COSC 310 – 001

Software Engineering

2021 Winter Term 2

Project Plan Document

Thomas Buchholz

Ryan Grant

Vinu Ihalagamage

Tanner Wright

Dima Zhuravel

Table of Contents

1.0 Project Description

The project was created for the UBC Okanagan third-year level course.

Psych Bot's goal is to give the user psychological advice*. This bot serves as an interactive conversational agent that takes the user's input (a sentence) and outputs an appropriate response. As this assignment does not require Machine Learning implementation, the chatbot may provide a reply that may not relate to the user's prompt.

* For legal reasons, neither the bot nor developers are certified to provide medical help.

Github Repository:

https://github.com/KentonMewling/Psych-Bot

2.0 SDLC

2.1 Rationale

After the first meeting with all team members, our group decided to use the **Agile Development Methodology** to develop the chatbot. The decision was based on analyzing the app's functionality – creating a chatbot. Since our team consists of people with vast background experience in a selected programming language, and the ability to work on each functionality, the incremental development will allow us to complete this project with best practices. Furthermore, completing every step in the SDLC by necessity will enable us to be more flexible than a plan-based approach (e.g., Waterfall Methodology). In other words, creating functionality, like Natural Language Processing, will be redundant as Python and its package management system (PIP) will provide all necessary functionality to build the app in time.

2.2 Task Breakdown

2.2.1 Requirements Analysis

1. Product: Interactive chatbot capable of responding to the user's input with appropriate responses.
    a. The chatbot should be developed using an Object-Oriented Programming Language.
2. Chatbot's Task: The chatbot should play the role of a psychologist capable of answering the question in a field-related manner.
3. User's Task: The user should have a conversation with the chatbot by entering questions related to the bot's assigned task.
4. End Goal: The program must hold a full and meaningful dialogue with the user for a minimum of 30 turns of conversation.

2.2.2 System Design

1. Programming Language: Python and its libraries will be used to develop the chatbot.
    a. Existing Python libraries will be used to process and analyze user responses.
2. The Dialogue: Possible responses to the user's questions will be stored in a text file.
3. Output and Prompts: The first version of the chatbot will be terminal-based, which means that the user's and chatbot input/output will be displayed in the command-line GUI.

2.2.3 Implementation

1. Create a JSON-based file that contains conditions and patterns that the chatbot should use for user responses.
2. Implement a class that will parse the JSON-based file.
3. Implement a class that will handle the conversation with the user based on their input and patterns/conditions stored in the JSON file.
    a. If the input is unknown to the chatbot, print out a random response that should have a meaningful answer.
4. Create a file containing the purpose of the chatbot and explain how to run it.
5. Make a well-documented code that will allow other developers to understand and read the code.

