview Workflow

## 5.2 Redis Workflow

The data for this project is manually gathered from various sections of SIT's website and stored into two formats: JavaScript Object Notation (JSON) files and Portable Document Format (PDF) documents. Some examples of JSON data include the contact information for career coaches from the Centre for Career Readiness (CCR), URL links for various courses, and data extracted from the General Employment Survey. These JSON files were manually created by extracting relevant information from website links. Each JSON object generally contains two parameters: the category to which the content is related and the specific content associated with it. Additionally, the PDF documents used in this project include SIT's matriculation guide, the Rules and Regulations of SIT, and the student handbook prepared for the school's open house. These information serve as the foundation for the chatbot's knowledge base, enabling it to provide accurate and contextually relevant responses to user queries.

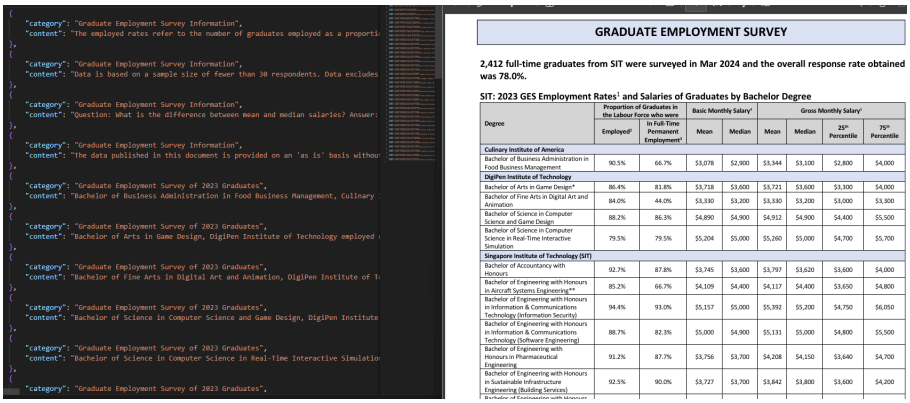



Figure 9: Sample of GES Json Data

### 5.2.1 Embedding Model Selection

The embedding model used in this chatbot is "text-embedding-3-small". The model is accessed via an Application Programming Interface (API) key provided by OpenAI's API platform. An API key serves as a unique identifier and authentication token, allowing the chatbot to securely interact with OpenAI's servers, granting access to the ChatGPT API to generate embeddings, process user queries and generate responses.

The "text-embedding-3-small" embedding model supports a maximum input of 8191 tokens and costs approximately 62,500 pages (800 tokens per page) per dollar, which is significantly cheaper than "text-embedding-3-large", which processes 9,615 pages per dollar [8]. Although its performance is only 1.3% lower than "text-embedding-3-large" on the MTEB evaluation dataset, the model was selected for its balance of performance and efficiency, making it ideal for tasks requiring high-quality text embeddings with minimal resource demands.

### 5.2.2 Data Storage Process

The PDF and JSON files are processed using a Jupyter Notebook created to embed the files and store them efficiently in the vector store. Redis is used as the database due to its ease of facilitating cloud storage. Additionally, Redis provides an application that offers a seamless view of the stored data, enabling users to easily manipulate and manage the database.

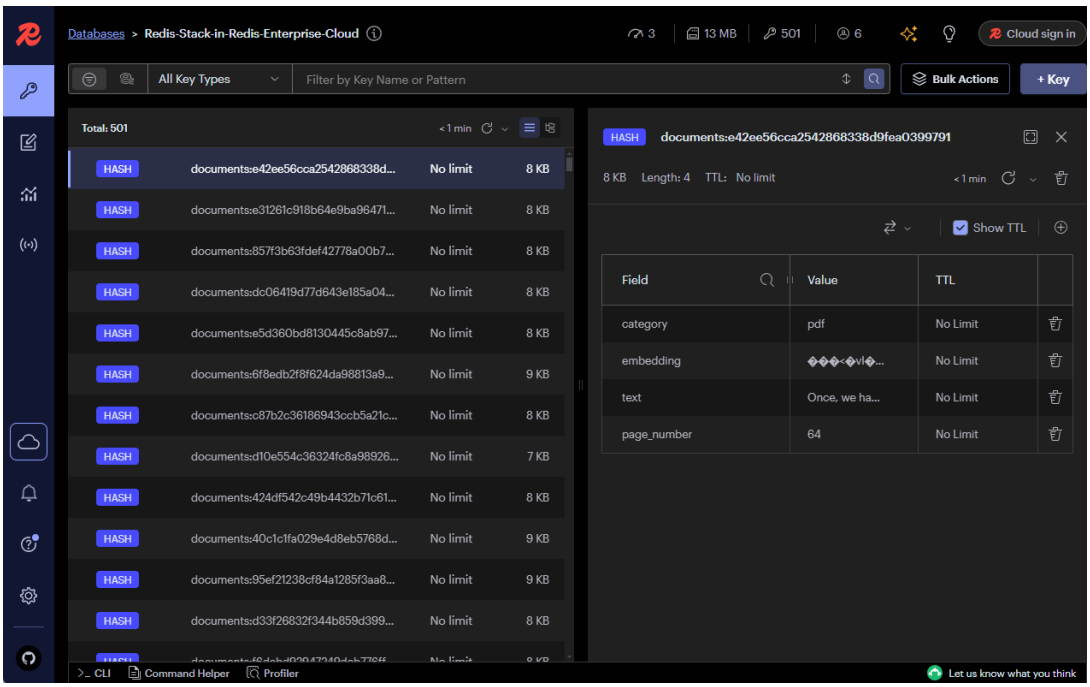



Figure 10: Redis Database Application

### 5.2.3 Storage of JSON files

The process carried out by the notebook to store the JSON file is visualized in Fig. 11. The first step involves reading the JSON file from the directory and accessing its contents. The second step extracts the content of a single JSON object. Lastly, the content is embedded using the text-embedding-3-small model and is a assigned to a unique ID in the Redis vector store. As it is being stored, the metadata, including the category of the embedded content and the content itself, is stored under the same ID. This process is repeated for all JSON objects present in the JSON file.

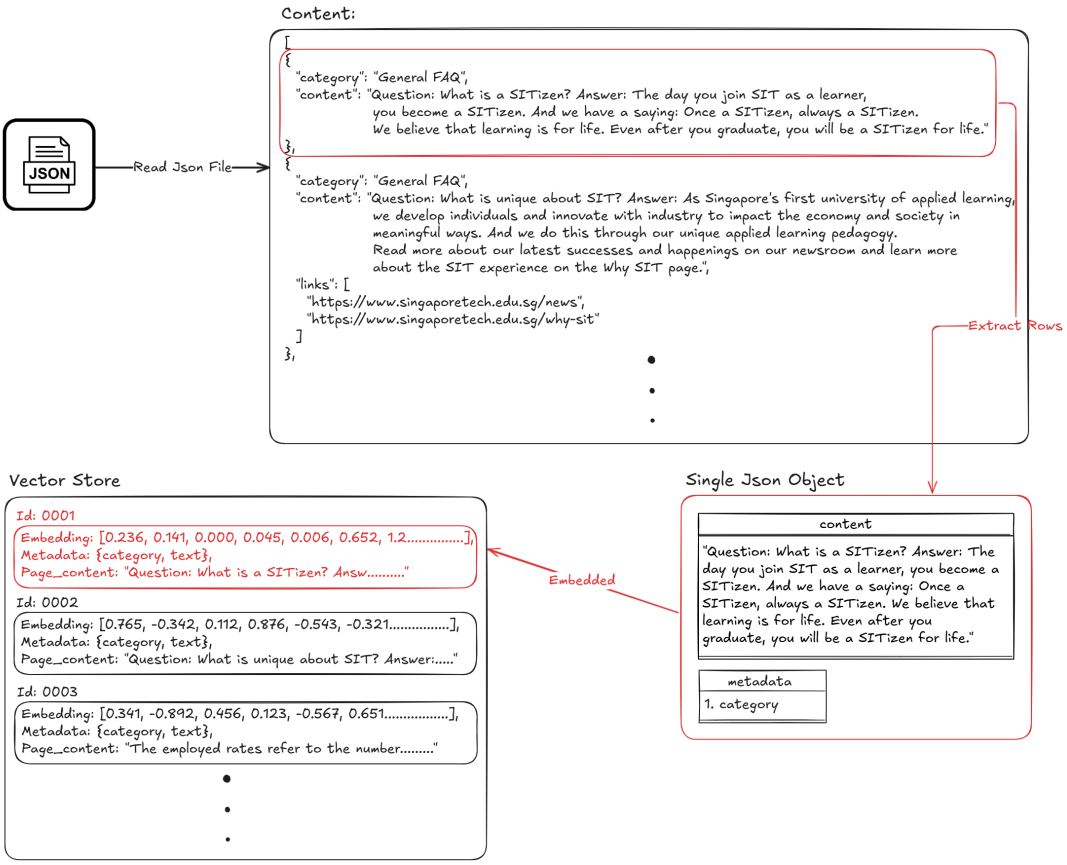



Figure 11: Json Handler

### 5.2.4 Storage of PDF files

The process for storing PDF files is visualized in Fig. 12. The first step involves extracting all the text from the PDF file. In the second step, the text is divided into multiple chunks, with each chunk containing a maximum of 1,000 characters and an overlap of 200 characters. The overlapping of chunks, highlighted in light yellow in the diagram, ensures that the context of the text is preserved as much as possible. In the third step, each individual chunk is embedded using the same embedding model used for the JSON files. The embeddings, page content, and metadata, including the category and page numbers, are then stored as a single document in the Redis vector store.

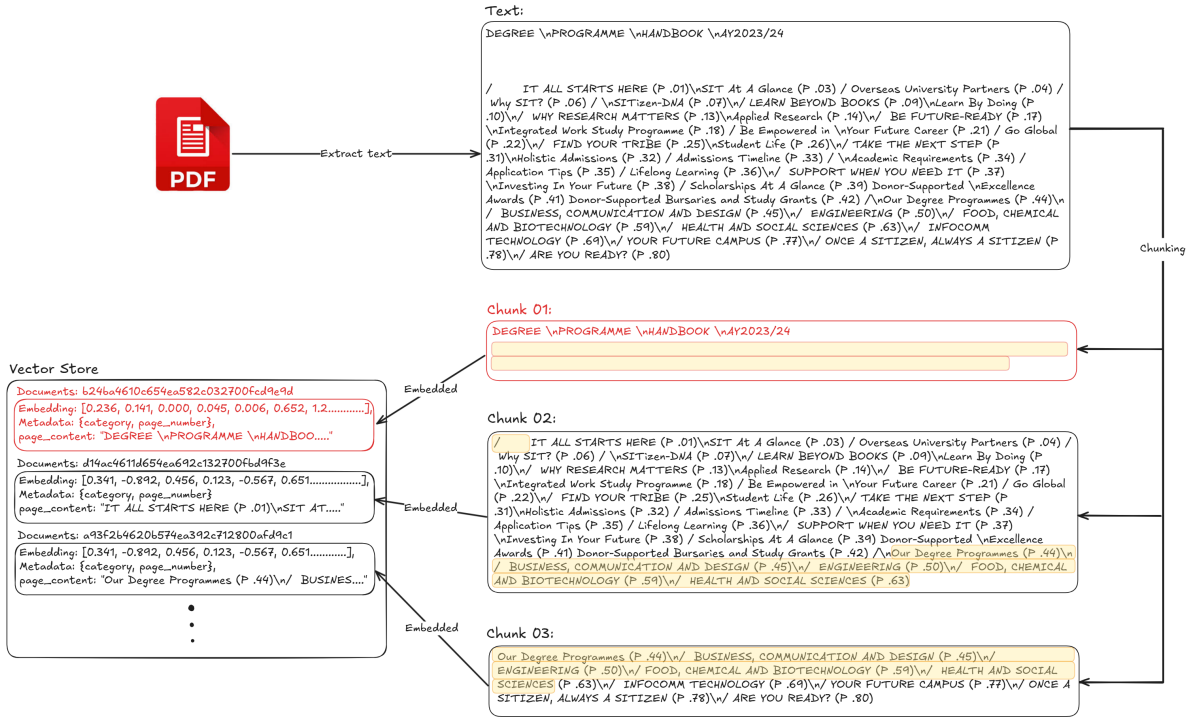



Figure 12: PDF Handler

## 5.3 Chainlit Workflow

This section describes how Chainlit has been integrated into the chatbot. The diagram in Fig. 13 illustrates this flow, showing how different decorators are used in Chainlit and how they interact to manage the chatbot's conversation flow.

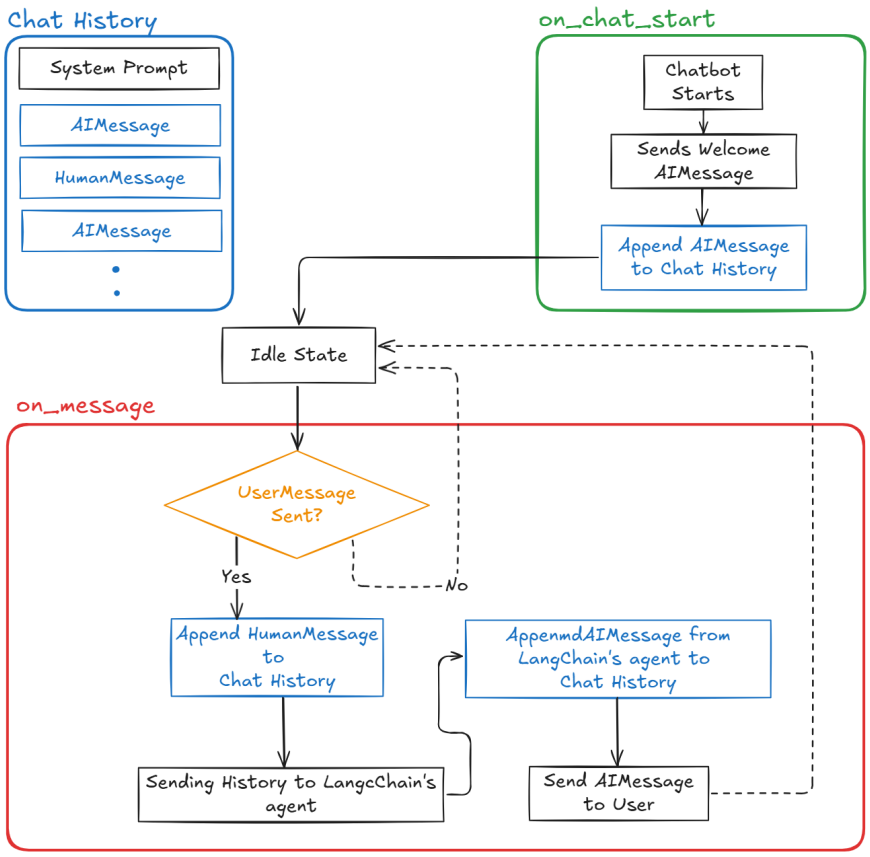



Figure 13: Chainlit Logic Flow

The following subsections provide a detailed explanation of each decorator.

1. **"on_chat_start"**: This decorator initializes the chatbot when a session begins. It starts the chatbot, sends a welcome message to the user, appends the AI's welcome message to the chat history, and moves the chatbot into an Idle State where it waits for a user input.

2. **"on_message"**: This decorator handles incoming user messages and processes them sequentially. When the user sends a message, it is appended to the chat history as a HumanMessage. The chatbot then sends the updated conversation history to LangChain's agent for processing. It then sends the generated response back to the user and moves back to the Idle State, waiting for the next interaction.

For every message sent by the chatbot or received from the user, these messages are appended and stored in the chatbot's chat history, as shown in the blue boxes in Fig. 13. The chat history ensures that the chatbot maintains the context of the conversation, allowing it to send the complete context to the LLM for generating more accurate and coherent responses. In cases where the chat history is not preserved, the chatbot will lose the conversation's context.

For example, if a user says, "Hi, my name is Joey!" and later asks, "What is my name?", the chatbot will be unable to provide an answer because the earlier message was not appended to the chat history. By storing the conversation history, the chatbot can retain context and deliver better responses during ongoing interactions.

