**Project Plan: A Conversational Agent**

**Apurva Narayan**

**COSC 310**

**University of British Columbia, Okanagan Campus**

Nate Charlesworth

Christopher Cheung

Connor Kingsmill

Felipe Portela

Daniel Walls

The purpose of this project is to implement a pseudo-intelligence conversational agent (a program) that will respond to user input (a human using the client software). As the true understanding and following response of a software agent to a human’s input is beyond the scope of our area of study, the program will therefore attempt to mimic intelligence through pattern matching and keyword recognition, hence the pseudo-intelligence moniker assigned to the agent. The goal of the agent’s response is to generate relevant output to the user’s input with generated sentences. Again, to contain the scope of the program, the agent will be assigned the specific role of a call centre agent and will be able to help with troubleshooting issues, handling customer complaints, assisting with product returns, etcetera.

**Project Repository Link:**  https://github.com/CrZ25/cosc310sdlc.git

**Project SDLC:** Agile: eXtreme Programming (XP)

We have chosen XP as the SDLC for this project as its scope is very development-oriented, which lends itself well to the XP paradigm. As our team-size is quite small, XP also seems relevant as there is less inherent micro-management and a focus on collaboration from the entire team to produce the best software possible. Since XP adapts a test-driven development (TDD) philosophy, we find this appealing as the success of the project will largely depend on whether the agent responds correctly to user input; by ensuring complete testing coverage through TDD, this will provide us the desired result. Further, as our team is likely to have different coding style, structure, and naming practices, an iterative refactoring process is beneficial to ensure our software remains uniform and understandable for both those developing the software itself or viewing the source later. As the requirements of the project are likely to change throughout the development process as we become more familiar with the design and implementation necessary for a functional conversational agent, something our team has no experience with, the XP approach seems most applicable as it allows for easy incorporation of changes when our requirements undoubtedly change throughout the process of the project.

**Phases:**

**Planning:**

* 1. Project outline
  2. Generate user stories (emulate them as our customer (professor) will not be providing them for this project)
  3. Requirements gathering for development iterations from user stories
     1. Testing requirements
     2. Software requirements
  4. Agree on system naming standards for classes, methods, variables, and uniformity of code style and structure, as well as documentation uniformity
  5. Formalize repository structure and create main iteration branches
  6. Outline task breakdown structure
  7. Form programming pairs and sign-up for task iterations

**Design:**

* 1. Define iteration features with simplest design
  2. Generate CRC cards for initial object identification

**Development:**

* 1. Test-driven development
     1. Design tests for feature iterations
     2. Implement code to fulfill all iteration tests
  2. Ensure required documentation is complete
  3. Release code and merge to repository
  4. Refactor repository code
  5. Repeat development phase for each feature iteration until all are implemented

**Review:**

* 1. Emulate customer acceptance tests through peer-review of implemented feature iterations
  2. Revise necessary project requirements
  3. Re-iterate through process of design-development-review cycle for any revised or additional requirements
  4. Finalize project report
  5. Release software

