

COSC310 SOFTWARE ENGINEERING

Assignment 3 Milestone Report

Group 17

Anamica Sunderaswara [80600851]

Andrew Dunn [73838070]

Qingyan Hu [64356884]

Rahman Ganiyu [23769748]

Segundo Parra Jurado [44934693]

Ryan Soon Han Lu [82078023]

TABLE OF CONTENTS

Contents	Pages
1.0 Updated README documentation 1.1 Project Description 1.2 SDLC Overview 1.3 Assignment 3 Milestone Updates	2
2.0 List of Implemented Features & Description 2.1 GUI 2.2 Part-Of-Speech Tagging 2.3 Named Entity Recognition 2.4 Perform Interactions with other agents	3 - 4
3.0 Level 0 Data Flow Diagram	4
4.0 Level 1 Data Flow Diagram	5
5.0 GitHub Commits & Development Graphs	6
6.0 Chatbot System Demonstration 6.1 Successful Conversation 6.2 Program Limitations 6.3 Unsuccessful Conversation 1 6.4 Unsuccessful Conversation 2	7 - 8
7.0 Five Design Features Sharable to others as an API 7.1 Feature 1 7.2 Feature 2 7.3 Feature 3 7.4 Feature 4 7.5 Feature 5	8

[Group 17 Repository URL](#)

1.0 Updated README Documentation

1.1 Project Description

The objective of this programming project is to practice team collaboration and management principles within a software engineering environment – through the creation of an interactive program. The program's aim is to create a dynamic conversational agent that is able to interact and respond to a user's input. Within our chosen setup, the agent will imitate an old friend and enable the user to participate in conversations centering around topics such as favorite food, hobbies, and books. The project will be completed using project management tools such as JIRA and employ Java as the object-oriented programming language.

1.2 SDLC Overview

The chosen Software Development Life Cycle (SDLC) model for the execution of this project is the incremental/agile model. Despite the limitations of the agile model, our team found it to be the best suited option for our requirements and team setup for the following reasons: First and foremost, our team decided to follow an iterative approach. With this, planning is incremental, and it is easier to change the process as required. Additionally, the amount of analysis and documentation that must be redone is much less than is required for the waterfall model. Lastly, we found that agile methods aligned with our team project management outlooks. Overall, making it the best choice for our team.

1.3 Assignment 3 Milestone Updates

The third milestone of this project focused on improving our already formed code, and adding features that would enhance the functionality of our Chatbot.

In terms of the project management structures we used to guide us, we decided to follow the same SDLC work cycle, with minor changes. This is because we found the second iteration of the project to put forward similar requirements as the first. Therefore, since our life cycle worked effectively for us last time, it was in our best interest to use it again. Additionally, as this was now our third time working together, it was easy to maintain and perfect the agile process we had been implementing until now.

Regarding the code, the features we decided to apply included: creating a graphical user interface, importing and integrating POS Tagging & Named Entity Recognition Toolkits, as well as configuring the Chatbot to be able to interact with other agents. The reason behind why we picked these aspects was because we had the goal to improve our Chatbot to the point where it could really emulate the feeling of talking to a friend; and found this was our way to do so.

Each feature applied is described in further detail in the next section.

2.0 List of Implemented Features & Description

- Graphical User Interface
- Part-Of-Speech Tagging
- Named Entity Recognition
- Perform Interactions with Other Agents

2.1 GUI

The first newly implemented feature assisting the improvement of our chatbot was the addition of a Graphical User Interface. The added GUI adds another level to our program by giving it an interactable channel for the user to use, rather than having to simply type within a running code box. In our GUI, first the user must click the “Chat with me!” button to start the program. Then, the user types their message in the bottom square, then clicks the button labelled “Enter” and receives the Chatbot’s answer as well as their original message in the top box. This top box keeps a log of all the messages sent and received, so the user can view the entire conversation as long as the program is running.

