Creating a Web Tool for Researching Conversational User Interfaces

A dissertation presented to the University of Dublin, Trinity College in partial fulfilment of the requirements for the degree of Master of Science in Computer Science.

By

Conor Church

Supervisor: Dr. Gavin Doherty

August 2022

Declaration

I, Conor Church, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

_____

Conor Church

August 19, 2022

Permission to Lend and/or Copy

I, Conor Church, agree that Trinity College Library may lend or copy this thesis upon request.

Conor Church

August 19, 2022

Acknowledgements

A big thank you to my supervisor Gavin Doherty, for giving me the opportunity to study this topic for my thesis. Thank you to Robert Bowman for the support, advice, and guidance throughout the year. Thank you to my peers within the master's course, I benefited greatly from collaborating with you all.

Thanks also to my parents Mary and Michael and my brothers Michael and Aidan, for their continuous and never-ending encouragement and support, I stand on the shoulders of those who came before.

Thank you to my friends: Luke, Joe, Caytlin, Katie, Conn, Donal, Laura, Anna, John, Seán, Aidan, Robert, Robyn, Andrea, and Shane, for all your help and support and being there when I needed to take a break from the studies to unwind.

Thanks to my girlfriend, Amy, for being very supportive and patient enough to put off a lot of our dates until the master's was finished.

Abstract

In this thesis, the objective was to create a tool to help research Conversational User Interfaces (CUIs). To create this, it was essential to research the area of Human-Computer Interaction and Interaction Design. It was important to study the current tools available, what they lacked, why this tool was needed and who would use it. This built the foundation for the requirements that would later mould how to design and implement the tool. The tool was evaluated against the requirements that were defined, whether the requirements were met sufficiently or not and to what extent. The tool was also evaluated against User Interface/User Experience (UI/UX) best practices. The level of success is measured while noting the limitations and possible future workings that can be added to the tool. Overall, the tool that was created has great potential to be used for further study in Human-Computer Interaction research.

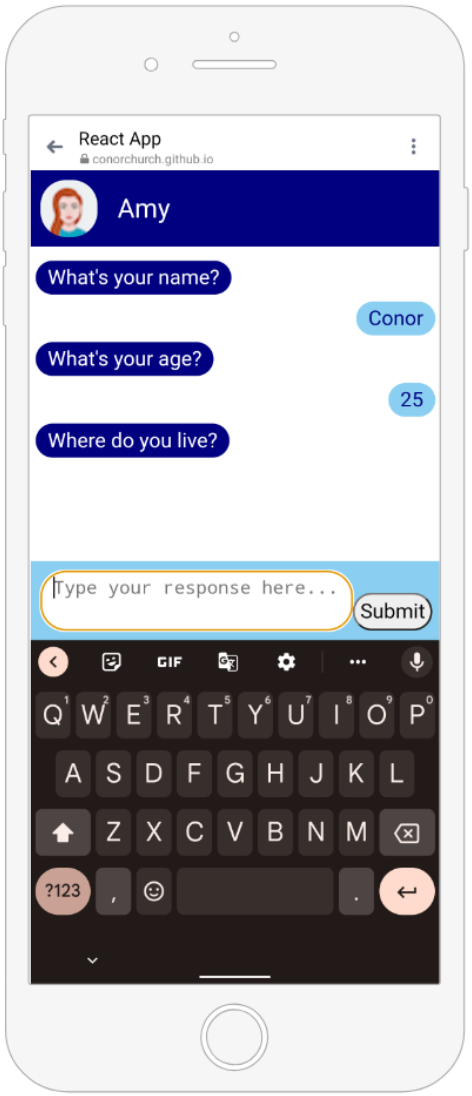

Table of Contents

Table of Figures

Chapter 1: Introduction

1.1. Motivation

The topic of Conversational User Interfaces (CUIs), and CUI tools, has been debated since their prominence in the 2010's. Their limits are yet unknown; however, many see these areas potential to be researched. To design better chatbots, research is needed to understand the effects of chatbot design on user interaction. To conduct this research better, tools are needed to prototype bots (without building full chatbots each time) that can be embedded into research workflow (e.g., Qualtrics surveys) (qualtricsXM, 2021). The need for these tools for research is the motivation behind this thesis.

Some debate that these tools developed with "no code" deployment in mind cannot be useful beyond being "toys" for the end user (Rough & Cowan, 2020). Følstad and Brandtzæg says there are still a lot of things we have not looked at in this area or have researched (Følstad & Brandtzæg, 2017). In (Murad & Munteanu, 2022), they say that current tools for Voice User Interfaces (VUIs) have "little to no validation, and very few have been adopted in industry". This shows that these tools are in their infancy and require more focus and attention. From these papers, very little is known on how to correctly design such tools and this paper seeks to investigate the considerations, potential and limitations such tools bring, while delivering a tool for other researchers to further develop and investigate.

Research in this area may benefit everyday life as there is more to be gained from using CUIs. For example, people are more likely to talk to a CUI regarding their mental health or their issues, as shame, guilt and embarrassment would be a deterrent from talking to a human (Deibel & Evanhoe, 2021). (Lucas, et al., 2014) detail how Virtual Humans (VHs) are used to build rapport and keep anonymity to increase the likelihood of people to express negatively thought emotions (e.g., sadness) and be more truthful when answering personal questions. This research could be very useful in healthcare when patients do not disclose fully what they feel/think for fear of the consequences.

As for designing and creating these CUI's, "Conversations with things" (Deibel & Evanhoe, 2021) explains the importance of language and why it should be at the forefront of designing a CUI. The book goes in depth into how CUI's can be designed poorly, making conversations feel robotic and not human like. It focuses on designing these Conversational interfaces to replicate human interactions and feel like a usual conversation with another human. This book also denotes how certain biases towards certain racial and ethnic groups are very prominent in all aspects of different technologies. This occurs when the design team is not aware or do not think about the impact of language for certain cultural/ethnic groups.

1.2. Significance of *this* tool?

As stated in the previous section, there is a need to study CUIs. However, the question stands as to why there is a need for another one? There have been many similar tools such as "Botsociety" and "Botmock" (Botsociety, 2021), (Botmock, 2021). These tools have similar uses to this tool created for this thesis but have their limitations. They allow users to create chatbots with their helpful User Interface (UI), with draggable options, buttons, and written questions. One limitation found in these tools is that they are not easily embeddable in third party surveys such as in Qualtrics (qualtricsXM, 2021), where somebody researching CUIs may want this tool as part of a parent survey.

Another limitation is the need for an account with some existing tools and the possibility of paying for them. The code will be available as an open-source project hosted on "GitHub" (GitHub, Inc., 2022) where anyone can download the code and run it without having to pay. A benefit to having the code openly available to everyone is that this tool can be configured to perform additional tasks if the research would like to modify it further. If they have experience with web technologies and coding, then they can add to it.

1.3. Research Objective

In this report, I shall first outline the background research in relation to CUIs, then I will outline the requirements of such a CUI tool. I will discuss the UI design of this tool along with the considerations, inspirations, choices that I chose to take, and difficulties with the design and implementation phases. Next, I will talk about the implementation of the tool, with technologies used, structure of the tool, architecture and the difficulties faced in the implementation. Then I will evaluate the tool, based on industry heuristics used to evaluate UI design. Finally, I will discuss the implications, limitations of the tool to date and future works that could be taken on going forward.

The main aim for this project is creating a tool for researching CUIs. This includes the considerations on how others will use the tool and process the language used later, but not directly used in the tool itself. For instance, the tool will not process the language used, but hopefully will help in another work later if it wishes to study language in response to certain questions or prompts.

Chapter 2: Background

2.1. CUI Research

The main reason for extensive research into CUI's is that nobody is fully sure what makes a good CUI and how they can be improved. There are lots of aspects that may help, like making them appear and converse more like a human, but how can CUIs become more like a human? This could include having a personality, have proper social cues (taking turns, not interrupting etc.) and building a rapport to make the user feel more comfortable. (Gratch, et al., 2013), describes how their "Rapport Agent" seeks to mimic human nonverbal cues, to help build a rapport with the human converser. Although there is a bonus for making these CUIs more human like, there are many advantages to having them keep a user's anonymity. Therefore, the likes of (Lucas, et al., 2014) look at the beneficial side effects of users knowing another human will not be able to identify them from the interaction.

One major area within this domain is the focus on natural language. This is to help the conversation flow and so interactions with CUIs are not merely one or two answers. (Harms, et al., 2018) discuss the concept of "Dialogue Management" (DM) and how commercial and research tools are constructing effective DM systems for CUIs to have more fluid conversations with their users. Chatbots, for instance, appear clunky and robotic, they do not converse as a human would, even on a messaging application. Evidence of this became quite apparent when Facebook rolled back it's chatbot Artificial Intelligence (AI) ambitions when its Facebook Messenger Instant Messaging (IM) service AI bot reported a seventy percent failure rate of human requests (Orlowski, 2017).

Both (Deibel & Evanhoe, 2021) and (Harms, et al., 2018), mention that it is important to understand human social behaviour and natural language understanding (NLU) to improve CUIs. We find it very difficult to accurately define human social interaction etiquette, so it becomes very difficult for other CUIs to mimic it. Such issues become problematic when users are marginalised due to biased phrasing used by CUIs (Deibel & Evanhoe, 2021). How a CUI may talk and engage with users is yet to be fully addressed.

2.2. What do Existing CUIs look like?

Since the early twentieth century, science fiction has depicted what sentient/autonomous CUIs may look like. One famous example is HAL 9000 from “2001: A Space Odyssey” (2001: A Space Odyssey, 1968).

Figure 1 - HAL 9000 from "2001: A Space Odyssey" Image source: (Wikipedia, 2022)

Interestingly, some of our current CUIs in the form of voice assistants, do not look too far off from HAL. The Alexa device is a physical speaker with a ring of light on the top or bottom of it (Amazon, 2022). The light, which is primarily blue around its speaker, changes colour/shape/movement to let us know when it is speaking, like HAL.

Other CUIs are much more digitalised, appearing on our screens. Voice assistants, such as, "Google Assistant" (Google, 2022), appears on our phone screens and asks what we need help with. Chatbots normally take the form of a messaging application where the user types their message to it just like another human. On commercial websites, a chat window appears and a chatbot attempts to aid navigation/intent. Take the clothing store "Zara", their chatbot appears and can offer a series of buttons to help locate what the user needs, if the free text answer does not work (Zara, 2022), as seen in Figure 2 below.

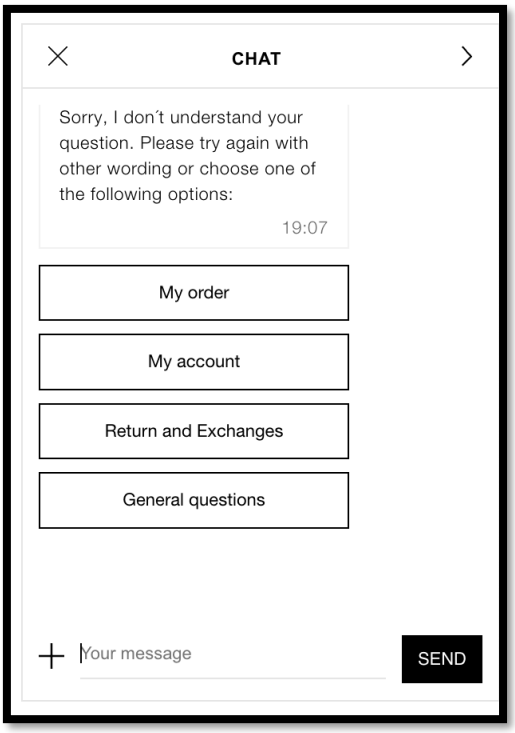

Figure 2 - Zara chatbot on their website. Image site: (Zara, 2022)

Other bots take the form of social media users on their platforms. This is seen on both Facebook Messenger and Twitter (Facebook, 2022) (Twitter, 2022). Companies will create chatbots from certain tools such as "Botsociety" and "Botmock" (Botsociety, 2021), (Botmock, 2021) and integrate them with the likes of Facebook Messenger, like in Figure 3. This is a chatbot within the Messenger app where your predefined responses can tell the chatbot what to say next. This chatbot was created to help with a request or help answer Frequently Asked Questions (FAQs) for the "Las Radas Wine & Tapas Bar" in Naas, Co.Kildare, Ireland (Las Radas, 2017).

Figure 3 - Las Radas Wine & Tapas Bar Chatbot integrated into Messenger. Image source: (Las Radas, 2017)

Twitter chatbots have been notorious for their bombardment of tweets from a number or accounts throughout the platform. They tend not to perform complex tasks, but they "follow" users to gain a large audience and then carry out tasks such as broadcasting, responding, and forwarding (Grudin & Jacques, 2019).

Other types of CUIs can take the form of individuals working in different areas, such as the healthcare sector. In (Trigo, et al., 2021), the participants are chatting to a digitalised doctor on a screen, rather than a messaging chatbot. Similarly, there was a Virtual Human (VH) on screen talking to participants in (Lucas, et al., 2014). Both studies used a virtual person to help gain rapport with the users interacting with them. Which is to say CUIs take the form that most suits their requirements and their creator's ability to create them. There is a vast array of CUIs that come in all shapes and sizes. In (Grudin & Jacques, 2019), they outline the different types of CUIs, in relation to ability rather than appearance. These are split into three categories: Virtual Companions, Intelligent Assistants and Task-Focused Chatbots. Virtual Companions can chat about wide range of topics, and keep a conversation going e.g., Cleverbot. Intelligent Assistants

chat about a broad range of topics but keeps the conversation short e.g., Alexa. Task-Focused Chatbots have a very narrow range of communication topics and keep the conversation relatively short e.g., Non-Playable Characters (NPCs) in computer games.

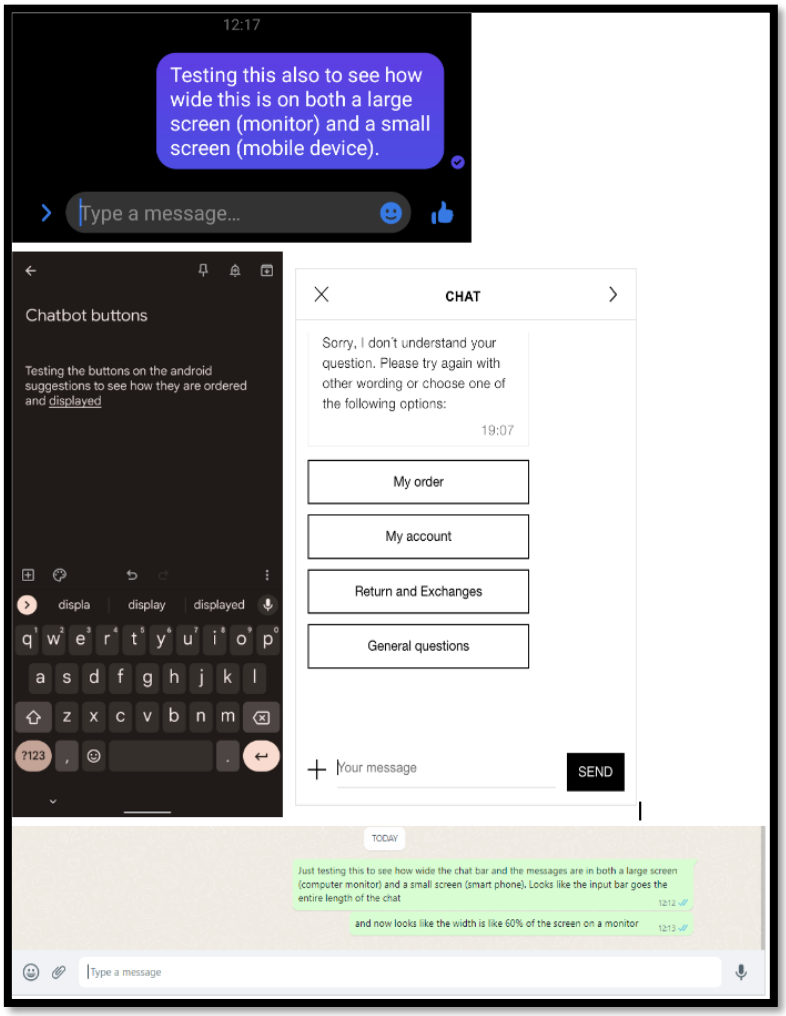

Figure 4 - Different UI example. Image sources: (Google, 2022), (Facebook, 2022), (Zara, 2022), (Facebook, 2022)

A consideration for this project in terms of UI design was making it seem familiar to the user. The user should know how to interact with the CUI with little context or help. The main inspiration for the design were well known messaging applications such as Facebook Messenger and WhatsApp (Facebook, 2022), (Facebook, 2022), as seen above in Figure 4. A lot of messaging apps today will follow a similar design format and so for familiarity this tool would follow suit. Certain aspects such as the "submit" button on the bottom right of the screen and the text input taking up the majority width of the window are common features of these apps. These features have become the industry standard and the look of a lot of messaging applications on social media. Many CUIs tend to mimic social media messaging applications to increase usability for the user.