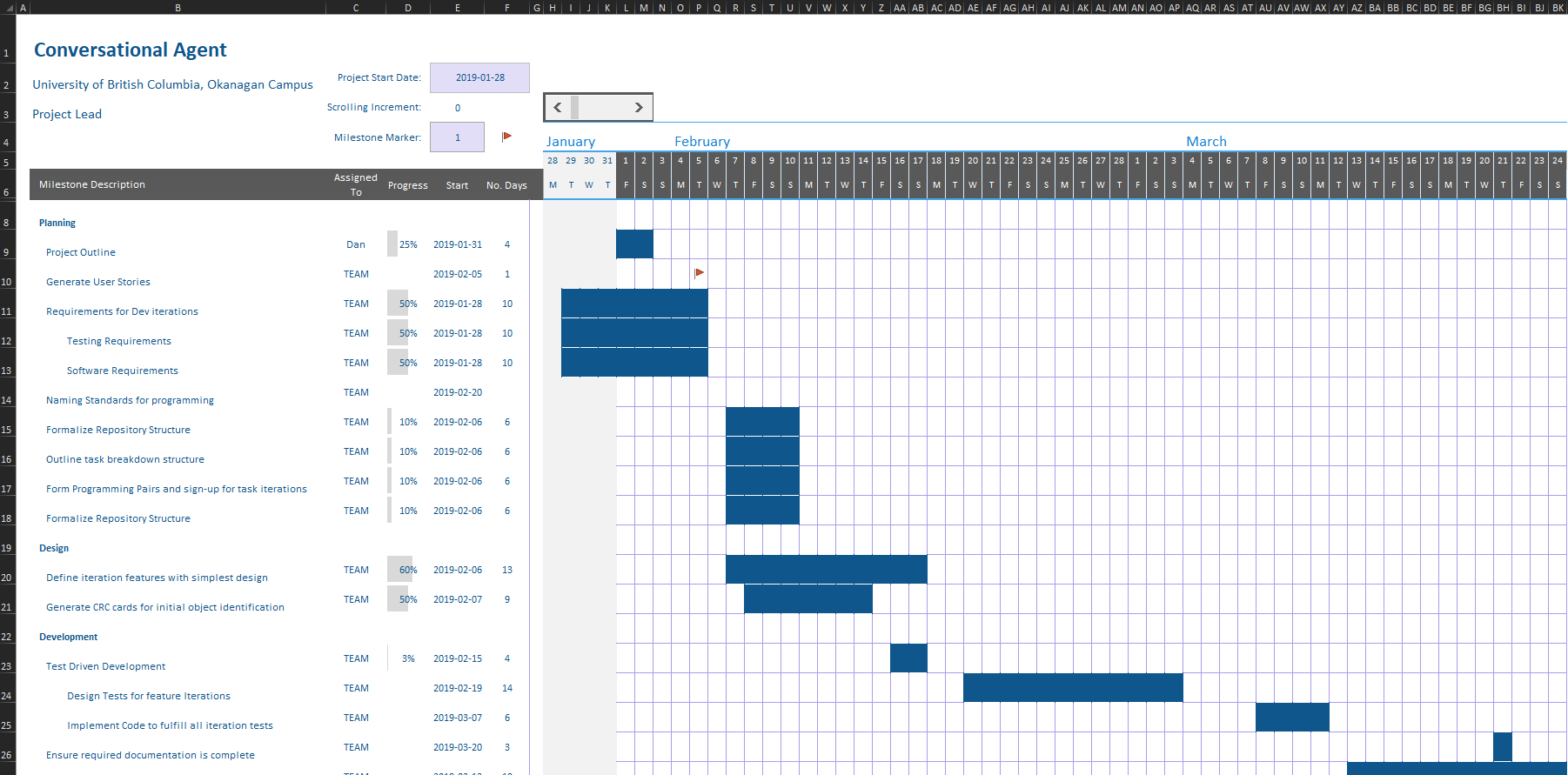
**Work Breakdown Structure:**

Our WBS is a loose outline of what to expect for the development of this project. As we are adapting the XP model, it is difficult to outline and adhere to a WBS during the development stage (3.x) as this is a reiterating process and subject to additional iterations when additional requirements are added or changed. Likewise, the design (2.x) and review (4.x) stages are also iterative in that they are repeated with the development stage when a design-develop-review iteration is undergone; however, the hour estimations for these two non-development stages are likely to be the same for each iteration, and as such are included in the current WBS as estimates for individual instances of the design-develop-review process. When the design-develop-review cycle is repeated, these stages (2.x, 3.x, 4.x) will be expanded upon and broken into their individual instances as necessary.

Further, as XP is a team-based approach to the development process, every task in the WBS is assigned as TEAM indicating that everyone will be involved in that task. This does not apply to discrete iterations of development stages (3.x), where a pair-programming team will undertake those feature iterations and be listed as such. Upon the completion of the planning stage (1.x) of the project when a rough guideline of system requirements will be known, the amount of discrete development iterations will be added to the WBS and pair-programming teams will sign-up for them as necessary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task No.** | **Task Title** | **Estimated Hours** | **Actual Hours** | **Pair-Programming Team** |
| 1.1 | Project Outline | 10 |  | TEAM |
| 1.2 | User Stores | 1 |  | TEAM |
| 1.3 | Requirements gathering |  |  |  |
| 1.3.1 | Testing requirements | 2 |  | TEAM |
| 1.3.2 | Software requirements | 2 |  | TEAM |
| 1.4 | System naming standards | 0.5 |  | TEAM |
| 1.5 | Formalize repository structure | 0.5 |  | TEAM |
| 1.6 | Outline task breakdown | 1 |  | TEAM |
| 1.7 | Form programming pairs and task sign-up | 0.5 |  | TEAM |
|  |  |  |  |  |
| 2.1 | Define iteration features | 2 |  | TEAM |
| 2.2 | Generate CRC cards | 2 |  | TEAM |
|  |  |  |  |  |
| 3.1 | Test-driven development |  |  |  |
| 3.1.1 | Design tests for feature iterations | ? |  | <member names> |
| 3.1.2 | Implement code to fulfill all tests | ? |  | <member names> |
| 3.2 | Ensure required documentation complete | 1.5 |  | <member names> |
| 3.3 | Release code and merge to repository | 1 |  | <member names> |
| 3.4 | Refactor repository code | 2 |  | <member names> |
| 3.5 | Repeat development phase for all discrete feature iterations until completed | N/A |  |  |
|  |  |  |  |  |
| 4.1 | Acceptance testing | 2 |  | TEAM |
| 4.2 | Requirements revisions | 2 |  | TEAM |
| 4.3 | Repeat design-development-review cycle for requirements additions or revisions | N/A |  | TEAM |
| 4.4 | Finalize report |  |  | TEAM |
| 4.5 | Release software | 0.5 |  | TEAM |
|  |  |  |  |  |