2.2 Part-Of-Speech Tagging

The integration of the POS Tagging Toolkit enables the Chatbot to perform keyword searching in a seamless manner. The way our program implements the toolkit is by first tokenizing any String the user inputs, and then administering the POS Tagging software to categorize the tokens into nouns, verbs, adjectives, or adverbs, etc. Next, with each word classified appropriately, they are inputted into corresponding array lists. With these array lists, the Chatbot is able to search through the appropriate array lists and see if any of the elements inside match the user input when generating a response. Overall, this feature enables the flow of conversation with the Chatbot to run smoother and more naturally. Therefore, effectively minimizing the gap between the user feeling like they are talking to programmed bot and a fellow friend.

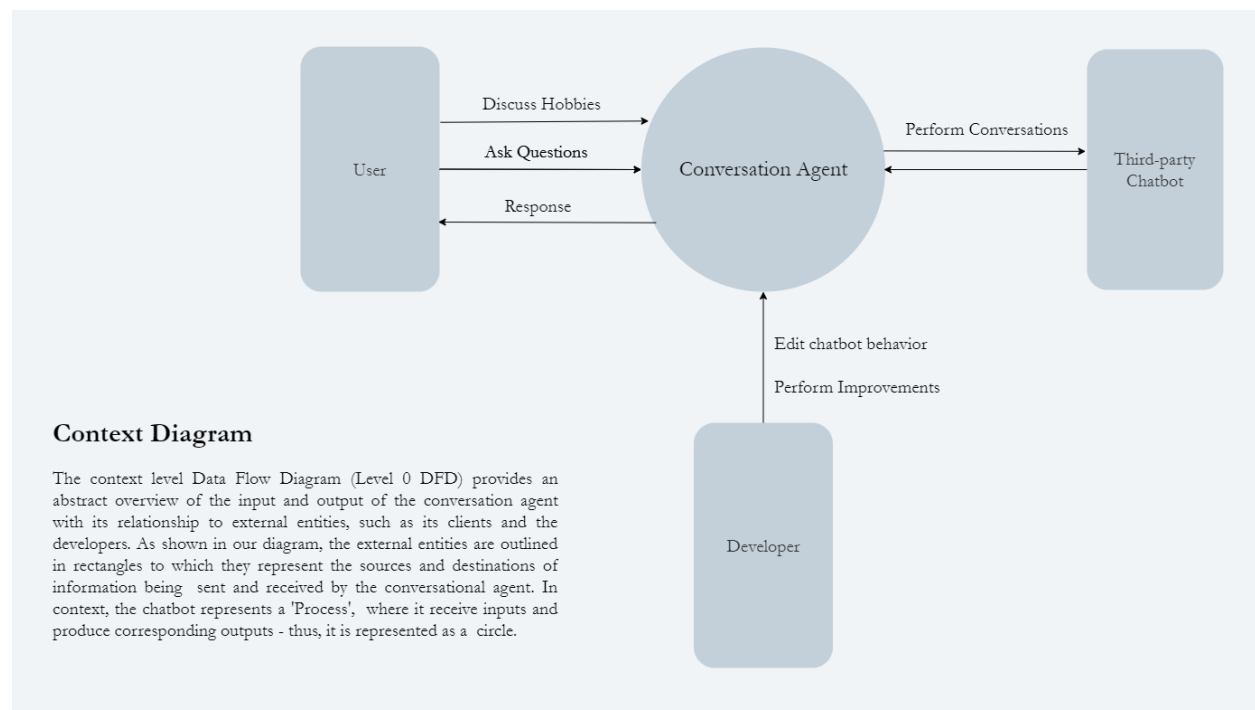
2.3 Named Entity Recognition

Another feature implemented within the update of our program is introduction of the Named Entity Recognition Toolkit. The integration of this toolkit helps the Chatbot recognize the names of people and locations. For example, if a user says, “My name is Mike” or “I’m Mike”, the Chatbot will recognize his name, and reply with “Hello, Mike! Nice to meet you!”. Additionally, the program can now also make informed decisions when responding. For example, if the user’s message includes a location, for instance, “I live in Ontario”, the bot may respond with something along the lines of “Oh

we live in the same country!’. Whereas, if the user enters a location name outside of Canada, Chatbot will generate something similar to “I haven’t been there before, but I hope I can go there someday”. This creates a more natural and friendly feeling to the bot and helps negate the automated response feeling that traditionally accompanies similar programs.