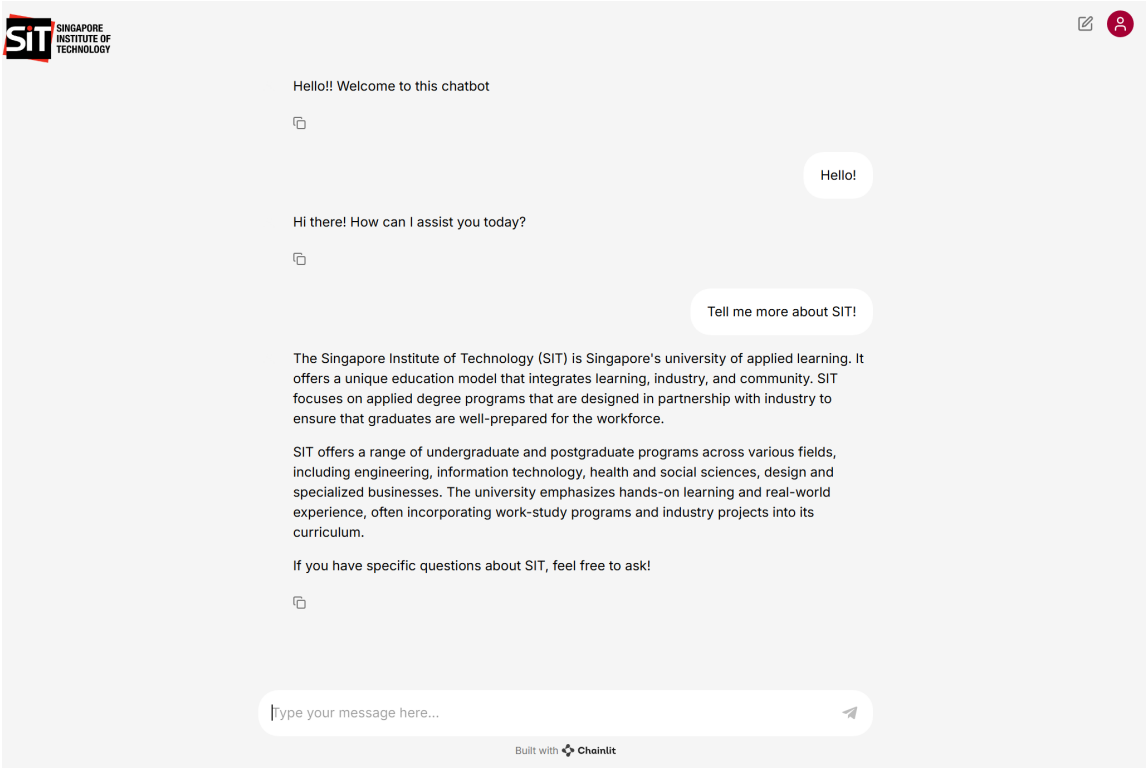



Figure 14: Chainlit Interface

## 5.4 Telegram Workflow

This section describes how Telegram has been integrated into the chatbot. The diagram in Fig. 15 illustrates this flow, showing how different decorators are used in the Telegram Chatbot and how they interact to manage the chatbot's conversation flow. The following subsections provide a detailed explanation of each decorator.

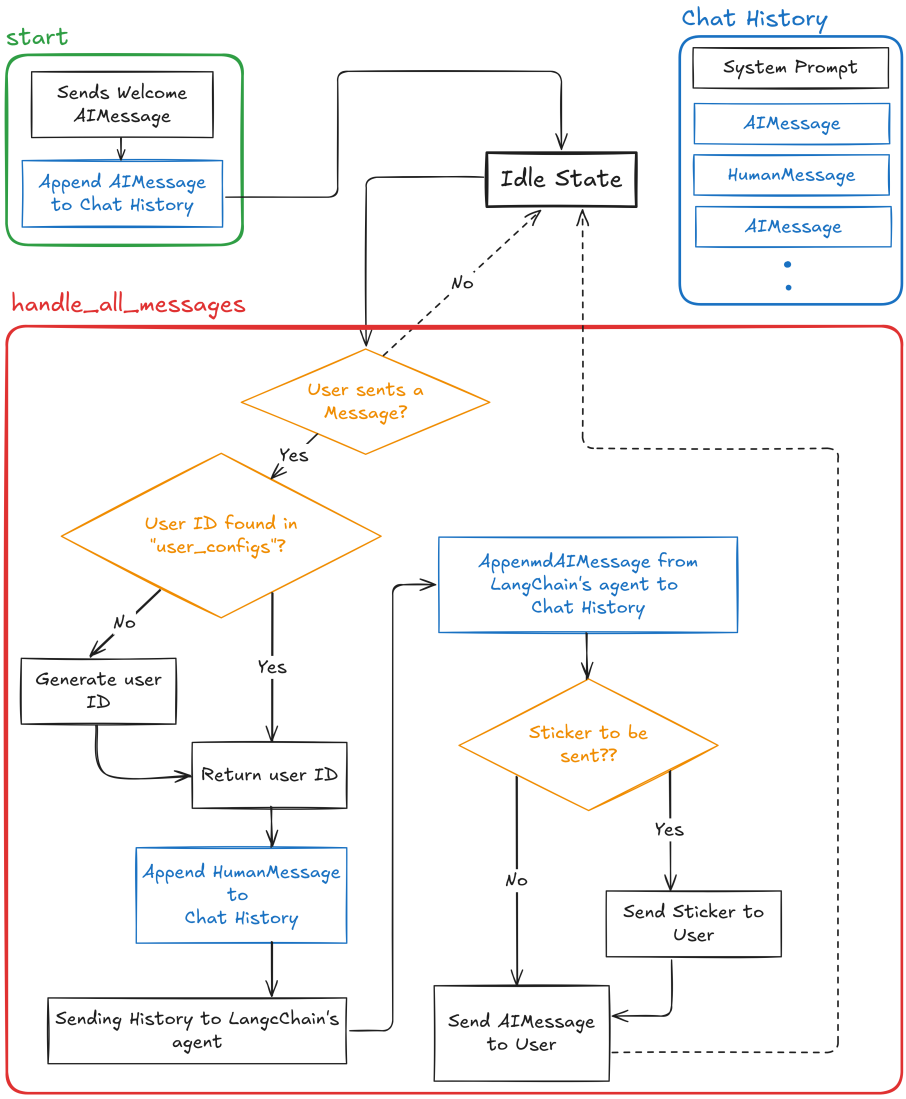



Figure 15: Telegram Logic Flow

The following subsections provide a detailed explanation of each decorator.

1. **"start"**: Similarly to Chainlit's "on_chat_start" decorator, this function initializes the chatbot at the start of a session. It launches the chatbot, sends a welcome message to the user, appends the AI's message to the chat history, and moves into an Idle State where it waits for a user input.

2. **"handle_all_messages"**: Similarly to Chainlit's "on_message" decorator, this decorator processes all incoming user messages and dynamically manages the conversation flow. When a user sends a message, the chatbot first checks for the User ID to determine which user to reply to. It appends the user's message to the chat history before sending it to LangChain's agent for processing. The agent's response is then appended to the chat history. Following this, the chatbot determines whether a Telegram sticker should be included in the response. If not, only a text message is sent to the user. The chatbot then returns back to the Idle State, awaiting further user interaction. The function for sending stickers will be explained later in this section.

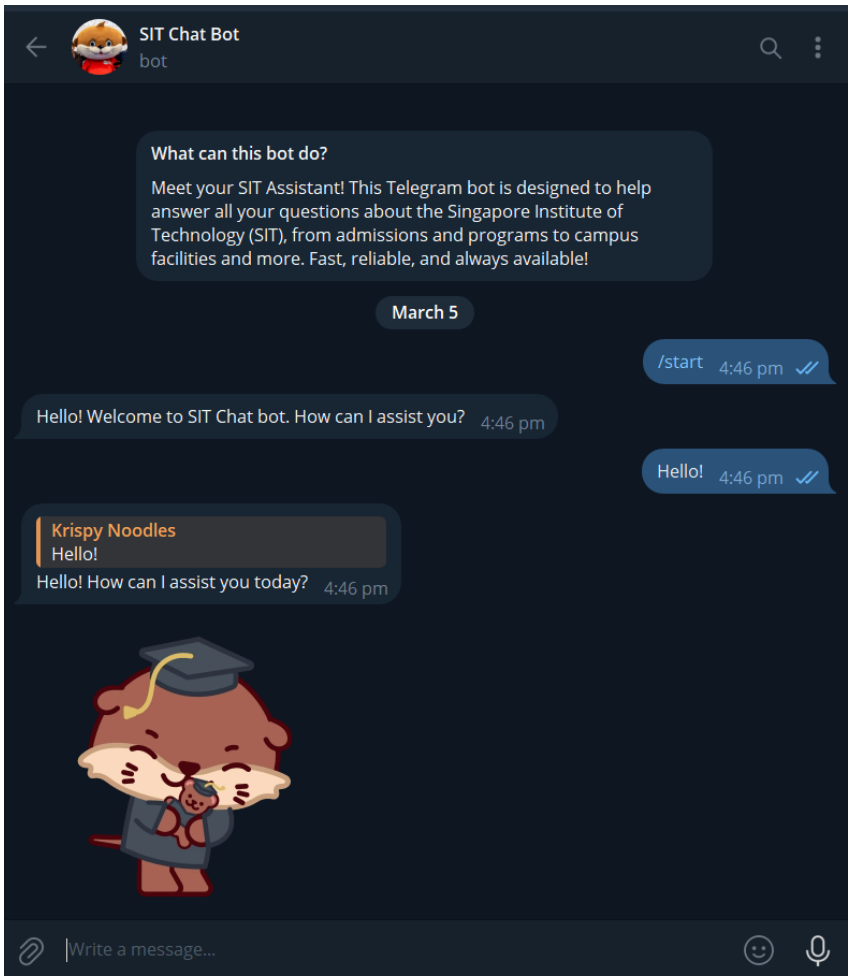



Figure 16: Telegram Interface

The Telegram bot is accessed via an API key generated using another telegram bot called "BotFather", an official chatbot maintained by Telegram for managing and creating new bots. Through "BotFather", developers can create a new bot, obtain the unique API key, and configure various bot settings such as the name, description, and commands [9].

### 5.4.1 Telegram Sticker Function

This section explains how the Telegram chatbot determines when to send stickers to users. The figure below, Fig. 17 presents a diagram illustrating how the function operates.

As the chatbot uses the response generated by the LLM to respond back to the user. It utilizes another LLM with a system prompt instructing it to categorize the response. If the category returned by the LLM matches an entry in the sticker dataframe, the corresponding sticker is sent to the user. This ensures that not every chatbot response includes a sticker, making interactions feel more human-like.

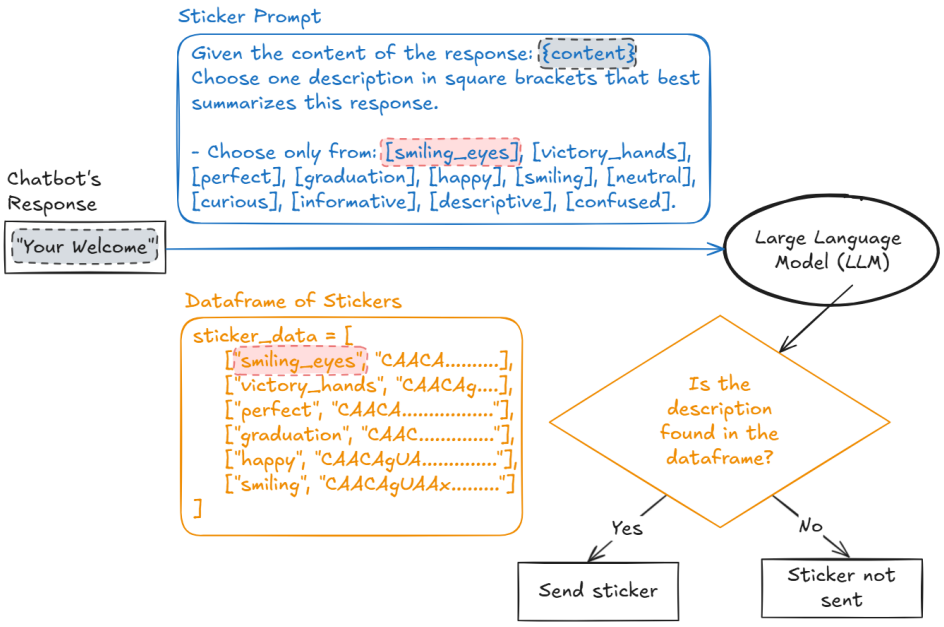



Figure 17: Sticker Generator Function

As shown in Fig. 17, when the chatbot generates the response "Your Welcome", the LLM classifies it under the label "smiling_eyes". The chatbot then searches the sticker dataframe for a matching entry and retrieves the corresponding sticker. The sticker's ID is then sent to Telegram, which delivers it to the user in a sticker format

## 5.5 LangChain Workflow

This section explains how LangChain is used to manage interactions on the backend of the chatbot. The chatbot uses LangChain's default agent, the "ReAct agent", which stands for "Reasoning and Action Agent". The agent can dynamically decide which tools to use from a toolbox, which consists of "redis_tool" and "google_tool". The ReAct agent uses an LLM to understand each tool's functionality based on its description.

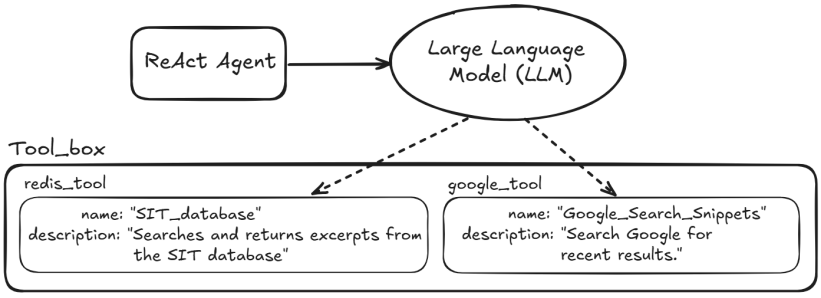



Figure 18: Toolbox of ReAct Agent

### 5.5.1 Redis Retriever Tool

The Redis retriever tool, referred to in the implementation as "redis_tool", is designed to fetch content from the database, which consists of previously stored JSON and PDF files. This tool retrieves the top five most relevant pieces of content based on the user's query. It uses a similarity-based search mechanism to identify the closest matches and feeds this content into the LLM to generate an accurate response.

### 5.5.2 Google Search Tool

The Google Search tool, referred to in the implementation as "google_tool", is designed to retrieve relevant information from Google when the required information is not available in the database. This tool leverages the Programmable Search Engine, a search engine API service provided by Google. To ensure that only SIT-related information is retrieved, the search is restricted to websites within the SIT domain.

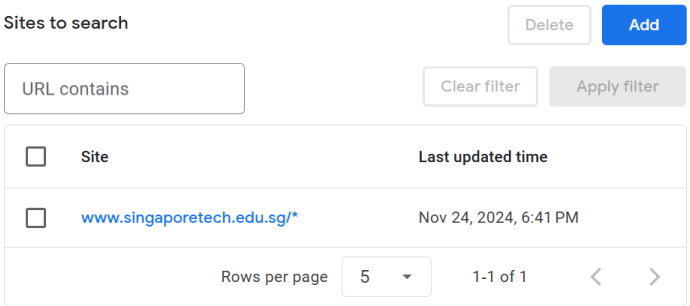



Figure 19: "Sites to search" of Programmable Search Engine

### 5.5.3 Retrieval Process

The "redis_tool" and "google_tool" are both retrieval tools. An example of how the ReAct agent uses these retrieval tools is shown below in Fig. 20.

