



PPM Appendix – The development of Chatbot (A.V.A)

Ayodamola Ajibola **N0590642**

Khawla El Msalti **T0051675**

Jia Ren Goh **T0070823**

Alex Moreton-Straw **N0494166**

Contents

[**1.** **Introduction** 5](#_Toc513310626)

[**2.** **Specification** 5](#_Toc513310627)

[**3.** **Research** 6](#_Toc513310628)

[3.1 Research intoExisting Apps 6](#_Toc513310629)

[Mitsuku 6](#_Toc513310630)

[Rose 6](#_Toc513310631)

[Foxsy 7](#_Toc513310632)

[Swelly 7](#_Toc513310633)

[4. Features descriptions 7](#_Toc513310634)

[Text recognition 7](#_Toc513310635)

[Speech recognition 7](#_Toc513310636)

[Learning from user 7](#_Toc513310637)

[4.1 Programming methods and roles 9](#_Toc513310638)

[5.0 Project Management 10](#_Toc513310639)

[6. Team Members & Responsibilities 12](#_Toc513310640)

[6.1 Sources of information, resources required 13](#_Toc513310641)

[6.2 Risk assessment table 15](#_Toc513310642)

[**7.** **Programming languages.** 17](#_Toc513310643)

[**8.** **Design** 18](#_Toc513310644)

[8.1 Photoshop Designs 20](#_Toc513310645)

[8.2 Chatbot Design 1 – iMessenger inspired design 21](#_Toc513310646)

[8.3 Chatbot design 2 – Web-chat inspired designed 22](#_Toc513310647)

[9. Project Plan, milestones, effort & timescale for whole project 23](#_Toc513310648)

[9.1 SDLC Software Selection 24](#_Toc513310649)

[9.2 Function Tracker Layout: 25](#_Toc513310650)

[Extras: 26](#_Toc513310651)

[**10.** **Development** 28](#_Toc513310652)

[10.1 Visual Studios and C++ 28](#_Toc513310653)

[10.2 Website Development in HTML/CSS 29](#_Toc513310654)

[10.2.1 Coding design 29](#_Toc513310655)

[10.2.2, Use case diagrams 30](#_Toc513310656)

[10.2.3, Concept map 31](#_Toc513310657)

[31](#_Toc513310658)

[10.3 Software Development in Java 31](#_Toc513310659)

[10.3.1 Coding design 32](#_Toc513310660)

[10.4 Current program Designs 33](#_Toc513310661)

[10.5 DialogFlow 34](#_Toc513310662)

[11. Testing 35](#_Toc513310663)

[11.1 Alpha/Beta Testing 35](#_Toc513310664)

[11.1.1 Pros and Cons 36](#_Toc513310665)

[11.2 Testing features 37](#_Toc513310666)

[Conclusion & Evaluation 39](#_Toc513310667)

[References 40](#_Toc513310668)

[Appendices 41](#_Toc513310669)

[Meetings 41](#_Toc513310670)

[Project Tasks 46](#_Toc513310671)

[Development of the project in the early stages moving to the final one 47](#_Toc513310672)

[Figures 51](#_Toc513310673)

# **Introduction**

Chatbots are commonly compared to ‘conversational partners’ simply due to their objectives. Creating such a program would mean following a strict set of rules that prevent the product from being placed in competitions with similar systems such as Personal Digital Assistants (PDA) or Interactive Artificial Intelligence (e.g. *Amazon Echo*). As the development of our project progresses meaningful knowledge has been acquired with regards to understanding the business of chatbot creation, and in relation to the improvement in our programming capabilities with specific emphasis placed on versatility – Our program now runs on an entirely different programming language than originally intended.

**1.1 Chatbot Features**

The primary functionality of the chatbot requires it to recognise messages and respond appropriately. This is done through a having a database of selected scripts that it can recognise and appropriate replies to them. If the chatbot is designed for a more business-based route with expectations to act to streamline customer service, this might include common customer queries and the basic answers to them. In the case of a chatbot designed for business, this should also include some responses either encouraging the user to contact, or directly transferring the user to, human assistants, as there are many cases where chatbots are not currently able to properly aid the user. These would include such cases as complaints and other emotional issues where the bot will have problems correctly responding, resulting in a poor experience with the customer. Tying into this would be ensuring the user is aware that the chatbot is a program and they are not interacting with another human, as this usually results in poor experiences for the user when they discover this to be the case. The easiest way to do this would be to simply give the program a name indicative of the fact that it is a bot and/or have the bot explain to the user that this is the case.

Care should be taken to ensure the bot does not produce any content that could be considered sexist, racist or otherwise discriminatory. Any such content can be considered the legal responsibility of the owning company and can result in major fallout – see for example the case of Tay, a bot designed by Microsoft to respond to tweets that had to be shut down after less than 24 hours due to its spouting an array of sexist and racist comments. Solutions to this can include a ‘blacklist’ of words to which the bot will respond with a generic message to the effect of either not understanding or requesting the user to avoid using such language. In addition to this, keeping logs of all users, conversations and times for later review can help those maintaining the bot to avoid any offensive material. The ability to easily search these logs would also help for speed of access.

# **Specification**

Table 1Specification

|  |  |
| --- | --- |
| Milestones | Completed |
| 1. The Chatbot should be able to start conversations and carry on with the conversations effectively. |  |
| 2. The Chatbot will be able to provide adequate responses based on keyword detection. |  |
| 3. The series of codes involved will be neatly packaged and showcase our proper understanding of object-oriented programming. |  |
| 4. Simple profile creation |  |
| 5. Database to retrieve past conversation and user information |  |

# **Research**

A Chatbot is a conversational agent that interacts with users with natural language sentences. Generally, a virtual agent works by a user asking a question or making a comment with the virtual agent and it replies to the user based on its knowledge which contains of a set of templates that may matches user inputs. As mentioned by (Huang, Zhou and Yang, n.d.) Several virtual agents have been used on the Internet for the purpose of seeking information, site guidance, FAQ answering. Beside these agents which are created for certain jobs there are also agents that socialize with users just similarly how a friend would communicate with them. In this paper we demonstrate the design and implementation of A.V.A social virtual agent app and also presents other social agents in orders to illustrate the improvements of A.V.A app in comparison with the existing apps.

## 3.1 Research intoExisting Apps

The list below contains information about existing software’s that have similar features and functionality to the chat bot that will be developing. The purpose of it is to contact a test to see the strength and weaknesses these software’s have and then consider methods on how the weaknesses can be solved and how their strengths can be further developed upon. The apps within the list below are chosen based on winners of the bronze medal Loebner Prize 2017, which ranks with bots depending on their ability to pass the Turing Test or be the most human like.

According to (Al-Rifaie, 2017) the most two social human-like bots are Mitsuku and Rose: //

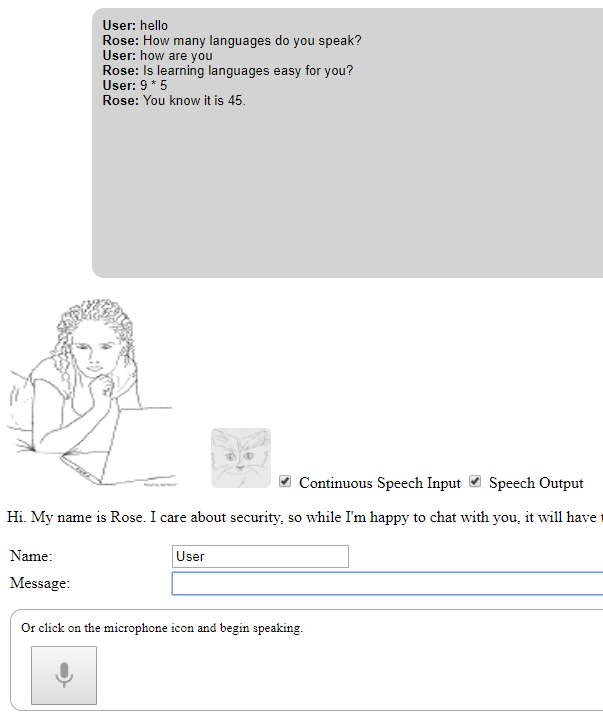
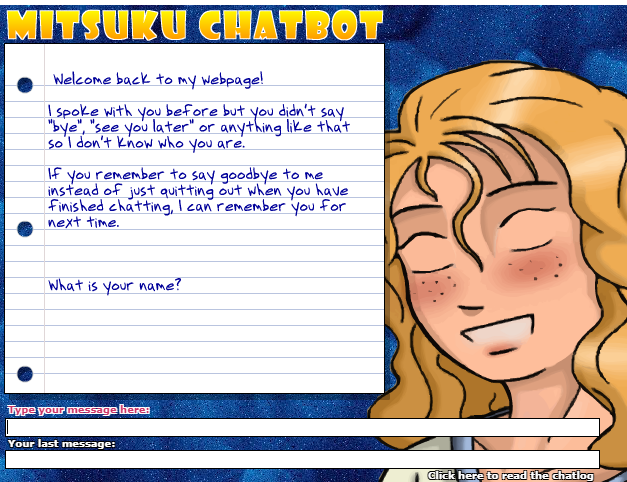
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **App** | **Score** | **Features** | **Strengths** | **Weakness** |
| Mitsuku | 27 | Text recognition  Avatar  Learning | Ability to learn from users  Reasons with specific objects  Speaks 3 languages | Not responding intelligently to tasks that user may ask and goes out of topics  i.e. bookings  Static avatar  No user profiles  no speech recognition |
| Rose | 23 | Text recognition  Speech recognition  Avatar | Input output text or voice | Cannot learn from user  Lack of spoken languages  Static avatar and old design  No user profiles  Goes aggressive and asks tons of questions one after another |

And as mentioned by (Chatbots org, 2017) the 1st ChatBottle Awards for social chatbots goes to Foxsy in the first place and Swelly in the second place:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **App** | **Votes** | **Features** | **Strengths** | **Weaknesses** |
| Foxsy | 354 | User Options  Avatar | Matching between users’ profiles | No text recognition    No speech recognition  Lack of user profile’ information can lead to unreliable matches  Lack of spoken languages  Static avatar |
| Swelly | 326 | User Options  Avatar | Users can upload photos or choice questions to be voted by others  Users can view others profiles and follow them | No text recognition    No speech recognition  Lack of spoken languages |

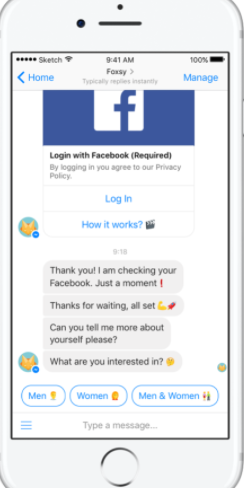
# Features descriptions

|  |  |
| --- | --- |
| **Feature** | **Descriptions** |
| Text recognition | Relies on comparing the user input characters with learned set characters stored in database and based on that an output of characters is displayed. |
| Speech recognition | Analog-to-digital converter translates the analogy wave into digital data that the computer can understand, then compares it with large library of known words. |
| Learning from user | When users input unknown words, the chat bot outputs sensible response i.e. “I’m not sure ...” and adds the user input to a file where then the developer can add it to the database. |



**Figure 1.1: Mistuku Chatbot**

**Figure 1.2: Rose chat bot**



**Figure 1.4: Foxsy**

**Figure 1.3: Swelly**

## 4.1 Programming methods and roles

|  |  |
| --- | --- |
| **Methods and roles** | **Descriptions** |
| **Figure 1.5: Database structure.** | **How the database works** This program uses a struct that holds stored variables in an array. The stored array acts like a database that holds possible user questions and expected response by the chatbot. The first position in the array “{0” holds the possible users input. the remaining positions in the array hold the possible expected response which is outputed randomly to reduce possible overlap with response.  **Why this should be implemented** This programming technique is needed to ensure the chat bot has a range of stored pre-assigned responses |
| **Figure 1.6: Find\_match method ().** | **Find\_match () method** takes the user sentence as a string input, then it searches for matching sentence with the user input from the start until the end (size of the database containing the sentences), then returns a vector of string (the matching sentence response), if no matches empty vector will be returned and handled in the main programme.  **The usage of the method** to find appropriate response to user inputs.  **Limits** responds may be repetitive.  **Improvements** may add variable storing previous respond and role that can make sure previous respond is not equal to current respond if this does not happen the method will choose another respond. |
| **Figure 1.7: learning from user role.** | **The respond method** of the class learner reads text file where phrases are stored, then it searches for matches, if no matches **the user input is written to the file** hence the programme is learning from the user.  **The usage of learning from user** the programme will have the ability to add to its knowledge phrases and responds that users may say.  **Drawbacks** users may enter wrong phrases and responds.  **Improvements** add another file where users can write to, then these files will be moderated by the developer, after that phrases and responds may be added to the original file. |

5.0 Project Management5.1 Project methodology

In considering how we would manage our project, there were two main choices that we looked into: Waterfall and Agile. Following the waterfall approach, each section of the design is completed in its entirety before moving on to the follow stage. We chose against this idea, as with multiple members working on different sections of the code and report this was inefficient. In addition, in this approach one member falling behind would result in the other members being held back.

