Logo

Description automatically generated

Mental Health AI Chatbot – Mobile Application

Ryan Harper – B00798693

Course – BSc Computing Science

Mentor – Naveed Khan

Contents

[Chapter 1 – Induction 4](#_Toc123747588)

[1.1 Problem Elucidation and statement 4](#_Toc123747589)

[1.1.1 The Main Problem 4](#_Toc123747590)

[1.1.3 The Proposed Solution 5](#_Toc123747591)

[1.2 Project aim 5](#_Toc123747592)

[1.3 Project objectives 6](#_Toc123747593)

[1.3.1 AP1 Objectives 6](#_Toc123747594)

[1.3.2 AP2 Objectives 7](#_Toc123747595)

[Chapter 2 – Literature Review 8](#_Toc123747596)

[2.1 initial investigation of the project/problem context area (Max 5 pages) 8](#_Toc123747597)

[2.1.1 Natural language processing (NLP) 8](#_Toc123747598)

[2.1.2 Machine learning 10](#_Toc123747599)

[2.1.3 AI Chatbots 11](#_Toc123747600)

[2.1.4 Mobile Applications 12](#_Toc123747601)

[2.2 Similar Solution investigation and analysis (Max 5 pages) 13](#_Toc123747602)

[2.2.1 WOEBOT 13](#_Toc123747603)

[2.2.2 WYSA 14](#_Toc123747604)

[2.2.3 Youper 15](#_Toc123747605)

[2.2.3 Mobile Application UI 16](#_Toc123747606)

[2.2.4 AI Chatbot 16](#_Toc123747607)

[2.2.5 Unique Selling point (USP) 17](#_Toc123747608)

[2.2.6 Conclusion 17](#_Toc123747609)

[2.2.7 Comparison of applications 17](#_Toc123747610)

[Chapter 3 – Project Plan and Requirement Specification 18](#_Toc123747611)

[3.1 Stakeholder Identification 18](#_Toc123747612)

[3.2 Justification of result\requirement gathering methodology 19](#_Toc123747613)

[3.2.1 Construction of the system 19](#_Toc123747614)

[3.2.2 Result of gathering methodology 25](#_Toc123747615)

[3.3 Justification of requirements prioritization strategy used 27](#_Toc123747616)

[3.4 System Requirements Specification 29](#_Toc123747617)

[3.4.1 Functional Requirement Specification 30](#_Toc123747618)

[3.4.2 Non-Functional Requirement Specification 31](#_Toc123747619)

[3.5 Justification for selected software Development lifecycle methodology 32](#_Toc123747620)

[3.6 Implementation plan 33](#_Toc123747621)

[3.6.1 Work Breakdown structure 33](#_Toc123747622)

[3.6.2 Effort Estimation 35](#_Toc123747623)

[3.6.3 Gantt Chart (1-2 pages) 37](#_Toc123747624)

[3.6.4 Resource Identification 40](#_Toc123747625)

[3.7 Verification plan 40](#_Toc123747626)

[3.7.1 Verification plan 41](#_Toc123747627)

[3.7.2 Verification Continuous Testing 43](#_Toc123747628)

[3.7.3 Verification Automated Testing 43](#_Toc123747629)

[3.7.3 User Survey 43](#_Toc123747630)

[3.8 Validation Plan 44](#_Toc123747631)

[Chapter 4 – Project Risk Assessment 45](#_Toc123747632)

[4.1 Risk Assessment & mitigation strategy (Technical Risks) 45](#_Toc123747633)

[4.2 Risk Assessment Table 45](#_Toc123747634)

[Chapter 5 – initial Functional Prototype 48](#_Toc123747635)

[5.1 Justification of risk selection 48](#_Toc123747636)

[5.2 Design Artefacts 49](#_Toc123747637)

[5.2.1 – Kivy App Life Cycle 49](#_Toc123747638)

[5.2.2 – Kivy Architecture 50](#_Toc123747639)

[Chapter 6 - References 57](#_Toc123747640)

[‌Chapter 7 - Appendix 63](#_Toc123747641)

[Appendix A 63](#_Toc123747642)

[Appendix B 63](#_Toc123747643)

[3.7.1 Verification plan Test cases 63](#_Toc123747644)

[Appendix C 73](#_Toc123747645)

[Appendix D – Code Manifest 75](#_Toc123747646)

# Chapter 1 – Induction

## 1.1 Problem Elucidation and statement

### 1.1.1 The Main Problem

The concept of mental health has been referenced since before the 20th century. Technical references to mental health were not found before 1946 [[1]](#_[1]). Two years later, in 1948, the new World Health Organization (WHO) was created, and, with that, mental health became more documented. In 1950, the WHO then defined mental health as a condition subject to fluctuations due to a person’s biological factors or social encounters and recommended mental hygiene as a way of maintaining mental health [[2]](#_[2]). Since then, mental health has been observed further to the point where people who need it can get the help they require.

In 2019, the WHO released a research paper that says that 700,000 people worldwide still die yearly from suicide. That’s one person every forty seconds, with it being the fourth leading cause of death of people between 15 and 29 [[3]](#_[3]). Regardless, that number has the potential to be higher as deaths are not reported as such. Even though this number is still shockingly high, it does not mean that mental health support has gotten worse over the years. There are huge pushes now compared to the pre-2000s asking people to speak up and support them, with huge marketing campaigns from orgs like the WHO and the Samaritans. There are helplines open day and night, celebrities, and TV shows showing that it’s all right to tell someone how you feel. But despite all of that, why are there still so many people taking their lives? When it’s been shown that it's okay to tell someone, this answer can change from person to person. No one's answer to that question can be wrong. People may feel like a burden to their family members if they were to tell them about bullying online or in public, current situations in life, the possibility of having no one to talk to, etc.

If people struggling with mental health are alone or struggling to talk to a loved one, how else can they help improve their mental health? Who else can they talk to or get advice from?

Chart, pie chart

Description automatically generatedIntroducing Ai Chatbots, interacting with users with spoken, text, or visual languages. Chatbots can potentially be beneficial for users struggling with mental health disorders, specifically those who are hesitant to tell loved ones due to fear of feeling like a burden. Studies on chatbots have proven valuable in mental health disciplines to assist therapeutic purposes, depression, anxiety, dementia, addiction, stress, and suicide. 53 chatbots, were given 39 studies, 17 for therapeutic delivery were delivered by chatbots of Woebot and Help4Mood, which were used to assist in cognitive behavioural therapy for patients with depression. There were 12 chatbots to assist in training patients with autism with their social skills, and 10 chatbots were implemented as screening tools for different disorders like dementia and addiction [[4]](#_[4]) [[Figure 1]](#_Table_1_–). Ai chatbots can replicate advice that a therapist or psychiatrist would give to a user, however, chatbots won’t know when they’re being lied to by a patient, nor can they pick up on subtle warning signs like body langue of the patient unless provided with additional ai help like imaging.

#### Figure 1 – Chatbot uses in mental health fields [[4]](#_[4])

At this current moment in time, there are few AI or AI chatbots that can replicate what a human can do without additional human intervention. There are, however, numerous AI/Chatbots that can assist with tasks. For example, AI can aid in transcribing medical prescriptions and monitoring heart rate and blood pressure, use in call canters and instant messaging. In relation, chatbots are found on web applications to provide the user with assistance navigating through the website or help them with queries. The most employed chatbots consist of voice-based, hybrid, social messaging, menu-based, rules-based, AI-powered contextual, and support-based chatbots. These are the most well-implemented chatbots, as there are various ways to design a chatbot. AI chatbots can detect the emotion behind the text and compile that data to understand the user and prompt them with a response relating to their current situation. Chatbots regarding mental health are common for phone applications and websites having them built into them or being cantered around chatbots to provide users who are suffering with the help they need.

### 1.1.3 The Proposed Solution

Diagram

Description automatically generated The proposed solution is to build an AI chatbot mobile application that helps people struggling with mental health through advice and one-on-one human interaction like that of a friend, that being the AI. This AI bot will not be implemented to diagnose the user, as anyone looking for a diagnosis of their mental health should seek advice from a trained professional. The mobile application is trying to make the user feel like they have someone to talk to like the user is having a conversation with a human. The chatbot's goal will be to keep this conversation going and chat with the user in a human-like manner. As that’s happening the chatbots' main goal is to try and provide the user with actual advice, then try to convince them to talk to a loved one based on the data the chatbot has gathered. This should flow smoothly so that it doesn’t appear the bot is being too hostile. So, at most, the bot is achieving four goals throughout its process, building a relationship, using the data that the user has provided, and providing information on mental health Its last goal is to try to get the user to talk to a loved one or professional. The final step will get pushed back and the loop continues back around to not seem aggressive toward the user. The overall solution is much more AI training-based and mixed with constant research on talking to people with depression. [[Figure 2]](#_Diagram_2_–)

#### Figure 2 – Logic flow chart

## 1.2 Project aim

Creation of an AI chatbot mobile application to help people suffering from mental health. This project will provide users who feel like they can’t tell anyone about their mental state a buddy to talk to. It will provide advice for the user’s current mental health statue with the side objective of getting the user to talk to someone in their life. To provide the creator further knowledge on AI based applications, creation a mobile application and the different alternatives that comes along with creating both components. This project also challenges the user critical thinking skills and time management.

## 1.3 Project objectives

### 1.3.1 AP1 Objectives

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Criteria Met** |
| AP1#1 | Research the Challenge on mental health. | Yes |
| AP1#2 | Research Current AI chatbots. | Yes |
| AP1#3 | Research how current AI chatbots are used within mental health. | Yes |
| AP1#4 | Deliver a proposed solution. | Yes |
| AP1#5 | Investigation of the project and the problems that come with it. | Yes |
| AP1#6 | Investigate any similar solutions for the project. | Yes |
| AP1#7 | Identify the stakeholders. | Yes |
| AP1#8 | Justification of requirement gathering methodology. | Yes |
| AP1#9 | Justification of requirement prioritization strategy. | Yes |
| AP1#10 | System Requirements Specification. | Yes |
| AP1#11 | Functional Requirement Specification. | Yes |
| AP1#12 | Non-Functional Requirement Specification. | Yes |
| AP1#13 | Justification for selected software lifecycle methodology. | Yes |
| AP1#14 | Implementation plan. | Yes |
| AP1#15 | Verification plan. | Yes |
| AP1#16 | Validation Plan. | Yes |
| AP1#17 | Risk Assessment & mitigation strategy (Technical Risks). | Yes |
| AP1#18 | Risk Assessment Table. | Yes |
| AP1#19 | Implementation Rationale for the chosen risks. | Yes |
| AP1#20 | Design Artefacts. | Yes |
| AP1#21 | Develop initial prototype based on the potential highest risk. | Yes |
| AP1#22 | Design a UI interface that’s not far off from the finished design. | Yes |
| AP1#23 | Get the chatbot up an talking and linked with mobile application. | No |
| AP1#24 | Chatbot can talk to the user and provide it advice, however human like conversation will be met later. | Yes |
| AP1#25 | Test the prototype Some errors can be expected but the main segment of the program must run. | Yes |
| AP1#26 | Show the initial prototype to show how the project will evolve. | Yes |
| AP1#27 | Demonstrate initial functionality. | Yes |
| AP1#28 | Display effective project management skills. | Yes |

#### Table 1 – AP1 objectives

### 1.3.2 AP2 Objectives

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Criteria Met** |
| AP2#1 | Finalise prototype and ensure that all parties agree on next phase. |  |
| AP2#2 | Vigorous coding and training of AI along. |  |
| AP2#3 | Vigorous testing as the coding goes on to ensure that codes are not interacting in a way that stop chatbot output. |  |
| AP2#4 | Potential check ins with mentor to ensure that the program is going well. |  |
| AP2#5 | Revision of code to ensure that all requirements have been met to the according standard. |  |
| AP2#6 | Check for additional feature to be added. (Only if time management is kept well) |  |
| AP2#7 | Test the program. |  |
| AP2#8 | Fix any issues |  |
| AP2#9 | Meeting to ensure program is work address any issues |  |
| AP2#10 | Author gaining a better understanding of the software they have used. |  |
| AP2#11 | Author gaining better management skills. |  |
| AP2#12 | Display learned knowledge well when doing the write ups. |  |
| AP2#13 | Start write ups. |  |
| AP2#14 | Review write ups after completion. (Plan for write up to be finish much before the deadline) |  |
| AP2#15 | Review source code again. |  |
| AP2#16 | Start video demonstration. |  |
| AP2#17 | Displayed technical skills gained throughout the experience in the video. |  |
| AP2#18 | Submit work. |  |
| AP2#19 | Commence oral exam. |  |
| AP2#20 | Demonstrate good communication skills. |  |
| AP2#21 | Be able to talk about project without additional help from notes. |  |

#### Table 2 – AP2 objectives

# Chapter 2 – Literature Review

## 2.1 initial investigation of the project/problem context area (Max 5 pages)

This investigation focuses to access areas in the related project. Including any alternative within a particular field e.g., machine learning and mobile applications. The challenges of each area will be taken into consideration as the strengths of each area.

### 2.1.1 Natural language processing (NLP)

Natural language processing is how computers understand and use natural language text and speech to complete tasks. As a result, NLP is used to try and understand text and speech the same way humans can [[5]](#_[5]). NLP has become more embedded in applications and websites: emails are protected from numerous amounts of spam, automatic machine translation, search engines have a high degree of linguistic sophistication, and dialogue systems are effective when acquiring information and sharing information. NLP draws on other intellectual practices, including computational linguistics, machine learning (ML), and artificial intelligence [[6]](#_[6]). NLP has five phases: processing lexical or morphological analysis, syntax analysis or parsing, semantic analysis, discourse integration, and pragmatic analysis.

**Lexical/Morphological – Phase 1**

Phase One is all about recognizing and analysing word structure. This collection of words is known as a lexicon. Lexical or morphological analysis is the process of breaking down the text file into paragraphs. The source code will then be checked as a stream of characters and altered into lexemes within this phase. Lexical is crucial during the early stages of NLP when text or sound waves will split into words [[8]](#_[8]). The length of lexical objects and their classification can hugely affect processing times in automated language understanding systems. To combat this simplification the storage is performed using morphological analysis of words [[7]](#_[7]).

**Syntax Analysis/Parsing – Phase 2**

Syntactic, also known as syntax analysis, is a technique for checking grammar, sorting words, then displaying the link between them. This analysis verifies if the structure of a piece of text is proper. It will try to parse the sentence to ensure that the grammar is accurate at the sentence level. This only happens if the part of speech generated from the previous phase is accepted, a syntax analyser assigns POS tags based on the sentence structure [[8]](#_[8]). For example, “The moon is in front of the sun” is the correct syntax, and “The sun is in front of the moon” is the incorrect syntax and would be rejected by the syntactic as the structure makes no sense.

**Semantic Analysis – Phase 3**

Phase three is about finding the meaning of the statement. It will focus on the literal meaning of the words, phrases, and sentences. Another priority for it is putting words together to form sentences. It will extract the text's meaning or dictionary definition. It achieves this by mapping the task domain syntactic structure and objects. For example, “The Apple ate my source code”, this sentence syntactically is correct, but it doesn’t make sense as an apple cannot eat, let alone eat my source code. Another example is a descriptive sentence like “Big Gerd” it will be joined in one phrase [[8]](#_[8]).

**Discourse Integration – Phase 4**

Phase four is about the effect of the previous sentence, from the sentence currently in consideration. For example, “Gerd loves hipsters. He wants to become a member.”, “he” will refer to “Gerd”, the same way “hipster” will refer to “member”.

**Pragmatic Analysis – Phase 5**

The final phase of NLP is pragmatics which interprets the given text using the data from the previous step. For example, “Turn off the music” is an order or request to turn the music off.

Diagram, timeline

Description automatically generated

#### Figure 3 – NLP flow

There are also types of NLP techniques that can be used. Below the author has researched techniques that they’ve seen that could be used to understanding before starting the project. These techniques like NLP are also subset of AI or machine learning so some of these techniques may cover aspects to those areas as well. These techniques also include NLP in them.

#### 2.1.1.1 Named Entity Recognition

The goal of name entity recognition (NER) is to recognize and classify names of a person, location, or organization, and numeric expressions like dates, currencies, and percentages. NER is valuable in natural language processing practices such as information transactions (IE) and questions and answering. There are three types of NER systems: monolingual, which deals with single target languages; bilingual, which refers to systems that can deal with two target languages; and multilingual, which refers to systems that can deal with more than two target languages. NER has extraction methods focused on finding and sorting entities into pre-classified groups. For example, “The temperature in Kilkeel, Co. Down is close to 100 degrees on the smelly May Day”. “Kilkeel, Co. Down” is the location, “100 degrees” is the temperature, and “May Day” is the date. NER is for the basic rules of grammar [[9]](#_[9]).

#### 2.1.1.2 Sentiment Analysis

Diagram

Description automatically generatedSentiment analysis is the extraction of information about an entity and the automatic finding of any subjectivities of that entity. The focus is to decide whether the text inputted by the user expresses their positive, negative, or neutral opinions. In terms of sentiment classification, it can be carried out at three levels of extraction: aspect or feature, sentence, and document level. Three different methods help to tackle the problem of sentiment analysis. The first is lexicon-based, the second is machine learning-based, and the third is a hybrid approach. Lexicon, as previously discussed, is the sentiment expressed using a dictionary-based method. Machine learning uses deep learning or traditional models for sentiment analysis. The hybrid approach combines the previous two approaches [[10]](#_[10]).