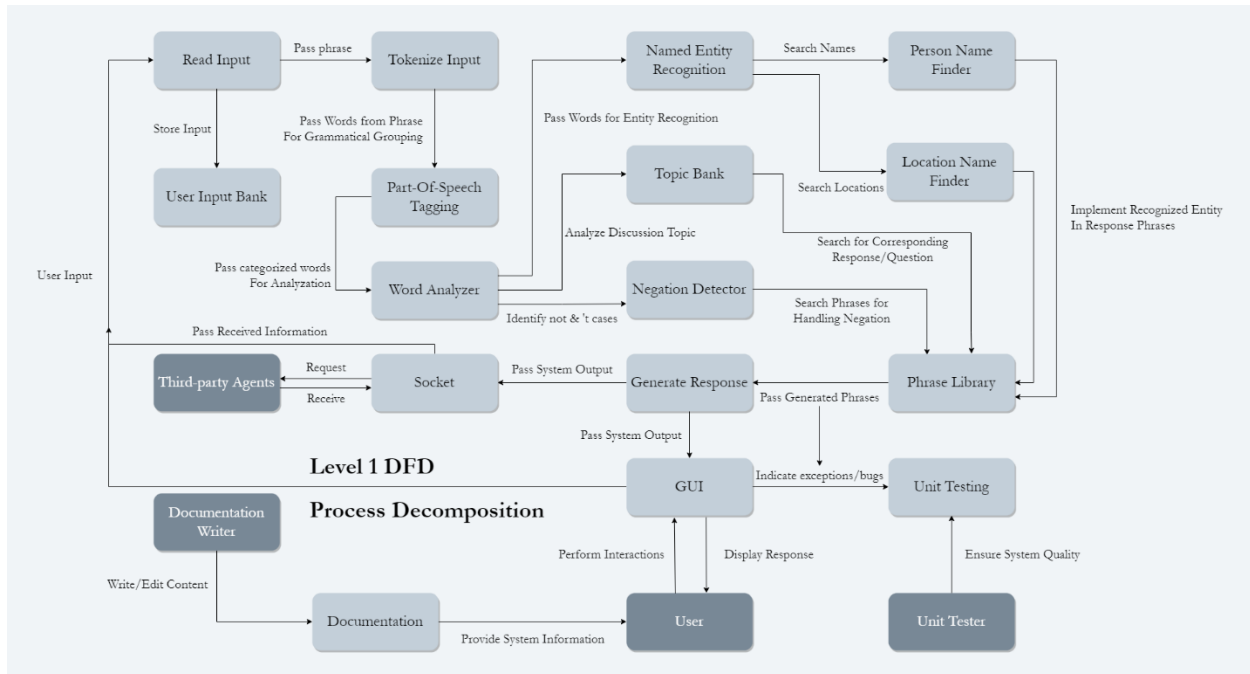
2.4 Perform Interactions with Other Agents

The last implemented feature is the ability for our Chatbot to interact with other Chatbots. The integration of this feature enables any other automated Chatbot to replace the role of the user and communicate with our Chatbot to have real-time conversations. Through sockets, our program is able to connect to another machine and conduct a conversation with another bot. In our demonstration, we acted as the client side, connecting to a server hosted on a different machine.



The context level Data Flow Diagram (Level 0 DFD) provides an abstract overview of the input and output of the conversation agent with its relationship to external entities, such as its clients and the developers. As shown in our diagram, the external entities are outlined in rectangles to which they represent the sources and destinations of information being sent and received by the conversational agent. In context, the chatbot represents a 'Process', where it receives inputs and produce corresponding outputs - thus, it is represented as a circle.