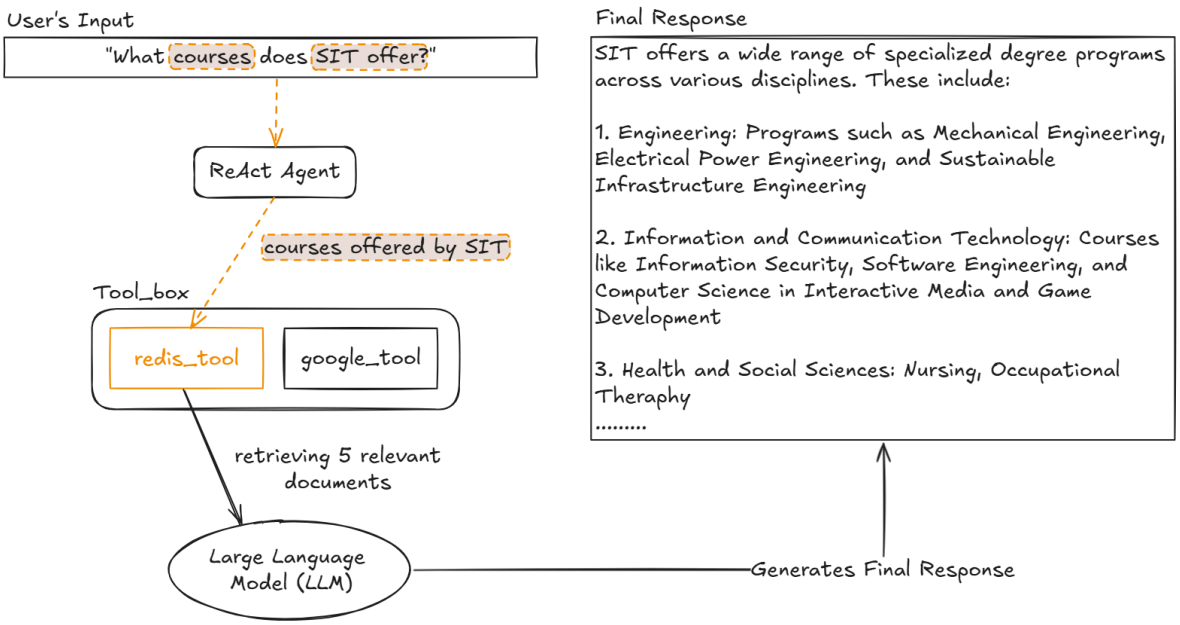



Figure 20: Example of Retrieval Process

The ReAct agent first decides which part of the user's input it wishes to use in the retrieval process. In this example, the words "courses" and "SIT offer" are selected and rephrased into "courses offered by SIT". It then uses this rephrased sentence in the "redis_tool" to search for similar documents. In some cases, the ReAct agent is able to call multiple tools on its own to answer a single query. As previously discussed in the RAG section, the agent embeds the rephrased sentence and applies cosine similarity to retrieve the top five most relevant documents from the database. These retrieved documents are then fed as context into the LLM along with the user's query to generate a final response to the user.

### 5.5.4 Large Language Model (LLM) Selection

The LLM model used in this chatbot is "GPT-4o", accessed via an API key provided by OpenAI's API platform. The "GPT-4o" model supports a maximum input length of 128K tokens, allowing it to process up to approximately 213 pages (assuming 800 tokens per page) in a single input. It is priced at $2.50 per 1 million input tokens and $10.00 per 1 million output tokens. The model's knowledge is trained up to October 2023, and it delivers responses three times faster than reasoning models such as "o1-preview" and "o3-mini". Therefore, "GPT-4o" is the most suitable choice for this chatbot, given its performance, relevance, and cost efficiency [10].

### 5.5.5 System Prompt

A key component in generating responses with the Large Language Model (LLM) through LangChain is the system prompt. The system prompt provides explicit instructions that guide the LLM in selecting the appropriate tool based on the user's query. It is typically written in text form and consists of a few hundred lines explicitly guiding the LLM in determining which tool to use based on the user's query.

An effective technique is using a decision tree in the system prompt. This helps the LLM determine whether the query should be answered from the Redis database. As shown in Fig. 21, the decision logic first checks whether the user's question can be found in the SIT Redis database. If so, the "redis_tool" searches for relevant content. Otherwise, the system evaluates whether the query is related to SIT. For queries related to SIT, the "google_tool" searches SIT's website for relevant information. If no relevant information is retrieved, the chatbot relies on the pre-trained knowledge of the OpenAI LLM to generate a response.

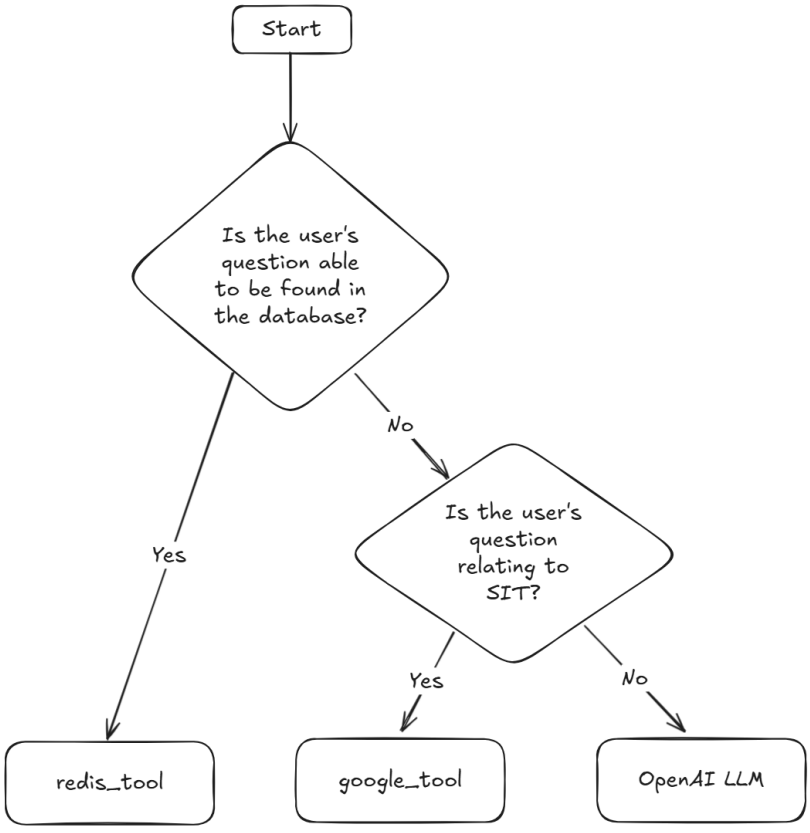



Figure 21: Pictorial Representation of Decision Flow

The system prompt is incorporated into the chat history, as shown in the Chainlit and Telegram Workflow section and the chat history is then fed into the LLM to generate a response.

# 6 Evaluation & Results

This section discusses the performance of the chatbot and the feedback gathered from users. A quantitative and qualitative user study was conducted with 23 participants to evaluate the chatbot's usability, efficiency, and user satisfaction. The Chainlit version of the chatbot was used by all participants to maintain consistency in evaluation.

## 6.1 User Study Process

The overall structure of the user study, includes "Pre-User Study Questions", two "User Journeys", and "Post-User Study Questions", as illustrated in Fig. 22.

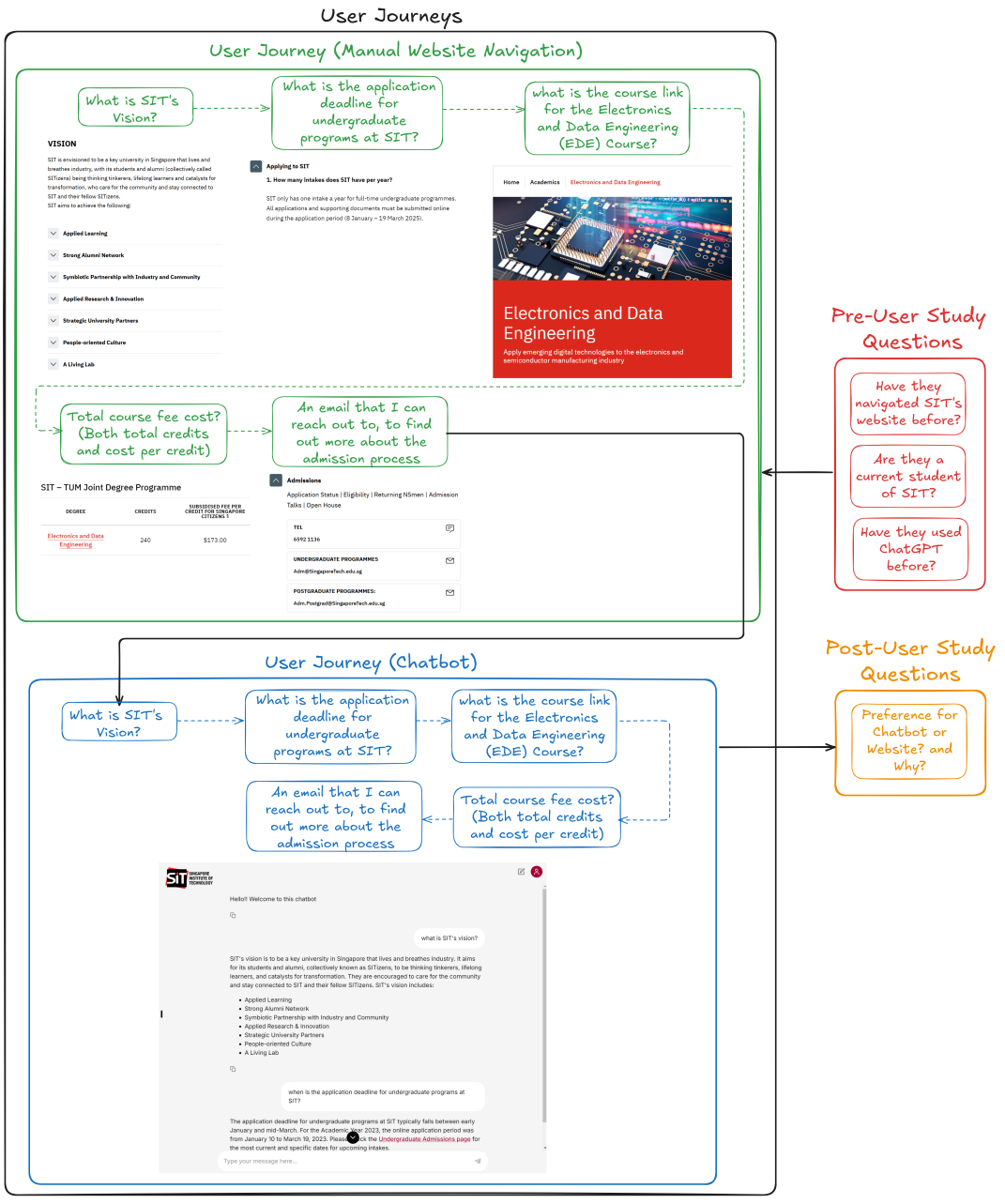



Figure 22: Overall Structure of User Study

### 6.1.1 Pre-User Study Questions

The user study begins by assessing the participants' familiarity with SIT's website and whether they are SIT students. This ensures that the results are not biased toward participants who are SIT students or those who have previously visited SIT's website before. Participants are then asked if they have used ChatGPT before to confirm their understanding on basic chatbot capabilities and interaction methods. A talk by Canva at SIGGRAPH Asia 2024 revealed that when chatbots were first introduced, some users mistakenly perceived them as questionnaires rather than interactive assistants. Therefore, determining participants' familiarity with ChatGPT is crucial in ensuring meaningful interactions [11].

### 6.1.2 User Journeys

Participants go through two different user journeys, each consisting of five questions. These questions simulate the inquiries of a student interested in applying to SIT. Participants first search for these answers manually using SIT's website and then attempt to find the same answers using the developed chatbot. The time taken for both methods is recorded, along with the number of clicks required to navigate the website and the number of prompts used when interacting with the chatbot.

### 6.1.3 Post-User Study Question

The study concludes by asking participants which medium do they prefer and their reasoning behind their choice.

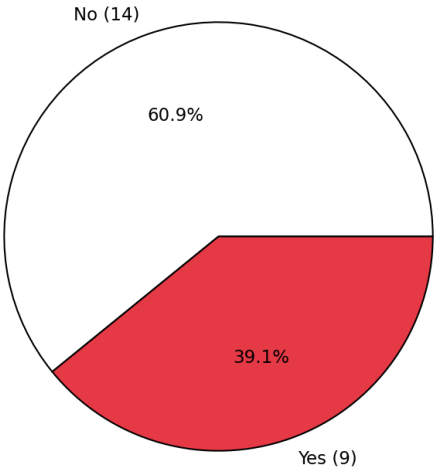
Each individual conversation, along with noteworthy interactions, is documented for further analysis. All collected data will be accessible through this Google Drive link.

## 6.2 Results

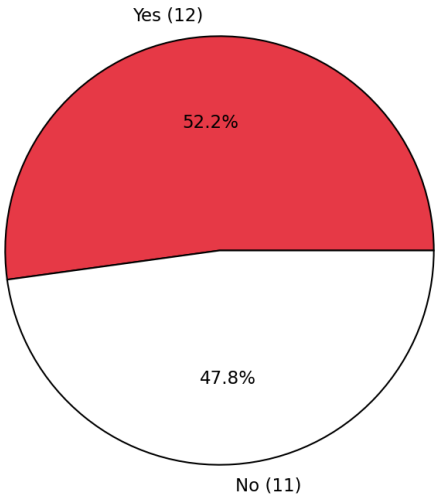
This section discusses the findings gathered from the user study conducted with 23 participants.

### 6.2.1 Pre-User Study Questions

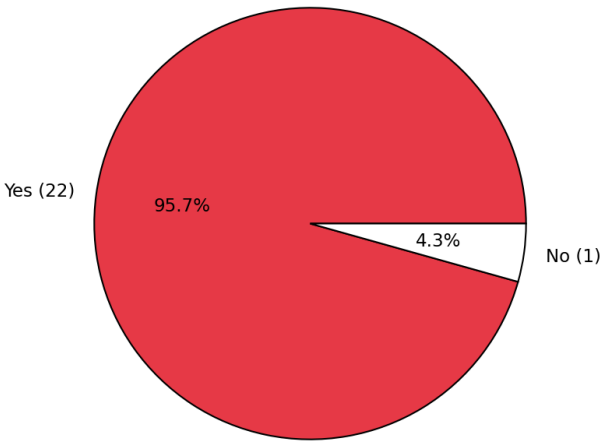
The results of the Pre-User Study, shown in Fig. 23a, indicate that approximately 60% of participants were unfamiliar with SIT's website, despite an even mix of SIT and non-SIT students. However, this was not a significant issue, as all participants had prior experience navigating websites and searching for information across different pages. One participant, who had never used ChatGPT before shown in Fig. 23c, received a brief introduction to basic chatbot interactions before proceeding with the user study.




(a) Have the participants visited SIT's website before?




(b) Are the participants SIT students?




(c) Have the participants used ChatGPT before?

Figure 23: Pre-User Study Questions

### 6.2.2 User Journeys

For each of the five questions, a time comparison was conducted between the two mediums: navigating the website manually and using a chatbot. The average time taken for each medium was also plotted. Additionally, a comparison was made across all participants based on the number of pages navigated and the number of prompts required before obtaining an answer.

