Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers

Andrew Laing

Presented in part fulfilment of the requirements
for the degree of BSc (Hons) Computing (OU top-up)

at the
City of Liverpool College 2019

Abstract

Background

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioural disorders. Symptoms include hyperactivity, impulsiveness, and difficulty in concentrating. Cognitive Behavioural Therapy (CBT) can help to reduce ADHD symptoms.

This paper details the development of a software application capable of being used within CBT scenarios to increase ADHD sufferers' levels of attentiveness. Users of the application try to increase their attention span by conversing with an Artificial Conversational Agent (chatbot), signalling each time it makes a conversational mistake with a button press.

Objectives

Using the application will help ADHD sufferers to increase their attentiveness. Conversation logs produced by the application will provide discussion material for follow-up sessions with CBT professionals.

Methods

Staff and students from the City of Liverpool College took part in a study in which they used the application for 10 minutes, playing the role of an ADHD sufferer. They then completed a questionnaire to provide information about their experience and make suggestions for improvements.

Results

The application was well-received by test participants, whose attentiveness increased during their use of the application. A basic analysis of chatlogs indicate that they could provide excellent material for CBT sessions. These findings suggest that chatbot-based applications can function as useful adjuncts to therapy sessions, and that further research could produce effective alternatives to current solutions.

Keywords

ADHD, Conversational Agent, Computerised Cognitive Behavioural Therapy

Acknowledgements

Firstly, I would like to thank my supervisor Mike Kimber and the rest of the teaching staff at the City of Liverpool College for their support and guidance throughout my studies.

I would also like to thank the open-source software community whose projects provided me with the tools and motivation necessary to learn programming.

Finally, I would like to thank my family and those who encouraged me to study computing at a higher level.

Contents

Table of Figures

List of Tables

List of Abbreviations

ADHD	Attention Deficit Hyperactivity Disorder
AIML	Artificial Intelligence Markup Language
A.L.I.C.E	Artificial Linguistic Internet Computer Entity
CBT	Cognitive Behavioural Therapy
cCBT	Computerised Cognitive Behavioural Therapy
CD	Compact Disc
GIMP	GNU Image Manipulation Program
GP	General Practitioner
GPL	GNU General Public Licence
GUI	Graphical User Interface
MACH	My Automated Conversation Coach
OOP	Object-Oriented Programming
PDF	Portable Document Format
PEP	Python Enhancement Proposal
PNG	Portable Network Graphics
RGB	Red, Green and Blue
SAPI	Speech Application Programming Interface
TTS	Text-to-Speech
OU	Open University
WAV	Waveform audio file format
XML	Extensible Markup Language

Chapter 1 - Introduction

1.1 Context

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioural disorders. Studies found that it affects 5% of children (Polanczyk, 2007) and 3% of adults. (Fayyad, et al., 2007)

Symptoms of ADHD include hyperactivity, anxiety, impulsiveness, and difficulties in concentrating. These symptoms can strongly interfere with the interpersonal relationships of an ADHD sufferer, affecting their performance at school or work.

As there is no consensus agreement as to what causes ADHD, current treatments focus upon providing relief from symptoms. Patients are usually treated with a combination of medication and cognitive behavioural therapy (CBT).

CBT sessions allow patients to acquire coping skills by talking through their problems with a trained therapist. These sessions often include elements of role-play where patients learn to see their behaviour through the eyes of others.

Computerised Cognitive Behavioural Therapy (cCBT) is a form of treatment which provides CBT through technology. These software-based solutions, generally designed to increase mental focus, can be used to support traditional forms of therapy.

Chatbots are computer programs designed to simulate a conversational partner. Users type in questions or statements and the program parses their input to find and return an appropriate response.

As there are currently no solutions available which allow ADHD sufferers to develop and practice their attentiveness skills during simulated conversations, a chatbot-based cCBT solution could be developed to fulfil this role.

1.2 Project aim

The aim of this project, as outlined in the proposal (see Appendix A), is to build an application capable of being used within CBT scenarios to increase ADHD sufferers' levels of attentiveness.

Users will focus upon how an Artificial Conversational Agent (chatbot) talks with them during their use of the application, and signal with a button press whenever the chatbot makes a conversational mistake such as going off topic.

Focussing upon recognising the chatbot's inappropriate responses will help users to develop their attentiveness and learn to avoid using inappropriate responses and behaviours during conversation.

1.3 Project objectives

- To research the techniques and technologies used to treat the symptoms of ADHD.
- To design a software application capable of raising ADHD sufferers' levels of attentiveness.
- To develop a working software application within the time allotted.
- To evaluate the final application.

1.4 Ethics and privacy

This study conforms to BERA ethical research guidelines and was approved by the City of Liverpool College's ethical research panel. (see Appendix B) The chatlog excerpts contained within Appendix S have been anonymised, and all personal information which could be used to identify test participants removed from this document.

1.5 Dissertation structure

This chapter has introduced the problems which this research project will attempt to resolve. Chapter 2 reviews existing literature related to the project. Chapter 3 contains the development methodology. Chapter 4 discusses and analyses the findings of the project. Chapter 5 draws conclusions and provides recommendations for future research.

Chapter 2 - Literature Review

2.1 Introduction

This chapter reviews existing literature related to the project.

2.2 Treating ADHD

There is no consensus agreement as to what causes ADHD, but studies suggest many factors are to blame including genetics, brain injuries, and environmental factors. (National Institute of Health, 2014)

NHS approved ADHD treatments focus primarily upon providing relief from symptoms. In the United Kingdom doctors usually prescribe a combination of medication and therapy. (National Health Service, 2017)

Medications are effective but can cause side effects such as mood swings, headaches, and insomnia. (Cascade, et al., 2010) Patients can also become dependent upon the drugs used to control their symptoms. (Graham, et al., 2011)

Medications offer only temporary relief from symptoms and many find that therapy offers a longer-term solution.

2.3 Use of the therapy-based approach

Cognitive Behavioural Therapy (CBT) sessions can help reduce ADHD symptoms. During these sessions, which often include elements of role-play, patients talk through their issues with a therapist to acquire coping skills.

A 2014 study (Knouse, 2014) examined how skills-based treatments are given to adults with ADHD. Knouse presents motivation as a key factor to acquiring new skills. Patients must see the value of coping skills in practice and implement through doing rather than understanding them purely theoretically.

A related study (Knouse & Safren, 2010) found evidence that skills-based training provides ADHD sufferers with greater symptom relief than therapy alone. Skill exercises performed at home improved participants' abilities to regulate behaviour and increased their capacity to cope even in the presence of negative emotional influences.

2.4 Mindfulness training as an adjunct to therapy

Mindfulness is a practice, often based upon Buddhist teachings, which trains people to remain consciously aware of who they are, where they are, and what they are doing by observing what is going on around them and inside their minds.

Mindfulness exercises commonly focus upon observing breath patterns and paying non-judgemental attention to thoughts and emotions. (Moore, 2017) These meditational practices are thought to have a strong effect in developing the basic attentional processes. (Wu, et al., 2013)

The Clinical Handbook of Mindfulness (Didonna, 2009) examines the subject in greater depth. It describes how the formal practices which develop mindfulness use narrow foci as an anchor to stop the mind from drifting.

Firstly, the subject brings attention to the focus of the exercise, normally a form of sensory input such as breathing. If attention wanders, the subject notes that a distraction has occurred and lets it go. Focus is then placed back upon the attentional anchor.

An article in the Futurist journal (Docksai, 2013) reports that Mindfulness Training helps ADHD sufferers cope with mind-wandering. Docksai also found that mindfulness can help students relax during tests and develop increased sensitivity to the needs of those around them.

Another study (Mrazek, et al., 2012) found that daily participation in mindfulness exercises can reduce mind-wandering. During a two-week program subjects were taught to focus upon their breathing. Tests performed after initial training showed that improvements to concentration developed in subjects during the program could be redirected towards more challenging tasks.

2.5 Keeping patients interested

ADHD sufferers often become bored with repetitive activities. (Kass, et al., 2003) When patients lose interest in treatment programs, this creates an obstacle to affecting positive change.

Researchers at the University of Aarhus (Sondergaard, et al., 2016) found that from 151 people attending an ADHD unit 27% dropped out of treatment before completion and 42% missed 3 or more appointments.

Software applications can provide one solution to the problem. Two studies on computer-assisted instruction (Mautone, et al., 2005) (Botsas & Grouios, 2017) found that in addition to keeping users interested, it can increase sustained attention and improve the school and work performances of ADHD sufferers.

As 90% of United Kingdom households have Internet access (Office for National Statistics, 2017) and thus access to a computer or smart phone, a software-based solution could be deployed to help ADHD sufferers remain within treatment programs.

2.6 Computerised Cognitive Behavioural Therapy

Computerised Cognitive Behavioural Therapy (cCBT) is a method of providing CBT using technology. The majority of cCBT solutions are currently mobile phone applications or websites designed to fit into a patient's schedule.

Applications to increase mental focus can be used with little guidance and produce measurable improvements. One study of a computerised cognitive training program (Smith, et al., 2009) found that it significantly enhanced the working memory and attention spans of a group of participants aged 65 and older.

During training, participants performed six daily brain plasticity exercises. Exercises included elements of pattern matching, sequence recognition and identifying details which had been presented verbally as a short story.

Although some participants exhibited symptoms of fatigue, headaches and frustration related to the exercises, most showed large improvements to their overall memory and information processing skills. Related projects showing similar results include (Barnes, et al., 2010), (Gray, et al., 2012) and (Sahakian, et al., 2015)

CCBT applications may also help people to develop their social skills. Slovák, Gilad-Bahrach and Fitzpatrick (Slovák, et al., 2015) interviewed and conducted design workshops with researchers and developers of Social and Emotional Learning programs for children to identify how curricula could benefit from the inclusion of supportive digital technologies.

Workshop participants highlighted their belief that assistive technologies could be used to help learners develop skills then removed as these skills became engrained into their personalities.

2.7 What are Chatbots?

Chatbots are computer programs designed to simulate a conversational partner. Users converse with the program which parses their input and provides appropriate responses.

Chatbots can provide an alternative approach to therapist led CBT sessions. Researchers at M.I.T. (Hoque, et al., 2013) developed and tested a virtual conversational agent to provide social skills training and improve the performance of job interview candidates.

The MACH (My Automated Conversation Coach) application allows users to participate in mock interviews. It uses an animated interviewer character to ask a series of pre-programmed queries based upon common interview questions.

Test participants took part in three mock interviews with the agent. The application then showed them videos of their performance alongside real-time information of the verbal and non-verbal behaviours which they exhibited in response to cues provided by the virtual agent.

The researchers found this self-reflective feedback had positive effects. Participants who used the MACH application scored much better in follow-up interviews than a control group shown 30 minutes of educational videos.

Part of the reason for MACH's success was due to the interviewer avatar. It was designed to look artificial to overcome feelings of uneasiness which people often experience when interacting with objects that appear almost human.

This phenomenon is referred to as the ‘uncanny valley’ effect. (Mori, 2012) In his essay Mori predicts that deliberately pursuing non-human designs will make it possible to create artificial entities which humans can feel greater levels of affinity towards.

Socially awkward patients may prefer computer-guided therapy as it spares them the embarrassment of face-to-face meetings with health professionals. (Kofmel, 2017) However, related research found that therapy programs with no human contact have higher drop-out rates than clinician guided ones. (Farvolden, et al., 2005) and although people can benefit from applications used alone at home, most will need some form of human guidance to gain the maximum benefit from them. (Marks, et al., 2007)

2.8 Summary

ADHD is a behavioural disorder whose symptoms include impulsiveness and the inability to concentrate. This can interfere with a sufferer’s interpersonal relationships and affect their performance at school or work.

Although medications reduce ADHD symptoms they cannot help sufferers to acquire soft skills. For this reason, General Practitioners often refer patients for therapy as an adjunct to medication. In addition to helping patients develop their social skills, some forms of therapy such as Mindfulness Training can also improve concentration and reduce levels of impulsiveness.

CCBT solutions are used as an adjunct to traditional therapy. They can provide measurable improvements in most ADHD sufferers and have lower drop-out rates than therapist led sessions.

Chatbot-based cCBT applications could provide a cost-effective alternative to traditional CBT. A well-designed chatbot could also help socially awkward people to develop the confidence and basic social skills necessary for them to benefit more from traditional sessions with a CBT professional.

Chapter 3 - Development Methodology

3.1 Introduction

This chapter contains details of how the project was conducted.

3.2 Aim of the Project

The aim of the project is to build an application capable of being used within CBT scenarios to help increase ADHD sufferers' levels of attentiveness.

3.3 Objectives

- Create an application capable of running on the Windows 10 Operating System.
- Design an easy to use interface.
- Application should parse natural language.
- Use an animated avatar to encourage user interaction.
- Include text-to-speech functionality.
- Log conversations and button presses.
- Create maintainable source code.
- Create an easy to install version of the application.

3.4 Project management

Development projects must use some form of management technique to finish on schedule. Two project management methodologies were considered for developing the application, the Waterfall and Agile methods.

The Waterfall Method keeps projects following a rigid sequence of pre-planned stages, only allowing team members to revert to a previous stage when necessary. Each stage is assigned start and end-points known as milestones.

Agile methodology is more flexible than the Waterfall method. It uses iteration to speed up the development of working applications. Projects are broken into small tasks, usually worked on in intensive sprints of between two and four weeks.

During each sprint, part of the project has its requirements defined, developed, and tested. After being evaluated by clients and developers, work is either incorporated into the working version or returned to a backlog of items to complete.

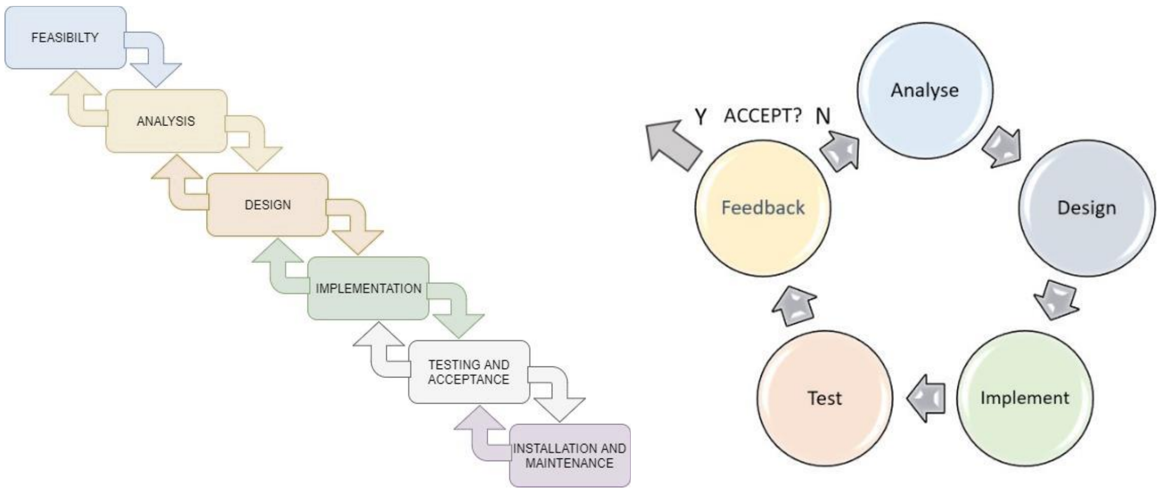


Figure 1: The Waterfall method on the left is more rigidly structured than the Agile method which follows a sequence of sprints.

The Waterfall method was chosen for this project for several reasons. As the project is the work of a single developer there was no need to coordinate the efforts of team members working upon different modules.

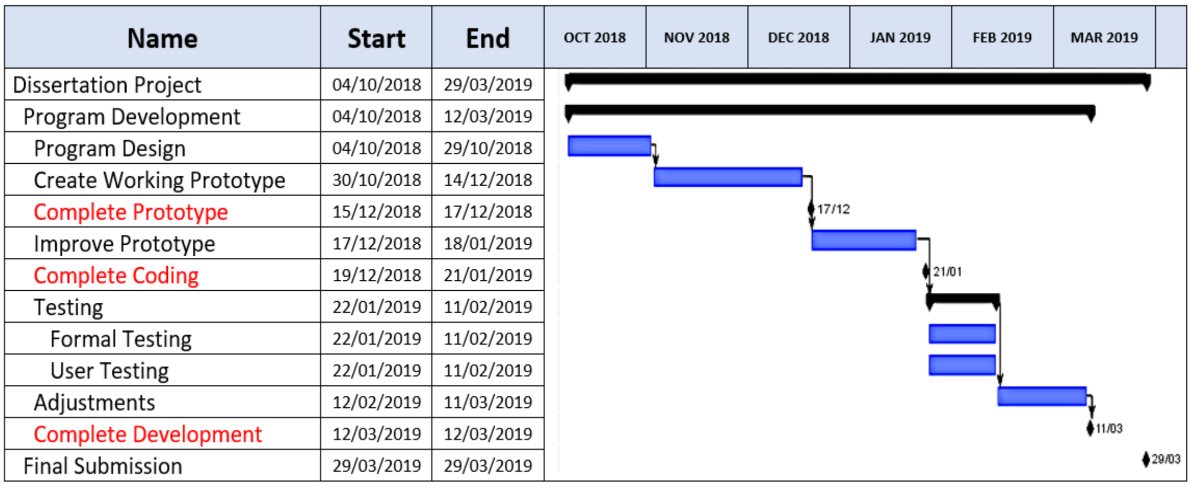
Project requirements were also unlikely to change during development. The third-party libraries used by the application are stable. Each has been rigorously tested by the open source software community for over ten years and is compatible with the current version of the Windows 10 operating system.

Although use of the Agile methodology may have resulted in an application better suited to clients' needs, the decision to follow the structured Waterfall method helped complete the planned application on schedule.

3.5 Program development schedule

Development was divided into five clear stages and a Gantt chart created to provide the developer with a clear visual representation of the project timetable.

Gantt charts are a form of bar chart which provide project leaders with an overview of the tasks involved in bringing a project to completion on schedule. At any time during a project, an examination of the Gantt chart permits actual progress to be compared with expected progress.



Name	Start	End
Dissertation Project	04/10/2018	29/03/2019
Program Development	04/10/2018	12/03/2019
Program Design	04/10/2018	29/10/2018
Create Working Prototype	30/10/2018	14/12/2018
Complete Prototype	15/12/2018	17/12/2018
Improve Prototype	17/12/2018	18/01/2019
Complete Coding	19/12/2018	21/01/2019
Testing	22/01/2019	11/02/2019
Formal Testing	22/01/2019	11/02/2019
User Testing	22/01/2019	11/02/2019
Adjustments	12/02/2019	11/03/2019
Complete Development	12/03/2019	12/03/2019
Final Submission	29/03/2019	29/03/2019

Figure 2: A Gantt chart showing the application development schedule. Project milestones are shown in red.

3.6 Designing the Application

This section looks at how the application was designed.

3.6.1 Usage scenarios

A usage scenario is a short list of the typical steps and actions involved during interaction with an application. Rather than describing how applications are implemented, usage scenarios concentrate upon how they will be used.[1]

Software projects are complex and often get sidetracked by minor issues. This can cause them to run over schedule or fail to implement functionality in a manner acceptable to clients.