Agile project management is an approach based on delivering requirements iteratively and incrementally throughout the project life cycle as mentioned by (AMP organization, 2017). Agile is the most suitable methodology for developing product like the Chabot where the customer/ user can engage with the developer team for any changes and improvements, as in agile approach the team will deliver small piece of requirements each time so that the customer/ user will give feedback and next deliverable will be promotion of continuous improvement.

**Team members roles (Belbin test)**

**Ayo** is the team leader. He is a dutiful, self-disciplined individual that is very adamant about hard work. He depends on his ability to keep the group organised, on target and ensuring everyone contributing to the best of their ability.

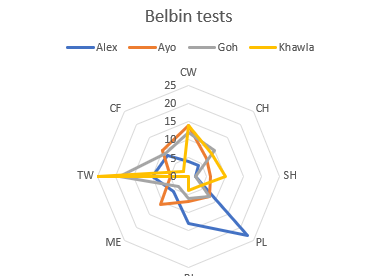
**Jia Ren** is the company worker of the team. He is practical, decisive and possesses good organizational skills. He analyses the given needs to produce working solutions for all members of the group to decide upon.

**Alex** is the plant in a team environment. He is creative and innovative. He can provide new ideas and solutions to problems in the group.

**Khawla** is the team worker when working within a team environment. She is socially orientated within the group. She can promote spirit to the team and also has the ability to respond to people.

**Ways of tracking team communication**

1. Hold effective team meetings, before start up a meeting, the team have to have reason for it so they will not waste time, then tell each member what they are expected to present in that meeting.
2. Make record of meetings, times, members who attend and deliverables, and send emails to whom missed the meetings and make sure they get on track again.



As a group a Belbin test was conducted to help full analyse each member of the groups strongest skills to ensure productivity with the group is done efficiently by each member being tasked with what they are best at.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | CW | CH | SH | PL | RI | ME | TW | CF |
| Alex | 4 | 4 | 2 | 23 | 13 | 6 | 10 | 8 |
| Ayo | 14 | 7 | 6 | 8 | 7 | 11 | 5 | 10 |
| Goh | 12 | 10 | 2 | 8 | 6 | 4 | 19 | 9 |
| Khawla | 14 | 9 | 10 | 4 | 4 | 0 | 25 | 2 |

**Figures 2.1 (left) & 2.2 (right): Belbin Test Results for our team, more specifically strength and weakness comparisons**

# 6. Team Members & Responsibilities

As the group consists of different behaviours in the workplace shaping up one team each person strengths developing the other weaknesses. The responsibilities have been given out to members according to their strength and also being involved in a secondary task which can be challenging for the member, but it will be with another team member who find the task easy.

|  |  |
| --- | --- |
| **Tasks** | **Team members** |
| **Project introduction**  Introduction of chat bots   Research of existing chat bots  Chat bots role  Chat bots aims   New ideas  Project Methodology   Members roles (Belbin test)  **Risk assessment table**  investigate all possible risks  **Concept map** planning creation analysis  **System proposal**  Sources of information, resources required  List of functional requirements   List of non-functional requirement   Use cases   DFDs | Ayo, Jia Ren Khawla Ayo Ayo Ayo  Khawla All group members  All group members  All group members  Ayo  Jia Ren and Ayo Started by Jia Ren Started by Jia Ren  Future implementation  Future implementation |
| **Project schedule**  Team responsibilitiesRadar Chart  **Gantt chart**  Planning Gantt chart  Creating Gantt chart (MS project) | Khawla Alex  Jia Ren & Ayo Jia Ren & Ayo |
| **Professional, social, ethical and legal issues   Design, Coding, Evaluation, Conclusion References Appendices** | Alex  Jia Ren & Ayo All group members |

**Tasks may be added during process, also updates may be added to some tasks.**

## 6.1 Sources of information, resources required

This section contains the resources required to successfully complete the chat bot.

|  |  |
| --- | --- |
| Software | |
| Visual Basics  Sublime/notepad++(any suitable text editor) | This is a great programming platform that can be used to develop applications in C++. It also allows the use of libraries and has a lot of resources to aid with project developments. |
| Drobox, One Drive –GitHub | This website / platform will be very useful to store and archive code. Plus GitHub can be used to complete tasks from long distances. |
| Operating systems | Windows 10 , Unix, Linux, Mac OS  A range of operating systems to test the functionality of the software plus internet access. |
| Hardware | |
| Computer systems ( PCs, Laptops and tablets) | * Pcs, Laptop, Android Phones, Tablets, Memory Secondary storage, Display adapter. Peripherals, Platform, APIs and drivers |
| CPU quad core processor, DirectX 10-compatible GPU: GeForce 9800GT 1GB or ATI Radeon HD 4870 1GB, 1 TB storage |

**Learning resources**While developing the Chabot it is important to conduct a lot of research into the fundamentals of human communications

|  |  |
| --- | --- |
| Books | |
| Useful Publications | The studying into the fundamental of A.I  **Real-Time C++** Efficient Object-Oriented and Template Microcontroller Programming, A **Hardback** edition by Christopher Kormanyos  Chatbots: An Introduction And Easy Guide To Making Your Own by Oisin Muldowney |
| Helpful Internet resources | Google Scholar: Chatbot, machine learning, AI research. other research websites like NTU research page(on AI/chatbots). |

**Extra resources**These are extra resources that will be useful in the development of the Chabot. Although they might not be used directly they can act as foundation to help expand further into our development.

|  |  |
| --- | --- |
| Chat Bot framework | A set if ore established conditions and rules that may perform tasks useful for further developments. |
| External services API keys | The use of API keys like amazons shipping details to create features that inform users about their purchase information. |
| Human communication | **Natural Language Processing, Natural Language response** |

**External text to speech programs**We considered a number of external text-to-speech programs in designing our chatbot, and eventually settled on using MaryTTS due to its ease of use and variety of voices.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Features** | **Strengths** | **Weaknesses** |
| Mary1 | Supports a wide range of languages and accents  Both male and female voices  Has toolkits for adding new voices  Software is free use  Can output voice as audio file | Variety of languages, accents and voices can be used to support user customisability  Available toolkits make further modification easy | Some voices come under creative commons  Comparatively large filesize  Problems relating to pronunciation of punctuation (for example, apostrophes in words such as I’m) |
| Java-Google-Text-To-Speech2 | Includes tutorial for how to build own text to speech  Software is free use | Smaller filesize in comparison to other text-to-speech programs  More flexible than other programs | A limited range of voices/accents, making customisation difficult |
| FreeTTS3 | Comes with several different voices of different qualities, with ability to import several more  Includes tools to add or import more voices  Capable of using diphone synthesis for better sounds  Can output in a variety of audio file types | Diphone synthesised voices can sound more natural to the user | Requires a specific Java edition  Better quality voices are limited domain  No real variety in voices or accents |
| Festival4 | Site includes a variety of different voices.  Comes with comprehensive manual  Includes tools for adding new voices. | Designed for use as a component of larger systems  Variety of importable voices makes it easier for user to customise experience | Written in C++, would require integration with java  Standard voices are largely limited to US male |

## 6.2 Risk assessment table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Risk event description | Impact Rating | Probability Rating | Risk Score | Risk Response Description | Trigger |
| 1 | Group members cannot work on project for a period of time due to illness or injury | High/ Medium | Medium/ Low | 2 | Ensure group members can help each other | Illness, injury and sickness |
| 2 | Team members not being able available to work and meet at the same time | High/ Medium | High/ Medium | 3 | Create a timetable for meetings to ensure all members can be able to attend | Schedule and other responsibilities |
| 3 | Conflicts within the group | Medium | Medium/ Low | 6 | Not only support own ideas but also others | Disagreements and different views |
| 4 | Team members finding module topics/ concepts difficult to grasp | High/ Medium | Medium | 5 | Help with each other in group with their knowledge | Lack of communication, attendance and seeking help |
| 5 | ICT resources may not be adequate or appropriately available for demands of the project | Medium/ Low | High/ Medium | 7 | Find information from professional website, then discuss with other members see if it is suitable | Bad luck, no access to library and bad system planning |
| 6 | Loss of project data and files | High | Medium | 1 | Use devices to backup, namely Cloud service or USB Stick | Carelessness lack of multiple backups |
| 7 | Loss of Group member | High | Low | 4 | Respects and discuss with group members before making decisions | Conflicts, personal issues |
| 8 | Unable to work with team members in non-term time | Low | Low | 3 | Make sure things are covered during term time or online communication. | Term finishes |
| 9 | Loss of Work | High | High | 2 | Need a back up  Increase of lost work. | Carelessness, no responsibility. |
| 10 | Living areas are different | High | High | 2 | Each member in the group stays far away from each other. Meeting cannot | School Logistics |

# **7. Programming languages.**

C++

Being the common programming language learnt by every member in our group, C++ built the foundation for our basic understand of how the final product could look like. After briefly designing the structure of the code, acquiring information on some aspects of the program (e.g. attempting to create an automatic learner class) and even creating the initial greeting phase of the program a massive realization came upon us – creating our chatbot using the language is not only rather complicated but also raises concerns when it comes to cross-platform compatibility and Graphical User Interface design. While there are numerous examples of C++ based chatbots available on the web (e.g. *A learning chatterbot in C++* by user *EngyFun* on *instructables.com*, *Chatbot Tutorial* by *Gonzales Cenelia* on *codeproject.com*, *A.I Program – Chatterbot* by *Gonzales Cenelia* on *PlanetSourceCode.com*, etc.) that prove such a project could be completed using this language and acted as references for the initial start of our project, we as a group decided to swap programming languages not only just to show originality in our work but also in taking up the challenge of learning a new programming language.