#### Figure 4 – Sentiment Analysis graph

#### 2.1.1.3 Text mining

Text mining is the discovery and extraction of data, non-trivial knowledge from free or unstructured text. That includes, everything from information retrieval, text classification, clustering, entity, relation, and event extraction [[11]](#_[11]). NLP is then used to normalize the text into structured data that’s suitable for analysis to run machine learning applications. The data from text mining can be displayed using clustered HTML tables, mind maps, charts, and other forms of display. The structured data is integrated into databases and utilized for descriptive, prescriptive, or predictive analytics [[12]](#_[12]). Text mining is integrated into various methods within IT and is utilized by many companies.

#### 2.1.1.4 Machine translation

Machine translation is a vital NLP tool that analyses and generates languages. Top commerce websites will incorporate machine translation into their systems to help break language barriers so everyone can buy from their website. Machine translation preserves the meaning of the text, whether it is using text or pre-input, and then produces fluent text in the language the user has chosen. It works with statistics and currency conversion. An example of machine learning being used in the world is Google. On average, they translate over 100 billion words every day [[13]](#_[13]). Over time, the translation quality has improved, with users’ emotions being contextualised more fluently.

### 2.1.2 Machine learning

Contemporary methods of NLP rely heavily on machine learning, which makes it possible to create complex computer programmes from examples [[6]](#_[6]). Machine learning is an automated process that enables machines to analyse a large data set and then learn from the data to aid in displaying it. Machine learning can aid different fields in IT, such as data analysis, game development, robotics, etc. Machine learning plays a big part in AI development as the AI will need a lot of training to understand most of the data that it takes in. Like NLP, machine learning is also a subset of AI [[14]](#_[14]).

Most chatbots use machine learning algorithms implemented by companies on mobile applications, websites, cloud services, or games so that users always have someone to speak with. Most chatbots are trained by using data from interactions among humans so that they can follow their speech patterns and answer questions. Machine learning needs filters and validation, as proven by the Microsoft chatbot Tay, released in March 2016. This bot aimed to mimic an 18-to-24-year-old American woman. The purpose of this bot was that the more you talk to it, the smarter it becomes. Initially, this was a big success. However, the bot was exploited by a user who found out about its learning process and shared the chatbot with racist slogans and insults. Microsoft responded by shutting down Tay [[15]](#_[15]).

There are four types of machine learning, supervised learning, unsupervised learning, semi-supervised learning, and reinforced learning. These methods are based on the desired outcome of the algorithm.

#### Diagram Description automatically generated 2.1.2.1 Supervised learning

Supervised learning is where the algorithm generates a function that maps inputs to the desired outcome. Supervised learning is common in the classification problem as the target is often to make the computer earn a classification system that has been created. Supervised learning often leaves for the probability for inputs being undefined. As input data is put inputted into the model, it adjusts its weights until the model has been fitted, this happens as a part of the cross-validation process. Regression is used within supervised learning to understand relationships between dependant and independent variables. It is also a common technique for training neural networks and decision trees. These techniques are highly dependent on the data inputted by the pre-determined classifications [[16]](#_[16]).

#### Figure 5 – Supervised learning flow

#### 2.1.2.2 Unsupervised learning

Unsupervised learning is when the computer’s goal is to learn how to perform tasks that it wasn’t taught how to do. Two approaches can be taken. The first approach is to teach the computer not by giving it explicit categorization but by utilizing a reward-type system to indicate success. The goal of this is not to produce a classification but to make choices that increase the reward. The second type of unsupervised learning is called clustering. In this method, the goal isn’t to maximize utility function, but to search for similarities in the training data. The reason for this is that clustering will match well with an intuitive classification. For example, clustering individuals based on demographics will result in the poor in one group and the wealthy in the other [[16]](#_[16]).

#### 2.1.2.3 semi-supervised learning

Semi-Supervised machine learning is the combination of unsupervised machine learning and supervised machine learning methods. This uses large amounts of unlabelled data available and many cases in combination with smaller sets of labelled data. Semi-supervised uses unsupervised and supervised to improve one of the two by utilizing information generally associated with the other. For example, when challenging a classification problem, extra data points for which the label is unknown can be used to help in the classification process. For the clustering method, the learning procedure might help from the information that a particular data point belongs to the same class [[17]](#_[17]).

#### 2.1.2.4 Reinforced learning

Reinforced machine learning is a technique for rewarding desired behaviours while punishing undesirable behaviours. Reinforced learning happens by acting through trial and error. This method helps to stop the system from stalling on less important goals. With time, systems will learn to avoid any negatives that it can and focus purely on the positives. Many other fields use reinforced learning. These include gaming, resource management, personalised recommendations, and robotics. The only issue with reinforced learning is the high demand computing and how frequently the environment can change. Supervised learning can provide much faster and more efficient results than traditional learning.

### 2.1.3 AI Chatbots

A chatbot is a computerised program that helps communication between humans and the bot. It’s a virtual assistant that has become popular recently due to the improvement of AI, machine learning, and NLP. Chatbots can effectively communicate with humans, providing them with answers, advice, and much more [[18]](#_[18]). Chatbots help to free up an organisation's call centres with simple questions like "how do I change my password", so that call assistants can help people with issues out of the user’s control. Chatbots are also available 24/7, ready to help the user and give it the best customer service they can get. Chatbots also use a markup language called Artificial Intelligence Markup Language (AIML), which is used to create chatbots like ALICE. This language is derived from extensible markup language (XML). AIML also has various API packages to make AIML files more flexible and interactive. Many types of chatbots are utilized throughout various applications. These types will be chosen based on what the user thinks is necessary for their application or website. Below are some of the chatbots that have been researched.

#### 2.1.3.1 Menu based

The simplest type of chatbot is the menu-based chatbot. A menu-based chatbot has buttons and top-down menus and follows a set of principles from a decision tree, where the user makes their decision to get their answer. It is up to the user to make these decisions by selecting their options and finding the appropriate response from the AI. The downside of these menu-based chatbots is that they are slower in terms of performance and aren’t reliable enough to get the user to their required question [[18]](#_[18]). Menu-based can, however, be extremely useful for answering users whose answers are not complex. Users may ask questions about resetting their password, trying to find a link to a page, and other simple commands. This helps to deal with the smaller problems the users are facing and helps the organisation as the users will be able to navigate their site much more efficiently.

#### 2.1.3.1 Keyword recognition based

These chatbots identify specific keywords to produce the desired result. The chatbot listens to what the user has entered and replies accordingly. With AI technology and customised keyword lists, the bot chooses an appropriate response for the user using algorithms. These chatbots will begin to fail when there's keyword redundancy between related questions. For example, if the user asks a question like "How do I sign out while also saving my files", the bot will most likely use the keywords "Sign" and "out" to determine which answer to respond with. [[18]](#_[18]) These chatbots are better than menu-based chatbots as the user will not have to think about what to click to get to their desired answer and, instead, they can type in a keyword to find it straight away.

#### 2.1.3.1 Contextual

Diagram

Description automatically generated Contextual-based chatbots are advanced. They apply machine learning and AI technologies like voice recognition, and speech-to-text conversion algorithms to interpret the user's emotions. The bot intends to identify the user's intentions and correspondingly produce a thoughtful response by deciphering the pattern in the database. The bot will learn more from each interaction over time by encountering more and more situations. Fast food delivery application utilizes these bots by storing the user's previous order on a database so that the bot can make recommendations based on the data provided [[18]](#_[18]).

#### Figure 6 – proposed classification of chat bots [[18]](#_[18])

### 2.1.4 Mobile Applications

There are many types of development approaches when building a mobile application. These are four major development approaches, these are native, cross-platform native, hybrid and progressive applications. Each has its own individual advantages and disadvantages [[19]](#_[19]).

#### 2.1.4.1 Native application

Native mobile applications run directly on the operating system of the device’s platform owner like IOS and android. The programming language and framework that is used for this are provided by the platform owner. An advantage of a native application is that it has the best run time performance and has direct access to device APIs. The downside to native is that the cost of building and maintaining the application is costly. Native also has multiple codebases for each platform [[19]](#_[19]).

#### 2.1.4.1 Cross Platform native Applications

Cross-platform native mobile applications can be built in multiple different programming languages and frameworks; however, they will be compiled into a native application that runs directly on the operating system of the device. Cross-platform has a single code base for multiple platforms while also being easy to build and maintain your mobile application. The downside of this is that it's dependent on bridges and libraries for native device features this provides performance issues because of the bridges [[19]](#_[19]).

#### 2.1.4.1 Hybrid-web applications

Hybrid mobile applications are created with standard web technologies, like JavaScript, CSS, and HTML5. These will then be grouped as app installation packages. Hybrid mobile applications work on a web container that offers browser runtime and a bridge for native device APIs. Hybrid mobile applications have a shared code base between web and mobile applications. Development teams with a web skillset can also help in the creation process of the application. But this comes with lower performance and limited support for native application features [[19]](#_[19]).

#### 2.1.4.1 progressive web applications

Progressive web applications (PWAs) provide a different method to traditional mobile app development, by skipping to the app store delivery and app installations. PWA is a web application that uses a set of browser capacities such as working offline, running in the background as a process, and adding a connection to the device and the home screen to create an app-like user experience. With progressive web applications, the same app is available for both web and mobile, and no installation is required through URL. However, progressive has limited support for native applications like most other development processes, and the application capabilities depend on the browser in use [[19]](#_[19]).

## 2.2 Similar Solution investigation and analysis (Max 5 pages)

### 2.2.1 WOEBOT

Woebot is an automated conversational AI designed to provide cognitive-behavioural therapy (CBT) in the format of short, regular conversations and mood tracking. Woebot works like an instant messenger application and can be used on a desktop or mobile devices. Woebot will start each interaction with a general inquiry about the user's current situation (context) and how that makes them feel (mood), with responses provided as words or images to represent the circumstances. Once Woebot has gathered its mood data, users are presented with core concepts of CBT with a link to a video or by using word games. Despite all this, the bot may not fully understand what the user needs, it's described as a choose-your-own-adventure. With each choice the user makes, the bot adjusts its output to match the needs of the user so that they can get help [[20]](#_[20]).

Text, letter

Description automatically generatedA screenshot of a phone

Description automatically generated with low confidence The technical side of Woebot employs several computational practises depending on the specific feature. The overreaching methodology was a decision tree with recommended responses that also acknowledged natural language inputs with distinct segments of NLP techniques embedded at certain points in the tree to decide routing to the successive conversational node. Woebots' conversational styles were based on human clinical decisions, creating the dynamics of social discourse [[20]](#_[20]). Woebot can detect harm-to-self phrases, which include miss spelling and slang phrases, with 98% precision. If the bot detects crisis language like "I want to kill myself", it will confirm to the user what they just said. If the user confirms that is indeed what they said, the bot will offer resources curated with expert advice [[21]](#_[21]).

The user experience for Woebot focuses on mood tracking and goal-oriented, tailored conversations. Depending on the user’s choice, it will focus on CBT mindfulness exercises, gratitude journaling, and reflecting upon patterns and lessons [[21]](#_[21]). Woebot starts by giving the user a short tutorial about the application. It does this by using positive language rather than robotic report-style information. It tells the user that the bot is there to help them. The first slide the user will go through at the start is the statement that "Woebot is not a crisis service." It is made clear that if the user is in deep distress, they should try seeking the help of a medical professional. Woebot chat feature is like text messaging, where you have your option for user input, emojis, and a text suggestion option, so all the users will have to do is click on that to send it to the bot. The way the bot talks to the user is friendly and patient, making the user feel like they are having a normal conversation. Even when Woebot is providing information or advice to the user, it comes across as very casual and doesn’t seem like it has just been thrown out of the blue at the user. The UI is simple and effective; there are no overlapping buttons, no errors when the bot is sending messages, and no long message or loading time issues when sending messages. It requires a few typed responses and allows the user to choose from the set that they want.

#### Figure 7 – WoeBot application

In conclusion, Woebot is a perfect example of what the developer of this project is trying to achieve. The bot doesn’t have any other flashy features that take away from the bot or the user experience. It has additional features to complement the chatbot, like the journal feature. The chatbot has a clear sense of humour that is instantly likable. This creates a character for the user to imagine when they are talking to it. Even when the bot is providing advice, the conversation goes smoothly, with the bot recommending the advice rather than listing it. The pre-built dialogue suggestions are a fantastic feature that lets the conversation flow more smoothly as it prevents the user from missing typing in a word and having the chatbot make them retype it again or produce a completely different outcome. The Woebot disclaimer screen is also a great addition as it tells the user how to communicate with the bot and the fact that it tells the user that this bot shouldn’t be used in a crisis and real help should be sought out. Overall, Woebot doesn’t have many weaknesses.

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| Chatbot has character to it. | Sometimes it will throw out random advice for no reason breaking up conversation. |
| Suggested text options. | Basic UI. |
| Additional features compliment the chat bot. |  |
| Can understand user through broken text. |  |
| Send images appropriate to the situations. |  |

#### Table – 3 Woebot Strengths and weaknesses

### 2.2.2 WYSA

Wysa is an AI-based, emotionally intelligent mobile chatbot application focused on creating mental resilience and improving mental well-being, employing a text-based conversational interface. Wysa helps the user to develop positive self-expression by using AI to develop a responsive self-reflection environment. Like other chatbots, Wysa uses practises like CBT and dialectical behaviour therapy (DBT). The application also uses tools for motivational interviewing, positive behaviour support, mindfulness, behavioural reinforcement, and guided micro-action. The application is free but also comes with a premium version that is £29.99. This grants the user access to a human coach who focuses on patient-specific treatment. The user will still have access to the tools listed and the chatbot on the free version. Wysa was originally developed for at-risk communities in inner cities with youth involved with gangs, which is why it’s not just focused on one area of mental health. The application is now used by users with self-reported symptoms of depression. Wysa has supported clinical services like the NHS, as stated in [[22]](#_[22]).

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedWysa was created by a "15-person team of psychologists, designers, and over 500,000 users" [[23]](#_[23]). The reason for this is that the bot is most likely learning while it's interacting with the user, making it more refined when having to deal with more user inputs. The application works under the WCAG (Web Content Accessibility Guidelines) requirement. The requirements they work on provide sufficient contrast between foreground and background so that users can read text on their applications or websites. They focus on providing consistent navigation options so that the navigation buttons are shown clearly. They include tags and labels to ensure that all forms can be read by the user correctly [[23]](#_[23]). Unlike Woebot, Wysa focuses on other areas more so than the chat, like their online sessions with professionals. However, the chatbot compliments these features better as they act as a tool to help navigate the user to these areas.

The Wysa UI is very clean and easy to use. It has more features than Woebot and more navigation options, but still, the screen doesn’t feel crowded, and the user can see every section. The chatbot section is extremely well done with the night sky background and is easier on the eyes than a white screen. The text bubbles being grey instead of white helps blend in with the background while still standing out for the user so they can easily read the text. The chatbot also has a text suggestion feature for the user, which can help the user get their point across quickly to the chatbot, so it can understand the suggested text better than by inputting dialogue. A downside to this bot is that the chatbot works better with pre-populated text rather than the user's text. On top of that, it feels more like a robot feeding you information rather than having a conversation.

#### Figure 8 – Wysa application

In conclusion, the Wysa application displayed a highly advanced application. Although the chatbot is more of a tool for other features in the application, it still shows the advanced AI chatbot. The fact that there is a premium service does not take away from the free application that is provided. However, users may feel like the chatbot is suggesting that the user swap to the premium version if they do need more mental guidance. Some of this guidance can come through text with a professional over the phone. More users would benefit from face-to-face if their situation was dire or if they were directed to a location near them where they could speak with a professional face-to-face.

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| The application is user friendly. | Only works well with prepopulated responses. |
| Chatbot conversation skills when giving advice | Initial tutorial page would need improvement. |
| Suggested text for chatbot. | The user must be 18, which excludes groups. |
| Chatbot UI. | Chatbot feels bland, less like a character at times and more like a robot feeding you techniques |

#### Table – 4 Wysa Strengths and weaknesses

### 2.2.3 Youper

Youper is a low-cost mobile application that provides a self-guided treatment that may not be accessible to users who have no other access to mental health care. Like the other applications discussed, Youper is a cognitive behavioural therapy application. Youper target audience is users with anxiety and depression. It also has a subscription service. Youper focuses more on user activities and monitoring their condition based on what the application shows. Unlike the other applications, though, Youper uses its AI chatbot as a guide rather than someone to talk to. There is no input, just click the option best suited to them. The application helps set the users' daily goals and includes a guided meditation. This changes based on the activity in which the user is currently participating. It also has personality tests that the AI can use to tailor the users' experience [[24]](#_[24]).

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedYouper utilises AI as it can uniquely combine clinical effectiveness and patient engagement. Youper is powered by three mental health solutions: a conversational agent that listens and interacts with the user; just-in-time interventions that help to manage the patients’ emotional challenges; and personalization to recommend techniques that fit the users’ specific needs [[25]](#_[25]). Youper AI is more of a menu-based chatbot. As the user clicks whatever command they want, the bot will provide them with the information they need. There is no user input in the application, just two options the user can choose. The bot acts more like a navigation system than an actual chatbot. This works for it as the users are still getting the help they need, if not quicker. The AI learns from your recent activities and can recommend more to the user through the chatbot. However, the Youper chatbot has the disadvantage of recommending the user the wrong activity, and the user will have no way to say what they specifically need. Instead, they will just have to click through commands until they get what they need.