A well-drafted usage scenario can communicate an application's needs clearly to its development team.[2] It must contain a set of concisely written goals so that at any stage team members can examine whether their work fulfils project requirements.

Three usage scenarios were identified for the application.

Therapist usage scenario

- Introduce application to an unskilled user explaining its purpose.
- Explain how to run and use the application.
- Give patient a copy of the application.
- Alternatively, patient uses a version of the application installed upon the therapist's computer.

Patient interaction scenario

- Install application.
- Run application.
- Talk to chatbot as instructed by the therapist.

[1] For an in depth look at how usage scenarios fit into the software development lifecycle read 'Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle' (Alexander & Maiden, 2004)
[2] Although usage cases are more suited to the Agile and Extreme Programming methodologies where client/developer interaction plays a key role (Shore, 2007), they can help to keep single developers without a team on track.

- Whenever chatbot makes a conversational mistake, press a button to signal that it was recognised.
- All conversations and button presses are written to a log file.
- Close the application smoothly.

Post usage analysis scenario

- Patient presents log file to the therapist.
- Therapist discusses chatbot session with the patient to help them make better decisions next time.
- Patient applies lessons learned to their own life.

These short scenarios served as guides during the design and development stages of the project.

3.6.2 Designing the interface

Having identified user requirements, research was conducted for the interface design. A review of existing solutions (see Appendix C) showed that popular chatbot interfaces contain three major elements:

a user input box

a chatbot response box

an image or avatar

With this knowledge in hand, design of the interface commenced. The design process began by creating pencil sketches. These rough designs were then refined using a professional online wireframing tool.[3] (see Appendix D)

Wireframes allow designers to plan the layout and functionality of their interfaces without becoming lost in details. (Grant, 2004)

[3] The wireframing tool available at https://mockflow.com allows for the creation of one free interface design using drag-and-drop tools. This design can be reworked multiple time and exported in the JPEG or PNG image formats.

After the basic layout had been finalised, different colour schemes were tried out on a mockup of the interface. A mixture of dark colours and soft pastel pinks was chosen for the final design. This decision was influenced by the Sandman graphic novel series by British author Neil Gaiman; more specifically the colour scheme used for the Delirium character. (see Appendix E)

Delirium exhibits ADHD symptoms taken to the extreme. She is a chaotic and highly unfocussed character whose '*hair and clothes shift from minute to minute with the only things being relatively consistent [being] her multi-coloured hair and eyes*.' (The Sandman Wiki, 2018)

Next, the chatbot avatar was designed. Avatars can reduce the barriers and preconceptions between user and application which arise in text-only interfaces. (Nowak & Rauh, 2005) Whilst some chatbot applications attempt to engage users by emulating texting applications, those that create a deeper sense of interpersonal closeness tend to include an avatar. (Kowatch, et al., 2018)

Keeping Mori's discussion of the uncanny valley effect in mind (Mori, 2012), it was decided that the avatar should be created in an iconic style to help reduce the uneasiness which first-time chatbot users sometimes feel.

As artificial entities with a humanlike appearance are attributed a level of experience which can escalate these feelings of uneasiness (Gray & Wegner, 2012), the avatar design includes elements to convince users that it is naïve and less experienced than them.

After much experimentation, it was concluded that a young female character with feline characteristics would be suitable for the avatar. Animal characteristics have been found to put people at ease and alter their tendency to focus upon negative aspects of their character. (Walsh, 2009)

A cat girl avatar was sketched with pencil and ink onto paper, then scanned and redrawn using the GIMP open-source image editor[4]. (see Appendix F) This image would serve as the model for the animation designs.

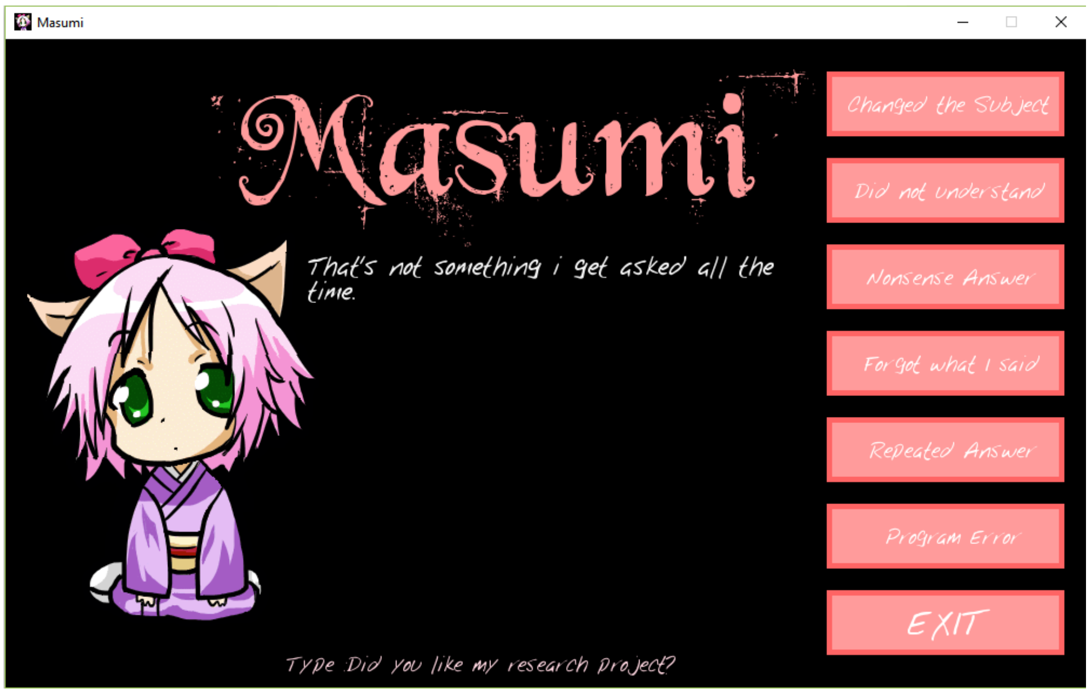
[4] Available for download at the time of publication from https://www.gimp.org/downloads

The first animation created for the avatar was a basic blink to prompt user interaction. (see Appendix G) Blinking reinforces the idea that conversation is a two-way activity, with short blinks drawing longer answers from the conversational partner. (Homke, et al., 2018)

A head movement animation was added to reassure users that the chatbot pays attention (see Appendix H) and nine transparent PNG[5] images of mouth shapes were created for lip synching with the text-to-speech conversion of the chatbot's replies. (see Appendix I).

Lip-synching is the process where a character's mouth movements are matched to the phonemes of a speech track. (Sanders, 2018) Sanders identifies nine basic phoneme[6] shapes for the English language - A, E, FV, LN, MBP, O, TS, UQ and WR. A simple line was drawn onto the chatbot avatar to represent the mouth at rest.

The final part of designing the interface was choosing text fonts. Notepad font is used for text boxes and buttons, and Beyond Wonderland for the application's logo. (see Appendix J) Decorative fonts such as these commonly decorate book covers (Farley, 2009), and fit into the interface's comic book aesthetic. With the graphical user interface fully designed development could now begin in earnest.



That's not something i get asked all the time.

Type Did you like my research project?

Figure 3: The final design of the interface.

[5] PNG is an acronym for the Portable Network Graphics file format.

[6] The Oxford English Dictionary defines a phoneme as;

'Any of the perceptually distinct units of sound in a specified language that distinguish one word from another, for example p, b, d, and t in the English words pad, pat, bad, and bat.'

3.7 Developing the Application

The application is built upon four major elements;

The Python Programming Language.

The PyGame multimedia library.

Artificial Intelligence Markup Language.

The PyAIML library.

This section looks at these elements and details development of the application.

3.7.1 Python

Python is an interpreted high-level language popular amongst the scientific community. Part of the reason for this popularity is due to the large number of data analysis and machine learning libraries freely available from the Python Package Index repository.

Well-written Python code is clear, concise and easy to maintain. It allows developers to use an Object-Oriented Programming (OOP) paradigm for creating interactive user interfaces.

OOP simplifies keeping track of the states of different objects, and the Masumi application needs to run many processes simultaneously including animations, sound, interaction with the chatbot’s kernel, and event handling for mouse and key presses.

3.7.2 PyGame

Although Python runs slower than compiled languages such as C or C++, it is easy to extend with compiled extensions written in different programming languages. The PyGame library is one such extension that allows developers to incorporate sound and graphics routines into their applications.

PyGame provides graphic applications with fast screen updates by allowing them to make calls to optimised C and Assembly code that runs over 100 times faster than pure Python.

PyGame runs on all major platforms and can be used with graphic renderers including DirectX, X11 and OpenGL. Aside from handling graphics, it contains methods for detecting object collisions and playing sound effects.

PyGame class methods are well-documented with plenty of good usage examples available online. This excellent library simplifies the use of graphics, audio and event handling so developers can concentrate upon application logic.

3.7.3 AIML

Chatbots process natural language to formulate and return a response. ELIZA, one of the first programs to attempt this simulates a mock Rogerian psychotherapist. (Weisenbaum, 1966)

ELIZA tokenises user input and calculates the best response based upon keyword distribution. As the program has no real-world knowledge or sense of context, it cannot answer questions or remember what it has been told previously.

Artificial Intelligence Markup Language (AIML) is an extensible markup language (XML) which improves upon the ELIZA concept by allowing programs to import knowledge trees. (Wallace, 2003) (see Appendix K) These knowledge trees allow AIML to store responses to an unlimited amount of queries, and the specification enables programs to keep track of what is being said by users.

The free availability of the open source AIML library provided by the A.L.I.C.E Foundation[7] was a key factor in choosing an AIML solution over alternatives such as RiveScript or ChatScript. Whilst these alternatives can produce chatbot personalities closer to that of an actual ADHD sufferer, they require these personalities to be created from scratch. This is an option which could be considered for future versions.

3.7.4 PyAIML

AIML is based upon the stimulus-response model. Input (stimulus) provided by the user is matched with a pattern stored in a knowledge tree and a response is returned. (Mikic, et al., 2008) AIML interpreters provide an interface between AIML files and the applications using them.

Whilst it is possible to create an AIML interpreter from scratch, many open-source interpreters are available in languages including Java, C++, Python and PHP. Using a third-

[7] Available online at the time of writing at https://code.google.com/archive/p/aiml-en-us-foundation-alice/

party interpreter reduced development time at the cost of losing the extra features which could have been added to a bespoke implementation.

As the Masumi application uses Python, the PyAIML interpreter was chosen. PyAIML is an open-source AIML interpreter implemented in Python. It contains all the features necessary to build and interact with a kernel created from AIML files.

The version of PyAIML used is a fork of the original, translated by developer Konstantin Weddige from Python version 2 to Python 3.[8] Although PyAIML is not supplied with extensive documentation, the library is well-commented and simple to use. A chatbot kernel can be implemented in 6 lines of code;

```
# Import the PyAIML library
import pyaiml

# Create the PyAIML kernel and load the aiml files
kernel = pyaiml.Kernel()
kernel.bootstrap (learnFiles="alice-startup.xml")
kernel.respond("load aiml b")

# Respond to user input
while True:
   print( kernel.respond( input("Enter your message >> ") ) )
```

[8] Available at the time of writing from https://github.com/weddige/pyaiml3

3.8 Setting up the development environment

To begin setting up the development environment, the latest version of Python 3 was downloaded and installed upon a desktop running the Windows 10 operating system.

Next the PyCharm Integrated Development Environment Community Edition was installed[9]. PyCharm Community Edition is free, well-maintained, and easy to use. It has many features which proved useful during development, for example, the intelligent refactoring functions which allow variable names to be updated application-wide.

Finally, PyGame was installed via PyCharm's package manager, and the PyAIML library downloaded from its developer's GitHub page[10]. Work could now start upon developing the application.

3.9 Development method

Development was to follow an object-oriented approach. Object-oriented programming is a paradigm modelled after the way objects interact with each other in the real world. Each object is a self-contained entity made from data which describes its attributes, and methods which allow it to perform operations.

Within an Object-Oriented application, objects can communicate with one another or remain unaware of one another's existence. Hiding objects' inner workings make them simpler to incorporate, and a stub can be put in place during development to simulate their functionality so work can start upon other classes.

This modular approach allows application components to be split up amongst a team for faster development. For this project, it allowed work upon the interface to commence whilst research was still being performed upon how to implement text-to-speech functionality and the PyAIML kernel.

After examining project objectives and looking at the interface storyboard, it was decided to handle the application's functionality using 4 classes.

[9] Available for download at the time of writing from https://www.jetbrains.com/pycharm/download
[10] https://github.com/weddige/pyaiml3

- The MasumiInterface class – used to handle the main event loop, interface with the PyGame library, and handle conversation logging.
- The MasumiBrain class – used to create the PyAIML kernel, load and save the brain file, and get responses to user input from the kernel.
- The MasumiVoice class – used to handle the text-to-speech functionality by creating a Speech Application Programming Interface (SAPI) voice object, streaming text converted to speech to a WAV file, and playing it with PyGame's audio library.
- The Button class – this would need to be created as PyGame does not contain its own button widget class.

Reference tables were created as methods and instance variables were added to the classes. (see Appendix L) A folder structure was imposed to help manage the many source files and images used by the application. (see Appendix M)

As work progressed, difficulties were encountered with text positioning. It was decided to include a third-party function to resize text and place it onto the interface within a PyGame surface object. (see Appendix N)

All modules were tested and debugged before being incorporated into the working version of the application. The modular approach paid off, creating a finished application which is easy to maintain and extend.

Explanations of how the application's main functionality was implemented can be found in Appendix O.

3.10 Building a distributable version

After the final version of the application was ready a distributable setup file was created for the Windows 10 operating system.

The first issue to resolve was the fact that Windows 10 has no native support for Python and although installing it is simple, non-technical people can be put off by the process.

PyInstaller[11] is an application which allows developers to create Windows, Linux and MacOSX executable files from Python programs. The PyInstaller application starts by analysing the Python script to discover any dependencies required for it to run.

Copies of these files are bundled with the script and a copy of the Python interpreter into a distributable folder. An executable file is then created which allows the application to be run with a single click.

With Python added to the Windows *PATH* environmental variable, PyInstaller was run from a command-line shell opened in the same directory as the Masumi script. Once a distribution folder had been created, directories containing the avatar images, fonts and a prebuilt brain file were added.

The Inno Setup[12] application could then be run to package the distribution files into a simple installation program. This executable file installs the Masumi application files, creates a directory structure, adds a program entry into the Start Menu, and creates a Desktop shortcut.

The installation application was then burned onto a compact disc along with a copy of the Masumi application source code, the GPL 3 licence agreement, and a simple PDF installation instructions file. The application was now ready to be tested and, if necessary, revised.

[11] Available to download for free at the time of writing from https://pypi.org/project/PyInstaller/2.0/
[12] Available to download for free at the time of writing from http://www.jrsoftware.org

3.11 Testing

This section looks at tests performed upon the finished application. Testing was divided into a series of two phases to provide quantitative and qualitative feedback upon whether the application had met its objectives.

3.11.1 Formal testing

Before commencing development, a suite of test cases was drafted to ensure the application would meet its requirements. (see Appendix P) The test plan includes visual inspections, input and output checks, and functionality tests.

Test cases are generic enough to be reused if any third-party libraries are replaced or structural changes made to the application.

Formal testing was carried out upon a working version of the application. Results can be found within the test log in Appendix Q.

As each class and function was vigorously tested before it was added to the working version of the application, the final version managed to pass all tests without error.

3.11.2 End user testing

A small group of beta testers were recruited and taught to use the application. After using the application for ten minutes, participants provided the researcher with subjective feedback about their experience.

3.11.3 Test documentation design

Prior to recruiting participants for user testing, a participant Information sheet, consent form and application usability questionnaire were designed. (see Appendix R)

The participant information sheet outlines the nature of the project, the role of participants including their rights and obligations, and details about how the feedback provided would be used.

The participant consent form was drafted to obtain written informed consent from the participant that, after reading the Information sheet, they were aware of what their role in the project would entail and were willing to take part.

Design of these two documents was based upon online guidance provided to students by Cambridge School of Biological Sciences (University of Cambridge, 2018)

The application usability questionnaire asks participants to rate on a 5-point Likert scale (in which 1 represents '*Strongly Disagree*' and 5 '*Strongly Agree*') how much they agree with 18 statements related to their use of the application.

After rating the statements, participants can provide subjective feedback upon how the application could be improved, or comments related to the testing experience.

3.11.4 Participants

A potential problem with recruiting end user testers was identified during the requirements analysis phase. Receiving ethical clearance to test the application upon actual ADHD sufferers would have proved difficult because the majority are below the age of 18.

Ethical approval was granted after the proposed testing group was changed to adults from within the college who would be instructed to play the roles of ADHD sufferers

All test participants were college-educated males above the age of 18 from a diverse range of economic and cultural backgrounds.

3.11.5 The user testing procedure

Candidates for end user testing were approached and provided with a participant information sheet. After reading the document and having any questions answered by the researcher, those that wished to take part completed a consent form.

From those approached, only three agreed to participate in the study. Candidates that refused stated that they were too busy to take part due to work or study commitments.

Participants were first instructed how to use the application, then held a 10-minute conversation with the chatbot whilst playing the role of an ADHD sufferer. After interacting with the application, participants completed a questionnaire about their experience.

3.11.6 Results

Responses to the questionnaire revealed participants had rated each aspect of the user experience as overwhelmingly positive. This implies that the application has been implemented to a high standard, the interface is well-structured, and that the objective of producing a product which could become a positive influence upon users' lives has been met.

Although participants did not rate any aspect of their experience negatively, they had a few concerns about the interface design which must be addressed.

The choice of fonts was a major concern for most respondents who reported that it made text difficult to read. The design of the input text box was also a major concern for one participant who suggested that it should be made to stand out more from the background.

The colour scheme presented problems for one user who noted that people with colour blindness may experience difficulties reading white text on a black background.

Overall, participants enjoyed testing the application and provided positive feedback about their experience. This shows that aside from needing minor interface changes, the application is fit for purpose.

As testing progressed, participants became more skilled at observing chatbot responses from an outside perspective and signalling conversational errors. This suggests that during testing they had exhibited signs of increased attentiveness.

3.12 Evaluation

This section examines whether the initial aims and objectives of the application were met and provides suggestions for improvements.

3.12.1 Aims

The aim of this project was to build an application capable of being used within CBT scenarios to increase ADHD sufferers' levels of attentiveness.

Below is a table which breaks up this aim into its three constituent parts, justifying that each part was met by the final application.

Aim	Justification
… build an application	The application was designed, developed and tested on schedule. It contains all functionality expected from it.
… capable of being used within CBT scenarios …	A single executable file was created for the application which allows it to be installed by a therapist, or upon the ADHD sufferer's own machine.
… increase ADHD sufferers' levels of attentiveness.	The method of using buttons to register when the chatbot makes a conversational mistake can increase levels of attentiveness. Participants paid increasing levels of attention to looking for flaws whilst using the application than during normal conversation.