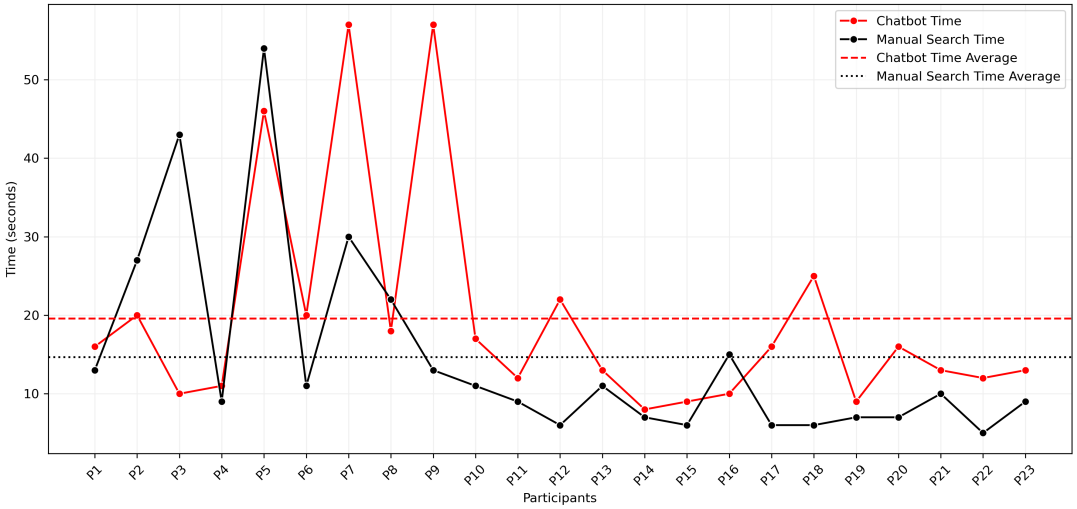



Figure 24: Comparison of Manual Time and Chatbot Search Time - Question 1

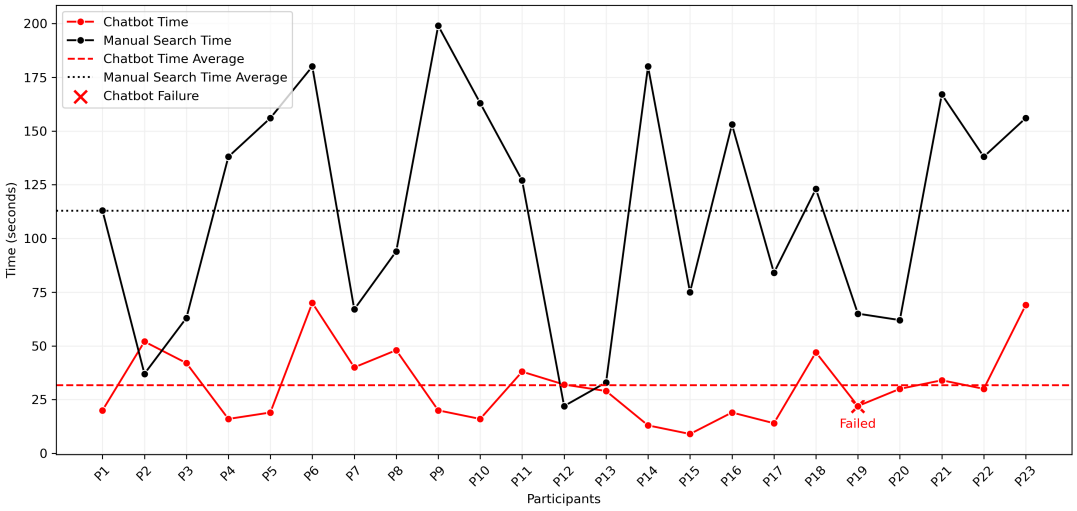



Figure 25: Comparison of Manual Time and Chatbot Search Time - Question 2

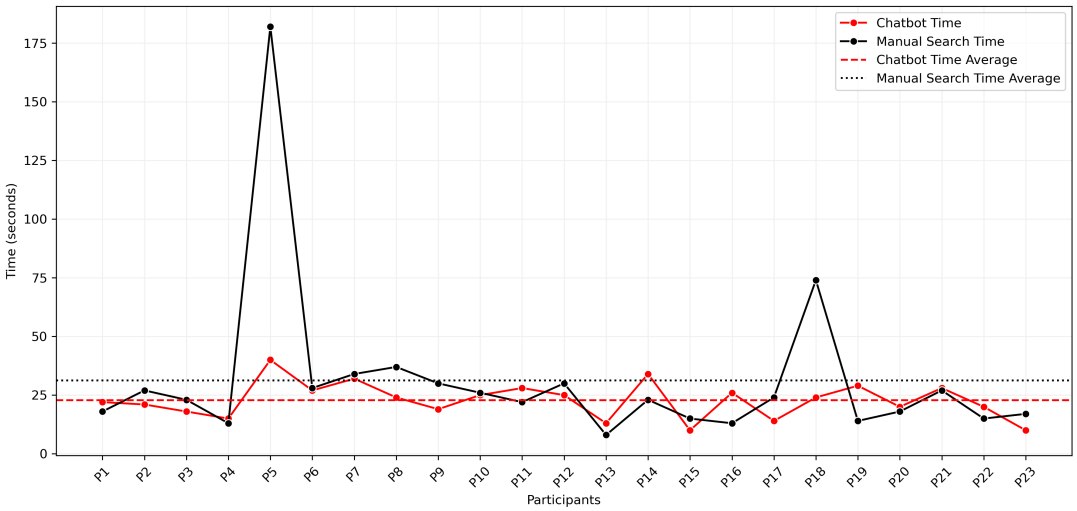



Figure 26: Comparison of Manual Time and Chatbot Search Time - Question 3

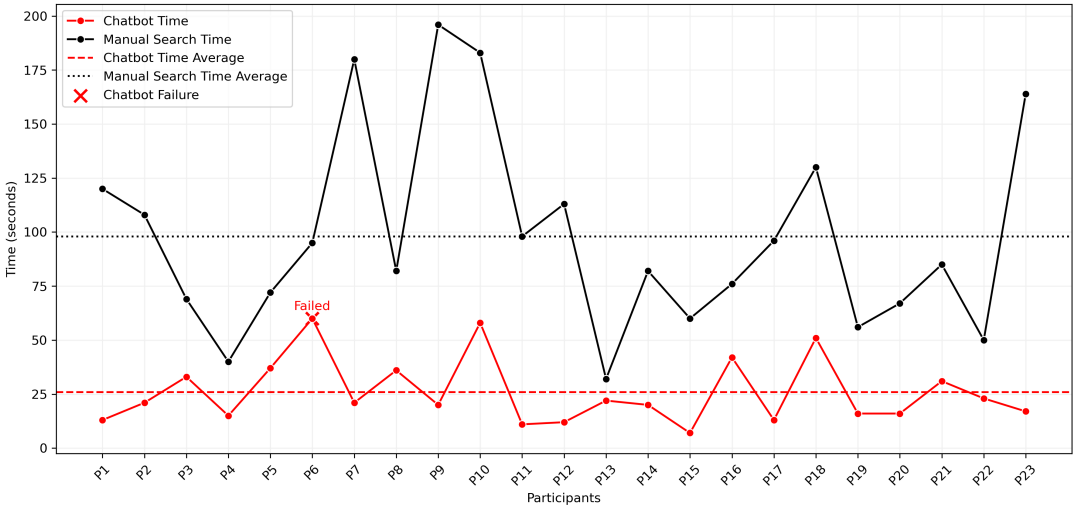



Figure 27: Comparison of Manual Time and Chatbot Search Time - Question 4

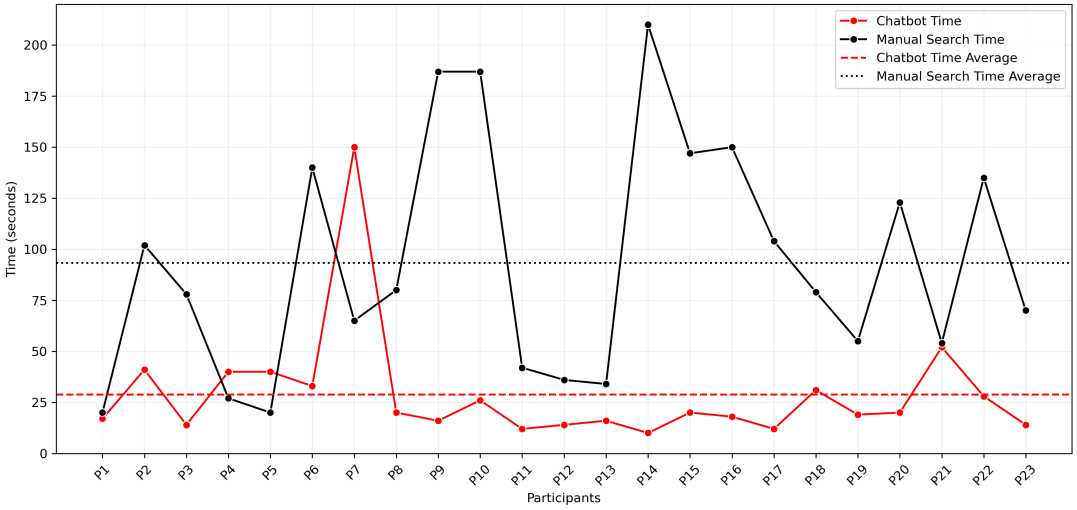



Figure 28: Comparison of Manual Time and Chatbot Search Time - Question 5

The results of the time comparison indicate that in 4 out of 5 cases, the chatbot had a faster average retrieval time compared to the website. The difference in retrieval speed ranged from 17.05% to 73.57%, highlighting the chatbot's efficiency in providing quicker responses. On average, across all five questions, the chatbot was nearly 300% faster than manual website navigation. However, the user journey study identified two instances where the chatbot hallucinated and provided incorrect answers—Participant 19 for Question 2 in Fig. 25 and Participant 6 for Question 4 in Fig. 27.

| | **Website Avg. Time (s)** | **Chatbot Avg. Time (s)** |
|---|---|---|
| **Question 1** | 14.65 | 19.57 |
| **Question 2** | 112.83 | 31.70 |
| **Question 3** | 31.22 | 22.78 |
| **Question 4** | 98.00 | 25.90 |
| **Question 5** | 93.26 | 28.82 |
| **Overall Average** | 70.00 | 25.75 |

Table 1: Comparison of Website vs. Chatbot Average Response Time

This section discusses the reasoning behind the hallucinated answers provided by the chatbot.

The correct answer was supposed to be "January 8 to March 19, 2025". However, the chatbot responded with "December 16, 2024, to January 15, 2025". The likely reason for this mistake is that it failed to fully recognise that the user was referring to the general application dates for all undergraduate programs.

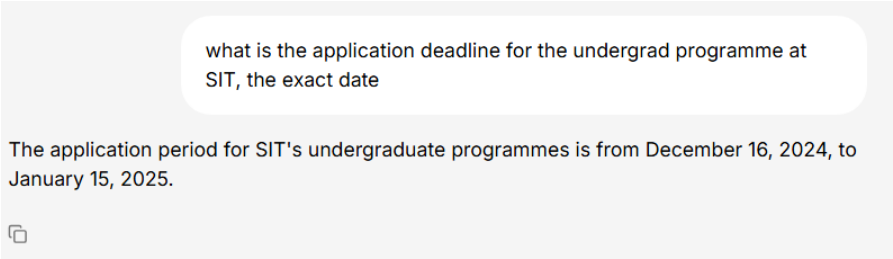



Figure 29: Hallucination in Participant 19's Response to Question 2

The chatbot failed to provide the course fee for the Electronic and Data Engineering course. However, when asked specifically about the course fee, it provided the correct answer. This mistake likely occurred because the database lists the information as "Course: BEng (Hons) in Electronics and Data Engineering, Total Credits: 240, Course Duration: 4 years. Subsidized fee per credit for Singapore Citizens: $173.00 ..." rather than explicitly stating "The cost of the course is...". As a result, the chatbot did not recognize that "course fee" and "cost" referred to the same concept.

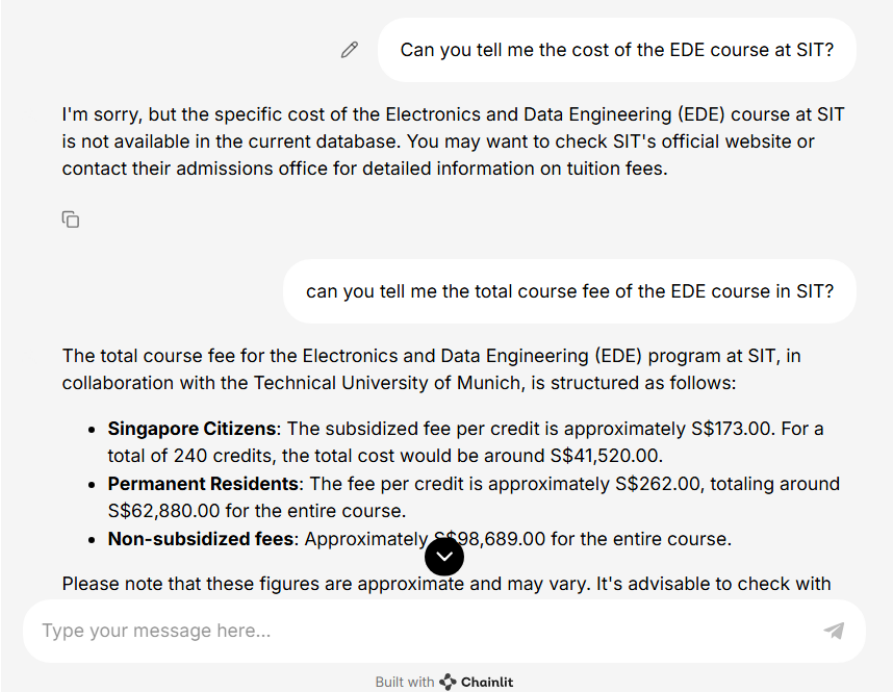



Figure 30: Hallucination in Participant 6's Response to Question 4

Bar charts were created to illustrate the number of different pages users clicked on while manually searching for answers on the website, as well as the number of prompts sent by users before receiving an answer. Since these are different metrics, they are evaluated separately rather than compared directly. The bar charts are included in the appendix under the "Pages Navigated by Users" section.

Below is a table summarizing the findings of users navigating the webpage manually.

| | **Website Avg. Clicks (no. of pages)** | **Website Avg. Time (s)** |
|---|---|---|
| **Question 1** | 1.52 | 14.65 |
| **Question 2** | 4.17 | 112.83 |
| **Question 3** | 2.26 | 31.22 |
| **Question 4** | 4.82 | 98.00 |
| **Question 5** | 3.78 | 93.26 |
| **Overall Average** | 3.31 | 70.00 |

Table 2: Average Pages Clicked per Question When Using the Website

The results show that the average number of clicks per question is around three. This is because the relevant information needed to answer each question is not easily intuitively located within the website's structure. Additionally, participants frequently scrolled past relevant information due to the overwhelming amount of content.

Below is a table summarizing the findings from users interacting with the chatbot. As mentioned earlier, the corresponding bar charts can be found in the appendix under the section "Prompts Entered by Users".

| | **Chabot Avg. Prompts Sent (no. of prompts)** | **Chatbot Avg. Time (s)** |
|---|---|---|
| **Question 1** | 1.73 | 19.57 |
| **Question 2** | 1.78 | 31.70 |
| **Question 3** | 1.04 | 22.78 |
| **Question 4** | 1.39 | 25.90 |
| **Question 5** | 1.26 | 28.82 |
| **Overall Average** | 1.44 | 25.75 |

Table 3: Average Prompts Sent per Question When Using the Website

The data indicates that, on average, one to two prompts are required per question, suggesting that the chatbot can accurately respond to most queries with minimal user input. This highlights its efficiency in processing and understanding user requests. However, response times of the chatbot could likely be improved, as these experiments were conducted locally on a laptop, where hardware limitations may have impacted performance. Additionally, some tests were performed via remote access, which may have introduced further delays.

### 6.2.3 Post-User Study Questions

The post-user study revealed that 82.6% of participants preferred using the chatbot, as shown in Fig. 31. The reasons for their preference, as well as the rationale of the 17.4% who favored the website despite evidence from the study indicating the chatbot's greater effectiveness, will be discussed in the following sections.

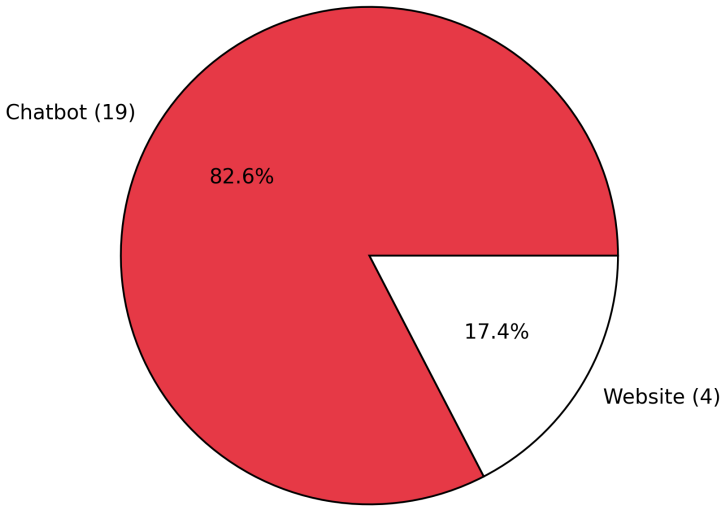



Figure 31: User Preference Between Chatbot and Website

From the post-user study questions, users indicated their reasoning for their preference. Therefore, a qualitative evaluation is conducted separately to identify common themes based on their choices.

A word cloud, as shown in Fig. 32, illustrates the responses of the 19 users who preferred the chatbot over the website.


Figure 32: Word Cloud of Chatbot Preference

The most common words found were "faster", "information", and "find", suggesting that most users agree it is significantly easier and quicker to retrieve information using the chatbot. To further analyze and classify the participants' feedback, their responses were embedded and clustered using K-means clustering. This technique provides a structured approach to evaluating qualitative data and is an adaptation of a method previously applied in a study published on December 20, 2024, which used Natural Language Processing (NLP) techniques for qualitative research [12].

The optimal number of clusters is determined by calculating the Mean Squared Error (MSE), as shown in the formula in Fig. 33. The steepest drop in MSE across different cluster sizes is identified as the inflection point, representing the optimal cluster size.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|x_i - c_{\text{assigned}}\|^2$$

Figure 33: Mean Squared Error (MSE) Formula

Where $x_i$ is a data point, $c_{\text{assigned}}$ is the centroid of the cluster to which $x_i$ is assigned and $\|\cdot\|^2$ represents the squared Euclidean distance and $n$ that represents total number of data points in the dataset.

As shown in Fig. 34, the MSE is plotted for the number of clusters ranging from 2 to 14, but the score only decreases gradually. This requires further investigation to determine the optimal number of clusters for grouping the feedback.

