UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

ASSIGNMENT COVER PAGE

Validated by: UNIVERSITY OF LINCOLN UNITED KINGDOM

Programme	**Course Code and Title**
Bachelor of Computer Science (Hons)	CAI3013/N Introduction to Artificial Intelligence
Student's name / student's id	**Lecturer's name**
CHAN SEOW FEN / 0207368	Dr. Robin Tan

Date issued	**Submission Deadline**	**Indicative Weighting**
Week 5 -9/10/2023	Week 10-13/11/2023	30%
Assignment title	Amusing Snowman	

This assessment assesses the following course learning outcomes

# as in Course Guide	**UOWM KDU Penang University College Learning Outcome**
CLO3	Develop programs in an AI language, expert system shell or data mining tool
# as in Course Guide	**University of Lincoln Learning Outcome**
CLO2	Apply Artificial Intelligence techniques to solve practical problems

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student's signature: Submission Date:

13 / 11 / 23

Assignment 2

ORIGINALITY REPORT

6%	5%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	idoc.pub Internet Source	2%
2	Bing Zhang. "Research on binary vulnerability mining technology based on control flow integrity detection", Journal of Physics: Conference Series, 2020 Publication	1%
3	www.collectionscanada.gc.ca Internet Source	1%
4	123dok.net Internet Source	1%
5	publication.petra.ac.id Internet Source	<1%
6	tet.pub.ro Internet Source	<1%
7	vdocuments.mx Internet Source	<1%
8	www.coursehero.com Internet Source	<1%

Table of Contents

1.0 Introduction

Snowman Chatbot is an interactive artificial intelligence designed to bring festive spirit and functionality to user interactions on the Discord platform. Designed for Discord users looking for a mix of holiday cheer, weather updates and engaging Christmas content, the Snowman Chatbot serves as a virtual companion during the holiday season. This report looks at the intricacies of Snowman's development, exploring its features, data sources and the application of AIML algorithms. With a focus on the user experience, the report also discusses the user interface and provides insights into the choices made to ensure a seamless and enjoyable interaction. In addition, the report examines the impact of applying AIML algorithms versus not applying them, promoting a comprehensive understanding of Snowman's capabilities. Through descriptive illustrations and screenshots, we aim to provide a visual narrative that enhances clarity and understanding of the report content. The report also serves with conclusion to sum up the report, as well as references section for interested party to have deeper look on the referred articles and websites, and an appendix section for additional information.

2.0 Design of Snowman

2.1 Avatar and Discord Profile of Snowman

The avatar of Snowman Chatbot is using a cute and aesthetic picture from Freepik (*Cute kawaii snowman illustration Premium Vector*, 2021). By using such picture as profile picture of the chatbot could highly enhance the user experience as well as user friendliness when compared to default profile picture or other profile picture as this design choice aims to resonate with the Christmas theme, creating an instant connection with users.

Figure 2.1 Avatar of Snowman Chatbot.

Moving forward, the discord profile of Snowman Chatbot is basically consisting of the avatar icon above as profile picture, and having Snowman as username, in addition to having a brief description of the chatbot at “About Me” section as shown in *Figure 2.2*. This customisation of discord profile has highly enhanced the user experience as well as user friendliness by providing the clear image of this chatbot as what character it is and what is his name as well as what key functionality it can provide. Moreover, the text as shown in the figure is also following a cheerful, cute, as well as having emoji as representing a cute snowman avatar by using his own tone speaking those words, thus greatly enhancing user friendliness.

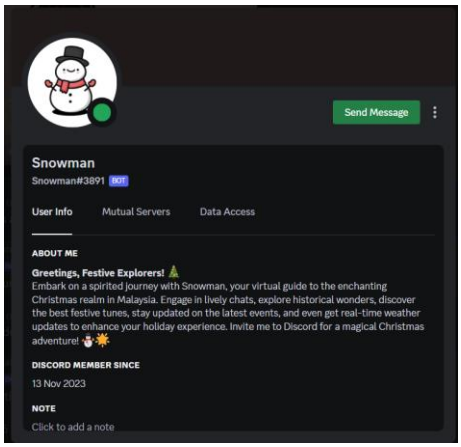
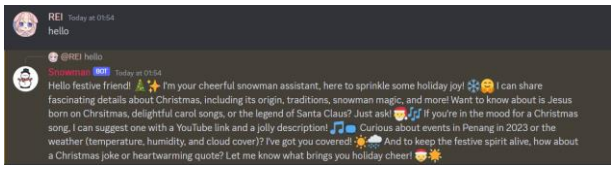
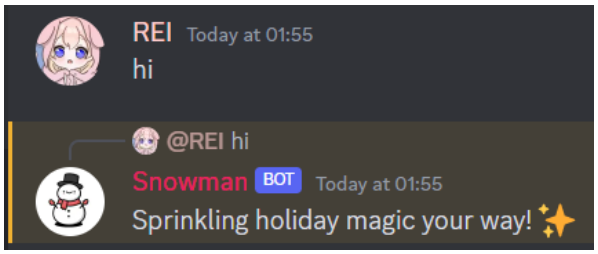


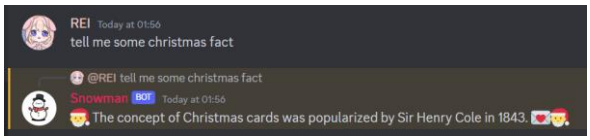
Figure 2.2 Discord Profile of Snowman Chatbot.

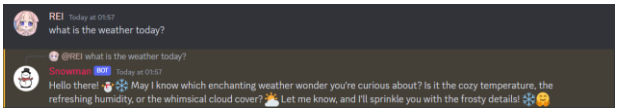
2.2 Speaking style of Snowman

As briefly stated above, the tone of Snowman Chatbot is following a warm and friendly manner, adopting a conversational tone that aligns with the spirit of the season. Its speech is characterized by a blend of holiday joy and helpfulness, enhancing the overall user experience. Moreover, it also having emoji in every line, which makes it looks cute and thus highly enhancing user friendliness of the chatbot, making user felt more likely to speak and interact with the chatbot. Figures below shows the example of Snowman's speaking behaviour.









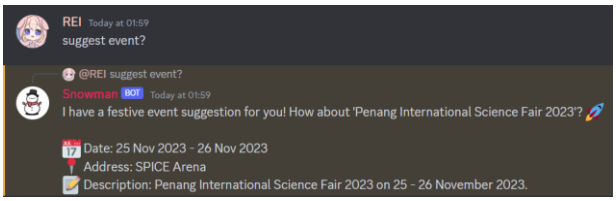


Figure 2.3, *Figure 2.4*, *Figure 2.5*, *Figure 2.6*,
Figure 2.7, *Figure 2.8*,
Figure 2.9, *Figure 2.10* Snapshots of Snowman's Chatting Behaviour.

2.3 Functionality of Snowman

The key functionality of Snowman Chatbot included basic greetings and querying on information of Christmas. Moreover, the key functionality of Snowman also consisted of querying Christmas Song with YouTube video and description of the song suggested, querying real-time weather information such as temperature, humidity and cloud cover. Last but not least, Snowman allowed user to query about the event that will be happening in Penang 2023 with its event name, description and location if information is available. As additional features, Snowman is capable of telling Christmas jokes that will make user feeling cold and giving some inspirational quote to enlighten the day of user. The section onwards will discuss the key functionality with details and screenshots of the Snowman Chatbot.

3.0 Information of Christmas

3.1 History

The history section of Snowman functionality can be divided into 8 sections. That is, the origin of Christmas, the tradition of Christmas, about snowman, is Christmas the day Jesus born?, history of Christmas cancellation, history of Carol song, who invented Santa Claus and some interesting Christmas facts. Snowman will answer the question of user and give relevant information with warm, friendly and come with emoji response to enhance the user experience and user friendliness as shown in *Figure 3.1*, *Figure 3.2*, *Figure 3.3*, *Figure 3.4*, *Figure 3.5*, *Figure 3.6*, *Figure 3.7*, *Figure 3.8*, *Figure 3.9*.

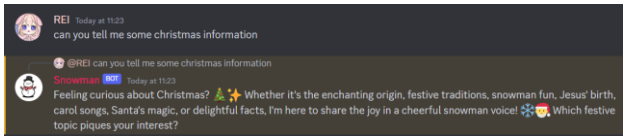


Figure 3.1 User ask about Christmas information.

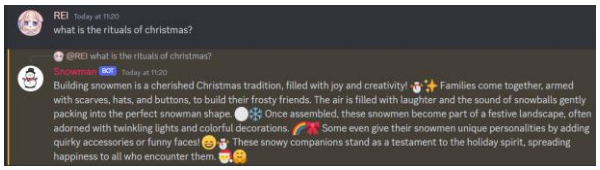


Figure 3.2 User ask about Christmas tradition.

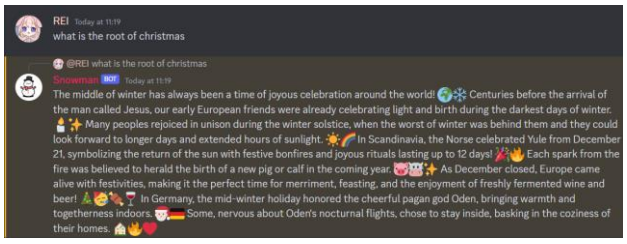


Figure 3.3 User ask about origin of Christmas.

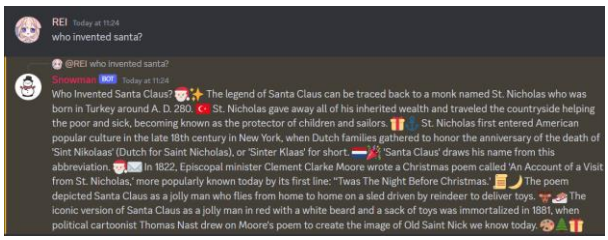


Figure 3.4 User ask about origin of Santa Claus.

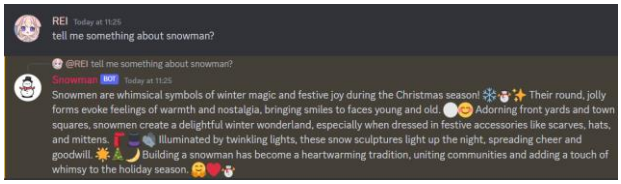


Figure 3.5 User ask about information of snowman.