Table 1: How project aims were met

3.12.2 Objectives

Below is a table containing a list of the objectives for the application and a justification that they were met by the final product.

Objective	Justification
Create an application capable of running on the Windows 10 Operating System.	The application was developed and tested upon three different computers running the Windows 10 operating system. No compatibility issues were encountered.
Design an easy to use interface.	The interface uses elements common to web browsers and chat applications so should feel familiar to most users. Each element serves a specific purpose without obscuring other elements. Buttons have mouseover and mouse click animations to show that they are interactive.
Application should parse natural language.	The use of the PyAIML and the A.L.I.C.E AIML libraries allow natural language to be parsed and return meaningful replies.
Use an animated avatar to encourage user interaction.	Testers found the animated avatar amusing and held conversations without feeling self-conscious. Although the recommended time for interaction with the chatbot was five minutes, most users conversed without pause for over ten minutes.
Include text-to-speech functionality.	Text-to-speech has been implemented using a Windows SAPI object and the PyGame audio library. Words are converted into speech and spoken clearly by an English voice provided with the Windows 10 operating system.

Log conversations and button presses.	All conversations and button presses are written to a log file stored upon the user's computer.
Create maintainable source code.	Source code follows the conventions outlined in the PEP 8 Style guide. (van Rossum & Warsaw, 2013) It is well commented and divided into a collection of manageable classes and methods. Files are stored within a well-named directory structure.
Create an easy to install version of the application.	An installer was created using Inno Setup and tested on three Windows 10 machines without Python 3 added. The application installed smoothly and ran without problems on all machines.

Table 2: How project objectives were met

3.12.3 Suggested improvements

This section suggests improvements which could be made to the application.

Minor changes to the current version

An introduction screen should be displayed when the application is started up to assure users that it is running. At present there can be a delay of up to 30 seconds, depending upon the specification of the host computer. This delay is caused by the need to load the brain file into the PyAIML kernel.

During testing, several participants found using the text input box difficult. They complained that it was not instantly identifiable, and that the font used was difficult to read.

This issue can be resolved by adding a flashing cursor at the point of text insertion, changing the text box background to a lighter colour, darkening the text colour, and using a more conventional font such as Arial or Calibri. The font on the feedback buttons will also need to be changed.

Participants had trouble understanding the meaning of certain texts used on the buttons. Clearer texts should be drafted with the help of a CBT therapist.

Addition of a menu bar

Adding a menu bar will allow users to customise the application to suit their needs. Basic functionality should include closing the application, viewing help files, and reading information about the project and its development.

Users should be able to toggle the logging functionality on and off from the menu bar and choose where the logfiles are saved. Although providing therapists with a logfile is seen as an integral part of the therapeutic use of the application, users may want to talk with the chatbot outside of this context to improve their general conversational skills.

Another feature which users should be able to toggle on and off is text-to-speech. It should also be possible for users to change the pitch or speed of the voice from the menu bar, but this would also require changes to be made to the lip synchronisation timings for the mouth animations.

New chatbot personalities

At present there is only one chatbot personality available for use with the application. This is based upon the default A.L.I.C.E AIML library. These files are however fully customisable, allowing responses to questions and the ways it parses natural language to be altered to create a new personality.

New chatbot personalities would require animations, mannerisms, and SAPI voice tags to be created to differentiate them from the current chatbot. Implementation of this feature would be more suited to a development team than a single developer.

If this feature is added, users should be able to select which personality they wish to speak to via an introduction screen, or the proposed new menu bar.

Different platforms

Versions of the application should be made for the MacOS, Linux and Android operating systems. Although most of the current libraries used by the application are compatible with these platforms, difficulties will be created by its dependence upon the Microsoft SAPI text-to-speech libraries.

Individual versions could be created for each platform using native TTS libraries or by including a third-party library. Alternatively, a wrapper library which provides cross-platform access to TTS across different platforms such as pyttsx[13] could be used.

Although the current version of the application will be fit for purpose after a few minor modifications, it still requires extensive user tests involving input from diagnosed ADHD sufferers and their therapists before a marketable version can be produced.

[13] Available at the time of writing from https://github.com/RapidWareTech/pyttsx

Chapter 4 - Discussion and Analysis of Findings

4.1 Introduction

This chapter discusses and analyses the key findings of the project.

4.2 Key findings

This paper originally proposed that focussing attention upon finding a chatbot's inappropriate responses could help users to develop their attentiveness.

The results of user testing confirm this hypothesis to a certain degree, but stress that more extensive testing is required before the application is suitable for open deployment.

The key findings of the project were;

- Levels of attentiveness increased in participants during user testing.
- Participants did not become bored during user testing and were comfortable conversing with the chatbot.
- Without the guidance of a trained therapist, participants may become distracted from the purpose of the exercise.

4.3 Increased levels of attentiveness

Chatlogs created during user testing (see Appendix S) indicate that participants' levels of attentiveness increased during their use of the application.

For example, in excerpt 1 the participant is clearly focussed upon the task given. He or she notices the chatbot's first inability to understand what was said immediately, and signals this by pressing the appropriate button.

Later in excerpt 1 the participant notices that the chatbot is using a generic response to a certain type of question. This observation is tested, then signalled by using the correct button press.

The participant was able to remain focussed upon the purpose of their interaction with the chatbot. This concurs with research findings discussed in the literature review that software-based solutions can increase sustained attention. (Mautone, et al., 2005) (Botsas & Grouios, 2017)

4.4 A sympathetic avatar

Although instructed to converse with the chatbot for five minutes, participants wanted to use the application for much longer. The interface design and chatbot's personality kept participants interested throughout the test.

During conversations with the chatbot there were no long pauses whilst participants searched for the correct words to use. Knowing that their conversational partner was a computer program allowed testers to openly question what was being said. For example, in excerpt 2 the participant examines the chatbot's logic in a fast exchange until the program makes the mistake of changing the subject.

The choice of avatar design played a major role in keeping participants interested and interactive during their use of the application. Using a young female character with cat traits helped participants overcome the inhibitions sometimes felt when using a chatbot for the first time.

This concurs with the research findings that animals can alter people's tendency to focus upon their negative aspects discussed in the literature review, and Mori's prediction that by deliberately pursuing non-human designs it would be possible to create artificial entities which humans could feel greater levels of affinity towards.

4.5 The need for a trained therapist

Whilst participants displayed increased levels of attentiveness during testing, on occasion they would misunderstand the chatbot's replies.

For example;

User: how long have you been running for

Bot: I was activated in 1995.

User: what do you mean activated

Bot: Human, sorry that was an obscure remark.

BUTTON PRESS: Nonsense Answer

In the exchange above, the chatbot was trying to apologise for saying something that the user could not understand. However, the participant signalled this as a '*Nonsense Answer*.'

Without a trained therapist examining the session with the participant, this type of mistake could reinforce bad conversational habits instead of correcting them.

During a post-session evaluation, a therapist could discuss why the user had perceived this to be nonsense and point out that sometimes people's choice of language makes their responses difficult for others to understand.

Excerpt 3 highlights another potential issue. During this exchange the test participant has forgotten the reason they are using the application and is making fun of the chatbot's replies rather than signalling the obvious mistakes it makes. For example, the chatbot has clearly changed the subject several times.

Another possible issue raised during this exchange is the aggressive tone of voice being used by the participant. A therapist could discuss the fact that this is generally perceived to be an unacceptable way of talking to casual acquaintances, although one may talk to close friends in this manner.

4.6 Summary

Test participants showed increased levels of attentiveness whilst using the application, and excerpts from the log files indicate that it can provide interesting material for therapists to discuss during follow-up sessions.

Testers chatted easily with the animated avatar, and these conversations were uninhibited and verbose. The avatar's animated mannerisms and quirky responses put participants at ease and kept them interested during testing.

The findings from these preliminary tests support the developer's assertion that using the application could help ADHD sufferers to increase their attentiveness.

Log file excerpts also show that use of the application can reveal aspects of a user's personality which may not appear within the formal atmosphere of the CBT therapy session.

The application was found to be fit for purpose. It can now be developed as part of a therapeutic approach which enables patients to examine and correct their interpersonal exchanges within a safe environment.

Chapter 5 - Conclusions and Recommendations

This paper has documented the development and testing of a new approach to increasing attentiveness in ADHD sufferers. Test results indicate that attentiveness increases during use of the application and that it could be used within CBT scenarios involving a professional therapist.

Masumi is designed to complement rather than replace traditional forms of CBT. It is intended to be used outside of formal therapeutic environment and will allow patients to develop their conversational focus at home.

One of the main benefits which the application will provide users is its portability. It can be used at any time of day when the patient is ready rather than having to depend upon the schedules of other people.

ADHD sufferers can use the application for as little or as long as they want, it will always be available to them. Examination of the logged conversations between chatbot and patient created during short five to ten-minute sessions will provide therapists with material for evaluating their patient’s progress.

5.1 Limitations of the study

The major limitation of this study was its lack of access to actual ADHD sufferers. This was due to difficulties gaining ethical approval. The researcher needed to compensate by gathering information from secondary sources, and by instructing test participants to play the role of an ADHD sufferer.

Whilst participants played their roles convincingly enough, a lack of experience with ADHD may have led them to portray its symptoms in a stereotypical rather than informed manner.

The number of participants who agreed to take part in testing was very disappointing and from a very narrow section of society. There were no female participants and the age range of the group was well above that of typical ADHD sufferers due to limitations imposed upon the study by the college’s ethical research panel.

Another issue was the lack of collaboration from a professional CBT therapist. As therapists will be the main consumers of data generated by the Masumi application, their input would have proved invaluable.

5.2 Recommendations for future research

Whilst this project has shown the potential benefits which the approach could provide to ADHD sufferers, further research must be performed before the application is ready for use within real-life scenarios.

Future research needs to include long-term studies, and a larger group of test subjects should be recruited from a wider demographic range to judge whether the application is suitable for all users.

Test subjects must be screened in advance to identify any potential mental health issues that may arise due to the unsupervised aspect of their interaction with the chatbot. Also, as during personality development hidden mental issues may come to the surface, 24-hour support must be made available to subjects and their immediate family.

Research should be performed into the efficacy of alternative interface designs, and the effects of different chatbot avatars and personalities upon user interaction. Implementations for different platforms and technologies should also be examined.

ADHD affects people from all walks of life. Its effect upon sufferers can be compounded by unfocussed environments such as television or the Internet. Whilst medications can provide a short-term solution to symptoms such as anxiety and hyperactivity, the concentration problems caused by ADHD can stop sufferers from achieving their true potential.

It is hoped that the results of this project and the research which follows it will motivate the creation of alternative approaches to the treatment of ADHD symptoms; approaches which will develop the confidence and critical skills of people who were until quite recently wrongly labelled as lazy and disruptive elements.

References

Alexander, I. F. & Maiden, N. (2004) *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle.* 1st ed. Chichester: John Wiley & Sons.

Bagwell, C. L., Newcomb, A. F. & Bukowski, w. M. (1998) Preadolescent friendship and peer rejection as predictors of adult adjustment. *Child Development,* Volume 69, pp. 140-153.

Barnes, D. et al. (2010) *Computer-Based Cognitive Training for Mild Cognitive Impairment: Results from a Pilot Randomized, Controlled Trial.* [Online]
Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2760033/
[Accessed 29 11 2018].

BBC (2018) *ADHD diagnosis for adults 'can take seven years'.* [Online]
Available at: https://www.bbc.co.uk/news/uk-england-44956540
[Accessed 04 10 2018].

Botsas, G. & Grouios, G. (2017) Computer assisted instruction of students with ADHD and academic performance: a brief review of studies conducted between 1993 and 2016, and comments. *European Journal of Special Education Research,* 2(6), pp. 146-179.

British Medical Journal (2015) Computerised cognitive behaviour therapy (cCBT) as treatment for depression in primary care (REEACT trial): large scale pragmatic randomised controlled trial. *British Medical Journal,* Volume 351, pp. 5627-5640.

Cascade, E., Kalali, A. H. & Wigal, S. B. (2010) Real-World Data on Attention Deficit Hyperactivity Disorder Medication Side Effects. *Psychiatry,* 7(4), pp. 13-15.

Daley, D. & Birchwood, D. (2009) *ADHD and academic performance:why does ADHD impact on academic performance and what can be done to support ADHD children in the classroom,* Bangor: Bangor University.

Didonna, F. (2009) *Clinical Handbook of Mindfulness.* 1st ed. Berlin: Springer Science and Business Media.

Docksai, R. (2013) A Mindful Approach to Learning. *Futurist,* 47(5), p. 8.

Farley, J. (2009) *The Decorative Typeface.* [Online]
Available at: https://www.sitepoint.com/the-decorative-typeface/
[Accessed 24 02 2019].

Farvolden, P., Selby, P. & Bagby, R. M. (2005) Usage and longitudinal effectiveness of a web-based self-help CBT behavioural therapy program for panic disorder. *Journal of Medical Internet Research,* 7(7).

Fayyad, J., De Graaf, R., Kessler, R. & Alonso, J. (2007) Cross-national prevalence and correlates of adult attention-deficit hyperactivity disorder. *The British Journal of Psychiatry,* 190(5), pp. 402-409.

Free Software Foundation (2007) *GNU GENERAL PUBLIC LICENSE,* Massachusetts: Free Software Foundation.

Garrod, S. & Pickering, M. J. (2004) Why is conversation so easy?. *Trends in Cognitive Sciences,* 8(1), pp. 8-11.

Graham, J. et al. (2011) European guidelines on managing adverse effects of medication for ADHD. *European Child and Adolescent Psychaitry,* 20(1), pp. 17-37.

Grant, J. J. (2004) *The Elements of User Experience - User-Centered Design for the Web.* 2nd ed. San Francisco: New Rider Publishing.

Gray, K. & Wegner, D. M. (2012) Feeling robots and human zombies: Mind percveption and the uncanny valley. *Cognition,* Issue 125, pp. 125-130.

Gray, S. A. et al. (2012) Effects of a computerized working memory training program on working memory, attention, and academics in adolescents with severe LD and comorbid ADHD: a randomized controlled trial. *Journal of Child Psychology and Psychiatry,* 53(12), pp. 1277-1284.

Hale, T. S. et al. (2005) Impaired linguistic processing and atypical brain laterality in adults with ADHD. *Clinical Neuroscience Research,* 5(5-6), pp. 255-263.

Homke, P., Holler, J. & Levinson, S. C. (2018) Eye blinks are perceived as communicative signals in human face-to-face interaction. *PLOS ONE,* 13(12), pp. 1-13.

Hoque, M. E. et al. (2013) MACH: My Automated Conversation coacH. In: *UbiComp 13New York.* New York: ACM Press, p. 697.

Johnsen, S. K. & Kendrick, J. (2005) *Teaching gifted students with disabilities.* 1st ed. Waco: Prufrock Press.

Kass, S. J., Wallace, J. C. & Vodanovich, S. (2003) Boredom proneness and sleep disorders as predictors of adult attention deficit scores. *Journal of Attention Disorders,* Volume 7, pp. 83-91.

Klein, R. G. & Mannuzza, S. (1991) Long-term outcome of hyperactive children: a review. *Journal of the American Academy of Child and Adolescent Psychiatry,* Volume 30, pp. 383-387.

Knouse, L. E. (2014) Cognitive-Behavioral Therapies for ADHD. In: R. A. Barkley, ed. *Attention-Deficit Hyperactivity Disorder: A Handbook for Diagnosis and Treatment.* New York: Guilford Press, pp. 757-773.

Knouse, L. E. & Safren, S. A. (2010) Current status of cognitive behavioral therapy for adult attention-deficit hyperactivity disorder.. *Psychiatric Clinics of North America,* 33(3), pp. 497-509.

Kofmel, J. J. (2017) *Online Versus In-Person Therapy: Effect of Client Demographics and Personality Characteristics,* Minneapolis: Walden University.

Kowatch, T. et al. (2018) *The impact of interpersonal closeness cues in text-based healthcare chatbots on attachment bond and the desire to continue interacting: an experimental design,* Zurich: Twenty-Sixth European Conference on Information Systems.

Lee, D., Oh, K.-J. & Choi, H.-J. (2017) *The chatbot feels you - a counseling service using emotional response generation.* Jeju, Korea, Korean Institute of Information Scientists and Engineers.

Marks, I. M., Cavanagh, K. & Gega, L. (2007) Computer-aided psychotherapy: revolution or bubble?. *The British Journal of Psychiatry,* Issue 191, pp. 471-473.

Mautone, J., DuPaul, G. J. & Jitendra, A. K. (2005) The Effects of Computer-Assisted Instruction on the Mathematics Performance and Classroom Behavior of Children With ADHD. *Journal of Attention Disorders,* 9(1), pp. 301-312.

Mikic, F. A. et al. (2008) *CHARLIE: An AIML-based Chatterbot which Works as an Interface among INES and Humans,* Vigo: University of Vigo.

Moore, J. A. (2017) *Examination of the Effects of Computer Assisted Mindfulness Strategies with Adolescents in an Alternative High School Setting,* Kalamazoo: Western Michigan University .

Mori, M. (2012) The Uncanny Valley (Translated from the Japanese by Karl. F. MacDorman and Norri Kageki). *IEEE Robotics and Automation Magazine,* Issue June 2012, pp. 98-100.

Mrazek, M. D. et al. (2012) *Mindfulness Training Improves Working Memory Capacity and GRE Performance While Reducing Mind Wandering,* Santa Barbara: University of California.

Mrug, S. et al. (2012) Peer rejection and friendships in children with Attention-Deficit/Hyperactivity Disorder: contributions to long-term outcomes.. *Journal of Abnormal Child Psychology,* 40(6), p. 1013–1026.

National Health Service (2017) *Attention Deficit Hyperactivity Disorder.* [Online]
Available at: https://www.nhs.uk/conditions/attention-deficit-hyperactivity-disorder-adhd/treatment/
[Accessed 22 12 2018].

National Health Service (2018) *Overview - Attention deficit hyperactivity disorder (ADHD).* [Online]
Available at: https://www.nhs.uk/conditions/attention-deficit-hyperactivity-disorder-adhd/
[Accessed 27 12 2018].

National Institute of Health (2014) Causes of ADHD. *NIH Medline Plus,* Volume 9, pp. 15-16.

Nowak, K. L. & Rauh, C. (2005) The Influence of the Avatar on Online Perceptions of Anthropomorphism, Androgyny, Credibility, Homophily, and Attraction. *Journal of Computer-Mediated Communication,* 11(1), pp. 153-178.

Office for National Statistics (2017) *Internet access – households and individuals, Great Britain: 2017.* [Online]
Available at:
https://www.ons.gov.uk/peoplepopulationandcommunity/householdcharacteristics/homeinternetandsocialmediausage/bulletins/internetaccesshouseholdsandindividuals/2017
[Accessed 30 12 2018].

Park, K.-M.et al. (2011) A virtual reality application in role-plays of social skills training for schizophrenia: A randomized, controlled trial. *Psychiatry Research,* 189(2), pp. 166-172.

Polanczyk, G. et al. (2007) The worldwide prevalence of ADHD: a systematic review and metaregression analysis.. *American Journal of Psychiatry,* Issue 164, pp. 942-948.