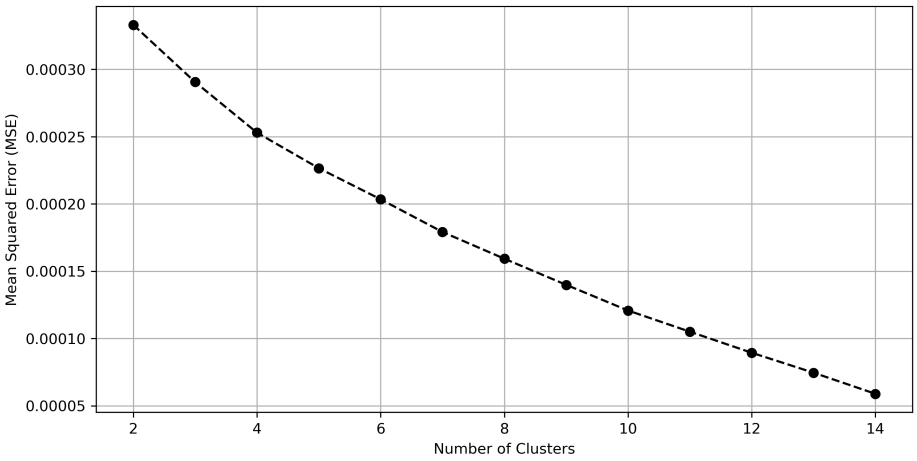



Figure 34: Elbow Method for Optimal Cluster Selection (MSE)

Assessing clustering quality involves calculating the silhouette score for each point within a cluster. However, before using the silhouette score to determine the optimal number of clusters, it is important to consider that the score relies on distance calculations. High-dimensional data, such as these embeddings with 1536 dimensions, can pose challenges due to the "curse of dimensionality" [13]. To mitigate this issue, embedding dimensions can be reduced using Principal Component Analysis (PCA) while preserving as much variance as possible before calculating the silhouette score across different clusters.

A common guideline suggests retaining approximately 80% of the variance for data analysis and 90% for predictive modeling [14]. Based on the variance score plotted across different numbers of components in Fig. 35, PCA with 11 components was selected.

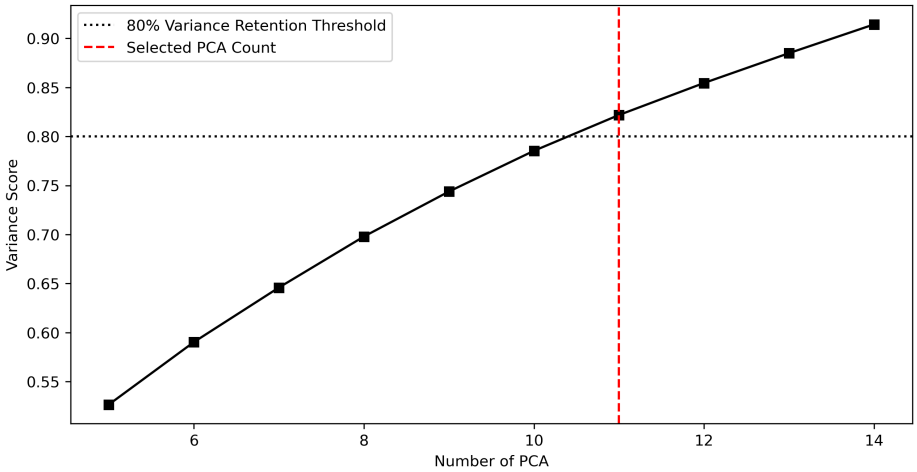



Figure 35: Variance Score for Optimal PCA Selection

The silhouette formula, shown in Fig. 36, produces a score ranging from -1 to 1, with values closer to 1 indicating well-separated clusters. The plotted silhouette scores across different numbers of clusters (2 to 14), as shown in Fig. 37, reveal that the most significant increase occurs between 3 and 4 clusters. Therefore, 4 clusters, with a silhouette score of 0.15, was selected as the optimal choice.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Figure 36: Silhouette Coefficient Formula

Where $s(i)$ is the silhouette score for data point $i$, $a(i)$ is the average distance from $i$ to all other points in the same cluster (intra-cluster distance) and $b(i)$ is the average distance from $i$ to all points in the nearest different cluster (inter-cluster distance).

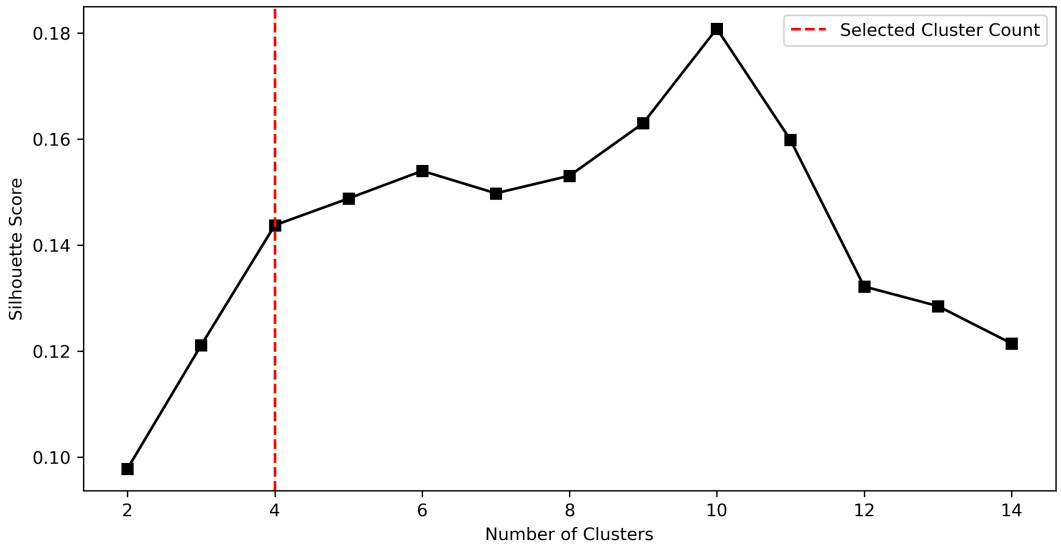



Figure 37: Silhouette Score for Optimal Cluster Selection

Clustering was performed using the 11 components from the optimal PCA selection, which were later reduced to two for visualization in the scatter plot shown in Fig. 38. The plot demonstrates that the four clusters effectively separate the data points into distinct categories. In the figure, "P" denotes a participant, with the accompanying number representing their assigned identifier (e.g., P1, P2, etc.).

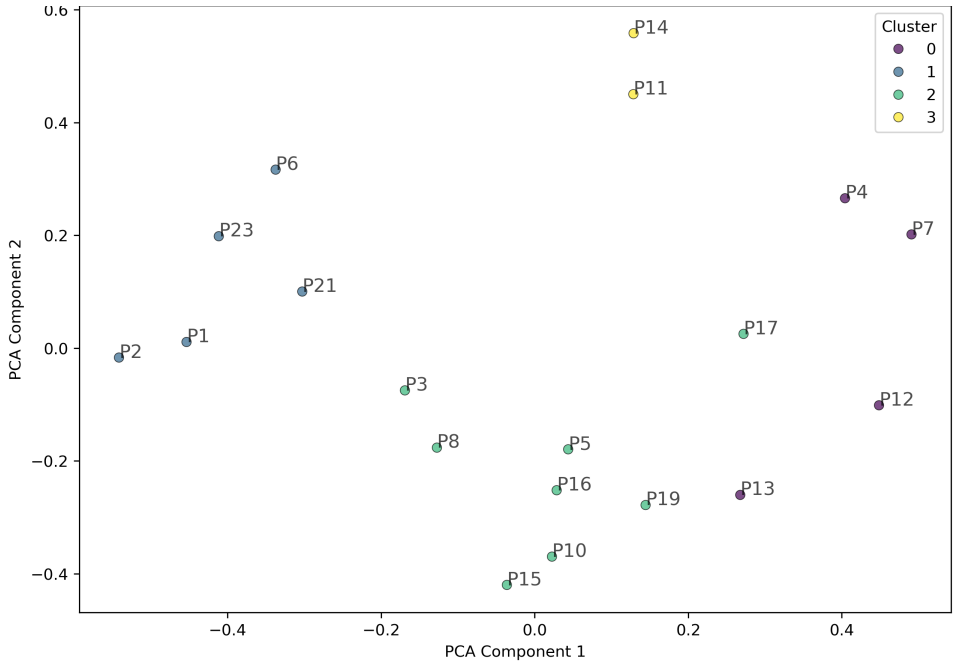



Figure 38: Visualization of Optimal Cluster Selection

The participants' feedback were grouped into four clusters, and their distribution is shown in Fig. 39, where each section of the pie chart represents a cluster of similar feedback from the participants. Each of the cluster categories is further elaborated and supported with exact quotes from participants in the section below.

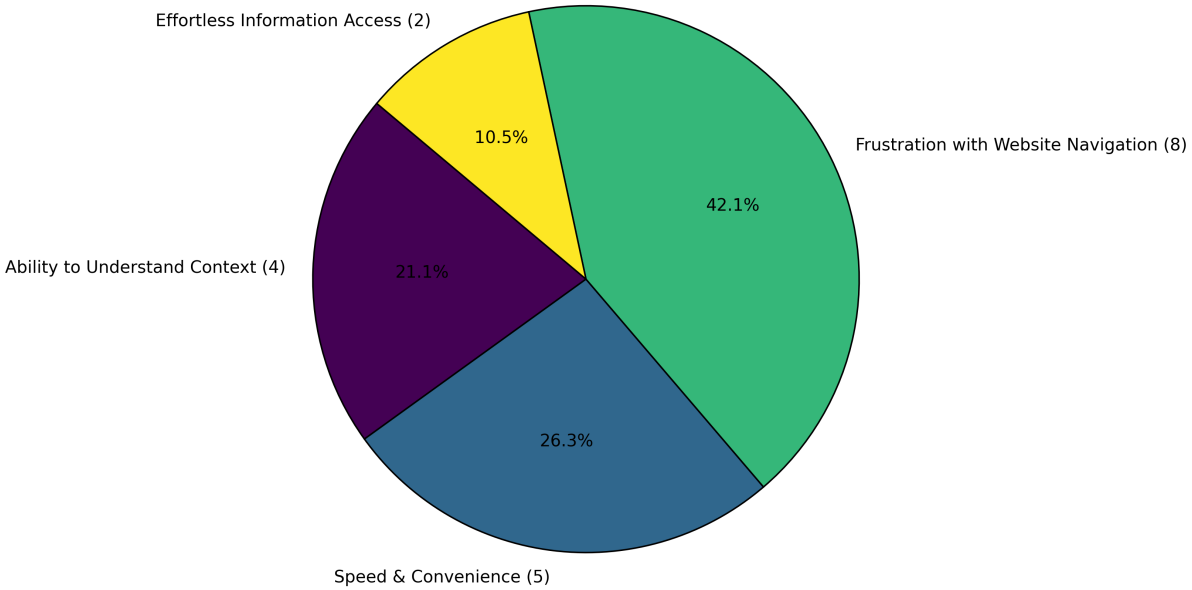



Figure 39: Distribution of Participants by Category

The four cluster categories identified are:

1. **Effortless Information Access**: Participants in this category appreciate the chatbot's ability to provide information quickly and with minimal effort. Unlike traditional website navigation, which often requires scanning through multiple pages or menus, the chatbot allows users to retrieve relevant information simply by typing a question or keyword, eliminating the need for extensive searching.

    - **P16**: "It is so much more convenient, you just need to type in key words and you dont have to squnich your eyes to look for everything on the website"
    - **P19**: "... it allows me to use natural language to ask my questions instead of clicking around ..."

2. **Frustration with Website Navigation**: Participants in this category experience significant frustration when navigating the website compared to using the chatbot. They find the website structure confusing, with information scattered across multiple pages, making it difficult to locate what they need.

    - **P11**: "Because some of the website information is so stupid and hard to find"
    - **P14**: "The website is very confusing in general and the way that is it categorized can be quite confusing ..."

3. **Speed & Convenience**: Participants in this category value the chatbot's quick response time and seamless interaction. While "Effortless Information Access" focuses on reducing the effort needed to find information, this category highlights the chatbot's speed in delivering answers, making it the preferred choice for quick, straightforward queries.

    - **P1**: "I get my question answered a lot faster than the website"
    - **P2**: "the chatbot is faster"
    - **P6**: "the chatbot is quicker ..."
    - **P21**: "... fast and straight forward information then it is chatbot ..."

4. **Ability to Understand Context**: Participants in this category appreciate the chatbot's ability to comprehend their questions with minimal context, reducing the need for repetitive clarifications or follow-ups.

    - **P4**: "... dont need to ask many questions ..."
    - **P7**: "all the question can be link to a single subject"
    - **P13**: "I can just ask it questions and immediately get the answer"

The next section evaluates users who preferred the website. A word cloud, as shown in Fig. 40, illustrates the responses of the four users who chose the website over the chatbot.

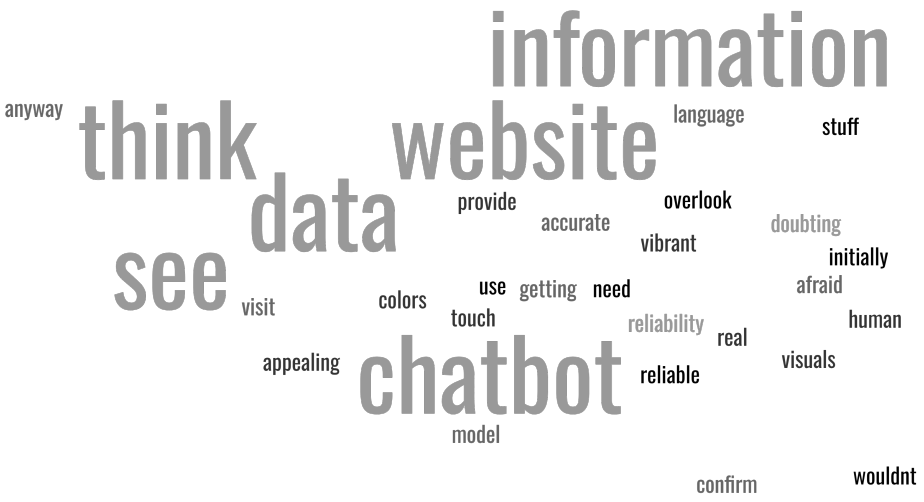



Figure 40: Word Cloud of Website Preference

The most common words found are "data", "information", and "see". These words suggest that users may be uncertain about the chatbot's reliability and instead prefer the visual and interactive experience of a website, which a chatbot may not fully provide. Since there are only four users, the themes are easily categorized and identifiable. Similar to the previous justification for each category, these categories are further elaborated and supported with exact quotes from participants in the section below.

The common themes identified are:

1. **Reliability of the Chatbot**: Participants in this category are concerned about the trustworthiness of the information retrieved by the chatbot and seek reassurance regarding its accuracy and chain of thought.

    - **P9**: "Doubting the reliability of the language model, afraid of not getting accurate data and I will visit the website anyway to confirm if the data real"

2. **Visual Appearance**: Participants prefer the interactive and visually engaging User Interface (UI) found on websites.

    - **P18**: "... I like to see the vibrant colors and the human touch in it. The visuals make it more appealing than the chatbot"

3. **Data Availablity**: Participants appreciate the broader range of information available on websites, including details they might not have initially considered. Even Participant 21, despite preferring the chatbot, indicated a preference for the website when seeking more in-depth information.

    - **P20**: "I think that the website will provide me to have more information that I would think initially that I wouldnt need, because I would have stuff that I would have overlook if I were to use the chatbot"
    - **P21**: "... if I require more indepth I will use the website"
    - **P22**: "I can see all the information and it is more reliable"

### 6.2.4 Notable Chatbot Interactions

This section highlights key insights and notable interactions from the user study.

The chatbot can understand the user's intent despite inconsistencies in spelling and capitalization, providing flexibility for users to type incomplete sentences or with spelling errors. This capability enhances the user experience by reducing the need for precise input, allowing for more natural and effortless interactions. The following screenshot in Fig. 41 and Fig. 42 illustrate instances where users engage with the chatbot using varied spelling, capitalization, and incomplete phrases, showcasing its ability to interpret and respond accurately.

