Requirements Engineering
CMS 340 – Software Engineering
Noah Harvey, Jenny Goldsher, Deandra Martin, Hiro Sato
11/1/2020

Virtual Tutoring Application

Narrative:

Students in CMS 120 have come to tutoring sessions with several consistent issues. For instance, students write algorithms with incorrect data types for the variables they use. This type of issue comes from how Python is formatted (not making programmers declare data types on assignment/declaration) and obfuscates details from programmers. Although the format of Python is oriented towards making programming easier, beginners do not inherently learn necessary concepts because the language covers up many aspects of programming complexity for the sake of ease of use.

Because CMS 120 is an introductory course, students are not expected to have prior knowledge of these concepts, however, it is good practice to teach students to code defensively and understand how the code they have written works. Since CMS 120 students are beginners coding in python – which allows them to program with more freedom than many other high-level languages – they may face more difficulty when they reach upper level CS classes if they never have a chance to develop a proper understanding of key programming concepts. This emphasizes the importance of having beginner students learn effective coding practices early on.

To combat this problem, there is a need for a web application that can read through student code snippets, give feedback, and ask questions. The program will not format the code for students, but it will ask related questions and challenge them to further their understanding of python. This would be a comprehensive way for CMS 120 students to exercise and reflect on their understanding of good coding practice taught in class while they are programming.

Requirements Elicitation:

To formulate the requirements, we gathered information from CS tutors by facilitating JAD Sessions. We scheduled meetings and reviewed other related applications (Slack, CodingBat, LeetCode). The tutors first described the project generally independent of a specific platform which allowed us to establish a mutual and solid understanding of what we wanted to achieve.

In the meeting we discussed topics such as:

- The objective of this project/what the client is expecting to get out of this project
- How students should be able to interact with the system
- The types of questions the system should generate
- How the system should respond to student's input

By holding a JAD session, we were able to pinpoint contradictions, ambiguities, and misinterpretation that could have occurred between clients and developers in real time. Further, we were able to discuss and address those issues. In addition, having both clients and developers in a meeting allowed us to make decisions that both sides could agree on quickly. We discussed whether to allow the system to format the code they received, and we decided that it would not – in keeping with the honor code of Rollins College to avoid unauthorized assistance. We also

clarified that the system would not execute code passed by the student as this represents a security vulnerability and depending on its use could also violate Rollin's Honor Code. We further discussed general types of questions the system would challenge the student with and clarified that they cannot be open-ended.

Requirements Analysis:

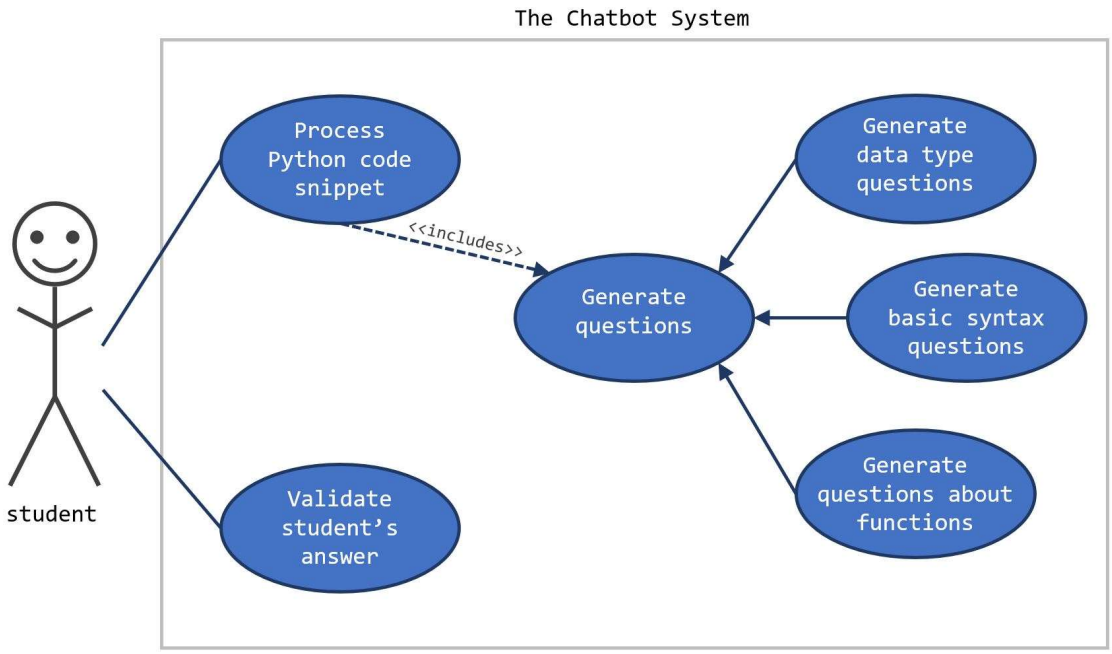
We reviewed the information we had gathered and polished our understanding of what the system needs to be able to do as well as how it should execute these tasks after the JAD session. During our review and discussion within the development team, we realized that students should be able to feed in only a snippet of code instead of the entire Python file. This was the most logical strategy as the file was not getting compiled but parsed, and it would make students practice their coding in real time. Uploading entire files could also be a security vulnerability where the integrity of the system might be compromised by flawed files.

We also briefly discussed what software development model we should use for this project and agreed to use an incremental model. We chose this model based on its increased flexibility that would give us liberty to easily add or remove features. With short development time and a novel environment, this increased flexibility would be important.

Requirements Specification:

After meeting with CMS tutors and conducting our analysis, we came up with a list of requirements:

- The system should be accessible to all students via the internet (Web application)
- The system should allow users to enter a snippet of Python code
- It should be clear to user where they should insert their code snippet
- The application should be in the form of a chatbot where user can easily interact
- The system should generate answers but not show the answer until students at least try to answer those questions on their own first.
- The system should generate set of questions based on the Python code that user feeds in
- The system should generate questions such as:
- Data type questions
- Basic syntax questions
- Questions about functions
- The system does not necessarily have to display all the possible questions, the number of questions can be selected or randomly generated (Default is 7).
- The system is a parser, not a compiler. This program should not execute the Python file and provide its output.
- The system should provide a link to a page that discusses common error messages and their meaning.



Use Case Diagram:

We also modelled our understanding of how students should be able to interact with the system using a Use Case diagram. Users will interact with the system when they insert the snippet of code or when they type in their answers to the questions generated by the system.