**Gantt Chart (Example):**



This Gantt chart is a representation of what we will use to track our development progress. The actual chart used throughout the project will have multiple iterations of the design-develop-review instances that will run in parallel with each other. Due to the size of our team, it will only be possible to actively work on two feature iterations at any given time; hence, at most, there will be two design-develop-review processes running in parallel with each other at any stage of the project. The timeslots outlined on the above chart are not representative of the actual project deadline for this stage of the project and will be adjusted accordingly throughout the planning and design stages.

**Limitations of Program:**

* Program will not be able to detect words that are spelled incorrectly.
* User input containing many key words may trigger incorrect pattern response.
* Limited number of inputs.
* Limited output sentence format.

**Sample Output:**

**Conversational Output:**

Bot: Hello, welcome to Electronics Incorporated support. My name is Bill. How are you today?

User: Hi Bill, I'm doing well.

B: Could I get your first name please?

U: My name is Sam

B: Thank you, Sam. What do you need help with today?

U: I ordered a computer from this store and it crashed and won't turn on.

B: What type of computer is it?

U: The Acme PC2000.

B: OK, your PC2000 crashed and you can't turn it on. Is this correct?

U: Yes

B: What were you doing before it crashed?

U: I was running a program.

B: OK, did the computer get hot while you were running a program?

U: No

B: How long have you been having this problem?

U: It started today

B: OK, your computer crashed and won't turn on starting today. Would you like me to try to troubleshoot the problem with you?

U: Yes please

B: Of course. Does the screen light-up when you try to turn it on?

U: Yes the screen comes on briefly

B: Does your computer boot into the BIOS?

U: No the screen just flashes?

B: Have you checked that your monitor is properly plugged it?

U: Yes

B: OK, Sam. Have you tried flipping the switch on the back of the PC?

U: Yes, it did not work

B: Are you using a power bar for your computer?

U: Yes

B: Is your power bar turned on?

U: Yes it is

B: Have you tried plugging it into different outlets?

U: Yes I have

B: Have you tried using the spare power cable that comes with the PC2000?

U: Yes

B: Have you tried using a different HDMI cable for your monitor?

U: Yes, it didn't fix it

B: Does your computer make any sounds when you turn it on?

U: Yes it makes some noise

B: What kind of noise does it make?

U: It makes a fan noise and then stops

B: OK, the fan makes some noise when you turn it on?

U: Yes, but then it stops

B: OK, the noise stops. I'm sorry that you haven't been able to fix it Sam. Would you like me to continue to troubleshoot with you?

U: I don't want you to troubleshoot anymore

B: OK, Sam. I can start a support ticket for you or I can help you start the return or refund process.

U: I think I would like to return the PC.

B: OK, I will help you return it. Do you have the order ID for your purchase?

U: Where do I find the order ID?

B: The order ID is in the confirmation email that was sent to the email you provided. Can you retrieve the order ID?

U: Yes, the order ID is 123456789.

B: OK Sam, I checked order 123456789. You would like to return the Acme PC2000 you ordered on December 15th 2018?

U: Yes that is correct

B: We will provide you with a shipping label. What city will you be shipping the return from?

U: Kelowna

B: OK, you will be shipping the return from Kelowna. Would you like to receive the shipping label at the email you provided for order 123456789 Sam?

U: Yes please

B: OK Sam, I will send the shipping label for a return from Kelowna to the email sam.sam@email.com. Is everything correct?

U: Yes thank you

B: You're welcome Sam. Before you go, would you be able to give me some feedback on our conversation today?

U: I can give feedback.

B: Thank you Sam. How helpful was I today on a scale from 1 to 5 if 1 is not helpful at all and 5 is most helpful.

U: I think 4

B: Thank you. Did I help you resolve your issue?

U: In a way yes.

B: OK. You said that I was a 4/5 and I did help you resolve your issue. Do you have any additional comments?

U: No

B: Thank you for your feedback Sam. Have a nice day.

**Sample error outputs:**

Bot: What can I help you with?

User: My computer is on fire.

Bot: I am not sure how to help you with that. Would you like me to connect you to live customer support?

Bot: We will provide you with a shipping label. What city will you be shipping the return from?

User: The city you shipped the stuff to

Bot: Sorry, I can't understand that. Could you please try saying that for me another way?