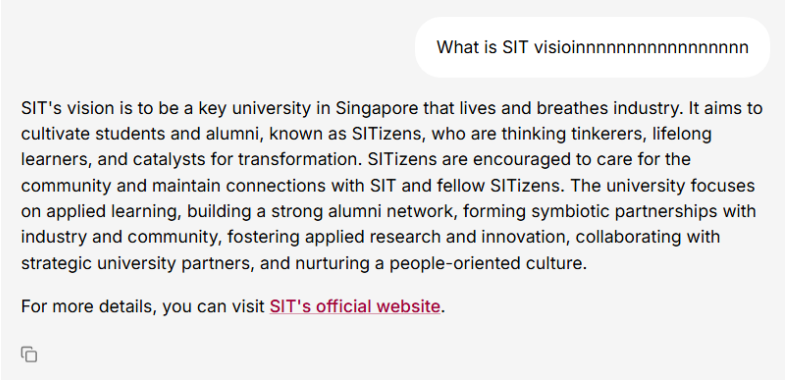



Figure 41: Chatbot Interaction with Participant 18 regarding SIT's Vision

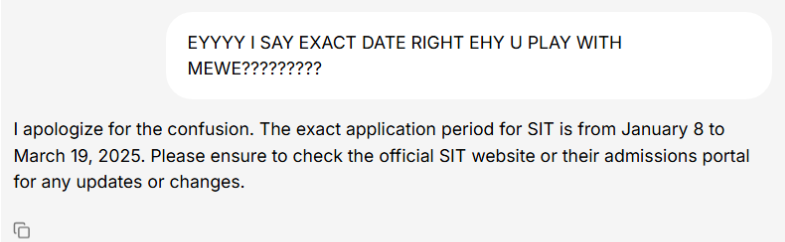



Figure 42: Chatbot Interaction with Participant 18 about Application Dates

The chatbot's ability to understand multiple languages is beneficial in accommodating diverse users with varying linguistic preferences. This feature allows users to interact naturally in their preferred language, enhancing accessibility and ease of use. The screenshots in Fig. 43 and Fig. 44 display real interactions where users engage with the chatbot in different languages, demonstrating its capability to process and respond accurately regardless of linguistic variations.

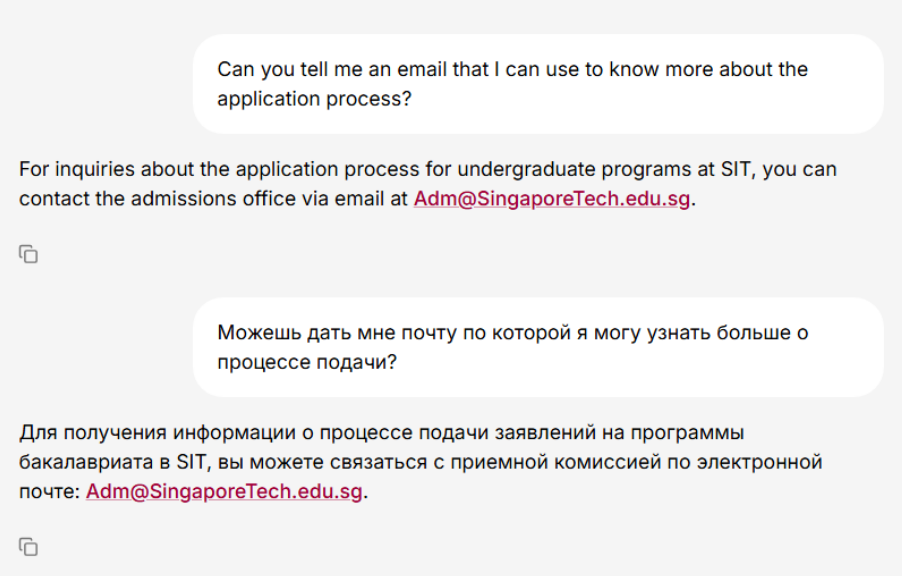



Figure 43: Chatbot Interaction with Participant 16 on SIT's Contacts in Russian


Figure 44: Chatbot Interaction with Participant 16 About SIT's Strengths in Chinese

## 6.3 Conclusion of User Study

This section outlines key insights from the user study regarding the chatbot's purpose and development. It highlights challenges identified by participants and presents proposed solutions for improvement. Further details and implementation strategies are elaborated in the next section.

1. **Reliability of the Chatbot**

    - **Problem**: Participants expressed concerns about the trustworthiness of the chatbot's responses and sought reassurance regarding its accuracy.
    - **Proposed Solution**: To enhance trust, the chatbot should state which tool it has used and provide details on what was searched within that tool to generate its response for the user.

2. **Limited Visual Appeal**

    - **Problem**: Some participants prefer the website due to its interactive UI and visually engaging elements, finding the chatbot's interface less appealing.
    - **Proposed Solution**: To improve user engagement, the chatbot could be integrated within the website at the bottom corner, ensuring easy accessibility while maintaining the site's interactive experience.

3. **Scalability and Performance Limitations**

    - **Problem**: All participants could only access the chatbot through a single local computer or via remote access, resulting in limited computing performance and restricting usage to a single user at a time.
    - **Proposed Solution**: To support multiple users, the chatbot should be hosted on the cloud using cloud technologies such as Microsoft Azure, Amazon Web Services (AWS), and Google Cloud Platform (GCP).

4. **Improving User Guidance**

    - **Problem**: The majority of participants were still unsure of the chatbot's purpose and usage despite having interacted with chatbots before.
    - **Proposed Solution**: To improve clarity, the chatbot should display a clearer welcome message stating its purpose and provide suggestion buttons to guide users on what types of questions to ask.

# 7 Implementation of Feedback

This section discusses the implementation of the solutions presented in the user study.

## 7.1 Updated Workflows

This section outlines the updated workflow for Chainlit and Telegram, aimed at addressing the issues of "Improving User Guidance" and "Reliability of the Chatbot". The areas highlighted in red indicate modifications made to the diagram following the user study.

### 7.1.1 Updated Chainlit Workflow

This section explains the updated Chainlit diagram shown in Fig. 45. The workflow updates include changes to the welcome message and the addition of a suggestion button generator to address the issue of "Improving User Guidance". Additionally, a tool display function has been introduced to enhance the "Reliability of the Chatbot", and a summary function has been implemented. The updated decorators are explained, while the rest of the workflow remains as previously described.

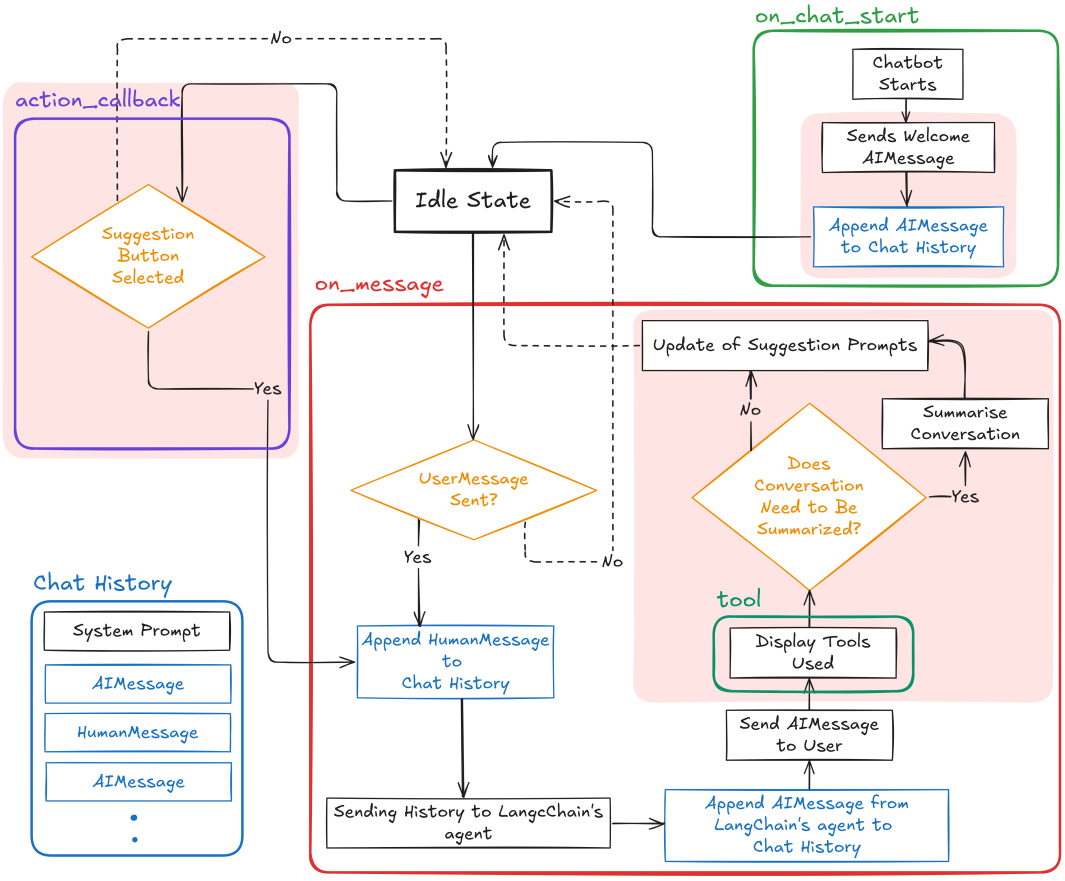



Figure 45: Updated Chainlit Logic Flow

The following subsections provide a detailed explanation on each updated decorator.

1. **"on_chat_start"**: The welcome message has been adjusted to shows more capabilities of what the chatbot is able to do. This can help the users better in knowing what kind of questions to ask.

2. **"on_message"**: This decorator has been updated to check whether the conversation needs to be summarized, and then updates the suggestion prompts based on the response. It then moves back to the Idle State, waiting for the next interaction. The summary function and the update of the suggestion prompt will be further explained in the later part of this section.

3. **"action_callback"**: This decorator has been created to manage interactions when the user selects a suggestion button. If a suggestion button is clicked, the corresponding prompt is captured and passed to the "on_message" decorator for processing, treating it as user input. The chatbot then generates a response based on the selected prompt and updates the chat history accordingly.

4. **"tools"**: This decorator has been created to be used within the "on_message" decorator to display the tools being used in generating the response, ensuring transparency in the chatbot's decision-making process.

The welcome message was refined to enhance user experience and provide clearer guidance on using the Chatbot. After reviewing multiple articles and blogs on crafting effective welcome messages, key suggestions were incorporated, such as clearly stating the Chatbot's capabilities and breaking the message into two separate parts to improve readability [15], [16]. These adjustments ensure that users can quickly understand the Chatbot's functionality and interact with it more effectively.

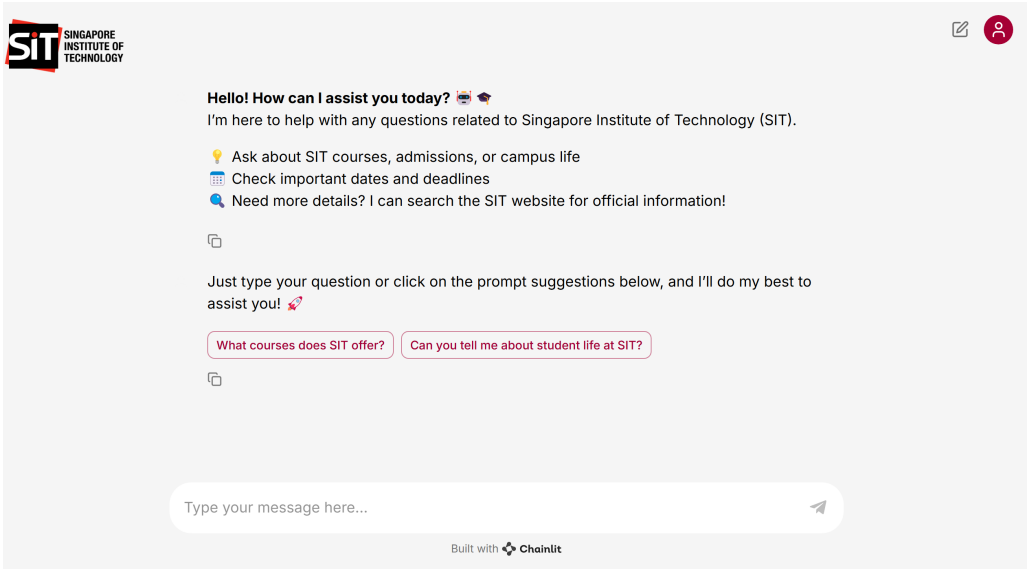



Figure 46: Chatbot Welcome Message in Chainlit

As shown in Fig. 47, the tool message indicates that the phrase "courses offered by SIT" is used to search for similar documents in the "SIT_database". This helps users understand how the chatbot retrieves information, including which keywords are used and which database is accessed. Additionally, the suggestion prompts are dynamically adjusted based on the chatbot's output. These new features have been implemented to address the challenges of "Reliability of the Chatbot" and "Improving User Guidance".

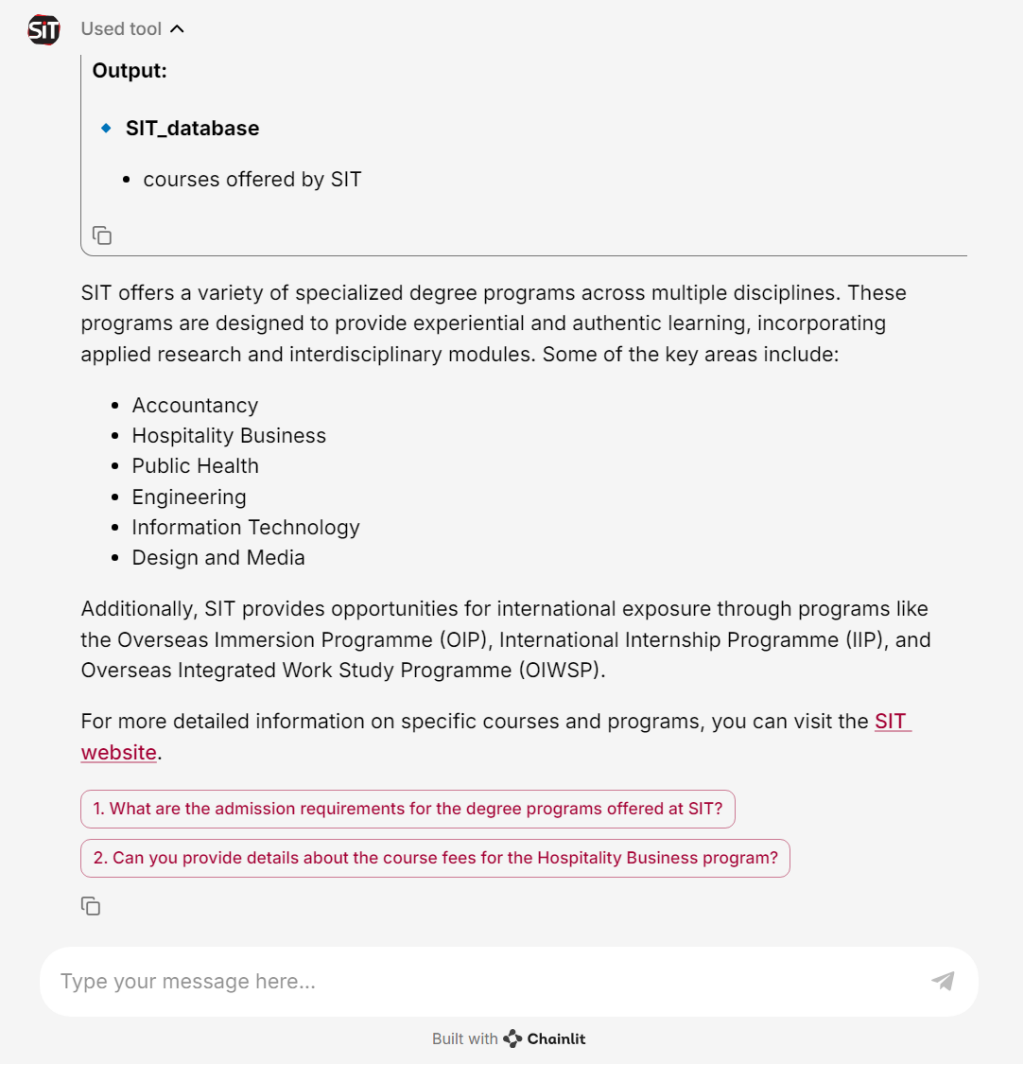



Figure 47: Tool Usage and Suggestion Prompts in Chainlit

### 7.1.2 Updated Telegram Workflow

This section shows the updated flowchart of the Telegram chatbot. The main difference is a summary function implemented into the "handles_all_message" decorator, the rest of the functionality remains the same as described in the section above.