2.3. Tools for Prototyping and Research

So far (Rough & Cowan, 2020) outline how existing commercial tools require coding experience to have any additional function other than rehearsed dialogue. They require Application Programming Interfaces (APIs) to act on the user's responses, if any action is to be taken from a user's responses. These CUIs are typically the Task-focused chatbots mentioned in the previous section, where the conversation is quite scripted.

An example of a tool to make such chatbots is "Botsociety" as mentioned previously. "Botsociety" helps create chatbots that can be used for applications such as Facebook Messenger; however, this is to be used within an application and cannot be embedded into other web forms such as third party surveys. These chatbots normally have simple dialogues with buttons to help users with their requests as shown in Figure 3. This is one of the most significant reasons for the need for a tool such as the one in this thesis: the lack of usability and configurability.

There has been more research into more elaborate tools for commercial and research use. (Harms, et al., 2018) outlines three different types of tools that are currently being used in research and commercial areas.

The first type is the "Handcrafted Approach", where the conversation has set states and handwritten rules for the conversation to follow. This offers more control to the creator of this CUI to define the direction of the conversation by the rules policy that has been written. The tool moves the "state" of the conversation to the next topic once different rules in the set policy have been satisfied. An example of this type of tool is "Flow XO", used for customer support in business-customer interactions (Harms, et al., 2018).

The next type of tools is from a "Probabilistic Approach", this requires more computing power and allows for a more fluid conversation. There are no set rules that the tool follows while conversing with a user. Instead, it learns rules from actual conversations (Harms, et al., 2018). It learns from training data and looks for key words that have appeared from a user's response to answer appropriately. An example of this open-ended conversation CUI is a "Chatterbot" (Cox, 2021).

The last type of tool is a "Hybrid Approach" of both where it uses the advantages of Handcrafted and Probabilistic approaches. One such example of a Hybrid approach is "Open Dial" (Lison, 2015), which uses if/else type rules to cut the possible "state" space to search using a partially observable Markov decision processes (POMDP) search algorithm.

Comparatively, (Harms, et al., 2018) notes that the first of these, the Handcrafted approach of CUI tools have the clunky dialogue structure since the conversation is following a certain path. The other two approaches allow for a much more fluid like and open-ended conversation with not as much a narrow path; however, these last two approaches grant less control of the conversation and are more difficult for users to use without prior software engineering experience.

Chapter 3: Requirements

3.1. Personas/Scenarios

Possible users of the CUI tool includes a “Creator” - they will be modifying the configuration file by updating the parameters, inputting the questions and question type. They will be hosting the tool and will be reviewing the conversation afterwards.

The second type of user is the “Participant” - they will be conversing with the rendered CUI and will be viewing the UI.

Below in Figure 5 and Figure 6 we have two personas to help form a basis for our requirements. Understanding the people involved in using the system: how they will use it and what is their expectations/needs of the system, can help form the requirements for the tool.

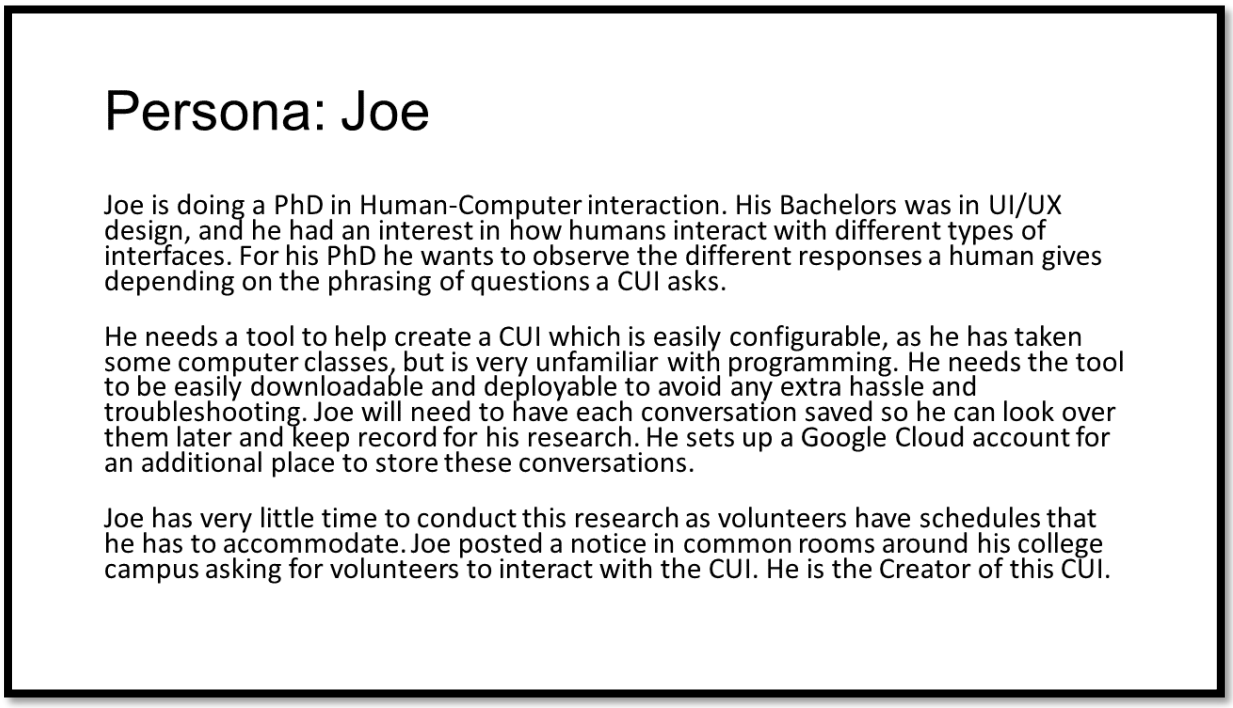

Figure 5 - Persona of Joe, the Creator of the CUI

Persona: Caytlin

Caytlin is doing a PhD in Analytical Chemistry. She has an Alexa device at home and has a personal interest in how machines are becoming more prevalent in our day-to-day lives. She saw Joe's notice looking for volunteers on the college campus and was happy to help. She is Joe's third participant.

Caytlin contacts Joe and they agree a time to interact with the CUI. Caytlin has a tight schedule, she has a PhD of her own and provides chemistry grinds on the side, so there are only a few time slots she is free.

Caytlin is the Participant in this research.

Figure 6 - Persona of Caytlin, the Participant interacting with the CUI

Scenario

Joe creates the CUI by downloading the source code provided from the repository and follows the instructions to configure the tool to his environment and add his questions. He chooses to have a mix of free text and multiple-choice questions.

He then hosts the CUI on his web server and embeds the link into a Qualtrics Survey.

Joe provides Caytlin with some background information on what to expect. He sends her a URL to the Qualtrics survey and asks her to answer the questions to the best of her ability.

Caytlin partakes in the survey while using her phone on her lunch break.

After completion of the survey, the conversation is saved and encrypted locally and sent to the cloud. Joe retrieves the data from the cloud and analyses the interaction.

Figure 7 - Scenario between Joe and Caytlin

From the scenario in Figure 7, there are two users of the system: the Creator (Joe) and the Participant (Caytlin). These personas and scenario help understand what is needed of the tool. The scenario is an informal way of describing the workflow of the system that is easy to understand without going into a lot of technical detail. This scenario also defines Joe's goals and hopes for the tool which are important for understanding the requirements (Preece, et al., 2015).

The requirements that are listed later in this chapter have been derived from this scenario and will be evaluated later in the thesis.

3.2. Where will it be Used?

This tool can be used in a variety of environments. As mentioned before, it has been embedded in certain third party surveys such as Qualtrics (qualtricsXM, 2021), if the creator of the CUI wishes to use it that way. Joe used this same third party survey to host his tool; however, there are many other options out there, such as a JavaScript library, "jsPsych", which allows people with JavaScript knowledge to create their own surveys. It would be possible for the Creator of the CUI to use this tool alongside that open-source software (Leeuw, 2015). For third party surveys to use this tool, there needs to be an Application Programming Interface (API) link to point to the hosted server address within a HTML "iframe" tag to render it into the survey webpage.

On webservers

The Creator of this CUI can host the tool locally on their machine or on a server where others can access it from outside the local network. This can be useful in a COVID filled world or if their research involves the answers of people that are geographically dispersed. With hosting alone, they can have participants interact with the tool without a third party survey.

On phones and monitors

The tool has been set up to have options to be viewed on a small screen and a large screen. This is a setting the Creator of the CUI can choose before deploying the tool. It can be viewed by the participant on different devices, such as a phone or on a computer monitor.

3.3. Establishing Requirements

When developing software, it is very important to establish the requirements the software is trying to achieve. Before we can establish requirements, we first must know who the users of the system will be, what they will use it for and the expected output/outcome of the system.

Understanding the type of users that will be using the system, their needs, and their abilities is very important. An example of this is outlined in the previous section with Joe and Caytlin's personas and the scenario they act out. This gives an insight into the different interactions that users will have with each other and the system itself. Crucially, requirements can be extremely important in detecting issues with the system or way of working early on before the system has been developed (Boehm & Basil, 2001).

Establishing requirements are an iterative process, and continuous refinement is expected. To obtain requirements, data is retrieved, analysed, and interpreted (Preece, et al., 2015). For this tool, data gathering required looking at why there was a need for the tool or the goal for the tool, what other tools were available and what could be improved or added to them.

3.4. What are the Requirements?

There are five main requirements of this tool:

Requirement 1: Accessible to both Researcher and User.

One of the main requirements is the accessibility and usability of the tool for both the person creating the tool and participant interacting with the chatbot. The tool needs to be created easily and should be accessible to anyone that wants to create a chatbot. Therefore, the CUI tool should be free and open-source - The code and instructions are stored on a free, public repository that anyone can access. The Participant should be able to access the CUI from their device.

Requirement 2: CUI to be embeddable and render in third party surveys e.g Qualtrics.

One improvement this tool hopes to make on similar tools is that it can be embeddable into third party surveys so the Creator can use it in conjunction with other forms of media. For instance, the CUI may be part of a larger survey and having this feature allows it to be used in different scenarios.

Requirement 3: Tool must save the conversation for the Creator.

The main purpose of this tool is to use it for research by the Creator. There are many ways they can use it in their research; however, they will always need to analyse the conversation. This requires them to review the conversation, which will require the conversation to be stored. The tool should be able to store the conversation so that the Creator can look over and examine the interaction between the chatbot and Participant when they are able to.

Requirement 4: Tool must be configurable to the Creator.

As mentioned previously, the researcher could be conducting research for several reasons. This would mean the conversations will be different for each Creator given the different requirements the Creator may have. From this assumption, the tool needs to be configurable to accommodate different conversations the Creator would be hoping to create. This includes changing the questions and even look of the chatbot, to some extent.

Requirement 5: CUI must be familiar to the Participant, as if messaging a friend, so they do not need to learn how to interact with it, with little to no instructions.

The primary area of research this tool will be used for, is Human-Computer interactions. Therefore, to allow the tool to be familiar and approachable, the tool should replicate a messaging application, where the participant messages a friend as usual (if they use messaging applications). There should be a human presence, such that there is a name and picture displayed to the Participant as if talking to a real person. The tool should be intuitive too, so not to solely rely on a user's experience with these applications, by giving instructions to the Participant where relevant e.g., labelled buttons/elements.

Chapter 4: Design

4.1. UI Design

The design of this tool was influenced by the design of messaging applications and similar tools as outlined in the "Background" chapter. Preliminary sketches were drawn such as in Figure 8. This sketch is to denote how buttons will be displayed to the user. For instance, should the free text option (where users can type their response freely) be displayed at the same time as the button options? It was decided to leave the free text option to leave more room for button answers. There was no need for the text input to still be visible since the Participant would not be using it. If the Creator needed the button responses to be more than one word, a design for the buttons and how they fit into the conversation needed to be thought of. This is where the mock-up shows that the buttons will go width ways across the screen and if one of the buttons is too long it will be placed under the previous button(s). There is a limit to the number of buttons and the number of characters a button can have so the buttons do not fill up the entire screen.

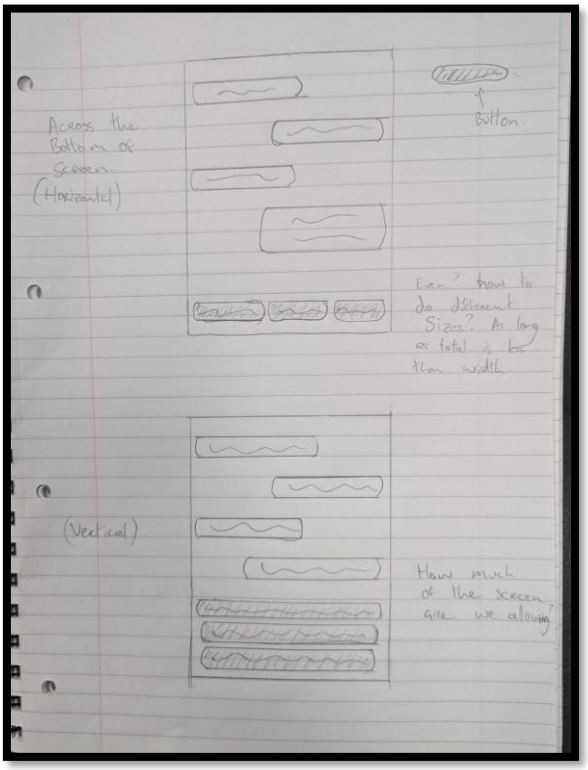

Figure 8 - Sketch if buttons within CUI tool

A rough rendering of the tool in Qualtrics can be seen in Figure 9. This simple UI is the first rendering of the tool within the survey, with a simple input text box and submit button and no header/footer. The chatbot messages were given a navy colour and the Participant messages were given a light blue colour to differentiate the two. The colour scheme was inspired by messaging applications and the colours of a sports team. Blues are normally a soft and appealing and the likes of Facebook Messenger use them (Facebook, 2022). The tool was rendered in Qualtrics to check if it would embed and adhere to "Requirement 2" in the "Requirements" chapter above.