Rost, T. et al. (2017) User Acceptance of Computerized Cognitive Behavioral Therapy for Depression: Systematic Review. *Journal of Medical Internet Research,* 19(9), p. 309.

Sadek, J. (2018) ADHD and Depression. In: *Clinician's Guide to ADHD Comorbidities in Children and Adolescents.* Cham, Switzerland: Springer, pp. 89-97.

Sahakian, B. et al. (2015) *The impact of neuroscience on society: cognitive enhancement in neuropsychiatric disorders and in healthy people.* [Online]
Available at: http://rstb.royalsocietypublishing.org/content/370/1677/20140214
[Accessed 29 11 2018].

Sanders, A.-L. (2018) *Basic Phonemes and Lip-Synching for Animation.* [Online]
Available at: https://www.lifewire.com/lip-synching-for-animation-basic-phonemes-140558
[Accessed 24 02 2019].

Saygin, A. P., Cicekli, I. & Akman, V. (2000) Turing Test: 50 Years Later. *Minds and Machines,* 10(4), pp. 453-518.

Schuetzler, R. M., Giboney, J. S., Grimes, M. & Nunamaker, J. F. (2018) The influence of conversational agent embodiment and conversational relevance on socially desirable responding. *Decision Support Systems,* Volume 114, pp. 94-102.

Shore, J. (2007) *The Art of Agile Development.* 1st ed. Sebastapol: O'Reilly Media, Inc..

Slovák, P., Gilad-Bahrach, R. & Fitzpatrick, G. (2015) *Designing Social and Emotional Skills Training: The Challenges and Opportunities for Technology Suppor,* Seattle: Microsoft Research.

Smith, G. et al. (2009) *A Cognitive Training Program Based on Principles of Brain Plasticity: Results from the Improvement in Memory with Plasticity-based Adaptive Cognitive Training (IMPACT) Study.* [Online]
Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4169294/
[Accessed 29 11 2018].

Sondergaard, H. M. et al. (2016) Treatment dropout and missed appointments among adults with attention-deficit/hyperactivity disorder: associations with patient- and disorder-related factors.. *Journal of Clinical Psychiatry,* 77(2), pp. 232-239.

Tantam, D. (2003) The challenge of adolescents and adults with asperger syndrome. *Child Adolescence and Psychiatric Clinics of North America,* Volume 12, pp. 142-163.

The Sandman Wiki (2018) *Delirium.* [Online]
Available at: https://sandman.fandom.com/wiki/Delirium
[Accessed 24 02 2019].

University of Cambridge (2018) *Participant information sheets and consent forms.* [Online]
Available at: https://www.bio.cam.ac.uk/psyres/informationsheets
[Accessed 2019 03 01].

van Rossum, G. & Warsaw, B. (2013) *PEP 8 -- Style Guide for Python Code.* [Online]
Available at: https://www.python.org/dev/peps/pep-0008/
[Accessed 2019 03 05].

Wallace, R. (2003) *The Elements of AIML Style.* 1st ed. New York: ALICE A.I Foundation.

Walsh, F. (2009) Human-Animal Bonds I: The Relational Significance of Companion Animals. *Family Process,* 48(4), pp. 462-476.

Weisenbaum, J. (1966) ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM,* 9(1), pp. 36-45.

Wilcox, B. (2019) *ChatScript Documentation.* [Online]
Available at: http://chatscript.sourceforge.net/Documentation/
[Accessed 2019 02 25].

Wu, Q., Shi, L., Xia, Z. & Lu, L. (2013) Effects of Duration and Contents of Mindfulness Training on Depression. *Psychology,* 4(6), pp. 163-8-17.

Appendix A: Dissertation proposal

Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers

by
Andrew Laing

A dissertation proposal presented in part fulfilment of the requirements for the degree of BSc Computing (OU top-up)

at the
City of Liverpool College
October 2018

Contents

1: Outline of the Problem

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioural disorders. It affects about 5% of children (Polanczyk, et al., 2007) and 3% of adults. (Fayyad, et al., 2007)

Recognised symptoms of ADHD include hyperactivity, anxiety, impulsiveness, inattentiveness and difficulty in concentrating. Whilst many people are affected by all these symptoms to a degree, for ADHD sufferers they can strongly interfere with interpersonal relationships and affect performance at school or work.

There is no consensus agreement as to the causes of ADHD, but studies suggest that many causes are to blame including genetics, brain injuries, and social or environmental factors. (National Institute of Health, 2014)

Without a clear cause, ADHD treatments can only focus upon providing relief from symptoms. In the United Kingdom doctors usually proscribe patients a combination of medication and therapy. (National Health Service, 2018)

ADHD medications are effective but can cause side-effects such as mood swings, headaches, decreased appetite and insomnia. Patients can become psychologically dependent upon the drug being used to control symptoms; having traumatic episodes when it becomes unavailable.

Cognitive Behavioural Therapy (CBT) helps people to develop coping strategies for ADHD by talking through problems with a trained therapist. CBT sessions can include elements of role-play to teach acceptable forms of social behaviour and the consequences of interactions with others. Through role-play patients can gain confidence to engage more actively in social situations.

Getting access to treatment can be slow. General Practitioners cannot formally diagnose patients as having ADHD and must refer them to specialists instead. Due to a lack of adequate investment there are long waiting lists for diagnosis. The charity organisation ADHD Action claims that '***some adults wait more than seven years to be diagnosed…***'. (BBC, 2018)

Meanwhile, children and adults living with ADHD try to develop their own coping strategies. Left without professional help, they may develop associated conditions such as depression and personality disorders. (Sadek, 2018)

2: Aims and Objectives

The aim of this project is to build an Artificial Conversational Agent (chatbot) application capable of being used within CBT scenarios to help increase ADHD sufferers' levels of attentiveness. The chatbot's name will be Masumi - a unisex given name from Japan which translates as 'true lucidity'.

Chatbots are computer programs which use natural language processing to simulate human conversation. Humans pass text to the program which provides responses based upon a series of predefined rules.

Although powerful modern chatbots such as Siri, Alexa or Cortana are highly advanced and can even respond convincingly to a user's emotional state, (Lee, et al., 2017) they are still apt to repeat themselves and reply with seemingly flippant or insensitive remarks. Observant users soon notice that chatbots lack empathy, do not understand context, and cannot remember everything said during a conversation.

Chatbots often respond badly to unknown scenarios. Developers try to work around this by programming them to respond to the unknown using stock phrases, dialogue for related subjects, or by changing the subject to a known one. This need to maintain an illusion of humanlike intelligence is often fuelled by the desire to create a chatbot capable of passing the Turing Test. (Saygin, et al., 2000)

The proposed application would hopefully fail a Turing Test. It will draw attention to the imperfections of its rule-based conversational abilities, so users can become aware of any inappropriate and off-topic responses they make during conversation.

Users will be taught during therapist-guided sessions to press one of a series of buttons to signal when the chatbot goes off-topic or changes the subject. Once patients become comfortable with the application, it can be given to them for use at home.

All interactions between user and chatbot will be logged. Post-session log examination by therapists will provide material to discuss during their next session. They could talk about a patient's correct and incorrect signalling of inappropriate chatbot behaviours and educate patients as to the consequences which these behaviours have in the real world.

As conversations between patient and chatbot contain sensitive personal information, chatlogs should be encrypted using a strong Public/Private Key encryption method.

The premise behind the proposed application is simple. Patients will be made aware from the beginning that the chatbot occasionally makes mistakes during conversation. The patient must spot as many of them as possible. After spending time focussing attention upon finding the chatbot's inappropriate responses, patients will notice and correct their own.

3: Literature Review

A preliminary examination shows that whilst most recent studies of Computerised CBT (cCBT) have focussed upon its uses for the relief of anxiety, depression and social phobias there is growing interest in applying it to help ADHD sufferers.

The majority of cCBT solutions are currently either mobile phone applications or websites. These solutions are designed to fit into patients' schedules, allowing them 24-hour access. They are simple adjuncts to therapy and can be used with very little guidance.

Studies show that many patients find these solutions as effective as medication. A British Medical Journal report found one online solution provided to adults, without clinical support, was able to effectively decrease symptoms in adults with moderate to severe depression. (British Medical Journal, 2015)

Solutions examined did not help all study participants and improvements discovered during testing were not found to be superior to those attained using traditional treatments offered by General Practitioners (GPs).

Computerised solutions are often presented as more time and cost-effective than alternatives. Papers taking social considerations into account seem to agree that cCBT solutions could be especially beneficial to those having difficulties receiving a diagnosis and primary care from their GP, or whom cannot afford treatments.

Common objections to cCBT mention problems with users accepting the technology as a serious form of treatment, difficulties interacting with poorly-designed user interfaces, and the potential dangers of patients not receiving timely diagnosis for ADHD related issues including feelings of isolation and depression. (Rost, et al., 2017)

ADHD sufferers process language differently from other people. They struggle to find the right words during conversations, and constantly go off-topic. (Hale, et al., 2005) Developers try to address these problems with 'Brain Training', scheduling and diary applications.

Whilst these applications can improve memory and organisational skills, most developers forget that during conversation people put on a façade to empathise with others. This façade is a learned behaviour which can often override common sense.

Conversational disorders can be crippling and affect all areas of a sufferer's life. Role-play has long been used as a way of enhancing speech and language skills. Conscious participation in role-play can help develop personality and boost a patient's confidence in their speaking abilities. (Park, et al., 2011)

As ADHD sufferers can feel ill-at-ease around unknown people, a conversationally-challenged chatbot could provide an ideal alternative role-play partner.

There is currently a lack of research into applications providing computer-based critical awareness training to ADHD sufferers. Mediated interactions with a chatbot could help sufferers to develop good conversational abilities. This project aims to provide a working solution.

4: Development Strategy

Due to limitations of time and resources, the Waterfall Method was chosen for the development model over the more flexible Agile Method. The developer will follow a sequence of Top-Down Stepwise refinements from design and development through to user testing.

Work will commence by creating user stories for a typical therapist and patient to determine how each can use the application. These stories will serve as a guide for designing the graphical user interface (GUI) and planning features to include.

Next, research will look for inspiration from existing chatbot interfaces, and a GUI mock-up will be created using GIMP or Photoshop.

After the initial design phase is completed, a working prototype will be built using the Python programming language, Open Source third-party libraries, and the PyCharm Integrated Development environment. This prototype will be created using a modular approach with component methods grouped into their own classes.

As work upon the application progresses, stable versions will be saved. The GIT versioning system will be used so changes can be rolled back quickly should errors be found. Each major version of the application will be tested before being pushed to the main repository.

Once the prototype is fully working, participants for user testing will be recruited from within the College. Tests will include a period of ten to fifteen minutes using the application followed by completion of a short Questionnaire about the experience. Each participant must provide written consent.

No prohibitive costs or equipment acquisitions are expected as the developer already has access to all necessary resources; including backup machines should the need arise. Costs for printing consent forms and questionnaires will only amount to several pounds using the College's printing services. The biggest investment required for this project will be development and testing time.

5: Ethics

After discussing ethical issues with the supervisor, it was decided that the only potential issues to consider were the application's use of third party-libraries and the involvement of participants in beta testing.

All third-party libraries used in the project will be Open Source. They are provided under the terms of the GNU General Public Licence (GPL). The GPL grants developers the ability to '***run, modify and propagate'*** covered software so long as the final product '***pass on to the***

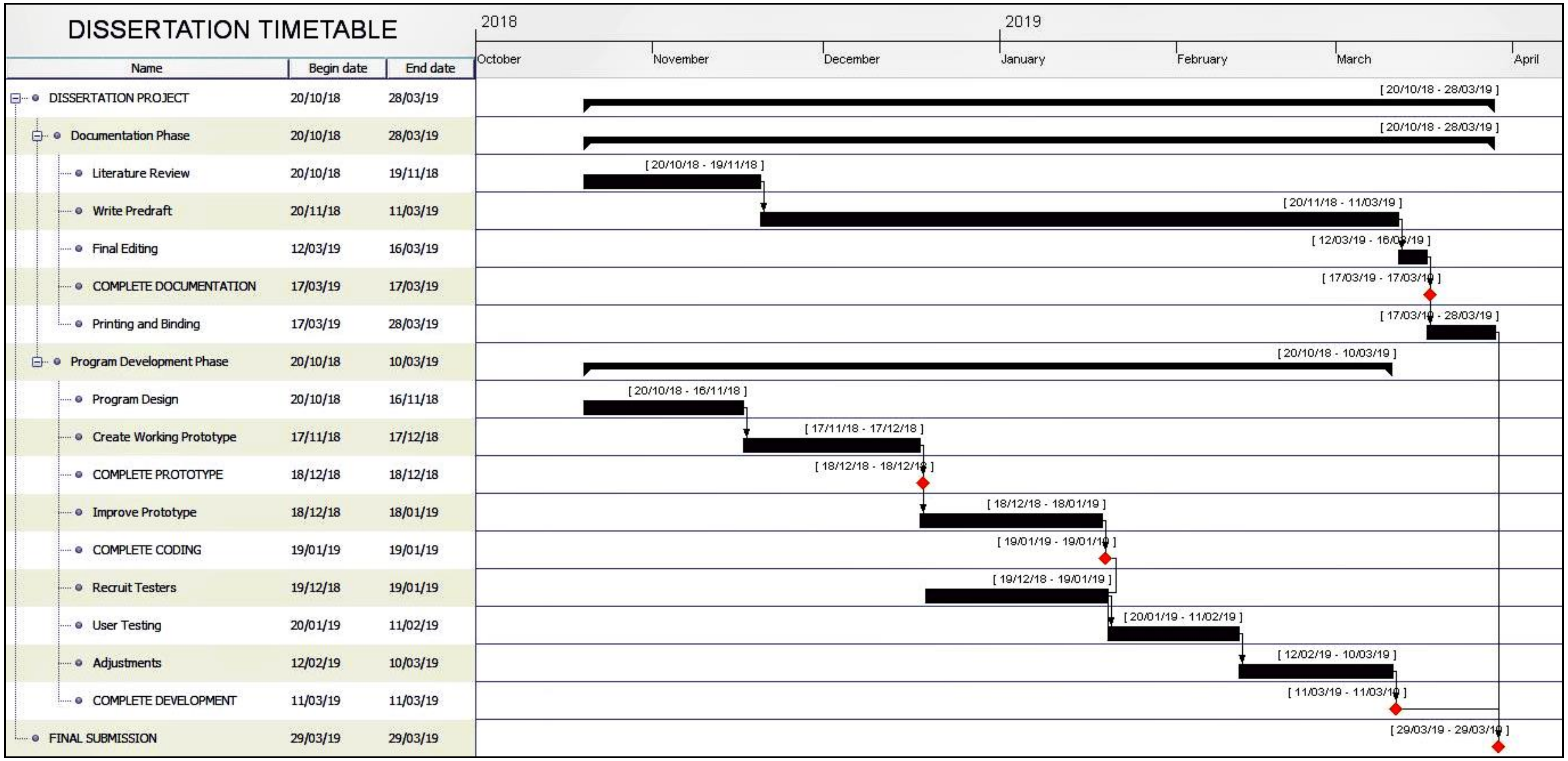
recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code.' (Free Software Foundation, 2007) This means that applications incorporating GPL licenced libraries are bound under the GPL's terms, as a way of respecting authors' rights.

After being clearly told about the application's purpose and the requirements of their role, participants will be required to give written consent before taking part in beta testing. The developer will ensure that participants come to no harm during testing and that their rights to confidentiality and privacy are respected.

All application logs and questionnaires will be anonymised to mask participant identity. Participants will be clearly informed of their right to withdraw from the research at any moment without consequences.

Furthermore, before undertaking development and recruiting test participants the developer will seek approval from the ethics panel upon the methods to be used.

6: Timetable



DISSERTATION TIMETABLE

Name	Begin date	End date
DISSERTATION PROJECT	20/10/18	28/03/19
Documentation Phase	20/10/18	28/03/19
Literature Review	20/10/18	19/11/18
Write Predraft	20/11/18	11/03/19
Final Editing	12/03/19	16/03/19
COMPLETE DOCUMENTATION	17/03/19	17/03/19
Printing and Binding	17/03/19	28/03/19
Program Development Phase	20/10/18	10/03/19
Program Design	20/10/18	16/11/18
Create Working Prototype	17/11/18	17/12/18
COMPLETE PROTOTYPE	18/12/18	18/12/18
Improve Prototype	18/12/18	18/01/19
COMPLETE CODING	19/01/19	19/01/19
Recruit Testers	19/12/18	19/01/19
User Testing	20/01/19	11/02/19
Adjustments	12/02/19	10/03/19
COMPLETE DEVELOPMENT	11/03/19	11/03/19
FINAL SUBMISSION	29/03/19	29/03/19

7: Draft plan of the project

1: **Title Page**.

2: **Abstract** – approximately 200 words.

3: **Acknowledgements** – thanking those who helped during work upon the dissertation.

4: **Table of Contents**.

5: **Table of Figures**.

6: **List of accompanying materials** – (e.g., application on CD.)

7: **List of Abbreviations.**

8: **List of definitions** – words or phrases with different uses within other fields.

9: **Short Introduction** – a couple of pages outlining the project, user testing and results. Short summary of how the project progressed.

Introduction to the Project

Development

User Testing

Summary of Results

10: **Literature review** – An examination of existing literature related to the uses of cCBT and Artificial Conversational Agents; a critical examination of arguments for and against.

Introduction to Literature Review

Statement of the Problem

Significance of Project

Computerised Cognitive Behavioural Therapy

Artificial Conversational Agents

Chatbots used in Therapeutic Situations

11: **Methods** – Details of research conducted for the creation of the application, technologies used, design of user tests, choosing participants, how tests were conducted, and issues related to user participation.

Application Design

Application Development

User Test Design

Testing

12: **Results** – Test results, and details of how well the final application met its specification.

The Finished Application

User Testing Results

13: **Analysis and Discussion** – This section analyses test results, discussing the problems and anomalies encountered, and examines their significance.

Analysis of Test Results

Problems and Anomalies Encountered

Significance of Test Results

14: **Conclusions & recommendations** – This section reiterates the project's key findings, considers beneficiaries, discusses limitations, and makes recommendations for further development which consider test findings and possible access to alternative technologies.

Conclusions

Recommendations for Further Development

15: **References.**

16: **Bibliography.**

17: **Appendices** – Anonymised examples of user interaction, copy of test participation agreement form, licences attached to third-party libraries (e.g., GNU Public Licence 3.0)

18: **Ethical Approval Form** – Copy of ethics form signed by dissertation author and approved by supervisor.

8: Conclusion

Computerised Cognitive Behavioural Therapy is still finding its place alongside traditional forms of therapy and this project is likely to provide developers with a new approach to creating therapeutic applications.

Instead of placing the artificial personality as an expert to please, patients will be presented with a flawed persona whom they may feel that they are helping instead. Patients aware of the artificial nature of their interactions with the chatbot will feel in a position of greater power over these conversations and their ability to change themselves for the better will increase.

The chatlogs produced by the application will allow therapists to gain meaningful insights into the personalities of ADHD sufferers who often have difficulties expressing themselves in face-to-face situations.