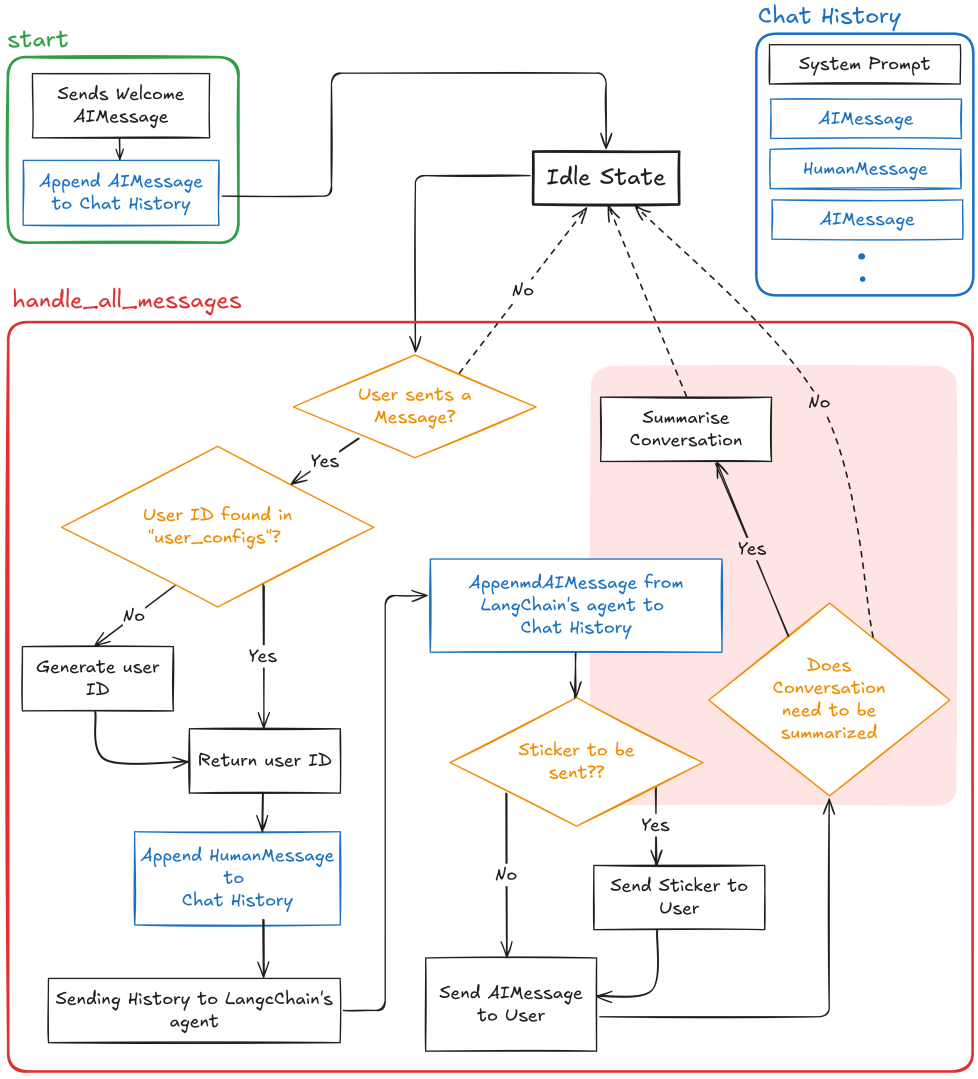



Figure 48: Updated Telegram Logic Flow

The following subsections provide a detailed explanation of each updated decorator.

1. **"handle_all_messages"**: Similarly to the updated Chainlit's "on_message" decorator, this decorator has been modified to evaluate whether the conversation needs to be summarized before displaying the response to the user. The summary functionality is identical to the one used in Chainlit.

### 7.1.3 Implementation of Prompt Suggestion and Summarization

The suggestion prompts are generated by a separate Large Language Model (LLM) that understands the chatbot's capabilities. The LLM provides suggested prompts for follow-up questions based on the chatbot's responses and these suggestions are then placed into the suggestion buttons.

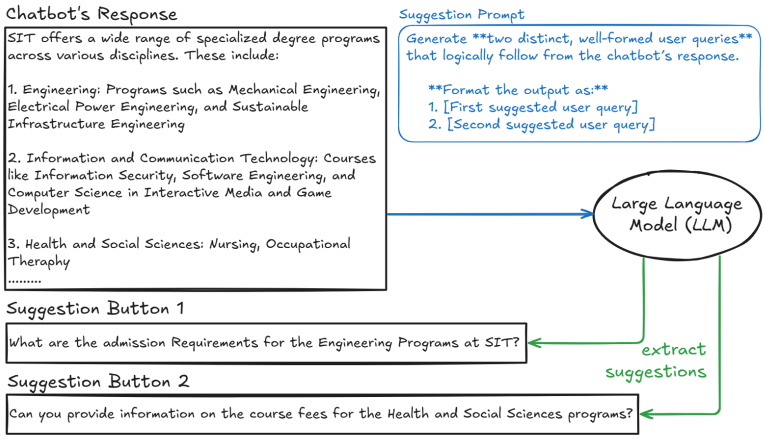



Figure 49: Suggestion Prompt Function

The summarization function ensures that the conversation length remains fixed and cost-efficient, as the cost of the LLM's API is based on the amount of text fed into it. The current chat history is processed before the "on_message" decorator finishes running. The function checks whether the conversation length is greater than or equal to 6. If it is not, the chat history remains unchanged, and the rest of the functions within the decorator continue running. However, if the conversation length exceeds 6, the system prompt, which contains the chatbot's instructions, is modified into a summary system prompt and fed into the LLM to generate a summary of the recent conversation. The chat history is then adjusted to retain only the summary and the system prompt.

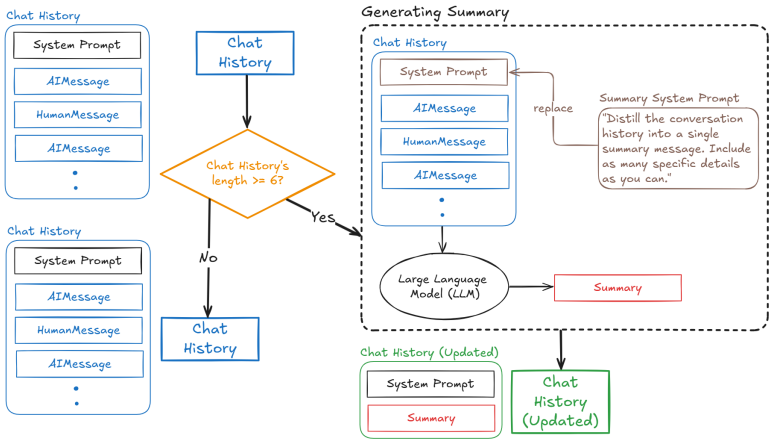



Figure 50: Summary Function

The LLMs used for these tasks above do not require highly accurate responses; therefore, "GPT-4o Mini" was selected due to its lower cost compared to "GPT-4o", with a price of $0.150 per 1 million input tokens, whereas "GPT-4o" costs $2.50 per 1 million input tokens [10].

## 7.2 Containerization and Deployment

Containerization has been used to deploy the application efficiently, addressing the issue of "Scalability and Performance Limitations". The logic of Chainlit and Telegram can be containerized, allowing them to access the Redis database through a dedicated server. This ensures that database updates do not affect the current working containerized instance of the chatbot. However, in Chainlit, users are not assigned individual logins, which may pose limitations in tracking and managing user sessions.

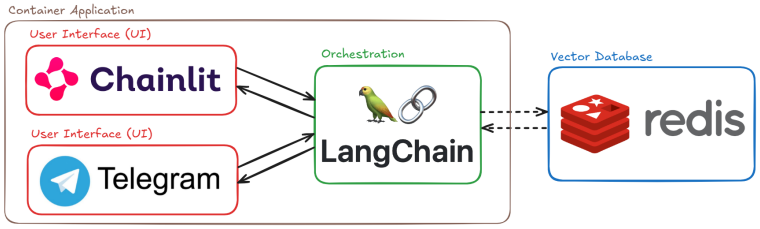



Figure 51: Overview of Deployment Workflow

## 7.3 Potential Implementations

A possible design on the SIT website to address the issue of "Limited Visual Appeal" is to position the chatbot at the bottom corner of the screen as shown in Fig. 52. This ensures that users can interact with the chatbot while simultaneously navigating the website.

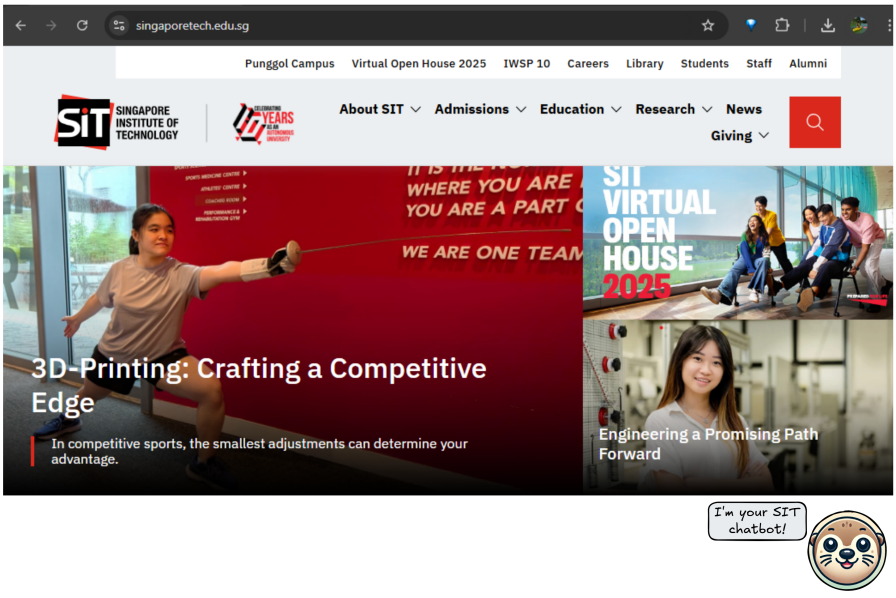



Figure 52: Proposed Chatbot Integration on SIT Website

# 8 Future Works and Conclusion

This thesis presented the development of an intelligent chatbot designed to assist students with administrative and academic queries. By leveraging LangChain for backend processing and integrating with platforms like Chainlit and Telegram, the chatbot provides a user-friendly interface for seamless interactions. The implementation of Retrieval-Augmented Generation (RAG) enhances response accuracy by retrieving relevant information from institutional documents, ensuring that students receive timely and reliable answers.

A user study with 23 participants was conducted to evaluate the chatbot's effectiveness, usability, and impact on user experience. Participants provided valuable feedback on functionality, response reliability, and user guidance, leading to refinements in the chatbot's design. Key improvements included enhanced welcome messages, dynamic suggestion buttons, and summarization functions, all of which contributed to a more intuitive and effective chatbot experience.

While the chatbot successfully improves accessibility to institutional information, certain limitations remain. Scalability challenges and performance constraints were addressed through containerization, but additional optimizations may be required for broader deployment. Future work could explore integration with Google Maps and Visual Language Models (VLMs) for campus navigation, as well as comparative evaluations of RAG-based responses versus Reasoning Models.

In summary, this chatbot serves as a foundational step toward intelligent student assistance. With further enhancements and expanded functionalities, it has the potential to become a robust tool for improving information accessibility and student engagement in academic environments.

# References


[1] Singapore Institute of Technology. *SIT Portal*. https://www.singaporetech.edu.sg/undergraduate-programmes. Accessed: Oct. 11, 2024.

[2] A. Vaswani et al. "Attention is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017). Accessed: Oct. 11, 2024. URL: %5Curl%7Bhttps://arxiv.org/abs/1706.03762%7D.

[3] 3Blue1Brown. *3Blue1Brown YouTube Channel*. https://www.youtube.com/c/3blue1brown. Accessed: Oct. 11, 2024.

[4] Shuhei Suzuoki and Keiko Hatano. *Reducing Hallucinations in Large Language Models: A Consensus Voting Approach Using Mixture of Experts*. https://www.techrxiv.org/users/794815/articles/1112497/master/file/data/v/v.pdf?inline=true. Accessed: Dec. 16, 2024.

[5] Patrick Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *arXiv preprint* arXiv:2005.11401v4 (2021). Accessed: Dec. 16, 2024. URL: https://arxiv.org/pdf/2005.11401.

[6] DataCamp. *Fine-Tuning Large Language Models: Techniques and Applications*. https://www.datacamp.com/tutorial/fine-tuning-large-language-models. Accessed: Dec. 16, 2024.

[7] eCommerceBonsai. *25+ Top Chatbot Statistics for 2024: Usage, Demographics, Trends*. https://ecommercebonsai.com/chatbot-statistics. Accessed: Oct. 11, 2024.

[8] OpenAI. *Embeddings Guide*. Accessed: March 5, 2025. 2024. URL: https://platform.openai.com/docs/guides/embeddings.

[9] Telegram. *Telegram Bot API*. Accessed: March 5, 2025. 2024. URL: https://core.telegram.org/bots/api.

[10] OpenAI. *OpenAI API Pricing*. Accessed: March 5, 2025. 2024. URL: https://openai.com/api/pricing/.

[11] Annabel Blake. *Birds of a Feather: From Chaos to Clarity - How to Turn Messy User Feedback Into Clarifying and Inspiring Action*. Presented at SIGGRAPH Asia 2024, G40B, G Block Level 4. Accessed: March 6, 2025. Canva, Dec. 2024.

[12] Tor Ole B. Odden et al. "Using text embeddings for deductive qualitative research at scale in physics education". In: *Physical Review Physics Education Research* 20.2 (2024), p. 020151. DOI: 10.1103/PhysRevPhysEducRes.20.020151. URL: https://journals.aps.org/prper/abstract/10.1103/PhysRevPhysEducRes.20.020151.

[13] Anoop Maurya. *Embeddings: A Deep Dive into Capturing Meaning in Low Dimensions*. Accessed: March 7, 2025. 2024. URL: https://medium.com/@mauryaanoop3/embeddings-a-deep-dive-into-capturing-meaning-in-low-dimensions-76c5a75d89d0.

[14] Ian T. Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202. DOI: 10.1098/rsta.2015.0202. URL: https://doi.org/10.1098/rsta.2015.0202.

[15] Ubisend. *Chatbot Welcome Message Best Practices*. Ubisend Blog. [Accessed: 28-Feb-2025]. 2025. URL: https://blog.ubisend.com/optimise-chatbots/chatbot-welcome-message-best-practices.

[16] LiveChatAI. *AI Chatbot Welcome Message Examples*. LiveChatAI Blog. [Accessed: 28-Feb-2025]. 2025. URL: https://livechatai.com/blog/ai-chatbot-welcome-message-examples.

# Appendix

## Pages Navigated by Users

The bar charts below show the number of pages participants had to navigate before finding the answers on SIT's website.