Figure 9 - Original Rendering of CUI in Qualtrics survey

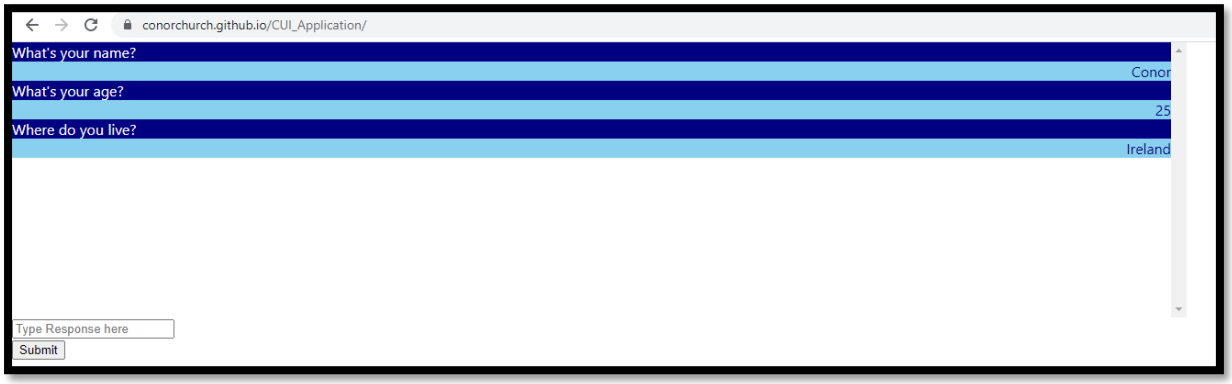

Figure 10 - Original Rendering of CUI tool on browser

In Figure 10, there is an example of the different colour schemes and the tool rendering in a browser. This showed a need to design the tool differently for different screen types. In Figure 11, there is a similar early design only with the button options are displayed instead of the free text input. In previous versions of the CUI tool in a web browser, the conversation takes up a great portion of the screen width which makes it seem very congested. In the WhatsApp web application screen at the bottom of Figure 4, the chat window does not expand the entire width of the screen, only a small segment, to the left of the chat window is a list of the chats available to message in.

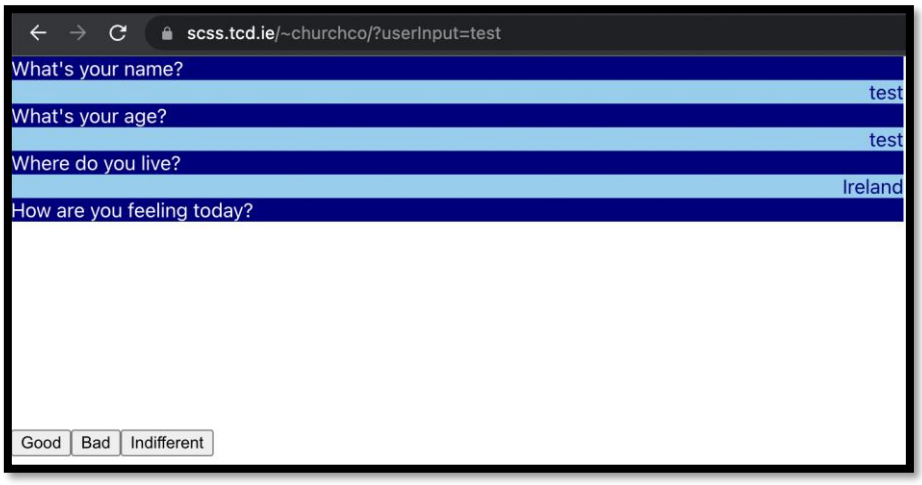

Figure 11 - Original UI of chatbot

This design was updated in Figure 12, where the monitor type screen only takes up a portion of the screen and is centred. The input and submit buttons are changed to take up the width of the conversation pane. However, there is no border to frame the conversation space like in other messaging apps. This was noted as an improvement for the next iteration of the monitor screen type design, which is implemented in Figure 15. A major difference is how the conversation is displayed to the user. Now there is white space horizontally and vertically between the chatbot's and Participant's messages. The space between messages helps the readability of the conversation along with the different colours. The rounder edges are also more subtle and appealing. This follows common industry practice as seen in Figure 4, as most chat messages normally expand horizontally to about sixty to eighty percent of the chat width with space between them and have smoother edges.

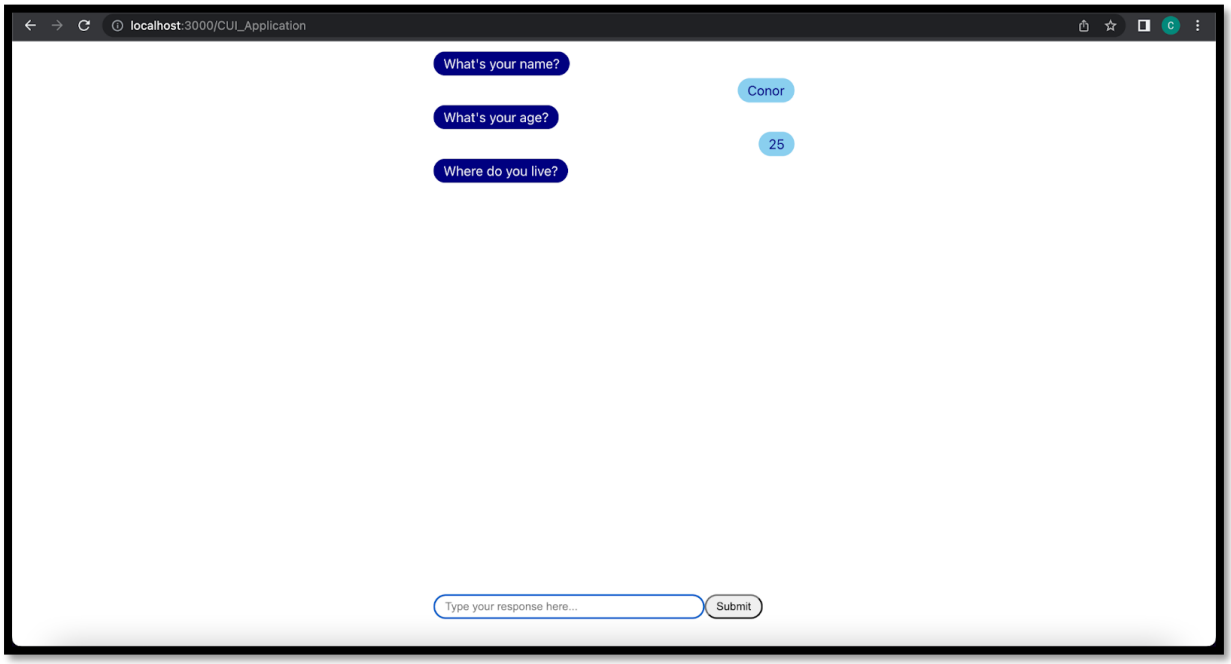

Figure 12 - Early Monitor Screen rendering in browser

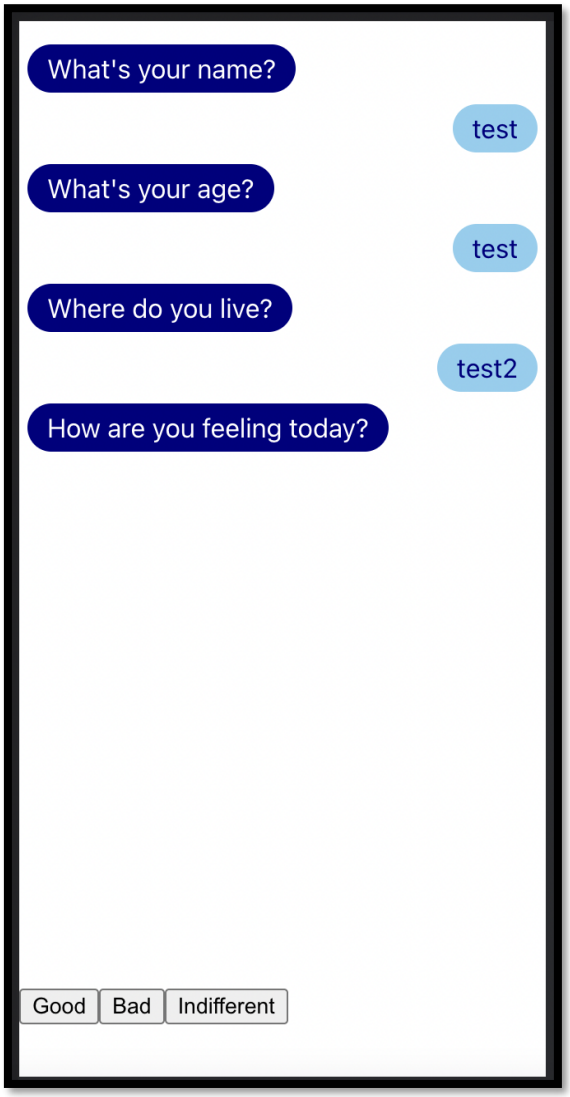

Figure 13 - Original button look in CUI tool

The round edges of each message are shown in a smaller portrait/small screen view in Figure 13. Having the colours different to the background means they pop out more at the Participant, while not taking up the entire page also, this makes it seem simple and spacious. This small screen example showcases the button options on the smaller screen too. Although progress is being made to make the tool more like the messaging applications mentioned in the "Background" chapter, the screen looks quite barren. The addition of a header and footer was needed to frame the conversation, along with updating the format and appearance of the input/buttons.

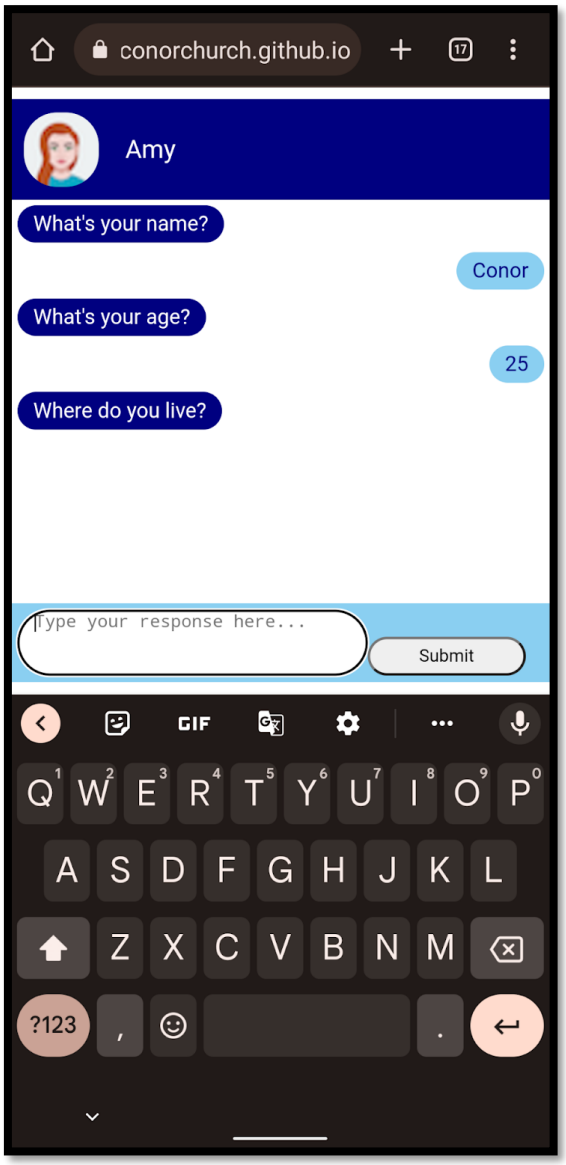

Figure 14 - Introduction of Header and Footer in tool

In Figure 14, a navy header with an “Avatar” picture and name along with a light blue footer background have been added to the rendered tool. The main motivation for header was to contain the name of the person you are talking to and their photo. This helps give a personality to the tool, as if messaging another person. It also adds a buffer between the start of the page and the conversation itself. It is in line with how other messaging applications design their applications. The footer wraps the text input/buttons well and separates it from the conversation and along with the header, frames the conversation, making it the clear focus of the Participant.

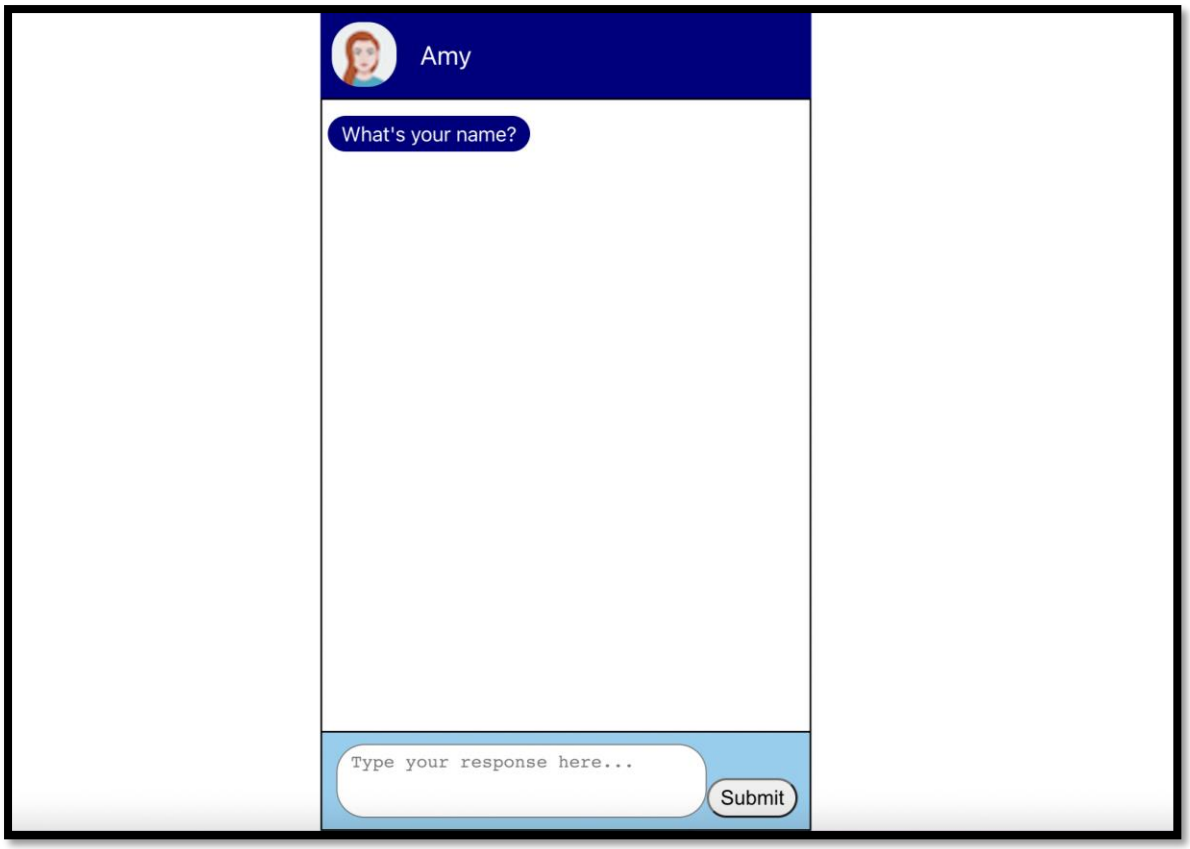

Figure 15 - Finished Monitor view for CUI

By Figure 16, the text within the text input is centred and expanding the header and footer to reach the top and bottom of the screen respectfully. Now the focus shifts to the dynamic design of the tool. For example, there is no limit to the number of characters a Participant can input and like other applications, the input box will extend vertically up to a certain point. This is to help the Participant see their response as they type. But after a time, the box will stop extending and it will start to scroll. This is a consideration for the conversation, so the input box does not extend the entire way up the screen so the Participant cannot see the conversation before, or the current question they are responding to.

The input component or footer follows designs of messaging applications where the footer width will fill the width of the screen (if the type of screen is a phone – specified in the configuration file, seen in Figure 16) or the predefined space (on the monitor screen type, seen in Figure 15).