The UI for Youper is the best of the three interfaces as it has relaxing creamy colours, all widgets are perfectly spaced, with good resolution photos, and the background and text never distract from the activities section. When users first enter the application, they can either create an account or just try it out without making an account. From this, they can explain to the AI why they are here. Is it for stress? Is it for anxiety? or for other reasons. The benefit of the user being able to use the application without signing in gives them the chance to interact with the application. This shows to the user that the creator is confident in what they have created. The application is easy to navigate, with the chatbot taking control of most of it for the user with their input. The application explains to the user how these activities can help them and lets them know what Cognitive behaviour therapy is.

#### Figure 9 – Youper application

In conclusion, the Youper mobile application is extremely well designed with good use of AI to help navigate their service. Even though the chatbot that they use is a different type than the one the developer is planning to create, this shows other alternatives that can be used for a chatbot if an alternative is needed. What this application did provide, though, is an additional feature that can be added to other mental health applications like the AI chatbot-controlled menu. Overall, the biggest takeaway from researching this application has been how Youper uses its chatbot.

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| Great UI | Chatbot doesn’t let user type leading to difficulty to finding exactly what they want. |
| Good use of navigation. | Any important features are locked behind subscription, so this makes the chatbot limited. |
| Good use of chatbot. | Chatbot can have long loading times. |
| Personality tests that help the AI to understand the users needs better. | Screen can get stuck when scrolling up and down chat. |

#### Table – 4 Youper Strengths and weaknesses

### 2.2.3 Mobile Application UI

UI is important for mobile applications as the user wants to use the application and efficiently find the information they need to use the features provided. UI is important for chatbots, as the user will have to spend time talking to it to gain help or advice. Messages must be clear, and the background must not distract from the messages.

The Woebot UI is the most basic of all the mental health applications researched. It's just a simple black background with a green text box to indicate the user. The chatbot textbox is the same as the background and mixes in with the background. This makes the text look more spaced. However, out of all the other applications, their chatbot was by far the best, and it's obvious that most of their time went into it.

Wysa had the second-best UI as its navigation was simplistic, the background colour didn’t take away from the information presented in front of it, and the font was neatly chosen and never overlapped. The UI for the chatbot section, however, was the best out of the three. The background didn’t blend in with any of the text boxes and didn’t distract the user from the messages themselves. The background was more calming and less bland than the other.

Youper had the best general UI out of the three applications. Each section's buttons stand out well from the background, and the images for the activities aren’t stretched or blurred. The navigation the user can do is simple, with the user able to find the section they need, and the overall colour scheme is good. The chatbot UI layout, like Woebot, is very bland and simple.

In conclusion, Youper had the better general UI for the whole application, minus the chatbot, and Wysa had the best chatbot UI. These will be taken into consideration when constructing the application and designing ideas around general AI.

### 2.2.4 AI Chatbot

Woebot had the best chatbot out of all the applications. Not only did it have suggested text that the user could use, but it was also advanced. The bot itself had character when communicating with the user. The transition from normal conversion to it giving advice was effortless. the chatbot can understand broken strings to complement its crisis detection if the user thought about doing anything harmful to themselves. The bot was also able to respond with images to the user. These images might be something it sends along with text or an actual reply to what the user said with just the image.

Wysa came in second in terms of chatbots. It did have suggestion text and input. Overall, it didn’t have much character, and the user input wouldn’t even get the correct response most of the time.

Youper chatbot is more menu-based, making it much slower, and was only used for navigational purposes so the user could find the appropriate one. This chatbot was still well made for what it was intended to do.

In conclusion, Woebot would be the standard to follow when developing a chatbot. It may not have as many features as Youper and Wysa, but it does have the best display of AI chatbots. It suits the narrative that the project is trying to achieve in building an almost human-like AI. The text suggestion idea is well-developed. The bot has a personality, it can provide advice without disrupting the flow of conversation. This most likely came to be through 1000 hours of training.

### 2.2.5 Unique Selling point (USP)

Woebot’s USP is the advanced AI chatbot they created that can communicate with the user. Youper USP has a clean UI interface and ease of navigation. Wysa is a chatbot that can provide information and techniques for dealing with mental health efficiently. In this project, the unique selling point will be an amalgamation of all three of these USPs. The bot has human-like text patterns, providing efficient information, and the mobile application has an interactive, clean UI. This application can then be developed further beyond these unique selling points. The only limitations to getting to the level of these applications are time and people power. As some of the applications had hundreds of thousands of user interactions to get to where they are, regardless, this is still the aim of the project.

### 2.2.6 Conclusion

In conclusion, many aspects of these applications can be utilised and used when creating the project. These ideas can then be expanded upon, and additional features can be added to the application to make it different from other mental health applications. The main lesson learned from this research is what a good AI chatbot should have as this will be the main feature of the project more so than additional features. It also provided information on what good and bad UI looks like in an application like this. Most applications went with soothing colours rather than bright or dark colours. It showed how some applications display their navigation systems and what makes some better than others. Overall, researching this application provided ideas on what this project should have when it comes to the functionality of the project and what is required for a mental health mobile application.

### 2.2.7 Comparison of applications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Application | AI Chatbot/Application  Strengths/Weakness | Features | interaction level | Relevance to Developers Application |
| 1 | WOEBOT | **Strength**  Advanced Chatbot  **Weakness**  Lack of UI. | The bot is like its own character.  Smooth conversations.  Great emotion detection.  Text suggestions. | 5 / 5 | 5/5 |
| 2 | WYSA | **Strength**  Good UI for chatbots.  **Weakness**  Cold lifeless chatbot, less human like. | Great in providing information.  Clean chatbot interface.  Text suggestions. | 3 / 5 | 3/5 |
| 3 | YOUPER | **Strength**  Amazing UI and navigation.  **Weakness**  Chatbot is less interactive. | Excellent UI  Good use of navigation.  Great additional features.  All features support each other. | 2 / 5 | 3/5 |

#### Table – 4 Comparison of applications

# Chapter 3 – Project Plan and Requirement Specification

## 3.1 Stakeholder Identification

Stakeholders are a crucial part of project management. The stakeholder is described by the project management institute (PMI) as "individuals and organisations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion" (Smith, L. W. (2000) [[26]](#_[26])). The stakeholder’s name is a general term used to illustrate individuals, organisations, or groups that have a shared interest in the project and can help provide resources to the developer to affect the outcome of the project. Typically, project stakeholders include a project manager, a customer, project team members, and the project sponsor, who helps fund the project. A successful project begins well. A good start comprises the project teams discussing their plan, evaluating, and documenting the basic requirements of the highly valued stakeholders. If necessary, key stakeholders’ needs can be monitored and revised throughout the project [[26]](#_[26]).

It’s important not to forget key stakeholders’ requirements, as this can ruin a plan. Plan your time correctly and decide:

* Who the project stakeholders are?
* Their expectations and individual impact on the project.
* Outline the requirement change process, knowing that there will likely be changes.
* Outline the needs and prospects of risk planning.
* Devotedly plan the project and communications strategies.

Stakeholder analysis is vital when developing a project. It indicates a wide array of techniques or tools to recognise and understand the needs and expectations of interests inside and outside the project. Seeing what stakeholder requirements collide and seeing if the possibility of a compromise can occur. The project owner must understand the environment of the project. It can help reduce the impact of unseen problems. The stakeholder analysis is best completed before the beginning phase of the project, as the analysis will involve sensitive interests. It gives the project owner time to uncover unproductive interests [[26]](#_[26]).

|  |  |  |  |
| --- | --- | --- | --- |
| Stakeholders | Role | Influential powers | Interests |
| George Moore | Module coordinator | The Head of COM668 wants to see a working project with creditable features. Will validate the project. | Want the author to create a project from their idea, combing all the teachings they learned throughout the university. |
| Naveed Khan | Mentor | Mentor to provide advice, feedback, and directions to the author/developer. | Wants to see the author implement their ideas to a professional standard while also providing feedback on that. |
| PSG Group | End users / support | Discuss the project with the group mentor. |  |
| Ryan Harper | Author/developer  /designer/tester | Creator of the project. Will be tasked with planning the project, time management, developing, testing, designing, and reporting the project. | To finish the project to the best standard they possibly can. Work hard throughout the project showing what they’ve learned. |
| Family/Friend/students | End user | To test the project and provide feedback on potential improvements. | To test the project throughout development. |

#### Table – 5 Stakeholder identification

## 3.2 Justification of result\requirement gathering methodology

### 3.2.1 Construction of the system

The construction of the system will consist of the method used to create the requirements for the project. Chapter two will help with the ideas that come from the methods as extensive research has been taken to understand what makes an application or AI chatbot good and what doesn’t.

#### 3.2.1.1 Brainstorming

Brainstorming is a technique used to find a conclusion for a particular situation by grouping a list of ideas contributed by members or one person. Four main rules can apply to brainstorming. The first rule is that every idea matters, a wide array of ideas can rule out what is needed and what is not to produce an industry-standard application. The second is to take note of every idea, thought process, implementation, and outcome. It helps to understand idea better as they will know why it was selected and what it will interact with on the system. The third rule is to embrace new techniques. Letting other members of the group represent their ideas and new processes to achieve them. The last rule is to innovate your process. Change the approach of each brainstorming session to come up with new ideas. Below are three mind maps for the mobile application, system, and chatbot. These ideas have come from the author and from the research the author has done in chapters one and two. Researching other applications helped to display the most important aspects to look at when creating a mobile chatbot application for mental health.

The first brainstorming session focused on the mobile application. The reason for implementing this mind map is so that mobile applications can understand the UI, navigation, and additional features that can be built. Some ideas within the mind map can fall into the system or even the chatbot, but the mobile application will interact with both. From research done in [chapter 2](#_Chapter_2_–), the author has chosen the best aspects of each application and noted them down in the brainstorming session. From this, the author also noted the ideas the researched application didn’t have, making the application unique. UI was covered the most in the brainstorming session, as an early plan is to have the UI finished and completed ahead of time to focus on the training of the chatbot. Most of the ideas from this brainstorming session are simplistic but help the UI application for early development. Functional and non-functional requirements came up during each brainstorming session.

Diagram

Description automatically generated

#### Figure 10 – Mobile Application Brainstorm

Diagram

Description automatically generatedFewer ideas were made for the system brainstorming session. It is needed to have realistic expectations of what to achieve and to focus on the application's crucial areas. Most ideas for the system are to help the chatbot function or assist the user with the login. More potential system ideas can come later to enhance the users' features, but these should be some of the focus when developing the system. The AI won’t have a high percentage of human-like dialogue, as research from Section [2.2](#_2.2.1_WOEBOT) showed that most AI chatbot applications required hundreds of thousands of hours of user input to achieve that level of human-like text speech. It's obvious for Woebot, as it has the best human-like interactions with a chatbot. Other mobile chatbot applications like Wysa didn’t have as much user input as Woebot, and it's obvious that they don’t reach the same level of instructiveness as Woebot does.

#### Figure 11 – System Brainstorm

Diagram

Description automatically generatedThe brainstorming session for the chatbot didn’t cover the technical aspects, as that was covered in the system session. This session covered how the AI will expect to interact with the user and what features it should have. This includes how it should offer advice to the user, whether it can understand the user if a words incorrect, and whether it can initiate a conversation with the user rather than requiring the user to initiate one. Text suggestion was derived from research in 2.2, as some mobile applications offered the option of responding to the AI with a prefilled text. It also benefits the AI by preventing the user from using a language that the chatbot may not understand or entering text that is beyond comprehension.

#### Figure 12 – Chatbot Brainstorm

#### 3.2.1.2 Prototyping

#### This research methodology for prototyping will come later, after AP1 is submitted. With the correct lifecycle methodology, the developer can come back and change requirements based on feedback from stakeholders. These ideas came from advancements or features that do not fit the user's needs. Prototyping will display what the application will look like; this will not be a product, so functionality will not be expected. However, most features will need to be implemented to give the stakeholders an idea of what it will look like and what each feature does. After the prototyping is completed, a group PSG meeting will take place, and the application will most likely be discussed in further detail. Prototyping can change the functional and non-functional requirements throughout the creation process, as some may be removed due to time or because the feature is redundant.

#### 3.2.1.3 PSG Meeting

These meetings don’t provide direct research but are used to point the developer in the right direction for their projects. This help comes from the mentor. They will help the developers understand if they can meet the goals or expectations that they have set. It helps with understanding what they should focus on and stops unnecessary work on features that do not help the system. The mentor can also instruct the developers on whether they should add more functionality to the project.

These meetings are performed once every two weeks with the mentor to update him on the progress of reports and applications. with the user updating their shared files so that the mentor can see the progress of the user.

#### 3.2.1.4 Document Analysis

#### Document analysis is another method that comes later in the project. It will happen during the prototyping phase or in the creation of the application. Document analysis is the process of looking back through all the requirements and information to analyse the system. This can help with any information missed in the documentation process or when developing the system. Notes can be documented for further evaluation by the developer to determine whether the information documented is needed. Document analysis can be done throughout the development of the document to help create the requirements for the user. When laying out the requirements and going back through the document, the user may find more to add and more non-functional items to add.

#### The document will be analysed once the write-up is complete and before the initial prototype has been started. Possible meetings may happen, which can help determine if more needs added to the write-up or if other areas need to be covered. The document will also be analysed at the end of the prototype before the video to finalise requirements.

#### 3.2.1.5 Analysis of similar solution

#### As seen in 2.2, an examination of other mobile applications was conducted to determine what services they provided. What made the applications good and what made them bad? This helps provide the developer with ideas for what could be added to the mobile application and what should be avoided. This research provides the user with information on what could be done differently in these applications to make them unique. This research provided the user with many functional and non-functional requirements to start the application. That, along with the brainstorming sessions, helps to visualise what features the application will have and how it will run. 3.2.1.6 User Stories

A user story is an informal explanation of a software feature written from the viewpoint of a customer or end user. A user story should display how a piece of work will give a particular value back to the user. User stories follow a straightforward format: "As a <role>, I want <goal>, so that <benefit>." That is then usually followed by an acceptance criterion from the developer or manager of the project. Its purpose is to determine the range and requirements for a project and consider whether the user stories are met [[27]](#_[30]). To assist developers in planning and calculating their effort, an acceptance criteria scope must be defined.

User stories should contain six attributes to make them successful, according to Mike Chon. The first is **independence** by avoiding any dependencies, as this can lead to difficulty in prioritising planning. In the case of this project, prioritisation won’t be handled by this method but instead with other methods. The second is **negotiable,** which means the user and developer stories should seem like a negotiation rather than a contractual obligation. The third criteria is the **value** of the user stories to the user or owner of the product, not just the developers. The fourth attribute is that the user stories are **estimable** in terms of story points. The fifth attribute is that the story should be as **short** as possible while still providing value. The last step is **testability,** as it demonstrates that the story meets the customers’ expectations. This is also known as I.N.V.E.S.T (Independent, Negotiable, Value, Estimable, Small, Testability) [[28]](#_[28]_1).

For this project requirement, prioritisations like MoSCoW and Wiegers' relative weighting will be used to assign true effort. The user stories will have their own priority value to dictate what features should be considered when delivering the functional and non-functional requirements. The user story method is useful because it fits well with agile methodologies, as its short stories provide a short problem with a short, answered solution. Agile iterations are one possible methodology that will be used by the author when creating the system. Agile works best with user stories, MoSCoW, and relative weighting, as these steps are a quick problem and solution identifiers.

|  |  |
| --- | --- |
| ID: | US1 |
| Title: | Retired living alone |
| User Story: | As a retired male living alone,  I want to get mental health advice on depression without having to look through the internet,  So that can better understand what help is out there for someone with depression. |
| Acceptance criteria: | **Acceptance criteria 1:**  Given that the user has the chatbot opened  When they ask it to give them advice on a certain subject like depression,  Then the chatbot will supply the user with advice where they can talk back and forth for more specific details.  AND ensure that the advice is more targeted to their situation  **Acceptance criteria 2:**  Given the user has the application opened,  When the customer clicks advice page,  Then they will be supplied with a coherent list of mental health problems that are all grouped together for easier viewing  AND ensure that they can get help. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 9 |

#### Table – 6 User Story 1

|  |  |
| --- | --- |
| ID: | US2 |
| Title: | Uni student living away from home |
| User Story: | As a university student living away from home alone,  I want to be able to talk to about my daily life problems,  So that I can have someone listen to me. |
| Acceptance criteria: | Given that the user has an account created,  When the user opens the chatbot,  Then the chatbot will engage in a conversation  AND ensure that they have someone to talk to. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 9 |

#### Table – 7 User Story 2

|  |  |
| --- | --- |
| ID: | US3 |
| Title: | Call centre worker |
| User Story: | Aa a call centre worker that works at any given time a day,  I want a chatbot that saves my text logs,  So that when I’ve not responded while at work, I can check what was last said. |
| Acceptance criteria: | Given that the user has had a prior conversation with the bot,  When the user exits out of the chatbot or application,  Then their chat messages will be saved  AND will be viewable the next time it is opened. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 7 |

#### Table – 8 User Story 3

|  |  |
| --- | --- |
| ID: | US4 |
| Title: | Counsellor |
| User Story: | As a counsellor,  I want an application that gives client advice as well as a journal,  So that we can check how they felt throughout the week. |
| Acceptance criteria: | **Acceptance criteria 1:**  Given that the user has the application journal opened,  When they write the journal,  Then the information will be saved for the user,  AND can be viewed at a later session.  **Acceptance criteria 2:**  Given that the user has the chatbot opened,  When have a conversation with the chatbot,  Then their conversation will be stored,  AND can be viewed for a later session. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 5 |

