Student Satisfaction Survey Chatbot

By

Lee Wei Jin

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERINNG

Faculty of Information and Communication Technology

(Kampar Campus)

January 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: __Student Satisfaction Survey Chatbot ___________________________

____

____

Academic Session: JAN 2024

I ___________LEE WEI JIN______________________________

(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:

__6, Lorong Jasa Indah 4,_____

__Taman Jasa Indah,_________

__14000 Bukit Mertajam_____

_Dr. Zurida Binti Ishak______

Supervisor's name

Date: ___26/04/2024__________________

Date: __28/04/2024______________

Bachelor of Information Systems (Honours) Information Systems Engineering
Faculty of Information and Communication Technology (Kampar Campus), UTAR

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

FACULTY OF INFROMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: ___26/04/2024_______________

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that *Lee Wei Jin* (ID No: *20ACB01909*) has completed this final year project entitled "Student Satisfaction Survey Chatbot" under the supervision of Dr. Zurida Binti Ishak (Supervisor) from the Department of Information System, Faculty of Information and Communication Technology .

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

Yours truly,

____

(*Lee Wei Jin*)

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Student Satisfaction Survey Chatbot**" 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 : _______Lee Wei Jin__________________

Date : _______26/04/2024__________________

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to both my previous and current supervisor. I want to thank my previous supervisor Dr. Dalvi Esfahani for providing me ideas for this project which made me research more on the matter and it benefitted in my expansion of knowledge. I also want to thank my current supervisor, Dr. Zurida Binti Ishak for guiding me on my project forward.

A huge thanks to my family members for constantly providing me with unconditional love and support throughout my university life which is what enables me to continue with this project.

This project modified open-source language models as well as open-source UI templates to construct the interfaces. The open-source resources used will be included in the reference section as acknowledgment of the contribution it had made for this project.

ABSTRACT

Feedback is a piece of external information that is crucial for improvement. It can be found from different sources, whether it is a user of a product, a teacher guiding a student, or a customer in a restaurant. This project will be focusing on student's satisfaction feedback about their university experience. Traditional web survey are widely used to collect feedback from students. University students tend to be more open to give feedback, and so, university management can take advantage of this to understand student's problem. This project explores the implementation of AI chatbot in conversations with students and ask follow-up questions to gain insight on student's university experience. Open-source libraries and models such as Natural Language ToolKit library, langchain and hugging face are used to integrate and modify the chatbot model and add functionality to it like predicting sentiment of a feedback. By leveraging on Natural Language Processing technology advancement, Large Language Models (LLMs) are used as the foundation for the text generation capabilities of the chatbot. To ensure high quality response, profanity filter and language detector models are also integrated using pre-existing python libraries like profanity-filter and langid. The result is a multifunctional chatbot system that can simultaneously predict sentiment of text, detect profanity, determine the language, and generate text.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
SSSC	Student Satisfaction Survey Chatbot
LLMs	Large Language Models
RNN	Recurrent Neural Network
RLHF	Reinforcement Learning with Human Feedback
SFT	Supervised Fine-tuning
GPT	Generative Pre-trained Transformers
BERT	Bidirectional Encoder Representation from Transformers

CHAPTER 1
Project Background

In this chapter, included below are the background and motivation for the research, contributions of the project to the field, and the problem statement.

1.1 Introduction

With the rapid advancement of Artificial intelligence (AI), smarter software and hardware are being developed which is also known as "intelligent agents" [1]. Now, one of these "intelligent agents" made its way into the Education sector in the form of assistive chatbots. Chatbot is a software that acts as a virtual assistant to simulate human conversations and provide correct responses to questions asked by the user [2]. This interaction is made possible through integration of different algorithms such as Nature Language Processing (NLP) Algorithm, Naïve Bayes Algorithm, Recurrent Neural Networks (RNN) and Markov Chains [3]. Chatbot is gaining popularity over the years because there are more users using messaging services especially during covid-19 pandemic period where almost everything is done online. Together with the advancement in Natural Language Understanding (NLU) and Machine Learning, chatbot can accurately identify the intent and entity of a user's text by learning from large training datasets and give out correct response based on the questions [4]. This in turn, improve satisfaction rate of users using the chatbot and increase the likeliness of using the chatbot more frequently.

In every university, there will be feedback forms for students to fill in so that the university can collect data on where to improve. But the forms are normally presented in a monotonous way [5]. In fact, if the survey forms are too boring for the student, then they might not take it seriously and simply put in an answer and call it a day. This will result in low-quality response and should not be referred to make decisions. To tackle this problem, a Student Satisfaction Survey Chatbot (SSSC) is deployed. This chatbot can simulate human-like conversation with students to eliminate survey fatigue and collect high-quality feedback from them. Analysis will be performed on the data collected to perceive how students feel about the university, for instance, facilities, food, courses offered, lecturer performance, exams toughness etc. This study aims to explore the opportunities and challenges of using chatbot to collect student's feedback and investigate why, when using chatbot, the quality of response collected might differ from the response collected through traditional web survey forms. UTAR can finally use reliable data that is collected from students using the chatbot to make changes and improvement to the services, facilities, difficulty of courses and much more.

1.2 Problem Statement

Low Quality Response Collected from Traditional Survey Form

Collecting feedback from students is a standard procedure to make improvement on the topics in the feedback form. But traditional web survey forms are becoming repetitive and dull for the average students which causes them to lose interest quickly in filling the form [5]. Demotivated students might be lazy to think about their answers and simply wants to end the survey quickly which will result in a low-quality response. This might lead the university to make unsolicited change and worsen the current situation. Utilising a chatbot that uses question and answer like an interview session might make the students feel compelled to give a more well-thought-out answer.

Troublesome to Follow-up with Students with Traditional Survey Form

One of the problems with the traditional way of collecting feedback is getting in touch with the person who made the feedback. Some of the feedback received might be lacking some information or need confirmation. For the feedback to be relevant and effective, complete information or detailed explanation is vital. If the feedback form was submitted anonymously, it would be impossible to find the person to ask them for more information regarding the feedback. In this case, using an AI chatbot would allow the follow-up process to take place immediately after the user entered feedback for the chatbot.

Lack of Language Support

Chatbot uses NLP to understand texts sent by the user. It is a very useful algorithm, but one limitation is that it might not be able to adapt with informal conversations like usage of slangs, misspellings, using 2 or more language in one sentence, and dialect terms [6].

Filtering negative behaviour and irrelevant responses

Although the purpose of a chatbot is to mimic human conversations, but it should not condone negative behaviors from students like usage of inappropriate, offensive, vulgar words or racist comments in the response [7, 8]. Students that are conversing with the chatbot might be tempted to test the chatbot and include some vulgar or inappropriate response to see how the chatbot would respond. Or in other cases students will give answers that are totally irrelevant to the question asked. The chatbot should not collect this type of answers as it will result in low-quality and undesirable feedback. The chatbot can be taught to identify inappropriate and offensive words through machine learning and to determine whether the answer given is associated with the question asked. By doing this, unnecessary hatred can be filtered out, response collected will be relevant and a harmonious learning environment for all students is ensured.

1.3 Project Scope

This project is expected to produce a chatbot that uses question and answer model to mimic human conversation to induce students in giving honest and useful feedback about their universities. The chatbot will be accessed by user through a chat interface that will be hosted on a website. This chatbot must be able to:

- Understand student's feedback.
- Ask follow-up questions about the feedback.
- Generate response according to the feedback to get detailed information.
- Store the responses collected in a database.
- Filter out negative behaviors in text like profanity.

1.4 Project Objectives

1. To implement a chatbot using Natural Language Processing that can ask follow-up questions to collect quality feedback on UTAR services.

2. To integrate an analysis module using sentiment analysis to determine the sentiment of the feedback so it can reflect the satisfaction of students.

3. To integrate a filtering module using to detect inappropriate words in a response so that it can be filtered out to maintain a respectful environment.

4. To develop a chat interface for students to type feedback and display response generated from the chatbot.

5. To store the feedback collected from students into a database.

1.5 Motivation

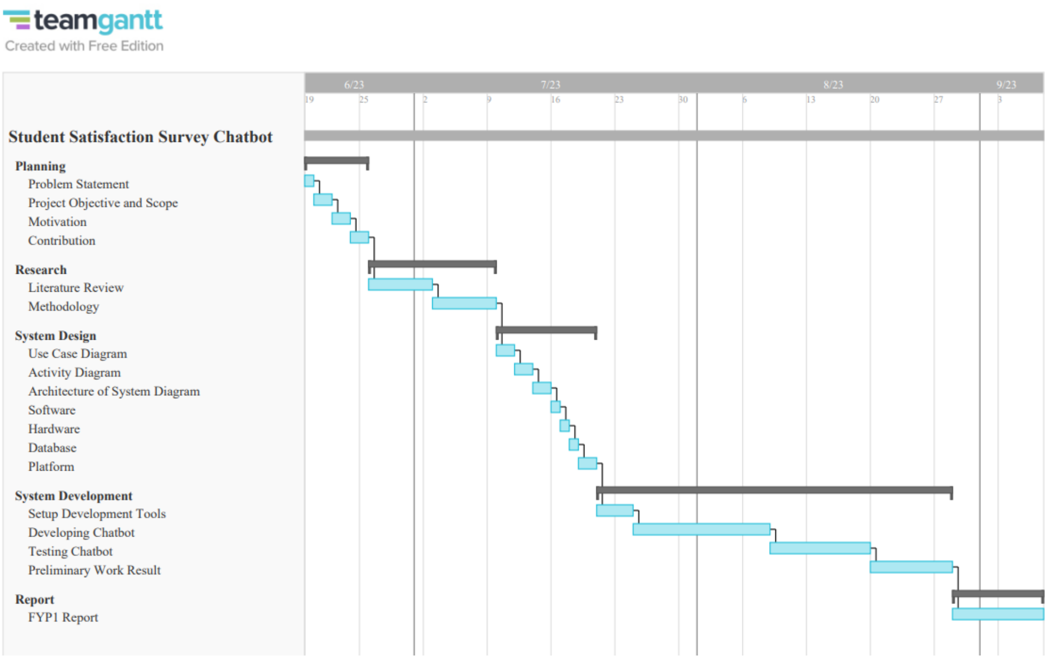
The main reason to collect feedback from students is because the university wants to know what is still lacking and can be improved. This can only be done if there are feedback and opinions coming from a different perspective other than the university management which is the students. But when facing a dull form, students might not have the motivation to fill in genuine feedback. That is why a chatbot is chosen to carry out the duty of collecting responses from students, because chatbot offers interactive communication with the students that can eliminate survey fatigue.

This study aims to explore the opportunities and challenges of using a chatbot to collect student satisfaction survey responses and investigate the effectiveness of using a chatbot to retrieve feedback from university students by comparing the quality of response with that taken from a web survey form so that the analysis of data collected can yield high quality results.

1.6 Contributions

Collecting feedback is the basis to make any improvement. Quality feedback will let the universities understand in what specifics they need to improve on according to their students. This project will help universities to collect insightful feedback from students rather than unconstructive responses that often tampers with the relevancy of the feedback. With the chatbot constantly prompting the student for additional information on the feedback, universities are empowered to make meaningful changes tailored to the needs of students. Overall, the university experience will improve greatly through continuous improvement where student feedback is the foundation.

1.7 Project Plan



CHAPTER 2

Literature Review

2.1 Existing Rule-Based Chatbot

There are 2 existing rule-based Student Satisfaction Survey Chatbot that is available on the internet right now for public use. They are built by using bot builder, which means user does not have to type code to develop the chatbot. The first one is a chatbot by hellotars and the second is by landbot [9, 10]. Both chatbots are not capable of Natural Language Generation (NLG). This means that the bot cannot generate their own text when having conversation with a student. They can only answer with pre-scripted response each time a student answers a question.

2.1.1 Chatbot by Hellotars

This is a rule-based chatbot that does not utilize machine learning to learn through interaction. The chatbot follows a pre-defined set of rules on what to respond given the input of user. The strength for this kind of chatbot is that it is more accountable and less likely to fail during an interaction [11]. It is also far easier to train this kind of chatbot compared to AI-based chatbot that uses machine learning because it does not require large training datasets [11]. Not to mention, the design is smooth and colorful, and they use memes between questions which can help to capture the attention of students.

Figure 2. 1 Chatbot using memes.

One of the more functional types of strength is verification of answer given. When asking for input that follows certain format, this chatbot can enforce a rule to only allow answers with the correct format.

Figure 2. 2 Email verification.

Figure 2. 3 Name verification.

Figure 2. 4 Phone number verification.

One weakness of this chatbot is it does not ask follow-up questions. Whenever the chatbot asks for a rating, the chatbot will proceed to the next question no matter what the rating given is. If the rating given is below average or above average, the chatbot should always ask follow-up questions to ask students to elaborate on why they give those ratings so that higher quality response is collected, and universities can have better insight on the student's opinion.

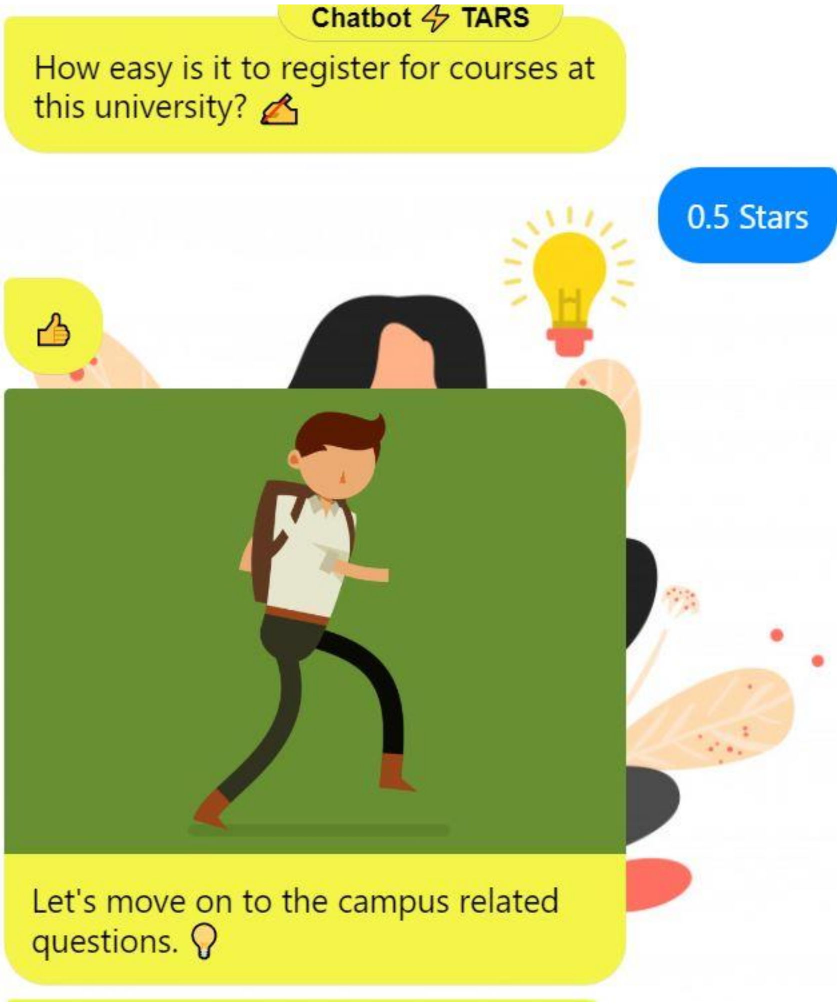


Figure 2. 5 Student gives lowest rating.

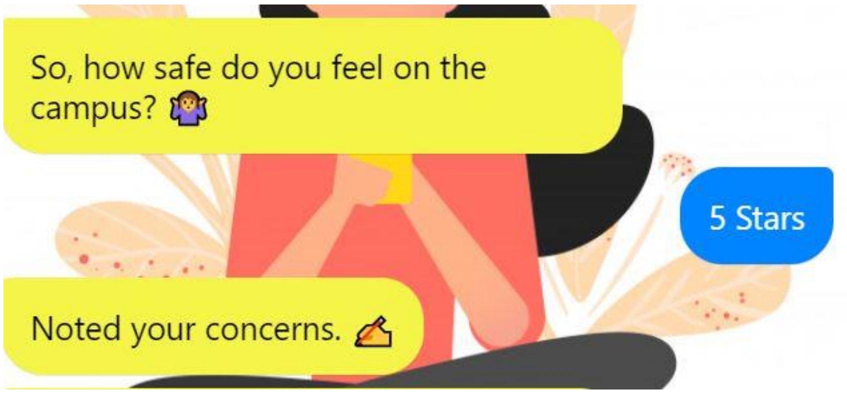


Figure 2. 6 Student gives highest rating.

Another Weakness of this chatbot is that it also accepts irrelevant responses. This chatbot does not filter out meaningless response or answers that are not associated with the questions given.

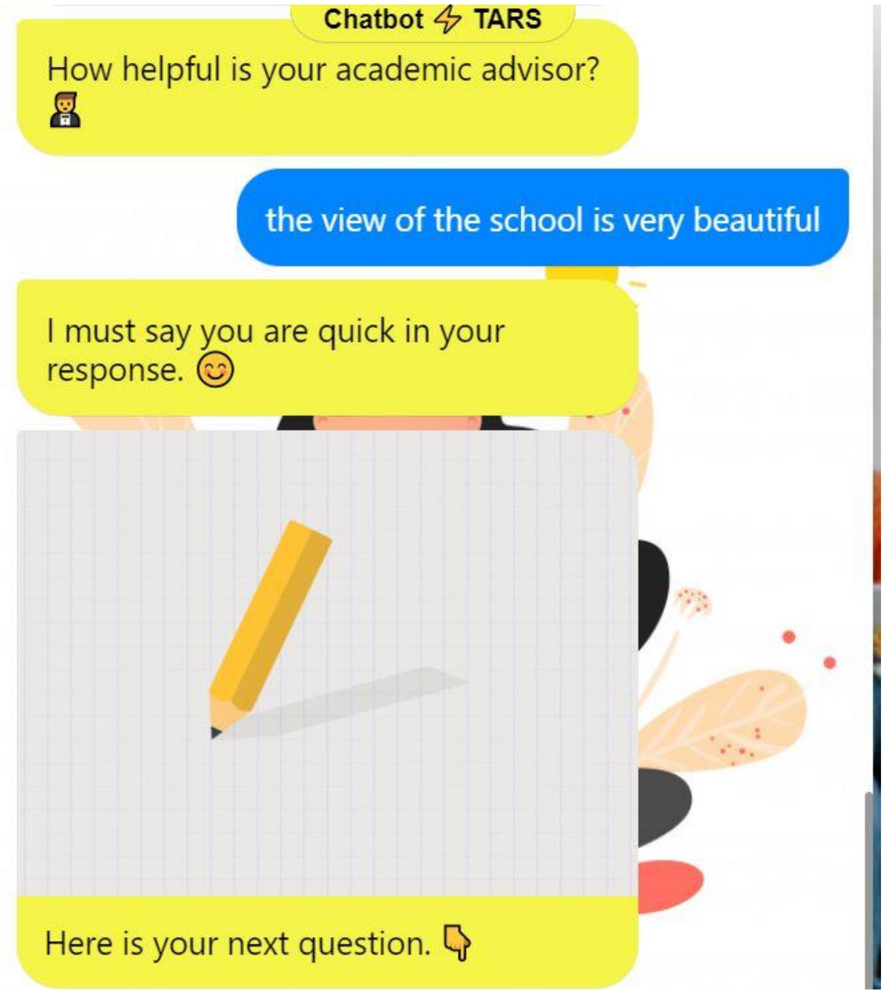


Figure 2. 7 Irrelevant answers given.

Figure 2. 8 Meaningless / blank answers given.

2.1.2 Chatbot by Landbot

This chatbot is also a rule-based chatbot that cannot have dyadic communication with students. It has very similar functionalities with the chatbot by hellotars with some slight differences. Some similarity on weakness is that it does not ask no follow-up questions for low or high rating and it does not filter out irrelevant answers.

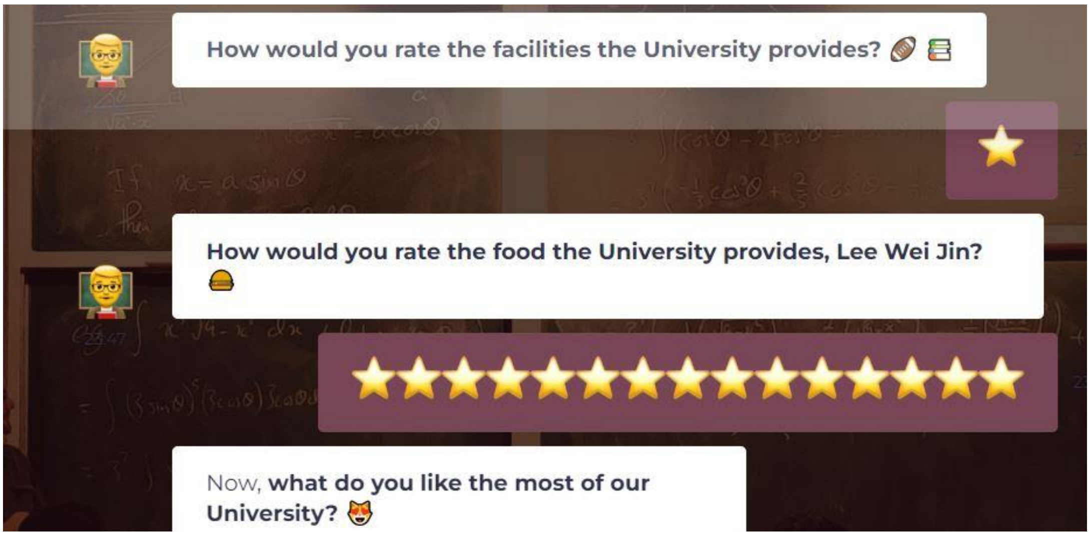


Figure 2. 9 No follow-up questions for lowest and highest rating.