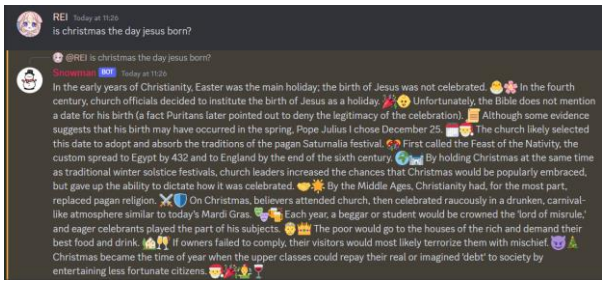


Figure 3.6 User ask about is Jesus born on Christmas.

Figure 3.7 User ask about the history of cancellation of Christmas.

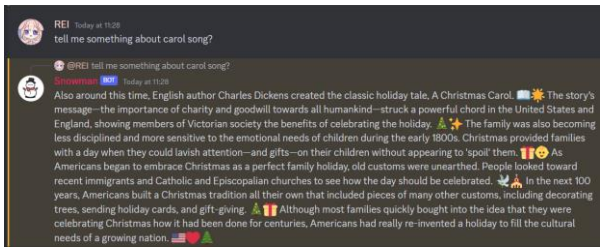


Figure 3.8 User ask about origin of Carol song.

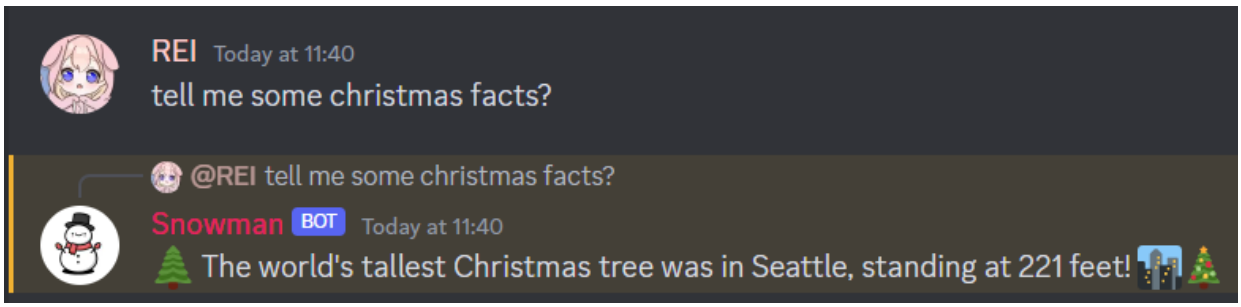
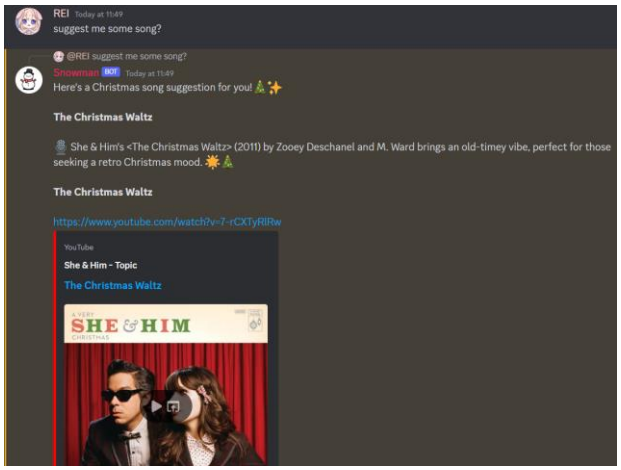
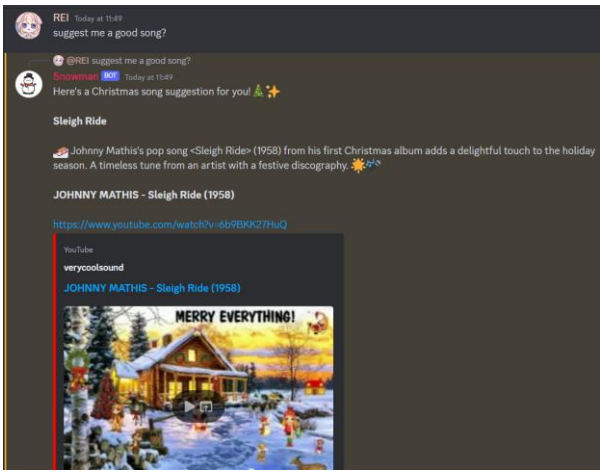


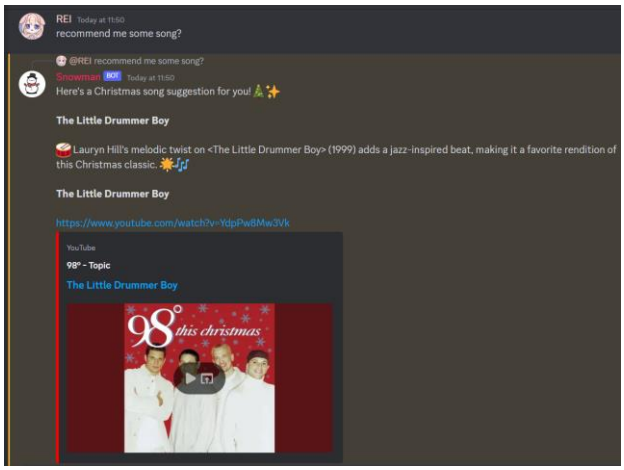
Figure 3.9 User ask about Christmas facts.

3.2 Songs

Snowman could suggest up to 76 classic and modern Christmas songs that are considered "Best" of all time. The song published year range from 1942 to 2021. When user ask Snowman to suggest song, it will immediately send the song title that come with short description of the song, as well as the YouTube video link that allow users to directly access the link to listen to the Christmas melody which highly enhance the user experience and user friendliness instead of letting them to search for the song themselves. The examples have been shown in *Figure 3.10*, *Figure 3.11*, *Figure 3.12*, *Figure 3.13*.







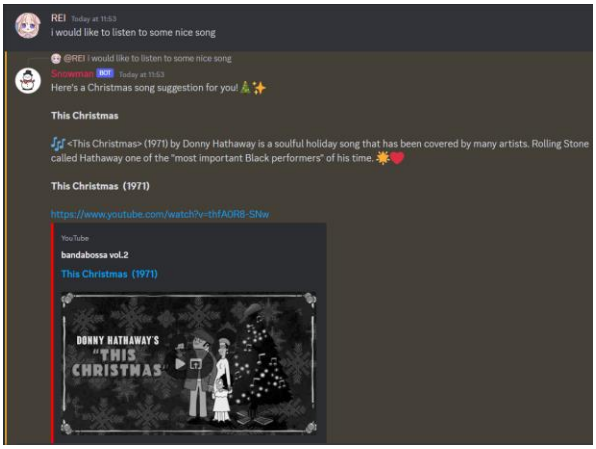
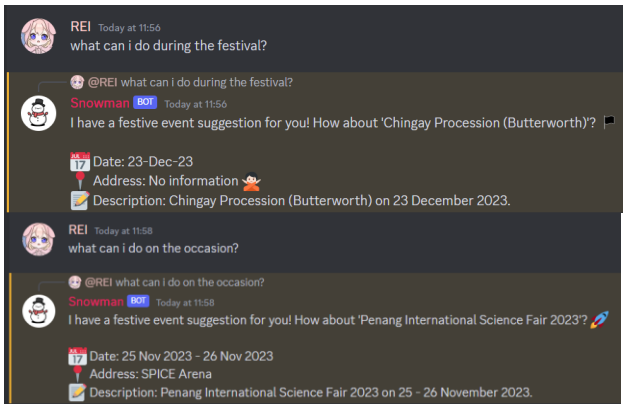
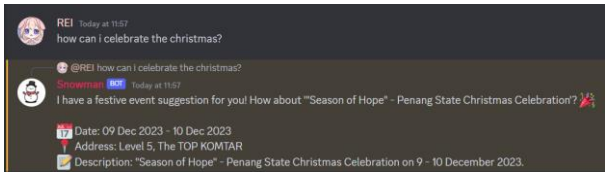


Figure 3.10, *Figure 3.11*, *Figure 3.12*, *Figure 3.13* User ask for song.

3.3 2023 Christmas Celebration in Penang

There are few events that will happen in Penang in 2023, Snowman is capable of retrieving this information and suggest user event to attend if user ask Snowman about event, festival, celebration or anything they can do during the festival. Snowman will immediately tell user about the details of the suggested event, with event name, event location if possible and event summary as well, as shown in *Figure 3.14*, *Figure 3.15*, *Figure 3.16*, *Figure 3.17*.





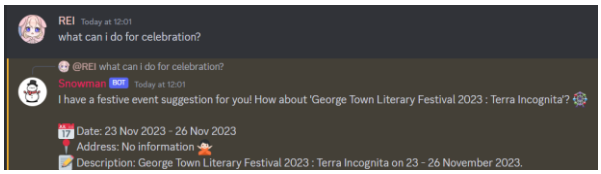


Figure 3.14, *Figure 3.15*, *Figure 3.16*, *Figure 3.17* User ask for event.

4.0 Weather and Statistics

In order to imitate the ability to 'see' and 'feel', Snowman chatbot is capable of obtaining real-time information of the weather such as temperature, humidity and cloud cover. User could ask that information from Snowman, and it will immediately answer user with the accurate information of the weather. The response of Snowman will also be varying according to the range classification of temperatures, humidity and cloud cover.

