COSC310A2 GROUP 6

Link to GitHub:

https://github.com/akshaj99/COSC310A2

Members:
Name Student # - Github username

Akshaj Srinivasan #50792175 - akshaj99
Yirui Li #14891501
Qianyi (Kitty) Wang #21683289 - QianyiWang18
Tony Yue #89740609 - HDILFG
Kexin Zhao #56958317 - deffyded
Peter Zhao #39461066 - CC-ACV

SDLC Description: Agile-SCRUM

Group 6 has chosen SCRUM, the agile development process, for our SDLC framework. We felt as though a lightweight, iterative and incremental SDLC with its simple implementation was the best fit for us. We do not have a project manager or a scrum master. This is a small project and the group is dependable, productive and work independently so we did not think the roles were required.

Product backlog:

Plan, organize and control
Decide upon appropriate SDLC
Decide software design for chatbot
Finalize chatbot's role and conversation topics
Code and design chatbot
Test and refine chatbot
Present Chatbot/ Video creation
Video editing

SDLC Rationale:

More than half the group is currently out of the country and we needed a framework which would make it easy for everyone to independently know what they had to do while frequently communicating. SCRUM works while the members remain productive and we already had a product backlog structure decided, all we needed to do was self-assign tasks, which went down smoothly. Everyone managed their tasks independently while checking in with the

group, updating on their completion. Sprints were made by subgroups independently, adhering to completion date and times.

Limitations:

1. The chatbot only focuses on 6 topics of hobbies
2. The chatbot only can have brief conversation with user
3. The chatbot only can work on desktop or laptop

Listing of phases

- 1.Plan, organize, control
  - 1.1 Project Preparation
    - 1.1.1 Github repository
    - 1.1.2 Work breakdown and assignment
  - 1.2 Initial Planning
    - 1.2.1Decide SDLC
    - 1.2.2Choose structure type
  - 1.3 Review roles
- 2. Software Design
  - 2.1 Software planning
    - 2.1.1 Decide Software
    - 2.1.2 Chatbot role and conversation topics
  - 2.2 Code and create chatbot
    - 2.2.1 Writing code
    - 2.2.2 Chatbot testing and refinement
  - 2.3 Chatbot Demo
- 3. Documentation
  - 3.1 Video
    - 3.1.1 Video Recording
    - 3.1.2 Review tests, git-flow and contributions
    - 3.1.3 Video Editing
  - 3.2 Listing
    - 3.2.1 List of limitations of program
    - 3.2.2 Listing of phases
    - 3.2.3 WBS
    - 3.3 Gantt Chart
  - 3.4 Description
    - 3.4.1 Project description
    - 3.4.2 SDLC description & rationale
  - 3.5 Software design
  - 3.6 Code documentation
  - 3.7 Testing strategy
  - 3.8 Tools