**Figure 3.1: The initial greeting phase of our program using C++**

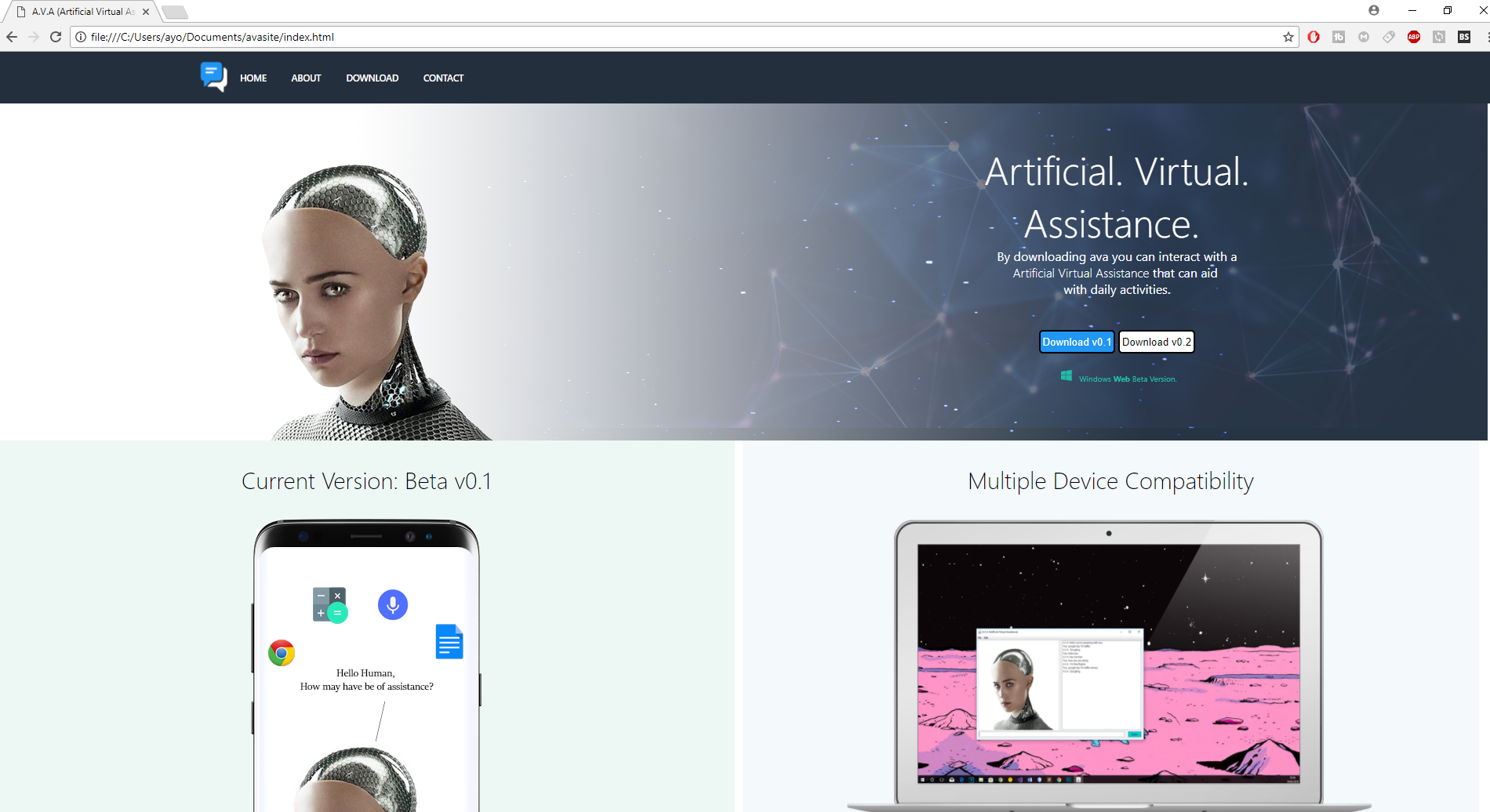
JavaScript

A minor addition to the amount of new languages/ skills acquired, JavaScript was used for our webpage design which has been planned to be the downloadable source location for new users. JavaScript was used to develop the slideshow for the website in increase the aesthetic appearance to ensure it attracts users.

The slideshow consists of variables that stores integers and strings in arrays that hold the position of both the caption and image files. A set image function was used to run the array to select the position of each image in the root folder, storing and then displaying it in a HTML tag. A similar system was used to create the thumbnails by creating a variable that changed the value of the HTML tag while being clicked. To ensure the slider ran automatically a set interval function to move between the images. At first this was hard because the variables wouldn’t display the images in HTML or it would move to the wrong position in the array. Furthermore the slideshow includes features to allow users to display the first slide, last slide or a random slide by using the “math.random()” function.

HTML & CSS

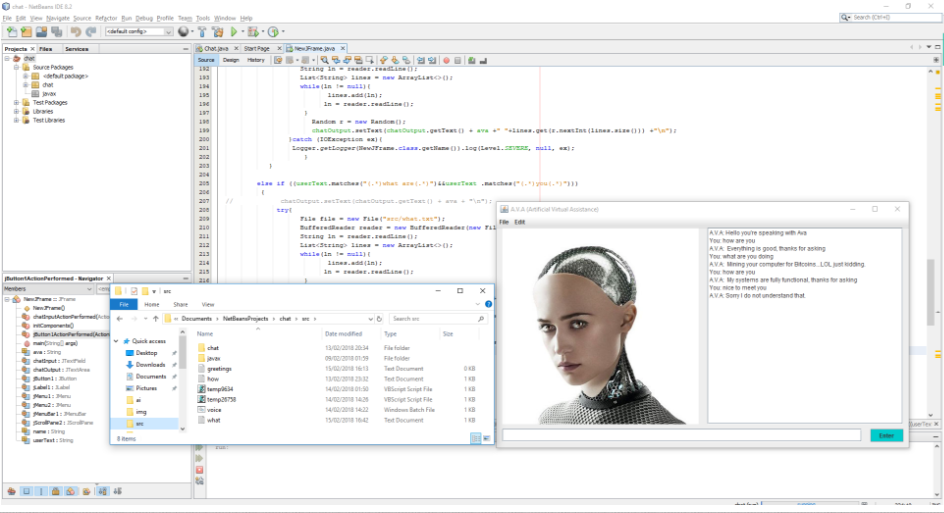
The basis of the marketing phase for our project, and a language already known by most of the group members prior to the project. HTML was the foundation of our webpage and serves as an opportunity to entice and attract users to try out our program. The availability of having a website opens the possibility of having an online web base chatbot to go with the downloadable version. This could act like a demo allowing users to get an idea of the chats functionality. Furthermore, as software developers this allows us to have a well structure portfolio displaying our development and research. In addition, within a business scenario this could serve to sell our software to customers they are interested in buying a bespoke version of it.



**Figure 3.2: The current layout of our website**

Java

The wildcard of our entire operation, and also the saving grace of our program. Java replaces the role of C++ in terms of the development and formatting of our program. Besides being much more manageable to code with compared to its predecessor in our project the ability to create our chatbot in *NetBeans* means having the ability to create simple Graphical User Interfaces with the assistance of *JFrame* containers. Similar to other diagram-drawing applications such as *Microsoft Visio*, *NetBeans’* built in GUI creator comes fully prepared with equipment such as Text Boxes, labels and simple shapes which can be altered slightly to suit our needs. It allowed us as programmers to have the direct ability to gain information in terms of the users’ perspective of our program.

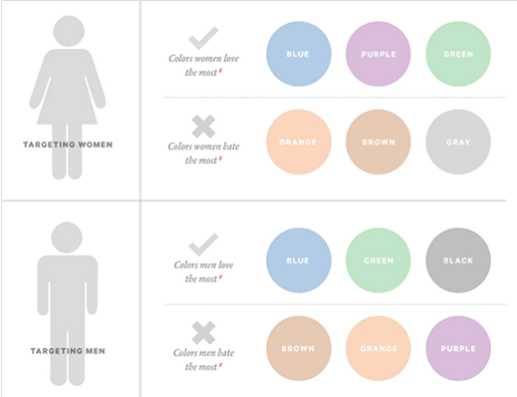


**Figure 3.3: An insight into the progress of our chatbot.**

# **8. Design**

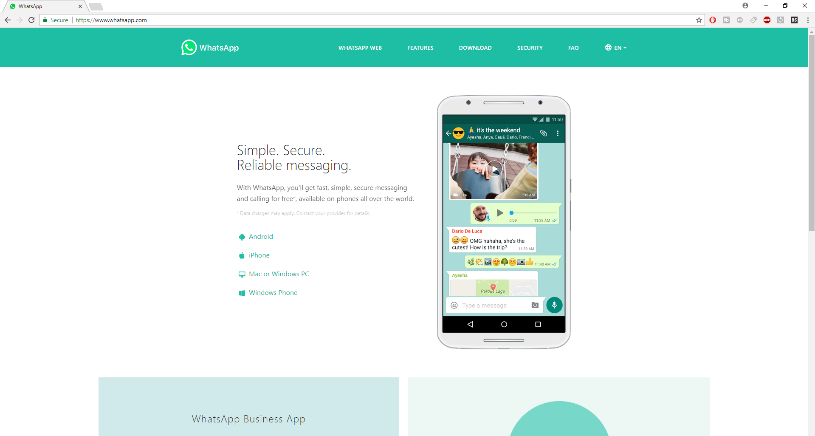
**Colour Scheme**

The colour scheme and aesthetics of both the website and software is very important as there are key factors in attracting users. Furthermore, in a very competitive computer age software appearance is very important to ensure the software developed is distinctive and unique from its competitors.

This image is based of a research conducted by kiss metrics[reference] into how colours affect conversion for both men and women. The conclusion they made was that one average men and women are both attracted to a combination of the colour blue. In addition, both men and women on average recognise green as a conforming colour. The relation between green and conform could be due to nature (trees and grass).

By developing a software that incorporates the colour scheme that’s most suited to the average user the software will be catering to the needs of the general user. Visually the software will be attractive, and conforming has it maintains the nature colour scheme of blues and green.

**Figure 4.1: Colours that affect conversions.**



The appearance of the software and the development style is very important has it’s what determines whether new users download or purchase the software and whether existing users will continue to use the software.

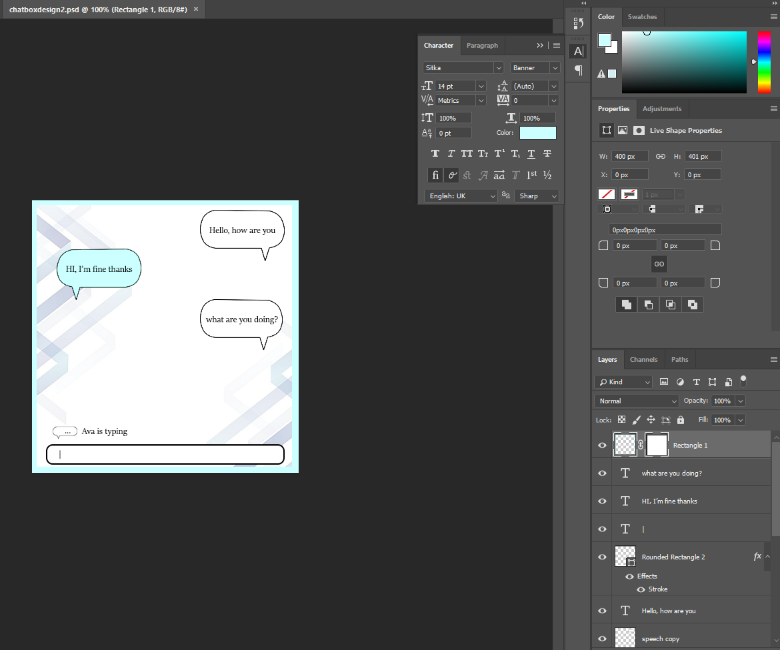
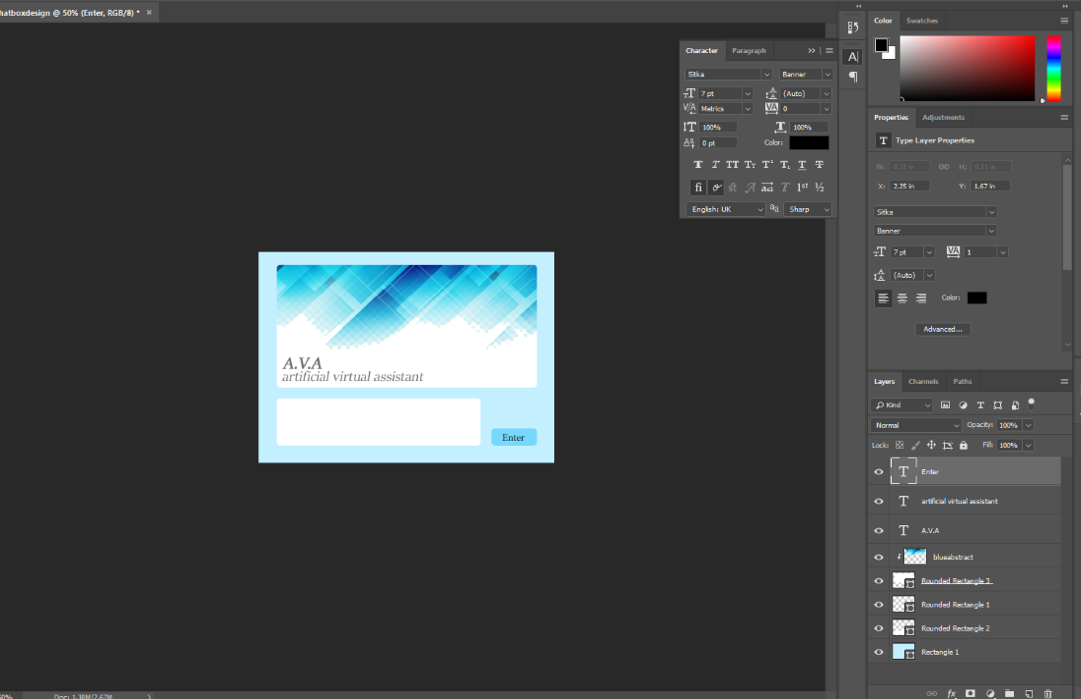
In addition, the visuals of the software are important especially for users who may require visual aid. While developing the software its crucial to stay away from high contrast colours or using a mixture of colours that clash which may prevent the visibility of certain features of the software.