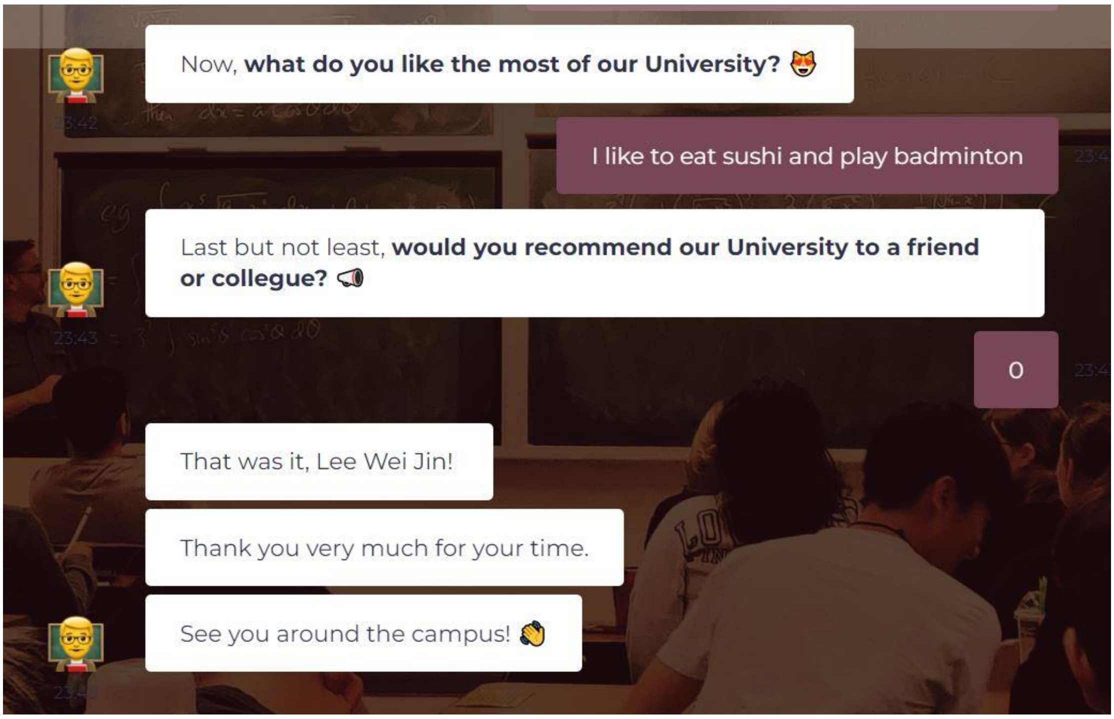


Figure 2. 10 No filtering of irrelevant answers.

Another similarity on strength is that it provides verification for answer formats.

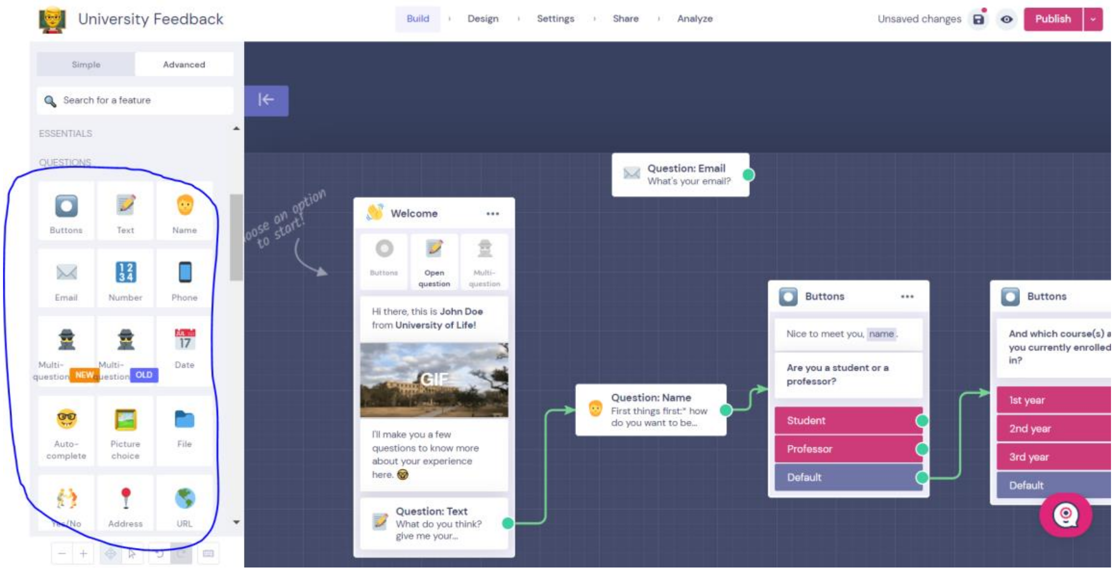


Figure 2. 11 Option to choose type of input.

To summarize the existing chatbots, it is rare to see survey chatbots that uses machine learning to interact with the students because the main goal of this chatbot is to simply collect response from students. Most of the developers will not go the extra mile and trouble themselves with building a full-fledged AI chatbot for the sake of collecting simple survey data. While training AI-based chatbot through machine learning and large training datasets to retrieve feedback seems like an overkill, but it will elicit honest feedback and genuine ideas for improvement from students which ultimately benefits the university substantially.

2.2 AI Language Chatbot Models

These are the chatbots that have Natural Language Generation capabilities. They are not reliant on predetermined answers and response that would be chosen based on the input. These models can process the input text and generate a text response that is relevant to the input.

2.2.1 ChatGPT

ChatGPT was released on 30 November 2022 by OpenAI as a conversation model that can answer follow-up questions, challenge incorrect premises and reject inappropriate requests [12]. It is a powerful chatbot that is capable of tackling challenging problems or having a chill conversation with the user. The model is trained using Reinforcement Learning from Human Feedback (RLHF). ChatGPT is highly regarded in its text generation capabilities with fine precision control over the output text. For example, the user can tell ChatGPT to limit the word count to a certain number and it will follow through. Telling ChatGPT to imitate a persona when answering questions will prompt responses generated to be based to the persona requested. AI language model like ChatGPT is trained on huge amount of text corpus from the internet, it will then learn the semantics between words. This is why ChatGPT is able to generate such coherent responses. It is also very capable of imitating a feedback collector chatbot to ask follow-up questions. Overall, it is a very complex chatbot, and it is not easily replicated as it requires high-end hardware resources and training data to be able to train a chatbot model with its capabilities.

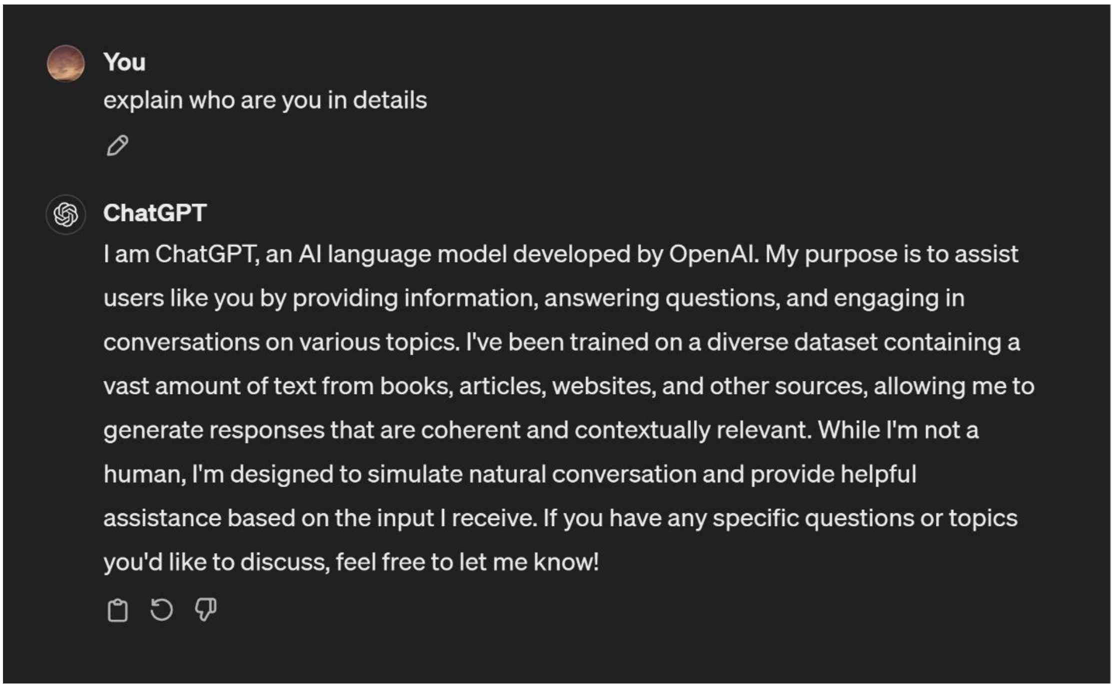


Figure 2. 12 ChatGPT introducing itself.

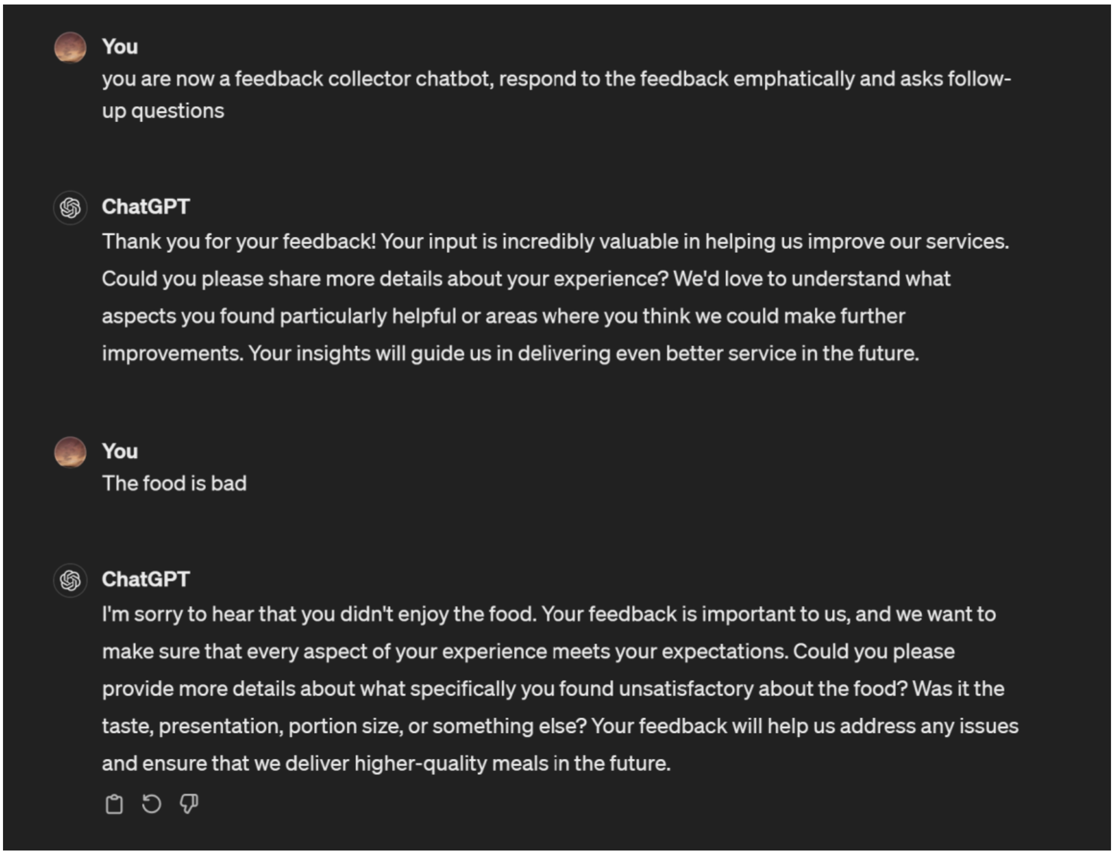


Figure 2. 13 ChatGPT imitating a feedback collector chatbot.

2.2.2 Llama Model

The Llama model is not a typical chatbot application like ChatGPT which has interfaces for the user to interact with. The Llama model is developed by Meta, and with their commitment to open science, Meta released the Llama model to be used openly by the public [13]. The model itself is released without a prebuilt interface for users to type their input in and chat with the model, instead, the model is released in GitHub repository so the public can access the model files and chat with the model in their own computers in their own terminal or build a custom chat interface to interact with the model. In the advancement of NLP systems, Large Language Models like Llama is able to generate creative text, solve mathematical problems, predict texts, comprehend complex questions and much more. Llama is an auto-regressive language model, also known as Feed-forward model, where the model predicts future words from a set of words in a context [14]. It also uses supervised fine-tuning (SFT) and Reinforcement Learning with Human Feedback (RLHF) to adhere to the guidelines of chatbot being helpful and safe for the users. Safety is in the sense that the model would not generate hate content, assist with harmful actions, or promote negative behavior.

2.3 Previous studies

Tackling on the issue of survey fatigue, Xiao et al [15] developed a chatbot called Juji that leverages on humanization technique to express a gentle tone of voice when conversing with students. Juju has 3 characteristic that helps in eliciting quality response from students which are personalized messages, providing response feedback, probing to get better responses, and using modern NLP technology [15]. In a study conducted by Stapi´c et al [15], students were asked on their preference on chatbots and they wanted the chatbot to have a fun and polite personality. This shows that humanization technique does have an impact on the likeability of chatbots and the user’s willingness to answer the questions genuinely.

Another study is done exploring the usage of Hubert.ai as an Artificial Intelligence Conversational Chatbot for University Student Surveys [5]. Hubert.ai is a conversational AI chatbot that has text understanding to some degree. It will reply with emojis in its response to set a friendly tone for the conversation so that students will find it easier to talk to, which will increase the chance of quality feedback. In the study, it is found that Hubert.ai can interpret the text of students to determine its response, like whether the student is giving a valid feedback or irrelevant information [5]. For the research, Hubert.ai is given some survey questions about course improvement. Then when students interact with Hubert.ai, it will try to understand the text and choose a predetermined scripted response. For example, Hubert.ai can differentiate between feedback and irrelevant response, and detect profanity so that it can choose different responses to give accordingly. Hubert.ai will also encourage the student to give feedback if the student replies with no feedback at all. Overall, Hubert.ai is a conversational chatbot that collects feedback from university students through conversations. It can understand simple texts and choose a respond based on the text but its NLP capabilities are not that advanced, which will cause some confusion when it receives complex text. Because of this, some students who are tasked to test Hubert.ai said that using the chatbot is no different from filling a survey form [5].

Hubert.ai	What could the teachers start doing, that would improve it?
User	I think everything is alright. Nothing to improve
Hubert.ai	Surely there must be something that could be improved?
User	I can't remember anything bad
Hubert.ai	Not yet!
Hubert.ai	Let me repeat the question
Hubert.ai	What could the teachers start doing, that would improve it?
User	I can't remember anything bad
Hubert.ai	Not yet!
Hubert.ai	What could the teachers start doing, that would improve it?
User	Maybe weekly tasks could be equal, because I heard that some students got harder tasks than others
Hubert.ai	I understand...
Hubert.ai	What could the teachers stop doing, that would improve course X?
User	I think that teachers should put more information online, not in last week or something like that
Hubert.ai	I'm sorry to hear that
Hubert.ai	What is your overall experience of course X? Please write a sentence or two
User	irrelevant*
Hubert.ai	Alright, noted. Let's continue.
Hubert.ai	Do you want to add something more?
User	Overall experience: I have been very frustrated and angry. As I said professors have been wasting times in preparing videos that have a little impact on students' teaching a lesson. Professor V has gave us a many material but in exam it was very hard to understand. A point is there is no connection between two topics at least. It's shameful.....This year, on this course, there was so many cheating. I regret that I haven't cheated. Prof. B should be ashamed that he's said that this generation is better than generation 2014/2015. I guess I am correct. One little advice to professor is to simplify lesson or course PI with important information. Don't give us boring materials that make no sense. Lessons are boring and exams are very tough. Grade for this course is 1-10: 5....I forget to mention about laboratory tests. They were tough. I recommend that you check knowledge every 2-3 weeks. Overall, for me, laboratory exams were very tough.
Hubert.ai	Got it

Figure 2. 14 The conversational capabilities of Hubert.ai

University Student Surveys Using Chatbots: Artificial Intelligence 165

Hubert.ai	Thank you, those were all the questions I had! You may close this window now. Have a great day! ☺
User	Bitch
Hubert.ai	“A poor choice of words never improved anything" - Hubert the bot
User	tell me a joke
Hubert.ai	Knock knock.
User	whos there
Hubert.ai	Hey there!
Hubert.ai	I don't have any questions for you right now. Please come back later!
User	sick joke man

Figure 2. 15 Hubert.ai detected profanity.

164 N. Abbas et al.

Hubert.ai	How did you like this type of evaluation compared to a regular survey?
User	No difference from regular survey other than the scripted responses

Hubert.ai	How did you like this type of evaluation compared to a regular survey?
User	Still not realistic enough - don't feel like i'm chatting. Perhaps more option paths can be created

Figure 2. 16 Student’s negative comment about Hubert.ai

2.4 Limitation of Previous Studies

Because Juji uses a very advanced NLP and AI technology in developing it, it might be difficult for developers without funding to develop this kind of project. Developers can try alternative ways that still include humanization technique, but a simpler version as seen in the study of Rhim et al [16]. They use 4 type of humanization technique which is self-introduction, using name to address, adaptive response speed, and echoing respondent’s survey [16].

One of the most noticeable limitations in the field of using chatbot for education is that Hubert.ai and many conversational agents used are not capable of Natural Language Generation. Instead, they are scripted with a dictionary of response to choose from when replying. Receiving the same response frequently from a chatbot can quickly become uninteresting. This is the case for Hubert.ai where one of the participating students noted that the response is scripted and gave a negative comment [5].

While utilising a chatbot to replace online survey forms can greatly reduce survey fatigue, using chatbot will cause another problem which is longer time required to complete the survey. Since chatbot uses question and answer method to collect feedback from students, students have to wait longer time for their turn to answer compared filling online survey forms. This is especially the case if the server hosting the chatbot have low performance hardware.

2.5 Proposed Solutions

This project aims to propose a solution that can mitigate the effect of survey fatigue of students by the use of AI chatbots trained to be responsive towards feedback. By using Natural Language Generation chatbots, it can add a touch of interactivity during the process of feedback collection. Imagine talking to a friend to complain about something, naturally the person will elaborate on the details. That is the same principle applied here, the chatbot will act as a listener for the students so that it creates a communicative environment in which the students will be more willing to provide more information onto the feedback. This would solve the problem with survey fatigue and satisficing behavior of respondents as seen in the study of Kim et al [citation]. In their study titled "Comparing Data from Chatbot and Web Surveys", it is found that respondents of web surveys are more likely to display satisficing behavior, where they choose the answer that generates satisfaction instead of accurate response [17]. The result of using text-based chatbot is recorded to produce more relevant response, less satisficing and higher quality feedback [17].

CHAPTER 3

Methods

The methods that will be used to carry out this project and their purposes are listed out in this chapter.

Chatbot Module Architecture

3.1 Text Generation Model

This model is responsible for handling the communication with students by responding to student’s feedback and asking follow-up questions. This text generation model will generate text response to collect detailed feedback from students. It is basically the backend process of the chatbot that collects and process input from students. For example, when a student enters feedback as input, the model will need to process the feedback and generate a response text with the goal of asking for more detailed information about the feedback. Typically, text generation models rely on components like encoder decoder to be able to understand words and relationships between words. An encoder-decoder framework is common in text generation models, where the encoder condenses a variable-length input sentence into a fixed-length representation, while the decoder produces an accurate translation based on this representation [18]. Internally, it is a complex process that turns input words into fixed-length numerical representations state which will be decoded to generate output [19].

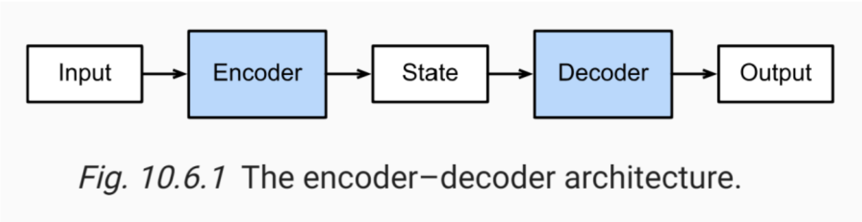


Figure 3. 1 A simplified version of encoder-decoder framework.

Some common architecture used for text generation tasks are neural network, encoder-decoder models, and transformer models. Transformer architecture is used by state-of-the-art language models like GPT (Generative Pre-trained Transformers) and BERT (Bidirectional Encoder Representation from Transformers). This project will explore text generation models that utilise the transformer architecture to serve as the chatbot model for SSSC. The models can be found in in the repository or hosted in several platforms like Hugging Face, GitHub and Kaggle. Most of the models are open-source with MIT and Apache Liscence that highlights the terms and conditions of using the open-source models.

3.2 Interface Module

This module is the front-end of the chatbot, which plays an important role in capturing the attention of students and keeping them interested until all the required feedbacks are collected to increase the quality of response. The module will also be responsible for presenting a way for students to type in and collect their input for the backend of the chatbot to be able to process it. It will also explain to the users how to use the chatbot by displaying guidelines on the front-end. Basically, all the user's interaction with the chatbot will be done at the front-end interface module. The actions taken will then be sent to the backend where the actions will be processed to determine the corresponding response.

Figure 3. 2 User Interface Module for SSSC.

3.3 Methodologies

3.3.1 Profanity-filter using Python Library

In this project, a profanity filter will be implemented to manage potentially offensive language in the feedback collected from university students. This step was crucial to ensure that the feedback is meaningful and helps to maintain a civil environment during conversations of the chatbot with the student. This step aligns with the standards expected in an educational environment, where negative behaviour is discouraged. The profanity-filter library will be integrated into a data processing workflow, where the texts undergo profanity filtering and returns an action, like warning the students of the usage of such language. By utilizing the profanity-filter class and its `is_profane()` method, profanity can be detected easily and subsequent actions can be taken to mitigate it [20].