4.0 Level 1 Data Flow Diagram



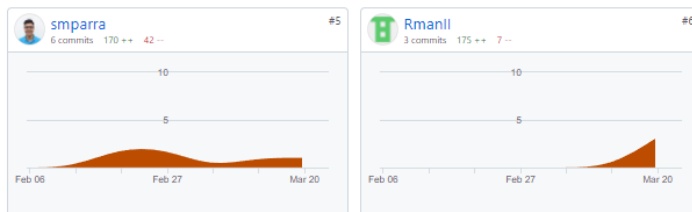
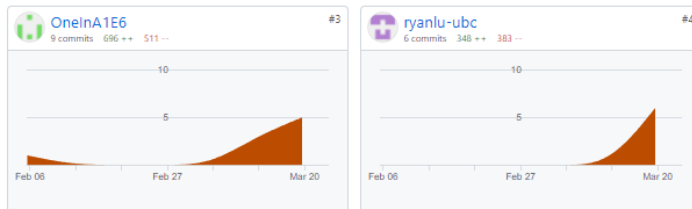
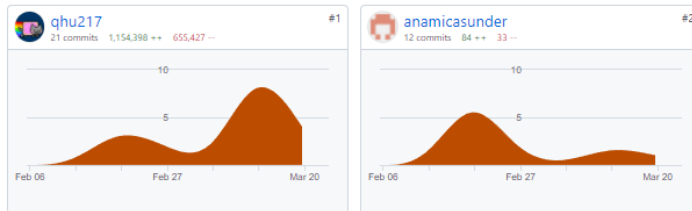
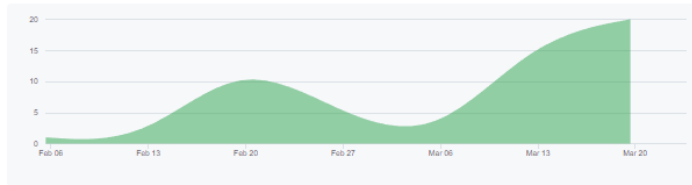
The Level 1 Data Flow Diagram decomposes the context level diagram into multiple in-depth processes. Process such as the “Conversation Agent” in Level 0 DFD is broken down into components (subprocesses) that details the major procedures that the system runs through. In our Process Decomposition diagram, darker colored rectangles represent external entities that interact with the system. System user for example, will input questions and responses using the graphical user interface and will also receive displayed system output through the GUI. The light grey rectangles denote subprocesses that makes up the conversational agent. Each process is interconnected with another process or external entities that passes and receives information.

5.0 GitHub Commits/Development Graphs

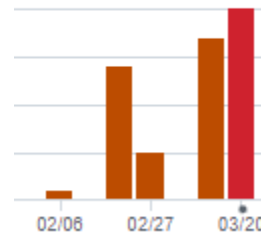
Feb 6, 2022 – Mar 24, 2022

Contributions: Commits ▾

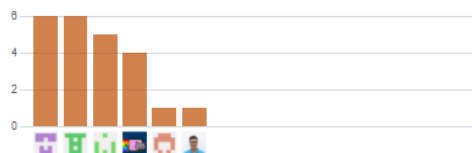
Contributions to main, excluding merge commits and bot accounts



20 commits the week of Mar 20



Excluding merges, **6 authors** have pushed **20 commits** to main and **23 commits** to all branches. On main, **4,470 files** have changed and there have been **12,833 additions** and **633,334 deletions**.



main ▾ 5 branches 0 tags

Go to file

Add file ▾

Code ▾

Switch branches/tags

Find or create a branch...

