

COSC310 SOFTWARE ENGINEERING  
Individual Project Report

Group 17

---

Anamica Sunderaswara [80600851]

## TABLE OF CONTENTS

Contents	Pages
1.0 Updated README documentation 1.1 Project Description 1.2 SDLC Overview 1.3 Assignment 3 Milestone Updates 1.4 Individual Project Milestone Updates	2 - 3
2.0 List of Implemented Features & Description from Assignment 3 2.1 GUI 2.2 Part-Of-Speech Tagging 2.3 Named Entity Recognition 2.4 Perform Interactions with other agents	4 - 5
3.0 List of Implemented APIs for Individual Component 3.1 Wikipedia API 3.2 Google Translate API	6
4.0 Level 0 Data Flow Diagram	7
5.0 Level 1 Data Flow Diagram	8
6.0 Chatbot System Demonstration 6.1 Successful Conversation 6.2 Program Limitations 6.3 Unsuccessful Conversation 1 6.4 Unsuccessful Conversation 2	9-10
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	11

## [Repository Link](#)

### 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 (Second Iteration of Project) Milestone Updates

The second phase 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 following sections.

### 1.3 Individual Project (Final Iteration) Milestone Updates

The third phase of this project involved individually modifying the Chatbot to now include two APIs. The purpose of integrating APIs was to enable the Chatbot to process real-world data and incorporate it into conversation. Therefore, furthering the dynamic range of conversation as well as reducing the feel of an automated script.

The APIs chosen to be implemented were the Google Translate API as well as the Wikipedia API.

The Google Translate API enables the user to enter any String, and the Chatbot will return the translation. The Wikipedia API enables the Chatbot to return any Wikipedia Description (within this implementation, the descriptions are focused on user specified locations).

Both API's functionalities will be further discussed within the documentation file as well as demonstrated within this phase's video.

Additionally, although this project was completed individually, the SDLC cycle followed mainly remained the same.

## 2.0 List of Implemented Features & Description from Assignment 3

- 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.

### 3.0 List of Implemented APIs for the Individual Component

Featured below is a brief description regarding the APIs implemented for the Individual Component of this Project.

#### 3.1 Wikipedia API

The first API chosen to be integrated within the Chatbot's features was the Wikipedia API. The desire was to add an API that did not take away from the conversation flow already implemented or unnecessarily disrupt/complicate the foundation of the code.

As a result, the way the Chatbot uses the Wikipedia API involves updates regarding when the User mentions a location. Previously, when the User indicated where they were from, if it was outside Canada, the Chatbot would return an automated response of "I haven't been to [insert User Location] before, but I hope I could go there one day!". This response is very stale and non-dynamic. It does not reference any details about the user location, nor does it promote further conversation.

However, now with the implementation of the API, if the user chooses to disclose where they're from, the Chatbot extracts knowledge about that location from Wikipedia and uses it to emphasize their appreciation for that city/country.

For example, if the user says they are from Osaka, Japan the Chatbot now returns:

"I haven't been to Osaka, but I've heard tons about it! Some things I know are that:

Osaka is a designated city in the Kansai region of Honshu in Japan. It is the capital and the most populous city in Osaka Prefecture, and the third most populous city in Japan, following Tokyo and Yokohama. With a population of 2.7 million in the 2020 census, it is also the largest component of the Keihanshin Metropolitan Area, the second-largest metropolitan area in Japan and the 10th largest urban area in the world with more than 19 million inhabitants."

Through this, the user is able to feel heard, as if they are truly interacting with the Chatbot, as it is showing it understands what they are saying and responding in a corresponding fashion, rather than following an automated script.

