# Section A: Explanation of Chatbot Functionalities

As interest in our Computer Science Bachelors Program grows (as well as the student population itself) we have observed Career Advisors are unable to accommodate the increasing workload. To help manage the more frequently asked questions these students pose, we will be implementing a Chatbot which can guide students through frequently asked questions, thereby freeing advisor workload for more specific and in-depth career guidance.

As a proof of concept, our first chatbot release will be able to do the following:

1. Gather information about our student's interests, strengths, and career goals

2. Recommend a Computer Science career domain based on the student's answers

3. Provide further information about the job recommendation via hyperlink (to encompass skills, duties, average salaries, etcetera).

# Section B: Identifying the Five (or Six) Jobs

The chatbot will recommend the following six jobs to our students:

1. Front End Developer
2. Back End Developer
3. Quality Assurance Specialist
4. Data Scientist
5. Network Engineer
6. Cybersecurity Specialist

# Section C: Providing the Code Files

Please see the attached Zip File for the code referenced in this document.

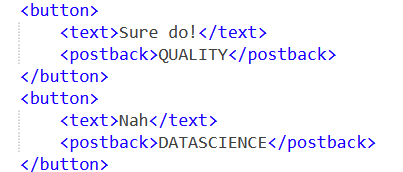
# Section D: Explaining our Training Cases

In this section, we will analyze our various test cases represented by the career paths provided, as well as beta testing which we performed with a small group of students. Each progressive beta test brought up more ways we could enhance our bot by design and by implementing additional Artificial Mark Up Language features to enhance functionality.

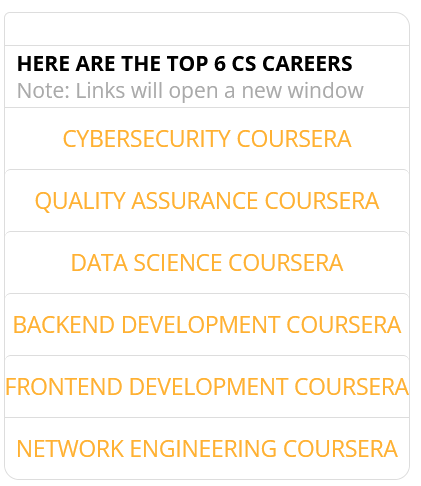
Our training cases were selected based on common student inquiries and select for the most divisive character traits which students tend to possess. We would deploy the bot and assess student responses until we pinned down what students generally want out of the bot. Our scenarios assess the student’s desire to build new things versus stick to the tried-and-true, the desire to solve problems versus rely on tradition, and their tendencies toward introversion versus extroversion. These end careers serve as training cases for the chatbot. Mind you, further releases will more thoroughly assess the applicability of the recommended fields, as these recommendations are relatively arbitrary at this time in version 1.0. Below are the example careers/test cases:

* Front End Developer
  1. The user is someone who likes to build things from scratch
  2. The user is someone who does not like to solve complicated problems
  3. The user tends towards extroversion
* Back End Developer
  1. The user is someone who likes to build things from scratch
  2. The user is someone who likes to solve complicated problems
  3. The user tends towards introversion
* Quality Assurance Specialist
  1. The user is someone who does not like to build things from scratch
  2. The user is someone who does like to solve complicated problems
  3. The user tends towards extroversion
* Data Scientist
  1. The user is someone who does not like to build things from scratch
  2. The user is someone who does like to solve complicated problems
  3. The user tends towards extroversion
* Network Engineer
  1. The user is someone who does not build things from scratch
  2. The user is someone who does not like to solve complicated problems
  3. The user tends towards introversion
* Cybersecurity Specialist
  1. The user is someone who does not build things from scratch
  2. The user is someone who does not like to solve complicated problems
  3. The user tends towards extroversion

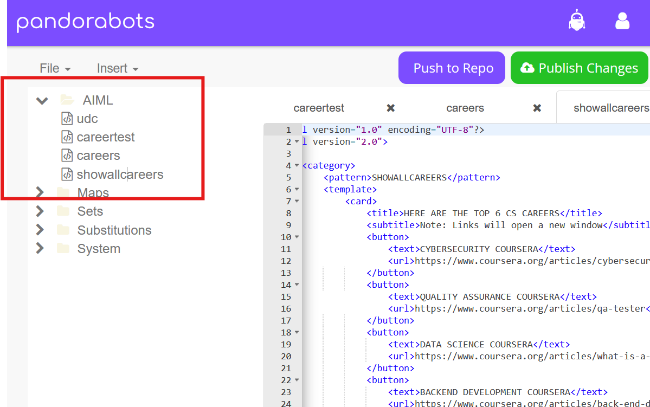
After our first deployment for testing, we noticed common words used by students would send the bot careening into its basic UDC command. Eventually we realized that hard coding every single possible question was time prohibitive. To fix this problem, we made our bot a binary tree which was strictly navigable by buttons. This serves to make the bot easier to use while also negating any issues we might have processing responses. There are always one or two buttons presented at most to the user so we can guide them more carefully through the decision tree to the end-point. After all, the purpose of the chatbot is to *save* the career department time, not add additional time requisite for creating a fully fleshed out chatbot. Below, we see an example of our button implementation:

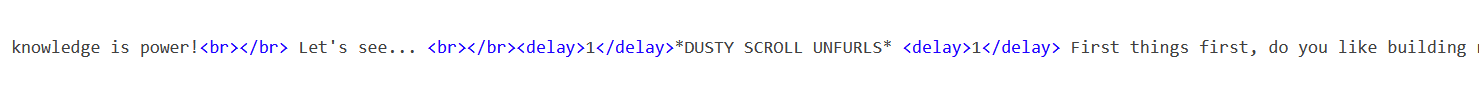


Further testing commenced, and we noted our students wanted to know what other career options existed beyond the single option presented to them at the test’s end. After careful consideration, we added a general “All Options” card which provided all the possible careers so they did not have to take the test multiple times with different answers to get the links at the end. Students need to merely click on the button provided and be automatically linked to the resource:

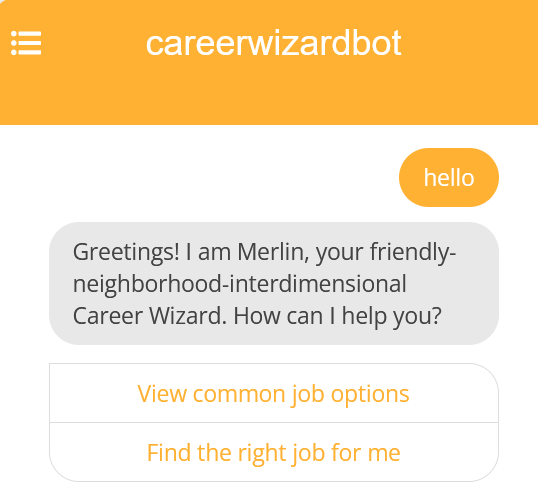


There are four AIML files to separate specific concerns within the chatbot: the “general udc” pattern, the “careertest” file, the “career” file, and the “show all” file. These files were separated to make the code more navigable. The first is a catch-all pattern for if we receive text in the chat that does not match our options. The second is the bulk of the bot, and guides the user through our decision tree. The third gives the career answers which are provided in the test. Finally, the fourth is a card which allows for the “show all options” feature in our test, if they prefer to see all options at once rather than navigating the test:



Students complained about the bot immediately responding to their answers without pause. Because there is a predictable pattern of question-response followed by two buttons, students who are not paying close attention miss if the bot processed their previous response. To solve this, we placed delay tags throughout the AIML code to make the bot seem more realistic. This delay makes our bot seem more realistic, as if it paused to think of a response or process our prompt:

We also endeavored to introduce character and persona into our bot after student comments that the bot was rather boring. After careful consideration, we made our bot a “career wizard” with its own personality and flavor. This makes the bot more approachable and fun to use.



# Section E: Installation Manual

Students will take the following steps to access our chatbot:

1. Navigate to the following link in their preferred web browser
   1. <https://playground.pandorabots.com/home.html>
2. Create an account and log in. If already logged in, click “Go To Dashboard” in the top right of screen
3. Select “DIRECTORY” in the left-hand toolbar and search for “CareerWizardBot”
4. Type “Hello” to receive the first prompt

# Section F: Assessing the Environment

We found the Pandorabots environment to be extremely easy to use. AIML is extremely intuitive and doesn’t require much syntax to learn. One can dive into developing a basic bot without much of any friction. The documentation was also readily available for learning how to implement various tags for additional features.

However, there were some glaring weaknesses to the platform as well. Using a XML-adjacent language meant more programming-related features were out of reach. We ended up refactoring our code with the hopes of using lists for the careers, storing character traits for the students, and programmatically selecting the right career from the list. This should have been a simple task at face value, but I found the mess of tags and lack of general intuitiveness of the programming to be too complicated. The window has a development field which begins halfway in the page and cannot be minimized. This means indented tags nestled within still more tags became prohibitively difficult to read. We ended up abandoning our more “programmed” solution, reverting to a forced version where each branch of the program was practically copy-pasted and funneled through buttons to prevent unanticipated answers.

# Section G: Maintenance and Future Monitoring

Our bot has a lot more work ahead of it before it can be truly considered for release. Ultimately, we would like to make a chat bot that can accommodate more general user responses and get away from the binary tree. Also, the career options presented to students are arbitrarily chosen to establish the proof of concept. Some are more aptly chosen and others are perhaps incongruent with the actual personality type one might have in each profession. We have essentially shoe-horned students based on three basic questions that are not as tailored or specific as a counselor might provide. We will continue to beta test and monitor feedback and general user experiences so we can release a true 2.0 version with more functionality. Based on feedback we hope that better questions will be posed, more career options provided, and better results experienced by all.

Section I: Sources Cited

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from <https://www.coursera.org/articles/cybersecurity-job-description>

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from <https://www.coursera.org/articles/qa-tester>

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from [https://www.coursera.org/articles/](https://www.coursera.org/articles/cybersecurity-job-description)what-is-a-data-scientist

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from <https://www.coursera.org/articles/back-end-developer>

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from <https://www.coursera.org/articles/front-end-developer>

Coursera. (n.d.). Cybersecurity Job Description: What to Expect. Coursera. Retrieved June 18, 2024, from <https://www.coursera.org/articles/network-engineer>