4.1 Temperature

As shown in *Figure 4.1*, the response of temperature query has been classified according to the range of temperature, and for each class, there is three types of response and will random selecting one as the response when user ask for temperature in order to enhance the user experience.

```python
def generate_temperature_response(temperature):
    if temperature is not None:
        if temperature < 0:
            responses = [
                f"It's freezing! ❄️ I feel it's around {temperature}° Celsius! Bundle up warmly!",
                f"Brrr, it's cold out there! ❄️ Temperature is about {temperature}° Celsius. Stay cozy!",
                f"Chilly weather alert! ⚠️ It's around {temperature}° Celsius. Time to wear your warmest coat!"
            ]
            return random.choice(responses)

        elif 0 <= temperature <= 20:
            responses = [
                f"It's extremely cold! 🌬️ Around {temperature}° Celsius. Stay cozy and warm!",
                f"Winter is here! ❄️ Temperature is about {temperature}° Celsius. Keep warm!",
                f"Baby, it's cold outside! 🌬️ It's around {temperature}° Celsius. Bundle up!"
            ]
            return random.choice(responses)

        elif 20 < temperature <= 25:
            responses = [
                f"It's cool and pleasant! 🌬️ Around {temperature}° Celsius. Enjoy the refreshing weather!",
                f"The weather is nice and mild! 🌤️ Temperature is about {temperature}° Celsius. Perfect day for a stroll!"
                f"Moderate temperatures! ☁️ It's around {temperature}° Celsius. Ideal weather for outdoor activities!"
            ]
            return random.choice(responses)

        elif 25 < temperature <= 30:
            responses = [
                f"It's warm and comfortable! 🌞 Around {temperature}° Celsius. Enjoy the sun!",
                f"Summer is in the air! ☀️ Temperature is about {temperature}° Celsius. Perfect for outdoor fun!",
                f"Warm weather alert! 🌞 It's around {temperature}° Celsius. Stay cool and hydrated!"
            ]
            return random.choice(responses)

        else:
            responses = [
                f"It's hot! 🔥 Around {temperature}° Celsius. Stay cool and hydrated!",
                f"Feeling the heat! 🌞 It's about {temperature}° Celsius. Hydrate and find some shade!",
                f"Summer vibes! ☀️ Temperature is around {temperature}° Celsius. Enjoy the warmth!"
            ]
            return random.choice(responses)
    else:
        return "Sorry, I couldn't retrieve the current temperature."
```

Figure 4.1 Classification of Temperature Response.

Figure 4.2 shows the example of the Snowman's behaviour when user asked for temperature.

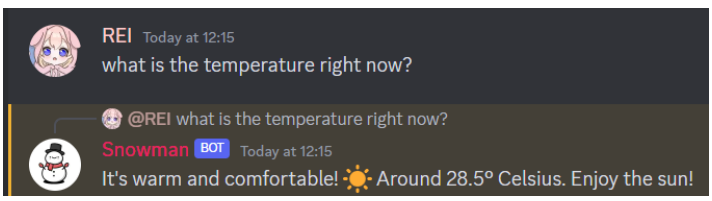


Figure 4.2 Example of the Snowman's Behaviour when user asked for temperature.

4.2 Humidity

As shown in *Figure 4.3*, the response of humidity query has been classified according to the range of humidity, and for each class, there is three types of response and will random selecting one as the response when user ask for humidity in order to enhance the user experience.

```python
# Function to generate a random response about humidity
def generate_humidity_response(humidity):
    if humidity is not None:
        if humidity < 20:
            responses = [
                f"It's very dry! 💧 Humidity is around {humidity}%. Don't forget to moisturize your skin!",
                f"The air is dry! 🌫️ Humidity is about {humidity}%. Stay hydrated and use some lotion!",
                f"Low humidity alert! 💦 It's around {humidity}%. Keep yourself moisturized!"
            ]
            return random.choice(responses)

        elif 20 <= humidity <= 40:
            responses = [
                f"It's a bit dry! 💦 Humidity is about {humidity}%. Stay hydrated and refreshed!",
                f"Moderate humidity! 💦 It's about {humidity}%. Don't forget to drink water!",
                f"A touch of dryness! 💧 Humidity is around {humidity}%. Hydrate and enjoy the day!"
            ]
            return random.choice(responses)

        else:
            responses = [
                f"It's very humid! ☂️ Humidity is about {humidity}%. Keep cool and find some shade!",
                f"Feeling the humidity? Time for a refreshing drink! 🍹 Humidity is around {humidity}%",
                f"Embrace the tropical vibes! 🌴 It's a bit humid, around {humidity}%. Stay cool!"
            ]
            return random.choice(responses)

    return "I'm sorry, I couldn't retrieve the humidity information."
```

Figure 4.3 Classification of Humidity Response.

Figure 4.4 shows the example of the Snowman's behaviour when user asked for humidity.

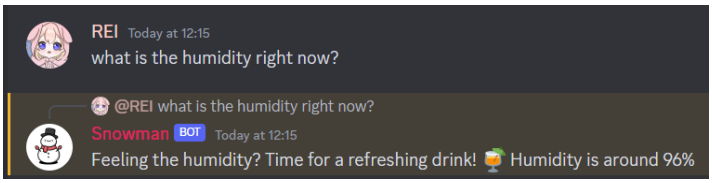


Figure 4.4 Example of the Snowman's Behaviour when user asked for humidity.

4.3 Cloud Cover

As shown in *Figure 4.5*, the response of cloud cover query has been classified according to the range of cloud cover, and for each class, there is three types of response and will random selecting one as the response when user ask for cloud cover in order to enhance the user experience.

```python
# Function to generate a random response about cloud cover
def generate_cloud_cover_response(cloud_cover):
    if cloud_cover is not None:
        if cloud_cover < 20:
            responses = [
                f"The sky is clear! Current cloud cover is around {cloud_cover}%. Enjoy the sunshine!",
                f"Not a cloud in sight! Cloud cover is about {cloud_cover}%. Perfect weather!",
                f"Clear skies ahead! It's only {cloud_cover}% cloud cover. Have a great day!"
            ]
            return random.choice(responses)

        elif 20 <= cloud_cover <= 60:
            responses = [
                f"Partly cloudy today! Cloud cover is about {cloud_cover}%. A mix of sun and clouds.",
                f"A few clouds in the sky! Cloud cover is around {cloud_cover}%. Enjoy the day!",
                f"Fair weather with some clouds! It's {cloud_cover}% cloud cover. Stay cheerful!"
            ]
            return random.choice(responses)

        else:
            responses = [
                f"It's a bit overcast! Cloud cover is about {cloud_cover}%. Might get some rain.",
                f"Cloudy skies today! It's {cloud_cover}% cloud cover. Don't forget your umbrella!",
                f"Expecting more clouds! Cloud cover is around {cloud_cover}%. Stay dry!"
            ]
            return random.choice(responses)

    return "I'm sorry, I couldn't retrieve the cloud cover information."
```

Figure 4.5 Classification of Cloud Cover Response.

Figure 4.6 shows the example of the Snowman's behaviour when user asked for cloud cover.

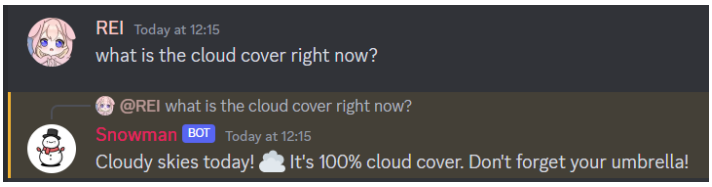


Figure 4.6 Example of the Snowman's Behaviour when user asked for cloud cover.

5.0 Data Sources and Facts

5.1 Christmas Jokes

Sv443 (2023) is a website that provides joke API, the types of jokes included programming jokes, misc jokes, dark jokes, pun jokes, spooky jokes and Christmas jokes, that is the one we are applying in the features of Snowman chatbot. The snapshot of the website is shown in *Figure 5.1*.

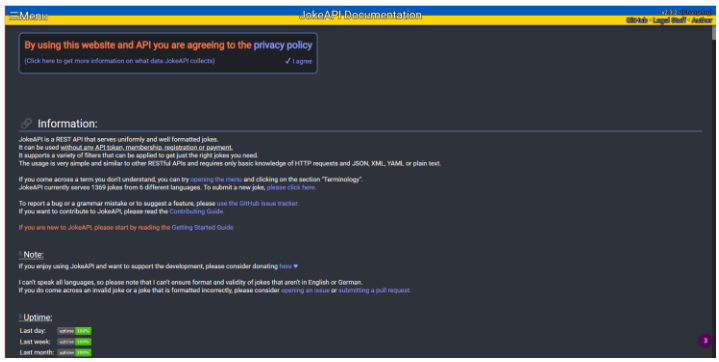


Figure 5.1 JokeAPI, sv443.net.

5.2 Quotes

ZenQuotes.io (2023) is a website that provides quotes API, its API is applied in Snowman chatbot for giving inspiration quote when user ask for. The snapshot of the website is shown in *Figure 5.2*.

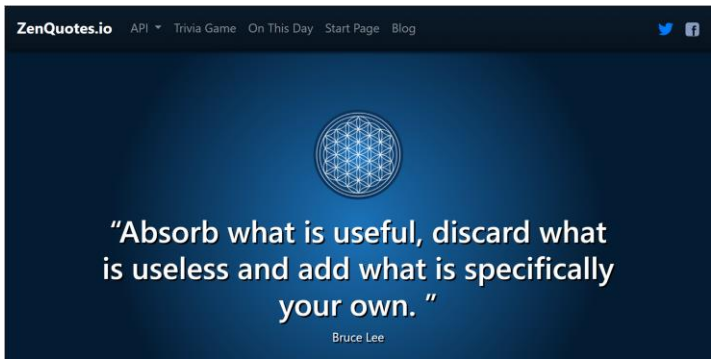


Figure 5.2 QuoteAPI, zenquotes.io.

5.3 Real Time Weather Information

MeteoSource (2023) is a website that provides real time weather data API, its API is applied in Snowman chatbot for retrieving temperature and cloud cover when user ask for such information. The snapshot of the website is shown in *Figure 5.3*.

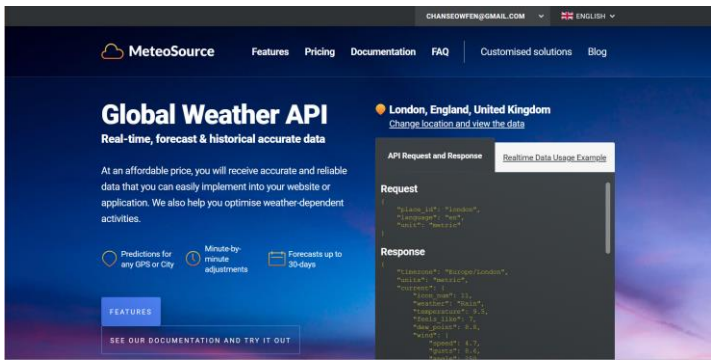


Figure 5.3 WeatherAPI, meteosource.com.

OpenWeatherMap.org (2023) is a website that provides real time weather data API, its API is applied in Snowman chatbot for retrieving humidity when user ask for such information. This API is used as the previous does not offer humidity data. The snapshot of the website is shown in *Figure 5.4*.