Branches

Tags

✓ main

default

GUI

POS-tagging

Socket

Updated

[View all branches](#)

README.md

Socket Updates

40 minutes ago

Fixed a minor bug in JUnit testing

8 hours ago

minor changes

14 hours ago

Video recordings +minor edits

5 minutes ago

Video recordings +minor edits

5 minutes ago

updated documentation

7 hours ago

gui working

13 hours ago

Update README.md

7 days ago

6.0 Chatbot System Demonstration

6.1 Successful Conversation

```
You: hi
Bot: Hey!
You: how are you
Bot: I'm good! How are you feeling today?
You: not so good
Bot: I'm sorry to hear that. I hope chatting with me will make you feel better.
You: thank you
Bot: You're very welcome, glad that I can help!
You: what's your name
Bot: My name is Adrian. What's your name?
You: ryan
Bot: Hello, Ryan! Pleased to meet you!
You: where do u live
Bot: I live in British Columbia, Canada. Where do you live?
You: I live in Alberta
Bot: Oh! We live in the same country!
You: how old are you
Bot: I'm less than one year old.
You: what's the day today
Bot: Today is WEDNESDAY.
You: what is the date today
Bot: Today is 2022-03-23.
You: how is the weather like
Bot: The weather is nice, not too cold, not too warm, can't complain.
You: what's your fav movie
Bot: I really like 'The Godfather'. Which movie do you like the most?
You: i love batman
Bot: I haven't watched this movie before, but I believe it's a good one!
You: It is a good one! which book do you like the most
Bot: I really like 'Pride and Prejudice'. Which book do you like the most?
You: my fav book is introduction to java programming
Bot: I have read it before and I really like it!
You: really? that's cool! which is your fav food?
Bot: I like ice cream the most. Which food do you like the most?
You: I like burgers
Bot: Really? I like it as well!
You: Great! What about sports?
Bot: I really like skiing. Which sport do you like the most?
You: My favorite sport is badminton
Bot: I have tried it before and I also think it's a good one!
You: Very nice. C u later
Bot: It's very nice to chat with you. Looking forward to talking with you next time!
```

6.2 Program Limitations

- The chatbot only understands certain inputs making the conversational abilities limited
- The chatbot relies on the user input to determine what it should say as a response, it seldom initiates topics
- The chatbot does not have a variety of available responses, making conversation repetitive
- The user is restricted to talking about what the chatbot understands rather than being able to talk freely as if the chatbot were truly their friend
- The user is restricted to talking about what the chatbot understands rather than being able to talk freely as if the chatbot were truly their friend
- Chatbot relies on keywords from the user to decide response. Thus, sometimes it does not understand single word responses
- The chatbot is unable to process inputs with double negatives (e.g., not bad)
- Repeated usage of keywords in user inputs will lead the chatbot into an endless loop

6.3 Unsuccessful Conversation 1

```
You: Where do you live
Bot: I live in British Columbia, Canada. Where do you live?
You: I also live in British Columbia
Bot: I haven't been to British before, but I hope I could go there one day!
```

In this example, the chatbot is unable to recognize the location 'British Columbia' as a whole. Instead, it highlights 'British' and outputs it as the location.

6.4 Unsuccessful Conversation 2

```
You: Which movie do you like the most
Bot: I really like 'The Godfather'. Which movie do you like the most?
You: I don't like watching movies
Bot: I haven't watched this movie before, but I believe it's a good one!
```

This unsuccessful conversation shows that the chatbot assumes the user has inputted a movie name. When the user enters irrelevant information, such as 'I don't like watching movies', the chatbot is not capable of handling such situation.

7.0 Five Design Features Sharable to others as an API

Feature 1 – Graphical User Interface

The GUI feature for our conversational agent can be deployed by others with a text-based conversational interface is required.

Feature 2 – Named Entity Recognition

The named entity recognition feature allows the conversational agent to recognize names and locations. It can be used by other developers to avoid hardcoding.

Feature 3 – Part-Of-Speech Tagging

The part-of-speech tagging feature allow other developers to better analyze user inputs and thus, helps the chatbot to generate smarter responses.