**Figure 4.2:Whatsapp visuals and colour scheme**

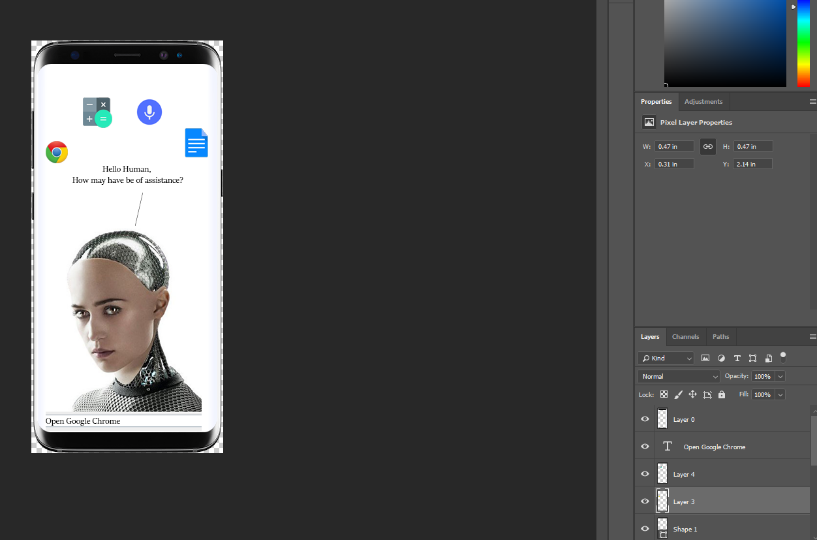
Although the software has text to speech output majority of the conversation conducted would be in text format. Therefore, it’s important to ensure that the readability of the software’s conversation is optimal. Figure 2.2 shows 2 examples of readable and un-readable information. These images are important to indicate good use of text colour and background colour. It also displays the use of bad text colour and bad background colour mixtures that shouldn’t be used.

**Figure 4.3: Text colour and background colour**

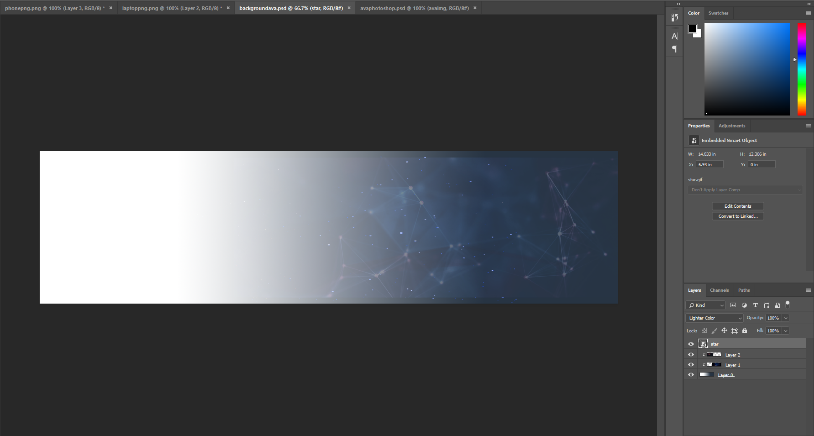
## 8.1 Photoshop Designs



**Figure 4.5: Initial concept of user interface layout**



**Figure 4.4: Original design of greeting page**



**Figure 4.7: Design progress for website background**

**Figure 4.6: Progress in creating suitable materials for website**

### 8.2 Chatbot Design 1 – iMessenger inspired design

This design was inspired by the iMessenger/whatsapp style of messenger applications. It’s simple and uses labels that are displayed as speech bubbles to indecate the userinput and chatbot response. It has an animated background of abstract lines that slides up and down the inner-frame of the chat display to give it a retro stimulated design.

A background will be an animated abstract line that moves up and down the background of the display to increase the aesthetic appearances of the chatbot.

A timer delay will be used to make the response seem realistic and not instant/robot like. A while loop will be used to display an animated effect that indicates “Ava is typing” to make the user experience seem human like.

A messenger-based style that uses speech bubbles as output on the main display of the GUI. Users can type their messages in the textbox located at the bottom of the inner frame.

|  |
| --- |
| **Figure 5.0: iMessenger inspired chat design** |

Different colour schemes that could potentially be used or made available if the users wanted to change the style of the chat bot.

**Pros and Cons**

The chat is simple and easy, making it accessible to even the “laziest user”. It has a familiar style that other similar software have used. Although the features of the chatbot discussed within the above design are good developments that should be incorporated into the final software, the style and appearance seem too simple and lacks anything unique that would make the chat design an improvement of its counterparts.

A function that holds the possibility of changing different styles and design of the chatbot to suit the user’s preference. This will be an advanced feature that will incorporate the development of multiple user interface and changes to their appearance (colour).

The textbox-input detects user-input and identifies certain phrases, then it uses an ‘if’ statement to check for the appropriate response to the user input. It returns an ‘else’ if the users input is invalid. If multiple invalid user-inputs are detected it gives the user access to its help function.

|  |
| --- |
| **Figure 5.1: iMessenger inspired chat design** |

### 8.3 Chatbot design 2 – Web-chat inspired designed

This design was inspired by basic web-based chat boxes like Xatchat and MSM. It displays the users text with an icon of the profile picture next to it. The colour scheme maintains the aquatic blue as the colours “aqua and sky blue” are associated with calm environments.

Further development could include users being able to ask the chatbot to open specific applications. An example would be “Open word doc ‘C.V’” into the textbox-user input the chat bot will be able to detect certain phrases then return a function that runs the window application by replacing a null String value with the name of the file from the user input

Will use voice response by converting the output text input “.bat” (batch files) that output text to speech. Further develop may allow the user to change the voice of the response and include the ability to replay the chat output.

**Pros and Cons**

This design is an improvement from the first design as it can use text to speech and can open programs using certain commands. It uses a retro MSN messenger type design that’s very accessible by new users. Furthermore, it has a log in feature that allows current users to log into their account, this function will make retrieving past conversation possible. It lacks any unique styles but makes up for it by providing better user interaction with voice chat and stored conversations.

The chatbot will use a function that can open programs within the users operating system. An example would be a “if statement” that “throws and catches” data. For example, on a windows operating system if the user was to type “open word” The chat bot will open the word application.

“Enter” button will send information from the text box which will trigger a while loop to enable the bot to detect keywords from the user and return with a reasonable response.

A login screen will be used to collect information of new users and restore past conversations of old users. This will store data regarding that user into a database.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task Name | Duration | Start | Finish | Predecessors |
| Project Definition | 27 days | Fri 01/12/17 | Mon 08/01/18 |  |
| Basic Question Answer system | 7 days | Thu 11/01/18 | Fri 19/01/18 | 1FS+2 days |
| 'Learner Class" Fully programmed | 20 days | Mon 29/01/18 | Fri 23/02/18 | 2FS+5 days |
| User-Interface Designed | 6 days | Mon 22/01/18 | Mon 29/01/18 | 2 |
| User-Interface developed | 17 days | Tue 30/01/18 | Wed 21/02/18 | 4 |
| Program completed | 11 days | Mon 26/02/18 | Mon 12/03/18 | 3,5 |
| Debugging phase-1 | 7 days | Sat 17/03/18 | Mon 26/03/18 |  |
| Debugging phase-2 | 6 days | Mon 26/03/18 | Mon 02/04/18 |  |
| Project Completed | 9 days | Wed 04/04/18 | Mon 16/04/18 |  |

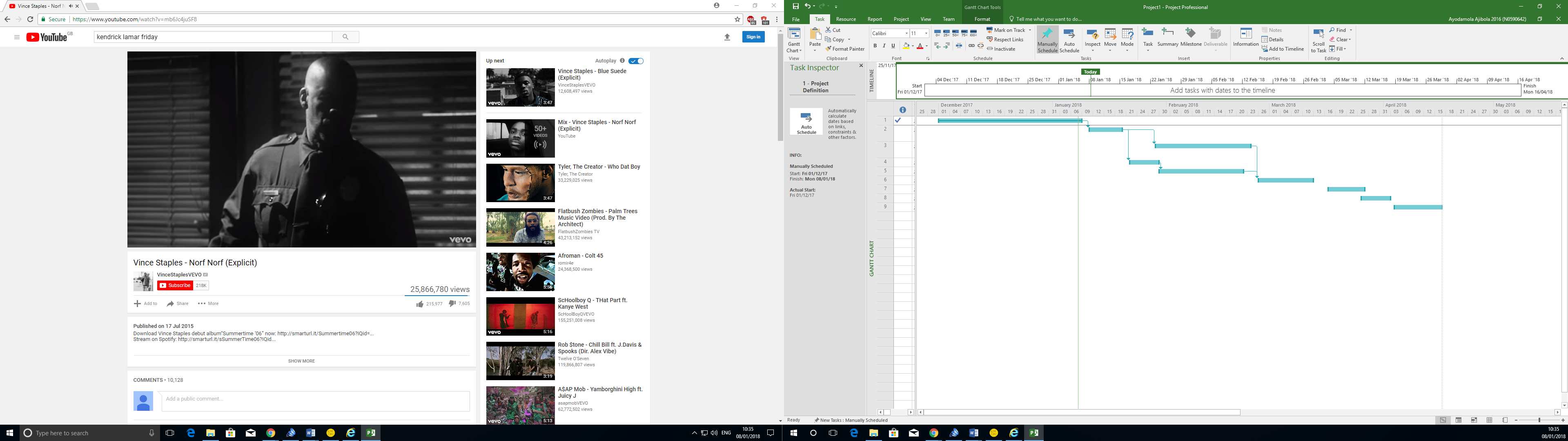
# 9. Project Plan, milestones, effort & timescale for whole project

This section consists of personal deadlines that were set to ensure the module hand in date were met with all tasks completed**.**

A rough layout of deadlines to meet has been setup which takes into consideration scheduled exams and the difficulty of various aspects of the project. It is important to note however that this table will eventually be altered based on various variables, but one thing will always remain – the expected finishing date of the project. The provided peer-assessment sheets at the end of everything will provide a breakdown of each week’s accomplishments with regards to the deadlines set out by the milestones established below.

*work. Consider whether you need to allow “contingency time” for the risks you have identified. State what you expect to have achieved by the Interim Review stage*

**Figure 6.0: A table indicating the initial deadlines set**



**Figure 6.1: A Gantt Chart showing the progress of the development of the program**

### 9.1 SDLC Software Selection

When looking into the organization of tracking the project development two pieces of software stood out the most – *Pivotal Tracker* & *Trello*. Both had their own advantages and disadvantages, but both followed the golden rule of using an Agile Practice model to track progress. While *Trello* uses a ‘Sticker-style’ setup to layout goals and objectives; *Pivotal Tracker* utilizes a ‘Storyboard’ method to showcase user interaction. The reason for *Pivotal Tracker* being our software of choice is solely due to its storyboard-styled management system which allows for a much clearer insight into the supposed workings of the program in the hands of users. On top of that the program also provides a rigid structure for both the documentation side and the coding side of our group to be able to share information and work in parallel without any major disruption since some of the group members under the documentation side could review the codes and react accordingly with the help of the storyboard laying out the concept actions that are to be taken. While it can be argued that *Trello* should be better due to its more generous offerings in terms of features towards free users compared to its rival *Pivotal Tracker* has a much simpler and smoother system to operate, along with the fact that not many group members are required to use the software anyways.