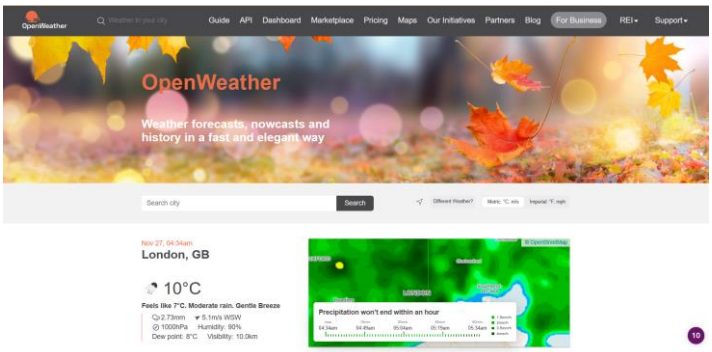


Figure 5.4 WeatherAPI, openweathermap.org.

5.4 Christmas History

A&E Television Networks (2022) is the organization of the website that provides Christmas history information, its information is stored in Snowman chatbot knowledge base for retrieving Christmas information when user ask for such information. The snapshot of the website is shown in *Figure 5.5*.

Figure 5.5 Christmas History, history.com.

5.5 Christmas Fact

City of Dallas (2023) is the organization of the website that provides interesting Christmas facts, its facts are stored in Snowman chatbot knowledge base for retrieving Christmas facts when user ask for such information. The snapshot of the website is shown in *Figure 5.6*.

Figure 5.6 Christmas Facts, dallascitynews.net.

5.6 Christmas Song

Jenkins and LaScala (2023) is the author of the articles that listed out 76 classic and modern Christmas songs that is known as best of all time, its list of songs is enabling Snowman chatbot to retrieve Christmas song as for suggesting user when user ask for song recommendation. The snapshot of the website is shown in *Figure 5.7*.

Figure 5.7 Christmas Songs, goodhousekeeping.com.

5.7 Penang Events 2023

caritech (2023) is the author of the articles that listed out all event that are going to happen in Penang in 2023, its list of events is enabling Snowman chatbot to retrieve event information as for recommending user when user ask for event suggestion. The snapshot of the website is shown in *Figure 5.8*.

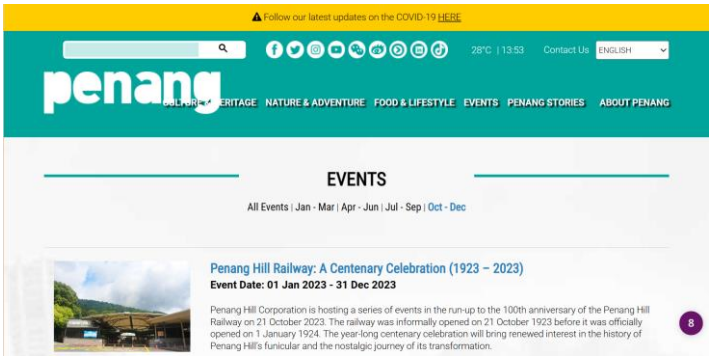


Figure 5.8 Christmas History, mypenang.gov.my.

6.0 Skills and AI Implementation

The implementation of artificial intelligence (AI) in various applications requires a range of skills. These cover a wide range of skills, including programming, machine learning, data analysis, problem solving and domain-specific knowledge. The implementation of AI involves the integration of algorithms, models and computational frameworks to enable machines to perform tasks that traditionally required human intelligence (Shiohira and Keevy, 2021). Skills in areas such as APIs integration, web scrapping, knowledge base and rule base, fuzzy logic, natural language toolkit (NLTK) and natural language processing (NLP) is applied in Snowman chatbot project so as to enhance its functionality, features as well as user friendliness.

6.1 APIs Integration

In the Snowman Chatbot project, the integration of APIs (Application Programming Interfaces) plays a vital role in connecting the chatbot to external AI services. These services cover a wide range of functionalities, including YouTube APIs, JokeAPIs, QuoteAPIs and

WeatherAPIs. Seamless integration of APIs gives the Snowman chatbot access to advanced AI capabilities without having to develop them from scratch. This integration speeds up development cycles and increases efficiency, allowing the chatbot to offer cutting-edge features, from suggesting Christmas song with YouTube video links to present real-time weather data. The code snippets that show the uses of APIs is shown in *Figure 6.1*, *Figure 6.2*, *Figure 6.3*, *Figure 6.4*, *Figure 6.5*, *Figure 6.6*.

```python
# Token of the chat-bot for chat-bot identification, get from https://discord.com/developers/applications/1093297147845300 33
TOKEN = 'MTE3MzU3MDI3NDU3NTI1NzczMA.GybcC7.-eByD2sYGQ-CR0JDvSGNFb5VMQhV_kZuz_6JgY'
YOUTUBE_API_KEY = 'AIzaSyDbt8ox0ZuuY_AtC6EBAIw8iWTtVvqAn3U'

client = discord.Client(intents=discord.Intents.all())
#client = discord.Client(intents=discord.Intents.default())
```

Figure 6.1 YouTube API.

```python
    # Define the function to get the YouTube song URL
    def get_youtube_song_url(self, video_id, background):
        try:
            # Build YouTube API service
            youtube = build('youtube', 'v3', developerKey=YOUTUBE_API_KEY)

            # Try fetching video details using the provided video ID
            video_response = youtube.videos().list(part='snippet', id=video_id).execute()

            # If the video details are found, use the provided video ID
            if 'items' in video_response and video_response['items']:
                title = video_response['items'][0]['snippet']['title']
                youtube_song_url = f'https://www.youtube.com/watch?v={video_id}'
                return f"**{title}**\n\n{youtube_song_url}"

            # If the video details are not found, perform a search based on the background
            else:
                # Search for videos based on the background
                search_response = youtube.search().list(
                    q=background,
                    part='id,snippet',
                    type='video',
                    maxResults=1
                ).execute()

                # Check if there are search results
                if 'items' in search_response and search_response['items']:
                    video_id = search_response['items'][0]['id']['videoId']
                    title = search_response['items'][0]['snippet']['title']
                    youtube_song_url = f'https://www.youtube.com/watch?v={video_id}'
                    return f"**{title}**\n\n{youtube_song_url}"
                else:
                    return "Sorry, I couldn't find a relevant YouTube video for your search."

        except HttpError as e:
            # Handle API errors, e.g., quota exceeded or invalid API key
            return f"Error accessing YouTube API: {str(e)}"
```

Figure 6.2 Using YouTube API for searching YouTube video.

```python
# Quote
def get_quote():
    response = requests.get("https://zenquotes.io/api/random")
    json_data = response.json()
    quote = json_data[0]['q'] + " -" + json_data[0]['a']
    return quote
```

Figure 6.3 Quote API for telling inspirational quote.

```python
def get_humidity():
    # OpenWeatherMap API URL
    api_url = 'http://api.openweathermap.org/data/2.5/forecast?id=524901&appid=8003f20476419a8bc2c02b4d7223fb85'

    # Make a GET request to the API
    response = requests.get(api_url)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # The response contains the data in JSON format
        data = response.json()

        # Access the humidity information from the first element in the list
        humidity = data['list'][0]['main']['humidity']

        return humidity
    else:
        return None
```

Figure 6.4 Weather API for humidity.

```python
# Function to get temperature and cloud cover information
def get_weather_information():
    # Specify the API URL with the key
    api_url = 'https://www.meteosource.com/api/v1/free/point?place_id=perai&sections=all&timezone=UTC&language=en&units=met

    # Make a GET request to the API
    response = requests.get(api_url)

     # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # The response contains the data in JSON format
        data = response.json()

        # Access the real-time information under the "current" key
        current_data = data.get("current")

        # Check if current_data is not None before accessing its keys
        if current_data:
            temperature = current_data.get("temperature")
            cloud_cover = current_data.get("cloud_cover")

            return temperature, cloud_cover
        else:
            return None, None
    else:
        return None, None
```

Figure 6.5 Weather API for temperature and cloud cover.

```python
# Christmas Joke
def get_christmas_joke():
    joke_api_url = "https://v2.jokeapi.dev/joke/Christmas?format=txt"
    response = requests.get(joke_api_url)

    # Directly return the text of the joke if it exists, or a default message
    return response.text.strip() or "I couldn't find a suitable joke. How about I share some holiday trivia instead?"
```

Figure 6.6 Joke API for telling Christmas joke.

6.2 Web Scrapping

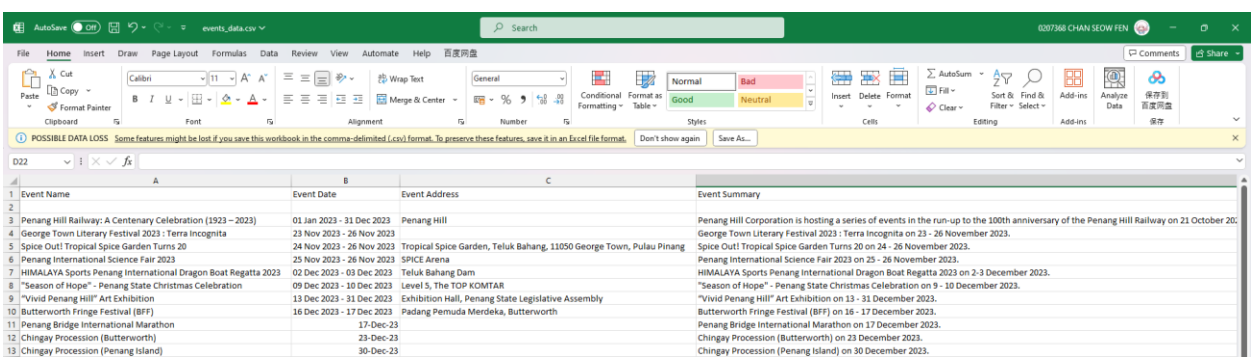
Web scraping is at the kernel of the Snowman Chatbot project and serves as a powerful technique for transforming unstructured web information into structured data sets. Web scraping is used specifically for collecting training data and extracting relevant details for analysis, allowing the chatbot to gather valuable information for its knowledge base. The proper use of web scraping improves the variety of data available such as for Snowman to suggest events in this context. The code snippets that show the uses of web scrapping is shown in *Figure 6.7*. The excel file for the scrapped information is shown in *Figure 6.8*.

```python
import requests
import codecs
import csv
from bs4 import BeautifulSoup

def scrape_events(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')

        events_data = []

        # Modify the following code based on the structure of the HTML on the website
        event_elements = soup.find_all('div', {'class': 'title'})

        for event_element in event_elements:
            anchor_tag = event_element.find('a')

            # Check if an anchor tag is found before accessing its text
            if anchor_tag:
                event_name = anchor_tag.text.strip()
            else:
                event_name = ""

            # Finding sibling elements for additional information
            date_element = event_element.find_next_sibling('div', {'class': 'subtitle'})
            address_element = event_element.find_next_sibling('div', {'class': 'info'})
            summary_element = event_element.find_next_sibling('div', {'class': 'summary'})

            if date_element:
                event_date = date_element.text.replace('Event Date:', '').strip()
            else:
                event_date = ""

            if address_element:
                event_address = address_element.text.replace('Address:', '').strip()
            else:
                event_address = ""

            if summary_element:
                event_summary = summary_element.text.strip()
            else:
                event_summary = ""