Feature 4 – Ability to identify and handle text-based emotions

Our chatbot can identify user emotions based on the user's inputs and is able to handle corresponding situations. In the case where the user display signs of despairing emotions, the chatbot will generate phrases that aims to cheer the user up.

Feature 5 – Ability to talk about multiple topics

Our chatbot can handle multiple discussion topics and can be implemented by others as topic banks and phrase libraries.

COSC 310 - Project Plan

Repository URL: <https://github.com/Group-17-COSC-310>

Project Description:

The objective of this programming project is to practice team collaboration and management principles within a software engineering environment – through the creation of an interactive program. The program's aim is to create a dynamic conversational agent that is able to interact and respond to a user's input. Within our chosen setup, the agent will imitate an old friend and enable the user to participate in conversations centering around topics such as favourite food, hobbies, and books. The project will be completed using project management tools such as JIRA and employ Java as the object-oriented programming language.

SDLC Overview:

The chosen Software Development Life Cycle (SDLC) model for the execution of this project is the incremental/agile model. Despite the limitations of the agile model, our team found it to be the best suited option for our requirements and team setup for the following reasons:

First and foremost, our team decided to follow an iterative approach. With this, planning is incremental and it is easier to change the process as required. Additionally the amount of analysis and documentation that has to be redone is much less than is required for the waterfall model. Lastly, we found that agile methods aligned with our team project management outlooks. Overall, making it the best choice for our team.

SDLC Breakdown:

- **Requirements Definition and Project Planning:**
 - **Github Creation:** create a github account and team repository for the project
 - **Criteria Itemization:** itemize project demands and listed criteria
 - **SLDC:** pick an appropriate SLDC that fits the team dynamic and project requirements
 - **Workload Division (WBS part 1):** divide the workload within the project team, each team member focusing on an area well-suited with their personal strengths
 - **Workload Equality:** within the workload division, ensure that each team member has an equal amount of work
 - **Sector Managers:** plan who will be in charge of the documentation, the project management responsibilities, the coding workload, and the final presentation
 - **Timeline Plan (WBS part 2):** according to the workload division, plan the project timeline and the estimated date of which each stage of the software development life cycle should be completed.
 - **Gantt Chart:** creation of a Gantt chart showing start and end dates for each task, and showing dependencies across task
- **Software Design:**
 - **Program Designing:** make a design of what the program/chatbot should do, this includes:
 - a written design on the possible auto-generated responses
 - a document detailing the conversations the chatbot will gear the users towards
 - a scheme of what the overall functionality of the chatbot should be

- **Program Limitations:** make a list of the limitations regarding the program
- **Program Prototype:** design a prototype including the basic functionality of the final program
- **Prototype Feedback:** ensure every team member is on board with moving forward according to the proposed prototype
- **Code Implementation and Unit Testing**
 - **Code the Program:** perform the coding of the program, following the previously agreed upon design
 - **Testing:** test the implemented code and make sure it meets all requirements
 - **Individual Testing:** ensure all team members are happy with the finalized code
 - **Implement Changes:** if the testing stage reveals that there are faults within the code or fails to meet the requirements, implement the needed changes
- **Project Completion and Final Steps:**
 - **Video Creation:** the creation of the video showcasing the implementation of the program
 - **Video Requirements:** the video should include a live demo of our program, an overview of the team responsibilities, PM graphs of our project repository and the project management tool we used.
 - **Video Approval:** ensure all team members are satisfied with the video
 - **Peer Evaluation:** submit a peer evaluation for all team members
 - **Project Submission:** submit the finalized project on Canvas

COSC310 SOFTWARE ENGINEERING

Assignment 3 - Automated Unit Testing Report

Group 17

Anamica Sunderaswara [80600851]

Andrew Dunn [73838070]

Qingyan Hu [64356884]

Rahman Ganiyu [23769748]

Segundo Parra Jurado [44934693]

Ryan Soon Han Lu [82078023]

Tasks

Design test cases using Junit

Have three test cases for major functions

Design a document for summarizing the test cases

Objectives

The objective for this part of the assignment is to test the output of the chatbot by examining the responses generated from different major functions. There are 26 major methods available for testing in our chatbot and we have chosen three key features to test.