## 9.2 Function Tracker Layout:

|  |  |
| --- | --- |
| **Stage 1 – Basic Input: (Average Difficulty Easy)** |  |
| * Users should be able to enter alphabetical, numeric and certain special characters into the chatbot. | ✔ |
| * The program should be able to provide evidence to repeating users with regards to their last encounter (e.g. sharing a secret that the user wishes the program to keep) |  |
| * Users should be able to freely insert capital letters and ‘spaces’, and still be recognized by the program. | ✔ |
| * Users should be able to submit their responses after clicking on the ‘send’ button or by pressing enter. | ✔ |

|  |  |
| --- | --- |
| **Stage 2 – Processing:** |  |
| * Chatbot should be able to learn new variations in sentences (e.g. “I’m doing great…” vs. “I’m having a jolly-good time…” etc.) | ✔ |
| * Chatbot should compare users’ inputs to a database to provide users with set responses where appropriate. | ✔ |
| * Chatbot should never be able to quit the program through conversation, only through the closing of the window or ‘Alt-F4’. | ✔ |
| * Chatbot’s code should be neatly organized following the principles of object-orientation programming. | ✔ |
| * Users should not be able to access or alter any parts of the program, and should just be restricted to the chat feature itself. | ✔ |
| * Chatbot should be able to hold names and other minor essential information into memory. |  |
| * The chatbot must be able to provide various styles of responses to set inputs (e.g. responding by saying “How are you?” or “What is your mood like today?”) | ✔ |

|  |  |
| --- | --- |
| **Stage 3 – Replies:** |  |
| * The chatbot should be able to provide basic replies to introductory sentences (e.g. Hi there.) | ✔ |
| * The program will be able to provide appropriate responses in relation to the user’s sentence through emotion-like recognition. | ✔ |
| * The chatbot may be capable of providing users with useful links to various sources depending on their intentions. |  |
| * Users should not be able to obtain factual information from the chatbot, to differentiate the program from personal assistants. |  |
| * Users should be able to have open, friendly conversations. | ✔ |
| * Chatbot should provide users with open-ended conversational questions and responses. | ✔ |
| * The program should not be able to provide vulgar responses. | ✔ |
| * The program should be able to ask for a clarification or repetition if a users’ input cannot be understood. | ✔ |

|  |  |
| --- | --- |
| **Stage 4 – User Interface:** |  |
| * The program should have a minimalistic looking interface. | ✔ |
| * The program should include a chat log, text input bar and an ‘enter’ button. | ✔ |
| * The program may include a slideshow of aesthetically pleasing images as the background. |  |
| * The user should be able to change the background image/ slideshow where necessary. |  |

## Extras:

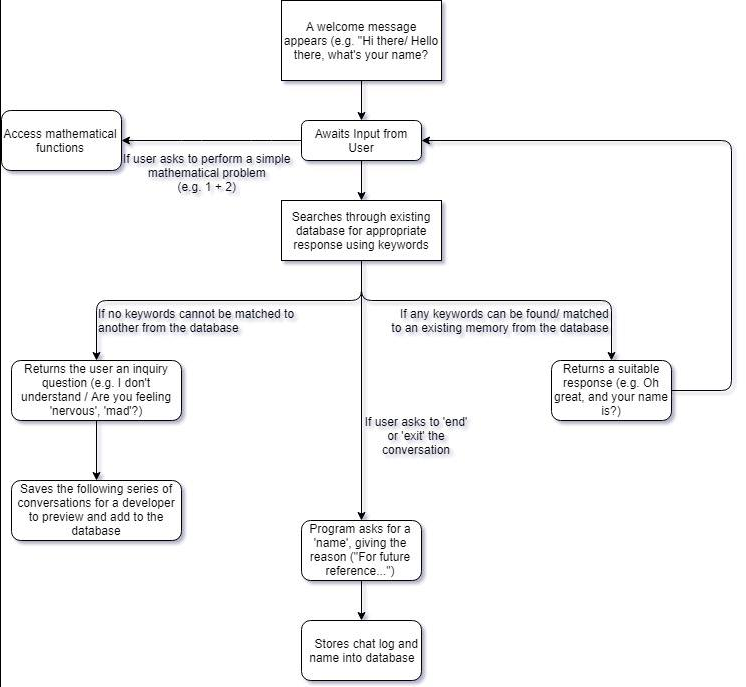
|  |  |
| --- | --- |
| **Profile Management:** |  |
| * Chatbot should be able to retrieve the names of previous users stored in a memory storage. |  |
| * Chatbot should be able to simply notify the user if they have used the program previously. |  |
| * Users’ chat history does not need to be stored in memory, to maintain security and protection of the user. | ✔ |

|  |  |
| --- | --- |
| **Emergency Services Links:** |  |
| * Chatbot may have a small database of useful hotline numbers/ contact numbers. |  |
| * Chatbot should be able to detect signs of distress, and provide emergency contact details where necessary (e.g. providing the user with the suicide prevention hotline number where sentences such as “I really have nothing else to live for…” as given). |  |
| * Users should be able to share anything with the program and feel safe and secured. | ✔ |

|  |  |
| --- | --- |
| **Mini-response databank:** |  |
| * The program may contain an array that stores certain important inputs from the user. |  |
| * Chatbot should be able to match certain user inputs to their appropriate names. |  |
| * Administrators should be able to access these information, but will have their name and time of access recorded. |  |
| * Users should not be able to access these specific responses by accessing their profile using their full-names |  |

|  |  |
| --- | --- |
| **Text-to-speech:** |  |
| * The program may be able to convert audio recordings into appropriate sentences. |  |
| * The program should be able to ask for clarification or repetition if a proper sentence is not constructed. | ✔ |
| * Users should be able to freely use the text-to-speech function, no matter their accents. | ✔ |
| * The program should be able to provide audio responses to the users. | ✔ |
| * The program should be able to alter between different voice packages upon the users’ request. | ✔ |

|  |  |
| --- | --- |
| **Device Compatibility:** |  |
| * The program should exist as a folder that may be transferred and used on other operating systems (e.g. Android or Macintosh) |  |
| * A website that can be accessed on multiple devices to enable users to access the site. | ✔ |



**Figure 7.0: A basic flowchart showcasing a step-by-step guide of the program functions in response to certain input**

# **10. Development**

## 10.1 Visual Studios and C++

While conducting research into methods of developing a chatbot Visual studios and C++ were used to develop a simple text-based bot that responses to specific array of strings. This was a simple application that detected specific sentences that were stored in arrays, but it can’t understand anything beyond what’s been specified in the string. Developing this application was a great start to get an insight into how a chat bot application will function and it provided enough basic challenges to push development to further understanding of requirements needed to complete a chatbot software.



**Figure 8.0: C++ code**

The early stages of this development used a struct to determine the layout of how the array will hold expected string inputs and a range of possible outputs generated randomly using the “math.random();” function. It’s a very simple method of developing a program that respond to specific phrases, but it lacks the ability to deal with integers and the intelligence to understand between specific sentences and identify words. Furthermore, the reason why it isn’t very intelligent is that users are required to input only the array of strings that have been stored as expected input before it gives an expected output. An example would be it accepts “hello” but not “hello, nice to meet you”. The difference between this version of the chatbot and the improved version is that instead of focusing on specific sentences it identifies a range of phrases then determines whether it’s a greeting or a question then response appropriately.

In addition, while developing the chatbot with visual studios using C++ there were a lot of Issues with visual studio forms. The issues that occurred were such things like internal compiler errors while trying to run the form without any code being added, windows run time error after typing into a textbox and other issues that would require time spent on finding solutions to solve than developing the chatbot itself.

These issues were mostly since Microsoft visual studios are pushing the use of VB.Net and completely ignoring C++ GUI development by not providing enough solutions to internal C++ GUI issues.

## 10.2 Website Development in HTML/CSS

|  |
| --- |
|  |
| **Figure 8.1: Website changes**  To ensure the project is professional a website was developed to act as an access point to the software. The benefits of having a website is as software developers it enables us to display the best features of the software and include information about the research which could aid any further chatbot developments. The website consists of a home page, about page, download page and contact page. The home page is a visually appealing section of the website that tries to catch the attention of new users by using design techniques to ensure its aesthetic enough to keep viewers. It also shows snippets of how the software look and provides download links to the latest version of the software. Furthermore, the website maintains a consistent style to match the styles of the website and software making it look professional. |

### 10.2.1 Coding design

|  |
| --- |
| **Figure 8.2: Website codes** |
|  |
| The website was developed using HTML and CSS. It consists of a navigation bar located at the top of the page with links to other pages within the website (home, about, download, contact), the logo of the software floating left within the navigation Div Tag. Everything within the navigation is held using unordered list and ordered lists within the HTML and style with CSS to align them in a box format floating left and with a padding that’s 25%. The second section of the website consists of container that displays a gradient background transitioning from white to blue with a fixed position background of stars and neutrons. Also, within the container there are two classes one that located on the left side of the page holding the avatar for the chatbot and the other displaying some text and buttons that link directly to the download for the chatbot. 10.2.2, Use case diagrams https://scontent-lhr3-1.xx.fbcdn.net/v/t1.15752-9/31948502_1618137411572609_7566203208280834048_n.jpg?_nc_cat=0&oh=0422b21bb9cac8bfc92ff4738cc3c85d&oe=5B555CD9  Figure 10.2.2 Use case diagrams |

## 10.2.3, Concept map

## 

## 

## 

Figure 10.2.3 Concept map

## 10.3 Software Development in Java

Originally the development of the chat bot was meant to have been done in C++ but due to a lot of issues with visual studios and the lack of cross platform compatibility while trying to develop the GUI

|  |
| --- |
| **Figure 8.3: Current chatbot layout** |
| The image above shows the use of the chat bot ava, a user interface has been developed that can understand basic inputs and gives appropriate response. The GUI consists of a title that shows the name of the software. It also has a tool bar that will have future sections that provide help, an edit bar that lets the user pick from a random of possible designs, a login section and a link to the website. Furthermore, the mid-section of the software GUI has an avatar that with future development will be animated to speak while the voice function is true. Also, within the mid-section on the right side is the main chat area that display the chat log, this is where the user and the chatbots output will be displayed. |

### 10.3.1 Coding design

|  |
| --- |
| **Figure 8.4: Codes for storing strings into text files** |
| The image above shows the process in method of the develop code. On the left shows how, the software used to use basic arrays to store strings by detecting a specific array of strings. This was a simple method that helped expand on the knowledge required to make the chat bot deal with more complex methods of handling input and how it should retrieve possible output i.e. using text files “.txt” to store and retrieve information. |

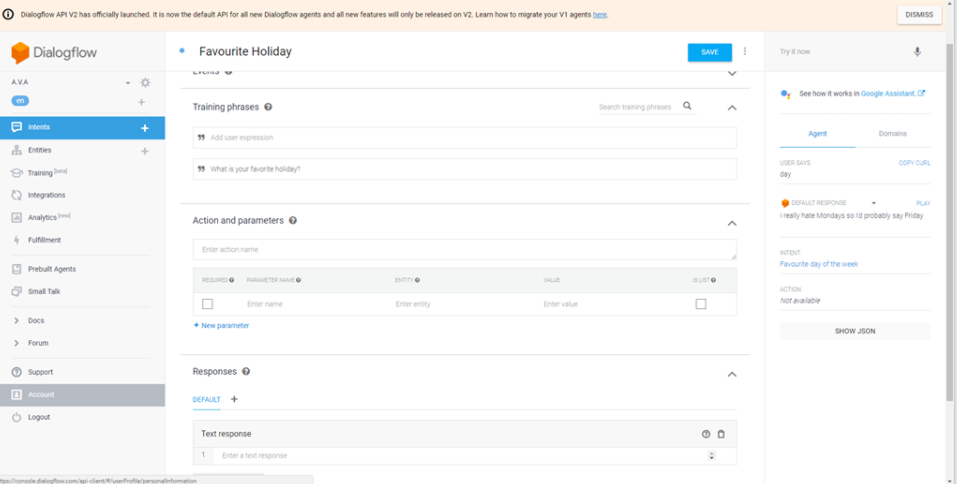
## 10.4 Current program Designs

Information in this section consists of the current software design and the programming methods used to develop specific features for the software.

