The Project Plan

1. The Automated Teaching Assistant

Our goal is to create a chat agent that can act as a first year computer science TA to students. Since the chat agent will take on the role of a mentor it will assume that the user will take on the role of a student. Specifically, this student will only be asking questions in the field of introductory computer science. Our system should be able to handle basic first year level questions, such as, “What should I use to represent whole numbers?”. In answer to this question the chat agent should respond with something along the lines of, “The int base type is used to store whole numbers.” and potentially even give an example code snippet. Since the project is written in Java and first year computer science at UBC Okanagan is taught in Java, the program will respond with Java related information. It is our hope to create a product that students could potentially use one day. Our progress can be seen at https://github.com/mbojey/COSC310A2.

2. Software Development Life Cycle

2.1 The SDLC Selection

After analyzing several SDLCs our group collectively decided that we would be best suited by a combination of the RAD and Scrum SDLC types, creating a new SDLC that we like to call the SCRAD. From the RAD, we adopted the central idea of getting an up and running prototype as soon as possible, with the intention of adding features as we go. Other features we took from the RAD structure were the usage of already generated code, since we will be using many of Java’s libraries and open source features. Qualities from the Scrum structure that we took were short sprints of coding and frequent meeting among the group. Finally, as many of our team have access to first year computer science, students it will be relatively easy for us to test our product on our target audience and receive feedback. This made the most sense to us as we are able to meet several time per week and often need to work alone or in small groups due to individual schedules. This allowed us to take advantage of our SDLC’s by meeting frequently and implementing small features between meetings.

2.2 The SDLC Structure

The work break down of our structure falls into two main categories that contribute to the completion of assignment two. The project plan is broken down into 4 sub-categories or tasks that make up the completion of the project plan. The second category, coding is broken down into iterations due to the nature of our SDLC. Each iteration starts with a group meeting to discuss what needs to happen. Here the WBS gets revised and updated by everyone from the previous iteration. Estimated times are made and recorded for the upcoming iteration as well. Everyone then goes off to start their task, test it and potentially recruits users to test the system if needed. Throughout our project we had 5 iterations.

Phase I: Group Meeting

- *Task I:* discuss what we have done, where we are at and what needs to get done
    - *Sub-task I:* merging of everyone's code so the system is up to its most current state
    - *Sub-task II:* pulling from GitHub so everyone is up to date.
- *Task II:* discover any solutions for any problems that may have arisen since the last meeting
- *Task III:* assign someone to test the system at its current state if needed
- *Task IV:* decide next meeting time (near future)

Phase II: Individual work

- *Task I:* consists of working on current assignment
    - *Sub-task I:* may include working with another team member depending if their assignments are closely related
    - *Sub-task II:* if assignment is new then there will be a creation of a new branch, if not then the old branch is continued
- *Task II:* work needs to be tested
    - Sub-task I: the feature may be tested by the designer or by actual users (students) it is up to the designers desecration
- *Task III:* 11th hour fixes or addition before next group meeting, final pushes to GitHub

3. Gantt Chart Elaboration

The chart on the following page is our team's Gantt chart. The chart starts on the 22nd of January and lasts 17 days, finishing on the 7th of February. The chart shows the completion of seven of the main tasks that are required to be completed before the finish date. The tasks on the Gantt chart do not directly match up will all of the tasks in the WBS table because many of the tasks in the WBS table could be combined and be represented as a single larger task. These combinations of tasks are shown below in Table 3.1.

Project Plan	Word Library	Build Prototype	Usability	Spell Check Feature	Google Feature	Error Logging
Overview	Topic Names			Design feature	Google Search	Storing of Unanswerable Questions
SDLC	Add Topic Functionality			Implement Feature	Wolfram Alpha .java	
WBS	Define Topics					
Gantt Chart	Define Sub-topics					
	Implement Sub-topics					

Table 3.1 Table of Main Tasks and Sub-tasks Comprising Each

The Gantt chart has several interesting features. First, the grey bar within the middle of each bar represents the percent of each task that is complete. All tasks are currently 100% completed. Second, you can see that all of the tasks below the “prototype” are depending upon the completion of the prototype, i.e., the prototype must be completed before these related tasks can begin. This relationship is represented in the Gantt chart by vertical black arrows connecting the related items. Further, it is clear that the two tasks above the prototype have no dependencies.

The task of creating the word library consisted of the implementation of and the creation of the library. Our team found it most efficient to alternate back and forth between these two sub-tasks, meaning that if we found we needed more words, we would create another library and then implement it, and repeat.

Note to TA: Some of the timings between the Gantt Chart and the WBS maybe be off a little due to the program that was used to make the Gantt Chart, the length of time is the same but the start and finish dates may slightly skewed.

4. Limitations of the Chat Agent

Currently the chat agent is able to provide definitions of basic terms to students and in many cases provide them with examples of how to use the programming concept. Below is a list of limitations of the chat agent that we hope to fix in the future.

- It cannot debug code
- The agent assumes the topic of conversation is about computer science
- The agent only accepts basic questions
- The chat agent is not capable of carrying on multiple conversations at once
- The chat agent may misunderstand the users’ questions as it currently functions by searching for keywords in the users’ question
- It only responds to English questions and only with answers in English
- The chat agent assumes Java is the language upon which all questions are based