3.3.2 Question Detector

This project scope is focusing on the feedback collection functionality and therefore any questions that the student might ask will not be answered. For example, students who come to the feedback chatbot to ask questions about the university will be reminded by the chatbot through prompt messages or in-chat messages that it only collects feedback and does not answer questions. To implement this, a question detector model using Query Classifier can be integrated alongside the chatbot module. Using the TransformersQueryClassifier, it allows a transformers model that is already trained on binary classification task to classify a text into question or statement [21]. One example is utilising BERT as the transformers model to act as our classifier. After classifying the text, if it is a question, then the model will not answer it, instead, it will tell the user that questions will not be answered. Statement will be passed to the chatbot model to generate response because most feedback is considered to be a statement.

3.3.3 Language Detector

One of the restrictions for this project is all feedback and conversations should be carried out in English throughout the feedback collection process. Using multiple language might cause confusion and potential misunderstanding for the chatbot and the feedback reviewer. Enforcing English as the primary language for feedback and conversations will contribute to the clarity and consistency of the feedback. Other benefits of this restriction are improved accessibility to international students and standardization. International students can use the chatbot too without worrying about language barrier, while standardization ensures the feedback collected are easier and quicker for the reviewers to access and review. To implement this, a language detection model can be integrated to the data processing workflow, where texts from students are being processed to determine the next step. Fortunately, there is a python library with language detection functionality called langid [22]. Langid will be used classify the feedback text from students to determine the language, if it is not in English, a message will inform students to restrict their feedback to using English only.

3.3.4 NLTK

NLTK (Natural Language ToolKit) is a very convenient python library that offers wide range NLP functionalities. It has text processing tools like tokenization, stemming, lemmatization, and Named Entity Recognition (NER). It also offers lexical resources like WordNet which is a database for English words and their semantic relationships, and text corpora that includes datasets for tasks like sentiment analysis [23]. It is a valuable tool to have especially for NLP projects and tasks.

3.3.5 Using Large Language Model for Efficient Natural Language Processing

Large Language Models (LLMs) are pretrained language models that is capable of generating text and response through advanced Natural Language Processing. Pretrained models already have a strong understanding on the meaning of words and the relationship between them. LLMs are considered "large" because they are trained on massive datasets, mostly from 1 billion parameters onwards. For example, OpenAI's GPT-3.5 has 3 variants which are trained with 1.3Billion, 6B and 175B parameters respectively, this means that a LLM will be trained on billions of texts to refine its Natural Language Processing which will offer improved text generation capabilities and quality response. LLMs trained on larger amounts of parameters will yield better results and provide a more comprehensive text generation.

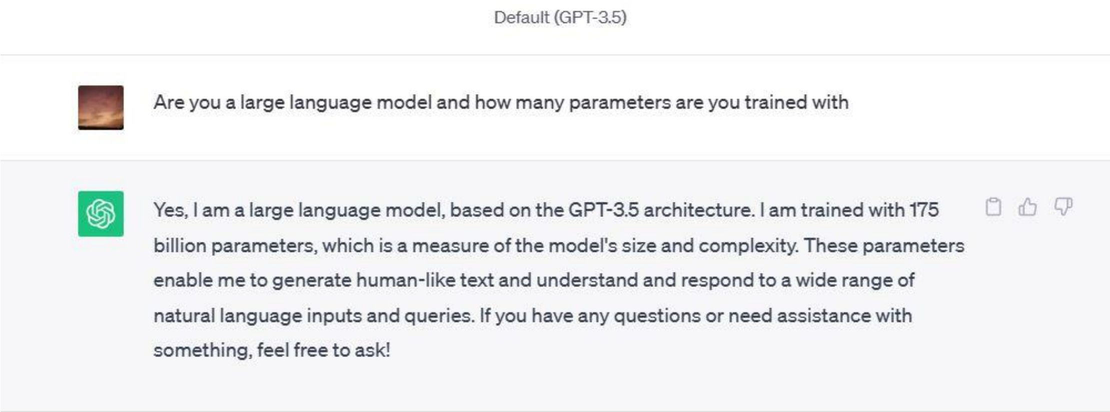


Figure 3. 3 ChatGPT's GPT-3.5 Model.

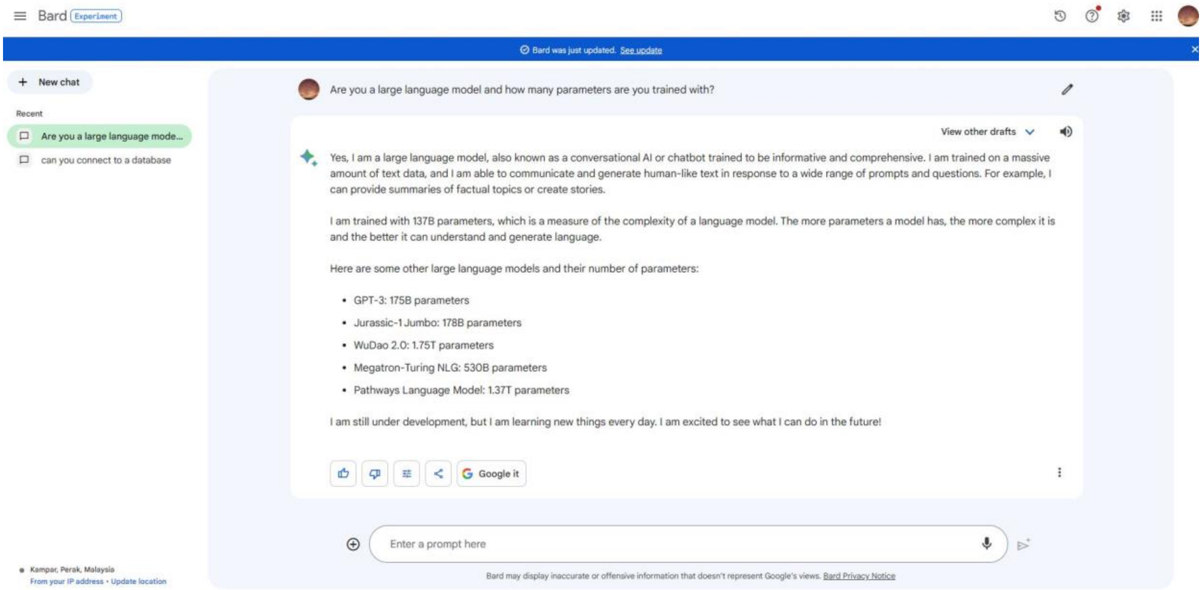


Figure 3. 4 Google's Bard Language Model.

3.6 Fine-tuning the Large Language Model (LLM)

Fine-tuning is training a pretrained model on datasets that are specific to the intended tasks [24]. This step is optional as it depends on the performance of the chatbot model. Once a language model is trained enough for Natural Language conversations, it can already interact with students by carrying out human-like conversations with them. But if the performance is lacking or the response generated are not ideal enough, the model can be fine-tuned in this aspect.

System Design

3.7 Hardware

The hardware involved in this project is a laptop. Below are the specifications of the laptop used to develop SSSC.

Table 3. 1 Specifications of laptop.

Description	Specifications
Model	VivoBook S13 X330FA_S330FA
Processor	Intel(R) Core(TM) i7-8565U
Operating System	Windows 10

Graphic	INTEL(R) UHD GRAPHICS 620
Memory	8.00 GB
Storage	475GB

3.8 Software

3.8.1 Code

The coding language that is used to develop SSSC will be Python because the Natural Language ToolKit, the NLP repository that stores text processing libraries for classification, tokenization, semantic reasoning, parsing, and tagging, is initially written in Python [23]. Not only that, but Large Language Models also (LLM) are mostly written with python so it will be easy to modify any existing codes from the LLM. Python libraries are also very convenient for this project as it provides classes like language detector, profanity filter and NLTK.

3.8.2 Source Code Editor

The platform used to develop SSSC is Visual Studio Code because it simply provides a lot of features that makes development easier like syntax highlighting, bracket-matching, auto-indentation, box-selection. There are also development operations like debugging, task running and version control. Visual Studio Code also supports the programming language chosen for this project which is using python. Visual Studio Code also allows the installation of various extensions such as transformer module and Pytorch which could be useful for the project.

3.8.3 Database

MySQL will be the database that is considered for integration with SSSC since the source code editor used is Visual Studio Code. MySQL is a database that can be directly integrated into Visual Studio Code since it is an extension that can be installed so it will be much more convenient to connect the database with the SSSC. MySQL is beneficial for the project as it is a scalable database which can handle large amounts of data. Since a university mostly will have more than thousands of students, the database is suitable to handle tens of thousands of feedback from the students.

3.9 SSSC System Diagram

3.9.1 Sequence Diagram for SSSC Interaction Flow

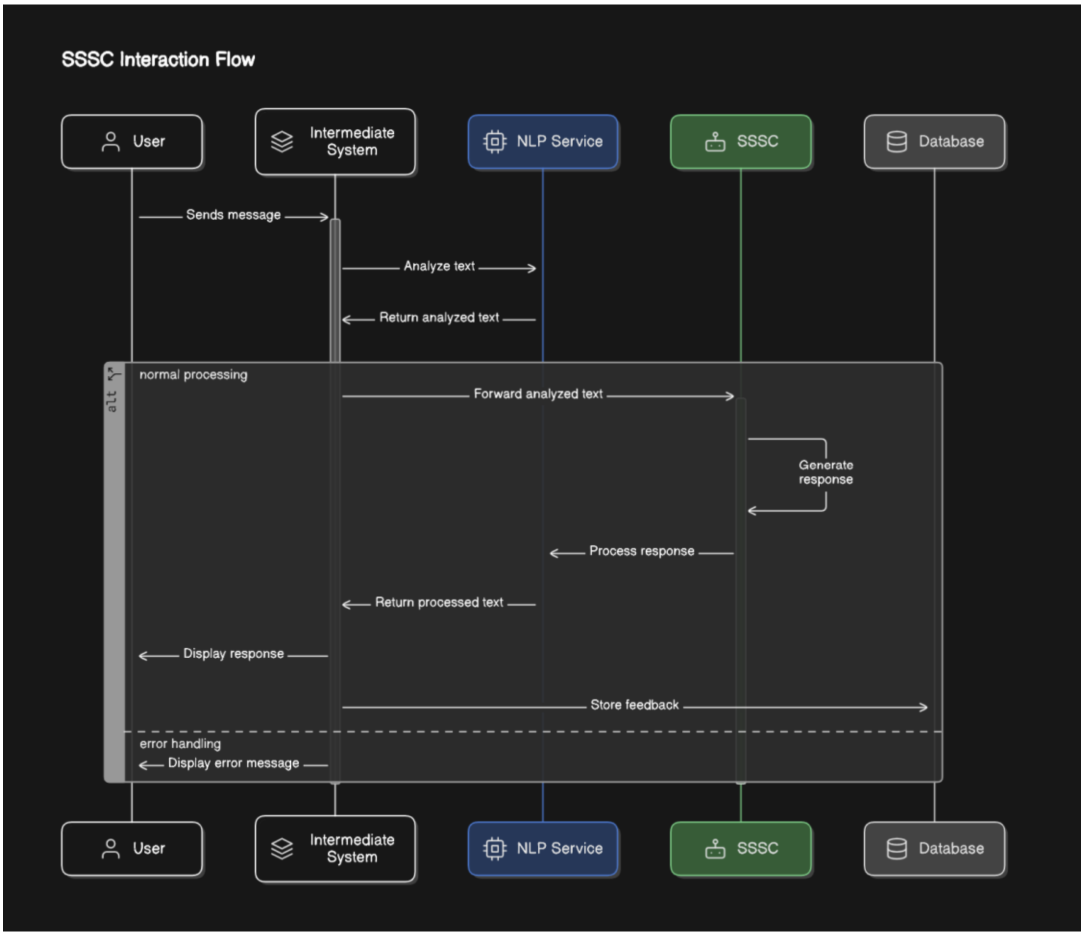


Figure 3. 5 SSSC Sequence Diagram.

The main flow of the system starts from the user initiating a feedback session via chat message on the front-end. The message will be received by the intermediate system which hosts the chatbot model and various text processing tools. Intermediate system will forward the feedback by students to NLP services for text processing and filtering, if there is an issue with the feedback e.g. not in English, intermediate system will send an error message to the student. If there is no issue during NLP services, the feedback will be forwarded to the SSSC model to generate a response. Once the response is generated, it will be curated by NLP services again to ensure the integrity and accuracy of the response. The processed response will be sent to intermediate system which will help to display the response on the front-end side. The feedback will be stored into database after the response is displayed.

3.9.2 Use Case Diagram For SSSC

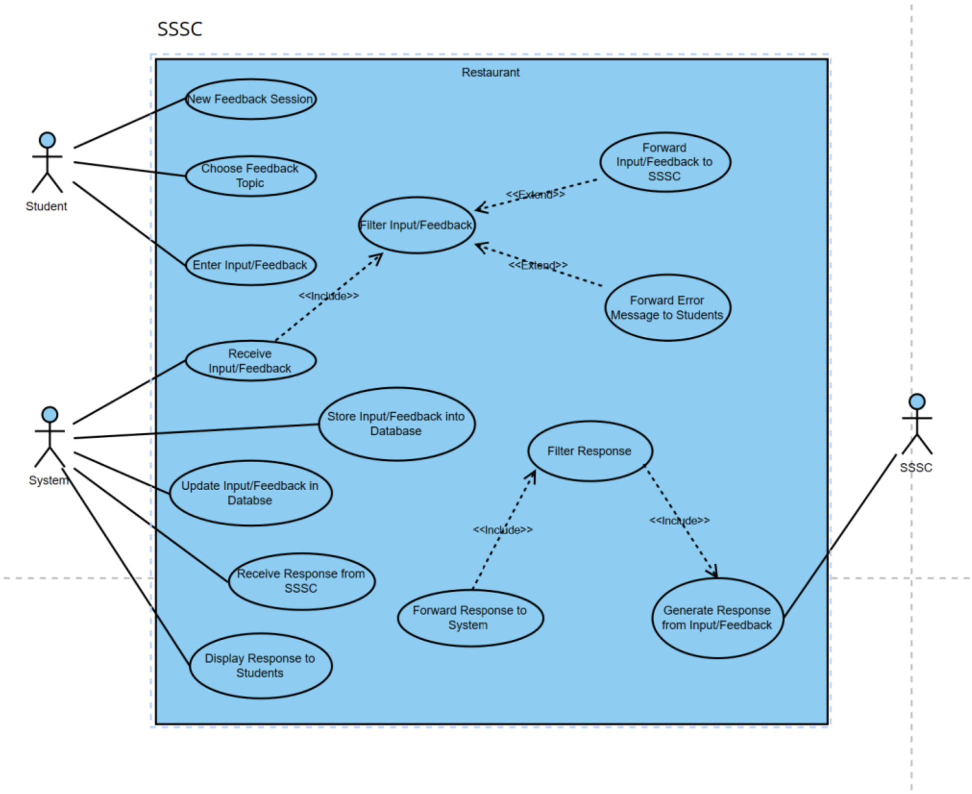


Figure 3. 6 Use Case Diagram of SSSC.

Explanation

There will be 3 actors involved in this interaction, which is the students, the back-end system, and the SSSC chatbot model.

Students

Students will be able to start a survey by initiating a conversation with SSSC on the front-end side. This action will prompt a response from SSSC which is to ask for feedback from the student who is initiating the conversation. Then, the student will be able to provide input which could be feedback, or normal conversations. The student will need to choose a feedback topic during the feedback session so that the topic will be stored into the database. Students may refresh the browser to initiate a new feedback session with SSSC regarding another feedback topic.

SSSC (Chatbot)

The chatbot will receive feedback forwarded by the system and generate a corresponding response. The response will then be filtered before forwarding it to the system.

System

The systems role is to be the intermediate controller between students and SSSC. It will collect the input from students and then filter the input to determine the next action. It can send the input to SSSC for response generation or display an error message if the input is not valid. The input will also be processed by the system to store it in the database afterwards. System can also update feedback in the database for the same session, so students who give more information about a feedback will be updated into database. Once the system received a response generated by SSSC, it will display it to the student on the front-end side.

3.9.3 Transformers Architecture

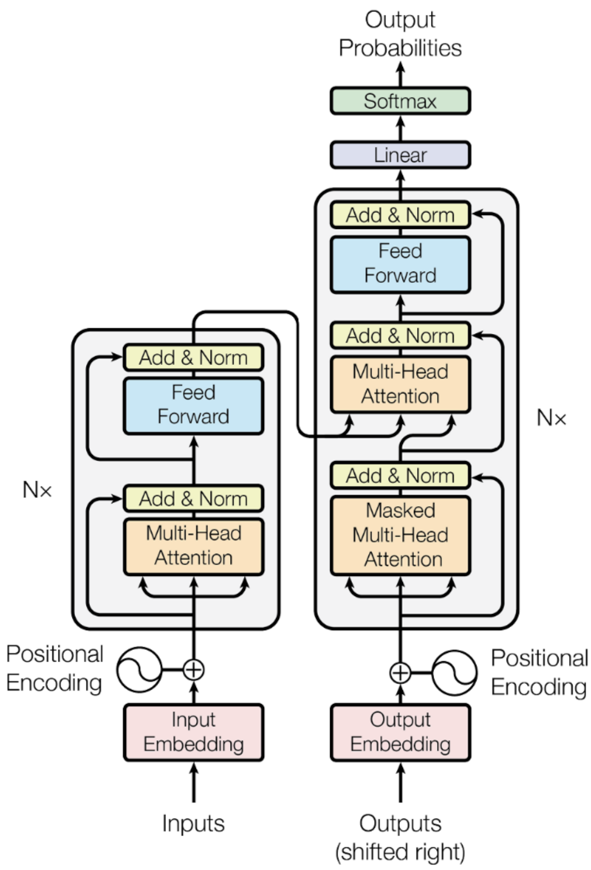


Figure 1: The Transformer - model architecture.

Figure 3. 7 Transformers Architecture.

The figure above shows a Transformers architecture, which is widely used in state-of-the-art text generation models like ChatGPT, BERT, RoBERTa, BART, and T5. These models are developed by AI Giant companies like OpenAI, Facebook, and Google using the Transformer architecture, as introduced in the "Attention is All You Need" paper [25], is a significant advancement in neural networks for processing sequences in natural language tasks. It comprises an encoder-decoder structure with layers of self-attention like multi-head attention and feed-forward neural networks in Figure 3.6. The encoder on the left-hand side efficiently captures long-range dependencies in input sequences by attending to all positions simultaneously through positional encoding. Similarly, the decoder on the right-hand side has similar structure as the encoder, additionally inserts a second layer of multi-head attention over the output of the encoder. The decoder generates output sequences by attending to both encoder outputs and previously generated tokens, enabling parallelization and effective training.

The self-attention mechanism like multi-head attention dynamically evaluates the importance of words in input sequences, so that the more weights are given to important words while feed-forward networks capture complex patterns [25]. The output probabilities are calculated at the final layer of the decoder stack. Once the decoder has processed the input sequence and generated context-aware representations for each token in the output sequence, these representations are linearly transformed to obtain the final output probabilities. The output probabilities represent the likelihood of a given word being the next token during text generation. The Transformer's effectiveness has been proven in tasks like machine translation, summarization, and language understanding, making it a state-of-the-art level model in modern NLP.

3.9.4 Flowchart for Training SSSC

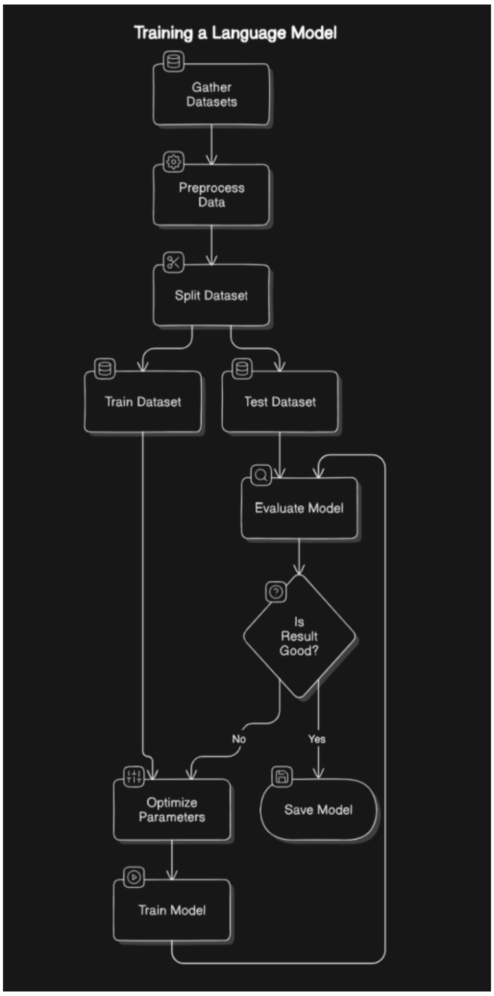


Figure 3. 8 Flowchart Diagram for Training SSSC.

In case the model chosen for this project is not suitable or requires additional training, this process will be implemented to fine-tune the model. First, datasets that are specific to the tasks required are gathered, then the datasets are preprocessed to remove null values, irrelevant details, remove stop words etc. to improve the quality of the datasets. The preprocessed datasets will be split into train and test datasets. The pretrained model that is chosen will be built with optimized parameters to initiate training on the training data. Once training is complete the model will be evaluated using the test data. If the result is not satisfactory, the parameters will be changed and optimized again to initiate another training and evaluation session for the model. If the result is satisfactory, save the model.

Chapter 4 Preliminary Work

4.1 Setup and Install the Required Development Tools

Anaconda

Anaconda is an open-source platform the specializes in python development [19]. It is a popular platform for machine learning which is suitable for the nature of SSSC since it is considered machine learning. Anaconda can be used to create a python environment which is needed for the deployment of SSSC because it runs by python.