Finally, it is suggested that for future research, artificial personalities created with the approach used in this project can be attached to Virtual Reality avatars. These avatars will be used within realistic environments which could allow safe interaction with simulated sufferers of a wide variety of Neurological and Mental disorders.

References

BBC (2018) ADHD diagnosis for adults 'can take seven years'. [Online]
Available at: https://www.bbc.co.uk/news/uk-england-44956540
(Accessed 04 10 2018).

British Medical Journal (2015) Computerised cognitive behaviour therapy (cCBT) as treatment for depression in primary care (REEACT trial): large scale pragmatic randomised controlled trial. British Medical Journal, Volume 351, pp. 5627-5640.

Fayyad, J., De Graaf, R., Kessler, R. & Alonso, J. (2007) Cross-national prevalence and correlates of adult attention-deficit hyperactivity disorder. The British Journal of Psychiatry, 190(5), pp. 402-409.

Free Software Foundation (2007) GNU GENERAL PUBLIC LICENSE, Massachusetts: Free Software Foundation.

Hale, T. S. et al. (2005) Impaired linguistic processing and atypical brain laterality in adults with ADHD. Clinical Neuroscience Research, 5(5-6), pp. 255-263.

Lee, D., Oh, K.-J. & Choi, H.-J. (2017) The chatbot feels you - a counselling service using emotional response generation. Jeju, Korea, Korean Institute of Information Scientists and Engineers.

National Health Service (2018) Attention deficit hyperactivity disorder (ADHD). [Online]
Available at: https://www.nhs.uk/conditions/attention-deficit-hyperactivity-disorder-adhd/treatment/
(Accessed 04 10 2018).

National Institute of Health (2014) Causes of ADHD. NIH Medline Plus, 9(Spring), pp. 15-16.

Park, K.-M. et al. (2011) A virtual reality application in role-plays of social skills training for schizophrenia: A randomized, controlled trial. Psychiatry Research, 189(2), pp. 166-172.

Polanczyk, G. et al. (2007) The worldwide prevalence of ADHD: a systematic review and metaregression analysis. American Journal of Psychiatry, Issue 164, pp. 942-948.

Rost, T. et al. (2017) User Acceptance of Computerized Cognitive Behavioural Therapy for Depression: Systematic Review. Journal of Medical Internet Research, 19(9), p. 309.

Sadek, J. (2018) ADHD and Depression. In: Clinician's Guide to ADHD Comorbidities in Children and Adolescents. Cham, Switzerland: Springer, pp. 89-97.

Saygin, A. P., Cicekli, I. & Akman, V. (2000) Turing Test: 50 Years Later. Minds and Machines, 10(4), pp. 453-518.

Bibliography

Barkley, R. A., (2006) Attention–Deficit Hyperactivity Disorder: A Handbook for Diagnosis and Treatment. 3rd ed. New York: Guilford Press.

Bostrom, N. (2014) Superintelligence - Paths, Dangers, Strategies. 1st ed. Oxford: Oxford University Press.

Braswell, L. & Bloomquist, M. L. (1991) Cognitive-Behavioural Therapy with ADHD Children: Child, Family, and School Interventions. 2nd ed. New York: Guilford Press.

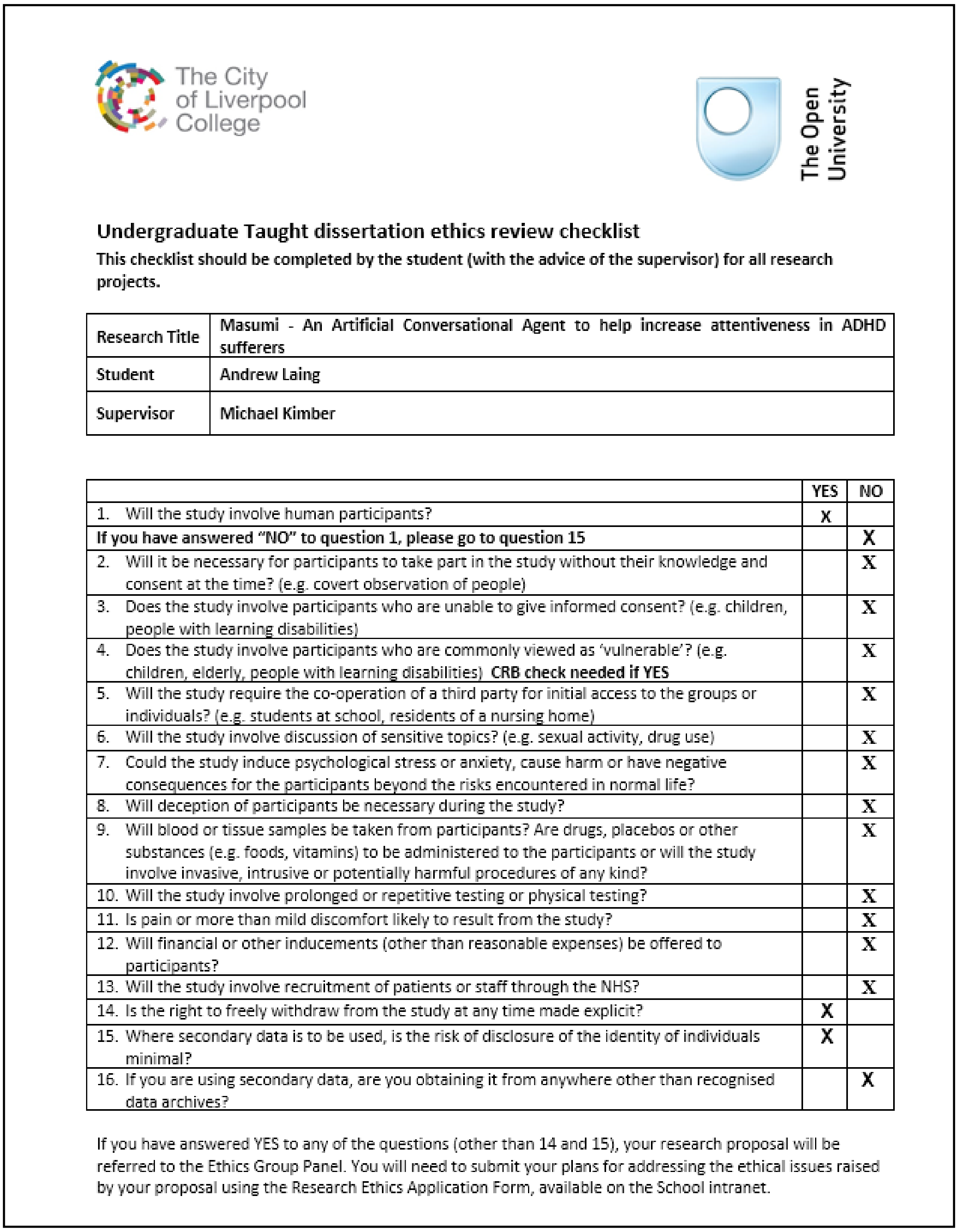
Kurzweil, R. (2005) The Singularity Is Near: When Humans Transcend Biology. 1st ed. New York: Penguin.

Norman, K. L. (2017) Cyberpsychology: An Introduction to Human-Computer Interaction. 2nd ed. Cambridge: University Printing House.

Parsons, T. D. (2017) Cyberpsychology and the Brain. 1st ed. Cambridge: Cambridge University Press.

Perez-Marin, D. & Pascual-Nieto, I. (2011) Conversational Agents and Natural Language Interaction: Techniques and Effective Practices. 1st ed. Hershey: Information Science Reference.

Dissertation Ethics Review Checklist



The City of Liverpool College

The Open University

Undergraduate Taught dissertation ethics review checklist

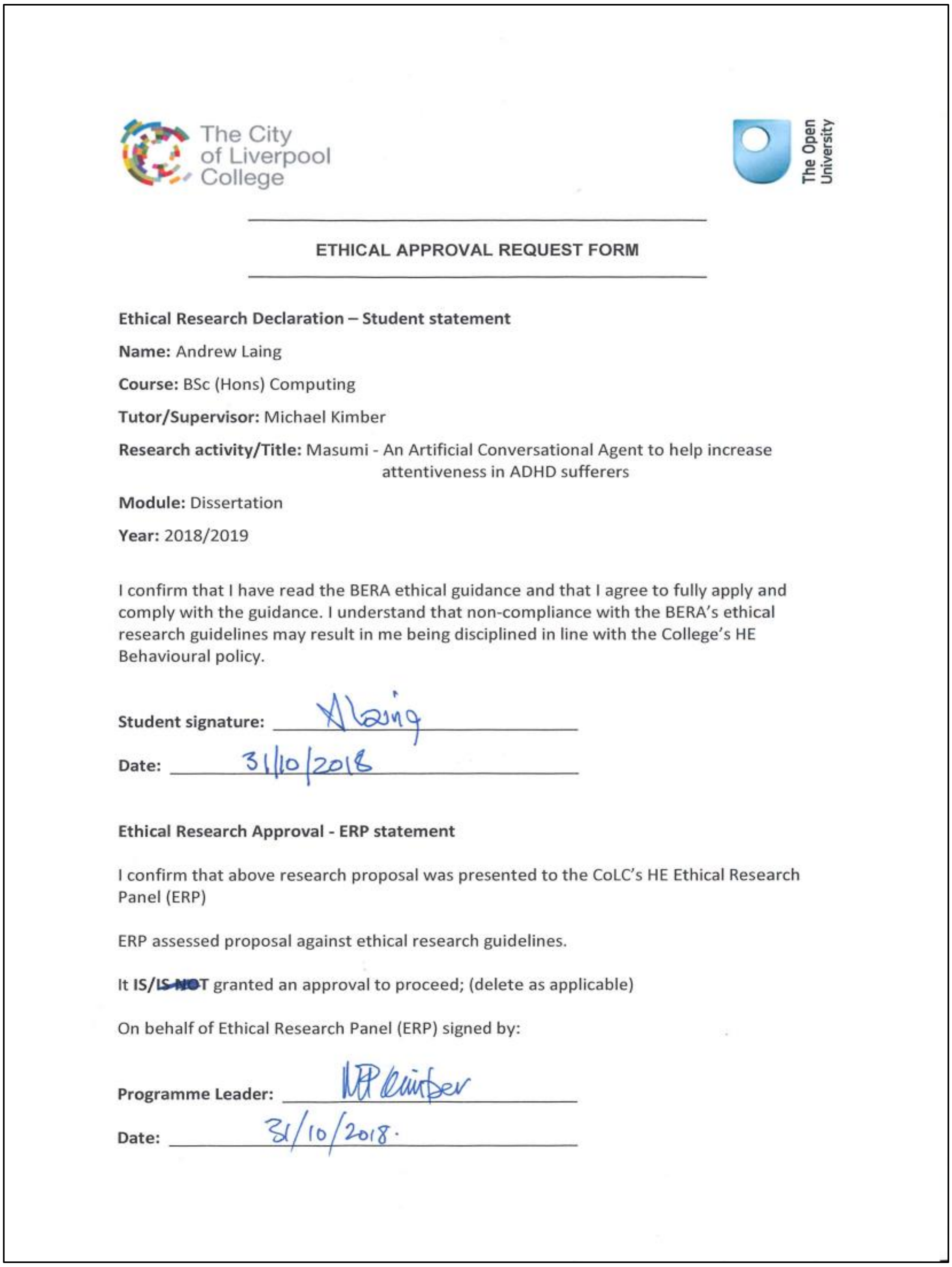
This checklist should be completed by the student (with the advice of the supervisor) for all research projects.

Research Title	Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers
Student	Andrew Laing
Supervisor	Michael Kimber

	YES	NO
1. Will the study involve human participants?	X	
If you have answered "NO" to question 1, please go to question 15		X
2. Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g. covert observation of people)		X
3. Does the study involve participants who are unable to give informed consent? (e.g. children, people with learning disabilities)		X
4. Does the study involve participants who are commonly viewed as 'vulnerable'? (e.g. children, elderly, people with learning disabilities) **CRB check needed if YES**		X
5. Will the study require the co-operation of a third party for initial access to the groups or individuals? (e.g. students at school, residents of a nursing home)		X
6. Will the study involve discussion of sensitive topics? (e.g. sexual activity, drug use)		X
7. Could the study induce psychological stress or anxiety, cause harm or have negative consequences for the participants beyond the risks encountered in normal life?		X
8. Will deception of participants be necessary during the study?		X
9. Will blood or tissue samples be taken from participants? Are drugs, placebos or other substances (e.g. foods, vitamins) to be administered to the participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?		X
10. Will the study involve prolonged or repetitive testing or physical testing?		X
11. Is pain or more than mild discomfort likely to result from the study?		X
12. Will financial or other inducements (other than reasonable expenses) be offered to participants?		X
13. Will the study involve recruitment of patients or staff through the NHS?		X
14. Is the right to freely withdraw from the study at any time made explicit?	X	
15. Where secondary data is to be used, is the risk of disclosure of the identity of individuals minimal?	X	
16. If you are using secondary data, are you obtaining it from anywhere other than recognised data archives?		X

If you have answered YES to any of the questions (other than 14 and 15), your research proposal will be referred to the Ethics Group Panel. You will need to submit your plans for addressing the ethical issues raised by your proposal using the Research Ethics Application Form, available on the School intranet.

Appendix B: Ethical approval request form



The City of Liverpool College

The Open University

ETHICAL APPROVAL REQUEST FORM

Ethical Research Declaration – Student statement

Name: Andrew Laing

Course: BSc (Hons) Computing

Tutor/Supervisor: Michael Kimber

Research activity/Title: Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers

Module: Dissertation

Year: 2018/2019

I confirm that I have read the BERA ethical guidance and that I agree to fully apply and comply with the guidance. I understand that non-compliance with the BERA's ethical research guidelines may result in me being disciplined in line with the College's HE Behavioural policy.

Student signature: ALaing

Date: 31/10/2018

Ethical Research Approval - ERP statement

I confirm that above research proposal was presented to the CoLC's HE Ethical Research Panel (ERP)

ERP assessed proposal against ethical research guidelines.

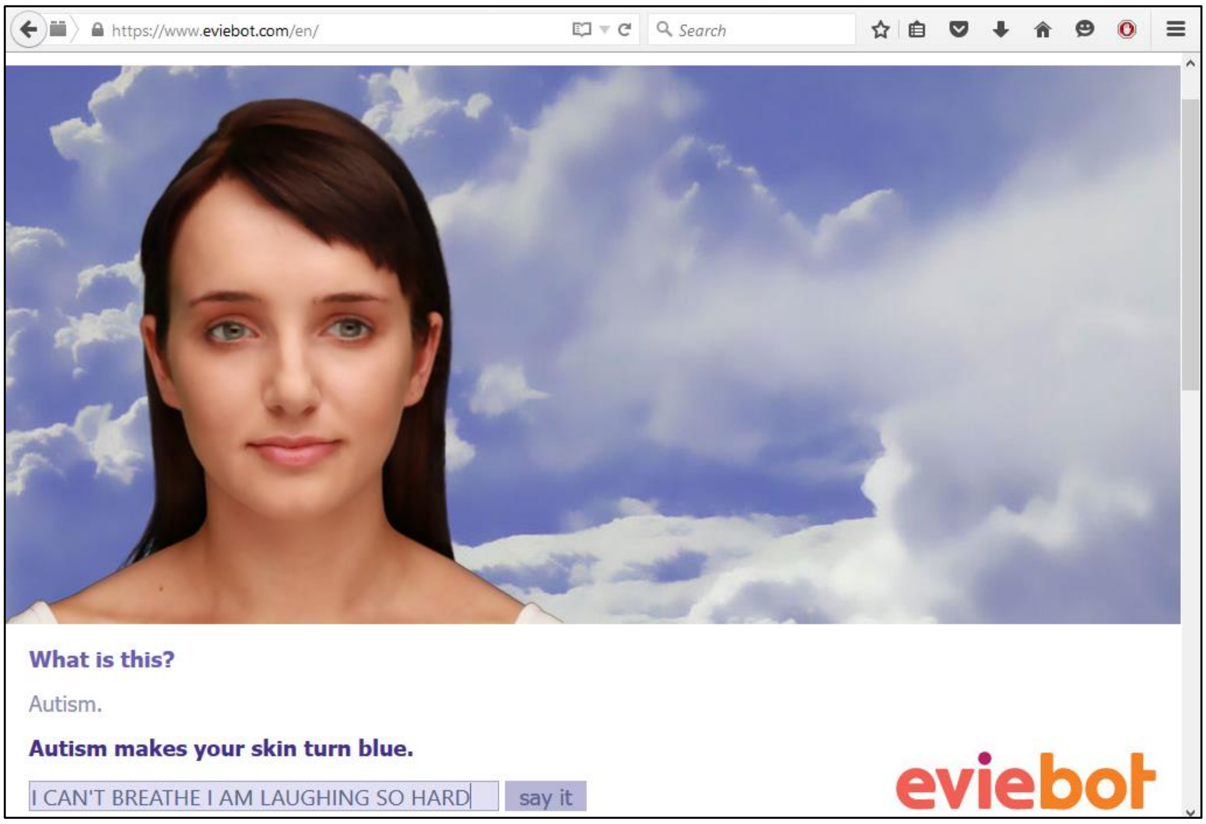
It **IS/~~IS NOT~~** granted an approval to proceed; (delete as applicable)

On behalf of Ethical Research Panel (ERP) signed by:

Programme Leader: [illegible]

Date: 31/10/2018.

Appendix C: Four examples of popular chatbot interfaces



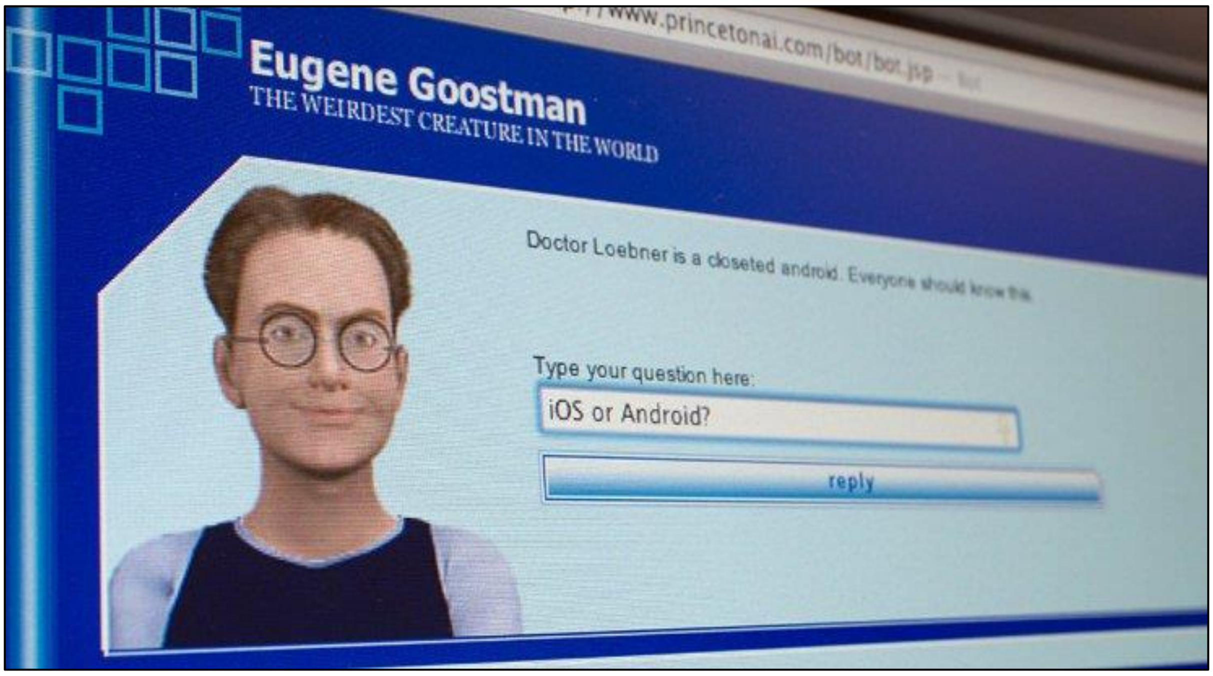
What is this?