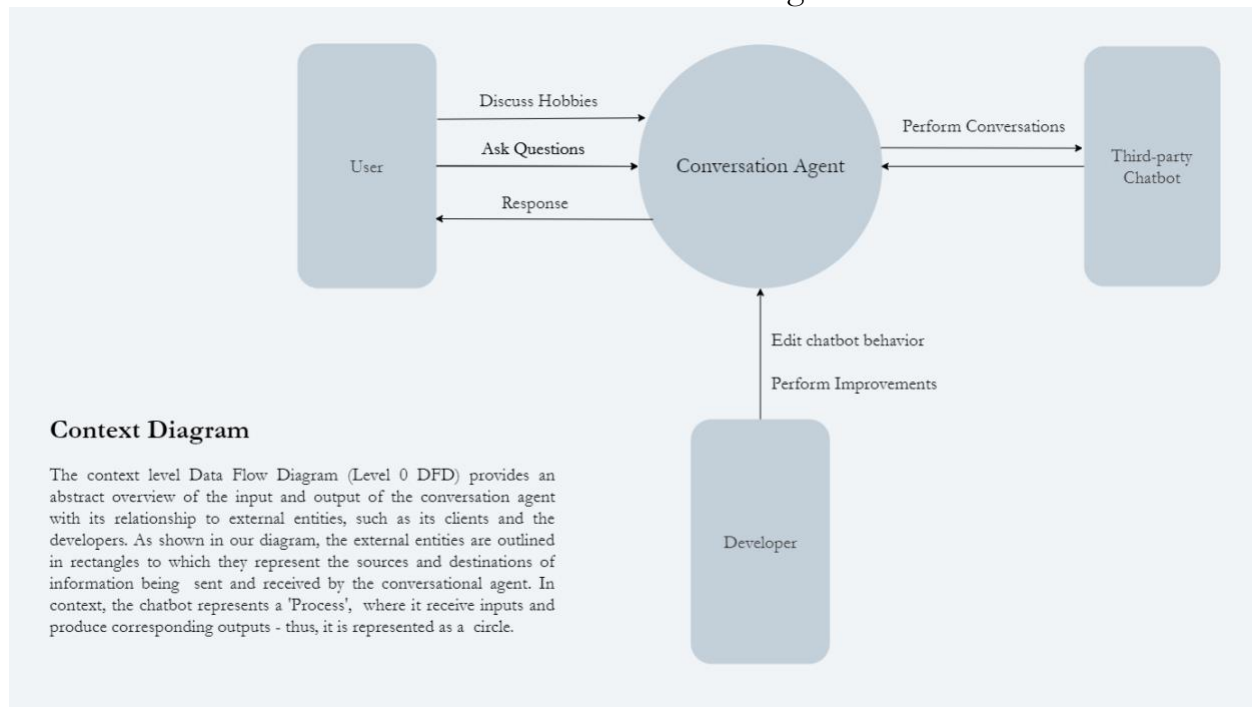
#### 3.2 Google Translate API

The second API chosen to be integrated within the Chatbot's features was the Google Translate API. The desire was to use an API to encourage the user to want to talk to the Chatbot again and be inclined to run the program numerous times.

As a result, the way the Chatbot uses the Google Translate API is during the closing message of the program. When the user decides to end the chat, before the bot allows the user to leave, it asks them if they want to learn something in the Bot's native language of French. If so, the user is prompted to enter any word or sentence they would like translated, and the bot does it instantaneously.

This is a powerful addition, as it teaches the User something new every time they run the program. Which not only provides incentive to return to the program, but also mirrors a real interaction with a friend, where one normally leaves feeling like they have gained some sort of knowledge or wisdom by the end of it.

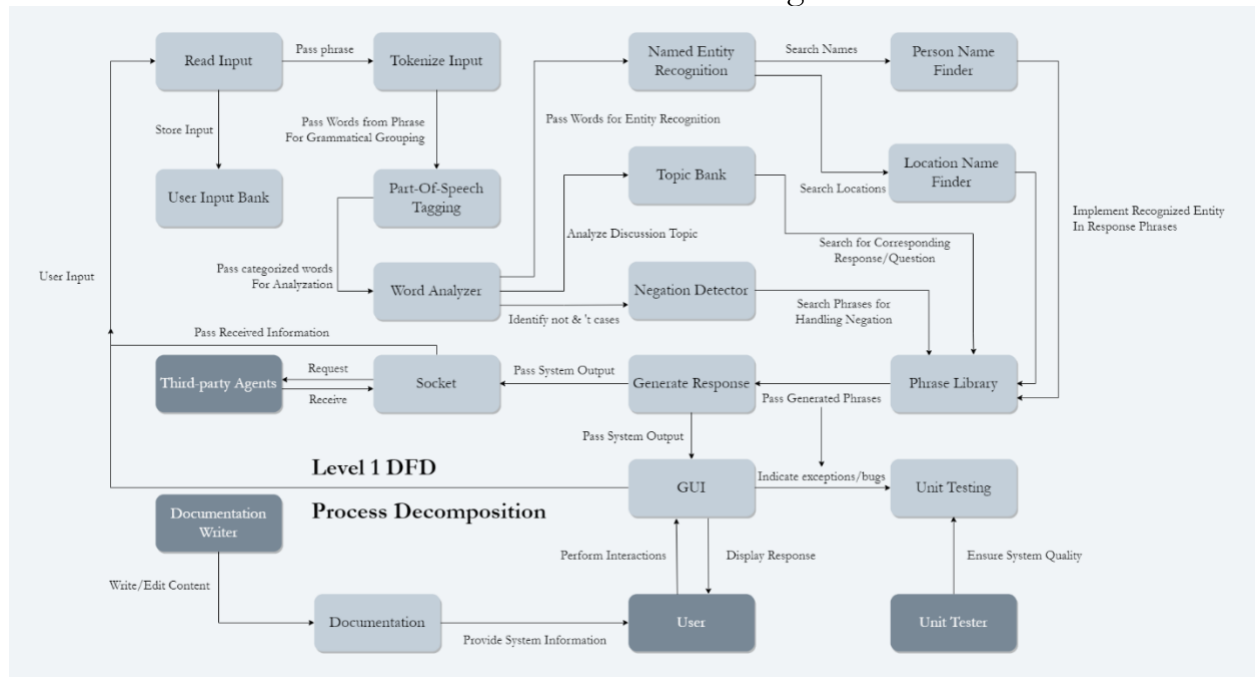
### 3.0 Level 0 Data Flow Diagram



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-dept 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 inter-connected with another process or external entities that passes and receives information.

## 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.