#### Table – 9 User Story 4

|  |  |
| --- | --- |
| ID: | US5 |
| Title: | High school student in finale year |
| User Story: | As a high school student,  I want a chatbot with different personalities,  so that I can match them with my friends to feel like I’m talking to them about my problems. |
| Acceptance criteria: | Given that the user has signed up for an account,  When the user opens the settings,  Then they can assign a personality to the chatbot  AND can have a conversation like their talking to their friend. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 3 |

#### Table – 10 User Story 5

|  |  |
| --- | --- |
| ID: | US6 |
| Title: | Highschool Teacher for learning for life |
| User Story: | As a high school teacher,  I want a mental health application with a login for each user,  So that I can sign up students to the application for exercises. |
| Acceptance criteria: | Given that the user has downloaded the application,  When they click into it,  Then they will be prompted to sign up for an account,  AND then they use the application for exercises. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 7 |

#### Table – 11 User Story 6

|  |  |
| --- | --- |
| ID: | US7 |
| Title: | Retirement home patient |
| User Story: | As a retirement home patient,  I want to talk to the bot through text the computer suggests, as I’m not very good and using keyboards,  So that I can talk to someone without taking too long. |
| Acceptance criteria: | Given that they have the application opened,  When they click into the chatbot,  Then they will be sent a greeting from the chatbot  AND then can use the text suggestion. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 6 |

#### Table – 12 User Story 7

|  |  |
| --- | --- |
| ID: | US8 |
| Title: | Football coach for under 18s |
| User Story: | As a football coach,  I want a application that can clear the chatbot text,  So that different players can use this phone for advice and the next person doesn’t see what they’ve typed. |
| Acceptance criteria: | Given that the user has interacted with the chatbot,  When they go into settings,  Then they can clear the previous chats that they have had  AND the next person can chat to the bot. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 7 |

#### Table – 13 User Story 8

|  |  |
| --- | --- |
| ID: | US9 |
| Title: | Young person with ADHD |
| User Story: | As someone with ADHD,  I want an application with a smooth interface with no background image or distracting colours,  So that I don’t lose track when talking to the chatbot. |
| Acceptance criteria: | Given that the application has opened,  When the user signs up for an account,  Then they can change aspects of the design like darkness mode  AND then the colours shouldn’t be distracted from the chatbot. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 7 |

#### Table – 14 User Story 9

|  |  |
| --- | --- |
| ID: | US10 |
| Title: | Delivery driver |
| User Story: | AS a delivery driver,  I want an application that save my data to the cloud instead of my phone,  So that I can save storage on my phone for other on the go application. |
| Acceptance criteria: | Given that the user has had a conversation with the chatbot,  When they click off the chatbot or application,  Then it will save their information to the cloud if they have a signal  AND save storage on their phone. |
| Definition of Done: | * Acceptance criteria met * Test passed * Code Reviewed * Function/non-functional requirements met |
| Priority (1 - 10): | 6 |

#### Table – 15 User Story 10

### 3.2.2 Result of gathering methodology

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | User Story relation ID |
| F1 | Must accept User input for chatbot even with numbers from the users. | N/A |
| F2 | Chatbot Must be responsive when the user asks it any questions. | US8/ US7/ US5/ US4/US3/US2/US1 |
| F3 | Chatbot message and user messages display each time when the user enters in text and when the bot is expected to output it. | US8/ US7/ US5/ US4/US3/US2/US1 |
| F4 | User must be able to navigate easily throughout the mobile application. | US9 |
| F5 | User Must be able to ask for advice and receive advice on the appropriate situation the user is discussing. | US2/US1 |
| F6 | User can sign in / log out of account through settings menu. | US6 |
| F7 | User can Sign up to create an account at the start screen. | US6 |
| F8 | Functional Mobile application. | N/A |
| F9 | Functional Chatbot. | N/A |
| F10 | Database Must store user login and details and can be possibly edited if the user needs to change. | US10/US6 |
| F11 | Save user information for signed in accounts and edit username and password. | US10/US6 |
| F12 | Should allow user to enter specific profile details. | N/A |
| F13 | Advice must be Accurate to the user situation. | US2/US1 |
| F14 | Database should store a particular users chatlog. | N/A |
| F15 | Application could have advice page based on user situation. | US1/US4 |
| F16 | Settings menu for user. | US5/US8 |
| F17 | Chatbot could have text suggestions. | US7 |
| F18 | Chatbot Should engage with the user first. | N/A |
| F19 | Chatbot can understand broken text. | N/A |
| F20 | User Can clear chat logs from data base. | US5 |
| F21 | Light/Dark mode in the settings menu using slider. | US9 |
| F22 | The chatlogs could be saved to the cloud. | US10 |
| F23 | Chatbot could send images or emojis along with the correct Text. | N/A |
| F24 | User could have profile picture show up when talking to the bot. | N/A |
| F25 | Chatbot won't have different personalities to fit different users. | US5 |

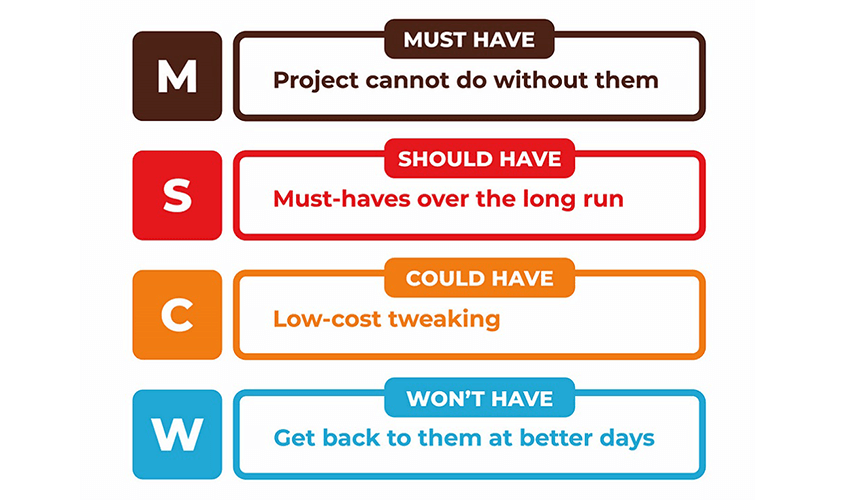
#### Table – 16 Functional Requirements non-prioritized

|  |  |
| --- | --- |
| Requirement ID | Description |
| NF1 | Human like text from the chatbot 50% to 90%. |
| NF2 | Understand users’ emotions. |
| NF3 | Understand when user gets defensive to change the subject. |
| NF4 | Understand the user’s situation. |
| NF5 | Provide the user with accurate responses. |
| NF6 | Chatbot can understand broken text. This was put into requirements as it can be hard coded in some respects to understand broken text or through lots of machine training. |
| NF7 | Chatbot never comes across as aggressive. |
| NF8 | Chatbot to convince the user to talk to a loved one. |

#### Table – 17 non-functional Requirements non-prioritized

Above are the results of the gather methodologies used to obtain the requirements for this project. These requirements are subject to change but are primarily used to guide the developer on the basics of what the system must do for it to be functional. After the prototyping phase more requirements can be added either to improves other areas or because most of the other requirements have been met. This system works better with an agile style of methodology. The thought process of doing the user stories was to create new features but also understand how existing one that were already created would work. This process helps the developer to understand feature when implementing them into the system.

## 3.3 Justification of requirements prioritization strategy used

Once most of the requirements have been identified, it's important to prioritise them to see what should be the main focus of the project, and what needs to be done to have a fully functional project. The functional requirements will be prioritized, and the non-functional requirements will also be. However, because most non-functional will be created with functional requirements in mind, assigning a weighting to non-functional will be more difficult. Prioritization may be changed for requirements as the method for gathering requirements will still be active with the prototyping and document analysis being documented in its table with any changes, or any high priorities identified with the prototyping phase.

#### 3.3.1 MoSCoW prioritization

The approach for MoSCoW prioritization originated from the Dynamic Software Development Method (DSDM). MoSCoW stands for: must have (or minimum usable subset), should have, could have, and won’t have (but would like in the future). Which might have been the first agile methodology even before some of the iterative developments like agile. MoSCoW is an easy way to assist in the sorting of features into priority order. This helps developers understand what the customer view on what is essential to their project and what is not [[29]](#_[27]).

#### Figure 13 – MoSCoW definition

**‘Must Haves’** - are features that must be included before the product can be released to the public or for company use. It’s important for developers to have clarity on this before the project begins, as this will be the minimum cope of the project to be useful.

**‘Should Haves’** – are features that are not considered crucial for the project to release. However, they are considered to be important to the user.

**‘Could Haves’** – are features that would be nice to have and could potentially be added to the project without incurring any additional effort or cost. These will be removed if the project fails to meet timescales.

**‘Won’t Haves’** – are features that were requested but have been purposely excluded from the project for the planned duration and may be added in a future phase of development [[29]](#_[27]).

When using the MoSCoW method it is good to have a large amount of should have and could have requirements. This will help to provide the overall project with more flexibility if problems start to occur during its duration, this gives the project a contingency for more features. For projects that only have 'Must Haves', the scope cannot be diverse. Meaning that the project will not have more features to add or fall back on when something goes wrong. Furthermore, cost and timescales won’t be able to get fixed as they don’t have a reasonable contingency plan. 'Could haves' help to stretch the task as they are features that will be included if it’s possible, but the release date of the project will not b moved to accommodate them if the developers run out of time [[29]](#_[27]). MoSCoW has its advantages and disadvantages, most advantages have already been listed. The disadvantages will only occur in bigger teams where there are more inputs from stakeholders and users or when trying to measure what from the 'Must Have' groups should be done first when no relative weighting is applied.

The MoSCoW approach also works well with iterative agile approach methodologies. Iterations are used to split the project into digestible chunks, with each iteration focusing on completing important components in the sprints. The MoSCoW approach is utilized to prioritize what requirements should be put on hold for each iteration and what should be focused on more. This supports the developer to produce solutions more rapidly. The project will also be more flexible with the adaptability to change, and issues being discovered quickly. This is the approach the author is going to be taking as seen in [chapter 3.5](#_3.5_Justification_for). These approaches will help the developer research and learn while also being able to add new components to the project as they go through the prototyping phase.

The MoSCoW method work in tangent with relative weighting. MoSCow is good a delegating what is necessary for a project into groups but, when they are in their groups it isn’t good at prioritizing what the developer should focus on first. For example, its ‘must have’ group could have three must-have requirements but if relative weight isn’t used then they will not know what of the three they should start [[30]](#_[28]). However, with relative weighting, the developers can easily understand the cost, time, and effort that will need to be applied to the three so they can easily weigh the three to focus on which one is more important.

#### 3.3.2 Relative weighting

Relative weighting allows developers to understand which features to implement and what priority order those features should come in. These priorities are set with a numerical value with the highest having the most priority and the lowest having the least. The high-priority features are ones that the stakeholders have requested, and lower priority may be additional functionality that can help the system but can run without it [[31]](#_[29]).

The relative weighting approach that the author has chosen to take is Wieger’s relative weighting. This method calculates the prioritization based on the value, cost, and risk of each feature. This works in tangent with the priority strategy MoSCoW. Wiegers method has an eight-step approach [[31]](#_[29]):

**Step 1** – List all the requirements, feature, or use cases that the developer or stakeholder wish to prioritize in a spreadsheet.

**Step 2** – Estimate the benefit that each requirement provides the customer or business from a scale of 1 to 9, where 1 will mean little to no benefit and 9 being of high importance or crucial.

**Step 3** – Estimate the relative penalty the customer or company would suffer if the feature were not included, this is also rated on a scale of 1 to 9.

**Step 4** –This sum comes from the benefit and penalty. By default, both would be weighted equally. However, the benefit is normally weighted twice as heavily as the penalty. This will then add up the total product value.

**Step 5** – Estimate the relative cost of implementing each feature. The same scale of 1 to 9 still stands. Then the percentage of the total cost for each feature will be added. The cost rating is estimated by factors such as requirement complexity, the extent of UI work required, the ability to reuse design or code, and the levels of testing needed.

**Step 6** – Estimate the relative degree of technical risks with each feature from the standard scale. A scale of one means is easy to program a scale of 9 means that the developer’s expertise might not reach the high standard needed. Cost and risk normally weight equally.

**Step 7** – Once all details are noted then the overall priority can be calculated. This is done by the formula:

#### Figure 14 – Wiegers relative weighting equation

The percentage of the value is dived by the risk and cost once they’ve been multiplied by their weight, they are then added together. Additionally, this can be like the formulae below with the user till getting the same results as the value is just the total of the top row but into a percentage.

#### Figure 15 – Wiegers relative weighting equation alternative

**Step 8** – Producing the sorted list from highest to lowest [[31]](#_[29]).

This is known as a semi-quantitative method and is not mathematically rigorous. It is limited by the user’s knowledge to estimate benefits, penalties, costs, and risks for each requirement. However, the model can be altered to give different requirements and different priorities based on stakeholders wanting different features added or feeling like another feature may need to be removed [[31]](#_[29]).Graphical user interface, application, table

Description automatically generated

#### Figure 16 – Requirements relative weighting table Not sorted

Graphical user interface, application, table

Description automatically generated

#### Figure 17 – Requirements relative weighting table sorted

## 3.4 System Requirements Specification

System requirements are essential for every project. They are a set of deliverables that are given to the developer by the stakeholder, customers, or requirements that they have set and are expected to be met. Non-functional requirements are different from functional requirements as they are what the user is to expect from the system, rather than a specific feature like a button for taking a photo. Non-functional covers performance, scalability, and security. This will be shown more in the tables below.

Every gathering method and every prioritization method used has been implemented into this. As seen in [figure 16](#_Figure_16_–) and [figure 17](#_Figure_17_–) the requirements have already been initialized a weighting has already been set for them. Below will cover the ID, describe the MoSCoW level, and the relative weight from highest to lowest.

### 3.4.1 Functional Requirement Specification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement ID | Description | MoSCoW Prioritization | Relative weighting | User Story relation |
| F1 | Must accept User input for chatbot even with numbers from the users. | M | 1.953 | N/A |
| F2 | The chatbot Must be responsive when the user asks it any questions. | M | 1.953 | US8/ US7/ US5/ US4/US3/US2/US1 |
| F3 | Chatbot messages and user messages display each time when the user enters text and when the bot is expected to output it. | M | 1.597 | US8/ US7/ US5/ US4/US3/US2/US1 |
| F4 | Users must be able to navigate easily throughout the mobile application. | M | 1.102 | US9 |
| F5 | The user Must be able to ask for advice and receive advice on the appropriate situation the user is discussing. | M | 0.919 | US2/US1 |
| F6 | Users can sign in / log out of accounts through the settings menu. | M | 0.911 | US6 |
| F7 | Users can Sign up to create an account at the start screen. | M | 0.855 | US6 |
| F8 | Functional Mobile application. | M | 0.837 | N/A |
| F9 | Functional Chatbot. | M | 0.837 | N/A |
| F10 | The database Must store user login and details and can be possibly edited if the user needs to change. | M | 0.822 | US10/US6 |
| F11 | Save user information for signed-in accounts and edit username and password. | M | 0.683 | US10/US6 |
| F12 | Should allow a user to enter specific profile details. | S | 0.651 | N/A |
| F13 | Advice must be Accurate to the user's situation. | M | 0.651 | US2/US1 |
| F14 | The database should store a particular user's chatlog. | M | 0.542 | N/A |
| F15 | The application could have an advice page based on the user's situation. | C | 0.521 | US1/US4 |
| F16 | The settings menu for the user. | S | 0.521 | US5/US8 |
| F17 | A chatbot could have text suggestions. | C | 0.498 | US7 |
| F18 | The chatbot Should engage with the user first. | S | 0.473 | N/A |
| F19 | A chatbot can understand the broken text. | S | 0.407 | N/A |
| F20 | User Can clear chat logs from the database. | S | 0.398 | US5 |
| F21 | Light/Dark mode in the settings menu using the slider. | C | 0.355 | US9 |
| F22 | The chatlogs could be saved to the cloud. | C | 0.341 | US10 |
| F23 | A chatbot could send images or emojis along with the correct Text. | C | 0.279 | N/A |
| F24 | The user could have a profile picture show up when talking to the bot. | C | 0.260 | N/A |
| F25 | Chatbots won't have different personalities to fit different users. | W | 0.089 | US5 |

#### Table 18 – Functional requirements

### 3.4.2 Non-Functional Requirement Specification

|  |  |
| --- | --- |
| Requirement ID | Description |
| NF1 | Human-like text from the chatbot is 50% to 90%. |
| NF2 | Understand users’ emotions. |
| NF3 | Understand when the user gets defensive to change the subject. |
| NF4 | Understand the user’s situation. |
| NF5 | Provide the user with accurate responses. |
| NF6 | A chatbot can understand the broken text. This was put into requirements as it can be hard coded in some respects to understand the broken text or through lots of machine training. |
| NF7 | Chatbot never comes across as aggressive. |
| NF8 | Chatbot to convince the user to talk to a loved one. |

#### Table 19 – Non-functional requirements

## 3.5 Justification for selected software Development lifecycle methodology

Diagram

Description automatically generatedThe methodology chosen for this project is iterative and incremental development (IID). IID is used because it enables fast reactions to adjustments. This means that IID is built through iterations, incrementally adding new features to the project [[33]](#_[33]). The iterative model moves like a mini waterfall linked together. It is, however, closely related to the agile scrum methodology. The reason for this is that both respond to changes in a set of plans. Iterative and agile software development both focus on producing fully functional slices of software and not just prototypes [[34]](#_[34]). This process may prove challenging when training the chatbot; however, this can be broken down by training the chatbot in certain areas of its responding method. For example, how does the bot respond when the user is happy, sad, embarrassed, and so on? As seen in [Figure 18,](#_Figure_18_–) the process of IID is simple, as it just goes through the planning for the iteration that is being added, the requirements that are being added, analysing the choices, and then implementing them. Then the developer tests the feature, development normally won’t come in the first cycle until testing has been done. Then the feature will be evaluated. The process will then be repeated until it is ready for deployment, at which point it will stop.