Autism.

Autism makes your skin turn blue.

I CAN'T BREATHE I AM LAUGHING SO HARD

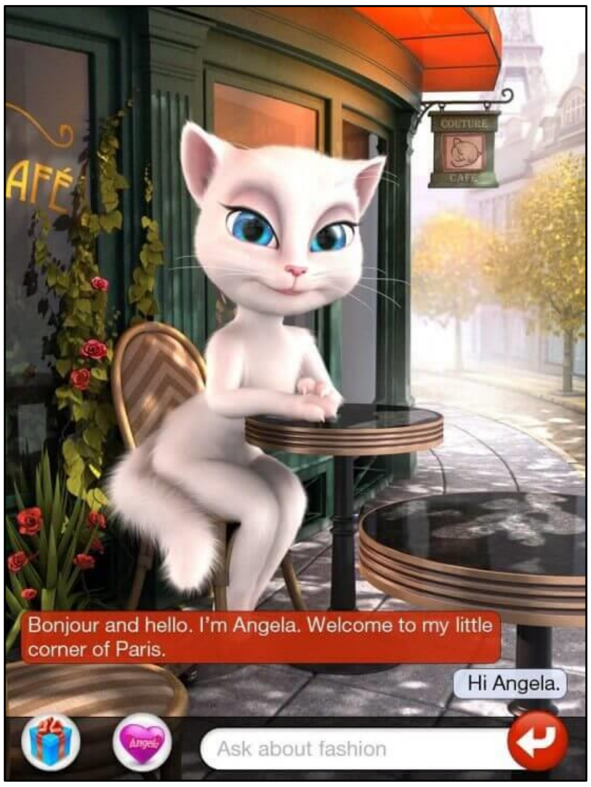
Figure 1: The online Eviebot application.

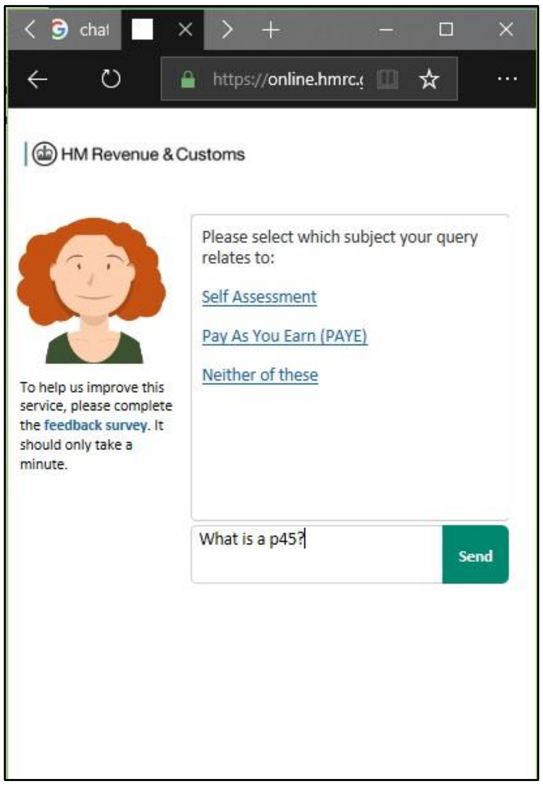

www.princetonai.com/bot/bot.jsp
Eugene Goostman
THE WEIRDEST CREATURE IN THE WORLD
Type your question here:
reply

Doctor Loebner is a closeted android. Everyone should know this.

iOS or Android?

Figure 2: The online Eugene Goostman application which convinced 33% of judges during an event marking the 60th anniversary of Alan Turing's death that it was human.


Bonjour and hello. I'm Angela. Welcome to my little corner of Paris.
Hi Angela.
Ask about fashion



Please select which subject your query relates to:

Self Assessment

Pay As You Earn (PAYE)

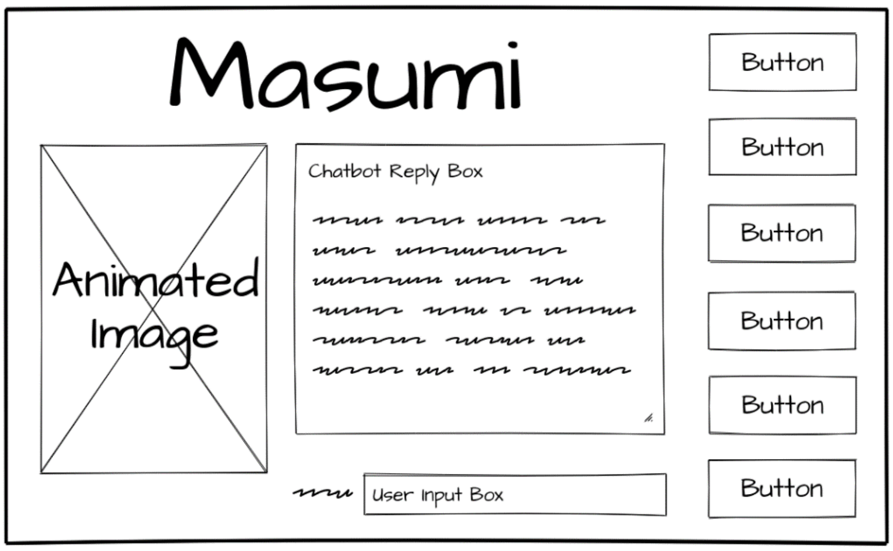
Neither of these

To help us improve this service, please complete the feedback survey. It should only take a minute.

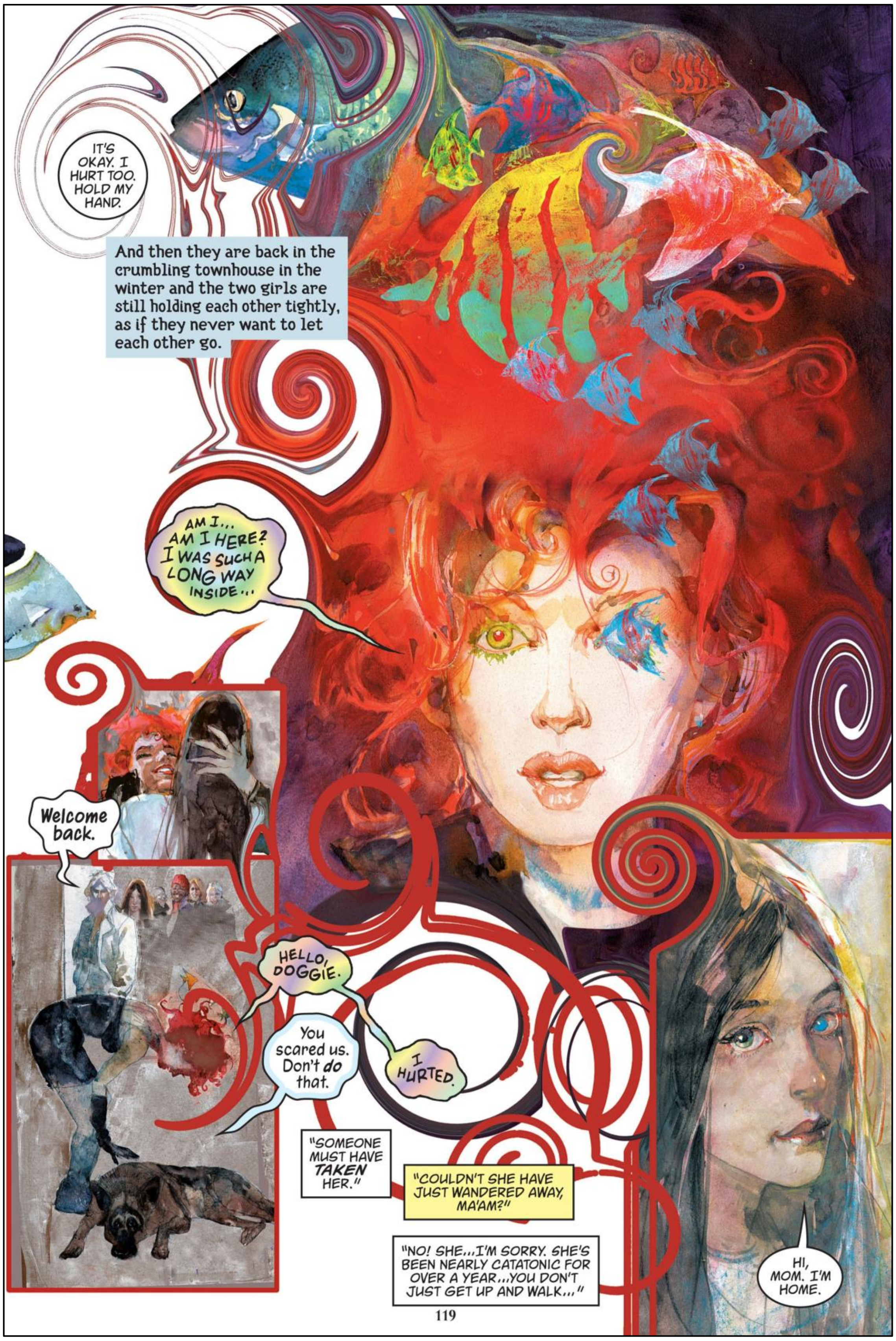
What is a p45?

Figure 3: Left the popular Talking Angela mobile phone application and right an information bot used upon the British Government's HM Revenue and Customs' website.

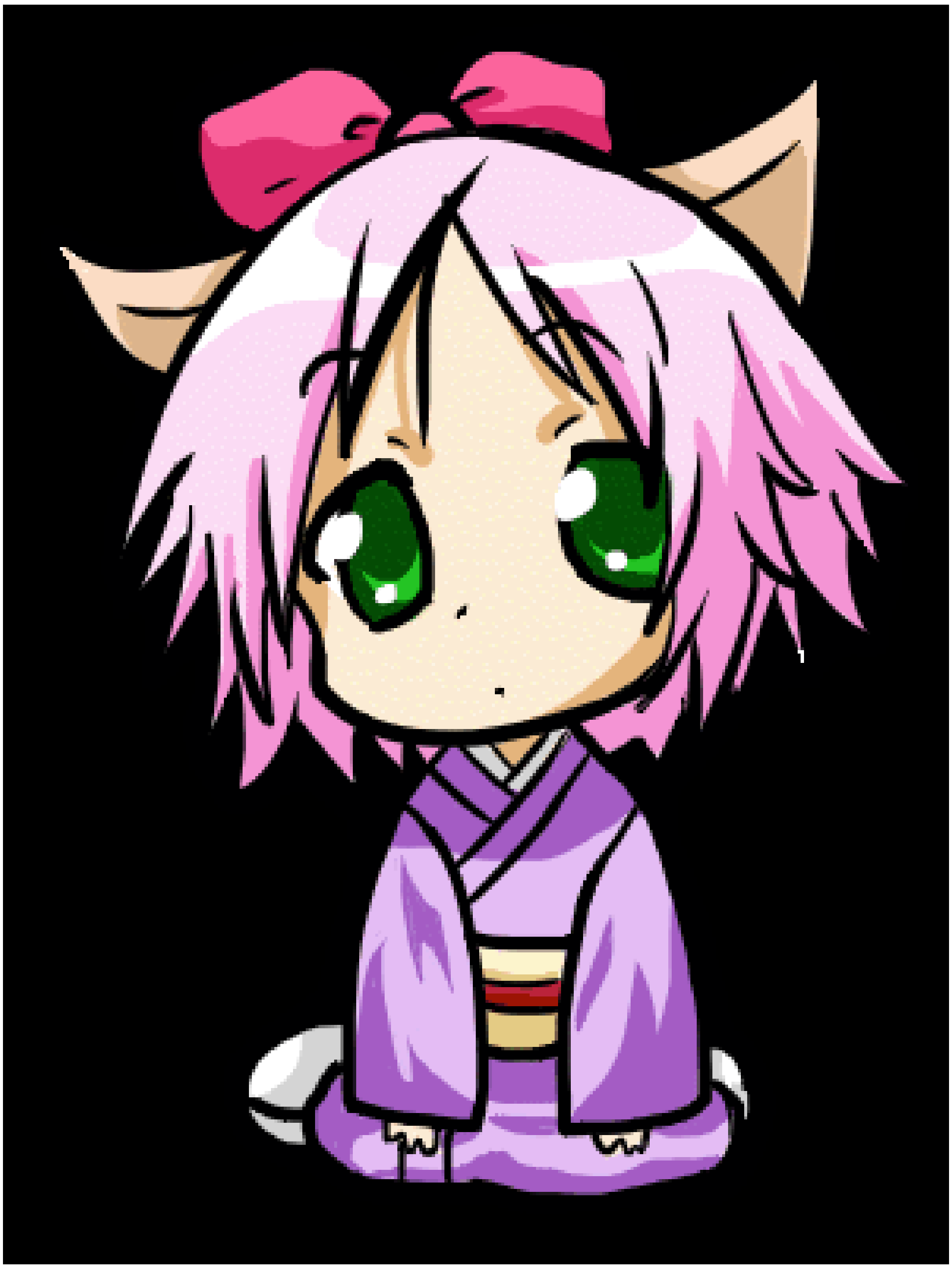
Appendix D: Interface design

PROJECT NAME: Masumi – ADHD Chatbot application.			
WIREFRAME:  Masumi Button Button Button Button Button Button Chatbot Reply Box Animated Image User Input Box			
SCREEN SIZE:	1000 x 600		
FONTS USED:	Beyond_Wonderland, Notepad		
FONT SIZES:			
Feedback Button text	Notepad 18		
Exit Button text	Notepad 32		
Logo text	Beyond Wonderland 150		
Bot response text normal	Notepad 18		
Bot response text small	Notepad 13		
User input text	Notepad 18		
TEXT COLOURS:	***RGB CODE***	***HEX CODE***	***EXAMPLE***
Bot response text colour	rgb(245, 245, 245)	#F5F5F5	⬜
User input text colour	rgb (255, 215, 225)	#FFD7E1	🟥
Logo text colour	rgb(255, 155, 155)	#FF9B9B	🟥
Feedback button text colour	rgb(255, 255, 255)	#000000	
Feedback button text mouseover colour	rgb(0, 0, 0)	#000000	⬛
Feedback button text pressed colour	rgb(255, 255, 255)	#000000	
ADDITIONAL COLOURS:	***RGB CODE***	***HEX CODE***	***EXAMPLE***
Interface background colour	rgb(0, 0, 0)	#000000	⬛
Button background colour	rgb(255, 100, 100)	#FF6464	🟥
Button background mouseover colour	rgb (210, 0, 0)	#D20000	🟥
Button background pressed colour	rgb(133, 60, 16)	#853CA4	🟪
Button foreground colour	rgb(255, 155, 155)	#FF9B9B	🟥
Button foreground mouseover colour	rgb(255, 62, 62)	#FF3E3E	🟥
Button foreground pressed colour	rgb(164, 92, 196)	#A45CC4	🟪

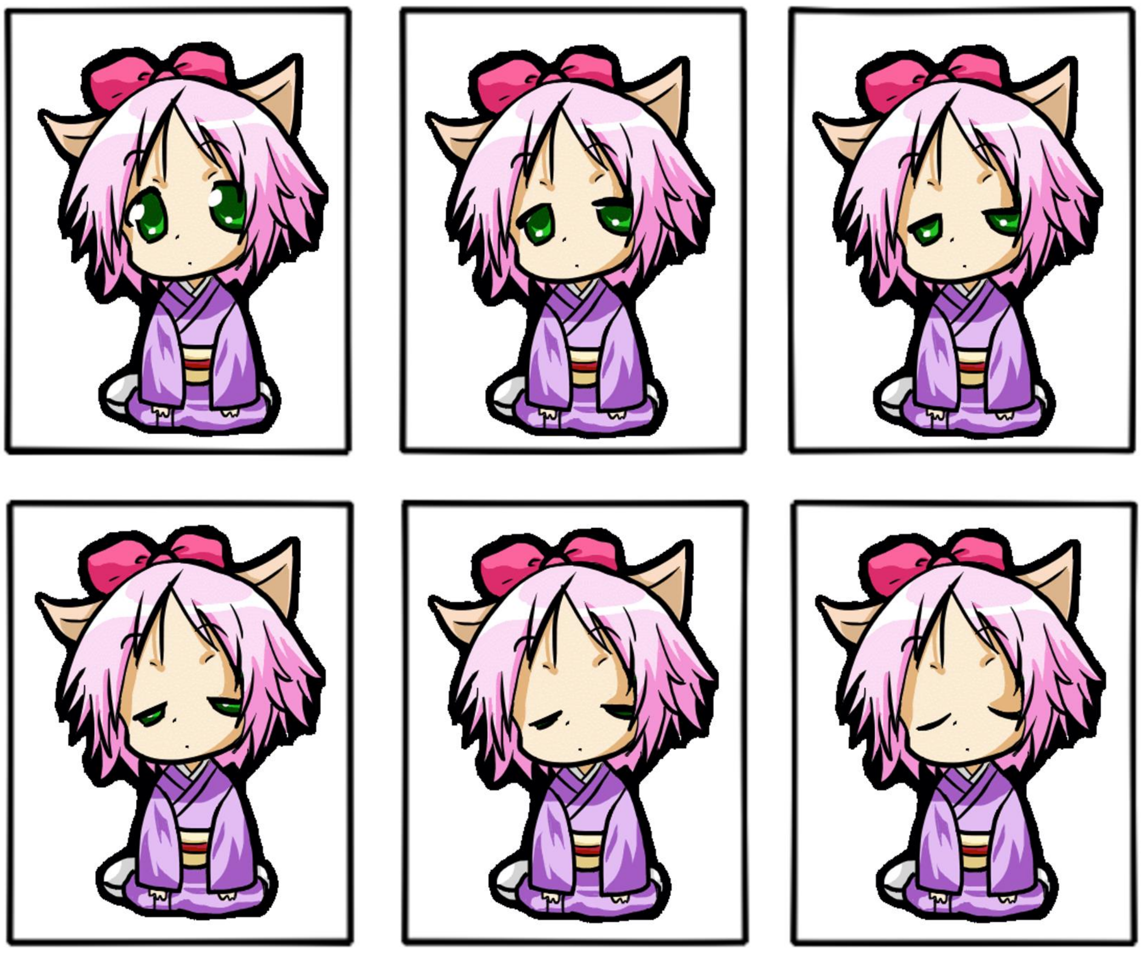
Appendix E: The Delirium character


IT'S OKAY. I HURT TOO. HOLD MY HAND.
And then they are back in the crumbling townhouse in the winter and the two girls are still holding each other tightly, as if they never want to let each other go.
AM I... AM I HERE? I WAS SUCH A LONG WAY INSIDE...
Welcome back.
HELLO, DOGGIE.
You scared us. Don't do that.
I HURTED.
"SOMEONE MUST HAVE TAKEN HER."
"COULDN'T SHE HAVE JUST WANDERED AWAY, MA'AM?"
"NO! SHE...I'M SORRY. SHE'S BEEN NEARLY CATATONIC FOR OVER A YEAR...YOU DON'T JUST GET UP AND WALK..."
HI, MOM. I'M HOME.
119