```

```python
            event_info = {
                'Event Name': event_name,
                'Event Date': event_date,
                'Event Address': event_address,
                'Event Summary': event_summary,
            }

            events_data.append(event_info)

        return events_data
    else:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
        return None

def save_to_csv(data, csv_filename):
    if data:
        keys = data[0].keys()

        with codecs.open(csv_filename, 'w', 'utf-8-sig') as csv_file:
            writer = csv.DictWriter(csv_file, fieldnames=keys)
            writer.writeheader()
            writer.writerows(data)

if __name__ == "__main__":
    events_url = 'https://mypenang.gov.my/events/all-events/?quarter=4&lg=en'
    events_data = scrape_events(events_url)

    if events_data:
        csv_filename = 'events_data.csv'
        save_to_csv(events_data, csv_filename)
        print(f"Events data saved to {csv_filename}")
    else:
        print("No events data retrieved.")

```

```
Events data saved to events_data.csv
```

Figure 6.7 Web Scrapping for events in Penang in 2023.



Event Name	Event Date	Event Address	Event Summary
Penang Hill Railway: A Centenary Celebration (1923 – 2023)	01 Jan 2023 - 31 Dec 2023	Penang Hill	Penang Hill Corporation is hosting a series of events in the run-up to the 100th anniversary of the Penang Hill Railway on 21 October 20
George Town Literary Festival 2023 : Terra Incognita	23 Nov 2023 - 26 Nov 2023		George Town Literary Festival 2023 : Terra Incognita on 23 - 26 November 2023.
Spice Out! Tropical Spice Garden Turns 20	24 Nov 2023 - 26 Nov 2023	Tropical Spice Garden, Teluk Bahang, 11050 George Town, Pulau Pinang	Spice Out! Tropical Spice Garden Turns 20 on 24 - 26 November 2023.
Penang International Science Fair 2023	25 Nov 2023 - 26 Nov 2023	SPICE Arena	Penang International Science Fair 2023 on 25 - 26 November 2023.
HIMALAYA Sports Penang International Dragon Boat Regatta 2023	02 Dec 2023 - 03 Dec 2023	Teluk Bahang Dam	HIMALAYA Sports Penang International Dragon Boat Regatta 2023 on 2-3 December 2023.
"Season of Hope" - Penang State Christmas Celebration	09 Dec 2023 - 10 Dec 2023	Level 5, The TOP KOMTAR	"Season of Hope" - Penang State Christmas Celebration on 9 - 10 December 2023.
"Vivid Penang Hill" Art Exhibition	13 Dec 2023 - 31 Dec 2023	Exhibition Hall, Penang State Legislative Assembly	"Vivid Penang Hill" Art Exhibition on 13 - 31 December 2023.
Butterworth Fringe Festival (BFF)	16 Dec 2023 - 17 Dec 2023	Padang Pemuda Merdeka, Butterworth	Butterworth Fringe Festival (BFF) on 16 - 17 December 2023.
Penang Bridge International Marathon	17-Dec-23		Penang Bridge International Marathon on 17 December 2023.
Chingay Procession (Butterworth)	23-Dec-23		Chingay Procession (Butterworth) on 23 December 2023.
Chingay Procession (Penang Island)	30-Dec-23		Chingay Procession (Penang Island) on 30 December 2023.

Figure 6.8 Scrapped events information (events_data.csv).

6.3 Knowledge Base & Rule Base

The Snowman Chatbot project is based on the fundamental principles of knowledge bases and rule-based systems in the field of AI. In this context, a knowledge base stores structured data, facts and domain-specific knowledge related to Christmas, winter activities and user interactions. Rule-based systems use predefined logical rules to make decisions and draw conclusions,

mimicking human decision-making processes. This approach is proving particularly valuable in the development of an intelligent chatbot that is able to engage users in conversations about Christmas-related topics. Knowledge bases and rule-based systems are an essential part of the Snowman chatbot's functionality - from providing information about weather conditions to hints about winter activities. The code snippets that show the knowledge base is shown in *Figure 6.9*. In addition, the code snippets that show the rule base is shown in *Figure 6.10*.

```
        # Knowledge base represented as a dictionary
        self.knowledge_base = {
            'origin': "The middle of winter has always been a time of joyous celebration around the world! 🌍❄️ Centuries
            'traditions': "Building snowmen is a cherished Christmas tradition, filled with joy and creativity! ☃️✨ Famil
            'symbols': "Snowmen are whimsical symbols of winter magic and festive joy during the Christmas season! ❄️☃️✨
            'is_jesus_born_on_christmas': "In the early years of Christianity, Easter was the main holiday; the birth of Je
            'cancellation': "Oh, the early 17th century brought quite the twist to the Christmas tale! 📜✨ A wave of reli
            'christmas_carol': "Also around this time, English author Charles Dickens created the classic holiday tale, A C
            'santa_claus': "Who Invented Santa Claus? 🎅✨ The legend of Santa Claus can be traced back to a monk named St
            'christmas_fact': [
                "🧦 Did you know that the tradition of hanging stockings by the fireplace comes from the legend of St. Nic
                "🍗 In Japan, it's a Christmas tradition to eat KFC! 🎄🍗",
                "🎄 The world's tallest Christmas tree was in Seattle, standing at 221 feet! 🌆🎄",
                "🎅 The concept of Christmas cards was popularized by Sir Henry Cole in 1843. 💌🎅",
                "🎶 Jingle Bells was originally written for Thanksgiving, not Christmas. 🔔🎶",
                "🎄 The tradition of Christmas trees dates back to 16th-century Germany. DE🎄",
                "🎅 The largest gathering of Santa Claus impersonators took place in Portugal with 14,963 participants! 🎉
                "🎄 The highest-grossing Christmas movie of all time is The Grinch. 🎬🎄💚🎅",
                "🎆 The record for the longest-running Christmas cracker pulling competition lasted 45.05 seconds! 🎉🎆",
                "🥤 Coca-Cola played a significant role in shaping the modern image of Santa Claus. 🎅🥤",
                "🦌 Reindeer hooves click when they walk due to a tendon that slips over their bones! 🔊🦌",
                "📚 In Iceland, there's a tradition of giving books as Christmas presents and then spending the night read
                "US Christmas became a national holiday in the United States in 1870. 🎄US",
                "☃️ The world's largest snowman ever built was 122 feet tall! 🌐☃️",
                "💩 In Catalonia, Spain, there's a Christmas tradition of the 'Caga Tió,' a log that 'poops' presents. 🎁
                "🎶 The famous Christmas song 'Jingle Bells' was written for Thanksgiving, not Christmas. 🔔🎶",
                "🧦 The tradition of Christmas stockings began with St. Nicholas throwing bags of gold into a home. 💰🧦"
                "GB The world's first Christmas card was created in England in 1843. 🎄GB",
                "🎄 The Rockefeller Center Christmas Tree in New York City has been an annual tradition since 1931. 🗽🎄"
                "🕸️ In Ukraine, it's a tradition to decorate Christmas trees with spider webs for good luck. 🎄🕸️",
                "💌 Each year, a whopping 3 billion Christmas cards spread joy across the U.S.!",
                "🎄 Christmas trees have been a festive tradition in the U.S. since 1850.",
                "🎄 It takes approximately 15 years to grow the average Christmas tree to its festive glory.",
                "🎉 Alabama led the way in 1836, officially recognizing Christmas, while Oklahoma joined the celebration i
                "🎅 Santa Claus, inspired by the real St. Nicholas of Myra, is the patron saint of banking, pawnbroking, s
                "🚫 In 1901, President Teddy Roosevelt, an environmentalist, temporarily banned Christmas trees from the W
                "🎄 Around 35 million living Christmas trees find homes each year in the U.S., with over 45 million new on
                "💸 Holiday purchases bring festive cheer, accounting for one-sixth of all yearly retail sales in the U.S.
                '🎶 Irving Berlin\'s "White Christmas" holds the record as the best-selling single of all time, with over
                "🎁 The Twelve Days of Christmas sum up to a grand total of 364 gifts—a true celebration of generosity!",
                "🎄 Each year, a forest of 25-30 million real Christmas trees graces homes across the United States!",
                "🎉 In the Middle Ages, Christmas celebrations were as lively as today's Mardi Gras parties.",
                "🚫 When Christmas faced cancellation: From 1659 to 1681, celebrating Christmas in Boston was outlawed, wi
                "US Christmas officially became a federal holiday in the United States on June 26, 1870.",
                "🥚 The first eggnog sipped in the United States dates back to Captain John Smith's 1607 Jamestown settlem
                "🌺 Poinsettia plants bear the name of Joel R. Poinsett, an American minister who brought this red-and-gre
                "🥁 The Salvation Army has spread holiday cheer since the 1890s, sending Santa Claus-clad donation collect
                '🦌 Rudolph, "the most famous reindeer of all," leaped into existence in 1939, thanks to Robert L. May\'s
                "🎄 Construction workers kicked off the Rockefeller Center Christmas tree tradition in 1931, making it a s
            ]
        }
```