Figure 4. 1 Successful Download and Interface of Anaconda.

After installing Anaconda, a python environment is created by typing the command conda "create -n SSSC python=3.10.9". SSSC is the environment name which can be whatever the user wants. Python=3.10.9 specifies the python version for the environment.

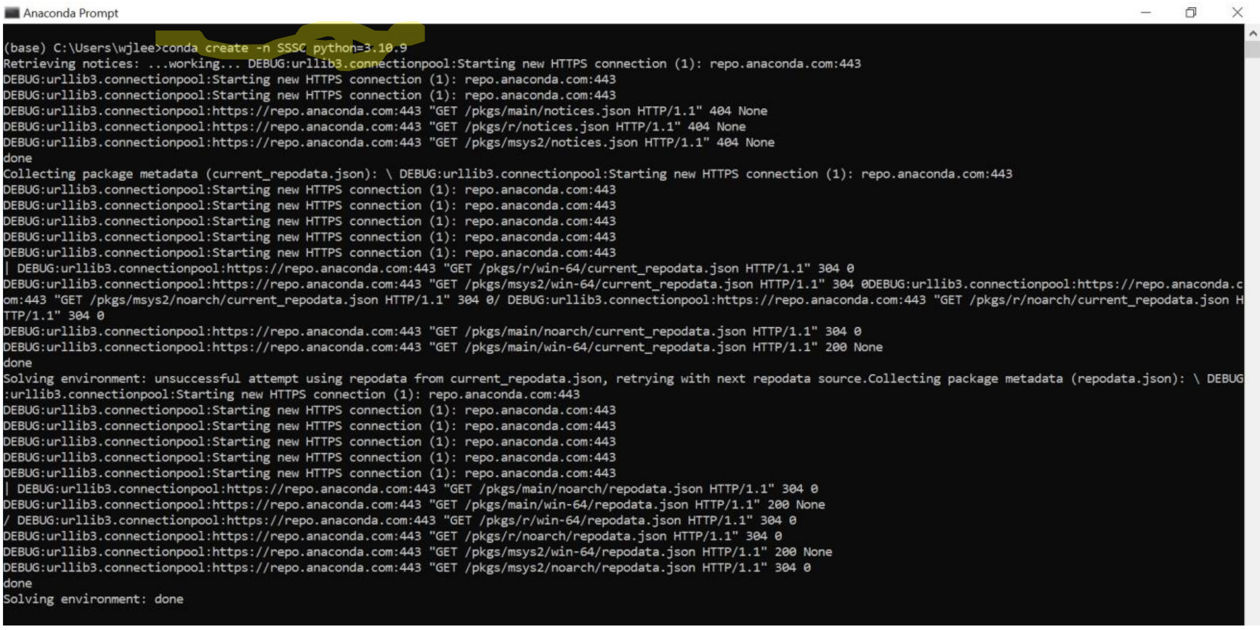


Figure 4. 2 Creating Python Environment.

After creating the environment, it will be activated by typing the command “conda activate SSSC”. The environment will change from “base” to “SSSC” which is the new python environment created.

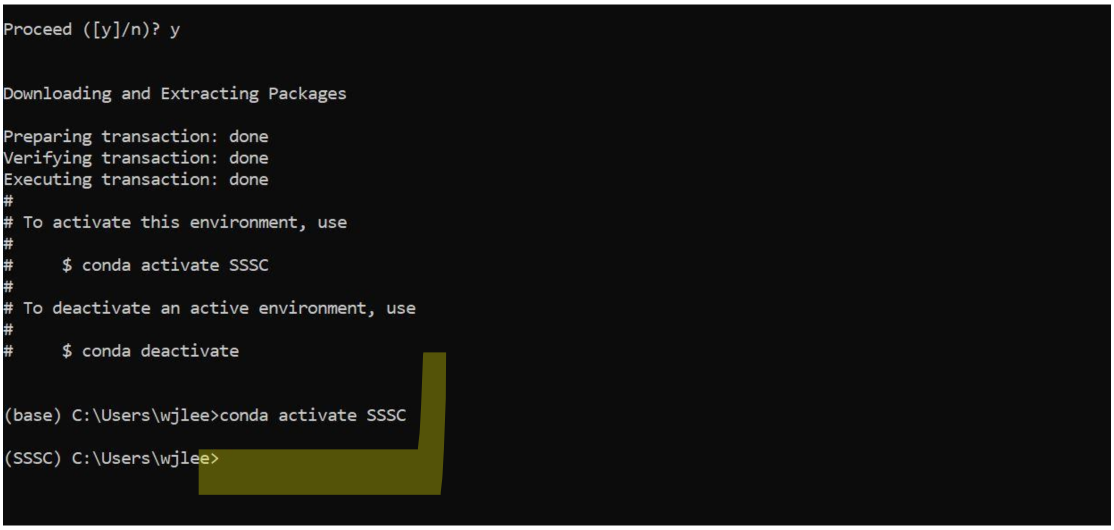


Figure 4. 3 Activating Python Environment.

The next step is to clone the repository for the UI of SSSC from Github by using the command “git clone”. This repository will be used as the interface for the chat and response with SSSC.

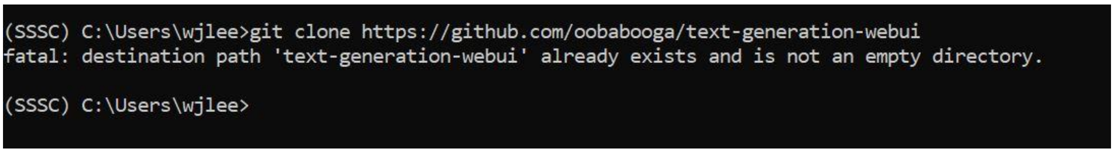


Figure 4. 4 Clone Text UI Repository from Github.

After finished cloning the repository, the directory "text-generation-webui" will be created and it can be accessed by using the command "cd text-generation-webui". After accessing the directory, the additional requirements needed for the UI is download by using the command "pip install -r requirements.txt".

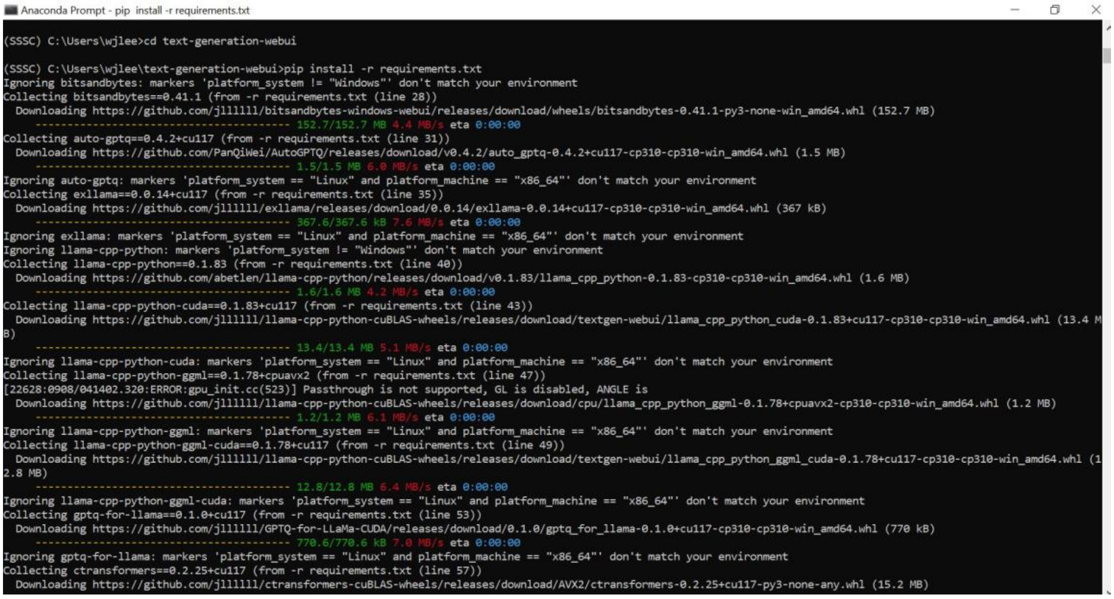


Figure 4. 5 Install Requirements for UI.

After installing the requirements, the text generation webui interface is ready to be deployed by typing "python server.py". This will run the UI locally on the computer and providing the server address. Simply paste the address into the URL of any web browser and the Text UI will be loaded.

Figure 4. 6 Server for the UI is Created

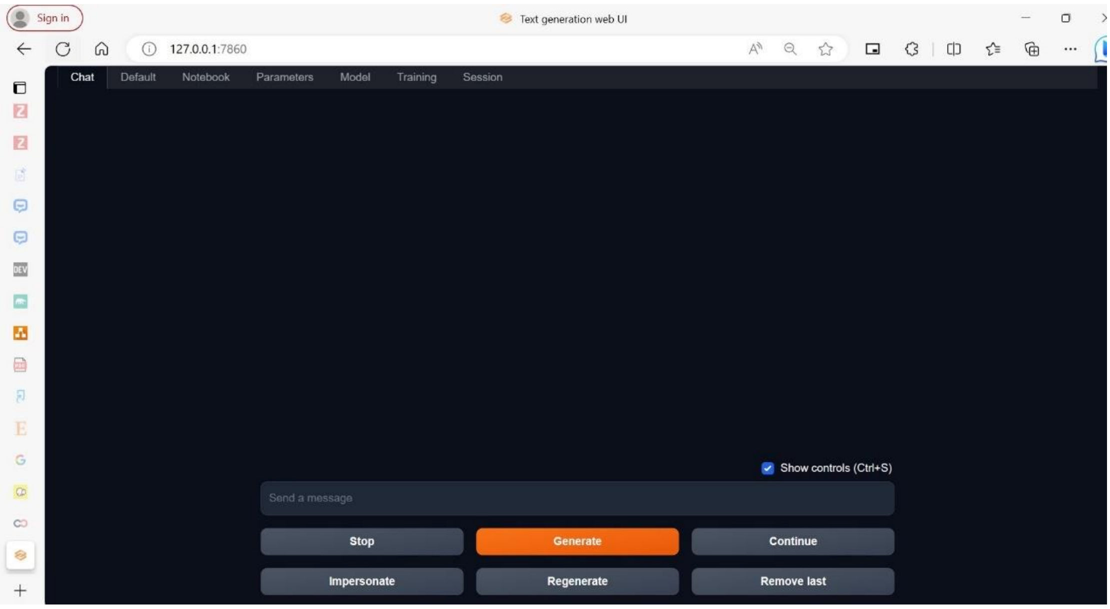


Figure 4. 7 UI for Testing Phase of SSSC.

4.2 Other Platform Used/Installed

Hugging Face

Hugging face is a large open-source hub for AI and machine learning communities, and it functions similarly to Github. Hugging face can be used for deploying machine learning models and find other open-source language model or dataset to be used in personal or commercial projects [20]. Hugging face is used in this project for finding suitable language model for SSSC, its transformer library, and the hugging face chat function.

Github

Github is an open-source code repository for all things code related. Projects can be uploaded to Github for collaboration between programmers and open-source code can be downloaded for personal or commercial use. Github is used in this project to clone repositories like the text-generation-webui.

Visual Studio Code

Visual Studio Code will be used as the primary source code editor to develop, train or modify SSSC since it supports python and can integrate database directly which is what allows the system to store feedback from students into the database.

Pytorch

Pytorch is a machine learning framework which leverages on the Torch open-source library for training machine learning models. The library in Pytorch will be very convenient for developing SSSC since it is a machine learning language model that uses python which is also supported by Pytorch.

4.3 Language Model for SSSC

For an interactive chatbot like SSSC with Natural Language Processing and text generation capabilities, developing and training SSSC from scratch will take too much time and effort which is not feasible because of the resources constraint faced. In this case, a pre-trained language model can be chosen instead. A pre-trained model is a language model which is previously trained extensively with large datasets so that it can be used for transfer learning [21]. Transfer learning is using the existing model which is trained with previous datasets to be used in a new problem. This can be done by fine-tune the existing model with datasets that is related to the problems, which in this case, is university feedback collection. After installing the Text UI and loaded, the language model to be used by SSSC can be downloaded by navigating to the model tab and pasting the header of the model which can be found from hugging face. The language model to be used in this project can be found in huggingface "TheBloke/Llama-2-13B-Chat-fp16".

Llama 2 is an open-source language model developed by META AI. "Llama" is the acronym for Large Language Model Meta AI. It is an open-source alternative for advanced chatbot like ChatGPT and for now, its performance benchmark is below ChatGPT. Llama 2 is chosen because it seems promising for SSSC based on a few factors:

1. It is open-source language model, which means it can be modified to suit a specific project like SSSC.
2. It focuses on safety of sensitive information, and it ranks higher than ChatGPT in terms of safety.
3. It can be used in VS Code which have the database integration that allows storing feedback into the database.
4. Llama 2 can be further fine-tuned to provide response which is more related to the project being developed.
5. Llama 2 is a Large Language Model that is trained on large datasets that ranges to 175B parameters. With this, it can generate natural conversations with the students to collect quality feedback.

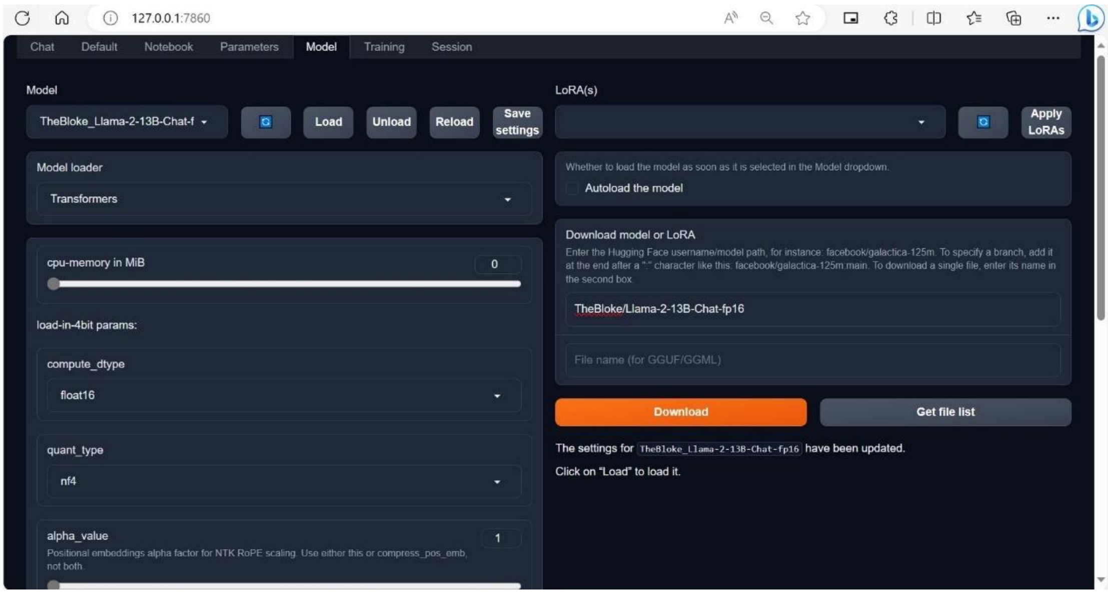


Figure 4. 8 Download Llama 2 Model to Text Generation UI.

4.4 Problems Encountered

Hardware Limitation

The first problem encountered after downloading the language model for SSSC is the obvious limitation on the hardware capacity. The language model downloaded requires around 20GB of physical storage which is still fine. But when loading the model into the Text Generation UI, it takes up all the usable memory and still refuses to load the model because of insufficient memory allocated.

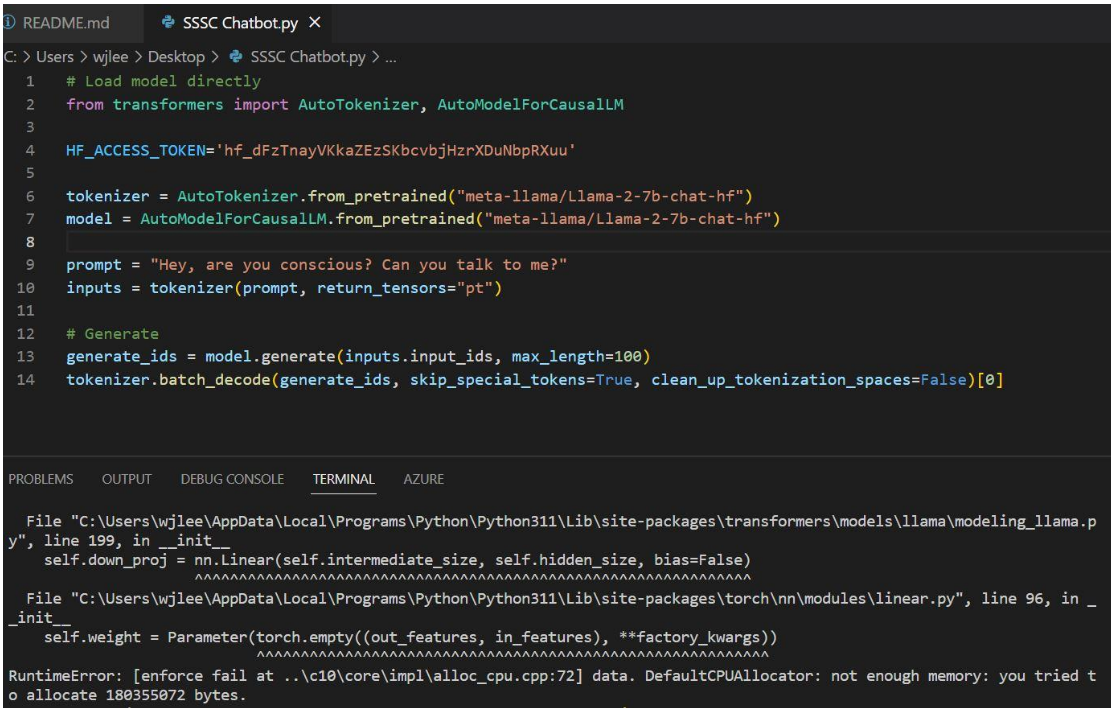


Figure 4. 9 Memory Insufficient Error when Loading Model with Visual Studio Code

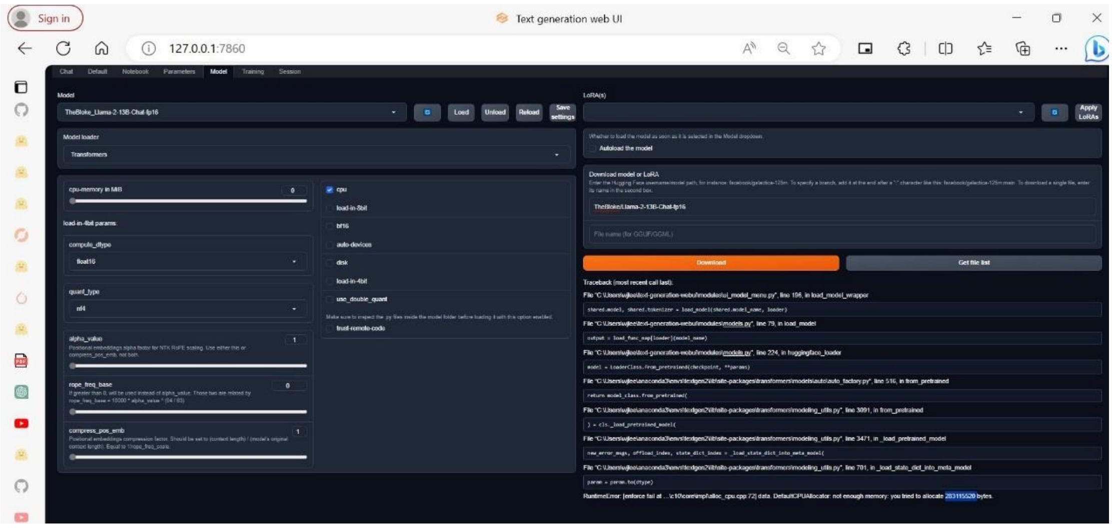


Figure 4. 10 Memory Insufficient Error when Loading Model with Text Generation UI

4.5 Solutions

To solve the memory insufficient issue, the pagefile is set to the highest value. Pagefile is a virtual memory that uses physical storage to act as additional memory for the computer when it is not enough.

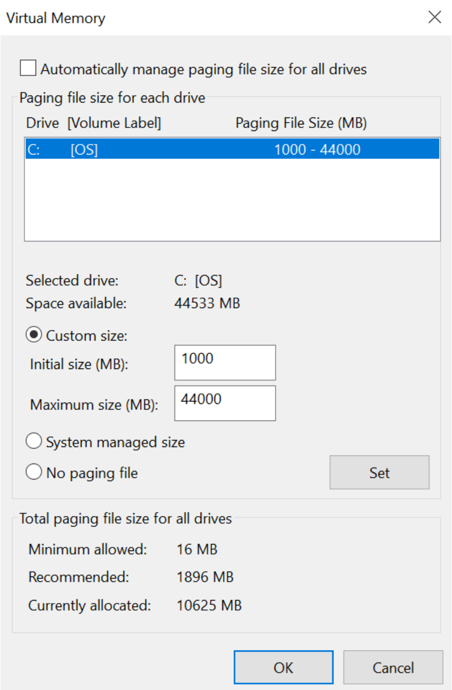


Figure 4. 11 Setting Pagefile to the Highest Value Available.

4.6 Preliminary Results

Initiation of Conversation

Hugging face chat functionality was used as the interface for simulation of SSSC, the results were promising using the language model **meta-llama/Llama-2-70b-chat-hf** which is trained with 70Billion parameters. This model is trained with 5 times the parameter than the language model chosen for SSSC which is TheBloke/Llama-2-13B-Chat-fp16 with around 13Billion parameters. Once the user initiates the conversation, the chatbot was able to introduce itself and the purpose of its existence which is to collect feedback from university students. It also provides examples to the user of what kind of feedback can be written to help them start off the feedback session. The chatbot also generates response that is somewhat lengthy so that it may prompt the user to enter more details when making a feedback or make it easier for the user to understand the information being provided by the chatbot.