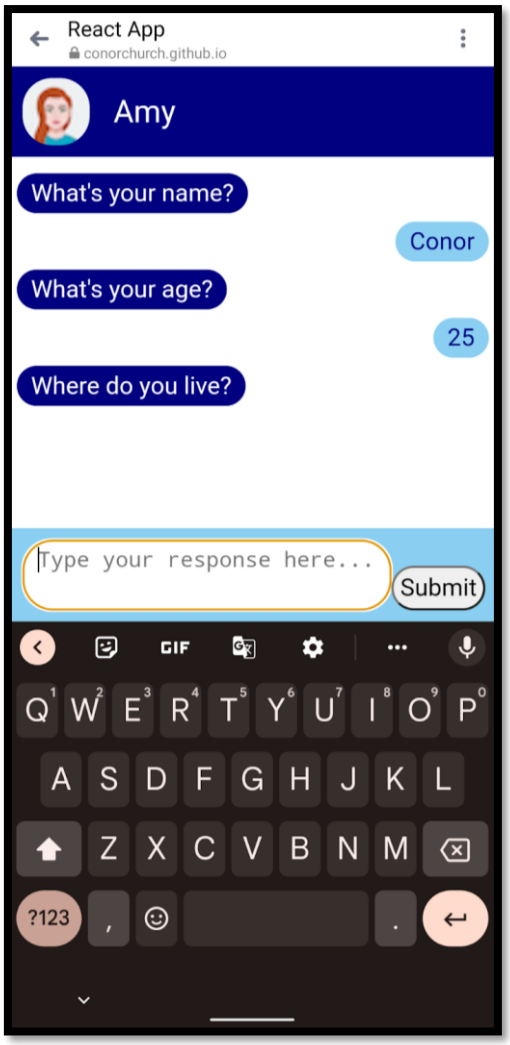

Figure 16 - Finished phone rendering in browser

The Submit button is slightly different in that it will always be beside the text component and will disappear when the text input disappears. The button is roughly thirty percent of the width of the conversation window as the text input box needs most of the space. This is a simple button for a simple function. Other applications have icons for their submit button, although in this scenario it was better to be explicit writing the word “Submit” on the button to make it easier on the Participant, so they understand it’s function rather than learning what it does.

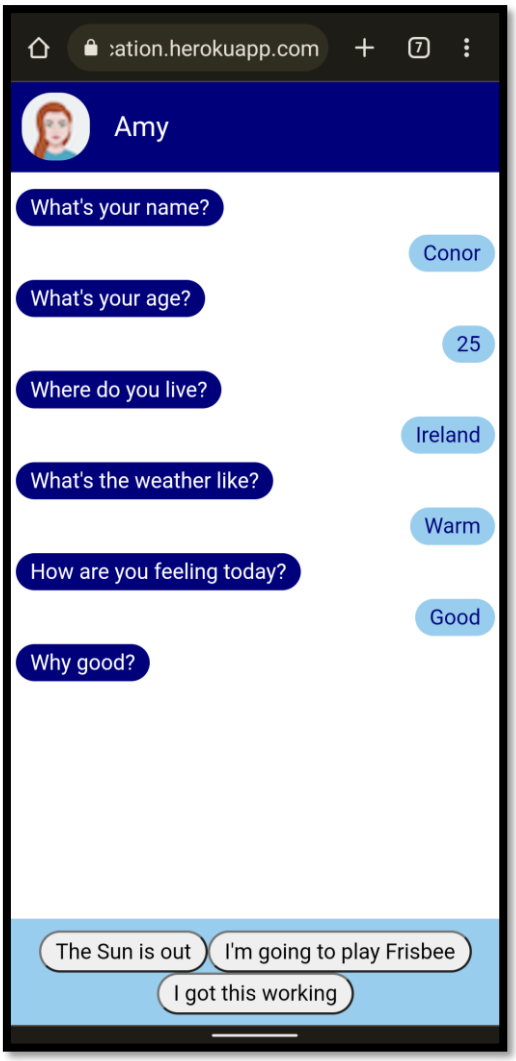

Figure 17 - Finished UI for Button options

The second type of input component is the button, this component fills up the entire footer area like the text and submit button. There are up to four options/answers allowed. They are each labelled with the text that will be added to the conversation when clicked/pressed. This isn't normally in messaging apps so inspiration for this was found on customer support chats such as Zara (Zara, 2022) mentioned in the "Background" chapter, where buttons are displayed to help direct users to the correct page. Figure 17 takes the inspiration for buttons from such applications to display them in the footer as shown. This builds on the UI shown in Figure 13, where the buttons are not part of a footer, smaller and not aligned to the centre of the screen.

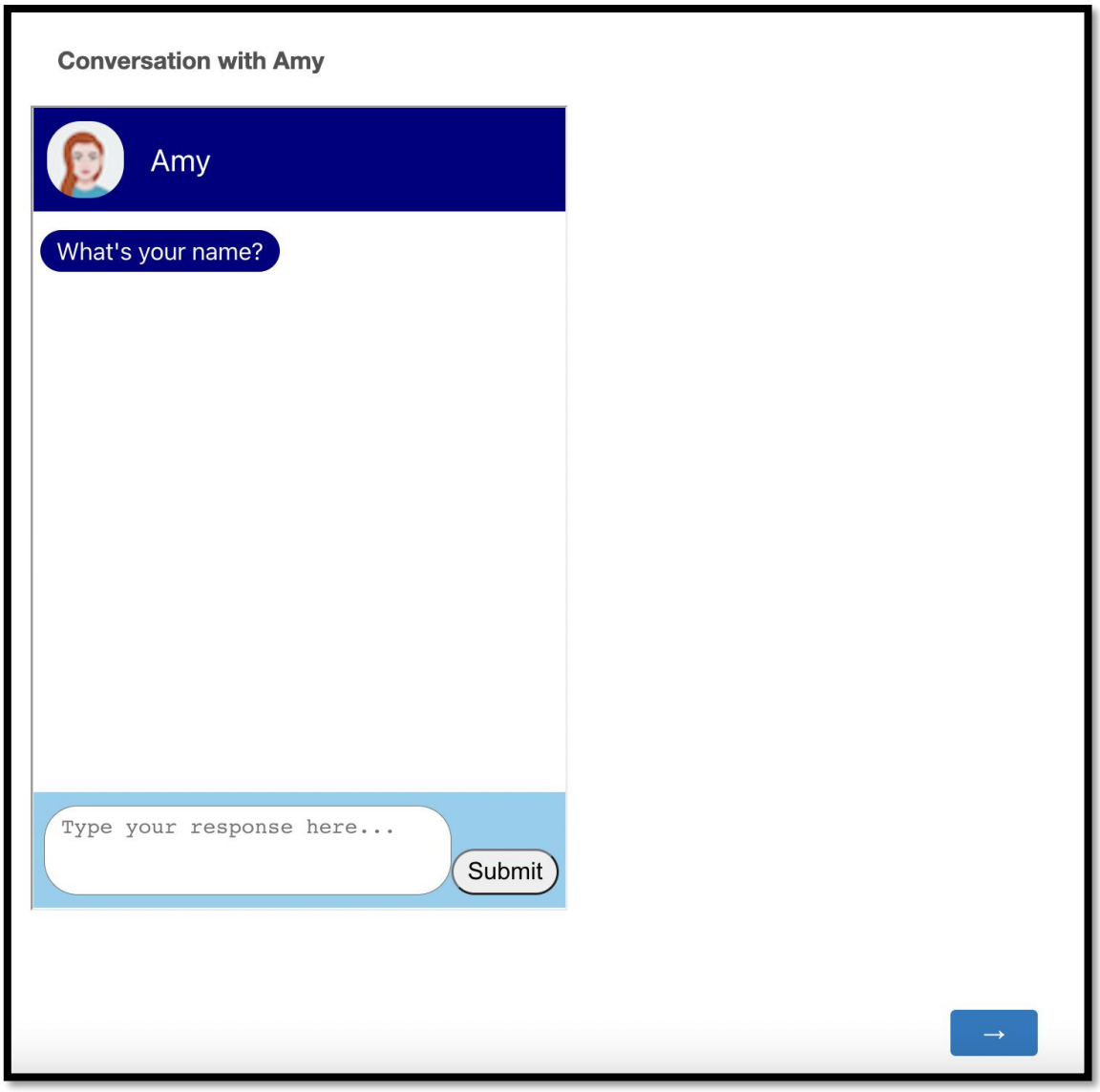

Figure 18 - Finished rendering in Qualtrics survey

In Figure 18, the final UI design of the CUI tool is shown within the Qualtrics survey, contrasting the first one that was shown in Figure 9, showing the progression of the design. Now the design is much more user friendly, with bigger input components, easier to follow message design and avatar replicating a friend's messaging profile.

The end page, shown in Figure 19, is a simple page that tells the Participant in some way that the conversation has ended. The Creator is free to put whatever text they wish to appear to the Participant. This accommodates different use cases such as if the tool is embedded into a survey as previously mentioned, the end message can prompt the participant to continue with the survey. When the tool is not embedded anywhere the end message can prompt the participant to alert the Creator that they are finished with the CUI.

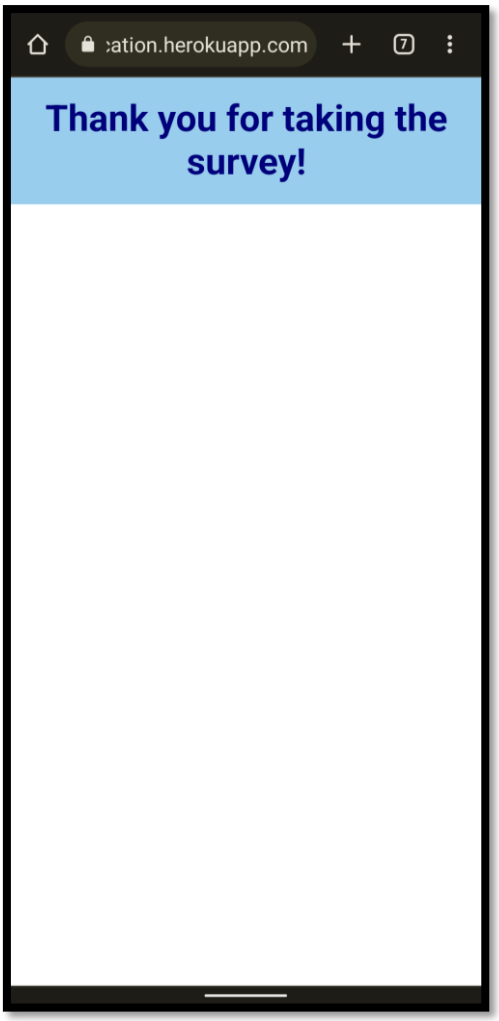

Figure 19 - Example end message for user

4.2. Human Conversation Design

4.2.1. Avatar

The “Avatar” in this tool, is meant to make the tool more approachable, as if talking to a human, and make it more familiar, like messaging someone in a popular social media application. Some situations require a more human like interaction and so the idea of adding a human-like actor into the tool would be useful.

Allowing this aspect of the tool to be configurable allows for the tool to appeal more to whatever audience the Creator shares it with. The “Avatar” can have any persona that the Creator wants it to have, especially if their research deals with people from different cultural and ethnic backgrounds.

4.2.2. Turn Taking

It was very important when designing this CUI to make it as approachable and human-like as possible, which to date, is very difficult to do. This paper looked at a variety of ways on how CUI's can become more human like and what leaps they need to bridge the gap in communication.

For instance, in normal human-to-human interaction, a normal conversation has a speaker and a listener. Without being rude, one normally talks at a time, but how does one know to listen and when to speak? Humans have an array of cues to help know when to chime in. There are both physical and audible cues. To name a few there are, hand gestures, facial expressions (physical cues) and tone of voice (audible cues) (Deibel & Evanhoe, 2021). This is very difficult to transfer to a digital space. Even when humans talk over messaging applications, they normally use punctuation and pauses to let their conversational partner(s) respond.

One design decision that was taken to help the tool appear slightly less robotic was having a pause after the Participant had finished their turn speaking/messaging, before the chatbot responded. At first without a pause, it was intimidating to the Participant communicating to the chatbot, and the conversation was more robotic. Now with a pause, the conversation flows more like a human-to-human interaction where a speaker (Participant in this instance) communicates what their thoughts are and then the listener (chatbot) needs to hear/read what is being communicated and understand and form a response themselves.

4.3. CUI Tool Design

The design of the tool itself was a major aspect. There were numerous considerations on how to design the tool to aid both the Creator and the Participant the most. An original workflow would have been that the tool was centrally hosted, and the Creator can customize it from where it was hosted, but this would have limited the Creator on how they wanted to use the chatbot.

So, a better workflow was having the tool's code be stored in a repository and the Creator would download the code and configure it before deploying it to their own webspace. This involved some thought into the configuration file and how this could be designed to make configuration easy.

The configuration file is a JavaScript Object Notation (JSON) file, that has the questions that the Creator can change and add to. Once changed they can deploy to their webspace. The JSON file is not necessarily very appealing or extremely straightforward if someone has not encountered it before and a UI would have been much better, but to get the tool operational, a

JSON file would be a good start. This is a central point of configuration for the tool, so there is no need for the Creator to change the code unless they would like to configure it more specifically.

There are also various parameters that the Creator may need to change in the configuration file, that relates more specifically to them and their research such as the “Avatar” customisation and the server address. The configuration file is loaded into the main code files so the code can use whatever is in the configuration file to render the tool.

A consideration in designing the tool was to make it as easy for the Creator as possible. This meant once the configuration file was complete, making it easily deployable was a main consideration. If the Creator is not well versed in the command line or coding, they may struggle with deployment, so the goal was to deploy with as few commands and changes as possible. There are only two commands at present the Creator needs to run to deploy the tool: one to build the frontend/UI and the other to run the server in their webspace.

Encrypting the data and storing it are all taken care of by the tool, so the Creator does not need to run those commands of creating a key and encrypting the files. The only other command they will need to run is the one responsible for decrypting the conversations on their server when they wish to view them.

4.4. Difficulties Faced

There were many considerations to be made when designing the tool. And with this came various difficulties. User Interface/User Experience (UI/UX) can be difficult when trying to design for a broad range of functions. This tool is designed to cater for different use cases and so it was important to keep in mind what the Creator might want to do and how to accommodate that.

A difficulty encountered was how to accommodate different screens. To help the tool’s usability, it must be able to function on different screen sizes. This was a consideration for the Participant where they may not be in a place that has a computer monitor and so they can complete the conversation away from the Creator or in the space the Creator provided. The different screens require different designs – on phone screens the conversation will take up the entire width of the screen to make it easily visible, but on computer monitors this would not look too appealing, and it was seen on web messaging applications that the conversation was only seen on part of the screen. This meant an option in the configuration file was given to allow the Creator to specify what screen the Participant will be using. Now it was easier to specify the screen type in the configuration file but moving forward the Participant could be

asked the screen they are using or better still, the tool could recognize what size screen is being used to display the tool appropriately.

When designing this type of tool, there are compromises to be made between configurability and design. For instance, there is only so many button options to be allowed on the screen at once and the number of characters, so it fits on the screen. If there were an infinite number of buttons allowed, then they would fill the entire screen with no room for the conversation. Likewise for the characters in each button, they would extend too much into the conversation. These types of compromises needed to be made for both the Creator and Participant.

Chapter 5: Implementation

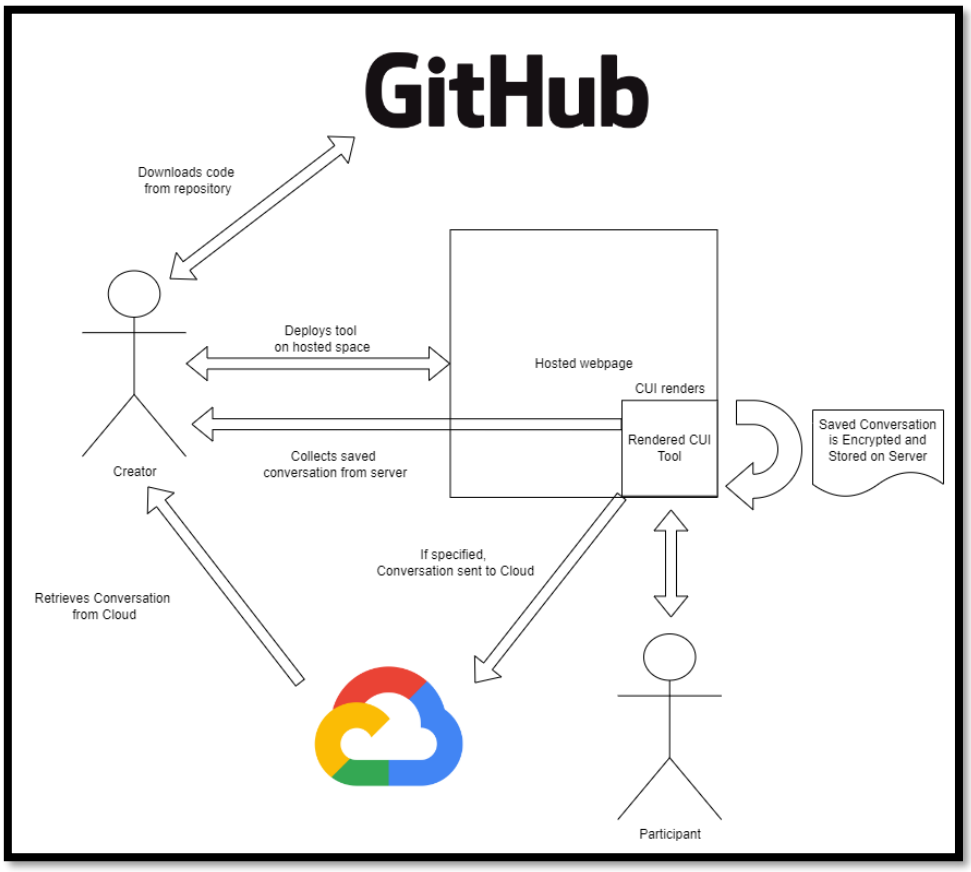

Figure 20 - Workflow of CUI tool, Image sources: (Google, 2022), (GitHub, Inc., 2022)

5.1. Technologies Used

5.1.1. Frontend – JavaScript/React

React is a JavaScript frontend framework developed by Facebook to help create dynamic UIs. React embraces the idea that rendering logic is coupled with other UI logic, i.e., how the data is prepared for display (Meta Platforms, Inc., 2022).