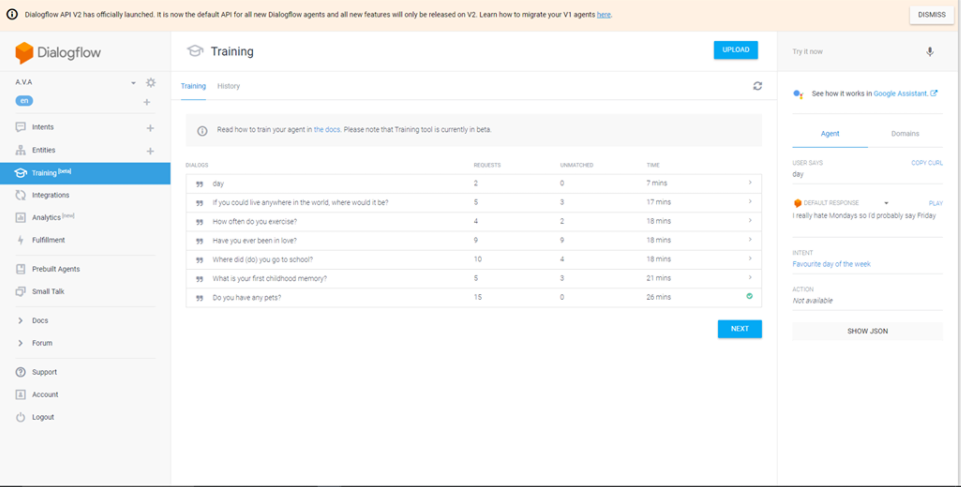
|  |  |  |
| --- | --- | --- |
| **Figure 9.0: Location of text files**    **Figure 9.1: Content of “what” text file**    **Figure 9.2: Java Libraries** | | **Figure 9.3: Java GUI code**    **Figure 9.4: Java code for user-input processing and answer generation** |
| This section of the code handles the file handling of possible user input and possible chatbot response. The user input is detected using an “if statement” that identifies whether a certain set of words or phrases have been said. If the user inputs a phrase for example “hello how are you”, an if statement is running to see if there’s a combination of those phrases once found a text file is opened that holds possible response then selects them using an array that randomly picks a response. That response is then displayed on chat text field. Otherwise if the user input can’t been found it returns an else statement specifying it doesn’t understand the user. An improvement to this would be to develop a learner classes that uses some sort of machine learning to be able to store new sequences of words. | | |
| **Figure 9.5: A.V.A. capable of opening internet browsers to search for information on various topics** | **Figure 9.6: Java code which allows for internet browser searching mechanic** | |
| This section of the code deals with the opening of other programs on the user’s system and does google searches by uses keywords like “google” followed by the user’s request. This was developed using a try and catch, the try outputs a message informing the user what the chat bots doing then depending on the request it runs an open function that open the programs file from the directory or does a google search if the “google” keyword is detected. | | |

## 10.5 DialogFlow

To help with the development of the learner class, research had begun into APIs (Application Programming Interface). In basic terms this application contains a set of existing definitions, protocols or even tools which can be used to build application softwares. For our chatbot we looked into the website *DialogFlow*, which allowed us to anticipate a certain amount of possible user inputs and generate certain set amount of responses based on a feature known as ‘intent’. Intent is similar to that of a class in programming, where this structure holds information on a specific topic. Whenever a new user input is detected *DialogFlow* stores the user input for the programmers to see. The programmers can either proceed to assign the specified input to an existing ‘intent’, or create a new one in which a new set of possible responses are stored. It is important to note that each intent may store several responses for multiple user inputs, just like a class or structure in programming is capable of storing several related variables within it.



**Figure 10.1: Screenshot of DialogFlow**



**Figure 10.2: A.VA API development and training**

# 11. Testing

## 11.1 Alpha/Beta Testing

Upon completion of the program, several users were given the opportunity to become alpha testers, helping out in the testing on our final product. These users range from online strangers to real life acquaintances. The results gathered from some of the volunteers selected are as follows:

|  |  |  |
| --- | --- | --- |
| **Name** | **Result** | **Evaluation** |
| Jin Wei OOI | Found it to be an interesting application. However it was difficult to install and get working. | Having coded the program in a ‘*Java*’ friendly environment the concept of ‘easy installation’ had not been thoroughly considered. |
| Clarence SIEW | Good attempt at creating a basic chat program. Not easy getting it to run smoothly and not all conversations lead to proper answers | Being a ‘concept’ of what a future version of this program might appear like, the current version only has a limited amount of registered state-and-respond circumstances |
| Niklas HALM | Conversations are very similar to human beings, just wish some of the voices were more realistic | With time an increase in better virtual voice programs will be added to the system. |
| Mahmoud AL-HASSAN | An interesting application. I am able to talk with the chatbot and even find out simple information. | Evident that features such as being able to perform automated internet searches are fully functional. |
| Setbyul KANG | I can talk to it. Simple dialogues work. Very difficult to get working. | Numerous testing results have shown that more effort is needed in creating a solution to easily install the program on any device. |
| Amy INGLIS | Found it interesting to be able to talk to such a human-like program. | With limited feedback the only information able to be retrieved from this party indicates that the main function works. |
| Ashley WONG | Simple and clean design. Main picture of robots might not suit everyone. | Instead of just including images of intimidating artificial intelligence possibly allow users to upload their own pictures or include friendly pictures. |
| Alessandro ZAMPRONI | Talking to this thing is very fun. The addition of being able to check for the weather and the date is especially good. However it is hard to get it to work from download. | The following party provides a rather clear-cut portrayal of what the overall program shines in and lacks in. |

### 11.1.1 Pros and Cons

Overall from the total amount of testers the following can be concluded:

* The Chatbot works as intended, allowing users to have conversations with the program.
* The program is not like most of its competitors as it features effective features to make dialogues seem more human-like.
* The design of the user-interface is rather controversial – some adore its futuristic, spotless appearance while some would prefer a friendlier look.
* One major flaw our product has is its difficult installation process for users.

Although the sample we had provided us with little-to-no feedback regarding the extra features added to the chat bot (e.g. open-ended responses, lack of vulgar terms, etc.) it is clear that the main goal of the product has been achieved – create a program that would allow users to have simple and fully interactive conversations that reflect the traits of common human conversations.

It should be noted that while alpha testing (a variant of Black-Box Testing) presented us with sufficient feedback to draw valid reflective information regarding our program it is possible that perhaps using a different type of testing method may yield different results. One test method that was considered was White Box Testing. This software testing method involved the programmers creating their own codes to test whether the functions in their program run as they should. Our group leaned more towards the Black Box Testing method due to the fact that this method involves testing the usability of the program towards the intended target market while the White Box Testing is simply an internal style of testing, making it rather unrealistic.

## Testing features

|  |  |  |  |
| --- | --- | --- | --- |
| 1.Test | Expected Result | Actual Result | Comment |
| Switching from Female to Male | By clicking on the “male” button Ava should switch from the female version to the male. | Worked as expected | When user clicks on the male button the theme changes to match the his/her choice. |
| //Evidence will be here | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.Test | Expected Result | Actual Result | Comment | Future work |
| Calculating ability  Addition, subtraction, division and multiplication | By entering the key word such as “add, divide” and following the regular expression “key word followed by colon and the numbers”, AVA will be able to output answer to the calculation. | Worked as expected | Calculating correctly when the user follows the specified structure | Can be detecting numbers and calculation symbols so the user gets the answer even without following the specified structure. |
| //Evidence will be here | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.Test | Expected Result | Actual Result | Comment | Future work |
| Learning ability | The user can teach the chat bot so the more it talks with people the more it gets smarter. | Worked as expected | When the user gets a respond such as “sorry I do not understand”, he/she can teach the chat bot so that in the next time they try to say the same thing AVA will be familiar with it. | Can be that AVA does not accept the input and the response until the developer approve it, because the user can teach unacceptable phrases. |
| //Evidence will be here | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.Test | Expected Result | Actual Result | Comment |
| Voice changes | By changing AVA from the female theme to the male one it also changes the voice accordingly. | Worked as expected | AVA has two voices and they changes according to the chosen theme [Male, Female]. |
| //Evidence will be here | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 5.Test | Expected Result | Actual Result | Comment |
| Website | To have modern website where users can find out about AVA and experience it, and to find out about the developers “about us and contact us pages” also to get users in touch so the app can be developed more. | Worked as expected | Having a website is one of the most tools to get people to know about your product and engage with them to find out ideas on how you can develop your app. |
| //Evidence will be her | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 6.Test | Expected Result | Actual Result | Comment |
| Typing feature | The output does not appear immediately after the user input. | … | When the user input something to AVA, it delays for a while then it output the response, so users can feel that they are talking to a human. |
| //Evidence will be here | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 7.Test | Expected Result | Actual Result | Comment |
| Matching inputs | AVA would be able to match the user input to the most possible response that can be the correct response. | Worked as expected | When the user inputs something that AVA does not exactly have it, but it has similar words to it, AVA takes the response for that sentence and output it to the user. |
| //Evidence will be here | | | |

# Initial design

## User manuals

* The users start with registering to Ava app by entering a username and password.
* The app will not accept existing usernames, nor wrong passwords.
* After the user successfully registered they can log in by providing their username and password.
* After a user successfully logged in, Ava main screen appears with the default theme which is the female one.
* Ava theme can be changed by the user choosing to change the theme to male, the voice also will be changed.
* Users can do a google search by asking Ava “google …”, Ava will open the browser and do the search.
* Users can do calculation by asking Ava i.e. “add: x+y”, Ava will prompt the answer and say it.
* Users can open their calculator by asking Ava “open calculator”, and Ava will do so.
* Users can teach Ava by entering a sentence and its response to the teaching area.

# Conclusion & Evaluation

At the end of the day it is evident that A.VA is a working example of possible improvements that can be made to modern day chat programs in order to make them more inviting and also provide users with a reason to use them outside of just an emotional need. Having such an informative and beneficial tool would also encourage users to use our chat program much more often, which might even allow it to become a competitor against personal assistants (e.g. *Amazon’s Alexa*, *Apple’s Siri* and *Samsung’s Bixby*). On top of that every member of this group has developed their teamwork abilities and their knowledge of programming in the form of practicality – knowing when to use which skillset (e.g. the decision to use java for the development of this program instead of *C++*), which would ensure that everyone becomes a beneficial programmer who serves the needs of their community, but also a wonderful role-model for future students in the computing field.

Probably the most challenging part of this project has been the acquiring of knowledge regarding the development of the chatbot’s ability to respond to human conversations - initially the idea was to simply store string of words in a data structure and return them to the user where necessary, eventually as our understanding of the concept of a chatbot improves so does our ability to create a program that delivers more realistic answers to the point that it does not simply just provide straight forward answers to the users, but rather also answer some questions with open-ended answers that encourage an extended duration of conversation while also incorporating humour into these answers. Research done has also pointed out that developers of existing chat programs have been given a helping hand in the form of APIs, which allows us to see the world of programming through a different view in the sense that the existing IT industry is not as crude as it is made out to be. In fact it should be understood if a certain function or concept cannot be developed in a set time. With that being said one glaring improvement that could be made to our program is the creation of artificial intelligence in order to make our program more human-like. Since our final product includes a text-to-speech system and minor visual aspects that seek to improve on the lack of human-like interactions that exist in chat programs on the market it only makes sense to look into creating an intelligent software that could take the place of our API, allowing the chat program to run and develop on its own if given more time and resources. That being said we are aware of the rising concerns over the impact of artificial intelligence in our social lives (e.g. the possibilities of a rogue A.I. as portrayed by numerous fictional films).

# References

* Cenelia, G. (2014). *Chatbot Tutorial - CodeProject*. [online] Codeproject.com. Available at: https://www.codeproject.com/Articles/36106/Chatbot-Tutorial [Accessed 15 Feb. 2018].
* Cenelia, G. (2015). *A.I Program - Chatterbot Eliza v2.0 (c++ eliza)*. [online] PlanetSourceCode.com. Available at: https://www.planet-source-code.com/vb/scripts/ShowCode.asp?lngWId=3&txtCodeId=10353 [Accessed 13 Feb. 2018].
* Instructables.com. (2015). *A Learning Chatterbot in C++*. [online] Available at: http://www.instructables.com/id/A-Learning-Chatterbot-in-C/ [Accessed 14 Feb. 2018].
* Al-Rifaie, M. (2017). AISB - The Society for the Study of Artificial Intelligence and Simulation of Behaviour - Loebner Prize. [online] Aisb.org.uk. Available at: http://www.aisb.org.uk/events/loebner-prize [Accessed 14 Feb 2018].
* Chatbots.org. (Unknown) - Virtual assistants, virtual agents, chat bots, conversational agents, chatterbots, chatbots: examples, companies, news,directory. [online] chatbots.org. Available at: https://www.chatbots.org [Accessed 14 Feb. 2018].
* Belbin.com. (2018). *Belbin | Great teams start with Belbin*. [online] Available at: http://www.belbin.com [Accessed 8 December. 2017].
* Mitsuku.com. (2016). Mitsuku Chatbot. [online] Available at: http://www.mitsuku.com [Accessed 9 December. 2017].
* *Lun, E. (2018).Chatbots.org - Virtual assistants, virtual agents, chat bots, conversational agents, chatterbots, chatbots: examples, companies, news,directory.[online] chatbots.org. Available at:* https://www.chatbots.org [Accessed 9 December. 2017].
* Al-Rifaie, M. (2017). *AISB - The Society for the Study of Artificial Intelligence and Simulation of Behaviour - Loebner Prize*. [online] Aisb.org.uk. Available at: http://www.aisb.org.uk/events/loebner-prize [Accessed 9 December 2017].
* Partridge, D., 1992. Artificial Intelligence and Business Management. 1st ed. USA: Intellect Books. [Accessed 24th December 2017]
* Reddy, T. (2017), *The code of ethics for AI and chatbots that every brand should follow* [online], available at https://www.ibm.com/blogs/watson/2017/10/the-code-of-ethics-for-ai-and-chatbots-that-every-brand-should-follow/ [Accessed 8 Jan. 2018].
* Morgan, B. and High, R. *Ethics and Artificial Intelligence with IBM Watson’s Rob High*, [online], available at http://www.blakemichellemorgan.com/the-modern-customer-podcast/ [Accessed 4 Jan. 2018]
* Perez, C., 2018. The Deep Learning AI Playbook. 1st ed. USA: I.M. Intuition Machine. [Accessed 4 Jan. 2018].
* Apm.org.uk. (2017). *What is agile project management? | APM*. [online] Available at: https://www.apm.org.uk/resources/find-a-resource/agile-project-management / [Accessed 5 Jan. 2018].
* Perez-Marin, D., 2011. Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices. 1st ed. Universidad Autónoma de Madrid: IGI Global. [Accessed 5 January 2018].
* Erwin van Lun. 2018. Papers on Artificial intelligence research focused on chatbots, conversational agents, virtual agents. [ONLINE] Available at: https://www.chatbots.org/papers/. [Accessed 5 January 2018].
* Software Testing Class. (n.d.). What is Alpha and Beta Testing?. [online] Available at: http://www.softwaretestingclass.com/what-is-alpha-and-beta-testing/ [Accessed 26 Apr. 2018].
* Software Testing Fundamentals. (n.d.). Black Box Testing - Software Testing Fundamentals. [online] Available at: http://softwaretestingfundamentals.com/black-box-testing/ [Accessed 26 Apr. 2018].
* Software Testing Fundamentals. (n.d.). White Box Testing - Software Testing Fundamentals. [online] Available at: http://softwaretestingfundamentals.com/white-box-testing/ [Accessed 26 Apr. 2018].