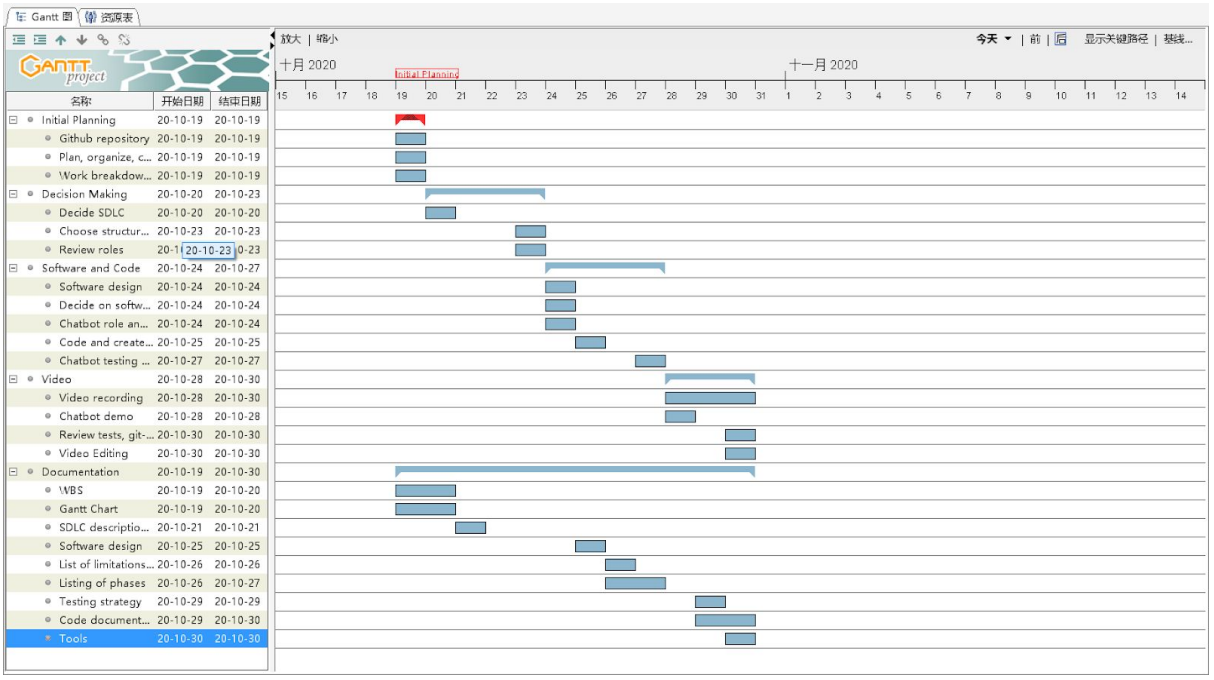
Word Breakdown Structure:

Task NO.	Task Title	Est. hrs	Real hrs	Est. start date	Est. complete date	Real start date	Real complete date	Assigned to	Done by
1	Plan, organize, control	2	2.5	Oct 19th	Oct 19th	Oct 19th	Oct 19th	Everyone	Everyone
1.1	Github repository	.5	.5	Oct 19th	Oct 19th	Oct 19th	Oct 19th	Everyone	Everyone
1.2	Work breakdown and assignment	1.5	2	Oct 19th	Oct 19th	Oct 19th	Oct 19th	Everyone	Everyone
2	Decide SDLC	1	1.5	Oct 20st	Oct 20st	Oct 23rd	Oct 23rd	Everyone	Everyone
2.1	Choose structure type	0.75	0.75	Oct 23rd	Oct 23rd	Oct 24th	Oct 24th	Everyone	Everyone
2.2	Review roles	0.25	0.5	Oct 23rd	Oct 23rd	Oct 24th	Oct 24th	Everyone	Everyone
3	Software design	4	4.5	Oct 24th	Oct 24th	Oct 24th	Oct 24th	Everyone	Everyone
3.1	Decide on software	0.5	0.5	Oct 24th	Oct 24th	Oct 24th	Oct 24th	Everyone	Everyone
3.2	Chatbot role and conversation topics	1	1	Oct 24th	Oct 24th	Oct 24th	Oct 24th	Everyone	Everyone
3.3	Code and create chatbot	2.5	3	Oct 25th	Oct 25th	Oct 25th	Oct 25th	Kitty	Kitty
4	Chatbot testing and refinement	3	4	Oct 27th	Oct 27th	Oct 27th	Oct 27th	Kitty	Kitty
5	Video recording	5-6	8	Oct 28th	Oct 30th	Oct 28th	Oct 30th	Everyone	Everyone
5.1	Chatbot demo	4	6	Oct 28	Oct 28th	Oct 28th	Oct 28thth	Kitty	Kitty
5.2	Review tests, git-flow and contributions	2	3	Oct 30	Oct 30	Oct 30	Oct 30	Tony	Tony
5.3	Video Editing	1-2	2	Oct 29/Oct 30th	Oct 30th	Oct 29th	Oct 30th	Peter, Akshaj, Kexin	Peter, Akshaj, Kexin
6	Documentation	5-6	10	Oct 19th	Oct 30th	Oct 19th	Oct 30th	Everyone	Everyone
6.1	SDLC description & rationale	1-2	1.5	Oct 21st	Oct 21st	Oct 21st	Oct 23rd	Akshaj	Akshaj
6.2	List of limitations of program	1	1	Oct 26	Oct 26	Oct 26	Oct 26	Tony	Tony
6.3	Listing of phases	3	3	Oct 26	Oct 27	Oct 26	Oct 27	Peter	Peter
6.4	WBS	1.5	1.5	Oct 19th	Oct 20th	Oct 19th	Oct 20th	Akshaj	Akshaj
6.5	Gantt chart	5	5	Oct 19	Oct 20	Oct 20	Oct 21	Tony	Tony