#### Figure 18 – Iterative process graph

Incremental approaches help break down the development process into small, convenient sections known as "increments." Every increment the user implements will be an improvement over the last, with these improvements happening step by step. Iteration is when software development events are repeated in cycles. This means that a new version will be produced until the correct feature has been achieved [[35]](#_[35]). IID has been shown to deliver a better product more quickly compared to something like a waterfall. IID is more manageable, and waterfall tends to be late with errors still in the system as it doesn’t go back in its process.

So far throughout the documentation, other methods like user stories, MoSCoW, and relative weighting are utilised within methodologies like IID to understand requirements faster [[Chapter 3.2]](#_3.2_Justification_of). These approaches were utilised as the author was in favour of a more agile approach to completing the project. As the project proposes new areas in which the author has not researched or utilised the software before. This methodology will provide flexibility when needed as the user can add requirements and remove them depending on the situation. Thanks to MoSCoW and relative weighting, the IID model can focus on building increments and iterations for that requirement. The developer will know if something requires a lot of effort because it may require more increments and iterations.

Another method that was considered for the software development life cycle was prototyping. This is a method that lets the user create the system when the stakeholders do not exactly know what the exact requirements they want are. This helps give the creator and stakeholders a visual aid when trying to understand what the system needs. The reason why this was considered was for the actual project, as a prototype must be made for it. However, in the prototyping methodology, the initial prototype gets thrown out for the building of the real project. Because of the time constraints in this project, the author has chosen to go iterative, so that the prototype can be kept while also being produced at a higher level with IID.

IID also provides a much more viable and understandable scheduling system [[35]](#_[35]). A workflow map can be drawn out with the increments and iterations that will be made throughout the project. The developer can follow this guide to know if they are meeting their targets and what the next area of the project they need to focus on is. Without this plan, it will be more difficult to set aside time when designing a Gannt chart. Effort estimation alongside the workflow map can help the Gannt chart have a more consistent flow.

## 3.6 Implementation plan

### 3.6.1 Work Breakdown structure

Diagram, schematic

Description automatically generated

#### Figure 19 – Work load Diagram

#### 3.6.1.1 increment 1

Increment one involves the report for AP1. This will cover all chapters of the module handbook. It will help with the development of the prototype and the overall application going forward with the project. AP1 is the overall basis of this project and will contain all of the research for the project, including the development methodology. AP1 will also be included in increment two; however, this will only be for the project evaluation by the stakeholders for marking. This increment will take about eight weeks to complete. This will most likely be the longest increment as project ideas had to be agreed upon, health and safety had to be established, and AP1 is the main setup for the project going forward.

#### 3.6.1.2 increment 2

Increment two will involve the prototype for the project. This will use the build-up from the previous increment to produce a suitable prototype to present to the stakeholders for evaluation. The evaluation will then further the project for AP2 and the rest of the increments. Increment two will work to develop the main requirements of the project. These requirements may not contain the highest risk but will need to work for the application to be fully functional. Increment two will take two to four weeks to complete. This is shorter than the previous increment as the developer only has to produce the bases of their application. The developer will also have other obligations so realistic time constraints and planning will have to be accounted for to produce an application that the stakeholders can evaluate.

#### 3.6.1.3 increment 3

Increment three will focus on login and sign-up for users, storing their details on a data base internally or possibly on the cloud, both ideas will be implemented at the time. The GUI will have to be drafted before the development of this feature to produce and early development design. This phase will be the shortest so that the author can focus on other increments.

#### 3.6.1.4 increment 4

Increment four will be the development of the AI chatbot. The phase will focus on the AI being able to provide the user with advice, detect their emotional state and have situational awareness of the user when interacting with them. This is the most important increment as it covers the main challenge of the overall project. The AI chatbot will have to be functional. This increment will most likely have the most time delegated to it throughout the entire project. This increment, if needed will be split into three other increments. One to detect emotions, the other to detect the user's situation, and the last one to give advice to the user.

#### 3.6.1.5 increment 5

Increment five is about reviewing and updating the GUI. The prototype will finalise what the GUI will look like, however, this will most likely only cover a simplistic version of the GUI. This phase, however, has the possibility of being changed to only add additional features. These features will be small but will be focused on improving the user experience. The reason GUI was chosen over this at the moment is for stakeholder guidance. This phase will likely be the shortest out of all of the increments, which is why it comes toward the end of the development cycle. This stage also includes a review stage, where the plan can be modified if necessary.

#### 3.6.1.6 increment 6

Increment six will be the final stage unless other increments need to be added before it. This stage will consist mostly of the evaluation of the project. This is where the author will reflect on what they have done and critically analyse what they could have done better throughout the project. The final tests will include a review of the final requirements to be delivered. This stage will most likely last two or three weeks, depending on the time frame left for the developer.

Overall, the first half of this plan will be followed. Changes may be made in increments of four for the AI chatbot's further development.

### 3.6.2 Effort Estimation

#### 3.6.1.1 September 2022 – January 2023

|  |  |  |  |
| --- | --- | --- | --- |
| Increment | Description | Estimated days | Actual days |
| **1 - Report AP1** | | | |
|  | Initial idea/report | 1 | 1 |
| Research | 7 | 7 |
| Introduction | 2 | 2 |
| Literature Review | 7 | 7 |
| Requirement Gathering and Prioritisation method | 7 | 7 |
| Requirements | 4 | 4 |
| Plan | 10 | 10 |
| Risk Assessment | 3 | 3 |
| Review Report (Bleeds into increment 2) | 15 | 15 |
| **Increment Total Days:** | | 56 | 56 |
| **2 - Prototype** | | | |
|  | Add libraries to project file (Set-Up) | 1 | N/A |
| Set Up RASA | 3 | N/A |
| Design UI Chatbot | 4 | N/A |
| Combine RASA & UI | 2 | N/A |
| Create User Input | 1 | N/A |
| Train AI to respond to basic commands | 7 | N/A |
| Testing | 1 | N/A |
| Evaluation | 7 | N/A |
| **Total Days:** | | 26 | N/A |
| **2 – Report** | | | |
|  | Initial Prototype | 2 | N/A |
| Review Requirements | 2 | N/A |
| Record Video Analysis | 2 | N/A |
| Submit AP1 | 1 | N/A |
| **Total Days:** | | 7 | N/A |
| **Increment Total Days:** | | 33 | N/A |

#### Table 20 – AP1 Time Effort

#### 3.6.1.2 January 2023 – May - 2023

|  |  |  |  |
| --- | --- | --- | --- |
| Increment | Description | Estimated days | Actual days |
| **3 - Login** | | | |
|  | Implement database | 1 | N/A |
| Implement Code | 2 | N/A |
| Attempt implementation on cloud | 3 | N/A |
| Testing | 1 | N/A |
| Evaluation | 1 | N/A |
| **Total Days:** | | 8 | N/A |
| **3 - Report** | | | |
|  | Design | 2 |  |
| Implementation | 2 |  |
| Validation | 2 |  |
| **Total Days:** | | 6 |  |
| **Increment Total Days:** | | 14 |  |
| **4 – Store User chatlogs** | | | |
|  | Implement database | 4 |  |
| Implement Code | 7 |  |
| Attempt implementation on cloud | 3 |  |
| Testing | 4 |  |
| Evaluation | 3 |  |
| **Total Days:** | | 21 |  |
| **4 - Report** | | | |
|  | Design | 2 |  |
| Implementation | 2 |  |
| Validation | 2 |  |
| **Total Days:** | | 6 |  |
| **Increment Total Days:** | | 27 |  |
| **5 – Training AI model** | | | |
|  | Train emotion detecting | 7 |  |
| Train Advice provisions | 7 |  |
| Train situational Awareness | 7 |  |
| Test Chatbot | 7 |  |
| Evaluate Chatbot | 3 |  |
| **Total Days:** | | 31 |  |
| **5 - Report** | | | |
|  | Design | 2 |  |
| Implementation | 2 |  |
| Validation | 2 |  |
| **Total Days:** | | 6 |  |
| **Increment Total Days:** | | 47 |  |
| **6 – Refine GUI** | | | |
|  | Review Old design | 2 |  |
| Decide improvements | 2 |  |
| Testing | 1 |  |
| Evaluation | 1 |  |
| **Total Days:** | | 6 |  |
| **6 - Report** | | | |
|  | Design | 1 |  |
| Implementation | 1 |  |
| Validation | 1 |  |
| Review plan to add more increments if needed | 1 |  |
| **Total Days:** | | 4 |  |
| **Increment Total Days:** | | 10 |  |
| **7 - Conclusion** | | | |
|  | Review code | 5 |  |
|  | Verify finale requirements | 2 |  |
|  | Run finale tests | 3 |  |
|  | Report evaluation | 4 |  |
|  | Review Plan | 2 |  |
| **Total Days:** | | 16 |  |
| **7 - Report** | | | |
|  | Conclusion | 3 |  |
|  | Video | 4 |  |
|  | Stakeholder evaluation | 2 |  |
|  | Submit AP2 | 1 |  |
|  | AP2 Oral examination | 1 |  |
| **Total Days:** | | 10 |  |
| **Increment Total Days:** | | 26 |  |

#### Table 21 – AP2 Time Effort

### 3.6.3 Gantt Chart (1-2 pages)

#### 3.6.2.1 AP1 Gant Chart

##### Graphical user interface Description automatically generated Increment 1

#### (Figure 20 – Gantt chart increment 1)

##### Chart Description automatically generated with medium confidence Increment 2

#### (Figure 21 – Gantt chart increment 2)

#### 3.6.2.2 AP2 Gant Chart

##### Increment 3

Chart

Description automatically generated

#### (Figure 22 – Gantt chart increment 3)

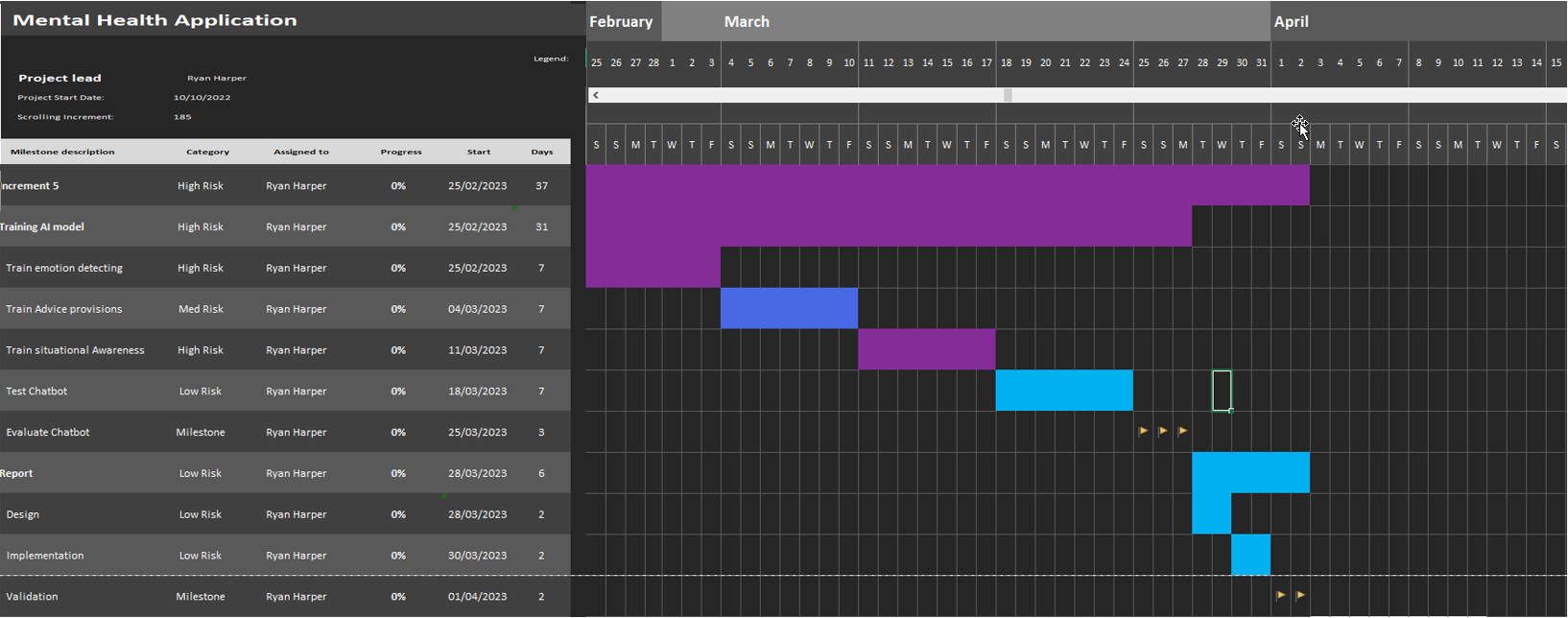
##### Increment 4

A picture containing timeline

Description automatically generated

#### (Figure 23 – Gantt chart increment 4)

##### Increment 5



#### (Figure 24 – Gantt chart increment 5)

##### Increment 6

Graphical user interface

Description automatically generated

#### (Figure 25 – Gantt chart increment 6)

##### Increment 7

Graphical user interface

Description automatically generated

#### (Figure 26 – Gantt chart increment 7)

### 3.6.4 Resource Identification

This is the initial chose of Software going forward into the prototype phase. However, this can change as seen with the alternative plan in the [Appendix A](#_Appendix_A). The reason for choosing alternative resources is to provide flexibility to the plan if anything were to go wrong during the prototyping phase.

|  |  |
| --- | --- |
| **Resource Name** | **Description** |
| **Razor blade stealth** | Laptop for application development. |
| **GIT HUB** | To store versions of application to go back to previous ones if necessary or if software changes. Acts as back up for storage. |
| **Python** | Programming language. |
| **RASA** | AI chatbot development. Also used for automated testing. |
| **Kivy framework** | Development of mobile application. |
| **Pycharm/Visual studio code** | IDE |
| **MONGODB/MS Database** | Store user details. |
| **MS Word** | To write reports and record tests. |
| **MS Excel** | To create Gantt charts and record time effort. |

#### Table 21 – Resource identification table

## 3.7 Verification plan

Software verification is the method of determining whether the software of a given phase of the development process fulfils the requirements that have been agreed on during the previous phases. An example of specification testing is when the implemented function is compared to the user requirements that were agreed upon and examined to see if these requirements and functions align. The test will happen at the end of each sprint to ensure that those areas are completed before moving on to the next sprint. This is so that the test plan keeps track of the software development plan that has been chosen. [[36]](#_[36])

In [Appendix B](#_Appendix_B) test cases have been created for many of the tests down below. The reason for creating these tests is to prepare for future testing of the application. This will provide an understanding of what the initial system must have. Tests were not carried out individually, as some of the tests worked in tandem to cover all of these areas. These test cases have a chance of changing throughout the process but planning these test cases first helps to provide the developer with a clear structure for how these tests should be done and how they should be planned out if more is needed.

There are many types of tests that can be carried out when testing. For this only the testing types in the table below will be explained:

* Unit: for individual tests on smaller chunks of source code, like validation.
* System tests: for the completed integrated system product to evaluate the systems cooperation with requirements.
* Functional testing: For quality assurance to check that the correct output is happening.
* Regression testing: Testing that is repeated to ensure that functional and non-functional previously tested still performs as intended after changes to the software. This helps to display that no negative impacts have occurred during development.
* Manual testing: Testing the product your self-inputting information instead of doing it automatically.
* Negative Testing: Testing that is done to ensure that invalid data or actions are handled correctly and don’t produce errors. For example, this could be users trying to have the same username.

For the testing, a variety of black-box and white-box techniques will be used to produce the tests. White-box testing is used to test the internal programming of the application. The black box is where the internals is unknown to the tester and will be mostly viewed through the user inputs and outputs. For black box testing, the possibility of having someone else test the system provides the developer with a different viewpoint to allow for more features or improvements to the system. One form of testing that will be focused on in the second part of the project is automated testing. This can be used for the chatbot or the mobile application itself to speed up the testing process for the developer.