1. MaryTTS. (2018). [online] Available at: https://github.com/marytts/marytts [Accessed 3rd May. 2018].
2. Java-google text-to-speech. (2018). [online] Available at: https://github.com/goxr3plus/Java-Google-Text-To-Speech [Accessed 3rd May. 2018].
3. FreeTTS. (2018). [online] Available at: https://freetts.sourceforge.io/docs/index.php [Accessed 3rd May. 2018].
4. Festival. (2018). [online] Available at: http://festvox.org/festival/index.html [Accessed 3rd May. 2018].

# Appendices

## Meetings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 1 | 27 October 2017 | All members | Meet for the first time since group designations. Exchanged contact details and decided upon project topic. Work assigned for next meeting involves research regarding chat bots in general. | * Complete Belbin Test * Assign Team Roles (e.g. group leader, coders, documentation team, etc.) | 30 minutes |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 2 | 17 November 2017 | All members | Collect and collate research information. Began work on Project Definition document as well. | * Review information gathered on existing products such as *Mitsuku*. * Input Belbin Test results on project definition document | 1 hour |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 3 | 24 November 2017 | All members | Check-up session (ensure all members are on task and have completed a certain amount of work). Assign objectives, deadlines and possible online meeting sessions in order to have Project Definition document completed by 27th December 2017. | * Collect and verify all research performed by each member * Input collected data into Project Definition document * Setup *Facebook Events* for possible online group meetings during winter break. | 1 hour |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 4 | 22 January 2017 | All members | Review upon feedback given by tutor. Coders to begin programming (Ayodomola & Alex). | * Take note of all changes required to be made following Project Definition submission meeting. * Coders distribute tasks among themselves * Documentation team (Jia Ren & Khawla) provide the programming team with key deadlines to meet for version 1 of the program. | 1 hour |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 5 | 5 February 2018 | All members | Review coding progress. Documentation team begins developing report, detailing all progress so far and providing further research support for coders. | * Coders present version 1 of A.VA chat bot. * Documentation team to research possible additional features to be added. * Documentation team notes down coders’ progress in terms of features successfully implemented. * Documentation team also starts looking into agile tracking applications (e.g. *Trello, Pivotal Tracker*, etc.) | 1 hour |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 6 | 19 February 2018 | All members | Continued work on coding. Documentation team discusses and finalises diagrams with rest of the team. | * Documentation team to finalize *Pivotal Tracker*-inspired storyboard diagram with coders. * Coders share further progress on the program with documentation team. | 2 hours |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 7 | 8 March 2018 | All members | Coding workshop. Documentation team are to track coding status. | * Coding team updates documentation team on current status * Coding team to discuss shift in programming language from *C++* to *Java* * Documentation team to discuss information that would be extracted from Project Definition file and used in the final report. * Documentation team to archive change in final report. | 2 Hours (Coding Team: Ayomodola & Alex)  1 Hour (Documentation Team: Khawla & Jia Ren) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 8 | 22 March 2018 | All members | Reflect upon information given by tutor from recent meetings. Coding slows down as nears end while  documentation team continues to track coding progress and also begin work on presentation. | * Coders to discuss possible disagreements with documentation team. * Documentation team starts work on the final report and begin creating the presentation slides. | 1 Hours (Coding Team: Ayomodola & Alex)  2 Hour (Documentation Team: Khawla & Jia Ren) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 9 | 30 March 2018 | All members | Begin to finish up all coding, while continuing progress on creating presentation and filling in report. Assign work and deadlines for after Easter break. | * Programmers finalize their code, and are to begin debugging stage. * Programmers provide documentation team with the entire code for the chat bot. * Documentation team continue filling in the final report and presentation slides. * Assign deadlines to be met for week after Easter break. * Coders to begin creating API using *Dialog flow.* | 2 hours |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 10 | 16 April 2018 | All members | Review all work done so far. Begin organization of report, program and presentation. Began practising for the actual presentation day. | * Present each member’s work done from Easter break. * Programmers to finish debugging, will start deciding on chosen testing method. * Programmers to continue work on the API. * Documentation team to organize the final report and presentation slides. | 2 hours |

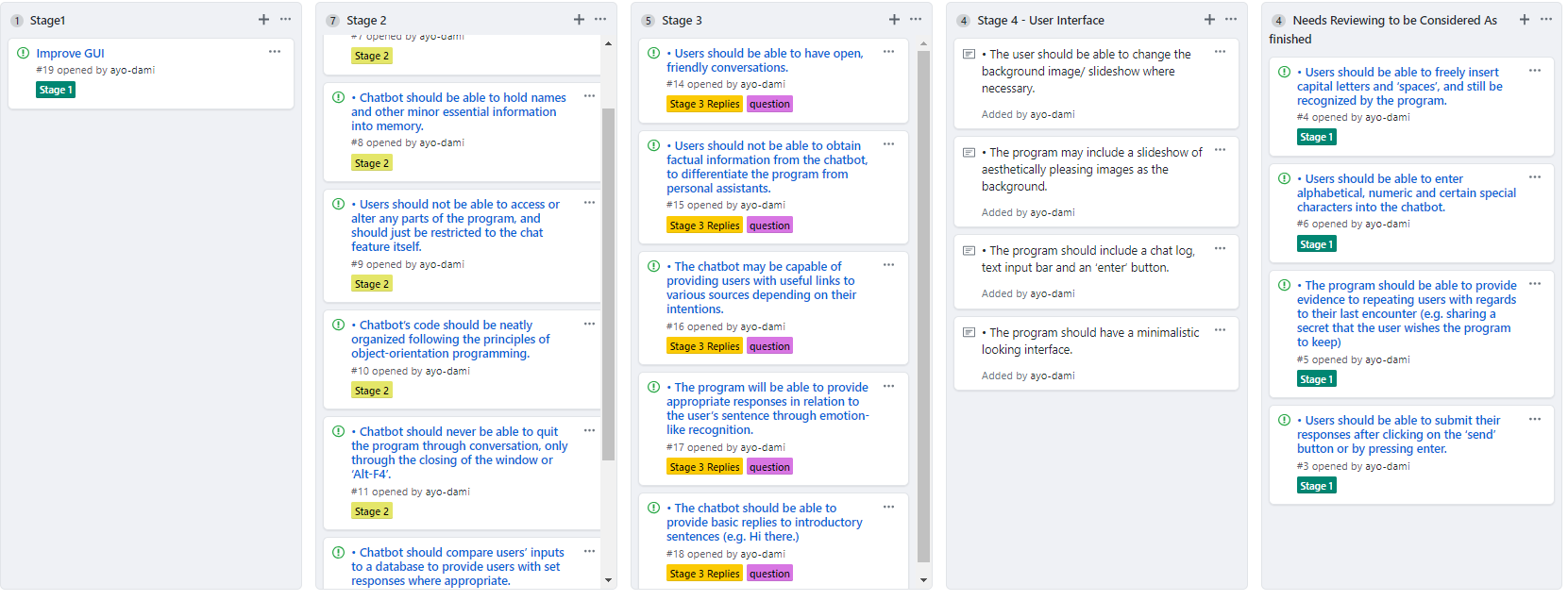
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 11 | 19 April 2018 | All members | Begin testing phase. Report and program is completed. Continue to practice for presentation day. | * Coding team start seeking volunteers for testing phase. * Coders to finish up API. * Documentation team begin proof-reading report. * Documentation team to just finish up presentation slide design and information organization. | 2 hours |

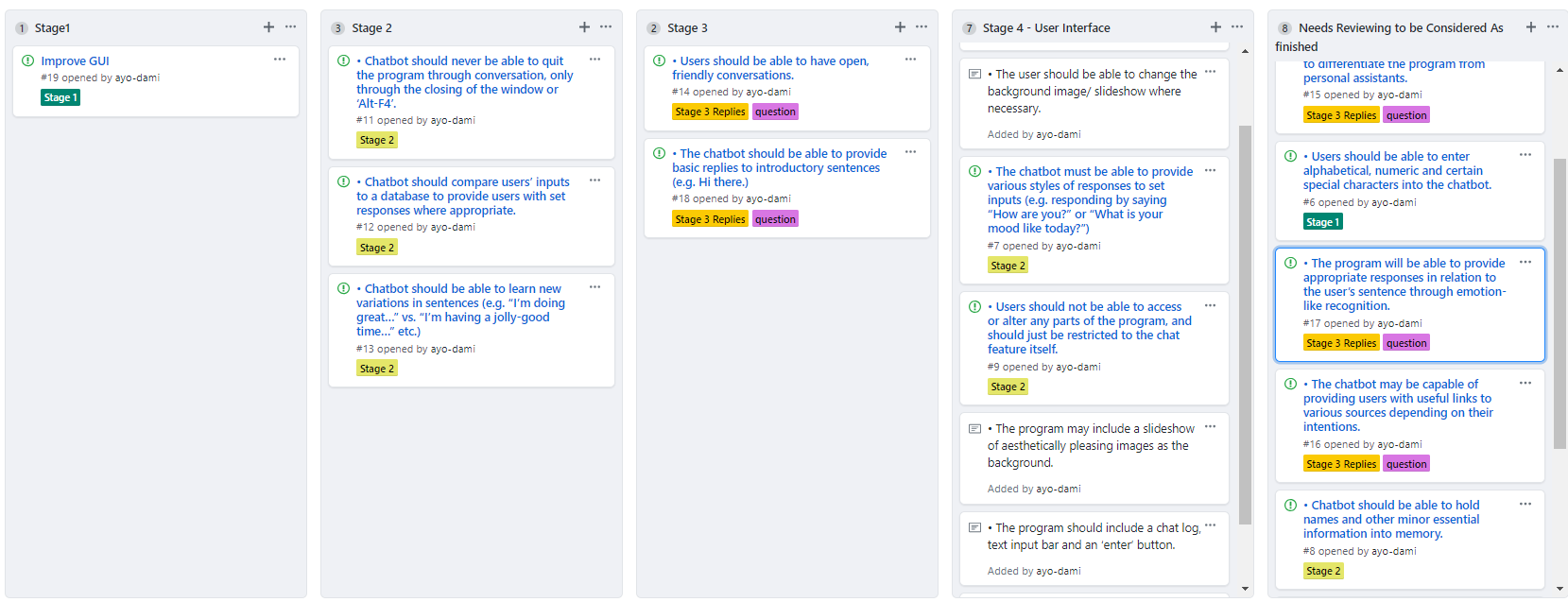
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 12 | 23 April 2018 | All members | Stumble upon problems with testing and presentation slides. Attempt to fix issues while also practising for presentation day. | * Review current test results, and to redo testing. * Set coders stricter deadlines for testing phase to be completed. * Entire team begin practising for presentation day. | 2 hours |