React has “elements” that describe how the UI should look. Markups and logic are not separate in React and so elements can contain both. React works by comparing React DOM (Document Object Model) with the regular HTML DOM and updates it to match the React Elements.

React was chosen for this project because of its popularity with the online developer community. It is widely used and so a lot of bugs and issues will be investigated quicker, and the code base will be constantly maintained. There is up-to-date documentation available which helped with troubleshooting and the creation of this tool.

There was no major advantage or disadvantage to using other frameworks so React was chosen.

5.1.2. Backend – Python/Flask

Python is a very straightforward language to use when designing a simple backend and prior experience led to it being chosen in this project. Flask is a Python web framework for making Application Programming Interfaces (APIs), which helped to store the conversation after it was finished.

Flask is used over other frameworks for more creative control when constructing applications (Grinberg, 2014). Flask allows fluidity when working with the different components that may be added to the application. It supports the choices a developer wishes to make when building alongside the framework. The basic framework has a solid core of functionality with possible extensions that can provide more. This means a developer can pick and choose packages leaving their application to be as lightweight as they want.

5.1.3. Cloud Provider – Google Cloud

The decision was made to accommodate one cloud provider as it would be out of scope to accommodate all the possible cloud providers. Google Cloud was the easiest to set up and to use. There is an abundance of documentation that helped in configuring it to the tool and the costs were seemingly low (Google, 2022). Other options could have been Microsoft Azure (Microsoft, 2022) or Amazon Web Services (AWS) (Amazon Web Services, Inc. , 2022); however, Google Cloud seemed the most straightforward when integrating it into the tool.

5.1.4. Code Repository/README – GitHub

GitHub is an online code repository that can be used to store, edit, and retrieve code. This tool is intended to be an open-source tool and this repository will be used to store it. It is free to make an account and use the repository (GitHub, Inc., 2022).

GitHub takes advantage of "Git" (Software Freedom Conservancy, 2022) which is a versioning control software, that allows someone to save previous versions of their code and add or remove whatever they want while still having access to previous versions of the codebase. On GitHub, comparisons can be made between versions which is very useful when troubleshooting and collaborating. There are many other online repositories like GitHub such as "GitLab" (GitLab B.V, 2022) and "Gerrit" (Gerrit, 2022), but GitHub was chosen due to familiarity.

5.2. Engineering the Tool

5.2.1. Configuration File

The configuration file is a JSON file that is read into the "UserConvo.jsx" file to provide the parameters and questions needed to render the CUI tool to the Creators specifications.

At the top of the configuration file there is a list of a parameters that the Creator can use to specify certain attributes of the tool that will be unique to them. Below are the following parameters that are at the start of the file:

```
-       "Parameters": [
-          {
-             "Screen Type": "phone"
-          },
-          {
-             "Avatar Image": "/images/AvatarMaker.png",
-             "Avatar Name": "Amy"
-          },
-          {
-             "Server Address": "http://127.0.0.1:33507"
-          }
-       ],
```

Figure 21 - Parameter example in JSON config file

The first parameter "Screen Type" denotes the type of screen that the tool will be viewed on. This will either be "phone" or "monitor", and each have slightly different styles to accommodate the viewing screen.

The second and third parameters relate to the "Avatar" the Creator wishes to use in the tool. The "Avatar Image" is the location where the picture of the "Avatar" is located. Creators are encouraged to save their pictures in that location under the name "AvatarMaker.png" and for

the best quality, to have it as a 20x20 pixel picture. The “Avatar Name” is the name that will be displayed alongside the picture in the header.

The next parameter is the “Server Address” which is the address of where the Creator is hosting the tool. This is important for both the frontend and backend to know where to store the conversation.

5.2.2. Conversation Hierarchy

In the “Conversation” array in the JSON file, in Figure 25, there are the list of questions to be asked by the chatbot. The tool will iterate over the questions in a linear fashion, pausing and waiting for an answer from the Participant before iterating over the next question. This is stored in JavaScript, throughout the conversation as an array called “conversation”, which appends both questions asked, and answers given to the array to show the full conversation.

The format of these questions in the configuration file is seen below:

```
-        {
-          "Type": "free",
-          "Message": "Where do you live?"
-        },
-        {
-          "Type": "choice",
-          "Message": "What's the weather like?",
-          "Choices": [
-            {
-              "Answer": "Warm"
-            },
-            {
-              "Answer": "Cold"
-            },
-            {
-              "Answer": "Windy"
-            }
-          ]
-        },
```

Figure 22 - Examples of question types in config file

Each question has a “Type” to say if it is a free text answer expected or a button response. This tells the tool to switch to the appropriate component at the bottom of the screen. The “Message” is the question to be asked to the Participant. There are also “Answer” parameters, these are the answers to the multiple-choice questions that the Participant can choose from. These are already predefined answers written by the Creator.

5.2.2.1. Free Text

This type of question expects a free text answer, where the user can type freely whatever they want into the input text box and submit it to be appended to the conversation. There is no limit to how much they can write, and they are free to edit their response before sending.

5.2.2.2. Multiple Choice

This type of question adds an extra attribute to the array object: “Choices” – this is an array with the options the Creator has provided to the Participant to select an answer. There is a limit of four options at max, for a given multiple choice question. This type of question tells the tool to hide the text input and submit button in replace of a list of buttons with the specified answers. There is also a limit to how large these buttons can be, so the Creator cannot use as much text as they wish. Using these types of predefined buttons allows for different paths in the conversation. This will be described in the next section.

5.2.3. Recursive Component

The recursive component is to allow the CUI tool to have a dynamic and specific conversation with the Participant. As mentioned previously there are two types of responses allowed in the CUI tool (free text or choice). For the choice answers it is possible to branch them so that if a Participant chooses an answer, they will be prompted by another follow-up question.

Take the figure below as an example:

```
-       {
-         "Type": "choice",
-         "Message": "How are you feeling today?",
-         "Choices": [
-           {
-             "Answer": "Good",
-             "Type": "choice",
-             "Message": "Why good?",
-             "Choices": [
-               {
-                 "Answer": "The Sun is out"
-               },
-               {
-                 "Answer": "I'm going to play Frisbee"
-               },
-               {
-                 "Answer": "I got this working"
-               }
-             ]
-           },
-           {
-             "Answer": "Bad",
-             "Type": "free",
-             "Message": "Why bad?"
-           },
-           {
-             "Answer": "Indifferent"
-           }
-         ]
-       },
```

Figure 23 - Example of nested multiple-choice questions in config file

Unlike the multiple-choice question in Figure 22, these answers have follow-up questions attached to them, depending on what the Participant answers. For instance, in Figure 23, if the Participant chooses to answer “Good” to the original question “How are you feeling?”, they will be prompted by another question “Why Good?”. The follow up question is another multiple-choice question; however, if they answered “Bad”, the follow up question will be different – “Why Bad?” and the CUI will ask for a free text response. And the last option “Indifferent” will not prompt the Participant for a follow-up question.

This gives the Creator a lot of options how to organise and construct the conversation, asking an array of multiple choice or free text questions that may/may not require a follow-up question.

The CUI tool, instead of linearly iterating over every question that it finds, will recursively go to the “child” questions until the line of questions has finished and then it will pop out of the recursive loop to the top level where it will linearly iterate over the next set of questions until another instance of recursion is found or the conversation has ended.

5.2.4. Storing Conversation (locally and cloud)

The conversation will be held in an array of objects in JavaScript. The structure looks like:

```
{'type': 'question', 'message': "What's your name?"}
{'type': 'answer', 'message': 'Luke', 'timeTakenForCurrentQuestion': 2}
{'type': 'question', 'message': "What's your age?"}
{'type': 'answer', 'message': '27', 'timeTakenForCurrentQuestion': 2}
{'type': 'question', 'message': 'Where do you live?'}
{'type': 'answer', 'message': 'Dublin', 'timeTakenForCurrentQuestion': 5}
{'type': 'question', 'message': "What's the weather like?"}
{'type': 'answer', 'message': 'Warm', 'timeTakenForCurrentQuestion': 3}
{'type': 'question', 'message': 'How are you feeling today?'}
{'type': 'answer', 'message': 'Good', 'timeTakenForCurrentQuestion': 2}
{'type': 'question', 'message': 'Why good?'}
{'type': 'answer', 'message': 'I got this working', 'timeTakenForCurrentQuestion': 3}
{'type': 'question', 'message': 'Did you enjoy chatting today?'}
{'type': 'answer', 'message': 'Sure', 'timeTakenForCurrentQuestion': 4}
```

Figure 24 - Example output of conversation

The messages are appended sequentially. The questions defined in the configuration file will only have a “type” and a “message” when appended to the array. The output of the answers, given by the Participant, will have “type”, “message” and “timeTakenForCurrentQuestion”. The last entity in the array is to tell the Creator how much time was taken (in seconds), by the Participant, to answer the previous question. This may be useful for their research, and it was not difficult to implement. A timer begins at the start of every question and when the answer is submitted the timer is reset. This is the format the conversation is saved in. It is saved locally, by creating a file and storing the JavaScript array in that file. The file is encrypted for security. There is also the option as previously described, of sending the conversation to the cloud.

The backend checks if the file containing the database configuration: "GOOGLE_APPLICATION_CREDENTIALS.json" is empty and if not, it pushes the file to the Google Cloud “bucket” called “CUI_Application” if it doesn’t already exist.

5.3. Cascading Style Sheets (CSS)

The tool is written in vanilla Cascading Style Sheets (CSS), without a CSS framework. Each component file received their own CSS file that helped design elements within the component. Aspects such as, relative size on the screen, colour and position on the screen were specified in these files.

As mentioned in “UI Design” in the “Design” chapter, the messages were given different colours, take up to sixty to eighty percent of the width and were spaced vertically apart from each other.

The only major difference between the style of the phone screen type and the monitor screen type is the relative width the CUI uses. For the phone screen type, the CUI will use one hundred percent of the screen, while the monitor screen type it will roughly use forty to fifty percent of the screen width as shown in Figure 15, in the “Design” chapter.

5.4. Security Considerations

One of the main issues concerning security with this tool is there are a lot of unknowns regarding how the Creator intends to use, store, and export the tool and the saved conversation. As it is intended to be open-source, there is no way to know who will be using it and for what purpose.

What questions will be asked of the Participant will also be a mystery. Is the research that is being conducted of a confidential nature and will the answers need to be kept private? For instance, if the Participant is being asked about health matters such as in the study (Trigo, et al., 2021), then the information should only be shared with those involved in the research.

It would be good to assume the content of the conversation between the chatbot and the Participant is confidential and needs to be private and secure. Ensuring the tool can provide some security for the Creator/Participant adds to the usability of the tool and it would take away some work from the Creator. This mostly involves encrypting the conversation and for it to either be stored where it is hosted or to be sent to a database specified by the Creator.

5.4.1. Where is the tool being Hosted? Is it Secure?

It will be up to the Creator to host the tool themselves. This could be on any domain on the web. The code for the tool will be stored on GitHub. Whomever intends on using it will download it and configure it to their needs before hosting it on their webpage and/or in their survey. Having the code available as an open-source software could be an issue if someone knows how the data is being encrypted/sent.

It is intended that the output file will be encrypted. This encryption will happen in the backend after the conversation has finished. The backend will be written in Python/Flask and the library in charge of encrypting the data will be "cryptography". This package uses Symmetric Encryption to encrypt the data, when using the "recipes layer" - specifically "Fenet" (pyca/cryptography, 2022).

5.4.2. Where is the Conversation Stored? Is it Secure?

5.4.2.1. Locally Stored:

There is a hope that the Creator will have a choice on how the conversation that takes place will be saved. Firstly, the conversation will be encrypted in a text file locally (as previously mentioned) so the Creator can easily access it wherever it is hosted. To help keep Participant data anonymous, each file containing each conversation will be named by collecting the timestamp at which the conversation is completed and appending it to the name of the file e.g., output2022-07-31 15:56:51.808335.

The tool uses encryption to first generate a "key" using the Fenet algorithm specified earlier. This key is stored in a file called "filekey.key", which is used to encrypt the file. The file is stored locally and when the Creator wishes to decrypt the conversation, they can login to their server and run a "decrypt.py" script that will use the key to decrypt the specified output file to show the plain text conversation.

It is recommended that once the conversations are recovered and moved somewhere more secure (perhaps a local machine), the key file should be deleted. This is to ensure that the same key is not used for all the conversations, and it is not sitting on the server in case an attacker invades the server and steals the key, thereby accessing the potentially private conversations.

5.4.2.2. Cloud Storage

If the researcher intends on storing the conversation data in a database, this would be preferable, as they can rely on the cloud's security protocol. The likes of Google, Amazon or Microsoft are better equipped to handle personal data rather than leaving it up to the individual.

The API endpoint you are sending to with these services usually use Secure Socket Layer (SSL) or Transport Layer Security (TLS) to encrypt the connection between the CUI location and the database, as seen with Amazon's relational database (Amazon Web Services, 2022).

These encryption protocols (SSL and TLS) are used throughout the web. Coupled with the encryption and authentication access of the database, this is the preferred method of sending, storing, and accessing the conversation.

5.4.2.3. Email

An option that has not been implemented in the tool but could be a potential option in the future is sending the completed conversation via email to the Creator. The Creator would need to provide an email address to send to, which will need to be encrypted.

The Python library "smtplib", provides a means to send emails. This also uses SSL and TLS, and as previously mentioned these protocols would be suitable (Python Software Foundation, 2022).

5.4.3. Security Concerns with Attacks

There are no real security concerns for anyone talking to the chatbot. No information can be obtained from the conversation or any internal workings if the Participant was to enter a script in the input field. The input field takes the response and adds it to an array that is later encrypted. No processing is done to the responses in the tool.

There is an intention to limit the number of responses for each question, which would prevent bot spamming the CUI. However, this isn't necessarily a security concern.

5.4.4. Conclusion

For this project, it is quite important to have security considerations while designing and implementing the CUI tool. Although these considerations are not necessarily the responsibility of the CUI tool, they are catered for to relieve some of the work on the Creator conducting the chatbot conversations, while also adding the usability to the tool. This includes encrypting the conversation that has taken place and storing it locally or ensuring secure transport of the saved conversation to a specified cloud provider. It is important to make precautions to secure the tool within the code, however some responsibility ultimately will fall on the Creator on how they implement the chatbot on their webspace.