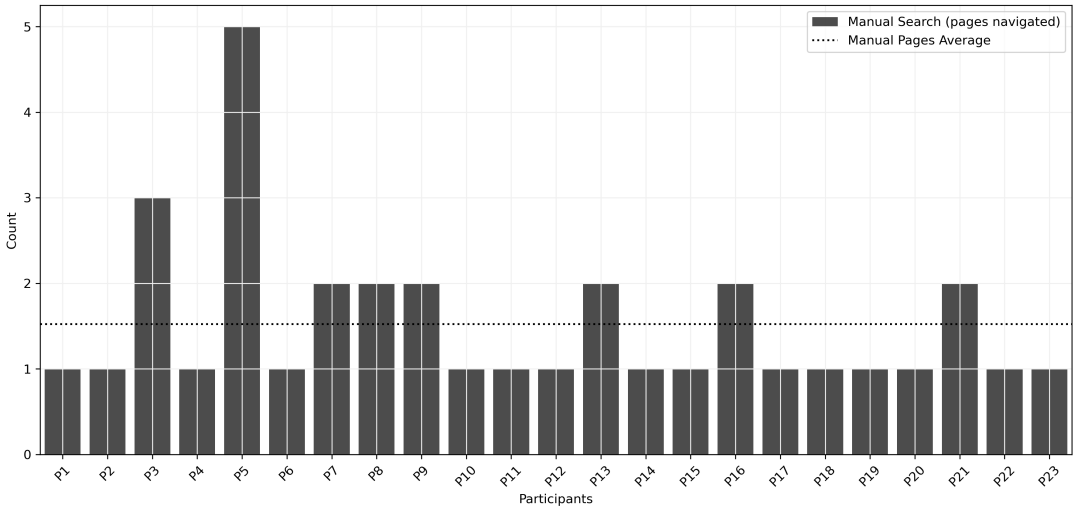



Figure 53: Comparison of Manual Pages Navigated - Question 1

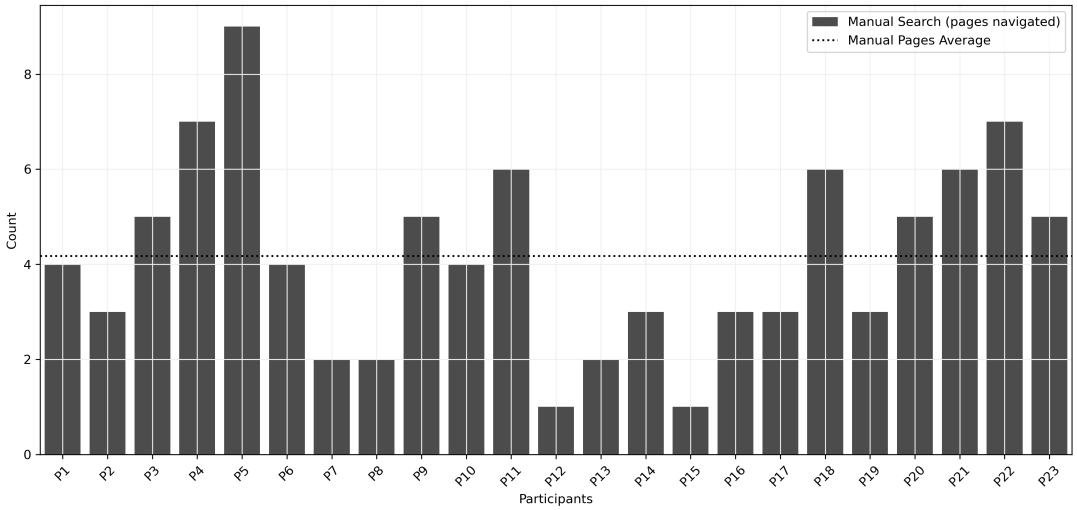



Figure 54: Comparison of Manual Pages Navigated - Question 2

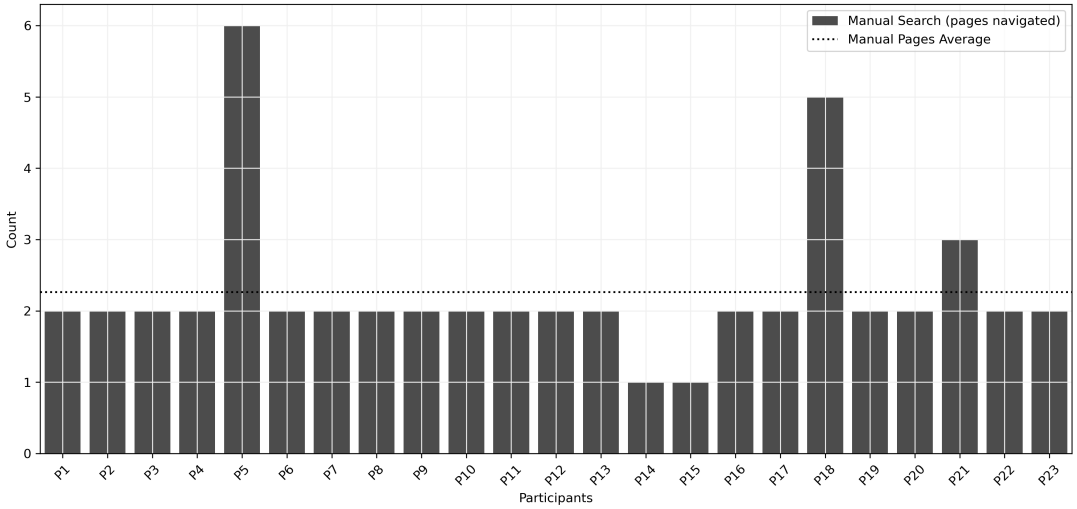



Figure 55: Comparison of Manual Pages Navigated - Question 3

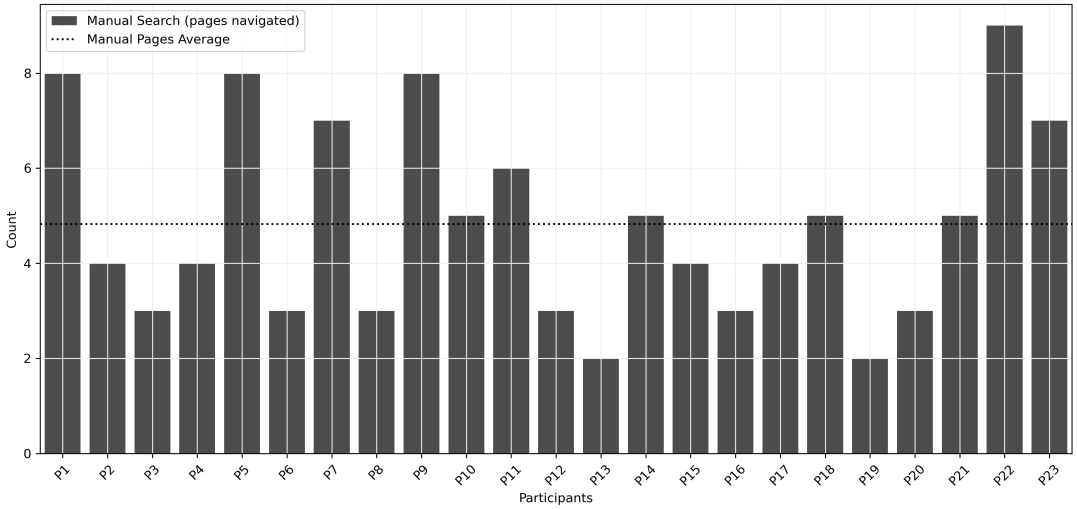



Figure 56: Comparison of Manual Pages Navigated - Question 4

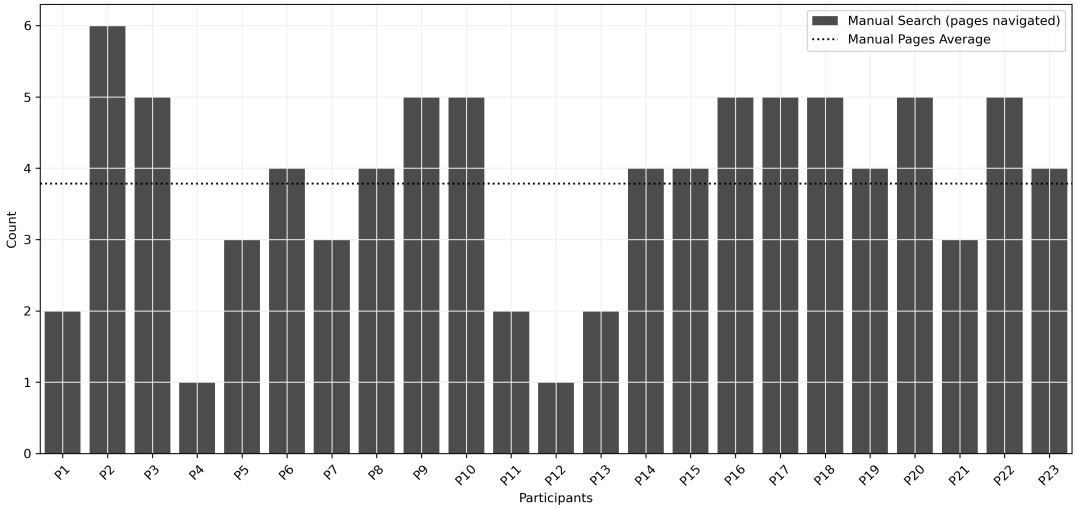



Figure 57: Comparison of Manual Pages Navigated - Question 5

## Prompts Entered by Users

The bar charts below show the number of prompts participants had to send before the answers were displayed using the chatbot.

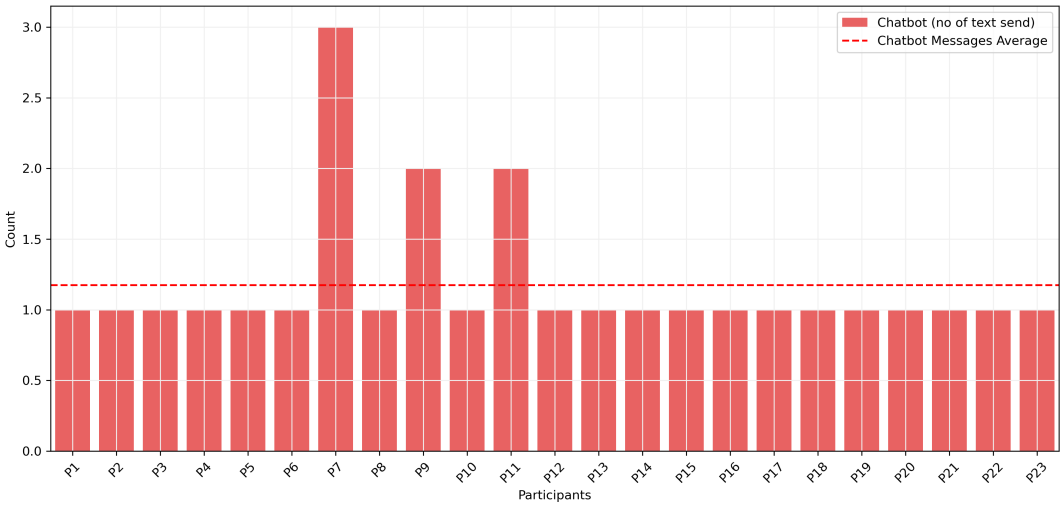



Figure 58: Comparison of Chatbot Messages Sent - Question 1

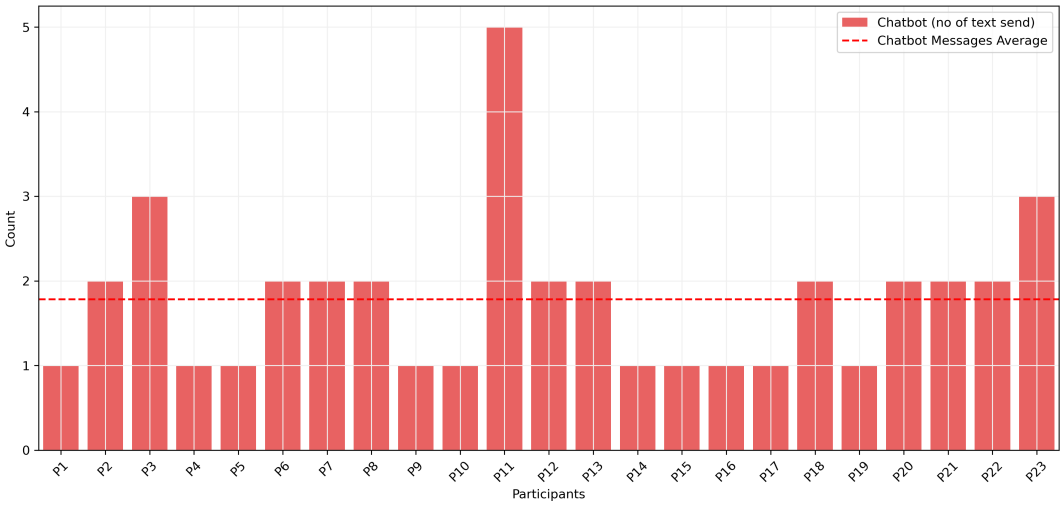



Figure 59: Comparison of Chatbot Messages Sent - Question 2

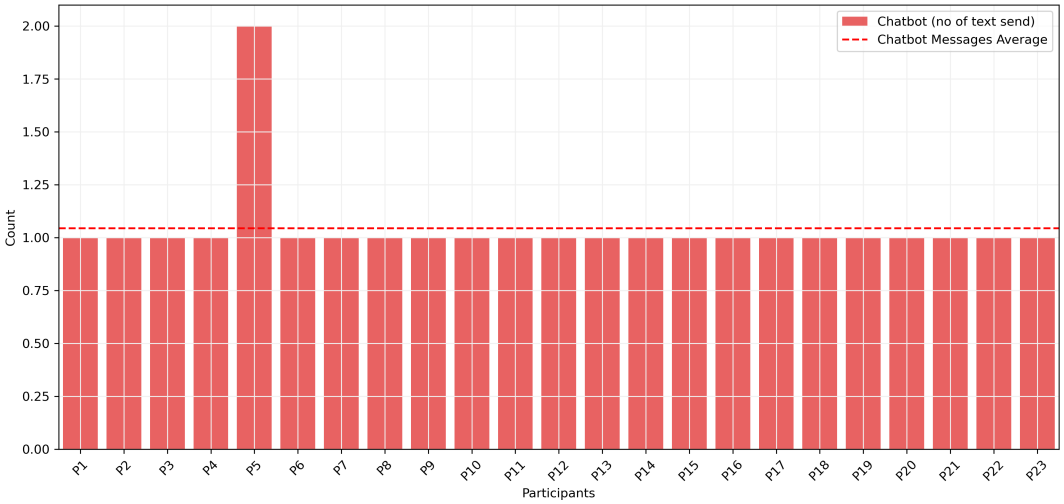



Figure 60: Comparison of Chatbot Messages Sent - Question 3

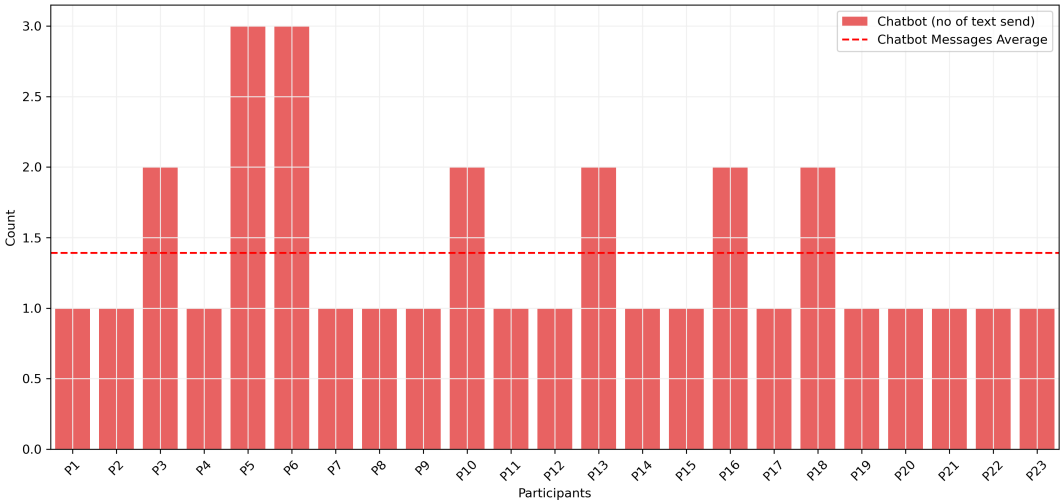



Figure 61: Comparison of Chatbot Messages Sent - Question 4

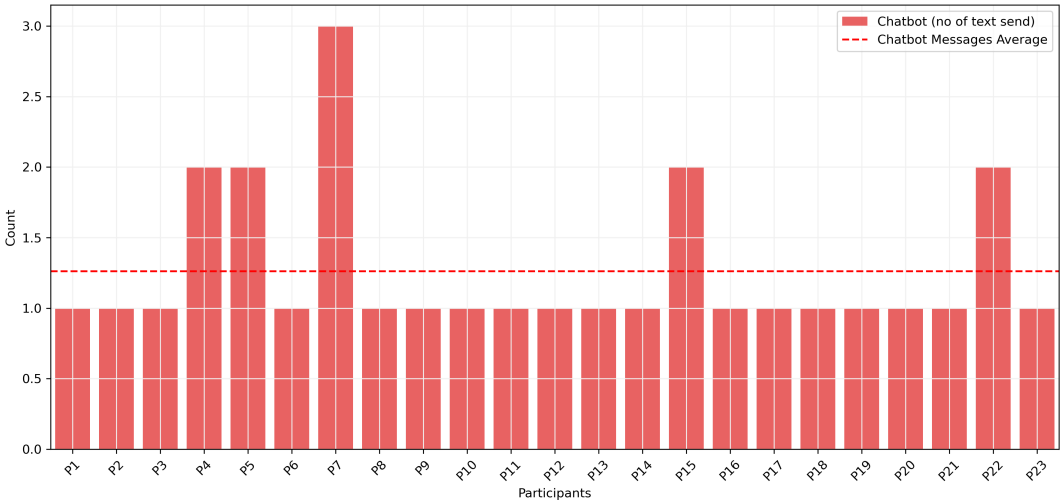



Figure 62: Comparison of Chatbot Messages Sent - Question 5