### 3.7.1 Verification plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Requirement tested ID | Test Case ID | Description | Test Type | Test Methodology | Result |
| T1 | F1 | TC1 | Must accept User input for chatbot. | Unit, Manual, integration, functional | User inputs text then sends it. |  |
| T2 | F2 | TC2 | Chatbot Must be responsive when the user asks it any questions. | Unit, manual, integration, functional | When the user inputs text the bot responds. |  |
| T3 | F3 | TC3 | Chatbot message and user messages display each time when the user enters in text and when the bot is expected to output it. | Unit, Manual, functional | When the user types or the chatbot outputs it is shown in the chatbot |  |
| T4 | F4 | TC4 | User must be able to navigate easily throughout the mobile application. | Unit, Manual | User can move from one page t o any page. |  |
| T5 | F5 | TC5 | User Must be able to ask for advice and receive advice on the appropriate situation the user is discussing. | Unit, Manual | When the user uses key word like depression the bot will be able to appropriately respond. |  |
| T6 | F6 | TC6 | User can sign in / log out of account through settings menu. | Unit, Manual, negative | User logs into/out account. |  |
| T7 | F7 | TC7 | User can Sign up to create an account at the start screen. | Unit, Manual, negative | User can create account from login page. |  |
| T8 | F8 | TC14 | Functional Mobile application. | Unit, Manual, Regression, System | A test that will most likely occur in each sprint and for the finale product. |  |
| T9 | F9 | TC15 | Functional Chatbot. | Unit, Manual, Regression, System | A test that will most likely occur in each sprint and for the finale product. |  |
| T10 | F10 | TC8 | Database Must store user login and details and can be possibly edited if the user needs to change. | Unit, Manual, negative | Most likely through mongo DB to store a wide arrange of user accounts. |  |
| T11 | F11 | TC8 | Save user information for signed in accounts and edit username and password. | Unit, Manual, negative | Mongo DB to storage into user account. |  |
| T12 | F12 | TC9 | Should allow user to enter specific profile details. | Unit, Manual | Name, age, gender any information that the user may find important. |  |
| T13 | F13 | TC5 | Advice must be Accurate to the user situation. | Unit, functional | Advice tailored to the subject of the user. |  |
| T14 | F14 | TC10 | Database should store a particular users chatlog. | Unit, Manual | Database to store chatlog for one user, or potentially store on the user’s device. |  |
| T15 | F15 | TC11 | Application could have advice page based on user situation. | Unit | Advice page based for the user on different type of subjects. |  |
| T16 | F16 | TC12 | Settings menu for user. | Unit, Manual. Regression | Simple setting menu to hold a small number of functions |  |
| T17 | F17 | TC13 | Chatbot could have text suggestions. | Unit, Manual, Regression | Another chatbot to suggest prompt that the user can tap on as an auto fill. |  |
| T18 | F18 | TC1, TC2, TC3, TC5 | Chatbot Should engage with the user first. | Unit | Chatbot prompts the user first. |  |
| T19 | F19 | N/A | Chatbot can understand broken text. | Unit, regression | Chatbot can understand slang or broken string. |  |
| T20 | F20 | TC12 | User Can clear chat logs from data base. | Unit, manual | User can remove all past chatlogs effectively resetting the chatbot. |  |
| T21 | F21 | TC12 | Light/Dark mode in the settings menu using slider. | Unit | Chang the background of the application. |  |
| T22 | F22 | N/A | The chatlogs could be saved to the cloud. | Unit | Possible feature, TBD |  |
| T23 | F23 | TC1, TC2, TC3, TC5 | Chatbot could send images or emojis along with the correct Text. | Unit, manual | Possible feature, TBD |  |
| T24 | F24 | TC12 | User could have profile picture show up when talking to the bot. | Unit, manual | Possible feature, TBD |  |

#### Table 23– Verification plan standard

### 3.7.2 Verification Continuous Testing

Continuous testing has been added to Function eight and nine. The reason for this is that through the sprints these are aspects that will need to be continuously tested. This is especially important for ‘F8’ as it tests the whole mobile application. This will be present throughout all the sprints whereas ‘F9’ will only be present for the chatbot stages of developments and the sprint to implements it into the application. Having this table and the test cases to go with it provides the developer with test going ahead in the development.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Requirement tested ID | Test Case ID | Description | Test Type | Test Methodology | Result |
| T8 | F8 | TC14 | Functional Mobile application. | Unit, Manual, Regression, System | A test that will most likely occur in each sprint and for the finale product. |  |
| T9 | F9 | TC15 | Functional Chatbot. | Unit, Manual, Regression, System | A test that will most likely occur in each sprint and for the finale product. |  |

#### Table 24– Verification plan Continuous testing

### 3.7.3 Verification Automated Testing

Automated tests can be carried out with the framework for the AI as most programming languages can build an automated test for the AI. This test will push commands into the chatbot and it will talk back like its talking to the user. The AI in turn can be trained by this process to better understand what inputs users will give it. By using this method, it is possible for the developer to check on the functionality of the chatbot and examine whether or not it understands the user situation, feeling and context. By using the test case the developer can create a test that is carried out exactly the same way as its steps. So the developer will only have to run one run command and look at the results instead of putting in multiple input.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Requirement tested ID | Test Case ID | Description | Test Type | Test Methodology | Result |
| T9 | F9 | TC15 | Functional Chatbot. | Unit, Manual, Regression, System | A test that will most likely occur in each sprint and for the finale product. |  |

#### Table 54 – Verification plan automated testing

### 3.7.3 User Survey

As seen in [Appendix C](#_Appendix_C) a survey has been created for the overall application. It does not go into specific details on every small aspect of the application but it does cover all the MUST HAVE user requirements. These are the most important to the application thus, must be covered thoroughly through testing and surveys to see if they have met their intended purpose. It is also possible down the line another survey is made to cover the smaller aspects of the application however this has been created so the developer can focus on testing the main requirements of the application. With this along with the rest of the testing plans should provide the developer the necessary steps to validate each function that has been added to the application and check if it meets the correct requirements.

### 

## 3.8 Validation Plan

Software Validation plans is the process of evaluating software at the end of its development cycle to check that it is free from any errors and meets the desired requirements. The failures are known as “incorrect product behaviour”. Validation is normally carried out through various testing methods like the Testing method listed above. [[36]](#_[36]) validation is needed as it will allow the developer to compare what they must what they need to have.

In this project validation will be carried out with multiple different approaches at the end of each sprint. First the normal test will be run. These will cover mostly unit and manual testing on the components added, then a system test will be run to see how all of the parts interact. Test cases will be used to test each feature added to the application. Then continuous testing will take place with each sprint to ensure that each part added does not interfere with other parts of the system. Automation testing will not be added to all of the sprints, it will only be added to sprints that focus on the AI chatbot. The last validation plan comes in the form of surveys. This can be used by other students, family members or mentors to display feedback to the creator on what areas of the application need improved upon. This survey in Appendix C can be used throughout the sprint, even though it covers areas like the chatbot that won’t be focused on until a later date. This survey will be used at the end of the development cycle to show feedback on the finale product. Thanks to the plans created and the development cycle used flexibility is added if features do need to be changed or can’t be met. This gives the developer the freedom to change tests or implement new tests. These plans will also allow the developer to quickly fix any short comings to the features that have been added. The testing process plan is intended to run like this:

Standard Test

Test Cases

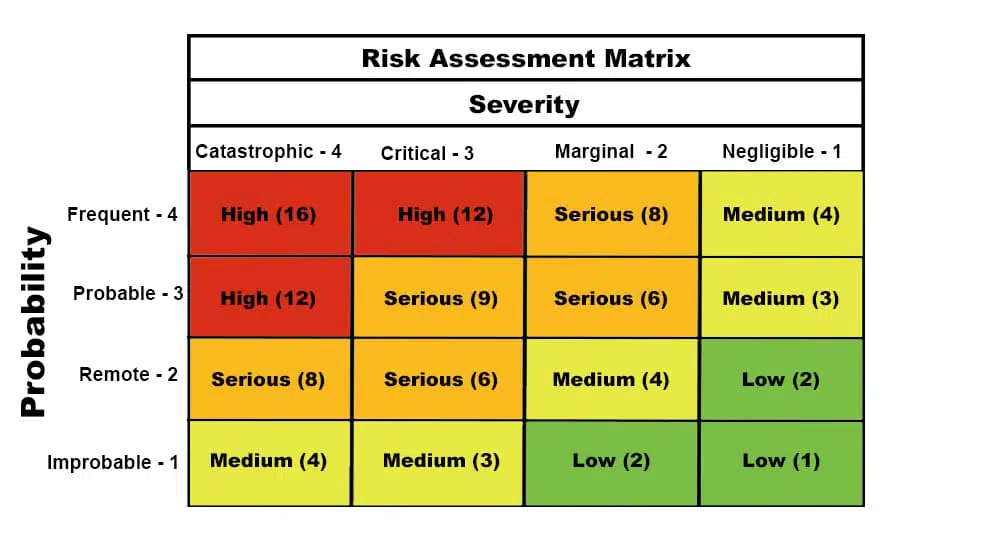
Continuous Testing

Automated Testing

Surveys

# Chapter 4 – Project Risk Assessment

## 4.1 Risk Assessment & mitigation strategy (Technical Risks)



#### Figure 30 - Risk Assessment Matrix [[37]](#_[37])

The project risk matric allows the developer to plot the likelihood of an event to occur from high to low. Using the matrix we can help to rank this risk from high to low, this helps to separate high severity from low severity. A risk matrix should be concise, simple and able to adapt to the project circumstances [[37]](#_[37]). Risk will be put into three zones:

* High risk (Red): Unacceptable
* Serious (Orange): nearly unacceptable
* Medium risk (Yellow): May or may not be acceptable
* Low Risk (Green): Acceptable

For this risk assessment values will be assigned to the risk. For example, if the severity of not having that function is high (4), and the probability of it happening is low (2) then using the matric we can multiply them 4 x 2 = 8, when looking at the matrix can be seen that this is a serious problem and thus will be coloured orange.

## 4.2 Risk Assessment Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk ID** | **Description** | **Threat Posed** | **Risk Threat Level** | **Risk Value** | **Mitigation Strategy** |
| R1 | The developer has little experience using the frontend python kivy. | An unusable front end renders the entire mobile application aspect unusable. Which impacts the intended requirements for the project. | **4 x 4 = 16** | **High** | The developer will use tutorials in order to gain a understanding of the steps needed to produces and acceptable front end. If the developer is unable to produce a suitable frontend other framework will be looked at. |
| R2 | The developer no experience setting up an AI framework before. | Results in low quality chatbot with no interaction capability or possible no chatbot at all. | **4 x 4 = 16** | **High** | The developer will go through tutorials on how these AI chatbots are set up and how to build interactions in the AI so that the developer an train it. |
| R3 | The AI chatbot has not been set up with simple talking command so that the developer can experiment with it. | As researched, for an AI chatbot to talk they need to be filled with commands to talk back to the user. If the developer is unable to set this up the chatbot won’t be responsive. | **4 x 2 = 8** | **Serious** | The developer will research the appropriate python setup used to produce and hold the chatbots text capabilities. (The developer has chosen two different AI strategies in RASA and python Chatterbot, Chatterbot is the back up to RASA in case the developer faces difficulties. As seen in [Appendix A](#_Appendix_A).) |
| R4 | The AI chatbot and frontend have not been linked together. | So the user can interact back and forth with each other, the user will be able to see their messages and the chatbots messages. This risk results in a unusable chatbot if set up correctly and producing results within its program. | **4 x 3 = 12** | **High** | The Developer will research and watch tutorials on how this has been achieved in similar solutions. A back up AI solution in python chatterbot has also been chosen to replace RASA if the developer struggles to implement it into the Kivy frontend. |
| R5 | Little experience using RASA. | As this is the first choice of the developer to produce a working AI chat, this would be important to learn so that it can be used within the project. | **3 x 3 = 9** | **Serious** | The developer has chosen two different AI strategies in RASA and python Chatterbot, Chatterbot is the back up to RASA in case the developer faces difficulties. As seen in [Appendix A](#_Appendix_A). |
| R6 | Finding a Data set for the API to pull to compliment the homepage to provide the user with appropriate information. | The homepage will need to be populated with data and given the topic of the application it may be hard to find a data set with appropriate data. | **1 x 2 = 2** | **Low** | This may cause the developer a little problem to find however, if an appropriate dataset does not exist it is possible for the developer to create their own data set just containing crucial information on certain topics. This approach is slow but also gives the developer a chance to understand the data they are work with better. |
| R7 | Storing user chat logs. | The developer has never stored messages in a system before. This could lead to user not having a past conversation | **3 x 3 = 9** | **Serious** | This will be having to be research by the developer later in the creation process when they have a better understanding of the application. This understanding could provide more insight on how the text can be stored and also gives the developer a chance to look at other solution to see how they’ve stored their data. |
| R8 | Unsure on API development when interacting with a mobile application. | Broken API or users being unable to pull data for logins or homepage. | **2 x 2.5 = 5** | **Moderate** | The developer over this past 4 months has gained and understanding of how APIs work and how to implement them. The developer will have to research how this can work with a mobile application and python kivy to display data. |
| R9 | Computer limitations when running tests. | Laptop may not be able to run automated testing efficiently or new advancements like RASA. | **2.5 x 2 = 5** | **Moderate** | The developer is using a laptop to create all the project. Unfamiliar software could potentially cause the laptop to crash as it has done during other University modules. If this occurrence happens using community libraries or ulster resources may have to be the alternative, although travel consideration will have to be considered. |
| R10 | Outside life aspects interfering with coursework. | Travel, work and other obligations like doctor appointments can lead to rushed development of the project leading to broken functions or functions not being added. | **2 x 3 = 6** | **Serious** | Since most information needed can be accessed online all the developer will need is an internet access to prevent some outside occurrences from happening and since it’s a laptop the developer can work from most places. |

#### Table 55 – Risk Assessment table

# Chapter 5 – initial Functional Prototype

## 5.1 Justification of risk selection

The project will have an early prototype built to give the developer an idea of how the project will be going forward. This will help to address any risks that the developer thinks they have and to help show any new risks early that have not been spotted. This prototype will be a learning experience for the developer as all frameworks being implemented have never been used by the developer. This prototype aims to get familiar with the front end and the AI chatbot's back end. The prototype will be far from the product but will be used as a stepping stone.

The first risk to be addressed is Risk One. Risk one will be addressed so the developer can learn how to use python Kivy to set up a mobile application. Without this, half of the project will not be completed. Also, as this is the developer's first time, the prototype will most likely present a basic application with some hard codded elements like a login to simulate what the application could be like going forward.

The second risk to be addressed is Risk Two. Risk two will address the setup and use of the AI chatbot. This will set up two different chatbot frameworks, one RASA and the other Python Chatterbot. Setting up two may seem like a waste of time; however, having two chatbots means having a backup ready in case one does not work how it is meant to. RASA is different from Chatterbot and even provides more features than it does. However, Chatterbot is more simplistic and easier to understand, which can help the developer with time management and meeting sprints on time. There may also be more information and solution for Chatterbot than RASA. The third risk to be partially implemented also factors in the choice of chatbot.

The third risk to be partially addressed is Risk Four. Risk four is linking the AI up to the front end. The reason why this is expected to only be partially addressed is that this takes knowledge of both python Kivy and one of the AI chatbot frameworks. This risk is the reason why two AI chatbot frameworks were chosen. Having two options provides the developer with flexibility when it comes to researching the process. This has the potential to be fully addressed; however, that will depend on the developer's progress when learning python kivy and the AI framework.

By addressing these risks, the developer can gain a higher understanding of how the rest of the project will come together, as these risks address the project's main components.

|  |  |
| --- | --- |
| Risk ID | Description |
| R1 | **Fully Addressed** - The developer has little experience using the frontend python kivy. |
| R2 | **Fully Addressed** - The developer no experience setting up an AI framework before. |
| R3 | **Partially Addressed** - The AI chatbot and frontend have not been linked together |

#### Table 56 – Risk Justification table

## 5.2 Design Artefacts

### 5.2.1 – Kivy App Life Cycle

[Figure 31](#_Figure_31_–) displays the cycle that an application will have created and run. First, the application is run from MyApp(). run() line of code in the main class. It will then build itself and start the application using the Def build(self) line of code, which can be viewed in the developer's project. This will be able to display all the frontend of the application as well as work with the back end of the application. The developer will then be able to see the results from the created code and have the option to kill the run or access other project features. This simple and easy-to-understand plan from python Kivy shows why this specific framework was picked for this project. Although more experimentation will have to come from the developer's side to create the application python, Kivy covers all the areas the project will need to produce the frontend of the project.

As the project developer has experience with python, they will be able to understand aspects of kivy. However, more work will need to be done to produce an industry-standard application. Thankfully learning the basics of how kivy runs gives the developer a much bigger understanding of creating their application. In addition, Kivy allows the developer to view their work instantly to make quick changes to the application. This helps when working on deadlines to complete sprints or with quick development methodologies like agile.

This diagram helps to break down the code needed to run the application and the possible processes that can happen when the developer or users interacts with the application.

Diagram

Description automatically generated

#### Figure 31 – Kivy App Life Cycle [[38]](#_[38])

### 5.2.2 – Kivy Architecture

Kivy consists of several building blocks, these are:

* Core providers and input providers – A piece of code that utilizes a certain API to communicate with the operating system on one side and kivy on the other side, which will act as a communication layer, this is what is called a core providers. This make kivy easier to package with it effectively reducing the size of distribution. A input provider is a piece of code that adds support for a certain input device such as mouses and track pads. [[39]](#_[39])
* Graphics – Kivy issues hardware-accelerated drawing commands using OpenGL.
* Core – The core package helps to provide features like clock for timer events, cache, Feature detection used to detect various types of strokes like circles, kivy language and properties which link the widget code to the user interface.
* UIX – Contains widgets and layouts that the developer can reuse to rapidly create a user interface .
* Modules – used to inject functionality into the ivy application even if the original developer did not use it.
* Input events (Touches) – Abstracts different input types and sources them as touches. Use the Touch() class to inform the application of the users input.
* Widgets and event dispatching – widgets are objects that receives input events. When a input is available kivy send out one event per touch.

Kivy architecture provides the developer with support when developing an application, this was one of the main reason why kivy was picked to create the project. This prototype will display some of the key components of kivy architecture, and opens of the possibility of additional functionality that the developer never considered. The main key building blocks that will be used is touch, widgets with inputs, UIX along with core providers.