5.5. Difficulties Faced

The implementation of this tool had a lot of challenges. When implementing the recursive button aspect of the tool, there needed to be a new function that was called when the tool recognised a new nested “Choice” parameter in the “Conversation” array within the JSON file. This function would address the nested questions without moving linearly through the array anymore. Once the last child question has been answered, the function pops back up to the surface of the nested question and continues the original loop. This was difficult to implement without the loop looking through all the nested questions, instead opting to display the ones that came directly above/below in the tree structure. The configuration had to be changed and tested with this new function too to make sure the right key words were giving the desired response.

There was a difficulty with the relative paths of the server address and cloud configuration files. It was difficult to setup for more than one environment since the path to these files would be different on a local machine versus on the Heroku (Salesforce.com, 2022) server that was setup to test this deployment procedure.

Setting up the cloud connection for the first time was difficult as there was no prior experience with Google Cloud Storage. Tutorials (Google, 2022) helped in the implementation of a “bucket” and how to post the output to the bucket using Python after the conversation had taken place.

The implementation of the CSS was one of the most difficult tasks, as the tool strives to accommodate different screens and the elements relative to the screen selected by the Creator. One aspect was the dynamic sizing of the input text box that was increased to a certain height before beginning to scroll. This was done by using a HTML “span” tag that will increase with the input and stating the max height of the element and providing an “overflow” option of “scroll” once the max height was reached.

Chapter 6: Evaluation

Due to the lack of Participant interaction with this tool, the main method of evaluating this tool is to compare the requirements that were specified in the “Requirements” chapter and how well these requirements have been met with a walkthrough of the tool from the perspective of the personas carrying out the scenario outlined in the same chapter.

Afterwards there are certain design rules called “Nielsen Heuristics” and “Cognitive Dimensions of Notations”, that will also be used to evaluate the tool. These are User Experience (UX) principles to evaluate applications on their usability and appearance. Jakob Nielsen had ten principles for evaluating interaction design and there are fourteen Cognitive Dimensions of Notations that the CUI will be evaluated against in this chapter.

It is important to evaluate what was required and if that was achieved; however, it is also important that the tool is evaluated by certain design best practices used in the wider community. These are tried and tested practices that remain relevant for applications today.

This chapter will give insights into these principles but for more detail, see the Appendix. Sections 6.2 and 6.3 will refer to different parts of the Appendix.

6.1. Reflection on Requirements


```
-   {
-     "Parameters": [
-       {
-         "Screen Type": "phone"
-       },
-       {
-         "Avatar Image": "/images/AvatarMaker.png",
-         "Avatar Name": "Amy"
-       },
-       {
-         "Server Address": "http://127.0.0.1:33507"
-       }
-     ],
-     "Conversation": [
-       {
-         "Type": "free",
-         "Message": "What's your name?"
-       },
-       {
```

```
-               "Type": "free",
-               "Message": "What's your age?"
-            },
-            {
-               "Type": "free",
-               "Message": "Where do you live?"
-            },
-            {
-               "Type": "choice",
-               "Message": "What's the weather like?",
-               "Choices": [
-                  {
-                     "Answer": "Warm"
-                  },
-                  {
-                     "Answer": "Cold"
-                  },
-                  {
-                     "Answer": "Windy"
-                  }
-               ]
-            },
-            {
-               "Type": "choice",
-               "Message": "How are you feeling today?",
-               "Choices": [
-                  {
-                     "Answer": "Good",
-                     "Type": "choice",
-                     "Message": "Why good?",
-                     "Choices": [
-                        {
-                           "Answer": "The Sun is out"
-                        },
-                        {
-                           "Answer": "I'm going to play Frisbee"
-                        },
-                        {
-                           "Answer": "I got this working"
-                        }
-                     ]
-                  },
-                  {
-                     "Answer": "Bad",
-                     "Type": "free",
-                     "Message": "Why bad?"
-                  },
-                  {
-                     "Answer": "Indifferent"
-                  }
-               ]
```

```
-           },
-           {
-               "Type": "free",
-               "Message": "Did you enjoy chatting today?"
-           }
-       ],
-       "Farewell": "Thank you for taking the survey!"
-   }
-
```

Figure 25 - Configuration file of a Researcher

As stated in “Requirement 1”, the tool must be accessible to both Creator and the Participant. This code, along with README with explanations are hosted on GitHub (GitHub, Inc., 2022), a free online repository for code, where everyone can see and use the code without having to pay. Once the Creator downloads the code, they can see the configuration file as shown in Figure 25. In the example of Joe and Caytlin, Joe can download the code from the repository and change the configuration file as specified in the README, achieving the fourth requirement, “Requirement 4”.

Although this satisfies Requirement 4, the configuration file could be intimidating for someone who is not familiar with JSON. Having a UI to help configure the tool would be a better option. Previously mentioned tools, such as “Botsociety”, allow this and it is easier for Creators to customize their chatbots. Having draggable buttons and responses and being able to view how these would look in the conversation could help the Creator with structuring the conversation.

The naming conventions can be a little confusing also. For instance, the “Message” tag in the configuration file, for the text displayed as the chatbot’s response, is not very self-explanatory and similarly for the “Answer” tag. More detail may be needed in the README file or better key words within the configuration file for the Creator’s understanding. The configuration file is a good start, but there is more that can be done to aid usability.

Something that may limit the success of “Requirement 1”, is the rendering of the tool for different screen types. Although the tool can render on a phone (small) or monitor (large) screen, this must be specified by the Creator of the tool before hosting it on their webspace. The issue with this is that not all the participants may use the same type of screen or at the very least it would be beneficial to not restrict all of them to a single type of screen.

```
conorchurch@Conors-MacBook-Pro my-cui-app % npm run build

> my-cui-app@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  55.72 kB (+8 B)  build/static/js/main.317bf769.js
  1.78 kB          build/static/js/787.c386f9ac.chunk.js
  1.18 kB          build/static/css/main.23fedbfa.css

The project was built assuming it is hosted at ./.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.

Find out more about deployment here:

  https://cra.link/deployment

conorchurch@Conors-MacBook-Pro my-cui-app % python3 flask/save_conversation.py
 * Serving Flask app 'save_conversation' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:33507
 * Running on http://192.168.4.152:33507 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 225-879-779
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /static/js/main.317bf769.js HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /static/css/main.23fedbfa.css HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /images/AvatarMaker.png HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /manifest.json HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /favicon.ico HTTP/1.1" 200 -
127.0.0.1 - - [12/Aug/2022 10:38:38] "GET /logo192.png HTTP/1.1" 200 -
```

Figure 26 - Running the tool on a server

Once the Creator of the CUI finishes configuring the tool via the Configuration file, they will deploy the tool on their server and Figure 26 is the output of that. Flask prints to the command line if somebody has retrieved their web page via a “GET” request. Here the Creator can check for any errors or when the conversation is done, it prints if the database option is available, and prints “Done”.

While deployment of the tool is quite simplistic, there is one slight modification that would aid usability and security to the tool. Using a Docker container (Docker Inc., 2022), could help run the tool in isolation from the Creator’s server it is being hosted on. This does not necessarily decrease the number of commands used for deployment but caters for different environments with less configuration needed between the tool and the host server. This is a minor tweak to the current tool but would be welcome to have.

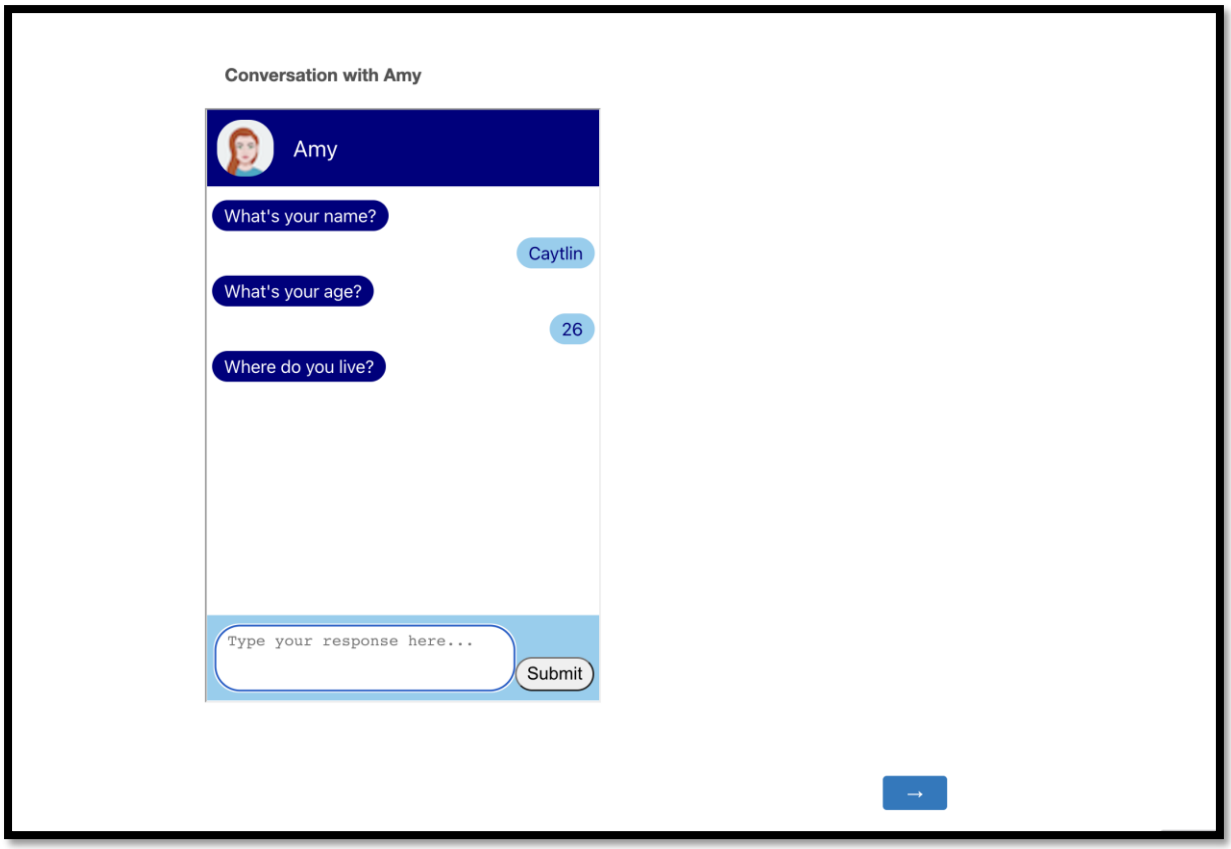

Figure 27 - Rendering tool in Qualtrics

The Creator can embed it into third party surveys such as Qualtrics seen in Figure 27 above, satisfying "Requirement 2". However, when using third party surveys, is no link between the completed conversation in the tool and the completed survey. This would mean somebody organising a survey with the embedded tool would find it more difficult to know which survey aligned to which chatbot conversation. This is a flaw with the implementation that will be worked on in the future. A minor flaw regarding this tool's usability is the surveys that the tool was integrated with. Libraries such as, "jsPsych", would give more options for Creators who are experienced with the JavaScript programming language and would like to customize the tool further within a survey, although "Requirement 2" has been satisfied, there are more surveys that the tool can be embedded in.

Figure 27 shows what the Participant sees while conversing with the rendered CUI tool. As described in the "Background" chapter, the UI is meant to mimic other messaging applications or customer service CUIs, both types following industry practices. This would satisfy "Requirement 5", meaning there is less dependence on the tool having to explain and be intuitive to the Participant, while also being familiar as if they have used a similar application before.

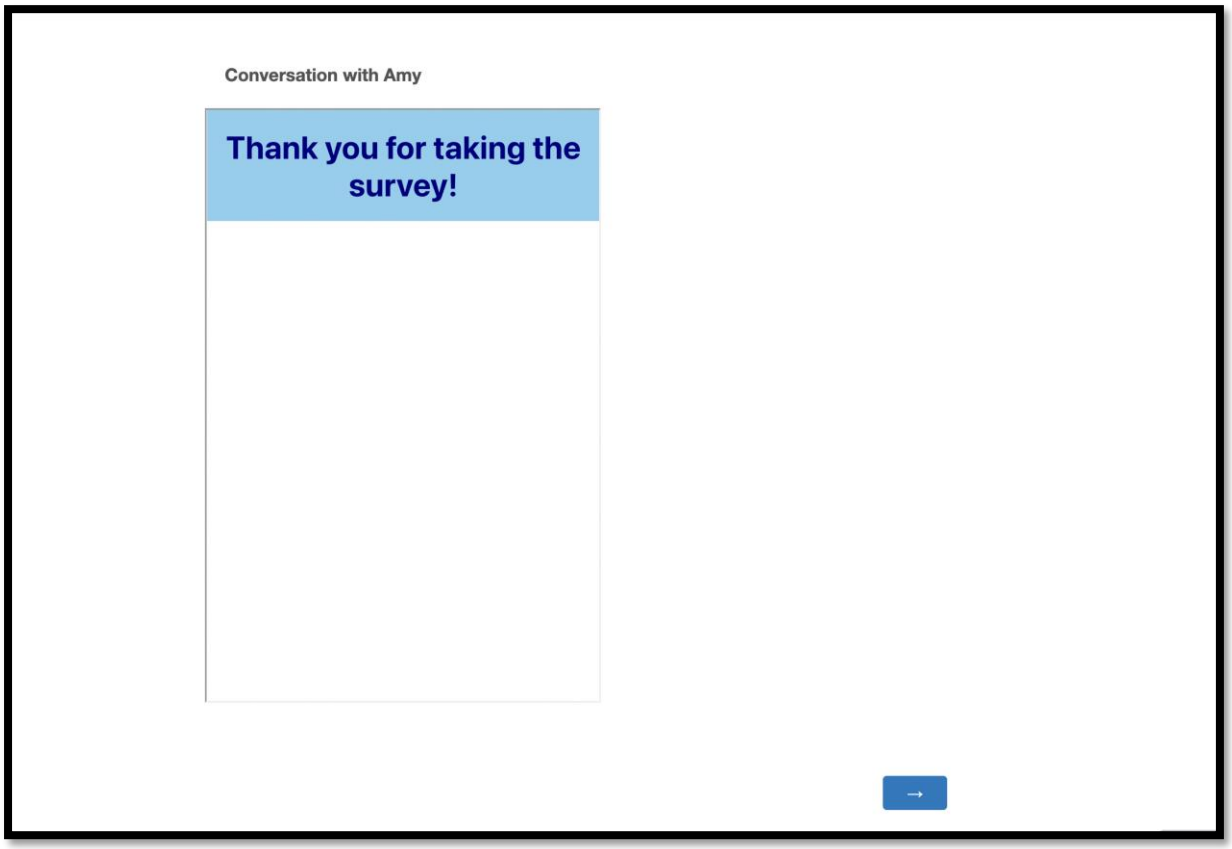

Figure 28 - End message to Participant after conversation

After the Participant, such as Caytlin in our example, has completed the interaction, they are presented with a message to inform them the conversation has finished. Figure 28 shows an example of such a message. This is defined in the configuration file as seen at the bottom of Figure 25.

```
conorchurch@Conors-MacBook-Pro my-cui-app % ls
Dockerfile              filekey.key             public
Procfile                flask                   requirements.txt
README.md               node_modules            src
build                   package-lock.json
conversations           package.json
conorchurch@Conors-MacBook-Pro my-cui-app % ls conversations
output2022-08-05 17:30:03.922906.txt    output2022-08-05 19:16:42.711846.txt
conorchurch@Conors-MacBook-Pro my-cui-app %
```

Figure 29 - Saved Conversations stored on the server

For the Creator of the tool, such as Joe in this case, they can view the contents of the "/conversations" folder and see an output like Figure 29 where files with a timestamp appear, each relating to a different conversation. Figure 30 is what the Creator can expect to see when these files have been decrypted to show the conversation output. This satisfies "Requirement 3" where the conversation should be saved for the Creator. The Creator can access these completed conversations in their server.