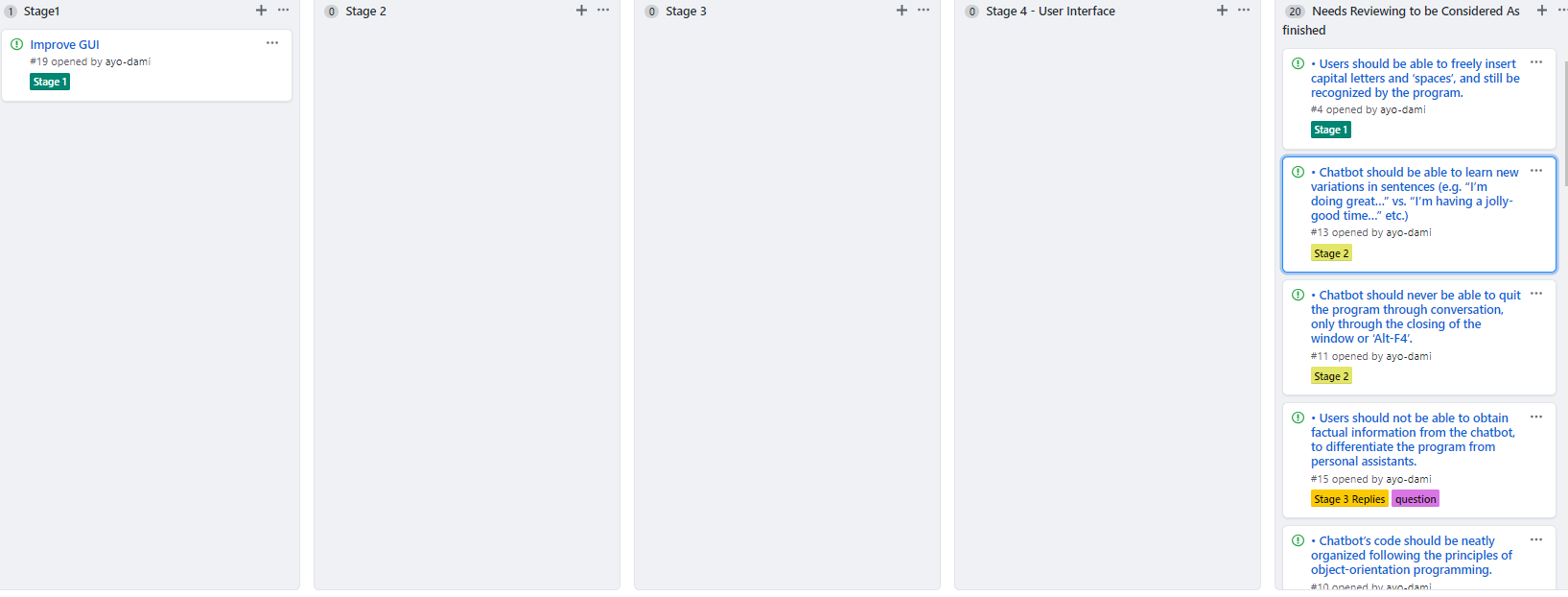
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 13 | 26 April 2018 | All members | Finished testing. Results are to be collected and documented in the final report. Presentation day practice continues. | * Coders provide testing results. * Documentation team to include testing results in the final report. * Entire team continues to practice for presentation day. | 2 hours |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Meeting number** | **Date** | **Attendance** | **Overview** | **Agenda** | **Duration** |
| 14 | 30 April 2018 | All members | Check that all documents are in order. Continue practicing for presentation day. | * Documentation team to ensure final report is perfected. * Practice presentation delivery continues | 2 hours |

## Project Tasks







## Development of the project in the early stages moving to the final one

Early stages of the development

|  |  |
| --- | --- |
| **Stage 1 – Basic Input: (Average Difficulty Easy)** |  |
| * Users should be able to enter alphabetical, numeric and certain special characters into the chatbot. |  |
| * The program should be able to provide evidence to repeating users with regards to their last encounter (e.g. sharing a secret that the user wishes the program to keep) |  |
| * Users should be able to freely insert capital letters and ‘spaces’, and still be recognized by the program. |  |
| * Users should be able to submit their responses after clicking on the ‘send’ button or by pressing enter. |  |

|  |  |
| --- | --- |
| **Stage 2 – Processing:** |  |
| * Chatbot should be able to learn new variations in sentences (e.g. “I’m doing great…” vs. “I’m having a jolly-good time…” etc.) | ✔ |
| * Chatbot should compare users’ inputs to a database to provide users with set responses where appropriate. | ✔ |
| * Chatbot should never be able to quit the program through conversation, only through the closing of the window or ‘Alt-F4’. | ✔ |
| * Chatbot’s code should be neatly organized following the principles of object-orientation programming. | ✔ |
| * Users should not be able to access or alter any parts of the program, and should just be restricted to the chat feature itself. | ✔ |
| * Chatbot should be able to hold names and other minor essential information into memory. |  |
| * The chatbot must be able to provide various styles of responses to set inputs (e.g. responding by saying “How are you?” or “What is your mood like today?”) | ✔ |

|  |  |
| --- | --- |
| **Stage 3 – Replies:** |  |
| * The chatbot should be able to provide basic replies to introductory sentences (e.g. Hi there.) | ✔ |
| * The program will be able to provide appropriate responses in relation to the user’s sentence through emotion-like recognition. | ✔ |
| * The chatbot may be capable of providing users with useful links to various sources depending on their intentions. |  |
| * Users should not be able to obtain factual information from the chatbot, to differentiate the program from personal assistants. |  |
| * Users should be able to have open, friendly conversations. | ✔ |
| * Chatbot should provide users with open-ended conversational questions and responses. | ✔ |
| * The program should not be able to provide vulgar responses. | ✔ |
| * The program should be able to ask for a clarification or repetition if a users’ input cannot be understood. | ✔ |
| **Stage 4 – User Interface:** |  |
| * The program should have a minimalistic looking interface. | ✔ |
| * The program should include a chat log, text input bar and an ‘enter’ button. | ✔ |
| * The program may include a slideshow of aesthetically pleasing images as the background. |  |
| * The user should be able to change the background image/ slideshow where necessary. |  |

|  |  |
| --- | --- |
| **Profile Management:** |  |
| * Chatbot should be able to retrieve the names of previous users stored in a memory storage. |  |
| * Chatbot should be able to simply notify the user if they have used the program previously. |  |
| * Users’ chat history does not need to be stored in memory, to maintain security and protection of the user. | ✔ |

|  |  |
| --- | --- |
| **Emergency Services Links:** |  |
| * Chatbot may have a small database of useful hotline numbers/ contact numbers. |  |
| * Chatbot should be able to detect signs of distress, and provide emergency contact details where necessary (e.g. providing the user with the suicide prevention hotline number where sentences such as “I really have nothing else to live for…” as given). |  |
| * Users should be able to share anything with the program and feel safe and secured. | ✔ |

|  |  |
| --- | --- |
| **Mini-response databank:** |  |
| * The program may contain an array that stores certain important inputs from the user. |  |
| * Chatbot should be able to match certain user inputs to their appropriate names. |  |
| * Administrators should be able to access these information, but will have their name and time of access recorded. |  |
| * Users should not be able to access these specific responses by accessing their profile using their full-names |  |

|  |  |
| --- | --- |
| **Text-to-speech:** |  |
| * The program may be able to convert audio recordings into appropriate sentences. |  |
| * The program should be able to ask for clarification or repetition if a proper sentence is not constructed. |  |
| * Users should be able to freely use the text-to-speech function, no matter their accents. |  |
| * The program should be able to provide audio responses to the users. |  |
| * The program should be able to alter between different voice packages upon the users’ request. |  |

|  |  |
| --- | --- |
| **Device Compatibility:** |  |
| * The program should exist as a folder that may be transferred and used on other operating systems (e.g. Android or Macintosh) |  |
| * A website that can be accessed on multiple devices to enable users to access the site. | ✔ |

Final stages

|  |  |
| --- | --- |
| **Stage 1 – Basic Input: (Average Difficulty Easy)** |  |
| * Users should be able to enter alphabetical, numeric and certain special characters into the chatbot. | ✔ |
| * The program should be able to provide evidence to repeating users with regards to their last encounter (e.g. sharing a secret that the user wishes the program to keep) | ✔ |
| * Users should be able to freely insert capital letters and ‘spaces’, and still be recognized by the program. | ✔ |
| * Users should be able to submit their responses after clicking on the ‘send’ button or by pressing enter. | ✔ |

|  |  |
| --- | --- |
| **Stage 2 – Processing:** |  |
| * Chatbot should be able to learn new variations in sentences (e.g. “I’m doing great…” vs. “I’m having a jolly-good time…” etc.) | ✔ |
| * Chatbot should compare users’ inputs to a database to provide users with set responses where appropriate. | ✔ |
| * Chatbot should never be able to quit the program through conversation, only through the closing of the window or ‘Alt-F4’. | ✔ |
| * Chatbot’s code should be neatly organized following the principles of object-orientation programming. | ✔ |
| * Users should not be able to access or alter any parts of the program, and should just be restricted to the chat feature itself. | ✔ |
| * Chatbot should be able to hold names and other minor essential information into memory. | ✔ |
| * The chatbot must be able to provide various styles of responses to set inputs (e.g. responding by saying “How are you?” or “What is your mood like today?”) | ✔ |

|  |  |
| --- | --- |
| **Stage 3 – Replies:** |  |
| * The chatbot should be able to provide basic replies to introductory sentences (e.g. Hi there.) | ✔ |
| * The program will be able to provide appropriate responses in relation to the user’s sentence through emotion-like recognition. | ✔ |
| * The chatbot may be capable of providing users with useful links to various sources depending on their intentions. | ✔ |
| * Users should not be able to obtain factual information from the chatbot, to differentiate the program from personal assistants. | ✔ |
| * Users should be able to have open, friendly conversations. | ✔ |
| * Chatbot should provide users with open-ended conversational questions and responses. | ✔ |
| * The program should not be able to provide vulgar responses. | ✔ |
| * The program should be able to ask for a clarification or repetition if a users’ input cannot be understood. | ✔ |
| **Stage 4 – User Interface:** | ✔ |
| * The program should have a minimalistic looking interface. | ✔ |
| * The program should include a chat log, text input bar and an ‘enter’ button. | ✔ |
| * The program may include a slideshow of aesthetically pleasing images as the background. | ✔ |
| * The user should be able to change the background image/ slideshow where necessary. | ✔ |

|  |  |
| --- | --- |
| **Profile Management:** |  |
| * Chatbot should be able to retrieve the names of previous users stored in a memory storage. | ✔ |
| * Chatbot should be able to simply notify the user if they have used the program previously. | ✔ |
| * Users’ chat history does not need to be stored in memory, to maintain security and protection of the user. | ✔ |

|  |  |
| --- | --- |
| **Emergency Services Links:** |  |
| * Chatbot may have a small database of useful hotline numbers/ contact numbers. | ✔ |
| * Chatbot should be able to detect signs of distress, and provide emergency contact details where necessary (e.g. providing the user with the suicide prevention hotline number where sentences such as “I really have nothing else to live for…” as given). | ✔ |
| * Users should be able to share anything with the program and feel safe and secured. | ✔ |

|  |  |
| --- | --- |
| **Mini-response databank:** |  |
| * The program may contain an array that stores certain important inputs from the user. | ✔ |
| * Chatbot should be able to match certain user inputs to their appropriate names. | ✔ |
| * Administrators should be able to access these information, but will have their name and time of access recorded. | ✔ |
| * Users should not be able to access these specific responses by accessing their profile using their full-names | ✔ |

|  |  |
| --- | --- |
| **Text-to-speech:** |  |
| * The program may be able to convert audio recordings into appropriate sentences. | ✔ |
| * The program should be able to ask for clarification or repetition if a proper sentence is not constructed. | ✔ |
| * Users should be able to freely use the text-to-speech function, no matter their accents. | ✔ |
| * The program should be able to provide audio responses to the users. | ✔ |
| * The program should be able to alter between different voice packages upon the users’ request. | ✔ |

|  |  |
| --- | --- |
| **Device Compatibility:** |  |
| * The program should exist as a folder that may be transferred and used on other operating systems (e.g. Android or Macintosh) | ✔ |
| * A website that can be accessed on multiple devices to enable users to access the site. | ✔ |

## Figures

* **Figure 1:AYO**
* **Figure 2:** Cenelia, G. (n.d.). Chatbot Tutorial - CodeProject. [online] Codeproject.com. Available at: https://www.codeproject.com/Articles/36106/Chatbot-Tutorial [Accessed 16 Feb. 2018].
* **Figure 3:** Instructables.com. (n.d.). A Learning Chatterbot in C++. [online] Available at: http://www.instructables.com/id/A-Learning-Chatterbot-in-C/ [Accessed 16 Feb. 2018].