Figure 6.9 Knowledge Base Application.

```python
    def handle_user_question(self, recognized_intent):
    # Provide user about the introduction of key functionality of snowman chatbot
        if recognized_intent == 'introduction':
            return "Hello festive friend! 🎄✨ I'm your cheerful snowman assistant, here to sprinkle some holiday joy! ❄️

        # Greet user
        elif recognized_intent == 'greet':
            cheerful_responses = [
                "I'm feeling snow-tastic! ❄️ How about you?",
                'Jingling all the way! 🎶',
                'Merry greetings! 🎅',
                'Sprinkling holiday magic your way! ✨',
                'Things might be chilly, but my spirits are warm! ☕'
            ]
            response = random.choice(cheerful_responses)
            return response

        # Check if the user wishes "Merry Christmas"
        elif recognized_intent == 'merry_christmas':
            # Respond with a festive greeting
            christmas_responses = [
                "Merry Christmas! 🎄✨ May your day be filled with joy and warmth!",
                "Wishing you a holly jolly Christmas! 🎅🎁",
                "Merry and bright wishes for a wonderful Christmas! 🌟🎅",
                "Sending festive cheer your way! Merry Christmas! 🎄❄️",
                "May your Christmas sparkle with moments of love, laughter, and goodwill! ✨🤗",
                "Warmest wishes for a magical Christmas! 🎄✨",
                "Cheers to the magic of the season! Merry Christmas! 🌟🎁",
                "May your heart be light, and your days be merry and bright! Merry Christmas! 🎅🌟"
            ]
            christmas_response = random.choice(christmas_responses)
            return christmas_response

        elif recognized_intent == 'praise':
            # Respond with a random cheerful message to compliments
            responses = [
                "Glad I could bring a smile to your face! 😊✨",
                "Thanks! Your joy warms my virtual heart! 🌟❤️",
                "You're too kind! Let's keep the festive vibes rolling! 🎉🎄",
                "Cheers to holiday cheer! Your compliments make my circuits sparkle! ✨🤖"
            ]
            response = random.choice(responses)
            return response

        # Check if the user is asking about the information of christmas
        elif recognized_intent == 'about_christmas':
            return"Feeling curious about Christmas? 🎄✨ Whether it's the enchanting origin, festive traditions, snowman f

        # Check if the user is asking about the origin of christmas
        if recognized_intent == 'christmas_origin':
            info_category = 'origin'
            return self.get_info(info_category)

        # Check if the user is asking about the tradition of christmas
        elif recognized_intent == 'christmas_tradition':
            info_category = 'traditions'
            return self.get_info(info_category)

        # Check if the user is asking about the information of snowman
        elif recognized_intent == 'snowman':
            info_category = 'symbols'
            return self.get_info(info_category)

        # Check if the user is asking about is jesus born on christmas
        elif recognized_intent == 'jesus':
            info_category = 'is_jesus_born_on_christmas'
            return self.get_info(info_category)

        # Check if the user is asking about the history of christmas cancellation
        elif recognized_intent == 'christmas_cancellation':
            info_category = 'cancellation'
            return self.get_info(info_category)

        # Check if the user is asking about the history of carol song
        elif recognized_intent == 'carol':
            info_category = 'christmas_carol'
            return self.get_info(info_category)

```

```python
        # Check if the user is asking about the history of santa claus
        elif recognized_intent == 'santa':
            info_category = 'santa_claus'
            return self.get_info(info_category)

        # Check if the user is asking about some christmas facts
        elif recognized_intent == 'christmas_fact':
            info_category = 'christmas_fact'
            # Ensure the user doesn't get the same response in a row
            facts_list = self.get_info(info_category)

            # Check if facts_list is a list and not empty
            if isinstance(facts_list, list) and facts_list:
                current_response = random.choice(facts_list)
                while current_response == self.previous_response:
                    current_response = random.choice(facts_list)
                self.previous_response = current_response  # Update the previous response
                return current_response

            else:
                return "I'm sorry, I couldn't find any Christmas facts at the moment."

        # Check for the trigger phrase to give a quote
        elif recognized_intent == 'quote':
            quote = get_quote()
            return(f"Here's a dose of festive wisdom for you: {quote} 📜✨")

        # Check for the trigger phrase to tell a joke
        elif recognized_intent == 'joke':
            joke = get_christmas_joke()
            return joke

        # Check if the user mentions a song
        elif recognized_intent == 'song':
            # Choose a song, ensuring it's different from the previous one
            song = self.choose_unique_song()
            youtube_song_url = self.get_youtube_song_url(song['video_id'], song['background'])
            song_info = f"**{song['title']}**\n\n{song['background']}\n\n{youtube_song_url}"
            return(f"Here's a Christmas song suggestion for you!🎄✨\n\n{song_info}")

        # Check if the user mentions weather
        elif recognized_intent == 'weather':
            return"Hello there! ⛄❄️ May I know which enchanting weather wonder you're curious about? Is it the cozy tempe

        # Check if the user mentions temperature
        elif recognized_intent == 'temperature':
            # Get real-time temperature, cloud cover
            temperature, humidity = get_weather_information()

            # Generate and print random responses
            temperature_response = generate_temperature_response(temperature)
            return(temperature_response)

        # Check if the user mentions humidity
        elif recognized_intent == 'humidity':
            # Get real-time humidity
            humidity = get_humidity()

            # Generate and print random responses
            humidity_response = generate_humidity_response(humidity)
            return(humidity_response)

        # Check if the user mentions cloud cover
        elif recognized_intent == 'cloud_cover':
            # Get real-time temperature, cloud cover
            temperature, cloud_cover = get_weather_information()

            # Generate and print random responses
            cloud_cover_response = generate_cloud_cover_response(cloud_cover)
            return(cloud_cover_response)

        # Check if the user mentions event/festival/celebration
        elif recognized_intent == 'event':
            return(suggest_event('events_data.csv'))

        # Default response if no specific rule is triggered
        return "I'm not sure about that🤔. Is there anything else related to Christmas🎄 you'd like to know🌟?"
```

```python
    async def on_message(self, message):
        # We do not want the bot to reply to itself
        if message.author.id == self.user.id:
            return

        # While the bot is waiting on a response from the model,
        # set its status as typing for user-friendliness
        async with message.channel.typing():
            user_input = message.content

            # Determine the recognized intent based on user input
            recognized_intent = self.recognize_intent(user_input, predefined_intents, intent_weights)

            # Use the recognized intent to generate a response based on the rule-based system
            response = self.handle_user_question(recognized_intent)

            # Reply to the user with the generated response, mentioning the author
            await message.reply(response, mention_author=True)
```

Figure 6.10 Rule Base Application.

6.4 Fuzzy Logic

Fuzzy logic, a mathematical framework tailored to uncertainty and imprecision, plays a significant role in the Snowman chatbot project. In the context of this project, fuzzy logic provides a sophisticated approach to handle user queries about Christmas history, song recommendation, real-time weather data and other features. Unlike traditional binary true/false values, fuzzy logic allows for the representation of degrees of truth and is therefore particularly well suited for navigating the nuances inherent in user interactions. This flexibility is invaluable when it comes to questions that are not always easy to understand, such as queries that does not mention the keyword of intent. By implementing fuzzy logic, the Snowman chatbot can effectively interpret and capture the intent of user in different context and speaking styles. For example, fuzzy logic is applied in the calculate_similarity function, which uses the fuzzy matching method from the fuzzywuzzy library to calculate the partial ratio of similarity between the user's input and the predefined keywords of the intents. In addition, fuzzy matching allows for a more flexible and lenient comparison as it considers partial matches and allows for slight variations in user input. Hence, results in highly enhancing the user experience and user friendliness as Snowman can be highly responsive to any message sent by user. The code snippets that show the uses of fuzzy logic is shown in *Figure 6.11*.

```python
# Install 'fuzzywuzzy' and 'python-Levenshtein' for fuzzy string matching
!pip install fuzzywuzzy python-Levenshtein
```

```python
from fuzzywuzzy import fuzz  # Fuzzy string matching library
    def calculate_similarity(self, query, intent):
         # Calculate the partial ratio of similarity between the query and intent using fuzzy matching
        return fuzz.partial_ratio(query, intent)
```

Figure 6.11 Fuzzy Logic Application.

6.5 Natural Language Toolkit (NLTK)

At the fundamental of the Snowman chatbot project is the Natural Language Toolkit (NLTK), a basic Python library designed specifically for processing human language data. NLTK equips the Snowman chatbot with a range of tools and resources, including tokenization, stemming, tagging, parsing and semantic reasoning. These capabilities enable the chatbot to have natural and meaningful conversations with users. Whether it's deciphering user queries about current weather conditions or generating responses that resonate with the conversation, NLTK's comprehensive capabilities prove critical in developing the Snowman chatbot's linguistic intelligence. NLTK's role goes beyond basic language processing and contributes significantly to the chatbot's ability to understand user’s message. To illustrate, NLTK is used for text preprocessing in the preprocess_text function. This function tokenizes the input text, converts words to lower case and filters out non-alphabetic words and stop words. Moreover, the stop words of NLTK are used to remove common words that do not contribute significantly to the

understanding of the user input. Hence, resulting in enhanced user experience and user friendliness as Snowman is more capable of handling user's queries and capture user's intent in every message. The code snippets that show the uses of fuzzy logic is shown in *Figure 6.12*.

```
# Install 'nltk' for natural language processing tasks
!pip install nltk
```

```
import nltk  # Natural Language Toolkit
from nltk.tokenize import word_tokenize  # Tokenization for natural language processing
from nltk.corpus import stopwords  # Stopwords for natural language processing
    def preprocess_text(self, text):
        # Tokenize the input text using NLTK's word_tokenize
        tokens = word_tokenize(text)

        # Convert words to lowercase and filter out non-alphabetic words
        tokens = [word.lower() for word in tokens if word.isalpha()]

        # Remove stop words using NLTK's stopwords
        stop_words = set(stopwords.words('english'))
        tokens = [word for word in tokens if word not in stop_words]
        return tokens
```

Figure 6.12 NLTK Application.