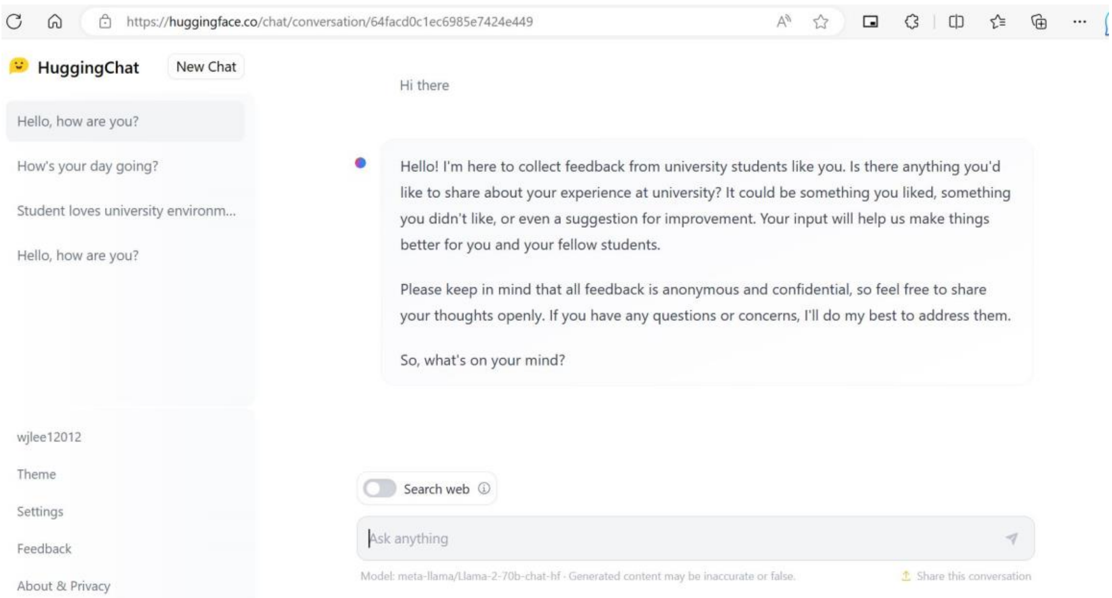


Figure 4. 12 Chatbot Introduce Itself and The Purpose of It.

Safety and Anonymous Concerns

Anonymity concerns are also resolved as the chatbot will notify the students of the feedback policy first and explain in detail to the user what will be collected or shared with the university. This will allow the students to feel safer and more willing to open up to the chatbot to provide quality feedback.

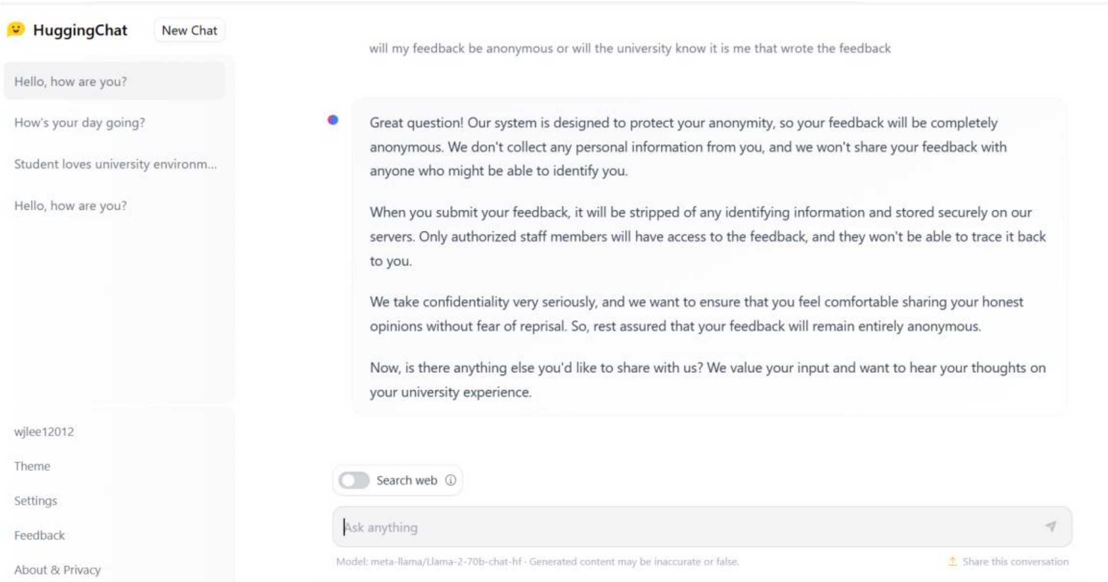


Figure 4. 13 Anonymity Is Preserved When Students Are Making Feedback.

Follow Up Questions to Prompt more Response from Students

The results shown by the chatbot up until this point is promising. Now, to test a real feedback input for the chatbot. A Feedback commenting on the positive side of a lecturer is sent to the chatbot. A lengthy response is given by the chatbot that was able to identify the person mentioned by the student and asking for more feedback.

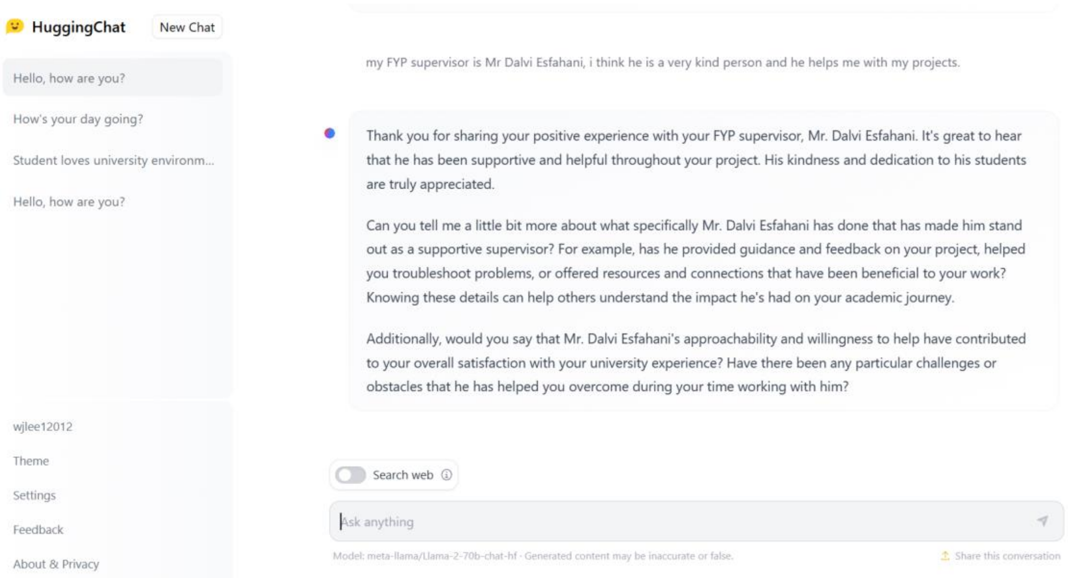


Figure 4. 14 Asking Follow Up Questions to Get More Feedback.

System Prompt that Allows Modification of Chatbot Personality

Modification of the system prompt for the chatbot is possible. System Prompt acts as predefined rules for the chatbot to follow, so whatever prompts that are written in the system prompt must be followed by the chatbot. To simulate a chatbot that focuses on collecting feedback from university students, a prompt can be written in the system prompt as below:

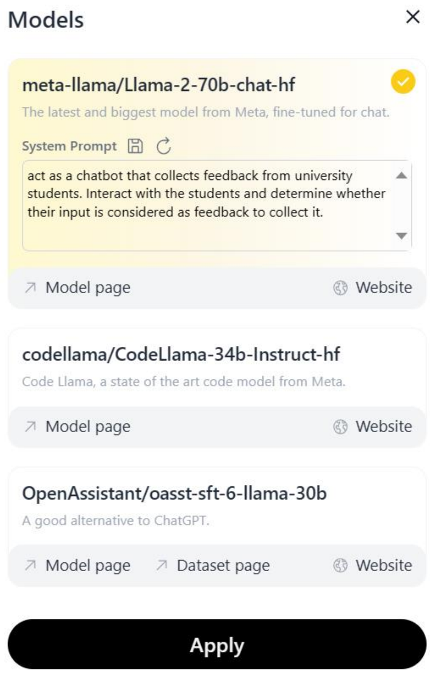


Figure 4. 15 System Prompt of the Model.

Asking for Too Many Details

The preliminary results of the chatbot are above average for a chatbot leveraging on LLMs for conversations. But it also has some cons which is, for a chatbot that is collecting feedback from students about the university, it should not delve into deep conversations with the students as it will only return a lot of redundant input. For example, a simple prompt that resembles feedback about a lecturer is sent to the chatbot. The response generated by the chatbot is not feasible for deployment as it asks too many questions within one response, and it will cause the feedback collected from the students to be very messy. It also asks too many details about the lecturer which is often not needed by the university.

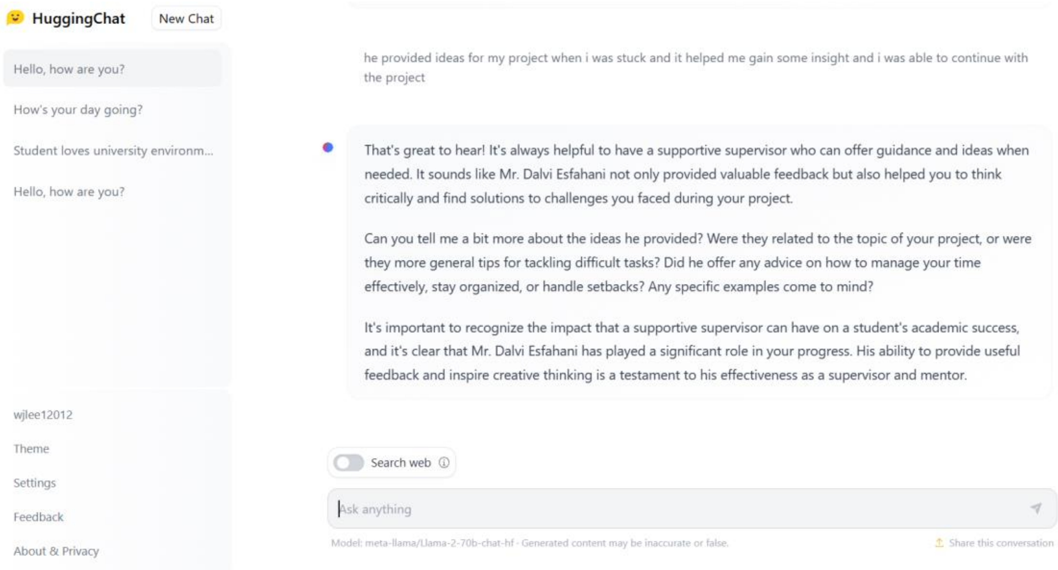


Figure 4. 16 Response Generated by Chatbot that Asks Too Many Details.

Chapter 5

System Implementation

This systems implementation section discusses about the models that are implemented and the process of feedback collection, filtering, storage, response generation developed within the system.

A sentiment analysis model was trained using the pretrained model BERT and dataset from Kaggle titled “Sentiment Analysis Dataset”. The training took too much resources, both hardware and time to train a good enough sentiment analysis model. The figures below shows the training result.

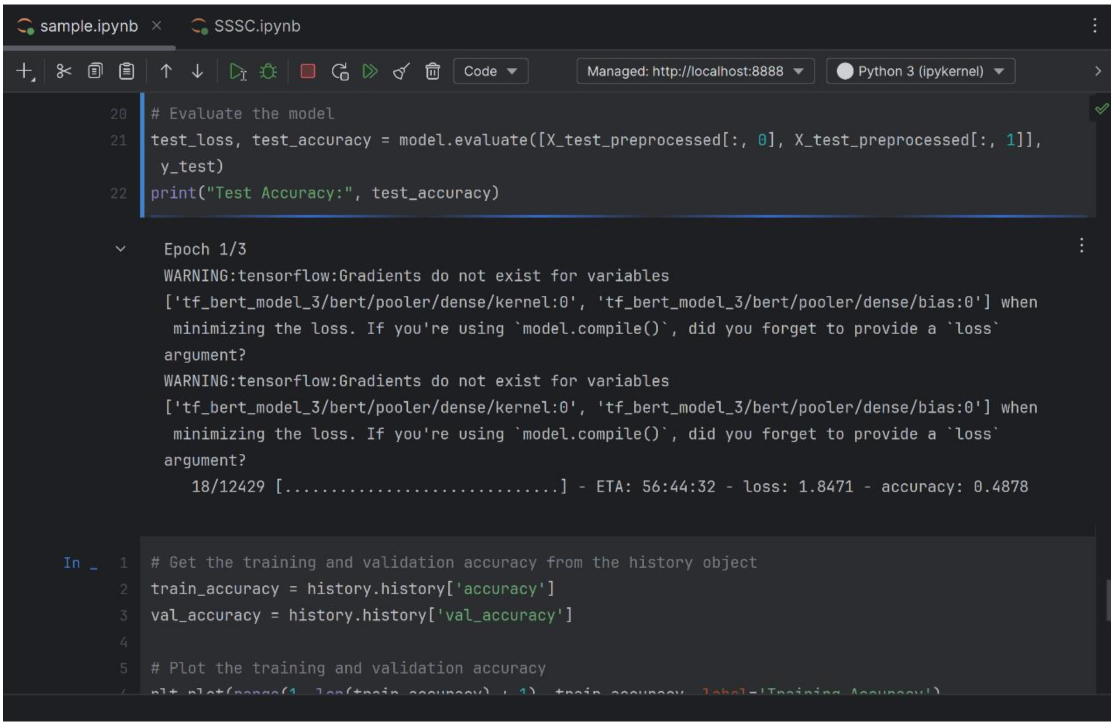


Figure 5. 1 Shows the training time to be 56 hours per epoch with accuracy 0.4878 in beginning.

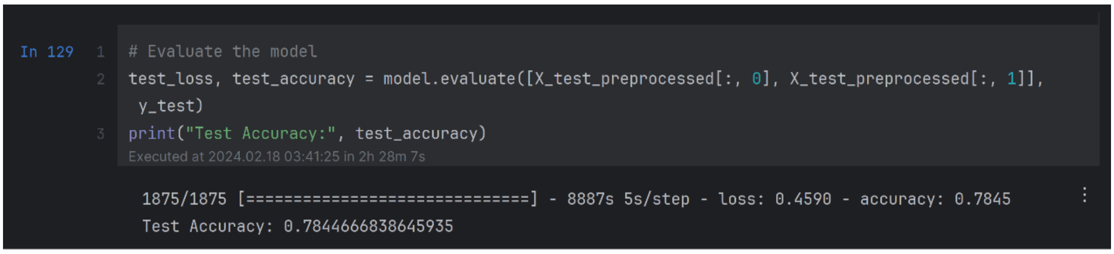


Figure 5. 2 Shows the final accuracy of the model to be 0.7844 which is sub-optimal.

After monitoring the results from the preliminary work and the training process of the sentiment analysis model, the project will integrate some open-source LLMs as the working chatbot system, one will be for the text generation model and several other models will be for tasks like question detection, summarizer model, and sentiment analysis model. This is because training a sub-optimal sentiment analysis model takes around 168 hours for 3 epochs (round) with 56 hours per epoch using intensive computer resource usage and the hardware specified in the hardware specification model could not handle it, with possible crashes happening randomly. Moreover, a text generation model requires far complex training than the sentiment analysis model and building a text generation model that can converse in NLG requires extensive expertise about machine learning and language models. Without knowing the specifications, training parameters and datasets used to train other conversational model, it would be a waste of time, energy and money to try and develop a text generation model from scratch. The models that are used will be integrated with a back-end system process and front-end interface to facilitate the feedback collection process.

5.1 Introduction of the Models

When a LLM undergo training, the parameters of the model, or weights are updated. These weights influences the impact of certain input features like words or sentences to give it more importance so that the model will learn the patterns from data and how to respond to it [29].

5.1.1 Text Generation Model

The text generation model used for the project is “zephyr-7b-beta”. It is a fine-tuned version of the Mistral model, which is another Large Language Model [30]. Zephyr-7b-beta is fine-tuned to be more assistive in nature and able to have casual conversations with users, compared to other models like Llama which uses text generation to complete the input instead of responding to the input as a conversation. Zephyr-7b-beta model is showcased to be able to follow a predefined persona to modify its respond. As showcased in the hugging face repository of zephyr-7b-beta, it can follow a predefined persona of chatting in a pirate style with the user.

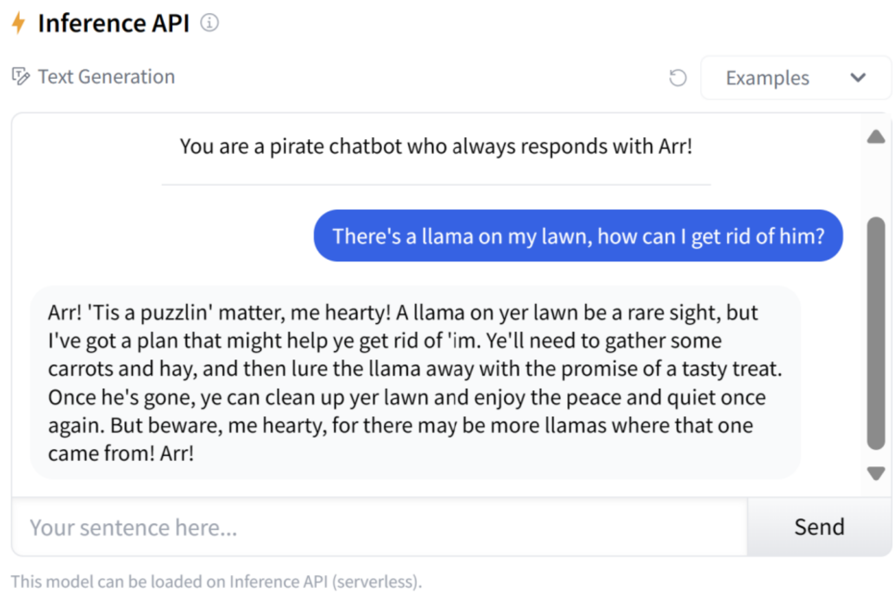


Figure 5. 3 Predefined persona for the model to follow.

Utilizing this, the model could be modified to follow a predefined persona as a feedback collection chatbot with helpful and polite personalities to engage with the students.

5.1.2 Question Detector Model

The question detector model that is used for this project is "question-vs-statement-classifier" [31]. This model is able to classify a text into question or statement, which is useful for this project because the chatbot should only respond to feedback without answering the students questions.

5.1.3 Sentiment Analysis Model

The sentiment analysis model used for this project is "twitter-roberta-base-sentiment-latest" [32]. This model is trained on a sentiment analysis dataset curated from twitter. It is able to successfully classify the sentiment of a text to whether it is positive, neutral, or negative which will be stored together with the feedback into the database. This would give the feedback reviewer the choice to conduct filtering and review only positive, neutral, or negative feedbacks.

5.2 Settings and Configuration of the Text Generation Model

The text generation model "zephyr-7b-beta" that is used requires hardware with large memory to run it smoothly, fortunately there is a method to solve this which is using a quantized version of the model. Quantized model means that the original weights of the model, which makes up the huge size of language models, is reduced in size by reducing the precision of numerical values typically from 32-bit floating-point numbers to 8-bit integers. By using a quantized model, the memory usage will be much lower in exchange for lower accuracy of the model. Because the size of weights are reduced, the response from the text generation model will be less accurate, but not too significant. The quantized version for the zephyr-7b-beta model is known as zephyr-7b-beta.Q4_K_M.gguf. By loading this model, it will require additional configuration and modification before it can be integrated to the chatbot system. Below is the configuration for the quantized version of the text generation model.

```
sssc.py    sssc2.py    chatbot_test.css    chatbot_test.js
sssc.py > ...


# This is the chat model configuration
zephyr_model = "zephyr-7b-beta.Q4_K_M.gguf"

config_zephyr = {
    'max_new_tokens' : 128,
    'repetition_penalty' : 1.1,
    'temperature' : 0.4,
    'top_k': 50,
    'top_p' : 0.9,
    'stream' : True,
    'threads' : int(os.cpu_count()-1)
}


sssc = CTransformers(
    model = zephyr_model,
    model_type = "mistral",
    lib="avx2",
    **config_zephyr
    )

```

Figure 5. 4 Configuration for zephyr-7b-beta quantized version

The configuration for the model is explained:

‘max_new_tokens’

– The maximum length of response to be generated

‘repetition_penalty’

– Whether the model would be penalized for generating a repeated word or phrase. A value of 1.1 will slightly penalize the model. This encourages the model to product diverse and varied outputs.

‘temperature’

– Controls the randomness of the output generated, a higher value like 0.9 will produce a more diverse and creative output. A value of 0.4 will ensure the model is following the predefined instructions on how to generate responses.

‘top_k’

– A technique to improve diversity and quality of response. Top_k set to 50 means that the model will consider the top 50 words that have the highest probabilities to be the next word in text generation.

‘top_p’

– A value of 0.9 for top_p means that tokens are considered until the cumulative probability exceeds 90%, ensuring a diverse and varied selection of tokens.

‘stream’

– Specifies whether the output should be return as soon as they are produced or wait for the entire sequence to finish. Setting to true enables real-time generation.

‘threads’

– Determine how many CPU threads to be used for text generation process.

5.3 Technical Implementation of the Models, System, and Front-end Interface

The models are first downloaded from the open-source platform, hugging face.