2.2.4 Testing

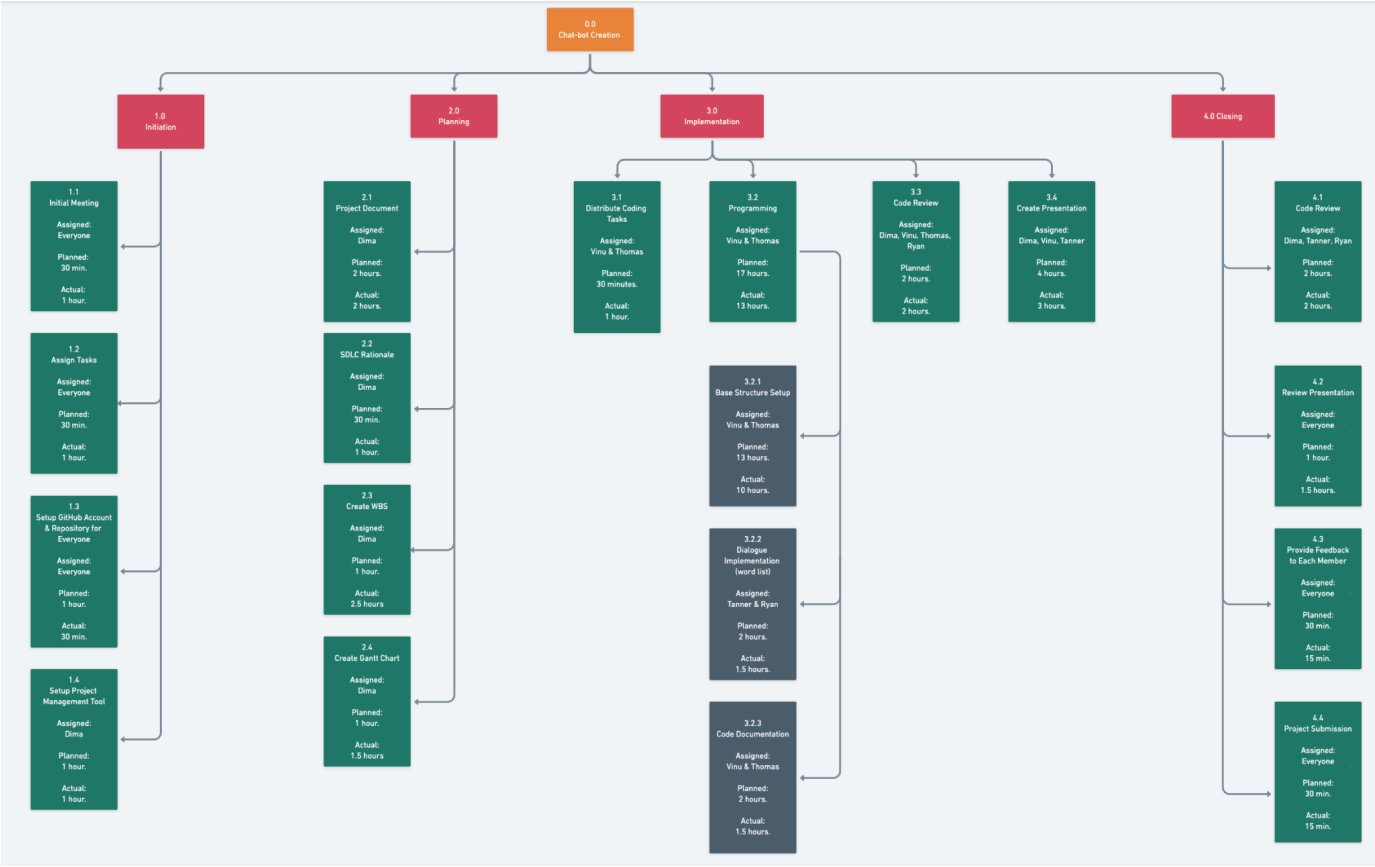
1. Test the chatbot by entering the responses used in the JSON file.
    a. Ensure that the chatbot understands the most common “yes/no” answers patterns.
2. Test the chatbot by entering the responses that are not in the JSON file.
    a. Ensure that the program will return a meaningful response.
3. Review the documentation of each class and method.
    a. Ensure that code documentation is understandable to everyone
    b. Ensure that code documentation has no grammar/spelling errors.
4. Test program for any exceptions or crashes.

2.2.5. Evolution

1. The program must evolve to meet changing requirements of the user and market.
2. The program must be modified in case of errors or vulnerabilities after the delivery.
3. Update test cases based on new functionality.

3.0 Work Breakdown Structure

The Work Breakdown Structure below is based on the tree-like structure. The root (title of the project) has four subtasks representing each project stage. Each subtask shows its subtasks that were performed during the time of this project. Each subtask has a title, assignees, estimated time of completion, and actual time of completion of the task.