Table

Description automatically generated

#### Figure 32 – Kivy Architecture Diagram [[39]](#_[39])

#### 5.2.3 – Frontend

Python kivy front end can be created into different forms. First, it can be incorporated through the main application file, for example, app.py. However, this can make the application confusing to understand as it grows. The next option is a KV file. This file act like a CSS file does to HTML, except the KV files can add the input, buttons, and labels to the application. This can be linked to the main application through its class. For example, [Figure 33](#_Figure_33_–) is the class used to interact with the KV file, as is the class returned in the MyApp class. This is where the application is built. When the file is building, it will automatically try to find a KV file that will go with it, and this is where it will find my.kv file in [Figure 34](#_Figure_34_–) will then build the code in the kv file to build the front end, as seen in [Figure 35](#_Figure_35_–).

These are all files that the developer is experimenting with and testing to help learn how to use Kivy and address the risk R1 In [table 56](#_Table_56_–). The developer will be able to expand on these files to push their understanding of kivy further and showcase that they have learned the basics of the application. Using the kv file, the developer has made the main application easier for them to understand. With past knowledge of how HTML interacts with CSS, the developer has familiarity with how components like this should interact. This gives the developer a slight edge when progressing into the project in the future. It is likely, though, that the frontend of the user will be at a basic level for the prototype.

Text

Description automatically generatedText

Description automatically generated

#### Figure 33 – Classes from app.py

Graphical user interface, application

Description automatically generated

#### Figure 34 – my.kv file to style output

#### Figure 35 – Output from KV and class file

#### 5.2.3 – RASA Architecture Overview

The two main components of RASA architecture are Natural Language Understanding (NLU) and dialogue management. NLU will help to handle intent classification, response retrieval, and entity extraction. This is why it is shown as an NLU pipeline, as it will process the user's utterances using the NLU model generated and trained by the pipeline. Dialogue management helps decide the following action in the conversation based on the given context the user has created. This is shown in the dialogue policies in the diagram. These components communicate back and forth with the agent as it interacts with the rest of the components.

This was the main reason RASA was considered for the project when covering the AI side of the application. RASA has many advanced features that can help the developer produce a comprehensive chatbot that can be easily trained and quickly produced into the application helping meet the timeline for each given sprint. RASA action severs also lets the application run live, which may be a consideration further into the project's development and a potential advanced feature that could be focused on; this also allows training from multiple users anywhere.

Diagram

Description automatically generated

#### Figure 36 – RASA Architecture Overview

#### 5.2.4 – Wire Frame

##### 5.2.4.1 – Login Drawn/computer insert

Graphical user interface, diagram

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

###### Figure 38 – Phone concept Sign in

###### Figure 37 – Wireframe Login

##### 5.2.4.2 – Sign Up Drawn/computer insert

Diagram

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

###### Figure 39 – Wireframe Sign up

###### Figure 40 –Phone concept Sign up

##### 5.2.4.3 – Home screen Drawn/computer insert

A picture containing whiteboard

Description automatically generatedGraphical user interface, text, application

Description automatically generated

###### Figure 41 – Wireframe Home

###### Figure 42 – Phone concept Home

##### 5.2.4.4 – AI Chatbot Drawn/computer insert

A picture containing text, screenshot

Description automatically generatedGraphical user interface, application

Description automatically generated

###### Figure 43 – Wireframe Chat screen

###### Figure 44 – Phone concept Chat screen

##### 5.2.4.4 – Settings Drawn/computer insert

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

###### Figure 45 – Wireframe Settings

###### Figure 46 – Phone concept Settings

#### 5.2.4 – Python chatterbot Logic flow

Python chatterbot is the AI framework beside RASA that the developer is working with as a backup in case issues occur when using RASA. Chatterbot is an excellent backup as it's more simplistic than RASA, even though RASA provides more features and versions when compared to chatterbot.

Chatterbot flow is simple, and it takes the inputs, then processes that input, uses the logic adapters to pick the best response of that chosen input then returns that processed response to the input. This flow will happen every time the user tries to communicate using the application. Using [Figure 44](#_Figure_44_–), the process of that flow can be seen, and the user inputs something, then the chatbot can respond to it within one or two messages then the user can enter a new input for the chatbot to respond to.

Text

Description automatically generated

#### Figure 47 – Python chatter bot logic flow

# Chapter 6 - References

#### [1]

Bertolote J. The roots of the concept of mental health. World Psychiatry. 2008;7(2):113-6.

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2408392/>

Accessed [19/10/2022]

#### [2]

Menniger, Dr.W.C., Norlund, Dr.E.B., Rodger, Dr.T.F., Roudinesco, M.J., de Ruyter, Dr.TH.H., Pacheco e Silva, Dr.A.C., Soddy, Dr.K. and Zimmerman, Dr.K. (1951). *Expert Committee On Mental Health*. [online] *WHO\_TRS*, Geneva: World Health Organisation, pp.2–5.

<https://apps.who.int/iris/bitstream/handle/10665/37982/WHO_TRS_31.pdf>

[Accessed 19 Oct. 2022].

#### [3]

‌World Health organisation (2021). *Suicide Worldwide in 2019*. [online] World Health Organisation.

<https://www.who.int/publications/i/item/9789240026643>

[Accessed 19 Oct. 2022].

#### ‌[4]

Abd-alrazaq, A.A. et al. (2019) An overview of the features of Chatbots in Mental Health: A scoping review, International Journal of Medical Informatics. Elsevier. <https://reader.elsevier.com/reader/sd/pii/S1386505619307166?token=07A7C01F014BB0CAA40D3EB0A3C160585A463D2219E57E56680E393AB8A6462504A0FA4ACB922C7ECC39047DC8B8F16> (Accessed: October 20, 2022).

#### [5]

Chowdhary, K.R. (2020). Natural Language Processing. *Fundamentals of Artificial Intelligence*, pp.603–649.

<https://link.springer.com/chapter/10.1007/978-81-322-3972-7_19>

[Accessed 22/10/2022]

#### ‌[6]

Eisenstein, J. (2019). *Introduction to Natural Language Processing*. [online] *Google Books*. MIT Press. Page 1 - 3

<https://books.google.co.uk/books?hl=en&lr=&id=72yuDwAAQBAJ&oi=fnd&pg=PR5&dq=natural+language+processing+introduction&ots=gWaO114dq5&sig=nOHr2xDVYVALdYcvuLOzuQ5fI-k&redir_esc=y#v=onepage&q&f=false>

[Accessed 22/10/2022]

#### ‌[7]

Cercone, N. (1977). Morphological Analysis and Lexicon Design for Natural-Language Processing. *Computers and the Humanities*, 11(4), pp.2-10.

<https://www.jstor.org/stable/30199902>

[Accessed 23/10/2022].

#### ‌[8]

Anon, (2020). *Stages of Natural Language Processing (NLP) | byteiota*. Pg 1- 7

<https://byteiota.com/stages-ofnlp/#:~:text=Morphological%20or%20Lexical%20Analysis%20deals>

[Accessed 23/10/2022].

#### ‌[9]

Sun, P., Yang, X., Zhao, X. and Wang, Z. (2018). *An Overview of Named Entity Recognition*. Pg 1 - 6

<https://ieeexplore.ieee.org/abstract/document/8629225>

[Accessed 23/10/2022].

#### [10]

Dang, N.C., Moreno-García, M.N. and De la Prieta, F. (2020). Sentiment Analysis Based on Deep Learning: A Comparative Study. *Electronics*, 9(3), Pg 1-10

<https://www.mdpi.com/2079-9292/9/3/483>

[Accessed 23/10/2022].

#### [11]

Kao, A. and Poteet, S.R. (2007). *Natural Language Processing and Text Mining*. *Google Books*. Springer Science & Business Media. Pg 1 - 5

<https://books.google.co.uk/books?hl=en&lr=&id=CVtxFWbKT7wC&oi=fnd&pg=PA1&dq=NLP+text+mining+introduction&ots=vNAUzfFWoe&sig=W1umflubQLngAQpkoYC1AVQKoFU&redir_esc=y#v=onepage&q=NLP%20text%20mining%20introduction&f=false>

[Accessed 23/10/2022].

#### ‌[12]

Linguamatics. (2019). *What is Text Mining, Text Analytics and Natural Language Processing?*

<https://www.linguamatics.com/what-text-mining-text-analytics-and-natural-language-processing#:~:text=Text%20mining%20(also%20referred%20to>.

[Accessed 23/10/2022].

#### ‌[13]

www.rosoka.com. (n.d.). *Types of Natural language processing techniques | Rosoka*. <https://www.rosoka.com/blog/types-natural-language-processing-techniques>.

[Accessed 23/10/2022].

#### [14]

Wang, W. and Siau, K. (2019). Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity. *Journal of Database Management*, 30(1), pp. Pg 1 -10

<https://www.igi-global.com/article/artificial-intelligence-machine-learning-automation-robotics-future-of-work-and-future-of-humanity/230295>

[Accessed 23/10/2022].

#### [15]

Rivas, P., Holzmayer, K., Hernandez, C. and Grippaldi, C. (2018). *Excitement and Concerns about Machine Learning-Based Chatbots and Talkbots: A Survey*. [online] IEEE Xplore. doi:10.1109/ISTAS.2018.8638280.

<https://ieeexplore.ieee.org/abstract/document/8638280>

[Accessed 23/10/2022].

#### [16]

Zhang, Y. (2010). *New Advances in Machine Learning*. [online] *Google Books*. BoD – Books on Demand. Pg 1- 10

<https://books.google.co.uk/books?hl=en&lr=&id=XAqhDwAAQBAJ&oi=fnd&pg=PA19&dq=types+of+machine+learning&ots=r2LocUFdQt&sig=v_k5xsi4Pkd8_zS-1Eu5tkdrUrM&redir_esc=y#v=onepage&q=types%20of%20machine%20learning&f=false>

[Accessed 23/10/2022].

#### ‌[17]

van Engelen, J.E. and Hoos, H.H. (2019). A survey on semi-supervised learning. *Machine Learning*. doi:10.1007/s10994-019-05855-6.

<https://link.springer.com/article/10.1007/s10994-019-05855-6>

[Accessed 24/10/2022].

#### [18]

Gupta, A., Hathwar, D. and Vijayakumar, A. (n.d.). *Introduction to AI Chatbots*. [online] Pg 1 - 4

<https://pdfs.semanticscholar.org/f5f4/746acffef08df37f184cb6acc0505362ea9b.pdf>

[Accessed 24/10/2022].‌

#### [19]

AWS (2019). *Mobile Application Development*. [online] Amazon Web Services, Inc. <https://aws.amazon.com/mobile/mobile-application-development/>

[Accessed 24/10/2022].

#### [20]

Fitzpatrick, K.K., Darcy, A. and Vierhile, M. (2017). Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial. *JMIR Mental Health*, [online] 4(2), p.e19. doi:10.2196/mental.7785.

<https://mental.jmir.org/2017/2/e19>

[Accessed 24/10/2022].

#### [21]

Prochaska, J.J., Vogel, E.A., Chieng, A., Kendra, M., Baiocchi, M., Pajarito, S. and Robinson, A. (2021). A Therapeutic Relational Agent for Reducing Problematic Substance Use (Woebot): Development and Usability Study. *Journal of Medical Internet Research*, 23(3), p.e24850. doi:10.2196/24850.

<https://www.jmir.org/2021/3/e24850/>

[Accessed 24/10/2022].

#### [22]

Inkster, B., Sarda, S. and Subramanian, V. (2018). An Empathy-Driven, Conversational Artificial Intelligence Agent (Wysa) for Digital Mental Well-Being: Real-World Data Evaluation Mixed-Methods Study. *JMIR mHealth and uHealth*,

<https://mhealth.jmir.org/2018/11/e12106/>

[Accessed 24/10/2022].

#### [23]

Wysa - your safe space in this difficult time. (n.d.). *FAQ*.

<https://www.wysa.io/faq>

[Accessed 24/10/2022].

#### [24]

Mehta, A., Niles, A.N., Vargas, J.H., Marafon, T., Couto, D.D. and Gross, J.J. (2021). Acceptability and Effectiveness of Artificial Intelligence Therapy for Anxiety and Depression (Youper): Longitudinal Observational Study. *Journal of Medical Internet Research*

<https://www.jmir.org/2021/6/e26771/>

[Accessed 24/10/2022].

#### [25]

youper.ai. (n.d.). *Our Tech - Youper*.

<https://www.youper.ai/our-tech#:~:text=Powered%20by%20artificial%20intelligence%20to>

[Accessed 25/10/2022]

#### ‌[26]

Smith, L. W. (2000). Stakeholder analysis: a pivotal practice of successful projects. Paper presented at Project Management Institute Annual Seminars & Symposium, Houston, TX. Newtown Square, PA: Project Management Institute.

<https://www.pmi.org/learning/library/stakeholder-analysis-pivotal-practice-projects-8905>

[Accessed 27/10/2022].

#### [27]

‌Lucassen, G., Dalpiaz, F., Werf, J.M.E.M. van der and Brinkkemper, S. (2016). The Use and Effectiveness of User Stories in Practice. *Requirements Engineering: Foundation for Software Quality Pg 209 - 215*

<https://link.springer.com/chapter/10.1007/978-3-319-30282-9_14>

[Accessed 28/10/2022].

#### [28]

Xp, A. and Cohn, M. (2004). *User Stories Applied For Agile Software Development*.

<https://www.mountaingoatsoftware.com/uploads/presentations/User-Stories-Applied-Agile-Software-Development-XP-Atlanta-2004.pdf>.

[Accessed 28/10/2022].

#### [29]

Waters, K. (2009). *Prioritization using MoSCoW*. [online] Available at:

<https://comp.anu.edu.au/courses/comp3120/local_docs/readings/Prioritisation_using_MoSCoW_AllAboutAgile.pdf>

[Accessed 31/10/2022].

#### [30]

‌Khan, J.A., Rehman, I.U., Khan, Y.H., Khan, I.J. and Rashid, S., 2015. Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique. *International Journal of Modern Education & Computer Science*, *7*(11).

[Accessed 31/10/2022].

#### [31]

First Things First: Prioritizing Requirements 1. (n.d.). Karl E. Wiegers

<http://www.tarrani.net/linda/prioritizing.pdf>.

[Accessed 31/10/2022].

#### [32]

DEV Community. (n.d.). *User Story Guidelines*. [online] Available at:

<https://dev.to/code2bits/user-story-guidelines-l2e>

[Accessed 2 Nov. 2022].

#### [33]

Jaccheri, L. ed., (2004). *The Benefits and Limitations of Knowledge Management in Global Software Development*. [online] IET Digtial Library. Pg 22 - 25.

<https://digital-library.theiet.org/content/conferences/10.1049/ic_20040312>

[Accessed 2 Nov. 2022].

#### ‌[34]

Plutora. (2019). *Software Development Life Cycle (SDLC): Making Sense of the Different Methodologies*. [online]

<https://www.plutora.com/blog/software-development-life-cycle-making-sense-of-the-different-methodologies>

[Accessed 2 Nov. 2022].

#### [35]

Wrike. (n.d.). *What is Iterative and Incremental Development?*

<https://www.wrike.com/project-management-guide/faq/what-is-iterative-incremental-development/>

[Accessed 2 Nov. 2022].

#### [36]

apps.dtic.mil. (n.d.). *Introduction to Software Verification and Validation*. [online]

Available at: <https://apps.dtic.mil/sti/citations/ADA236117>

[Accessed 2 Nov. 2022].

#### [37]

Anon, (2022). *Risk Assessment Matrix: Definition, Examples, and Templates |*. [online] Available at: <https://pmstudycircle.com/risk-assessment-matrix/>

[Accessed 20 Nov. 2022].

#### [38]

Kivy.org. (2019). *Kivy Basics — Kivy 1.11.1 documentation*. [online]

Available at: <https://kivy.org/doc/stable/guide/basic.html>

[Accessed 15 dec. 2022].

#### [39]

‌ kivy.org. (n.d.). *Architectural Overview — Kivy 2.1.0 documentation*. [online]

Available at: <https://kivy.org/doc/stable/guide/architecture.html>

[Accessed 15 dec. 2022].‌

# Bibliography

Used for learning when doing particle:

kivy.org. (n.d.). *Kivy Basics — Kivy 2.1.0 documentation*. [online] Available at: <https://kivy.org/doc/stable/guide/basic.html#kivy-app-life-cycle>

[Accessed 20 dec. 2023].

‌techwithtim.net. (n.d.). *Python Kivy Example GUI Tutorial*. [online] Available at: <https://www.techwithtim.net/tutorials/kivy-tutorial/example-gui/>

[Accessed 20 dec. 2023].

upGrad blog. (2020). *How to Make a Chatbot in Python Step By Step [Python Chatterbox Guide]*. [online] Available at: <https://www.upgrad.com/blog/how-to-make-chatbot-in-python/>

[Accessed 20 dec. 2023].

rasa.com. (n.d.). *Installing Rasa Open Source*. [online] Available at: <https://rasa.com/docs/rasa/installation/installing-rasa-open-source/>

[Accessed 20 dec. 2023].