Appendix F: Avatar design

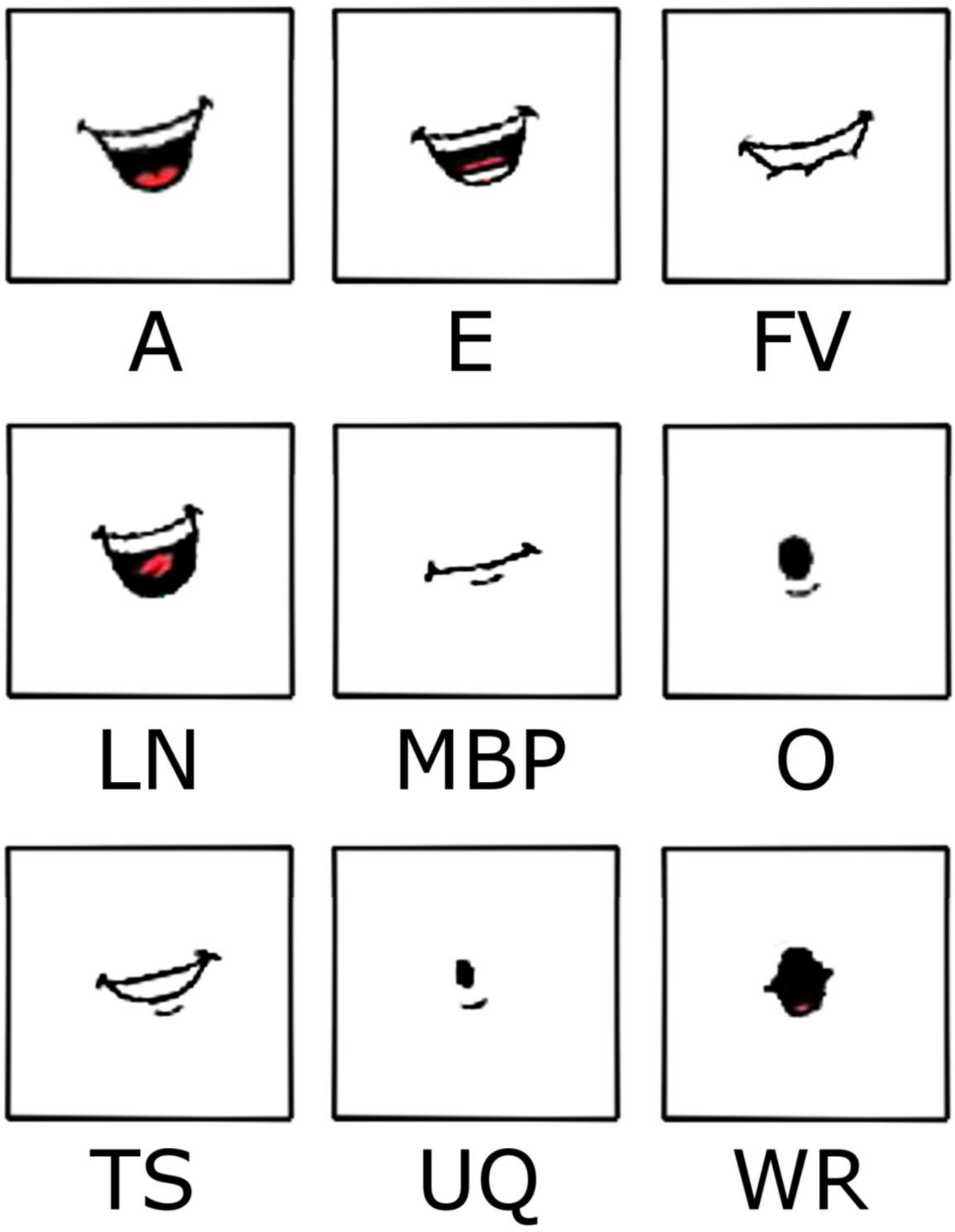


Appendix G: Blink animation images



Appendix H: Head nod/move animation images

Appendix I: Mouth shape images



The nine basic phoneme shapes identified by Sanders for the English language. (Sanders, 2018)

Appendix J: Fonts used by the application

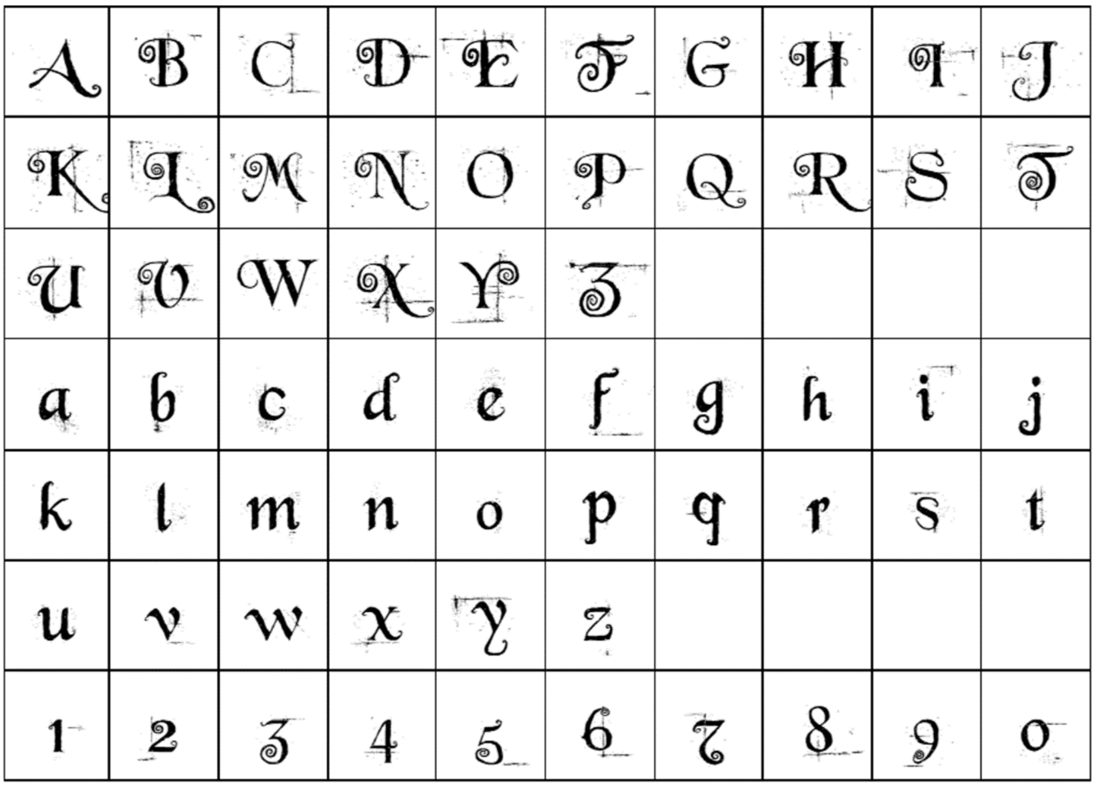


Figure 1: Wonderland font.

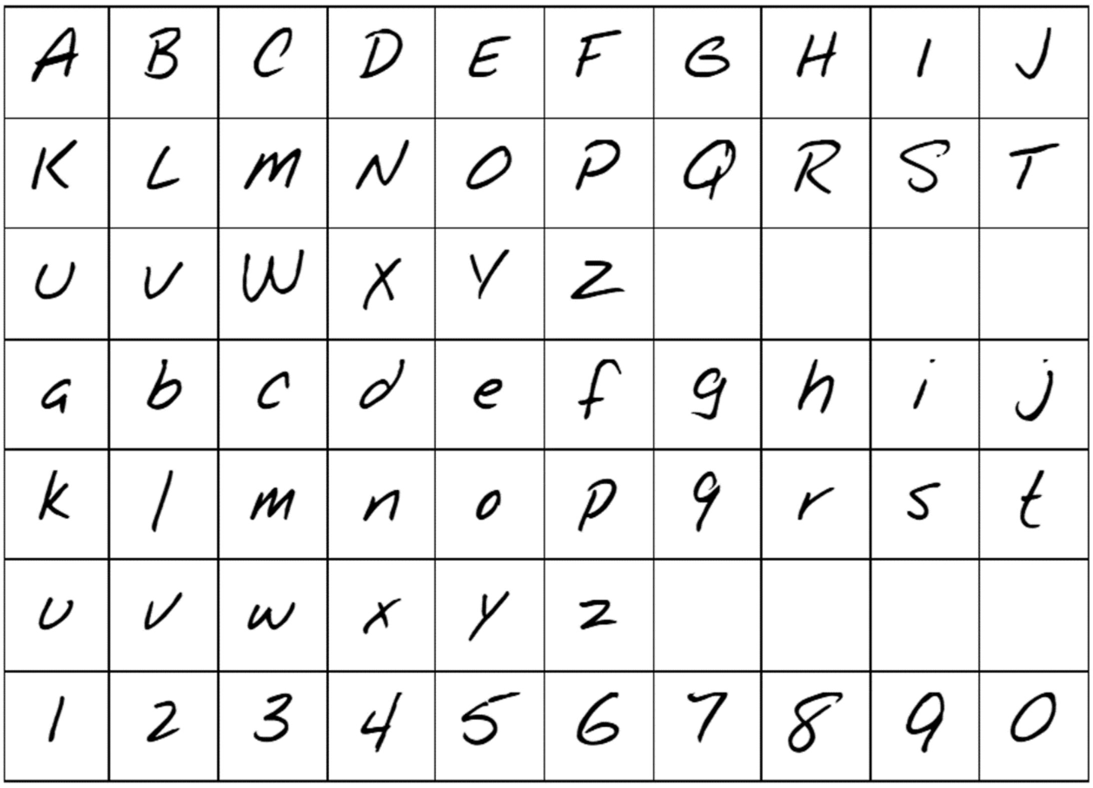


Figure 2: Notepad font.

Appendix K: Using AIML

This appendix contains a beginner's guide to AIML. It is included to give the reader an idea of how chatbot personalities are created.

AIML files contain a collection of knowledge units which provide responses to user input. The simplest AIML knowledge unit is known as a category. This is comprised of a pattern and template as in the example below.

```
<category>
  <pattern>WHO IS SPIDERMAN</pattern>
  <template>Spiderman is a comic book character created by Stan Lee.</template>
</category>
```

An AIML pattern is a question or phrase extracted from user input. It is usually a simple phrase but can also include a wildcard character to enable it to match multiple input phrases.

AIML interpreters search recursively through knowledge trees to find the best match for a given pattern. They will break down each sentence into small patterns, searching until matches are found or the recursion limit is reached.

Templates are the responses to patterns. Whilst they can be simple phrases, AIML allows interpreters to substitute stored variables into the reply to give it more relevance to the current conversation as in the example below.

```
<category>
  <pattern>NATIONALITY</pattern>
  <template>My current nationality is <bot name="nationality"/>.</template>
</category>
```

Another key AIML element is the symbolic reduction. Symbolic reductions are used to convert complex patterns into much simpler forms. When one of the complex forms is encountered its reply will be derived from the shorter one. For example, in the AIML category below the complex pattern 'DO YOU KNOW WHO SPIDERMAN IS' is reduced to its atomic version 'WHO IS SPIDERMAN.'

```
<category>
   <pattern>DO YOU KNOW WHO SPIDERMAN IS</pattern>
   <template><srai>WHO IS SPIDERMAN</srai></template>
</category>
```

Symbolic reductions are useful for deriving meaning from sentences containing synonyms or argot. In the absence of a built-in spellchecker they can also help to parse common typos.

Another feature worth mentioning is AIML's ability to provide random replies to patterns. Considering the number of times that people ask related questions or use stock phrases, adding a choice of different answers can help to stop conversations from becoming stale.

For example;

```
<category>
   <pattern>WHO IS STAN LEE</pattern>
   <template>
    <random>
   <li>The creator of Spiderman.</li>
   <li>An American comic book writer.</li>
   <li>The son of Romanian immigrants and creator of Iron Man.</li>
   </random>
   </template>
</category>
```

One of the main features which distinguishes AIML from earlier attempts at simulating conversations is its ability to store and retrieve the context of a conversation. AIML can use two varieties of context, 'topic' and 'that.' 'Topic' refers to the topic of a conversation whilst 'that' refers to a specific thing or utterance.

The 'topic' tag is used to group a collection of categories together. AIML first instructs the interpreter to set the name of the current topic, so that when a matching pattern is encountered, the context in which it was spoken can help to determine the response.

```
<category>
    <pattern>LETS DISCUSS COMICS</pattern>
    <template>Yes <set name = "topic">comics</set></template>
</category>

<topic name = "comics">
    <category>
      <pattern>FAVORITE ARTIST</pattern>
      <template>I really like the work of Steve Ditko. </template>
    </category>

    <category>
      <pattern> I LIKE HORROR </pattern>
      <template>Yes, horror comics are great. </template>
    </category>
</topic>

<topic name = "painting">
    <category>
      <pattern>FAVORITE ARTIST</pattern>
      <template>I really like the work of Mark Rothko. </template>
    </category>
</topic>
```

AIML can also set the value of other context variables such as ‘he’, ‘it’. More importantly it can also set a series of variables known as predicates which store information about the current user.

Predicates can include the user’s name, age, sex, and even boyfriend or girlfriend’s name and these can be easily recalled when the AIML interpreter detects a pattern which requires their use.

For example;

```
<category>
  <pattern>MY NAME IS *</pattern>
  <template>
    <think>
    <set name="username"/><thatstar/></set></think>Nice to meet you <thatstar/>
  </template>
</category>

<category>
  <pattern>DID YOU LIKE SPIDERMAN</pattern>
  <template>Yes, I think Spiderman is great <get name="username"/></template>
</category>
```

Note that in the example above, '<thatstar/>' extracts the user's name from the wildcard position within the input phrase and sets it as the 'username' predicate. Also, code between the '<think>' tags is performed without being added to the reply.

AIML has many more features which enable it to mimic human conversation. A more detailed examination of the topic can be found inside of The Elements of AIML Style. (Wallace, 2003)

Appendix L: Class references

Button Class	
Description	The Button class is used to implement simple buttons for PyGame.
Attributes	
Attribute Name	**Details**
bg	A PyGame rectangle for the button's background.
bg_colour	The button's current background colour.
border_width	The width of the button's border.
btn_bg_colour	The background colour of the button.
btn_bg_mouseover_colour	The mouseover colour of the button.
btn_bg_pressed_colour	The colour of the button when it is pressed.
btn_fg_colour	The foreground colour of the button.
btn_fg_mouseover_colour	The foreground mouseover colour of the button.
btn_fg_pressed_colour	The foreground colour of the button when it is pressed.
btn_txt_colour	The colour of the button text.
btn_txt_mouseover_colour	The mouseover colour of the button text.
btn_txt_pressed_colour	The colour of the button text when the button is pressed.
caption	The button’s text.
fg	A PyGame rectangle for the button's foreground.
fg_colour	he button's current foreground colour.
over_button	True if the cursor is over the button, otherwise False.
pressed	True if the button is being pressed, otherwise False.
text	A text object to hold the button caption.
textFont	The font used to display the button caption.
txt_colour	The button's current text colour.

txt_left	The position for the left edge of the text.
txt_top	The position for the top edge of the text.
Methods	
Method Name	**Details**
__init__	Description: Initialises ButtonClass with a caption, button background attributes, a position for the text, and button background attributes.
	Parameters: caption: A text string cation for the button. bg_attributes: A tuple containing the position and size attributes for the button's background rectangle. (Pixels from left-hand edge, width in pixels, Pixels from top edge, height in pixels) txt_position: A tuple containing the top-left position of the buttons caption. (Pixels from left-hand edge, Pixels from top edge) font: The font used to display the button caption. fontsize: The caption's font size.
convertTextToSpeech	Description: Creates a temporary Text-to-Speech WAV of a text, plays it then deletes it
	Parameters: textToSpeak: A line of text to convert to speech.
createWAV	Description: Creates a Text-to-Speech WAV from a text string.
	Parameters: textToSpeak: A line of text to convert to speech.
button_was_pressed	Description: Detects whether the button is being pressed. If yes, it will update the button press state variables.
	Parameters: event: A PyGame event object.
	Returns: True if the button is being pressed, otherwise False.
initialiseButtonRects	Description: Initialises the PyGame rectangles used to make the button.

	Parameters: bg_attributes: A tuple containing the position and size attributes for the button's background rectangle. (Pixels from left-hand edge, width in pixels, Pixels from top edge, height in pixels)
initialiseCaption	Description: Initialises the PyGame text object used to hold the button caption.
	Parameters: caption: A text string cation for the button. txt_position: A tuple containing the top-left position of the button's caption. (Pixels from left-hand edge, Pixels from top edge) font: The font used to display the button caption. fontsize: The caption's font size.
initialiseColours	Description: Initialises the default colours used on the button.
is_collision	Description: Detects whether the mouse is over the button.
	Parameters: mousePosition: A PyGame mouse position.
	Returns: True if the mouse is over the button, otherwise False.
is_over_button	Description: Returns the state of the over_button variable.
	Returns: True if the mouse is over the button, otherwise false.
is_pressed	Description: Returns the state of the pressed variable.
	Returns: True if the button is being pressed, otherwise false.
render	Description: Blits the button onto a PyGame surface object.
	Parameters: surface: A PyGame surface object.
set_over_button	Description: Used to update the over_button state variables, and button colours each time the user mouses over, or off the button.
set_pressed	Description: Used to update the button pressed state variables and button colours each time the user presses or releases the button.

MasumiBrain Class	
Description	The MasumiBrain class is used to handle the application's use of the PyAIML library.
Attributes	
Attribute Name	**Details**
k	A PyAIML kernel.
voice	A Microsoft SAPI voice.
Methods	
Method Name	**Details**
__init__	Description: Initialises MasumiBrain.
createNewBrain	Description: Loads an AIML knowledge tree into the PyAIML kernel and saves it to a brain file (.brn) for faster loading the next time that the application is opened.
getBotResponse	Description: Sends a text string to the PyAIML kernel for a response from the stored knowledge tree.
	Parameters: inputText: A text string.
	Returns: A tidied response to the text string.
loadBrain	Description: If a brain file (.brn) exists loads it into the PyAIML kernel, otherwise loads an AIML knowledge tree into the PyAIML kernel and saves it to a brain file (.brn) for faster loading the next time that the application is opened.
saveBrain	Description: Writes out the knowledge tree stored in the PyAIML kernel to a brain file (.brn).