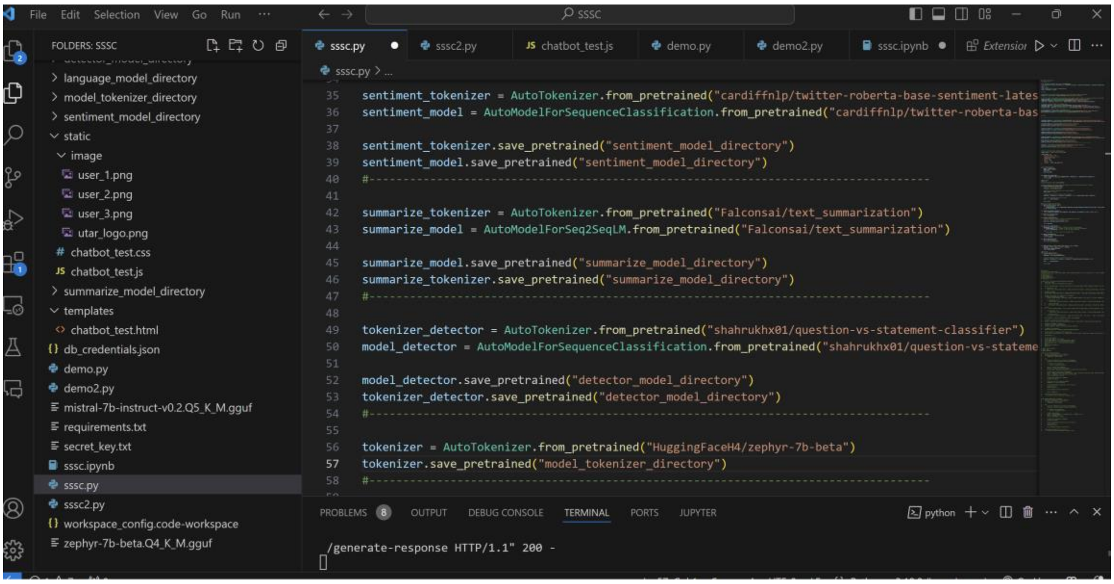


Figure 5. 5 Getting models from hugging face repositories.

The from_pretrained method in AutoTokenizer.from_pretrained and AutoModelForSequenceClassification.from_pretrained are hugging face libraries that gets the model from their repository. AutoTokenizer.save_pretrained and AutoModelForSequenceClassification.save_pretrained methods save the model files into a directory for local and offline usage of the model.

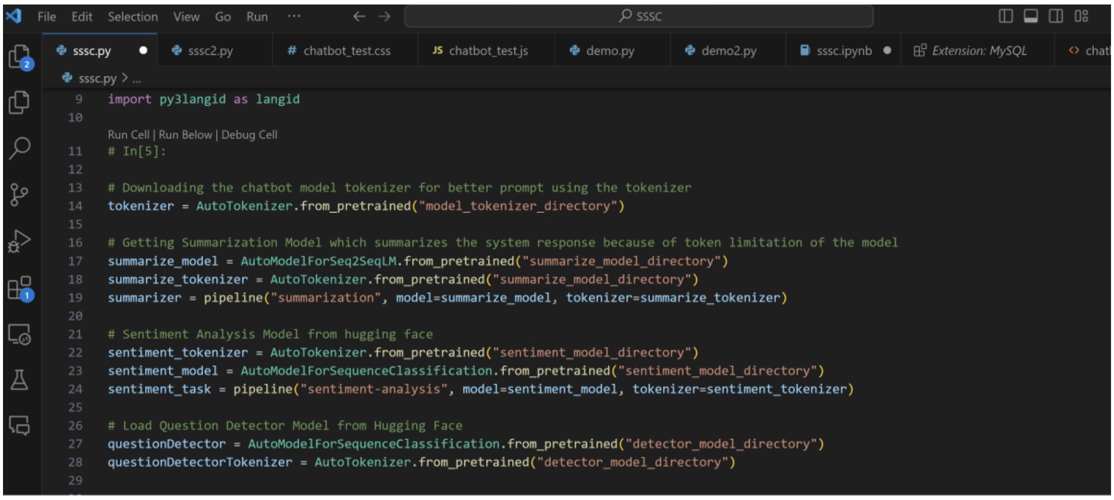


Figure 5. 6 Load the saved models.

The models are loaded into objects that can utilise the generation or classification functions of the model. Pipeline is a wrapper object that simplifies the generation process or classification process. Normally when a model receives text, it will need to tokenize the texts according to every model's own tokenizer function (.tokenize) so that the model understands the text. A pipeline skips this step for the user and implement tokenizer to the texts automatically.

5.3.1 Chatbot Model

The chatbot model is loaded exactly with the code in figure 5.2. Now the model is ready to be called using the variable sssc. The model is called to generate response by this function sssc(text) where text is the input. But in this case, where a conversation is happening, the model would need to remember previous messages. This can be done by using the innate function of model tokenizer (.apply_chat_template()). Below shows the function defined to generate a response based on the chat history. "chat" is the chat history of the conversation that can be in the form of lists, where each dialogue is appended to the lists as new elements. The apply_chat_template() function will convert the chat history lists into a string that is compatible with the format set by the chatbot model. Now, using generate_response(), the model will be able to read the chat history as well to generate an accurate respond.

```
def append_message_to_chat_history(chat_history, role, content):
    message = {"role": role, "content": content}
    chat_history.append(message)
    return chat_history

def prompt_user_input(chat_history):
    prompt = tokenizer.apply_chat_template(chat_history, tokenize=False, add_generation_prompt=True)
    return prompt

def generate_response(prompt):
    return sssc(prompt)
```

Figure 5. 7 Defining functions for text generation.

```
chat_history = []
chat_history = append_message_to_chat_history(chat_history, "system", "I am a helpful chatbot that collects feedback!")
chat_history = append_message_to_chat_history(chat_history, "user", "Hello")

prompt = prompt_user_input(chat_history)
text = generate_response(prompt)
pprint.pprint(text)
```

```
c:\Users\wjlee\VisualStudioCode\Projects\SSSC\.venv\Lib\site-packages\langchain_core\_api\deprecation.py:117: LangChainDeprecationWarning: The functio
  warn_deprecated(
('Hi! How may I assist you today? Please let me know what kind of feedback '
 "you'd like to provide. Is it related to our website, customer service, "
 'product features or something else entirely?\n'
 '\n'
 'Alternatively, if you prefer to share your thoughts anonymously, you can '
 'click on the "Feedback" tab in the main menu and complete a short survey. '
 'Your responses will be kept confidential and used only to improve our '
 'products and services.\n'
 '\n'
 'Thank you for taking the time to provide feedback - it is greatly '
 'appreciated!')
```

Figure 5. 8 Showcase the model text generation capabilities responding to "hello".

5.3.2 Question Detector Model

The question detector model is defined as questionDetector in figure 5.4. The model plays a filtering role in the feedback collection system. Inputs or texts that are entered by the students will go through an intermediate system that handles data processing and filtering. Because the chatbot does not answer any questions, the question detector will help to filter out questions from students. The question_detector() function is applied to inspect user_input and determine if the user_input is a question or a statement. The function itself returns a binary value of 0 and 1, where 0 represents statement and 1 represents question. This way, questions can be filtered out to prevent misinformation generated by the chatbot.

```
def question_detector(user_input):
    with torch.no_grad():
        predicted_label = torch.argmax(questionDetector(**questionDetectorTokenizer(user_input, return_tensors="pt")).logits).item()
    return predicted_label
```

Figure 5. 9 Function for question detector.

```
print(question_detector("What is UTAR?"))
print(question_detector("I like the whether today"))
```

```
1
0
```

Figure 5. 10 Showcase functionality of question detector model.

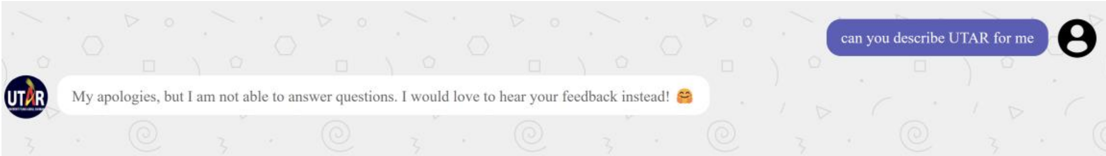


Figure 5. 11 Actual implementation of the question detector model.

The above figure shows the actual implementation of the usage of the detector model. When it receives a question, the system will change the response to be system generated error message and display to user. This is to prevent inaccurate information being generated by the model.

5.3.3 Sentiment Analysis Model

The sentiment analysis model is used to determine the sentiment of the text to determine whether the feedback given is positive, neutral, or negative. The function below predict_sentiment(text) takes in text as input to return the polarity which is the sentiment of the text. This function is cleaner because it uses a pipeline wrapper as opposed to the question detector model.

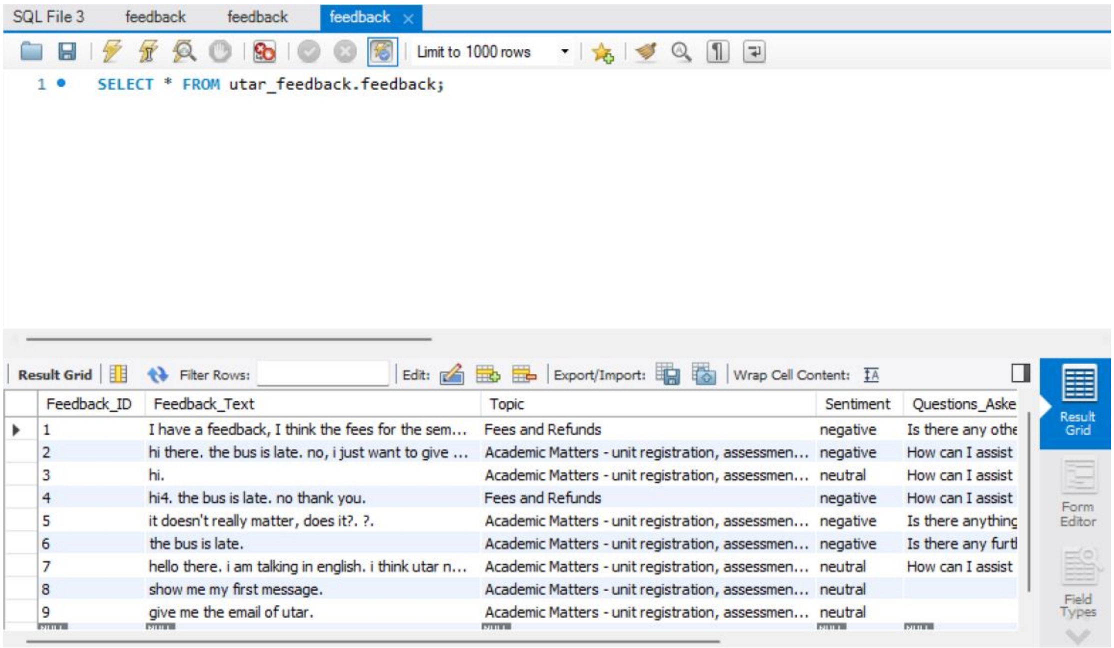
```
def predict_sentiment(text):
    # Use the sentiment model to predict the sentiment of the text
    # Get the sentiment with the highest score
    polarity = sentiment_task(text)[0]['label']

    return polarity
```

Figure 5. 12 Function for sentiment analysis.

```
print(predict_sentiment("I am happy"))
print(predict_sentiment("I am angry"))
print(predict_sentiment("its ok"))
[59]  ✓ 0.1s
... positive
    negative
    neutral
```

Figure 5. 13 Showcase functionality of sentiment analysis model.



	Feedback_ID	Feedback_Text	Topic	Sentiment	Questions_Aske
▶	1	I have a feedback, I think the fees for the sem...	Fees and Refunds	negative	Is there any othe
	2	hi there. the bus is late. no, i just want to give ...	Academic Matters - unit registration, assessmen...	negative	How can I assist
	3	hi.	Academic Matters - unit registration, assessmen...	neutral	How can I assist
	4	hi4. the bus is late. no thank you.	Fees and Refunds	negative	How can I assist
	5	it doesn't really matter, does it?. ?.	Academic Matters - unit registration, assessmen...	negative	Is there anything
	6	the bus is late.	Academic Matters - unit registration, assessmen...	negative	Is there any furt
	7	hello there. i am talking in english. i think utar n...	Academic Matters - unit registration, assessmen...	neutral	How can I assist
	8	show me my first message.	Academic Matters - unit registration, assessmen...	neutral	
	9	give me the email of utar.	Academic Matters - unit registration, assessmen...	neutral	

Figure 5. 14 Actual implementation of sentiment analysis model to store sentiment into database.

5.3.4 System Processes

The models reside separately with the system that controls the data flow and enforce data processing for the texts received. The back-end system is responsible for loading the front-end interface, getting the input from users at the front-end, process/filter the input, determine if the input should be passed to the chatbot model or display error message. The system also keeps a chat history of the conversation for the running session so it remembers the conversation for that session only.

```
@app.route('/generate-response', methods=['POST'])
def generate_response():

    print("before session variables set")
    if 'chat_counter'not in session:
        session['chat_counter'] = 0
    if 'feedback_list' not in session:
        session['feedback_list'] = ""
    if 'questions_list' not in session:
        session['questions_list'] = ""
    if 'chat_history' not in session:
        session['chat_history'] = []
    if 'id' not in session:
        session['id'] = ""

    data = request.get_json()
    topic = data['topic']
    print(topic)
    # Get the user input from the request
    user_input = data['user_input']
    print(user_input)
    user_input = user_input + ". "

    # This is the system prompt to tell the chatbot how it should respond
    if session['chat_counter'] == 0:
        # chat_history.append({"role": "assistant", "content": "Welcome to Universiti Tunku Abdul Rahman (UTAR) Feedback Chatbot! Your sole
        session['chat_history'].append({"role": "assistant", "content": "Welcome to Universiti Tunku Abdul Rahman (UTAR) Feedback Chatbot!
```

Figure 5. 15 Declaring chat history as variables to store the conversation.

Filtering

Below is the process of filtering the output generated by the model.

```python
# Check profanity in the user input
if sssc.detect_profanity(user_input) == False:

    # Check if the user input is a question
    if sssc.question_detector(user_input) == 0:
        response_filtering_1= sssc.generate_response(prompt)
        response_filtering_2 = sssc.remove_sentences_with_email(response_filtering_1)
        response_filtering_3 = response_filtering_2.split('<|user|>')

        if (len(response_filtering_3) > 1):
            response_filtering_4 = response_filtering_3[0]
        else:
            response_filtering_4 = response_filtering_2

        response = sssc.get_first_paragraph(response_filtering_4)

        sssc.append_message_to_chat_history(session['chat_history'], "assistant", response)
        session['chat_history'] = apply_sliding_window(session['chat_history'], 3)

        print("chat length: ", len(session['chat_history']))
        pprint.pprint(session['chat_history'])

        session['feedback_list'] += user_input
```

Figure 5. 16 Filtering process of the output.

Database

One of the major functions of the system is to store the received feedback into database. For the storing process, questions asked by the chatbot will also be extracted to include in database storage, this is to let the feedback reviewer have more context on the feedback given by the user.

```python
def insert_feedback(feedback, questions):
    data = request.get_json()
    # Load the credentials from the JSON file
    with open('db_credentials.json') as f:
        credentials = json.load(f)
    try:
        connection = mysql.connector.connect(**credentials)
        if connection.is_connected():
            cursor = connection.cursor()

        selected_topic = data['topic']
        polarity = sssc.predict_sentiment(feedback)
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        query = "INSERT INTO feedback (Feedback_Text, Topic, Sentiment, Questions_Asked, DateTime) VALUES (%s, %s, %s, %s, %s)"
        values = (feedback, selected_topic, polarity, questions, current_time)
        cursor.execute(query, values)
        connection.commit()
        feedback_id = cursor.lastrowid
        cursor.close()
        connection.close()

        print("Feedback stored successfully!")

        return feedback_id
```

Figure 5. 17 Function for storing feedback received into the database along with sentiment and other details.

Profanity filter

The system also uses a profanity filter to detect profanity in student's feedback, if detected it will ask the students to rephrase the feedback without using profanity. The profanity filter is implanted through the python library profanity-filter.

```python
def detect_profanity(text):
    pf = ProfanityFilter()
    return pf.is_profane(text)
```

Figure 5. 18 Function for profanity filter using profanity-filter library.

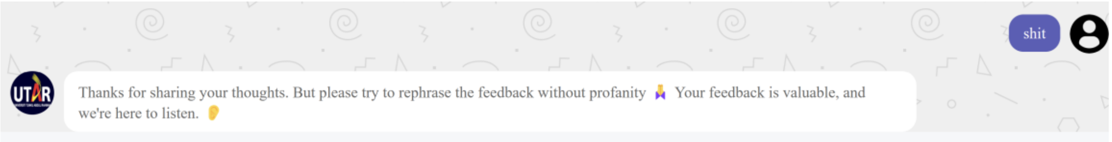


Figure 5. 19 Actual Implementation of the profanity filter.

Normal Process

This is the normal process flow that should happen most of the time.

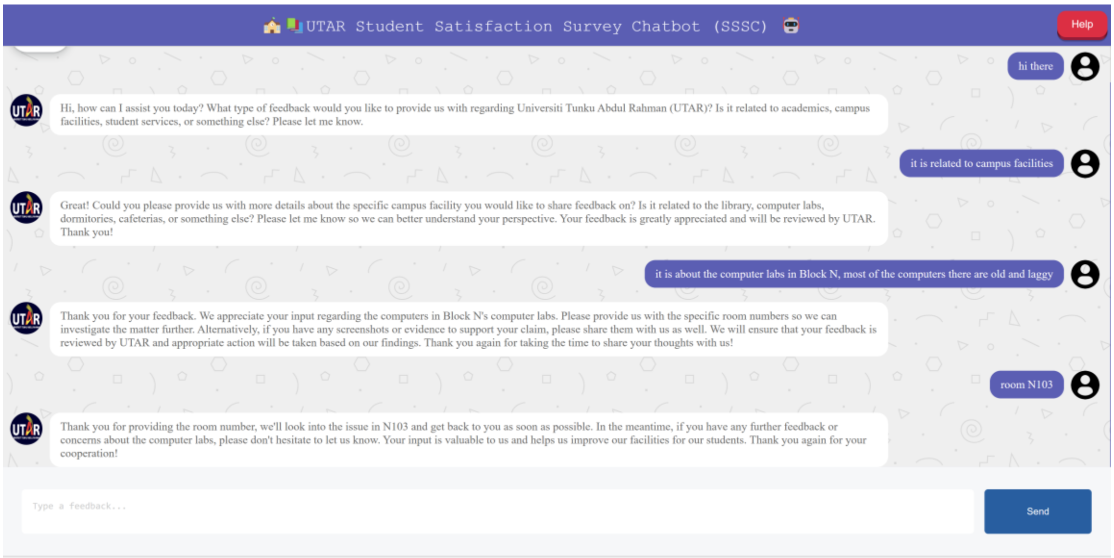


Figure 5. 20 The normal process flow for a feedback session between user and chatbot.

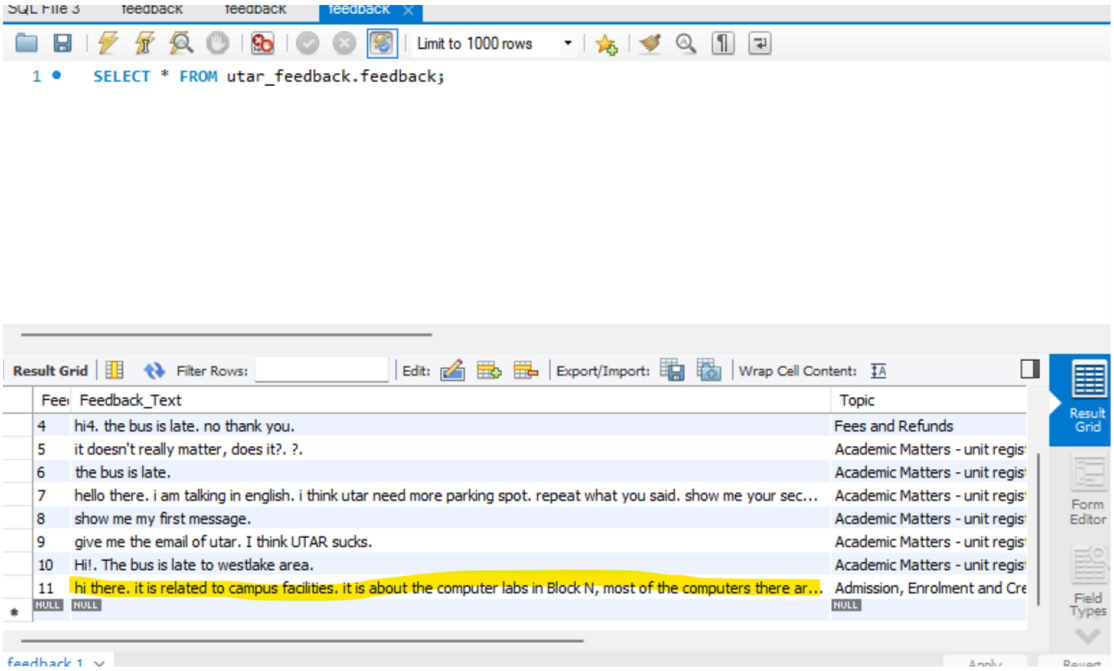


Figure 5. 21 The feedback from the session is stored into the database.

5.3.5 Front-end Interface

A chatbot interface is developed to allow users to interact with the chatbot by typing feedback and submit it for the chatbot to respond to. When first loaded into the browser, 2 windows will popup. The first is a feedback topic selection form where users are required to choose a feedback topic from the selection. The topic selection is taken from the official UTAR portal student feedback form section. If the user wants to close the form without choosing a topic, a system message will popup telling the user to choose a topic. Once the user closes the topic selection form, another window will popup which is the guideline on how to use the chatbot. Inside the guideline, there is a list of the limitations of the chatbot and how to structure the feedback message correctly for better response. There is also a disclaimer that states the response generated by the chatbot does not reflect the official view of UTAR.