6.6	Software design	3	4.5	Oct 25	Oct 25	Oct 26	Oct 26	Kitty	Kitty
6.7	Code documentation	4	4.5	Oct 29	Oct 30	Oct 29	Oct 30	Peter	Peter
6.8	Testing strategy	3	4.5	Oct 29	Oct 29	Oct 29	Oct 30	Kitty	Kitty
6.9	Tools	1	2	Oct 30	Oct 30	Oct 30	Oct 30	Peter	Peter
7.0	Editing, finalization & submission	2	2	Oct 30	Oct 30	Oct 30	Oct 30	Kexin	Kexin

Work distribution

We created a wechat group for our work distribution. We listed our tasks in that group and everyone chose their work preference.



Software Design

The program is created and tested on IntelliJ IDEA CE, which is an interactive, text-based chatbot which is capable of completing 30 turns of conversation in a realistic manner. The program is simulating a conversation between a pair of new friends. It will focus on the topic of "hobbies". According to the user's questions, the chatbot "Mike" will give multiple responses.

At this stage, the program contains 6 different topics of hobbies, which are reading, basketball, swimming, cooking, painting, and movies. And each topic contains 3 types of

questions: Do you like the hobby? What do you know about the hobby? And what is your habit in that hobby? According to different types of questions, “Mike” can provide a corresponding answer.

The main form of the stimulating conversation is the user enters a question, and “Mike” gives a response. Therefore, the program scans in the user’s input sentence, splits it into words and finds the keyword. The different answers will be printed out according to different keywords. For example, if the user enters “Do you like playing basketball?” The keywords “like" and “basketball” will be extracted, and the response will be “I love playing basketball!”

Documentation of Each Code File

Main: the class of running the whole program.

Opening: the class that contains the “Mike’s” greeting content.

Conversation: the class contains the 4 methods: **public ArrayList<String> input(){}** scans in the users input. **public ArrayList<String> splitOnSpace(String line){}** splits the input sentences into words and saves them in an ArrayList. **public void userAskQuestion(ArrayList<String> question){}** distinguishes the topics of the input question according to the keyword, and provides corresponding answers. **public void differentQuestions(ArrayList<String> q, Answers a){}** distinguishes the types of the input question according to the keyword.

Answers: the interface which contains the three methods corresponds to three types of questions for all topics: Do you like the hobby? What do you know about the hobby? And what is your habit in that hobby? The three methods are **void like(); void knowAbout(); and void habit();**.

Reading: this class implements Answer, which contains the corresponding responses of the reading topic.

Basketball: this class implements Answer, which contains the corresponding responses of the basketball topic.

Cooking: this class implements Answer, which contains the corresponding responses of the cooking topic.

Swimming: this class implements Answer, which contains the corresponding responses of the swimming topic.

Painting: this class implements Answer, which contains the corresponding responses of the painting topic.

Movies: this class implements Answer, which contains the corresponding responses of the movies topic.

Test Strategy

In this program, there are 9 test classes totally, and all have 100% test coverage. There are 22 test methods passed totally which cover all situations in the "talking hobbies with a new friend conversation".

Each test class is running in a similar way. The purpose is to test whether the program will give a present answer based on user input by **assertEquals(~, ~)** method. For example, in the **TestTopicBasketball** class, the user input is set to "Do you like playing basketball?" The test method will be passed only if the output response is "I love playing basketball!"