Methodology

Our group implemented the dialogue agent using the JAVA programming language. In correspondence, we have chosen Junit 5 Jupiter unit testing framework to deploy and test the features of our chatbot.

Subjects Under Test

We have chosen three major features of our conversational agent to perform the unit testing. Each of the three methods will be tested with one test case. The subjects under test are:

1. Discuss Feeling feature (src/Bot.class/discussFeeling)
2. Discuss Location feature (src/Bot.class/discussLocation)
3. Ability to generate responses relevant to hobbies (src/Bot.class/hobbyResponse)

Test Cases

testDiscussFeeling()

This test method checks the display output of discussFeeling method in the Bot.java class. Ideal responses from this method are in which the method will be able to detect the user's current emotions based on the keywords of his/her input. For our test case, we have chosen a situation where the user inputs "Not so good". The chatbot will be able to respond with phrases like, "I'm sorry to hear that. I hope chatting with me will make you feel better.".

Test case: "Not so good"

Output: "I'm sorry to hear that. I hope chatting with me will make you feel better."

Chatbot output status: Success!

testDiscussLocation()

This test method checks the display output of discussLocation method in the Bot.java class. Ideal responses from this method are in which the method will be able to detect the user's location based on the keywords of his/her input. For our test case, we have chosen a situation where the user inputs "I live in Ontario". The chatbot will be able to recognize that Ontario is a location based on locationNER function and respond with a phrase like, "Oh! We live in the same country!".

Test case: "I live in Ontario"

Output: "Oh! We live in the same country!".

Chatbot output status: Success!

testHobbyResponse()

This test method checks the display output of hobbyResponse method in the Bot.java class. Ideal responses from this method are in which the method will be able to detect the user's hobby and the topic his/her would like to discuss based on the keywords of his/her input. For our test case, we have chosen a situation where the user inputs "Which movie do you like the most?". The chatbot will be able to recognize that the user is asking a question about the chatbot's favorite movie and generate a reply such as "My favorite movie is 'The Godfather'. Which movie do you like the most?"

Test case: "Which movie do you like the most?"

1 of 3 Outputs:

Chatbot: "My favorite movie is 'The Godfather'. Which movie do you like the most?"

Chatbot: "I like 'The Godfather'. Which movie do you like the most?"

Chatbot: "I really like 'The Godfather'. Which movie do you like the most?"

Chatbot output status: Success!

Summary

In this unit testing session, we have examined and tested the display outputs of three major functions included in our dialogue agent. These functions are: discussFeeling, discussLocation and hobbyResponse. Each of the subject were assigned with one test case in which it has to succeed. In our case, these three methods in our chatbot source file were able to produce the desired output and successfully passed all sections of the unit testing.

Program Limitations

Listed below are a few limitations of our program:

- The chatbot only understands certain inputs making the conversational abilities limited
- The program does not currently have a working GUI
- The chatbot relies on the user input to determine what it should say as a response, it seldom initiates topics
- The chatbot does not have a variety of available responses, making conversation repetitive
- The user is restricted to talking about what the chatbot understands rather than being able to talk freely as if the chatbot were truly their friend
- The user is restricted to talking about what the chatbot understands rather than being able to talk freely as if the chatbot were truly their friend
- Chatbot relies on keywords from the user to decide response. Thus, sometimes it does not understand single word responses
- The chatbot is unable to process inputs with double negatives (e.g. not bad)
- Repeated usage of keywords in user inputs will lead the chatbot into an endless loop

Opening lines

can use a function to randomly generate one of the following opening prompts, so that it changes on each run; maintaining a dynamic interface

- "Hello!"
- "Greetings!"
- "Hey!"
- Extra fancy
 - "Good morning/afternoon/evening!" (using system time)
- "Hello! My name is ... What's your name?"
chatbot response to user input: "that is such a beautiful name!"