Having more options for Joe to store the conversations would certainly add to "Requirement 3"'s implementation. If there were more cloud providers to offer, or the option to be emailed the conversation.

```
{'type': 'question', 'message': "What's your name?"}
{'type': 'answer', 'message': 'Caytlin ', 'timeTakenForCurrentQuestion': 6}
{'type': 'question', 'message': "What's your age?"}
{'type': 'answer', 'message': '26', 'timeTakenForCurrentQuestion': 4}
{'type': 'question', 'message': 'Where do you live?'}
{'type': 'answer', 'message': 'Maynooth', 'timeTakenForCurrentQuestion': 4}
{'type': 'question', 'message': "What's the weather like?"}
{'type': 'answer', 'message': 'Warm', 'timeTakenForCurrentQuestion': 6}
{'type': 'question', 'message': 'How are you feeling today?'}
{'type': 'answer', 'message': 'Good', 'timeTakenForCurrentQuestion': 3}
{'type': 'question', 'message': 'Why good?'}
{'type': 'answer', 'message': 'I got this working', 'timeTakenForCurrentQuestion': 4}
{'type': 'question', 'message': 'Did you enjoy chatting today?'}
{'type': 'answer', 'message': 'Sure', 'timeTakenForCurrentQuestion': 6}

```

Figure 30 - Output of Conversation that has taken place

Overall, all requirements have been satisfied to some extent. They can be improved upon, which this thesis discusses in further in "Future work" in the next chapter.

6.2. Nielsen Heuristic

The Nielsen Heuristics are broad rules of thumb that are used for interaction design as principles to follow (Nielsen, 1994). These are ten rules that have been at the forefront of evaluating user-facing applications since their conception in 1994.

Evaluating this tool using the Nielsen Heuristic principles gave insight to what the tool had already thought of, but what also could be improved on. For instance, some of the heuristics that were well met were: "Visibility of System Status", "Consistency and Standards", "Error Prevention" and "Aesthetic and Minimalist Design". Certain aspects such as the "Visibility of

System Status" were achieved originally by looking at how other applications guide the user to accomplish their goal. The first message rendered in the tool will be from the chatbot itself, prompting the Participant for a response, by usually asking a question. The means of answering a question via free text or multiple-choice question is clearly labelled, for instance, the text input is labelled "Type your response here..." and the submit button is labelled "Submit". "Consistency and Standards" was followed when creating the tool by looking at designs of other like application shown in the "Background" chapter to view what can seem familiar to the Participant. The header and footer placement and the shape of the messages are like other social media messaging applications, making it easier for the Participant to recognise a similar format and similar functions of elements.

Some areas which the tool could improve on were: "User Control and Freedom" and "Help Users Recognize, Diagnose, and Recover from Errors". For instance, for "User Control and Freedom", there is no "escape" option for the Participant of the tool to alert the Creator of an issue, which was not originally thought of but was used to recognise what was missing. Another rule that could be learned from is "Help Users Recognize, Diagnose, and Recover from Errors". There are currently no error messages for the Participant to recognise. This could be an issue if something happens that is unforeseen up to this point.

6.3. Cognitive Dimensions of Notations

The purpose of the Cognitive Dimensions of Notations is to evaluate how easy a UI is to use (Muniz, 2016). A notation is an element within an interface that when interacted with can change the system (Muniz, 2016).

There are fourteen Cognitive Dimensions of Notations, some of which gave understanding how a Participant may interact with the tool/system and what was done already to make the tool easy to use, but also what could be done further to improve the usability.

Certain aspects the tool had done well already was the "Visibility" of the tool. The tool was well labelled/described ("Submit" on submit button/"Type your response here..." in the text input) or in the README file (How to deploy the tool and how to configure the parameters and questions). The scenario and requirements that were established, helped to make the tool more visible.

One principle that the tool may not have met as well was "Error-Proneness" due to there not being as much testing on different Creators' environments. This is something going forward that will be focused on to ensure the tool is usable on more than one set-up, other than Heroku server and the local machine used in this thesis.

There was a lot to learn from both these sets of principles about what was missing in the tool and what was already implemented from looking at different messaging applications and chatbots. It helped shape what to focus on in going forward to improve the tool's usability.

Chapter 7: Discussion

7.1. Implications

This project set out to design a tool to help researchers study CUIs and investigate how they can be designed. This required learning about the requirements such researchers have for these tools and why this specific tool is needed.

This tool is ready to be used in academic research for simple conversations. The results of which can be easily studied. The tool can be built on, providing greater functionality. One example that would help the tool's initial usability is having the tool dynamically render to suit the screen the Participant is viewing it on, without the Creator having to specify it. The tool can still cater for different screens now, but it must be manually specified which may make it more awkward to use if the Creator does not know the type of screen the Participant will use.

Another feature that may hinder the immediate use in academic research, would be the lack of integration into other surveys such as "jsPsych". A missing feature with the integration with third party surveys is the connection between the completed survey and the completed chatbot conversation. This may have to be corrected if a researcher is to use the tool in their work.

One obvious short coming is that this chatbot CUI is lacking in natural language understanding (NLU), that would have greatly broadened its usability, but that did not fall into the initial scope of the project.

Otherwise, the tool is ready for basic Human-Computer Interaction for free and it is easy for anyone who has no programming experience to deploy. The tool is configurable to such an extent that a researcher can use it in their studies while studying simple chatbot design and user interaction with chatbots. The conversations can be saved, and options are available for them to store these conversations to be used later in their research.

The goals of this tool for the purpose of configurability, usability and embeddability have been achieved, but there are still aspects of the tool that can be improved moving forward.

If anyone else was about to undertake a similar project, they should refer to the relevant research in this area and fine tune their requirements. There is a vast scope for research in Human Interaction Design and refining what exactly someone wants to achieve helps them focus on their goals and design of the system to be constructed.

7.2. Limitations

Due to the scope and time restraint of this research project, there were certain limitations that restricted the features of the tool that will be noted in this section.

The Creator of the tool does the processing of the questions/answers pairing after the conversation has finished. The tool does not have enough computing power to process the Participant's answers while still conversing. This would require more depth of knowledge in natural language.

Another limitation that was encountered was not having Participants use the tool to evaluate the effectiveness of it. This would require ethical permission, which was not obtained due to limited time while also building the tool. Not having Participants meant that there was not as much diverse feedback about the tool. It would benefit from different opinions from others with different cultural/professional backgrounds.

7.3. Future Work

There are some areas of improvement which can be considered after this thesis.

One of the requirements of this tool was for it to be embeddable into other survey tools. One possible extension of this is creating a plugin for this tool in the "jsPsych" library. This would grant more customization if the Creator of the tool had experience with the JavaScript programming language.

While using third party surveys, it would be a welcome addition for the Creator of the survey could pass a token to the tool, to match the chatbot conversation with the completed survey, as described in the "Implications" section above. This would mean the Creator of the survey/tool would know which completed survey linked to which output file on their server. This would be a priority going forward as researchers may have many Participants in a study so they will need a way to link the surveys to the conversations.

To make the CUI seem more human like and more familiar to the Participant, a possible addition could be to have something visually fill the delay between the Participant responding, the CUI "thinking" about what was said and its response. This could be in the form of three dots that move that is seen in some messaging applications indicating the other converser is typing. This is to convey to the Participant to expect a response from the chatbot.

To make this tool more dynamic, the tool could recognise and style for the type of screen the Participant is conversing on. This would mean the Participant could access the tool from whatever screen, giving more flexibility for different Participants with different screen types using the tool at the same time. An in-between of the current state of the tool and this preferential way, is to ask the Participant what screen type they are using before rendering the tool. This means less setup for the Creator.

As mentioned in the previous section, another thought moving forward would have Participants use the tool to give feedback on the usability and the design aspects of the CUI. This could only happen with ethical approval and so with more time this could be achieved.

It is difficult to cater for every Creator's environment when setting up the tool. Therefore, the use of a Docker container may help in deploying it on whatever server the Creator runs it on. This has added benefits of security by not allowing any intruders into the parent server it is running on (Docker Inc., 2022). A Docker container wraps the application into an isolated process with ports to access it. It can be scaled up or down with multiple instances of the Docker container and with port access it can be independent of the parent webserver it is being hosted on.

The configuration file is quite archaic and so developing a UI for Creators to use with their CUI's would be a focus in the future. The input file sometimes is not very user friendly and intuitive. Having a UI which has buttons and draggable icons could be easier for Creators without coding experience to use. Even running commands on a server may be difficult. One of the main focuses was making the tool easy to use, this would be the next step in this direction.

Another possible future work could be to add an audio version of the tool so Participants could speak to the tool instead of typing their responses. The tool could have an automated voice, that would converse with the Participant. This would be helpful for Participants who are visibly impaired or are not able to use a keyboard. However, this would be a very ambitious expansion of the project.

In Future, the tool could be made "Smarter" by adding Artificial Intelligence (AI) and Machine Learning to gauge the language used in answers and changing the questions. Due to the time and the resources available, it was not feasible for the tool to have a component that dealt with conversational language. This would have required a lot of time and computing power. This would be for the purpose of taking in the free text answers and looking in depth at their meaning. One example of this is if the CUI needs the Participant to retype their message, this can be done if the input is being reviewed in real time. Humans go through "conversational repair" very often in everyday conversations (Deibel & Evanhoe, 2021). It would be helpful to add a response in circumstances where the bot does not fully understand what the Participant

has typed. If it does not know it can ask the Participant to clarify their response. This usually happens between humans, if the speaker cannot be heard, if the listener requires more information to understand what the speaker means and so on. This is an important aspect of conversations that could be looked at in the future when the bot is able to make sense of free text responses. The implementation of asking to repeat, would be in the same manner as the recursive button question, where an additional message is added to the conversational array without the array counter incrementing.

Chapter 8: Conclusion

This thesis set out to create a compact, configurable, and easy to use tool for researchers in the field of Human-Computer Interaction. To do so, I needed to research why tools like this were important and why there was a need for this tool above others that had already been created.

I have gained insight into the growing body of software and research surrounding the topic of Human-Computer interaction. There has been a significant uptake of these technologies in our daily lives, but there is still so much that can be improved upon. Even the smallest things can make a huge difference in how we interact with our devices. Language being quite at the forefront of this, especially with increasing use of voice assistants. The project wasn't just simply about a chatbot, it gave insight into how humans communicate with each other too.

UI design was a huge part of this project. How elements of the chatbot were laid out before the Participant became crucial to in its implementation. The tool needed to be easy to use and appealing to whomever used it. Therefore, requirements building dictated how the tool behaved and what it did. This thesis describes the considerations, compromises and choices that went into the Design and Implementation of the tool, which were later evaluated and discussed in detail.

One of the biggest difficulties was establishing the requirements of this project and refining the scope and the direction in which to take. This area of research is very broad, and no two people would have the exact same tool by the end. Another difficulty during the process was designing for a broad range of use cases, as each Creator's use may be different. Another was designing for configurability while trying to balance design and functionality. This led to compromises throughout the process.

As laid out in this thesis, I have shown the inner workings of a tool I created that can be configured and rendered easily for different goals and in different environments. I explain its functions and its limitations and what work can be done to improve it in the future.

I feel I have achieved my goals I set at the start of this project, by Creating a Web Tool for Researching Conversational User Interfaces, with the potential to build on going forward.

Bibliography

Python Software Foundation, 2022. *smtplib — SMTP protocol client.* [Online] Available at: https://docs.python.org/3/library/smtplib.html [Accessed 15 August 2022].

2001: A Space Odyssey. 1968. [Film] Directed by Stanley Kubrick. s.l.: Stanley Kubrick Productions.

Amazon Web Services, Inc. , 2022. *AWS Home Page.* [Online] Available at: https://aws.amazon.com/ [Accessed 16 August 2022].

Amazon Web Services, 2022. *Connecting to an Amazon RDS DB instance.* [Online] Available at: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_CommonTasks.Connect.html [Accessed 22 March 2022].

Amazon, 2022. *Amazon shopping: Echo Dot (4th Gen, 2020 release) | Smart speaker with Alexa | Charcoal.* [Online] Available at: https://www.amazon.com/dp/B07XJ8C8F5/ref=s9_acsd_al_bw_c2_x_1_t?pf_rd_m=ATVPDKIKX0DER&pf_rd_s=merchandised-search-4&pf_rd_r=FTM5RRCWX67JBBC66KTX&pf_rd_t=101&pf_rd_p=d1a1911b-4074-43b1-b16e-87680164fdb0&pf_rd_i=9818047011 [Accessed 15 August 2022].

Boehm, B. & Basil, V. R., 2001. Software Defect Reduction Top 10 List. *Computer,* 34(1), pp. 135-137.

Botmock, 2021. *Botmock.* [Online] Available at: https://app.botmock.com/ [Accessed 10 November 2021].

Botsociety, 2021. *Botsociety.* [Online] Available at: https://botsociety.io/ [Accessed 10 November 2021].

Cox, G., 2021. *About ChatterBot.* [Online] Available at: https://chatterbot.readthedocs.io/en/stable/ [Accessed 12 August 2022].

Deibel, D. & Evanhoe, R., 2021. *Conversations with Things.* New York: Rosenfeld.

Docker Inc., 2022. *Docker overview.* [Online] Available at: https://docs.docker.com/get-started/overview/ [Accessed 2 July 2022].

Facebook, 2022. *Facebook Messenger*. [Online] Available at: https://www.messenger.com/ [Accessed 20 March 2022].

Facebook, 2022. *Whatsapp.* [Online] Available at: https://www.whatsapp.com/ [Accessed 20 March 2022].

Følstad, A. & Brandtzæg, P. B., 2017. Chatbots and the new world of HCI. *Interactions, Volume 24, Issue 4*, pp. 38-42.

Gerrit, 2022. *Gerrit Code Review*. [Online] Available at: https://www.gerritcodereview.com/ [Accessed 15 August 2022].

GitHub, Inc., 2022. *GitHub home page.* [Online] Available at: https://github.com/ [Accessed 5 March 2022].

GitHub, Inc., 2022. *GitHub Logos.* [Online] Available at: https://github.com/logos [Accessed 18 August 2022].

GitLab B.V, 2022. *About GitLab.* [Online] Available at: https://about.gitlab.com/ [Accessed 15 August 2022].

Google, 2022. *Android.* [Online] Available at: https://www.android.com/ [Accessed 15 August 2022].

Google, 2022. *Cloud Storage client libraries.* [Online] Available at: https://cloud.google.com/storage/docs/reference/libraries#client-libraries-usage-python [Accessed 15 August 2022].

Google, 2022. *Google Cloud.* [Online] Available at: https://cloud.google.com/ [Accessed 2 July 2022].

Google, 2022. *Hey Google - Google Assistant.* [Online] Available at: https://assistant.google.com/ [Accessed 15 August 2022].

Gratch, J., Kang, S.-H. & Wang, N., 2013. Using Social Agents to Explore Theories of Rapport and Emotional Resonance. *Social Emotions in Nature and Artifact*, pp. 181-197.

Grinberg, M., 2014. *Flask Web Development.* s.l.:O'Reilly Media, Inc..

Grudin, J. & Jacques, R., 2019. *Chatbots, Humbots, and the Quest for Artificial General Intelligence.*. Glasgow, Scotland UK, Association for Computing Machinery.

Harms, J.-G., Kucherbaev, P., Bozzon, A. & Houben, G.-J., 2018. Approaches for Dialog Management in Conversational Agents. *IEEE Internet Computing,* 23(2), pp. 13-22.

Las Radas, 2017. *Las Radas Home page.* [Online] Available at: https://www.lasradas.ie/ [Accessed 10 August 2022].

Leeuw, J. R. d., 2015. jsPsych: A JavaScript library for creating behavioral experiments in a Web browser. *Behaviour Research Methods,* pp. 1-12.

Lison, P., 2015. A Hybrid Approach to Dialogue Management Based on Probabilistic Rules. *Comput. Speech Lang.,* 34(1), pp. 232-255.

Lucas, G. M., Gratch, J., King, A. & Morency, L.-P., 2014. It's only a computer: Virtual humans increase willingness to disclose. *Computers in Human Behavior,* Issue 37, pp. 94-100.

Meta Platforms, Inc., 2022. *React A JavaScript library for building user interfaces.* [Online] Available at: https://reactjs.org/ [Accessed 25 January 2022].

Microsoft, 2022. *Azure.* [Online] Available at: https://azure.microsoft.com/en-us/ [Accessed 16 August 2022].