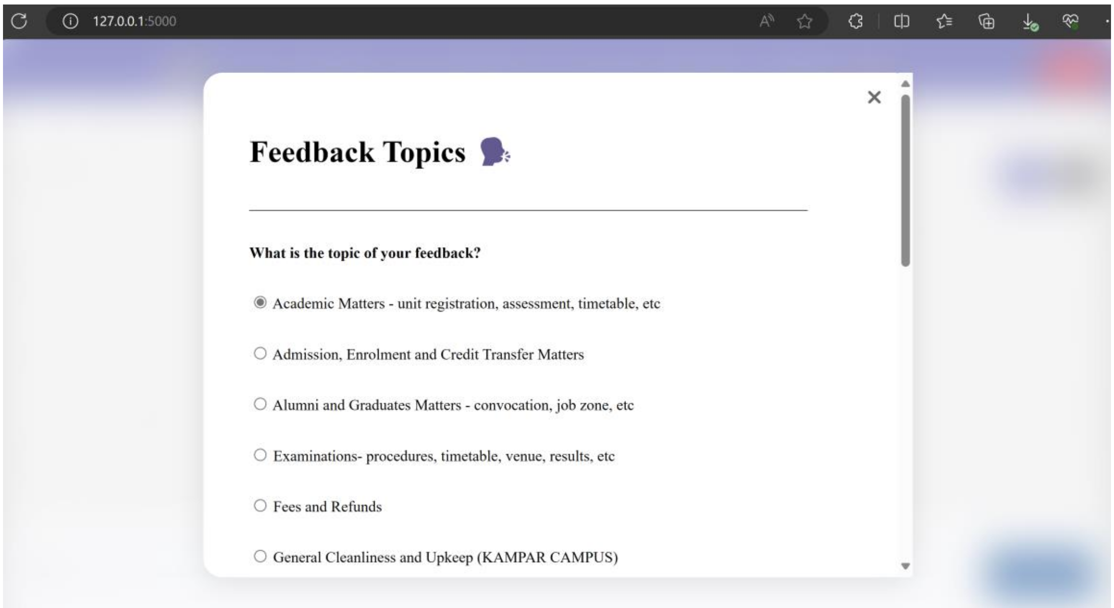


Figure 5. 22 Shows the feedback topic selection form.

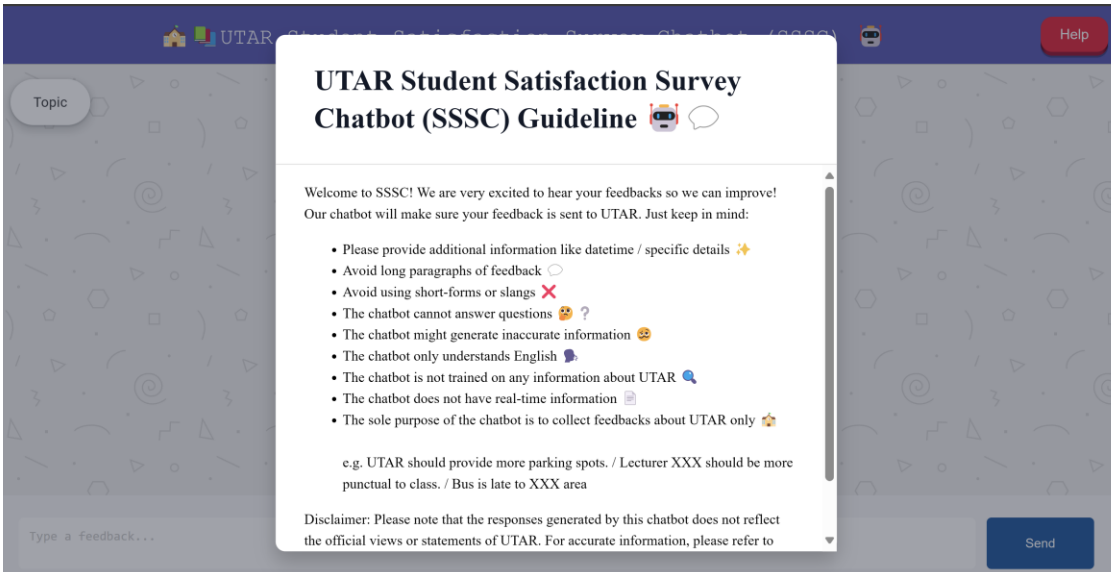


Figure 5. 23 Shows the SSSC usage guideline.

Figure 5. 24 Shows the buttons for opening the topic selection form and the SSSC guideline.

5.4 Implementation Issues and Challenges

Hardware Limitations

There are some challenges throughout the implementation of the systems. One of it is the hardware limitations. Memory is one of the most important resources when loading a model. Initially when finding for a suitable text generation model, loading it would take up all the memory of the hardware and cause crashes to happen. This happened once when loading a Google developed model called gemma-2b.

```
In 1
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-generation", model="google/gemma-2b")
Executed at 2024.03.09 00:04:26 in 29s 447ms

> Traceback…
RuntimeError: [enforce fail at alloc_cpu.cpp:114] data. DefaultCPUAllocator: not enough memory:
 you tried to allocate 2097152000 bytes.
```

Figure 5. 25 Not enough memory when loading model gemma-2b.

Inaccurate Information

Another challenge about the chatbot model is the chatbot would sometimes hallucinates, generating non-existent information. For example, when chatting with the chatbot about feedback that requires assistance, it will reference a non-existent email of UTAR for the student to contact. This action is mitigated by implementing a filtering function for removing sentences with the “@” tag which normally represents email.

```python
from nltk.tokenize import sent_tokenize

# Remove sentences with email addresses because it is most likely generated by the model
def remove_sentences_with_email(text):
    # Split the string into sentences
    sentences = sent_tokenize(text)

    # Define the regular expression for an email address
    email_regex = r'\S+@\S+'

    # Remove sentences that contain an email address
    sentences = [sentence for sentence in sentences if not re.search(email_regex, sentence)]

    # Join the sentences back into a string
    text = ' '.join(sentences)

    return text
```

Figure 5. 26 Function for removing sentences with email in it.

Figure 5. 27 Actual implementation to prevent the chatbot from giving false emails.

<u>Response Time</u>

Another issue is the response time of the chatbot. The response time is very long because of the size of SSSC of 4GB combined with poor hardware specifications of the laptop used. Below shows a response time by SSSC to be almost 4minutes which is unbearable for a normal conversation.

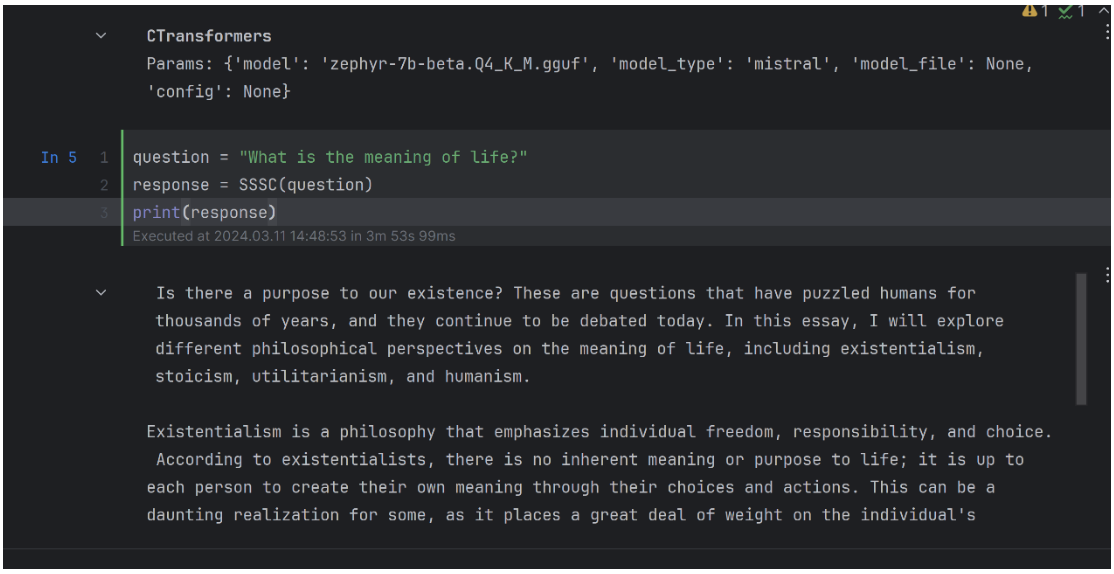


Figure 5. 28 Shows the response generation time to be 3m 53s

The way to improve this is by restricting the chatbot's respond to be more focused like giving it a predefined instructions to follow e.g. follow the setting of a helpful feedback chatbot. Modifying the configuration of the model is also helpful like changing the temperature, top_k, top_p parameters of the model in Figure 5.4. Using a better hardware like using GPUs instead of CPU will also improve the response time significantly.

5.6 Concluding Remark of the System Implementations

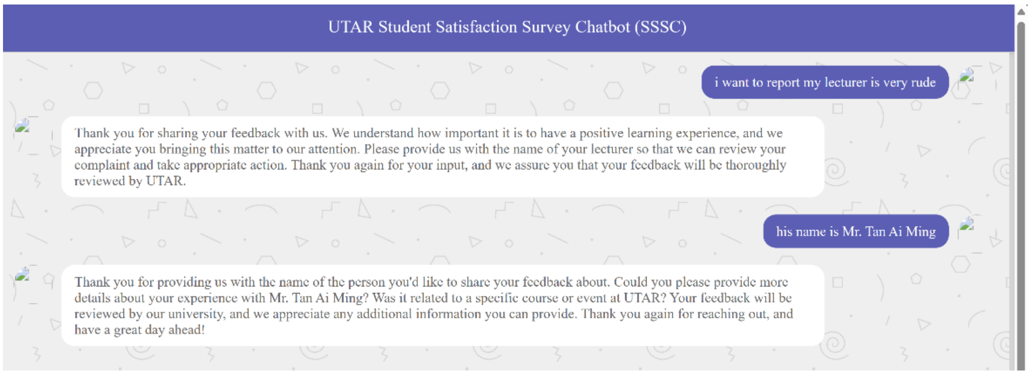
The entire system comprises of the front-end interface, the back-end system and the text generation chatbot model. The front-end interface allows users to interact with the chatbot through conversations. The back-end system receives the input text from conversations and proceed to filter the texts and forward the texts to the next destination. The text generation model SSSC will receive the text from back-end system and generate a response according to the text. The back-end system will get the response generated by the model and display it for the users to continue the conversation.

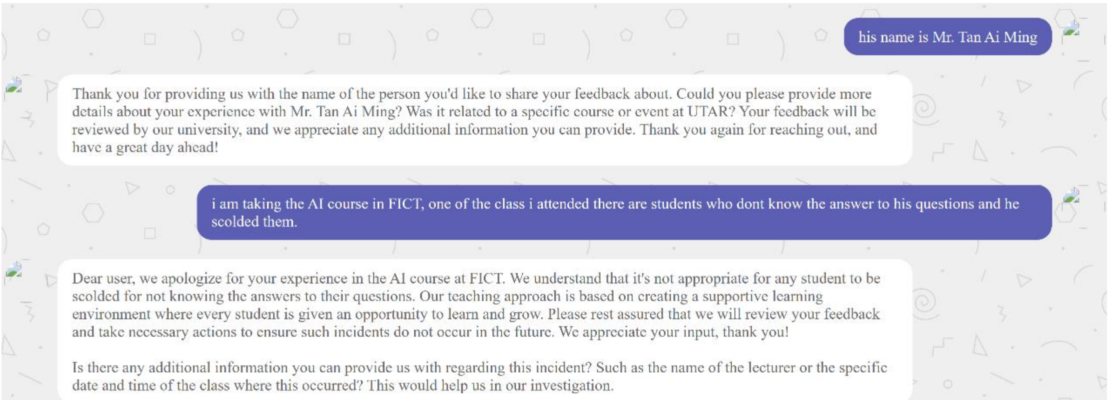
Chapter 6

System Testing and Evaluation

6.1 System Testing

Test Case 1: Testing normal feedback session. (Success)





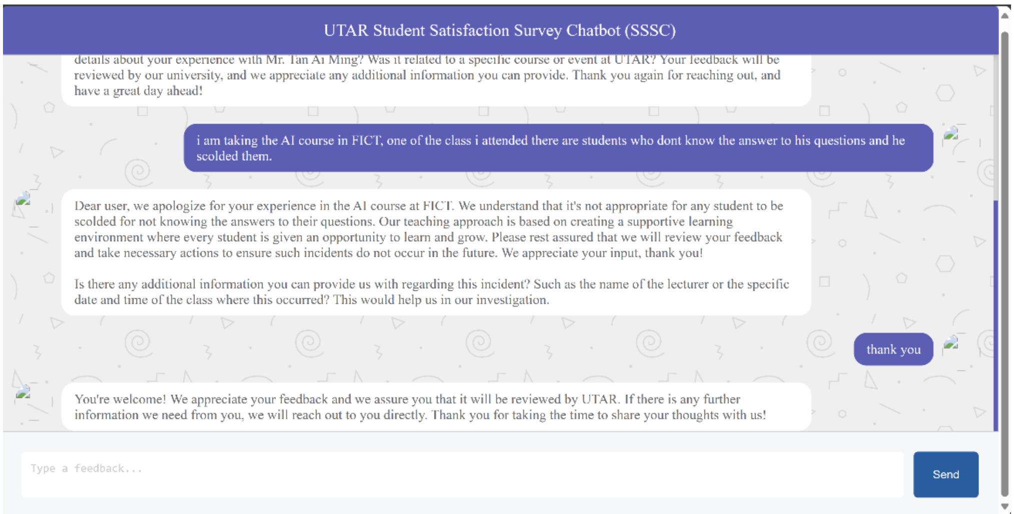


Figure 6. 1, Figure 6. 2, Figure 6. 3 Shows a normal feedback session between user and chatbot.

Test Case 2: Will the chatbot generate information at request and would it be accurate?

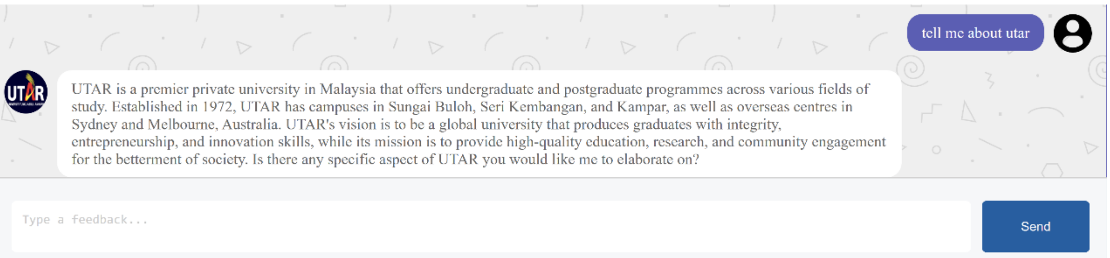


Figure 6. 4 Shows SSSC generating inaccurate information.

Solution: Modify the system prompt that instructs the chatbot model to avoid giving information if unsure of the validity and reject answering requests if necessary.

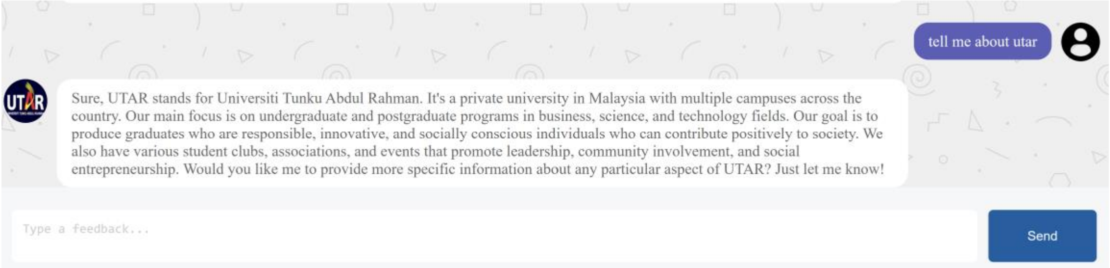


Figure 6. 5 Shows SSSC generating general information without giving misinformation.

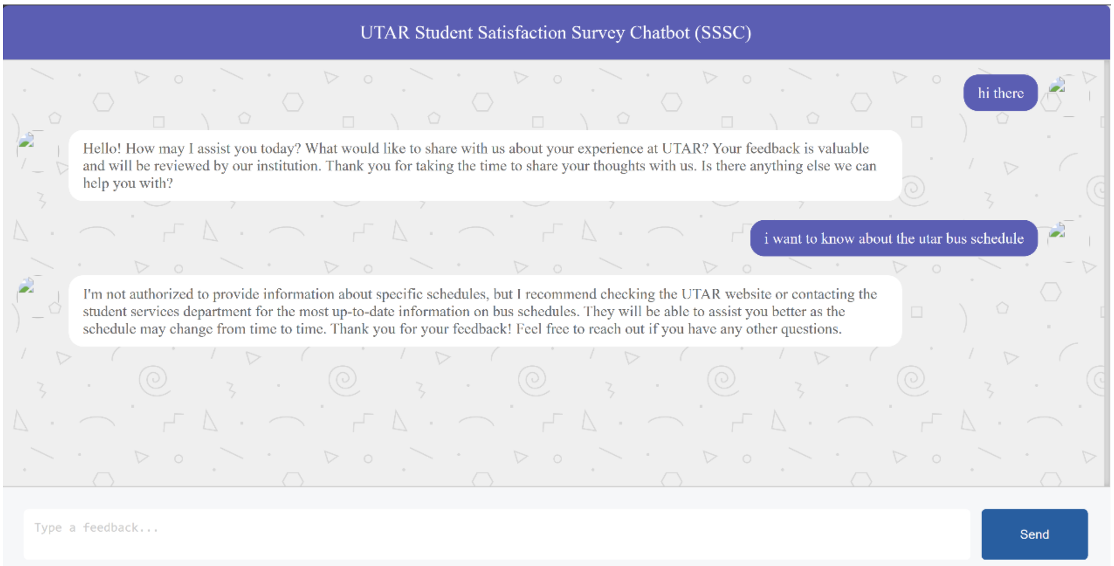


Figure 6. 6 Shows SSSC unable to provide bus schedule information.

Test Case 3: Testing Language Detector. (Success)

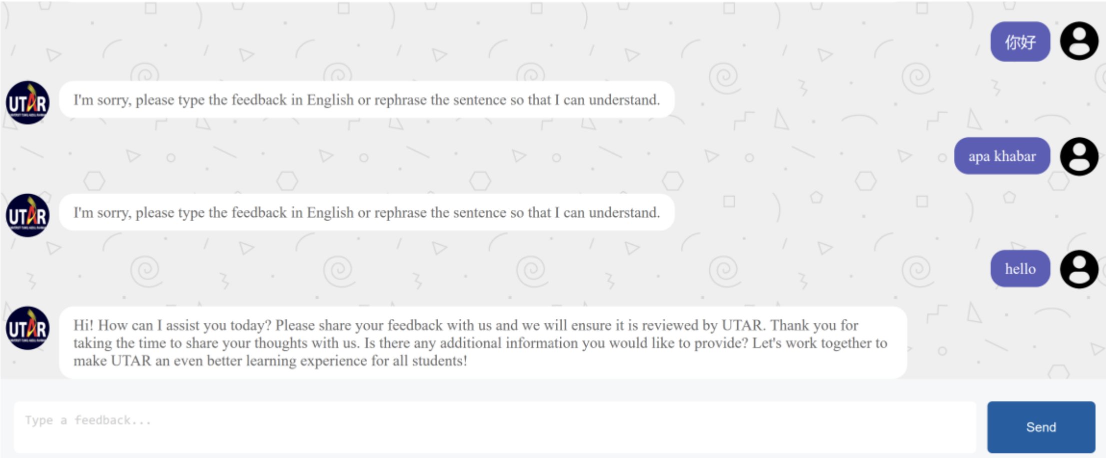


Figure 6. 7 Shows SSSC successfully detected foreign language.

Test Case 4: Testing Question Detector. (Success)

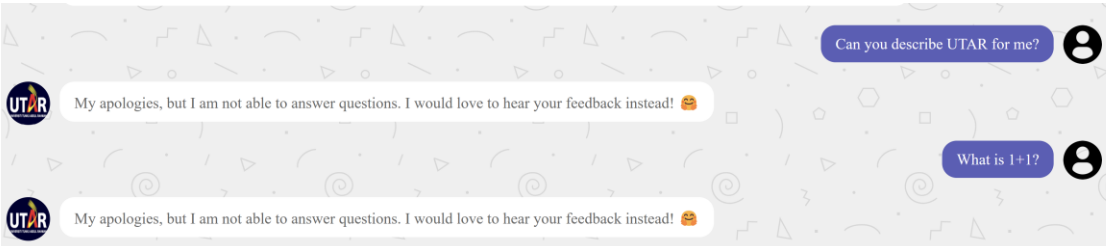


Figure 6. 8 Shows SSSC successfully filtered out questions.

Test Case 5: Testing Profanity-Filter. (Success)

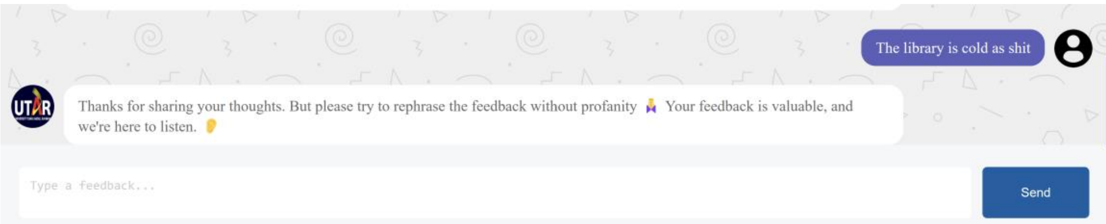


Figure 6. 9 Shows SSSC successfully detected profanity.