‌

‌

‌

# ‌Chapter 7 - Appendix

## Appendix A

To cover for possible errors o RASA not correctly working on developer machine, this is the main concern when using this method of AI chatbot.

|  |  |
| --- | --- |
| **Resource Name** | **Description** |
| **Razor blade stealth** | Laptop for application development. |
| **GIT HUB** | To store versions of application to go back to previous ones if necessary or if software changes. Acts as back up for storage. |
| **Python** | Programming language. |
| **Python Chatterbot** | AI chatbot development. Links better with python kivy providing the developer time to focus on other areas of the project. |
| **Kivy framework** | Development of mobile application. |
| **Pycharm/Visual studio code** | IDE |
| **MONGODB/MS Database** | Store user details. |
| **Microsoft Azure** | To store chatbot speech. |
| **MS Word** | To write reports and record tests. |
| **MS Excel** | To create Gantt charts and record time effort. |

#### Table 22 – Resource identification table

## Appendix B

### 3.7.1 Verification plan Test cases

These are test cases that will/can be used through the project duration. These test cases will likely change over time but have the base built for each test chase should show an understanding of what is expected from each feature. Test cases were not done for a majority of the could haves as these work in tangent with some of the test cases. The test cases covered are the amin part of the system.

|  |  |
| --- | --- |
| **Test Case ID:** TC1 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** USER INPUT | **Test Executed by:** Ryan Harper |
| **Test Name:** User can enter text into chatbot | **Test Executed Date:** TBD |
| **For Test ID:** T1 | **Dependencies:** F1 / F3 / F4 / F9 / F6 / F18 / F23 |
| **Description:** Testing that the user can enter information into the text bar and submit it. | **Precondition:** User has something typed into the chat bar. |

#### Table 25 – Test Case 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Login | Username: user  Password: Password123! | User logs in |  |  |  |
| 2 | Navigate to chatbot screen | N/A | Chatbot pages opens |  |  |  |
| 3 | Click on chat bar and type | Hello | Chat bar accepts input. |  |  |  |
| 4 | Click send | N/A | Allows for information to be sent into chat box and chat bot responds |  |  |  |

#### Table 26 – Test Case 1

|  |  |
| --- | --- |
| **Test Case ID:** TC2 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Responsive chatbot | **Test Executed by:** Ryan Harper |
| **Test Name:** Responsive chatbot | **Test Executed Date:** TBD |
| **For Test ID:** T2 | **Dependencies:** F2 / F3/ F1 / F18 / F23 |
| **Description:** Chatbot Must be responsive when the user asks it any questions. | **Precondition:** User is logged in and in the chatbot screen |

#### Table 27 – Test Case 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Click on chat bar and type | Hello | Chat bar accepts input. |  |  |  |
| 2 | Click send | N/A | Users text is shown. |  |  |  |
| 3 | Chat bot responds. | N/A | Chatbots text should be shown replying to the user. |  |  |  |

#### Table 28 – Test Case 2

|  |  |
| --- | --- |
| **Test Case ID:** TC3 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Message Display | **Test Executed by:** Ryan Harper |
| **Test Name:** Displaying chatbot and user message | **Test Executed Date:** TBD |
| **For Test ID:** T3 | **Dependencies:** F2 / F3/ F1 / F18 / F23 |
| **Description:** Chatbot message and user messages display each time when the user enters in text and when the bot is expected to output it. | **Precondition:** User is logged in and in the chatbot screen |

#### Table 29 – Test Case 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Click on chat bar and type | Hello | Chat bar accepts input. |  |  |  |
| 2 | Click send | N/A | Users text is shown. |  |  |  |
| 3 | Chat bot responds. | N/A | Chatbots text should be shown replying to the user. |  |  |  |
| 4 | Check that both user and chatbot text can be seen | N/A | Both text input can be seen. |  |  |  |

#### Table 30 – Test Case 3

|  |  |
| --- | --- |
| **Test Case ID:** TC4 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Navigation | **Test Executed by:** Ryan Harper |
| **Test Name:** Easy navigation | **Test Executed Date:** TBD |
| **For Test ID:** T4 | **Dependencies:** F4 / F6 / F7 |
| **Description:** User must be able to navigate easily throughout the mobile application. | **Precondition:** User has an account |

#### Table 31 – Test Case 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Login | Username: user  Password: Password123! | User Logs into page |  |  |  |
| 2 | User goes from home screen to chatbot. | N/A | Opens chatbot |  |  |  |
| 3 | User goes from chatbot to home screen. | N/A | Home screen opens |  |  |  |
| 4 | User logs out. | N/A | User goes back to login page. |  |  |  |
| 5 | User clicks sign up |  | Sing up page appears |  |  |  |
| 6 | User creates account | Username: user2  Password: Password123! | Brought back to login |  |  |  |
| 7 | Users signs in with new account | Username: user2  Password: Password123! | User is met with home screen. |  |  |  |
| 8 | Checks profile | N/A | Opens profile |  |  |  |
| 9 | Open settings | N/A | Settings open |  |  |  |

#### Table 32 – Test Case 4

|  |  |
| --- | --- |
| **Test Case ID:** TC5 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Receive advice | **Test Executed by:** Ryan Harper |
| **Test Name:** User receives advice when directly asking for it | **Test Executed Date:** TBD |
| **For Test ID:** T5 / T13 | **Dependencies:** F5 / F1 / F2 / F3 / F18 / F23 |
| **Description:** User Must be able to ask for advice and receive advice on the appropriate situation the user is discussing. | **Precondition:** User is logged in and in the chatbot screen |

#### Table 33 – Test Case 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Click on chat bar and type | “Advice on depression” | Chat bar accepts input. |  |  |  |
| 2 | Chatbot goes through its procedure for specific request. | N/A | Chatbot outputs any information related to depression that it has. |  |  |  |
| 3 | Chat bot prompts user if they would like to know more. If yes. | “Yes” | Chatbot outputs more |  |  |  |
| 4 | Chat bot prompts user if they would like to know more. If no. | “No” | Chatbot continues interacting with the user on other areas. |  |  |  |

#### Table 34 – Test Case 5

|  |  |
| --- | --- |
| **Test Case ID:** TC6 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** LOGIN | **Test Executed by:** Ryan Harper |
| **Test Name:** Can the user login | **Test Executed Date:** TBD |
| **For Test ID:** T6 | **Dependencies:** F6 / F7 |
| **Description:** User can sign in / log out of account through settings menu. | **Precondition:** Application is opened and on login screen. |

#### Table 35 – Test Case 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User Enters invalid username | Username: user12121  Password: Password123! | Prompts user with error that username or password is incorrect |  |  |  |
| 2 | User Enters invalid password | Username: user  Password: Password | Prompts user with error that username or password is incorrect |  |  |  |
| 3 | User enters valid username and password | Username: user  Password: Password123! | User is logged into home screen able to use all application features. |  |  |  |

#### Table 36 – Test Case 6

|  |  |
| --- | --- |
| **Test Case ID:** TC7 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** SIGNUP | **Test Executed by:** Ryan Harper |
| **Test Name:** Can the user sign up to application | **Test Executed Date:** TBD |
| **For Test ID:** T7 | **Dependencies:** F6 / F7 |
| **Description:** User can Sign up to create an account at the start screen. | **Precondition:** Application is opened and on login screen. |

#### Table 37 – Test Case 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User clicks on signup | N/A | Brought to sign up page. |  |  |  |
| 2 | User Enters a username already chosen | Username: user | Prompts user with error that the username is already chosen. |  |  |  |
| 3 | User Enters weak password | Password: Password | Prompts user with error that the password is weak and needs symbols or capitols |  |  |  |
| 4 | User enters valid username and password | Username: user3  Password: Password123! | Credentials are accepted. Bringing them back to the login page |  |  |  |
| 5 | User enters username and password | Username: user3  Password: Password123! | User is logged in. |  |  |  |

#### Table 38 – Test Case 7

|  |  |
| --- | --- |
| **Test Case ID:** TC8 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** store user login and details | **Test Executed by:** Ryan Harper |
| **Test Name:** store user login and details | **Test Executed Date:** TBD |
| **For Test ID:** T10 / T11 | **Dependencies: F6 / F7 / F10 / F11** |
| **Description:** Database Must store user login and details and can be possibly edited if the user needs to change. | **Precondition:** Application is opened. |

#### Table 39 – Test Case 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User enters valid username and password | Username: user6  Password: Password123! | User is logged into home screen able to use all application features. |  |  |  |
| 2 | Check Database for login | N/A | User details should be entered. |  |  |  |
| 3 | Go to forgot password in login page | N/A | Edit login detail page appears. |  |  |  |
| 4 | Go to profile page | N/A | Change password |  |  |  |
| 5 | Type in old password | Password123! | New password section should appear |  |  |  |
| 6 | Type in new password | Password123# | Password changes |  |  |  |

#### Table 40 – Test Case 8

|  |  |
| --- | --- |
| **Test Case ID:** TC9 | **Test Designed by:** Ryan Harper |
| **Test Priority:** MEDIUM | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Profile Details | **Test Executed by:** Ryan Harper |
| **Test Name:** Enter profile details | **Test Executed Date:** TBD |
| **For Test ID:** T12 | **Dependencies: F12** |
| **Description:** Should allow user to enter specific profile details. | **Precondition:** User is logged in on profile page. |

#### Table 41 – Test Case 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User Can enter their age | 23 | Accepts input |  |  |  |
| 2 | User can enter real name | Blah Blah | Accepts input |  |  |  |
| 3 | User can enter gender | Male | Accepts input |  |  |  |
| 4 | User clicks save | N/A | Sends profile information to database. |  |  |  |
| 5 | Check database to see if users’ profiles has been updated. | N/A | Database has been updated. |  |  |  |

#### Table 42 – Test Case 9

|  |  |
| --- | --- |
| **Test Case ID:** TC10 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Chatlog | **Test Executed by:** Ryan Harper |
| **Test Name:** store a particular users chatlog | **Test Executed Date:** TBD |
| **For Test ID:** T14 | **Dependencies:** F14 |
| **Description:** Database should store a particular users chatlog. | **Precondition:** User is logged in on Chat box. |

#### Table 43 – Test Case 10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User inputs text | Hello | Hello is outputted |  |  |  |
| 2 | Chatbot responds | N/A | Chat is outputted |  |  |  |
| 3 | Check that text got added to database | N/A | Check the user accounts that it happened on |  |  |  |
| 4 | View current chat log | N/A | Chatlog appears |  |  |  |
| 5 | Return to application | N/A | Chat box should still display chat logs in database. |  |  |  |

#### Table 44 – Test Case 10

|  |  |
| --- | --- |
| **Test Case ID:** TC11 | **Test Designed by:** Ryan Harper |
| **Test Priority:** MEDIUM | **Test Designed Date:** 25/11/2022 |
| **Module Name:** API / Homepage | **Test Executed by:** Ryan Harper |
| **Test Name:** API / home page | **Test Executed Date:** TBD |
| **For Test ID:** T15 | **Dependencies:** F15 |
| **Description:** Application could have advice page based on user situation. | **Precondition:** User is logged in on home page |

#### Table 45 – Test Case 11

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Scroll throughout home page. | N/A | User should be able to scroll up and down |  |  |  |
| 2 | Possibility of clicking into section to expand it for more information. | N/A | Information expands downwards to allow the user to see more information |  |  |  |
| 3 | Go from page to page | Click | User can go from one page to another to view information |  |  |  |

#### Table 46 - Test Case 11

|  |  |
| --- | --- |
| **Test Case ID:** TC12 | **Test Designed by:** Ryan Harper |
| **Test Priority:** MEDIUM | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Settings | **Test Executed by:** Ryan Harper |
| **Test Name:** Menu for Settings page | **Test Executed Date:** TBD |
| **For Test ID:** T16 | **Dependencies:** F16 / F20 / F21 / F24 |
| **Description:** Settings menu for user. | **Precondition:** User is logged in on home page |

#### Table 47 - Test Case 12

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Click gear icon on top right-hand corner. | Click | Takes the user to the settings menu |  |  |  |
| 2 | User can change background from light to dark mode. | Click | Changes to dark mode. |  |  |  |
| 3 | Change profile pic of user for chat bot | Insert image | Image icon appears on user messages. |  |  |  |
| 4 | User can clear chatlogs with chatbot | Click | Removes chatlog from database. |  |  |  |
| 5 | User can delete account | Click | Users account is removed. Returns to home page. |  |  |  |

#### Table 48 - Test Case 12

|  |  |
| --- | --- |
| **Test Case ID:** TC13 | **Test Designed by:** Ryan Harper |
| **Test Priority:** MEDIUM | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Suggestion | **Test Executed by:** Ryan Harper |
| **Test Name:** Text Suggestion | **Test Executed Date:** TBD |
| **For Test ID:** T17 | **Dependencies:** F17 |
| **Description:** Chatbot could have text suggestions. | **Precondition:** User is logged in on Chatbot |

#### Table 49 - Test Case 13

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User Starts conversation | Hello | Users message appears in chat. |  |  |  |
| 2 | Chat bot responds | N/A | Appears as message in chat |  |  |  |
| 3 | User will then be prompted with chat messages that they can ask. | N/A | Sends chat message into chat. |  |  |  |

#### Table 50 - Test Case 13

|  |  |
| --- | --- |
| **Test Case ID:** TC14 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Functionality of mobile application | **Test Executed by:** Ryan Harper |
| **Test Name:** Is the mobile application functional. | **Test Executed Date:** TBD |
| **For Test ID:** T8 | **Dependencies:** All |
| **Description:** Functional Mobile application. | **Precondition:** Application is open. |

#### Table 51 - Test Case 14

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | User clicks on signup | N/A | Brought to sign up page. |  |  |  |
| 2 | User Enters a username already chosen | Username: user3 | Prompts user with error that the username is already chosen. |  |  |  |
| 3 | User Enters weak password | Password: Password | Prompts user with error that the password is weak and needs symbols or capitols |  |  |  |
| 4 | User enters valid username and password | Username: user4  Password: Password123! | Credentials are accepted. Bringing them back to the login page |  |  |  |
| 5 | User enters username and password | Username: user4  Password: Password123! | User is logged in. |  |  |  |
| 6 | User goes from home screen to chatbot. | N/A | Opens chatbot |  |  |  |
| 7 | User goes from chatbot to home screen. | N/A | Home screen opens |  |  |  |
| 8 | Opens profile from home screen | N/A | Opens profile |  |  |  |
| 9 | Open settings from profile | N/A | Settings open |  |  |  |
| 10 | Logouts of account | N/A | Brought to home screen. |  |  |  |

#### Table 52 - Test Case 14

|  |  |
| --- | --- |
| **Test Case ID:** TC15 | **Test Designed by:** Ryan Harper |
| **Test Priority:** HIGH | **Test Designed Date:** 25/11/2022 |
| **Module Name:** Functionality of Chatbot | **Test Executed by:** Ryan Harper |
| **Test Name:** Is the chatbot functional | **Test Executed Date:** TBD |
| **For Test ID:** T9 | **Dependencies:** F1 / F3 / F4 / F9 / F6 / F18 / F23 / F24 |
| **Description:** Functional Chatbot communication. | **Precondition:** User is logged in and on chatbot page |

#### Table 53 - Test Case 15

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Test Step | Test Data | Expected results | Actual Results | Status (Pass / Fail) | Notes |
| 1 | Users click on chat bar. | N/A | User can now type. |  |  |  |
| 2 | User sends message | Hello | Message is displayed in text box. |  |  |  |
| 3 | Chatbot responds | N/A | Chatbots message is displayed. |  |  |  |
| 4 | User asks for advice on problem. | Advice on depression. | Message is displayed. |  |  |  |
| 5 | Chatbot responds. | N/A | Chatbot message is displayed |  |  |  |
| 6 | User chooses whether they want more advice from chatbot. | Yes | Response is sent. |  |  |  |
| 7 | Chatbot responds. | N/A | Chatbot continues with advice |  |  |  |
| 8 | User chooses whether they want more advice from chatbot. | No | Response is sent. |  |  |  |
| 9 | Chatbot changes the conversation. | N/A | Message is displayed. |  |  |  |

#### Table 53 - Test Case 15

## Appendix C

Graphical user interface, text, application, email

Description automatically generated

#### Figure 27 – survey part 1

Graphical user interface, text, application, email

Description automatically generated

#### Figure 28 – survey part 2

Graphical user interface, text, application, email

Description automatically generated

#### Figure 29 – survey part 3

## Appendix D – Code Manifest

|  |  |  |
| --- | --- | --- |
| File | Location | Purpose of file |
| Created Code Files | | |
| Test.py | Project\Frontend\test.py | For navigation bars for different screens using kivymd. |
| App.py | Project\Frontend\app.py | App to display simple login and how it interacts with the system. |
| My.kv | Project\Frontend\my.kv | Formatting for the test.py so that it can graphically be displayed and linked up with the backend. |
| Test.kv | Project\Frontend\test.kv | Formatting for the test.py so that it can be graphically be displayed and linked up with the backend. |
| Modified code files | | |
| Nlu.yml | data\nlu.yml | Used to hold natural language understanding so the chatbot can understand what the user is talking about. |
| Rules.yml | data\rules.yml | For predefined rules for the dialouge policy. |
| Stories.yml | data\stories.yml | The representation of a conversation between the user and the chatbot, except the user inputs are expressed as intents rather than words. |
| Config.yml | \Project\V 1.0\config.yml | For making predictions based on the user input. |
| Credentials.yml | \Project\V 1.0\Credentials.yml | Talk to and share bot used for rasa version x. |
| Domain.yml | \Project\V 1.0\Domain.yml | Everything you chatbot currently knows. |
| Endpoints.yml | \Project\V 1.0\Endpoints.yml | For APIs. |
| Forms.yml | \Project\V 1.0\Forms.yml | To collect information from the user. |
| Responses.yml | \Project\V 1.0\Responses.yml | Used to choose from all conditional response variations. |
| Test\_stories.yml | tests\test\_stories.yml | Represents a conversation between the use and the AI, converted into a format were the users inputs are intents and the AI outputs are actions. |