Muniz, F., 2016. *A Usable Guide to Cognitive Dimensions.* [Online] Available at: https://www.uxbooth.com/articles/a-usable-guide-to-cognitive-dimensions/ [Accessed 2 August 2022].

Murad, C. & Munteanu, C., 2022. *"Voice-First Interfaces in a GUI-First Design World": Barriers and Opportunities to Supporting VUI Designers On-the-Job.* Glasgow, s.n.

Nielsen, J., 1994. *10 Usability Heuristics for User Interface Design.* [Online] Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed 2 August 2022].

Orlowski, A., 2017. *The Register.* [Online] Available at: https://www.theregister.com/2017/02/22/facebook_ai_fail/ [Accessed 11 August 2022].

Preece, J., Rogers, Y. & Sharp, H., 2015. In: *Interaction Design.* West Sussex: John Wiley & Sons Ltd., pp. 350-384.

pyca/cryptography, 2022. *Welcome to pyca/cryptography.* [Online] Available at: https://cryptography.io/en/latest/ [Accessed 22 March 2022].

qualtricsXM, 2021. *Qualtrics home page.* [Online] Available at: https://www.qualtrics.com/uk/ [Accessed 20 November 2021].

Rough, D. & Cowan, B., 2020. Don't Believe The Hype! White Lies of Conversational User Interface Creation Tools.. *In Proceedings of the 2nd Conference on Conversational User Interfaces (CUI '20),* p. 1–3.

Salesforce.com, 2022. *Heroku Login.* [Online] Available at: https://id.heroku.com/login [Accessed 24 July 2022].

Software Freedom Conservancy, 2022. *Git.* [Online] Available at: https://git-scm.com/ [Accessed 16 August 2022].

Trigo, M. J. G. et al., 2021. *ALTCAI: Enabling the Use of Embodied Conversational Agents to Deliver Informal Health Advice during Wizard of Oz Studies.* New York, s.n.

Twitter, 2022. *Twitter Explore page.* [Online] Available at: https://mobile.twitter.com/explore [Accessed 15 August 2022].

Wikipedia, 2022. *HAL 9000.* [Online] Available at: https://en.wikipedia.org/wiki/HAL_9000 [Accessed 5 August 2022].

Zara, 2022. *Zara help page.* [Online] Available at: https://www.zara.com/ie/en/help [Accessed 2 July 2022].

Appendix

Nielsen Heuristics

Visibility of System Status

This section specifies that the design should, through appropriate feedback and a reasonable amount of time, constantly keep the users informed about what is happening (Nielsen, 1994).

For this CUI, the conversation will start with an opening message to the chatbot. The first message is generally a question for the Participant to engage with. It is the Participant's turn to respond just like on a regular messaging application.

The Participant is indicated by the input to "Type your response here...", when responding to the bot. This is quite straightforward, as it is like other messaging applications.

When the conversation with the bot is complete, the CUI window will render a message on screen and display to the Participant. The Creator of the CUI can change this to whatever they would like, but it normally will inform the Participant that the interaction has finished. By default, it says "Thank you for taking the survey!".

The CUI could be slightly better here when introducing the Participant to the chatbot, this could be fixed in the first message by the bot, or an introductory page, explaining what will happen. The assumption here is that the Creator will inform the Participant on how to interact with the tool before they proceed. Adding the three "dots" while the bot is responding would have helped keep the Participant informed on what was happening.

Having the "Farewell" page at the end tells the Participant that the conversation has ended, and they can inform the Creator or can close the tool.

Match between System and the Real World

This section specifies that the design's language should be understandable to the user. The user should be able to read the words, phrases and concepts used in the design, without it being unintelligible. The flow of the design should follow a logical and natural order (Nielsen, 1994).

Here this heuristic is not necessarily relevant to the tool. The Creator of the tool is responsible for configuring the name of the chatbot, the "Farewell" page and all the questions.

The only relevant part is possibly the README for the Creator, which is written in English and refers to aspects of the configuration file to help them set up and deploy the tool.

User Control and Freedom

This section specifies that the design should allow for user's mistakes. They need an "emergency exit" to leave the current action without a tedious amount of effort (Nielsen, 1994).

The tool does not necessarily have a clear exit button, like in messaging apps where they need to exit the application/tool if they wish to exit the conversation. Unfortunately, they cannot undo a response if they have already sent it, like messaging apps. However, the Participant does have the option to write a response and then to delete or edit their it before hitting send. There is no time limit so they can take as much time as needed to edit and send their response. This gives the control and freedom of time and space.

For button responses they have as much time as they need, but they have only a select few options, that the Creator has given them.

Consistency and Standards

This section specifies that the design should make it simple for the user by following industry standards. The user should not have to think the words or situations, or actions have different meanings (Nielsen, 1994).

This tool does well here as it is designed like many messaging applications. There is an input, which says “Type your response here...”. This is straight forward and is like other applications. Likewise, there is a submit button which submits the Participant’s response, which is self-explanatory and performs the same action as other applications.

Some confusion may come from the button options; however, it would be like customer service chatbots where a question is asked, and button answers are shown. Also, like suggestion buttons on the Android operating system (Google, 2022).

Error Prevention

This section specifies that the design should prevent problems or issues arising from the beginning. This can be done by check for error-prone conditions or removing them and ask the user to confirm an action before it is committed (Nielsen, 1994).

Some considerations to the errors were, at first, only allowing one answer per user response. This was to stop if a user wanted to spam the chat, or a bot spammed the chat for the researcher. However, for free text options the user can write as much as they would like.

Another is if the Participant was to try a SQL injection of some code; however, the answers to the server as stored in an array and no operation is executed on those answers after they are submitted, simply they are appended to a JavaScript array and saved to a file.

Recognition rather than Recall

This section specifies that the design should make options, actions, and elements visible to decrease the memory load of the user. Information about aspects of the interface should not have to be remembered by the user. The design’s information should be easily retrievable and visible whenever the user needs it (Nielsen, 1994).

As stated in a previous section, the actions a Participant can take are labelled well for them to understand what to do if interacting with this tool for the first time even without having used a messaging application. The input section clearly tells the user to type their response and the submit button is labelled “Submit”, for them to know.

The button options are slightly less explanatory, however when the submit button vanishes and the button selection appears, the buttons look like the submit button. From this, it is possible for the Participant to assume they need to select/click that option to submit it.

Flexibility and Efficiency of Use

This section specifies that the design should allow for shortcuts for experienced users if they are familiar with the design to speed up the interaction of the system (Nielsen, 1994).

The tool does not necessarily allow for shortcuts for experienced users. The only "shortcut" that the tool caters for is if the Participant is interacting with the tool with a keyboard and mouse, they can hit the "Enter" key to submit their "free text" responses without having to click the "Submit" button. Otherwise, the interaction is similar for both experienced and inexperienced Participants.

Aesthetic and Minimalist Design

This section specifies that the design should not contain any unnecessary information. If the interface is crowded with information not yet needed by the user it hinders their relative visibility (Nielsen, 1994).

The design of this tool only has relevant aspects displayed to the Participant. They need the text input, the message telling them to type their response, the submit buttons and the text in the buttons. They are displayed the conversation that has come before and the name and picture of the "Avatar" they are talking to.

There is also a message at the end of the conversation to tell them/indicate that the conversation has ended with the chatbot.

Help users Recognize, Diagnose, and Recover from Errors

This section specifies that the design should help users recognize, diagnose, and recover from errors. The errors should be presented in language that is understandable and outline the problem (Nielsen, 1994).

So far, there is no error messages that can be displayed to the Participant. There is no reason for such a message, unless the chatbot does not load or render.

Help and Documentation

This section specifies that the design should provide help and documentation. It may not be necessary to provide additional explanation, but the user may need documentation to help complete their tasks (Nielsen, 1994).

There is documentation provided for the Creator of the CUI on how to customise their design on the GitHub page. This provides information on what the tool will allow, for instance, there is a limit to how many button options the tool allows and how many characters for each option, and what is best for the appeal of the application, like the “Avatar” picture is best at 20x20 pixels.

The README outlines how best to customize and then deploy the tool on their environment, how to setup the accepted cloud provider (Google Cloud), followed by how to retrieve the information, via the cloud or decrypting the messages locally.

Cognitive Dimensions of Notations

Viscosity

Viscosity in relation to systems, is how much work a user must do for a small change to occur. A highly viscous system requires a lot of effort for the user to complete their goal (Muniz, 2016).

For the Participant that converses with the CUI, there is not much they need to do except answer whatever questions the Creator has put into the configuration file and this may be more taxing if the responses are free text.

For the Creator of the system, there have been efforts to make sure they need very little setup before they can run the CUI on their server. For instance, if they have configured their parameters and questions in the configuration file and have run “npm run build” to build the

application, they should just need to run the Python/Flask server to deploy the tool. The script was purposely made to look at the build directory of the React application.

However, there are certain aspects of the tool that the Creator of the CUI will need to fix that could be tedious, for instance, the routes are not relative so they may need to change relative to their directory/server path.

Visibility

This aspect relates to how easily users can find elements of the system that will help them accomplish their goals (Muniz, 2016).

For the Participants interacting with the system, the input element is clearly labelled for them on how to respond. The submit button is also clearly labelled. The button options have words that could be thought of as responsive, however the system doesn't say to push them, which may lead to confusion.

For the Creator of the CUI, the README on the GitHub repository, outlines where to go to configure the CUI questions, "Avatar" characteristics and their server address. It also outlines how to deploy it on their system, setup the preferred "Cloud" operator and decrypt the encrypted conversations.

Premature Commitment

There is no premature commitment for the creator or the user interacting with the CUI (Muniz, 2016).

The only commitments are made by the Creator when deciding the type of questions to have, the text of the questions, the "Avatar" they have and where they are hosting the tool (server address). This is all relevant information that needs to be obtained by the tool before deploying. There is not premature commitment here.

Hidden Dependencies

There is no dependency in the Interaction of the CUI (Muniz, 2016).

There are dependencies when creating the CUI. The server address and any other configurations need to be given before the tool is deployed, but this is stated in the README for the tool to work on the Creator's server.

Role-Expressiveness

This aspect notes that elements of a system should be easily understood or obvious to the user (Muniz, 2016).

For the Participant interacting with the system the input field is clearly stated and the submit button is labelled. The button options look like the submit button, but they have a similar function to be pressed and submit an answer.

For the Creator in the fields of the configuration file, the parameters and their key names are clearly outlined to describe what they relate to. For example, a question "Type" is either "free" or "choice" to denote if the CUI renders a free text answer or a button choice answer. This is also explained in the README.

Error-Proneness

As described in the previous section, this relates to how likely it is for a user to be guided poorly (Muniz, 2016).

The elements of the rendered CUI are clearly labelled to the Participant. The Participant cannot submit more than one answer and is told when the conversation has ended.

The issues may arise when first deploying the CUI. The README outlines how to build and deploy the CUI tool, however if the Creator does not input the correct server address when the

tool is deployed, or the correct “Cloud” configuration file, there could be an issue. This will be addressed in the README and made clear that the tool will not work otherwise.

Abstraction

This refers to the way in which different elements are grouped together to be viewed as one entity either to help a UI element be understood by the user or to lower viscosity (Muniz, 2016).

For the Participant interacting with CUI, the level of abstraction is low, as there are only two elements grouped together to talk to the CUI. This is the submit button and the input text box. There is also the submit buttons which contain the answers that will be submitted.

For the Creator of the system, there are several elements together that help lower the viscosity of the tool. For instance, the Python script when first executed, deploys the frontend, so there is no need for the Creator to also deploy that also. They only need to build the frontend before deploying.

They configuration file parameters are also tied closely to the code so they Creator does not need to change the codebase to change their questions, type of questions or the “Avatar”, they just need to change the configuration file and the place where the “Avatar” attributes are stored for the CUI tool to render.

Secondary Notation

This denotes any extra information used to describe a UI element to the user (Muniz, 2016).

There would be no Secondary Notation for the Participant interacting with the CUI tool; however, there would be some for the Creator of the tool.

For the Creator of the tool, if they wish they can try changing the codebase after downloading it from the repository. This is so they can make the tool more specific to their needs if they have knowledge of JavaScript and can read and understand the code. The Secondary Notation here is the comments in the codebase to provide information about what different aspects of the code does. For instance, if the Creator wants to know the inner workings of a function in a

file there will be comments outlining what the function does and what are the inputs and outputs of the function.

Closeness Of Mapping

This describes how accurate an element is to the action it performs (Muniz, 2016).

For this tool, the label “Submit” on the submit button is accurate as it submits the Participant’s response to the tool. The label “Type your response here...” on the input box is accurate as the Participant can type whatever they would like into the chat box. The other buttons submit exactly the text that is on the buttons, which is not necessarily straight forward, but they are like the submit button in design, so the Participant is to assume the same function (to press/click them).

For the Creator of the CUI, certain actions are also quite related to elements. For instance, the Flask script for running the application, does indeed deploy both the backend server and the built frontend to where it is hosted.

Consistency

This follows the idea that there are consistent patterns in the system, to let the user recognise familiar symbols as they use the tool (Muniz, 2016).

For the Participant interacting with the CUI, the buttons for submitting the free text response and the buttons for submitting the predefined answers are consistent in look as they carry similar functions – to be pressed/clicked. The input box is the same for every free text answer so no need to relearn here.

For the Creator setting up the CUI, the actions to deploy the tool are slightly different but there are not many ones. The act of changing the questions is outlined in the README and follow a similar pattern.

Diffuseness

This is the amount entities or symbols that are required to express a meaning (Muniz, 2016).

Here there are very few symbols/entities to express something to the Participant. The buttons and input box are clearly defined and are labelled. The size of these elements is also reasonable, and thought were put in to limit the size of such elements by the Creator.

For the Creator, the codebase is quite simple in design, there are a couple important files that need changing and they are organised well. The README is quite verbose however, to explain the tool and how to customize and use it. This is a necessity for the sake of usability by the Creator.

Hard Mental Operations

This refers to operations that require the user to think hard to interact with the system in some way. If they need a pen and paper, it would be very complex (Muniz, 2016).

For the Participant interacting with the system, there is no real hard or complex actions they need to take. They just need to be able to think of the response the chatbot is asking them and respond via text or buttons. There is a bit of the complexity left up to what questions the chatbot asks, which is decided by the Creator of the app, but otherwise the tool is designed for them to easily respond and visually follow the conversation.

For the Creator of the tool, the complexity of the tool depends on how much customization they wish to achieve. For instance, if they wish to change the questions alone, then the complexity is fairly each, but if they wish to change how the chatbot behaves, then this requires them changing the codebase and adds complexity to the tool.

Provisionality

This refers to users performing non-permanent actions before committing fully to their choice of action, increasing the usability of the system (Muniz, 2016).

For the Participant interacting with the CUI, there is a little provisionality in the CUI, but not much. They can type their message in the text box and review it before hitting send, and they can view the button options before hitting send, but once they hit send, they cannot undo that action.

For the Creator of the CUI there is a lot more provisionality. They can run the system locally on their local machine before they deploy it to their server where it will be hosted. This gives them an option to see the tool and how it renders before deploying. It allows them to make changes in the configuration file and cloud storage also. Even once the tool is being hosted it can be rebuilt and run again on the server. The only thing that it cannot do is have these changes implemented while a Participant is interacting with the CUI, this will cause disruption and loss of conversation.

Progressive Evaluation

This section asks if a system is partially complete can it be executed to receive feedback (Muniz, 2016).

For the Participant interacting with the system, there is no real feedback to be given, unless the Creator puts in messages along with their questions to say “You’re doing great! Only a few more Questions” and by this time the system is fully up and running.

For the Creator this aspect is more applicable. It is possible for the system to be progressively evaluated. The Creator can input some but not all their questions, for example, to see how they render and what it looks like. Even before they change anything, they can run the tool locally and can see what it looks like and how it interacts with.

There are chances to evaluate the tool before it is hosted on a server. The feedback can be very visual as the Creator can see the questions that they put into the configuration file, appearing on the screen. Running it locally and seeing the changes that have been made or testing if the cloud is properly connected is another example.