Chatbot questions

- "How are you today?"
- "What's your name?"
- "What do you want to talk about? (Enter something like 'books', 'hobbies', 'foods', 'movies', etc.)"
chatbot transition: "Great! I wanted to talk about that too, so..(insert next question)"
- "What is your favourite book/movie/food?"
chatbot response to user input: "I've heard of that movie! Unfortunately, I've never watched it myself!"
- "What genre of books/movies do you like?"
chatbot response to user input: "I love those genres too!"
- "What are your hobbies?"
chatbot response to user input: "Those all sound like so much fun!"

Chatbot responses (chatbot answering questions posed by user)

- Prompt keywords: "movie", "?" (e.g. what is your favourite movie?)
 - "My favourite movie is ____!"
 - "I loved watching ____!"
- Prompt keywords: "book", "?" (e.g. what is your favourite book?)
 - "My favourite book is ____!"
 - "I loved reading ____!"
- Prompt keywords: "What's your favourite food", "?"
 - "I love a lot of foods! It's really hard for me to pick one. What's your favourite food?"
- Prompt keywords: "How are you", "How are you doing", "?"
 - "I'm doing great!"
 - "I'm doing great, how about you?"
- Prompt keywords: "hobbies", "hobby", "like/love doing & free time", "?" (e.g. what are you hobbies/ what do you like doing in your free time?)
 - "I love ____! What about you?"
- Prompt keywords: "What day is it today", "?"
 - "It's ..." (using system time)

- Prompt keywords: "Where do you live", "Where are you located", "?"
 - "I live in British Columbia, Canada. What about you?"
- Prompt keywords: "Have you ever been to ...", "?"
 - "No, I haven't. But I do hope that I can go there one day."
- Prompt keywords: "How old are you?"
 - "My birthday was on ... (date we finished the chatbot). How old are you?"
- Prompt keywords: "What is the weather like where you are?"
 - "the weather is nice, not too cold, not too warm, can't complain"

Other responses (don't know what keywords to look for yet for these)

- Sample input: "I'm doing great!"
 - Response: "That's nice"
- Sample input: "I don't feel so good", "Not so good" or "I'm sad/mad/angry/depressed" (negative moods)
 - Response: "I'm so sorry to hear that, is there anything you want to talk about that can make you feel better?"
 - if the user says no: "Okay, I understand. Always remember you can come to me if you need to talk about anything in the future"
 - if the user says yes: "Okay, what did you want to talk about?"
- Sample input: "Have you ever watched (movie name)?"
 - Response:
 - If (movie_name != chatbot_favourite_movie_name):


```
Print("No I haven't watched it before. Do you like it?")
If (user_input == "yes"):
    Print("I'm glad that you like it!")
Elif (user_input == "no" || "so so"):
    Print("-What's your favorite movie?")
```
 - If (movie_name == chatbot_favourite_movie_name):


```
Print("Yes I've watched it before! I really like it, do you like it?")
If (user_input == "yes"):
    Print("I'm glad that you like it!")
Elif (user_input == "no" || "so so"):
    Print("What's your favorite movie?")
```
- Sample input: "Have you ever eaten/tried (food name)?"
 - Response:
- Sample input: "Have you ever read (book name)?"
 - Response:
- Sample input: "" (*empty string*)
 - Response: "What do you want to talk about? (Enter something like 'books', 'hobbies', 'foods', 'movies', etc.)"

- Sample input: "hdfakldfjqwjgq;n" (*gibberish or scenarios where chatbot doesn't understand user inputs*)
 - Response: "I'm sorry that I don't understand your message. What do you want to talk about? (Enter something like 'books', 'hobbies', 'foods', 'movies', etc.)"

Restrictions

- All questions to chatbot must end with "?"
- Does chatbot need to prompt conversation or only give replies?