Test Case 6: Testing irrelevant response. (Success)

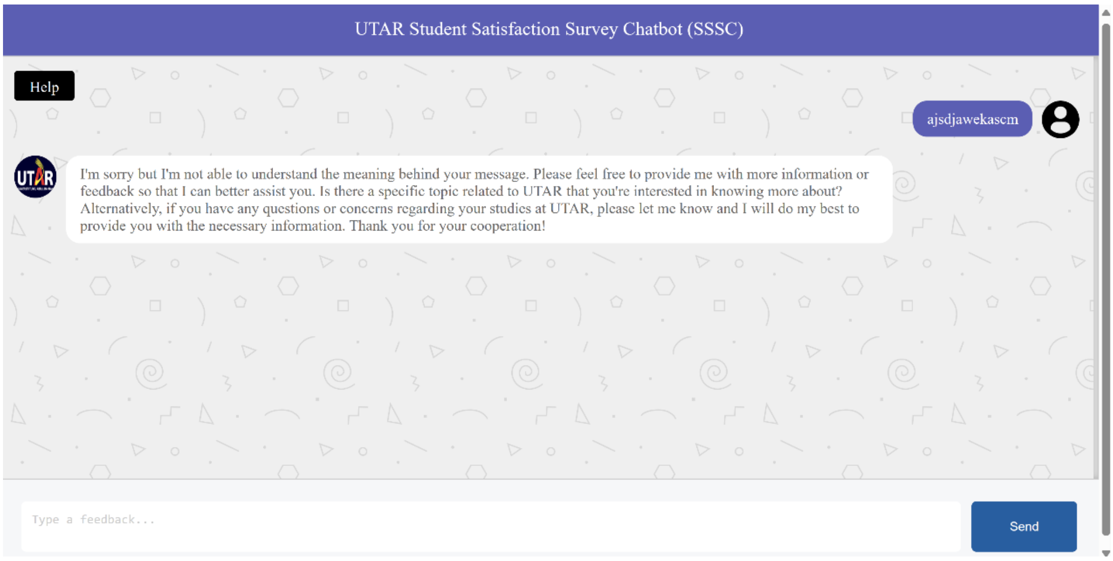


Figure 6. 10 Shows SSSC successfully asked for relevant response.

Test Case 7: Offer assistance to users. (Success)

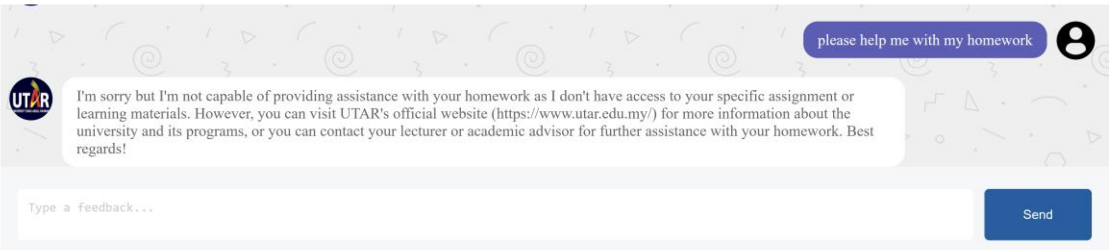


Figure 6. 11 Shows SSSC unable to provide assistance to users on homework.

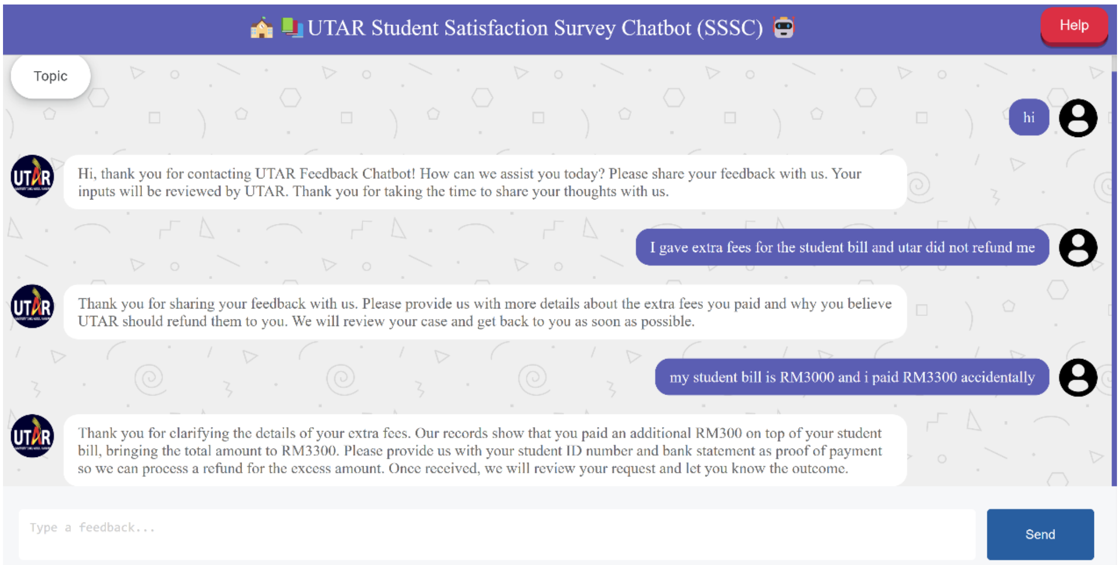


Figure 6. 12 Shows SSSC helping user on student bill refund feedback.

The most important aspect of generating relevant response according to the project's criteria is the system prompt that is used for the text generation model. The prompt helps to define instructions for the model to follow strictly like how to answer certain response and defining the model's personality when responding. Below is the finalized system prompt that is used to guide the chatbot model on answering which generated satisfactory results based on the test cases.

Prompt: "Welcome to Universiti Tunku Abdul Rahman (UTAR) Feedback Chatbot! Your sole purpose is to collect and acknowledge feedback from students without taking any direct actions. Respond empathetically in short sentences and assure users that their feedback will be reviewed by UTAR. Avoid initiating any actions. Request for further information about the feedback received. Thank the user for their input and encourage them to share any additional feedback they may have."

6.2 Project Challenges

Zephyr-7b-beta is not trained within the RLHF phase so the model might produce problematic output when asked to do so [30]. But a partial solution is implemented with the help of a question detector to filter out requests like "Can you teach me how to make a bomb?" When the user asks the chatbot to provide assistance to negative behaviors, the question detector will filter out the questions and display an error message that the chatbot does not answer questions. However, it does not solve the problem completely, as the user can mask their intention as a statement/request rather than a question. For example, "teach me how to make a bomb" is interpreted as a statement by the question detector and therefore the request will not be filtered out. To solve the problem in a more robust way, the model should be trained with human safety preferences within the RLHF phase to teach the model that negative behavior should be discouraged.

Using an AI Large Language Model for text generation is almost impossible to control the exact output of the model. Because the model is trained on large text corpora and have its own understanding of words, it can generate different types of response for the same question. If one of the response is not ideal for the circumstances of the project, then the best way to mitigate it is by defining the system prompt. If the solution is not working, the model can only be retrained to make it adhere to specific restrictions which is very resource intensive.

6.3 Objectives Evaluation

Objective Evaluation:

1. Chatbot Implementation: The chatbot implementation of SSSC successfully achieved the objective of using Natural Language Processing (NLP) to ask follow-up questions and collect quality feedback on UTAR services. The chatbot interface helped to facilitate conversations and collect feedback from students.

2. Sentiment Analysis Integration: The sentiment analysis module effectively analysed the sentiment of the feedback collected by the chatbot to store it in the feedback database.

3. Filtering Module Integration: The filtering module successfully detected inappropriate words in student's feedback response, therefore able to maintain a respectful conversation environment.

4. Chat Interface Development: The development of a chat interface for students to provide feedback and receive responses from the chatbot was completed. The interface also collects feedback topic from students to store it in the database. For clarity on the usage of the chatbot, a guideline is included into the chat interface through a "help" button.

5. Feedback Storage in Database: The system successfully stored the feedback collected from students into a database. The database ensures the feedback collected are organized and maintained for the feedback reviewer to check.

Overall, the project objectives were all met, the result is a functional feedback collection chatbot called SSSC powered by NLP chatbot technology. The chatbot integrated with a back-end system and front-end interface is capable of asking follow-up questions, analysing sentiment, filtering inappropriate language, providing a user-friendly interface, and storing feedback data. This ensures the effectiveness of the feedback collection process during conversations of UTAR students with the chatbot.

Chapter 7

7.1 Conclusion

There are 3 problems faced which is Low Quality Response Collected from Web Survey, Lack of Language Support, and Filtering negative behavior and irrelevant responses. When using the traditional chatbot to collect feedback from students, it will return low quality response due to survey fatigue. Lacking language support will also cause the feedback collected to be less optimal because the question cannot be translated for non-English speaking individuals. Finally, the traditional chatbot cannot filter out negative behavior and irrelevant responses which is a major factor in contributing to low quality feedback collected from university students.

The motivation behind this project is to help universities to improve based on the feedback from the students. University management will not be on the field most of the time with the students, so they are not able to grasp the situations faced by students and therefore unable to provide solutions for the students. But through an interactive chatbot that can carry out human-like conversations with the students, students can use the chatbot to relay their problems to the management just like complaining a problem to a real human being. This way of providing feedback would provide a higher quality and useful feedback for the university.

The proposed solution for this problem is to build an interactive chatbot called SSSC to leverage on machine learning to replace the traditional web surveys to collect feedback from university students. The chatbot can have a natural conversation with students and asks follow-up questions to prompt more quality feedback from the students. The feedback will then be stored in a database managed by the university so that they can analyze the feedback collected by the chatbot.

Finally, SSSC is successfully developed with the purpose of extracting higher quality feedback from UTAR students through natural human-like conversations and asking follow-up questions or requests additional information from the student.

7.2 Recommendations

One recommendation for this project is to allow students to upload images for the chatbot. The chatbot would then store the image into the database along with the feedback from students. This improvement would allow better context for the feedback and can serve as evidence. This project was not able to apply the improvement because of time constraints problems.

Another recommendation is to use better hardware to serve as the server that hosts the chatbot model, this way the response time would be improved. Response time is one of the most important aspects of a technological product, having to wait for minutes for a reply is unacceptable. Ways for improvement in this aspect can be found in the chatbot model itself by quantizing it to be less than 8-bit integer although it will affect the quality of response the more it is quantized. Future study on this project should focus on improving the response time to be as fast as possible.

REFERENCES

[1] F. Clarizia, F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, “Chatbot: An education support system for student,” *Cyberspace Safety and Security*, pp. 291–302, 2018.

[2] E. Adamopoulou and L. Moussiades, “An overview of chatbot technology,” *IFIP Advances in Information and Communication Technology*, pp. 373–383, 2020.

[3] V. V., J. B. Cooper, and R. L. J., “Algorithm inspection for Chatbot Performance Evaluation,” *Procedia Computer Science*, vol. 171, pp. 2267–2274, 2020.

[4] V. Hristidis, “Chatbot Technologies and challenges,” *2018 First International Conference on Artificial Intelligence for Industries (AI4I)*, 2018.

[5] N. Abbas, T. Pickard, E. Atwell, and A. Walker, “University student surveys using Chatbots: Artificial Intelligence Conversational Agents,” *Learning and Collaboration Technologies: Games and Virtual Environments for Learning*, pp. 155–169, 2021.

[6] “Challenges in chatbot development,” Ideta, https://www.ideta.io/blog-posts-english/challenges-in-chatbot-development-chatbot-strategy (accessed Sep. 8, 2023).

[7] Z. Lin *et al.*, “Caire: An end-to-end empathetic chatbot,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 09, pp. 13622–13623, 2020. doi:10.1609/aaai.v34i09.7098

[8] J. Feine, S. Morana, and A. Maedche, “A chatbot response generation system,” *Proceedings of the Conference on Mensch und Computer*, 2020. doi:10.1145/3404983.3405508

[9] “Upgrade to pro plan,” TARS, https://chatbot.hellotars.com/conv/S7aT_q/ (accessed Sep. 8, 2023).

[10] “University feedback form: Chatbot Template,” Landbot, https://landbot.io/chatbot-templates/university-feedback-form (accessed Sep. 8, 2023).

[11] “Rule-based Chatbots vs. AI Chatbots: Key differences,” RSS, https://www.hubtype.com/blog/rule-based-chatbots-vs-ai-chatbots (accessed Sep. 8, 2023).

[12] J. Schulman et al., “Introducing chatgpt,” Introducing ChatGPT, https://openai.com/blog/chatgpt (accessed Apr. 24, 2024).

[13] Meta-Llama, “Model Details,” GitHub, https://github.com/meta-llama/llama/blob/main/MODEL_CARD.md (accessed Apr. 24, 2024).

[14] T. Jesuthasan, “Autoregressive (AR) language modelling,” Medium, https://tonyjesuthasan.medium.com/autoregressive-ar-language-modelling-c9fe5c20aa6e (accessed Apr. 24, 2024).

[15] Z. Xiao *et al.*, “Tell me about yourself,” *ACM Transactions on Computer-Human Interaction*, vol. 27, no. 3, pp. 1–37, Jun. 2020. doi:10.1145/3381804

[16] J. Rhim, M. Kwak, Y. Gong, and G. Gweon, “Application of humanization to survey chatbots: Change in chatbot perception, interaction experience, and survey data quality,” *Computers in Human Behavior*, vol. 126, p. 107034, Jan. 2022. doi:10.1016/j.chb.2021.107034

[17] S. Kim, J. Lee, and G. Gweon, “Comparing data from chatbot and web surveys,” *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, May 2019. doi:10.1145/3290605.3300316

[18] “10.6. the encoder–decoder architecture¶ colab [pytorch] open the notebook in colab colab [mxnet] open the notebook in colab colab [jax] open the notebook

in colab colab [tensorflow] open the notebook in colab sagemaker studio lab open the notebook in SageMaker Studio Lab," 10.6. The Encoder–Decoder Architecture - Dive into Deep Learning 1.0.3 documentation, https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html (accessed Apr. 25, 2024).

[19] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," arXiv.org, https://arxiv.org/abs/1409.1259 (accessed Apr. 25, 2024).

[20] R. Inflianskas, "Profanity-Filter," profanity-filter: A Python library for detecting and filtering profanity, https://pypi.org/project/profanity-filter/ (accessed Apr. 25, 2024).

[21] "Query classifier for semantic search and QA," Query Classifier for Semantic Search and QA, https://www.deepset.ai/blog/save-resources-with-query-classifier-for-neural-search (accessed Apr. 25, 2024).

[22] "Py3langid," PyPI, https://pypi.org/project/py3langid/ (accessed Apr. 25, 2024).

[23] NLTK, https://www.nltk.org/ (accessed Sep. 8, 2023).

[24] D. Bergmann, "What is fine-tuning?," IBM, https://www.ibm.com/topics/fine-tuning (accessed Apr. 26, 2024).

[25] A. Vaswani et al., "Attention is all you need," arXiv.org, https://arxiv.org/abs/1706.03762 (accessed Apr. 26, 2024).

[26] D. Ellis, "What is Anaconda for python & why should you learn it?," HubSpot Blog, https://blog.hubspot.com/website/anaconda-python (accessed Sep. 8, 2023).

[27] "Hugging face hub documentation," Hugging Face Hub documentation, https://huggingface.co/docs/hub/index (accessed Sep. 8, 2023).

[28] “Transfer learning and fine-tuning: Tensorflow Core,” TensorFlow, https://www.tensorflow.org/tutorials/images/transfer_learning (accessed Sep. 8, 2023).

[29] G. Broadhead, “A brief guide to LLM numbers: Parameter count vs. training size,” Medium, https://medium.com/@greg.broadhead/a-brief-guide-to-llm-numbers-parameter-count-vs-training-size-894a81c9258 (accessed Apr. 26, 2024).

[30] “Huggingfaceh4/zephyr-7b-beta · hugging face,” HuggingFaceH4/zephyr-7b-beta · Hugging Face, https://huggingface.co/HuggingFaceH4/zephyr-7b-beta (accessed Apr. 26, 2024).

[31] “Shahrukhx01/question-vs-statement-classifier · hugging face,” shahrukhx01/question-vs-statement-classifier · Hugging Face, https://huggingface.co/shahrukhx01/question-vs-statement-classifier (accessed Apr. 26, 2024).

[32] “Cardiffnlp/twitter-roberta-base-sentiment-latest · hugging face,” cardiffnlp/twitter-roberta-base-sentiment-latest · Hugging Face, https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest (accessed Apr. 26, 2024).

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 2**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Done with finding suitable models as the chatbot model.
Done with setting up the database.

2. WORK TO BE DONE

Test the chatbot model and modify it according to project criteria.
Chat interface.

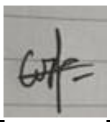
3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 4**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Preliminary testing the chatbot model done, decided to choose and modify model zephyr-7b-beta
Setup chat interface in visual studio code to test the model.

2. WORK TO BE DONE

Chat interface.
Filter response.

3. PROBLEMS ENCOUNTERED

-The chatbot cannot remember previous conversations

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 6**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Able to filter response from students.
Modified the system prompt to be more restrictive to not attend to requests from students.
Done partial front-end chatbot interface, able to chat using the interface.
Defined a chat history list to store conversations per session for the chatbot to remember.

2. WORK TO BE DONE

Final Front-end Chatbot Interface
Add guideline on how to use the chatbot.

3. PROBLEMS ENCOUNTERED

-Chatbot occasionally generates faulty response. E.g. completing the sentence instead of responding to it.

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's signature

Student's signature

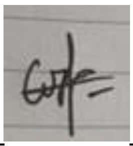
FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: **3,3**	**Study week no.: 8**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE
[Please write the details of the work done in the last fortnight.] Able to filter response from model too, when faulty response is generated, it is most likely able to be filtered. Able to detect language, filter profanity and filter out question from the input for better response. Chat interface guideline added to teach users how to use the model.
2. WORK TO BE DONE **Adding topic selection form for students to choose from.** **Finalizing the system**
3. PROBLEMS ENCOUNTERED -
4. SELF EVALUATION OF THE PROGRESS -

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 10**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Added the feedback topic selection form.
Deployed a flask application to host the chatbot and system.
Added animations to js script for the front-end.

2. WORK TO BE DONE

-Final Report

3. PROBLEMS ENCOUNTERED

-Chatbot response is very slow (2 min per conversation)

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 12**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Finalizing system is done.
Final Modification to the system prompt done.
Chatbot Interface done.
Database Storing functionality done.

2. WORK TO BE DONE

-Final Report

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	**Study week no.: 14**
Student Name & ID: Lee Wei Jin (20ACB01909)	
Supervisor: Dr. Zurida binti Ishak	
Project Title: Student Satisfaction Survey Chatbot	

1. WORK DONE

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

Final Report done

2. WORK TO BE DONE

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

-

Supervisor's Signature

Student's Signature

Poster


Student Satisfaction Survey Chatbot (SSSC)
WHAT IS SSSC?
SSSC is an AI chatbot designed to collect feedback from university students through having natural human-like conversations. SSSC is different from traditional chatbots as it is capable of understanding the student's question and answer them accurately. SSSC generates text based on the input given instead of answering based on predefined rules.
WHY NEED IT?
Study has found that the traditional web survey forms used to collect feedback is not effective. The forms caused survey fatigue among respondents and resulted in lower quality feedback. SSSC can facilitate natural conversations with the students like follow up to prompt more details so that it can collect meaningful feedback.
WHAT CAN SSSC DO?
• Understand student's input
• Answer student's questions about the university.
• Store student's feedback in a database
• Have natural conversations with students
• Detect negative words
HOW TO DEVELOP IT?
SSSC will be developed on Visual Studio Code using python and MS SQL database. SSSC will utilize Large Language Models (LLMs) to save time. The LLM will enable SSSC to generate texts based on input given. SSSC will later be fine-tuned with university information so that it can answer student's questions.
Platforms/Tools Used
• Python
• Visual Studio Code
• Anaconda
• HuggingFace
• Github
• Pytorch
• MS SQL
• Jupyter
SSSC will need to be integrated with a UI so that it can be displayed for the students to interact with. It can be a pop-up window or integrated into the website directly or using third-party API.

Plagiarism Result

Document Viewer

Turnitin Originality Report

Processed on: 26-Apr-2024 21:54 +08
ID: 2362620296
Word Count: 11564
Submitted: 1

SSSC.docx By Lee Wei Jin

Similarity Index	Similarity by Source	
9%	Internet Sources:	7%
	Publications:	6%
	Student Papers:	N/A

exclude quoted | exclude bibliography | exclude small matches | mode: quickview (classic) report | print | download

1% match (Internet from 22-Dec-2023)
https://www.coursehero.com/file/p596trc/Chapter-1-Chapter-1-Chapter-1-Chapter-1-Chapter-1-Chapter-1-Chapter-1-Chapter-1/

<1% match (Internet from 12-Mar-2021)
https://www.coursehero.com/file/55236510/Lecture11pptx/

<1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my

<1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my

<1% match (Internet from 12-Dec-2023)
http://eprints.utar.edu.my

<1% match (Internet from 12-Nov-2023)
https://export.arxiv.org/pdf/2205.15104

<1% match (Internet from 14-Feb-2024)
http://export.arxiv.org

Originality Report Form

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)	Lee Wei Jin
ID Number(s)	20ACB01909
Programme / Course	IA
Title of Final Year Project	Student Satisfaction Survey Chatbot

Similarity	**Supervisor's Comments** **(Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index:___9_____ % **Similarity by source** Internet Sources: _______17________% Publications: ______6______ % Student Papers: _______0_______ %	
Number of individual sources listed of more than 3% similarity: _________	

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

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

__ _______________

Signature of Supervisor

Name: _Dr. Zurida Binti Ishak__

Date:
____28/04/2024_______________________

FYP Checklist

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB01909
Student Name	Lee Wei Jin
Supervisor Name	Dr. Zurida Binti Ishak

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review

	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

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

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

____

(Signature of Student)

Date: 26/04/2024