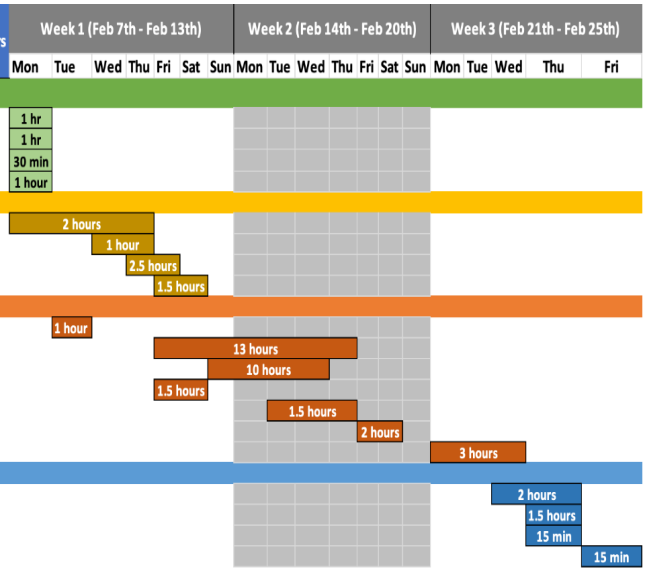


See the README file for a larger version of the file.

4.0 Gantt Chart

The Ghant Chart below represents a tree-like structure based on the WBS but condensed into a schedule-like form. Unlikes WBS, this chart contains more detailed information on the start and end date of the task, what task should be completed first to be processed with the next one.

Task Number	WBS Number	Task Title	Task Participants	Start Date	End Date	Duration	Predecessors
1	1	**Initiation**					
2	1.1	Initial Meeting	Everyone	2/7/22	2/7/22	1 day	-
3	1.2	Assign Tasks	Everyone	2/7/22	2/7/22	1 day	-
4	1.3	Setup GitHub & Repository	Everyone	2/7/22	2/7/22	1 day	-
5	1.4	Setup Project Management Tool	Dima	2/7/22	2/7/22	1 day	-
6	2	**Planning**					
7	2.1	Create Project Document	Dima	2/7/22	2/10/22	4 days	2, 3, 4, 5
8	2.2	SDLC Rationale	Dima	2/9/22	2/10/22	2 days	7
9	2.3	Create WBS	Dima	2/10/22	2/11/22	2 days	7
10	2.4	Create Gantt Chart	Dima	2/11/22	2/12/22	2 days	7
11	3	**Implementation**					
12	3.1	Distribute Coding Tasks	Vinu & Thomas	2/8/22	2/8/22	1 day	2,3,4,5
13	3.2	Programming	Vinu, Thomas, Ryan, Tanner	2/11/22	2/17/22	6 days	12
14	3.2.1	Base Structure Setup	Vinu & Thomas	2/13/22	2/16/22	3 days	13
15	3.2.2	Dialogue Implementation	Tanner & Ryan	2/11/22	2/12/22	2 days	12
16	3.2.3	Code Documentationn	Vinu & Thomas	2/15/22	2/17/22	2 days	14, 15
17	3.3	Code Review	Tanner, Ryan, Thomas, Vinu	2/18/22	2/19/22	1 day	14, 15, 16
18	3.4	Create Presentation	Dima, Vinu, Tanner	2/21/22	2/23/22	2 days	17
19	4	**Closing**					
20	4.1	Code Review	Dima, Tanner, Ryan	2/23/22	2/24/22	1 day	17
21	4.2	Review Presentation	Everyone	2/24/22	2/24/22	1 day	20
22	4.3	Provide Feedback to Each Member	Everyone	2/24/22	2/24/22	1 day	20, 21
23	4.4	Project Submission	Everyone	2/25/22	2/25/22	1 day	20, 21, 22



See the README file for a larger version of the file.

5.0 Project Limitations

At this stage of the project, the chatbot is not capable of performing the following tasks:

- Unable to handle incorrect spellings.
- The sample model of available answers is limited.
- The chatbot is not trained on unknown responses.
- Unable to recognize different parts of speech.
- Unable to identify synonyms.

6.0 Sample Output

6.1 Proper Conversation Handling

6.2 Improper Conversation Handling