Logan Drda
Student ID: 011010779
C951 – Task 1

## A.

This chatbot helps student users identify one of five relevant IT jobs that best matches their preferences. The motivation behind the creation of this chatbot is to reduce the workload of a university career advisor to address the rising enrollment of computer science students. At its core, the chatbot helps the student navigate through a perfect binary tree, where each level presents the user with a choice. The chatbot is programmed to respond to each unique choice with follow-up questions that eventually narrow the five IT jobs down to a single, most suited job for the student. The chatbot makes use of randomized responses in some cases to "humanize" itself and make the interaction feel more personal for the student.

## B.

Below is a list of the 5 types of jobs that the chatbot can recommend to the user.

1. Software Developer
2. Data Scientist
3. Network Engineer
4. Cybersecurity Analyst
5. UX/UI Designer

## C.

Please see the attached .zip file in the project submission for the code.

Logan Drda
Student ID: 011010779
C951 – Task 1

## D.

The first training case was a student who was fairly confident in their ability to read code, and they knew they wanted to be involved in an ethical field. The second training case was a student who considered themselves very analytical and did not like to focus on the user-experience portion of a project. Both of these training cases were selected because they represented users who did not have a clear-cut path forward for determining their career choice. In other words, these cases represent users who would benefit the most from the chatbot and experience all of the functionalities of the chatbot.

To enhance the chatbot's functionality, I used several features of AIML (examples follow this paragraph). To keep track of the navigation through the perfect binary tree structure, I used the category pattern/template duos to continually route the user to appropriate follow-up questions. To further enhance the functionality, I implemented postback buttons to send a predefined "secret" message to the chatbot while ensuring readability and consistent natural language for the user. I also used <srai> tags to prevent the duplication of templates across multiple categories, which helped to keep the chatbot code more concise and less prone to error. Finally, I made use of the <random> tag to allow the chatbot to provide more personal and varying responses to the user.

**Examples of AIML in the chatbot code:**

<postback> button:

```
<category>
    <pattern>ANSWERDETAILED1</pattern>
    <template>
        Good! I have another task for you.<br/>
        Choose the option that fits you the best.
           <button>
               <text>Ethical</text>
               <postback>ANSWERETHICAL1</postback>
            </button>
           <button>
               <text>Empathetic</text>
               <postback>ANSWEREMPATHY1</postback>
            </button>
    </template>
</category>
```

<srai> tag:

```
<category>
<pattern>HEY ^</pattern>
<template>
    <srai>HI</srai>
</template>
</category>
```

Logan Drda
Student ID: 011010779
C951 – Task 1

<random> tag:

```
<category>
    <pattern>HI ^</pattern>
    <template>
        <random>
            <li>Hello! I am your Career Helper, here to assist you in finding a career in IT that best suits you.</li>
            <li>Hey there! I'm the Career Helper, and I'm here to help you find an IT career that matches your style.</li>
            <li>Welcome! You can call me the Career Helper. My purpose is to help you find a suitable IT career.</li>
        </random>
        <srai>ASKFIRSTQUESTION</srai>
    </template>
</category>
```

To test the training cases with the chatbot, a full session for each training case was initiated and the chatbot's responses were evaluated during each step of the process. For both cases, the user was simulated by choosing responses that would best represent that specific user. If the chatbot recommended erroneous careers for the user, an adjustment was made in the code to ensure that the chatbot was functioning properly. Similarly, if the chatbot was routing the user to the incorrect follow-up question, the code was debugged and fixed to ensure the continued functionality of the chatbot.

## E.

Please see below for a step-by-step installation manual for the chatbot.

1. Open a web browser and navigate to https://pandorabots.com
2. Log in to your account
3. Navigate to https://home.pandorabots.com/dash/bot-directory
4. Locate the search bar and enter "WGU_C951_Career_Bot_0779"
5. Click on the result
6. In the chat window that opens, enter "hi" or "hey" or "hello" to begin

## F.

The development environment of the chatbot consisted of the Pandorabots IDE and the AIML language. There are several strengths and weaknesses that come with the use of this environment.

The first strength of the environment is that it is user-friendly and offers a simple path to deploy the chatbot for public use. A developer with minimal coding knowledge can skim through the documentation and quickly pick up the basics to begin creating their own chatbot. The numerous shortcuts, such as a button to add a link into the template, assist in development time and reduce the chance for errors. Thanks for the simple deployment process, a developer can have a publicly-accessible chatbot in a short amount of time. All of this leads to a simple, quick, and safe development process of the chatbot.

Another strength is that, by allowing for AIML scripting, Pandorabots offers great extensibility and support from the AIML community. This aids the development process because the environment allows for the use of libraries. The backing of the AIML community helps developers get quick and detailed answers to their questions, which can significantly cut down on development time.

One weakness of the environment is that extensive scripting is required due to the nature of the pattern/template couplings within categories. This is a weakness because the time taken to create all of the input/output pairs can become immense if the chatbot is complex. This would require additional attention to detail, increased organization, and a much higher chance for human error with the scripting, leading to longer development times.

A second weakness is that the Pandorabots chatbots cannot start the conversation with the user, so the user must initiate without a prompt from the chatbot. This is a weakness because it can lead to confusion for the user when there is no starting prompt to guide them. A user might not know what to enter, and they might give up on the chatbot without ever experiencing its functionalities. Due to this, additional patterns must be accounted for to handle unexpected user input in their first entry.

## G.

To monitor the chatbot, all expected inputs will be entered in a unique session. In the case of this application, there are 30 possible paths to take once the initial data collection stage begins. Therefore, all 30 cases will be tested to ensure that the chatbot responds as expected.

To maintain the chatbot with the goal of improving the user's experience, session logs will be reviewed periodically to determine if the users are expecting additional functionality from the chatbot or if a certain functionality of the chatbot is not operating as expected. If a specific condition appears frequently, then the code will be adjusted to address the issue so that the users have a better overall experience with the chatbot.

## H.

The link to the Panopto video is provided in the project submission.

## I.

No outside sources were quoted, paraphrased, or summarized.