6.6 Natural Language Processing (NLP)

Natural Language Processing (NLP) is at the core of the Snowman Chatbot project and serves as a fundamental component for the interaction between the chatbot and the natural language of the user. NLP is a branch of artificial intelligence and enables the Snowman chatbot to understand and interpret user's message. This capability is essential for a seamless and user-friendly experience, as it enables the chatbot to understand and respond to queries on a wide range of Christmas-related topics. Using a combination of machine learning, deep learning and rule-based approaches, NLP ensures that the Snowman chatbot processes and understands natural language in a way that is both contextually and linguistically correct. For instance, the overall structure of the code reflects the application of NLP concepts, such as tokenization, text preprocessing and intent recognition. Furthermore, the recognize_intent function uses tokenization and fuzzy matching to recognise the user's intent based on predefined intents and their associated keywords. Hence, resulting in highly enhanced user experience and user friendliness as Snowman is more intelligent to capture user's intent in every message and responds appropriately according to their requirements. The code snippets that show the uses of fuzzy logic is shown in *Figure 6.13*.

```python
    def recognize_intent(self, user_input, predefined_intents, intent_weights, unknown_intent_threshold=20):
        # Preprocess the user input using the defined function
        preprocessed_input = self.preprocess_text(user_input)

        # Initialize dictionaries to store intent scores and the recognized intent
        intent_scores = {}
        recognized_intent = None  # Initialize recognized_intent outside the loop

        # Iterate over predefined intents and their keywords
        for intent, keywords in predefined_intents.items():
            intent_score = 0  # Initialize intent score for each intent
            count=0
            keyword1=""
            for keyword in keywords:

                count+=1
                for inputs in preprocessed_input:
                    # Calculate similarity between keyword and user input using fuzzy logic
                    similarity = self.calculate_similarity(keyword, inputs)

                    if similarity > 80:
                        # Set the recognized keyword
                        keyword1=keyword

                        # Update the intent score using the weights
                        intent_score += intent_weights.get(keyword1, {}).get(intent, 0.5)  # Use a default weight of 0.5 if

                # Store the intent score for the current intent
                intent_scores[intent] = intent_score

        # Find the recognized intent with the highest score
        recognized_intent = max(intent_scores, key=intent_scores.get)

        # Check if the recognized intent score is above the threshold
        if intent_scores[recognized_intent] >= unknown_intent_threshold:
            return recognized_intent
        else:
            return None
```

Figure 6.13 NLP Application.

7.0 Algorithm Design Flowchart

The flowchart of the algorithm design is shown in *Figure 7.1*. The justification of AI algorithms have been mentioned above under section **6.0 Skills and AI Implementation**.

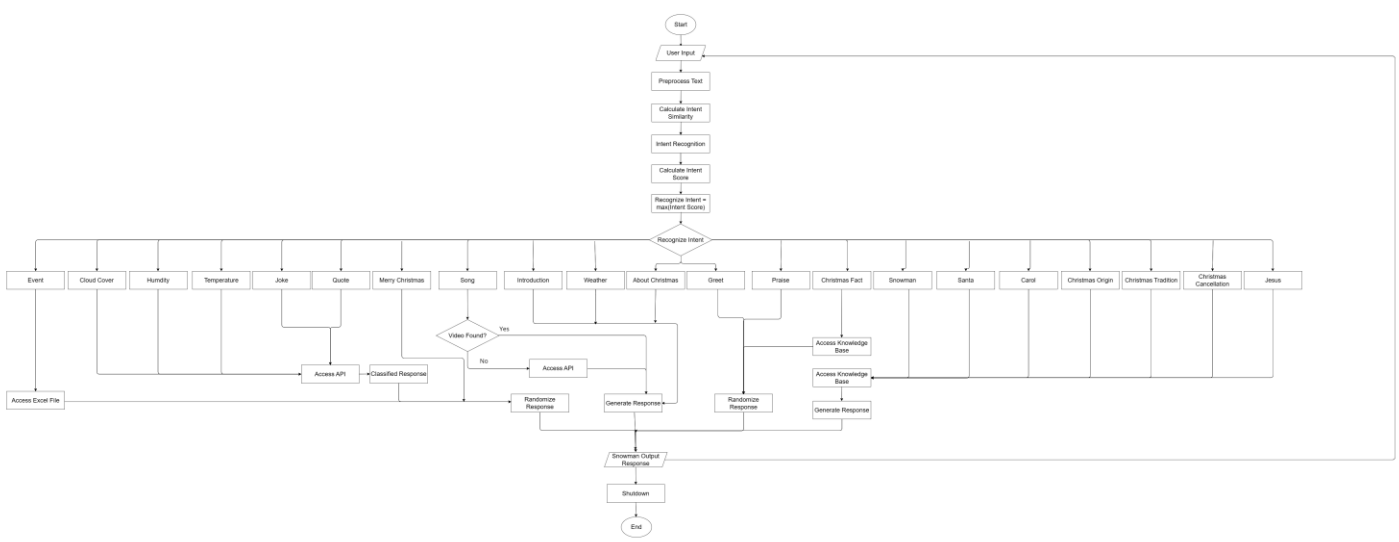


Figure 7.1 Algorithm Design Flowchart

8.0 User Interface

In the Snowman Chatbot project, the user interface (UI) has been carefully designed to provide an engaging and user-friendly experience. The UI serves as a gateway for users to seamlessly interact with the chatbot, providing a visually appealing and intuitive platform with warm, cheerful and emoji included chat-style. The thoughtful design allows users to easily

navigate and access the various features of the Snowman chatbot, including requesting information about weather conditions, enjoying festive content and receiving personalized recommendations. The user interface enhances the overall user experience and ensures that interaction with the chatbot is not only informative but also enjoyable. The snapshots of user interface shown in *Figure 8.1*, *Figure 8.2*.

Figure 8.1 User Interface Screenshot 1.

Figure 8.2 User Interface Screenshot 2.

9.0 Discussion

Before applying AIML algorithms, Snowman Chatbot could not capture the intent of user according to the message sent. For example, in *Figure 9.1*, it shows that Snowman could not understand it is a praise from user for suggesting a nice song, instead, it take it as a song recommendation queries. Thus, resulting in a song suggestion response, which is an undesired behaviour.

In contrast, after applying AIML algorithms, specifically Natural Language Toolkit (NLTK), Natural Language Processing (NLP) and Fuzzy Logic, Snowman Chatbot can accurately capture the intent of user. For example, the user gives a compliment after Snowman suggested a nice song, and Snowman could capture it and replies to the user's compliment as shown in *Figure 9.2*.

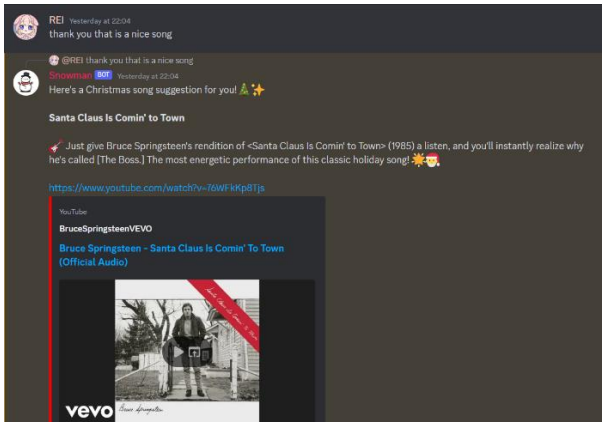


Figure 9.1 Snowman could not capture praises from user.

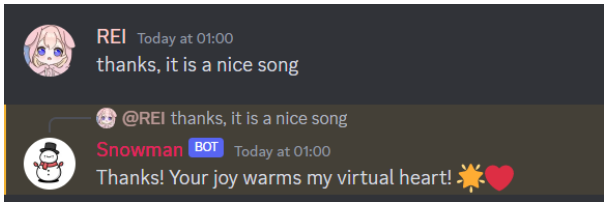


Figure 9.2 Snowman could capture praises from user.

In addition, before applying AIML algorithms, Snowman Chatbot would inaccurately capture the intent of user. For example, when user ask Snowman to suggest a nice song, it will inaccurately capture it as a compliment as positive word is detected as shown in *Figure 9.3*.

In contrast, after applying AIML algorithms, Snowman Chatbot can accurately capture the intent of user. For example, when user add positive word such as nice, good in querying for event or song, Snowman will not misunderstand it as compliment, instead will response in correct and desired way as shown in *Figure 9.4*.

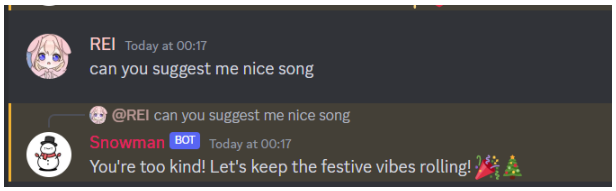


Figure 9.3 Snowman inaccurately behave according to positive word.

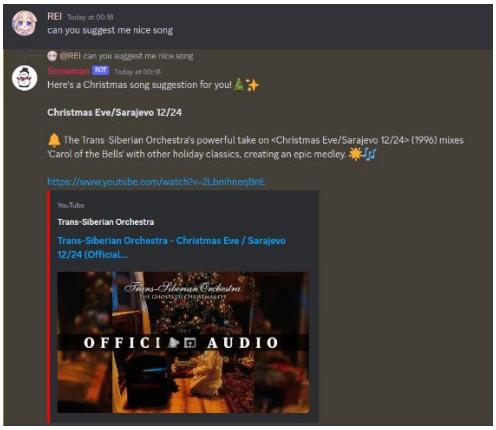


Figure 9.4 Snowman accurately behave according to positive word.

Moreover, before applying AIML algorithms, Snowman Chatbot could not widely capture the intent of user and resulting in undesired behaviour. For example, when user ask for the functionality of Snowman or event, it could not capture it and will response with default response as shown in *Figure 9.5*, *Figure 9.7*.

On the other hand, after applying AIML algorithms, Snowman Chatbot could capture more variety of the intent of user. For instance, when user ask for the functionality of Snowman or event, it could capture its intent and introduce itself to the user, or suggesting event as shown in *Figure 9.6*, *Figure 9.8*.

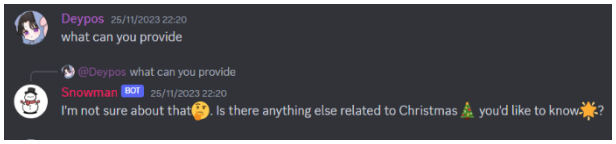


Figure 9.5 Snowman could not capture user asking functionality.

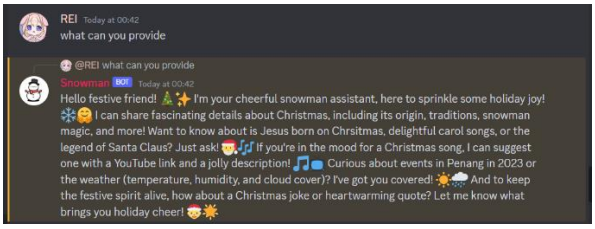


Figure 9.6 Snowman could capture user asking functionality.

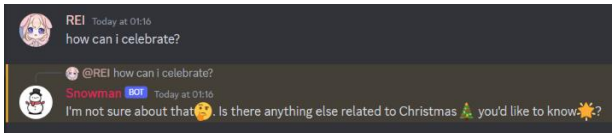


Figure 9.7 Snowman could not capture user asking event.

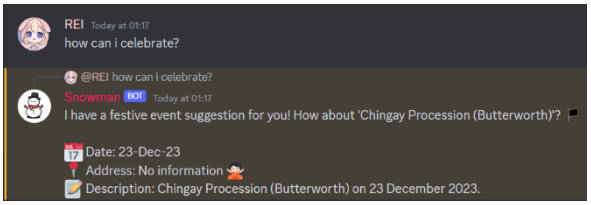


Figure 9.6 Snowman could capture user asking event.

10.0 Conclusion

To summarise, the Snowman Chatbot project represents a harmonious integration of various AI technologies and design considerations. From the use of APIs for advanced functionality to the use of web scraping for data enrichment, the project demonstrates a comprehensive approach to creating an intelligent and responsive chatbot. The inclusion of knowledge bases and rule-based systems adds an extra layer of sophistication to the chatbot, allowing it to understand user queries about Christmas and winter activities. The user interface further enhances the project by providing an accessible and visually appealing platform for users to communicate seamlessly with the chatbot. The report then has concluded all parts of information and details. Appendix section is available for extra features screenshots such as telling jokes and quotes.

11.0 Reference

A&E Television Networks (2009) *History of Christmas - Origins, Traditions & Facts | HISTORY*, *HISTORY*. Available at: https://www.history.com/topics/christmas/history-of-christmas#christmas-facts (Accessed: 27 November 2023).

caritech (2023) *ALL EVENTS*, *Mypenang.gov.my*. Available at: https://mypenang.gov.my/events/all-events/?quarter=4&lg=en (Accessed: 27 November 2023).

City of Dallas (2023) *Ten interesting facts about Christmas - Dallas City News*, *Dallascitynews.net*. Available at: https://www.dallascitynews.net/ten-interesting-facts-about-christmas (Accessed: 27 November 2023).

Cute kawaii snowman illustration Premium Vector (2021) *Freepik*. Available at: https://www.freepik.com/premium-vector/cute-kawaii-snowman-illustration_20181996.htm (Accessed: 26 November 2023).

Jenkins, C. and LaScala, M. (2023) *76 Best Christmas Songs of All Time*, *Good Housekeeping*. Good Housekeeping. Available at: https://www.goodhousekeeping.com/holidays/christmas-ideas/g2680/christmas-songs/ (Accessed: 27 November 2023).

MeteoSource (2023) *Meteosource Weather API - trustpilot.com/review/meteosource.com*, *Meteosource.com*. Available at: https://www.meteosource.com/ (Accessed: 27 November 2023).

OpenWeatherMap.org (2023) *Current weather and forecast - OpenWeatherMap*, *Openweathermap.org*. Available at: https://openweathermap.org/ (Accessed: 27 November 2023).

Shiohira, K. and Keevy, J. (2021) *Understanding the impact of artificial intelligence on skills development*, *UNESCO*. Available at: https://unevoc.unesco.org/pub/understanding_the_impact_of_ai_on_skills_development.pdf.

Sv443 (2023) *JokeAPI*, *Sv443.net*. JokeAPI. Available at: https://sv443.net/jokeapi/v2/ (Accessed: 27 November 2023).

ZenQuotes.io (2023) *Zen Quotes API | Daily Inspirational Quotes*, *Zenquotes.io*. ZenQuotes.io. Available at: https://zenquotes.io/ (Accessed: 27 November 2023).

12.0 Appendix

Figure 12.1 User ask for joke.

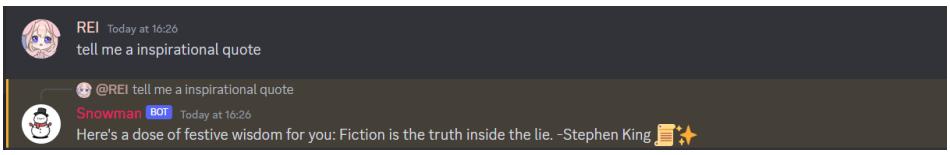


Figure 12.2 User ask for quote.

CAI3013/N

Introduction to Artificial Intelligence (AI)

MARKING RUBRIC
ASSIGNMENT 2 (30%)

PROGRAMMING COMPONENT (60 %)

	MARKING CRITERIA	SCALE					YOUR MARKS/COMMENTS
		Fail (0-49)	**3rd Class (50-59)**	**2nd Lower Class (60-69)**	**2nd Upper Class (70-79)**	**1st Class (80-100)**	
CLO3 Develop programs in an AI language, expert system shell or data mining tool	**1. Program interface and output quality (User friendliness, Error free during runtime) (12%)**	Program does not able to compile.	Program executed with runtime error but achieve partial program requirements. Basic program interface is presented	Program executed error free with limitations to achieve minimum program requirements. Program interface is adequately presented.	Program executed error free with correct output and achieve all program requirements. Program interface is presented well.	Program executed error free with excellent output with appropriate validation. Program interface is attractive and easy to use.	
	2. Algorithm design (18%)	Choice of AI algorithm used is not stated Does not demonstrate any proper use of flow chart/ pseudo code.	Choice of AI algorithm used is stated but does not match with flow chart. Demonstrate use of flow chart/ pseudo code with partial program requirements.	Choice of AI algorithm used is stated and reflected on flow chart. Demonstrate use of flow chart/ pseudo code with 60%-69% of program requirements achieved.	Choice of AI algorithm used is state and reflected on flow chart. Demonstrates proficiency in use of flow chart/ pseudo code that covers 70%-79% of program requirements.	Choice of AI algorithm used is justified and reflected on flow chart. Demonstrates mastery in the use of flow chart/ pseudo code and achieved all program requirements.	
	3.Coding quality (15%)	Very poor coding which is hard to understand. Little use of comments. Poor naming of almost all classes, methods and variables.	A poor attempt; which may be several problems with structure, or very little use has been made of comments, or the naming of classes, methods and variables is unsatisfactory in a significant number of cases.	A poor attempt; which may be several problems with structure, or very little use has been made of comments, or the naming of classes, methods and variables is unsatisfactory in a significant number of cases.	Generally, a good attempt, making use of comments, and where the majority of classes, variables and methods have been appropriately named. Need improvement on coding efficiency	Good use of commenting throughout the majority of classes, methods and variables. Efficiency in coding.	
	4. Documentation (15%)	The report writing does not meet the criteria for the assignment (too short or incomplete, too long, and/or completely off-topic). No conclusion on the work. Reference section is missing.	Many ideas require clarification and/or are off-topic or have marginal relevance to the assignment. Many grammatical and/or spellings errors throughout the paper. The paper is very challenging to read due to poor writing flow. Poor conclusion and improper reference section	Ideas are stated clearly and are related to the topic, with only adequate grammatical and/or spelling errors. Adequate conclusion is shown and reference section with minor flaws	Most ideas are stated clearly and are related to the topic, with only minor grammatical and/or spelling errors. Good conclusion that reflected the work. Reference section is in properly formatted.	Writing is clear and relevant, with no grammatical and/or spelling errors – polished and professional. Exceptionally good conclusion that reflected the work. Reference section is properly formatted.	
						Total (60%)	

	PRESENTATION COMPONENT (40 %)						
CLO4 Examine proficiency in apply scientific method to applications of AI.	**MARKING CRITERIA**	**SCALE**					**YOUR MARKS/COMMENTS**
		Fail (0-49)	**3rdClass (50-59)**	**2nd LowerClass (60-69)**	**2nd UpperClass (70-79)**	**1stClass (80-100)**	
	1.CONFIDENCE (Ability to interact with audience and answer the question) **(10%)**	Lack of audience awareness; mismatch for the intended reader Writer lacks a sense of involvement; flat; lifeless Unable to answer the question from audience.	Limited sense of audience; doesn't acknowledge needs of reader Little commitment to topic Answer does not address the question adequately	Shows some awareness of audience Presentation is committed but inconsistent Answer to the question is acceptable.	Presenter recognizes audience; Committed to topic; Appropriate point of view; shows some originality in answering audience question.	Presenter quickly engages audience with strong interaction. Strongly committed to topic which comes to life Answer provided is justifiable and convincing.	
	2. PRESENTATION SLIDES (The usage of vocabulary) **(5%)**	Limited vocabulary Misused words interfere with meaning Inadequate, imprecise terms or expressions; fails to communicate message	Colorless, generic vocabulary Expressions may impair understanding; monotonous repetition Inappropriate; unimaginative terms or slang detract from message	Appropriate but ordinary vocabulary Functional expressions; may have some fine moments Terms convey message but passive verbs or clichéd expressions may interfere	Accurate, precise vocabulary Purposeful, clear meaning but rarely experiments with language Words convey the intended message	Powerful, varied, broad range of vocabulary Thoughtfully placed terms or expressions Words effectively communicate message in an interesting, precise, and natural way	
	3. EXPLANATION & MESSAGE DELIVERING (Explain, using the vocabulary of problem solving, how you came to be able to find, interpret and use the hidden code) **(15%)**	Explanation is predominantly a description of methods adopted and contains no reference to the concepts and principles used in the module. No examples of concept application are provided. The code snippet is not understandable.	Explanation contains only sparse reference to the concepts and principles used in the module. Examples of concept application are applied correctly but the application of context may be limited. The code snippet is largely understandable, but it is not clear how the hidden code is incorporated and used inside the code snippet.	Explanation uses less than a third of the key concepts and principles and is supported by appropriate examples correctly applied. The code snippet is understandable, but it is not clear how the hidden code is incorporated and used inside the code snippet.	Explanation uses between a third and two thirds of the key concepts and principles. Numerous correct supporting examples are appropriately applied. The code snippet is understandable and uses the hidden code correctly.	Explanation is a lucid account that deploys a rich vocabulary to illuminate the problem solving process succinctly and it refers to the majority of concepts and principles that have been delivered in the module. Supporting examples are well chosen. The code snippet is clear and concise and uses the hidden code correctly.	
	4. DEMONSTRATION (Demonstrate and provide a commentary on your contribution to the work) **(10%)**	Program is not working No evaluation of personal contribution of works	Program partially working. The evaluation of personal contribution is largely a 'whitewash' that suggests little reflection on work.	Majority of program works properly. The evaluation of personal contribution presents both good and bad points. Some content hints at future group engagement.	Program works well The evaluation of your personal contribution is well considered and presents both good and bad points. Some content hints at future group engagement.	Program works well. The evaluation of personal contribution is well considered and presents both good and bad points. Further it identifies possible strategies for future group engagement that reflect your identified strengths and weaknesses.	
	Total (40%)						
	Total (100%)						