MasumiInterface Class

Description	The MasumiInterface class is used to display the application's GUI, handle user input, handle events, and write texts to a log file.
Attributes	
Attribute Name	**Details**
blinkActive	True if the blink animation is active, otherwise False.
blinkImages	A list of the blink/default animation images.
blinkIndex	Index of the blink/default animation image being displayed.
blinkRate	How often the blink animation should be displayed (e.g., 3 for every 3 seconds)
Brain	An instance of the MasumiBrain class.
btn_1	A user feedback button.
btn_2	A user feedback button.
btn_3	A user feedback button.
btn_4	A user feedback button.
btn_5	A user feedback button.
btn_6	A user feedback button.
btn_exit	A button used to close the application.
clock	Timer used for the PyGame game loop.
currentMannerism	The mannerism currently being animated. (0 – default/blink, 1 – head nod/move, 2 – mannerism2)
currentTime	The current time.
is_speaking	True if TTS is playing, otherwise False.
lenPromptText	The length of the input prompt text.
logoFont	Font used to display the application log.
logoSurface	A PyGame surface used to hold the logo text.
mannerism2Active	True if the mannerism2 animation is active, otherwise False.

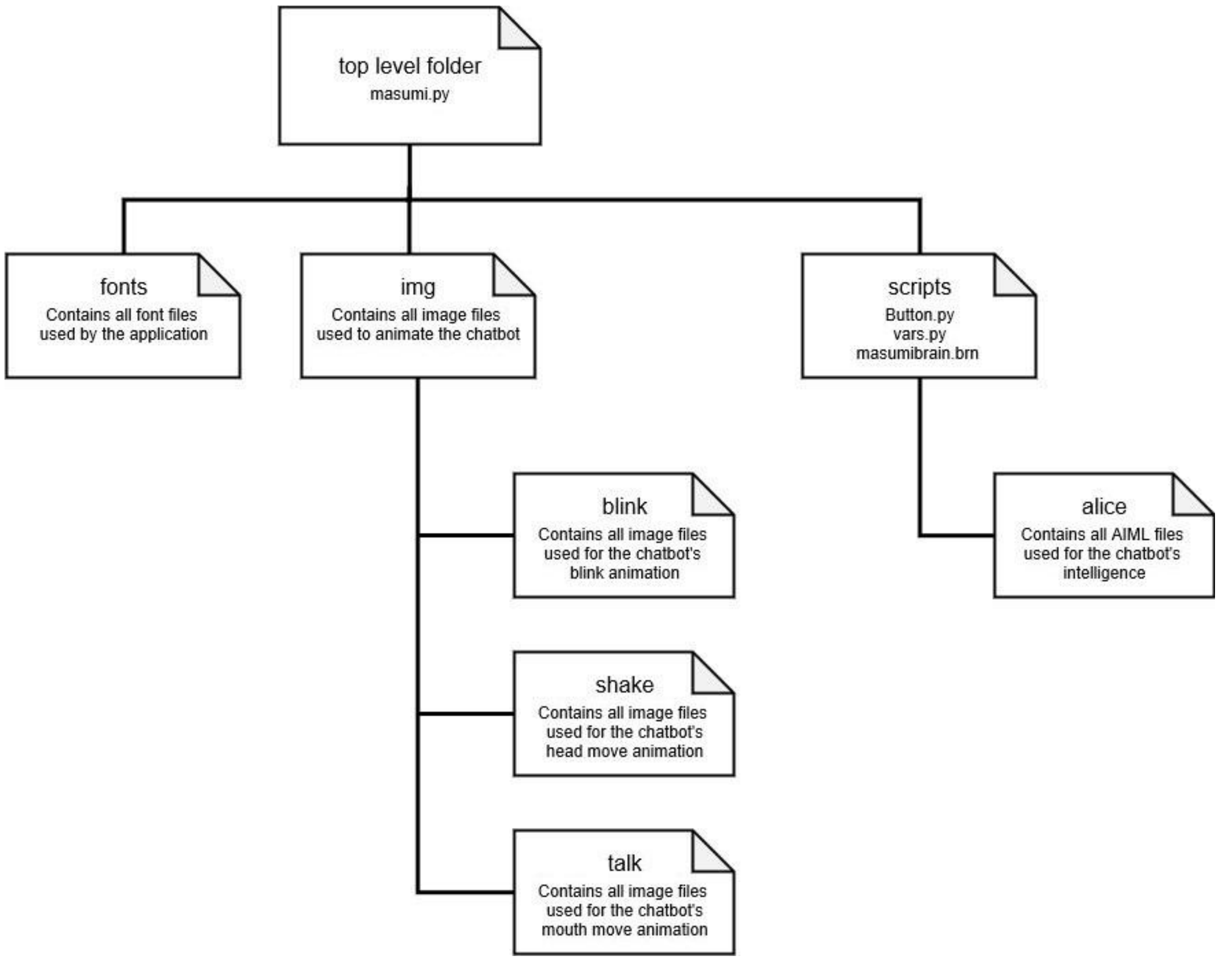
mannerism2ImageList	A list of the mannerism2 animation images
mannerism2Index	The index of the mannerism2 image being displayed.
miniResponseFont	Font used for response when it is too long to fit on the screen.
mouthIndex	The index of the mouth image being displayed.
mouthImages	A list of mouth images used for the speaking animation.
mouthMoveImageList	A list of mouth images for the current reply being spoken with TTS.
nodActive	True if the head nod/move animation is active, otherwise False.
nodImageList	A list of the images used for the head nod/move animation.
nodIndex	The index of the head nod/move image being displayed.
nodStart	A time used to regulate the head nod/move animation's frequency.
promptText	A string used as a prompt in the user input box.
responseFont	Font used for the bot's response text.
screen	A PyGame screen object.
startTime	A time used to regulate mannerism selection.
textForResponseBox	A string to display in the response box.
userInputFont	The font used to display the user's input text.
userText	A string input by the user.
Voice	A MasumiVoice instance.
Methods	
Method Name	**Details**
__init__	Description: Initialises MasumiInterface.
checkForButtonPress	Description: Checks for a feedback button or exit button press. Writes feedback button text to the log file.
	Parameters: event: A PyGame event.
	Returns: True if the exit button was pressed, otherwise False.

closeApplicationCleanly	Description: Speaks the closing text. Saves any changes to the brain file. Writes the closing time to the log file. Closes the application
createButtons	Description: Creates the user feedback buttons.
createFonts	Description: Creates the fonts used to display text on-screen.
createLogfile	Description: Create a logfile for the session.
createMouthMoveImageList	Description: Populates the list used for animating the bot's response texts.
	Parameters: text: A text string.
initialiseBrain	Description: Initialises an instance of MasumiBrain.
initialiseImageLists	Description: Loads the images used to display the chatbot's avatar animations.
initialiseInterface	Description: Initialises the user interface.
initialiseScreen	Description: Initialises the GUI screen.
initialiseVariables	Description: Initialises MasumiInterface variables.
initialiseVoice	Description: Initialises an instance of MasumiVoice.
keyPressCallback	Description: Checks for a key press event and deals with it appropriately.
	Parameters: event: A PyGame event.
loadBaseMasumiImages	Description: Loads the blink/default images for the chatbot's avatar animation.
loadHeadMovementImages	Description: Loads the images to animate the chatbot's head nod/move animation.
loadMannerism2Images	Description: Loads the images for the Mannerism2 animation (a mouth movement resembling a goldfish which is activated after a long period of user inactivity).
loadMouthShapeImages	Description: Loads the images used for the chatbot's speech animation.
renderBlinkAnimation	Description: Determines which blink/default animation image to display and blits it to the PyGame surface.

renderBotResponseText	Description: Renders the chatbot's response text on-screen.
renderButtons	Description: Renders the user feedback buttons on-screen.
renderHeadMoveAnimation	Description: Determines which head nod/move animation image to display and blits it to the PyGame surface.
renderLogo	Description: Renders the logo text on-screen.
renderMannerism2Animation	Description: Determines which mannerism2 animation image to display and blits it to the PyGame surface.
renderMouthShape	Description: Determines which speaking animation image to display and blits it to the PyGame surface.
renderTextInputBox	Description: Renders the text input box on-screen
runLoop	Description: The main PyGame game loop used to display the GUI.
setCurrentMannerism	Description: Used to detect the current animation mannerism to display.
startTimers	Description: Initialises timers for the PyGame game loop.
writeButtonPressToLogFile	Description Writes the text of the button pressed to the logfile.
	Parameters: message: The button's caption text.
writeClosingTimeToLogFile(	Description: Writes the time that the application closed to the logfile.

MasumiVoice Class	
Description	The MasumiVoice class is used to handle the application's Text-to-Speech functionality.
Attributes	
Attribute Name	**Details**
engine	A SAPI SpVoice Interface object.
voice	A Microsoft SAPI voice.
Methods	
Method Name	**Details**
__init__	Description: Initialises MasumiVoice with a Microsoft SAPI voice.
	Description: voice: A Microsoft SAPI voice description
convertTextToSpeech	Description: Creates a temporary Text-to-Speech WAV of a text, plays it then deletes it
	Parameters: textToSpeak: A line of text to convert to speech.
createWAV	Description: Creates a Text-to-Speech WAV from a text string.
	Parameters: textToSpeak: A line of text to convert to speech.

Appendix M: Development Folder Structure


top level folder
masumi.py
fonts
Contains all font files used by the application
img
Contains all image files used to animate the chatbot
scripts
Button.py
vars.py
masumibrain.brn
blink
Contains all image files used for the chatbot's blink animation
shake
Contains all image files used for the chatbot's head move animation
talk
Contains all image files used for the chatbot's mouth move animation
alice
Contains all AIML files used for the chatbot's intelligence

Appendix N: The TextRender class

The TextRender class is a wrapper put around developer David Clark's render_textrext function which creates a PyGame text rectangle. This function word-wraps and anti-aliases the text string sent to it, returning a surface containing the text rectangle.

It was decided to put the method within an uninstantiated class so that the MasumiInterface class could call it as a static method. This allows the function to be used in a manner which explicitly references its source.

```
rendered_text = TextRender.render_textrect(self.textForResponseBox,
                                           self.responseFont,
                                           my_rect,
                                           WHITE, BLACK, 0)
```

The class itself is shown below.

```
# Author:       David Clark
# Email:        silenus@telus.net
# Last Updated: 23/05/2001
# Description:  Used to create add a text rectangle onto a PyGame surface.
# Changes:      Added the render_textrect function to a class so that it could be
#               called as a static class method (Andrew Laing 17/02/2019).

import pygame

class TextRectException:
    """ An exception class used by TextRender. """
    def __init__(self, message=None):
        self.message = message

    def __str__(self):
        return self.message


class TextRender:
    """ A utility class, never implemented, used to contain
        the render_textrect() function. """
    def __init__(self):
        pass
```

```
@staticmethod
def render_textrect(string, font, rect, text_color,
                    background_color, justification=0):
    """ Returns a surface containing the passed text string, reformatted
    to fit within the given rect, word-wrapped as necessary. The text
    will be anti-aliased.

    Takes the following arguments:

    string - the text you wish to render. \n begins a new line.
    font - a Font object
    rect - a rect giving the size of the surface requested.
    text_color - a three-byte tuple of the rgb value of the
                 text color. ex (0, 0, 0) = BLACK
    background_color - a three-byte tuple of the rgb value of the surface.
    justification - 0 (default) left-justified
                    1 horizontally centered
                    2 right-justified

    Returns the following values:

    Success - a surface object with the text rendered onto it.
    Failure - raises a TextRectException if the text won't fit
              onto the surface.
    """
    final_lines = []
    requested_lines = string.splitlines()

    # Create a series of lines that will fit on the provided rectangle.
    for requested_line in requested_lines:
        if font.size(requested_line)[0] > rect.width:
            words = requested_line.split(' ')
            # if any of our words are too long to fit, return.
            for word in words:
                if font.size(word)[0] >= rect.width:
                    raise TextRectException("The word " +
                                            word +
                                           " is too long to fit in the rect passed.")
            # Start a new line
            accumulated_line = ""
            for word in words:
                test_line = accumulated_line + word + " "
                # Build the line while the words fit.
                if font.size(test_line)[0] < rect.width:
                    accumulated_line = test_line
                else:
                    final_lines.append(accumulated_line)
                    accumulated_line = word + " "
            final_lines.append(accumulated_line)
        else:
            final_lines.append(requested_line)

            # Let's try to write the text out on the surface.
    surface = pygame.Surface(rect.size)
    surface.fill(background_color)

    accumulated_height = 0
    for line in final_lines:
        if accumulated_height + font.size(line)[1] >= rect.height:
            raise TextRectException("Text string was too tall.")
        if line != "":
            tempsurface = font.render(line, 1, text_color)
            if justification == 0:
                surface.blit(tempsurface, (0, accumulated_height))
            elif justification == 1:
                surface.blit(tempsurface,
                ((rect.width - tempsurface.get_width()) / 2,
                  accumulated_height))
            elif justification == 2:
                surface.blit(tempsurface,
                (rect.width - tempsurface.get_width(),
                 accumulated_height))
            else:
                raise TextRectException("Invalid justification argument: " +
                                        str(justification))
        accumulated_height += font.size(line)[1]

    return surface
```

Appendix O: Code examples

Introduction

This appendix explains how some of the application's main functionality was implemented. It is divided into five sections which look at:

the main loop

how the application responds to user input

text-to-speech functionality

implementing the lip synchronisation animation

conversation logging

The application's main loop

The application is run inside of a loop which updates at the rate of 30 frames per second. Before the loop is started, the ***startTimers*** method is called to set several variables used to help the application time events related to updating the interface.

```
def startTimers(self):
    """ Initialises timers for the PyGame game loop. """
    self.clock = pygame.time.Clock()
    self.startTime = time.time()
    self.nodStart = time.time()
```

The application then runs a while loop which can only be escaped by closing the program. Put simply, the application checks for events, processes them and calls a series of methods to update the interface 30 times per second.

Two class variables are set within the loop, ***currentTime*** and ***currentMannerism***, which are used to select which animation frames to use.

```
def runLoop(self):
    """ The main PyGame game loop used to display the GUI. """
    # Start the timers
    self.startTimers()

    while True:
        # Update the current time
```

```
self.currentTime = time.time()

# Determine current mannerism
self.setCurrentMannerism()

# Check if the voice is being used for animating the mouth
if pygame.mixer.get_busy() == 1:
    self.is_speaking = True
else:
    self.is_speaking = False

# Check for events
for event in pygame.event.get():
    if event.type == QUIT:
        self.closeApplicationCleanly()
        return
    elif event.type == KEYDOWN:
        self.keyPressCallback(event)
    else:  # event listeners for button events
           # returns 1 if exit is pressed
        if self.checkForButtonPress(event) == 1:
            return

# Update the screen
self.screen.fill(BLACK)
self.renderLogo()
self.renderBotResponseText()
self.renderButtons()
self.renderTextInputBox()
self.renderHeadMoveAnimation()
self.renderBlinkAnimation()
self.renderMannerism2Animation()
self.renderMouthShape()

pygame.display.update()
self.clock.tick(30)
```

Responding to user input

The user's key presses are drawn to the interface along with the current response from the chatbot at each pass through the main loop. The chatbot's response is set when the user presses the ***ENTER*** key.

Firstly, when a key press is detected within the main loop, the ***keyPressCallback*** method is called.

```
# Check for events
for event in pygame.event.get():
    if event.type == QUIT:
        self.closeApplicationCleanly()
        return
    elif event.type == KEYDOWN:
        self.keyPressCallback(event)
    else:  # event listeners for button events
           # returns 1 if exit is pressed
        if self.checkForButtonPress(event) == 1:
            return
```

The ***keyPressCallback*** method first checks for a ***RETURN***, ***BACKSPACE*** or ***ESCAPE*** key press and handles them accordingly. If none of these keys were pressed, and the key was an allowed character, it will be added to the ***userText*** string used to hold the user's input. The text currently input by the user is rendered to the screen later when the main loop calls the ***renderTextInputBox*** method to update the interface.

For responding to user input, when the ***ENTER*** button is pressed it will first check whether the user wishes to save the ***PyAIML*** kernel to the brain file or exit the program, if not it will call the ***getBotResponse*** method.

```
def keyPressCallback(self, event):
    """ Checks for a key press event and deals with it appropriately.
    :param event: A PyGame event.
    """
    if event.key == K_RETURN:
        userSays = (self.userText[self.lenPromptText:]).lower()
        self.userText = self.promptText  # clear user input text

        if userSays == "save brain":
            self.Brain.saveBrain()
```

```
                self.textForResponseBox = "My brain has been saved."
            elif userSays == "exit program":
                self.closeApplicationCleanly()
                return
            else:
                self.textForResponseBox = self.Brain.getBotResponse(userSays)
                self.userText = self.promptText  # clear user input text

            self.Voice.convertTextToSpeech(self.textForResponseBox)

            # If there is no response (for example, because of recursion
            #   depth exceeded error)
            if len(self.textForResponseBox) <= 1:
                if logConversation:
                    with open(logfilename, "a") as myfile:
                        myfile.write("\nBot:  <<< OUTPUT ERROR!!! >>>")

        elif event.key == K_BACKSPACE:
            if len(self.userText) > self.lenPromptText:
                self.userText = self.userText[:-1]

        elif event.key == K_ESCAPE:
            self.closeApplicationCleanly()
            return

        else:
            carac = event.dict['unicode']
            if carac in sv.allowed:
                self.userText = self.userText + carac
```

Aside from logging conversations, the ***getBotResponse*** method sends user input to the PyAIML kernel for a response using the ***respond*** method. This response is tidied up with a regular expression then returned to the ***keyPressCallback*** method where it is used to set the ***textForResponseBox*** variable. This variable is later used when the main loop calls the ***renderBotResponseText*** method to update the interface.

```
def getBotResponse(self, inputText):
    """ Sends a text string to the PyAIML kernel for a response
        from the stored knowledge tree.

    :param inputText: A text string.
    :return: A tidied response to the text string.
    """
```

```
''' Get response and tidy it up. '''
# Log the user's text
if logConversation:
    with open(logfilename, "a") as myfile:
        myfile.write("\nUser: ")
        myfile.write(inputText)

a = self.k.respond(inputText)

# Tidy up the bot's response
tidiedResponse = " ".join(a.split())
tidiedResponse = re.sub(r'\s([?.!"](?:\s|$))', r'\1', tidiedResponse)
tidiedResponse = tidiedResponse.capitalize()

# Log the bot's response
if logConversation:
    with open(logfilename, "a") as myfile:
        myfile.write("\nBot:  ")
        myfile.write(tidiedResponse)

# If there was no response, change the subject :D
if tidiedResponse=="":
    tidiedResponse="Lets talk about something else, please."

return tidiedResponse
```

Implementing the text-to-speech functionality

When the application is opened the ***MasumiInterface*** class creates an instance of the ***MasumiVoice*** class to handle text-to-speech functionality using the ***initialiseVoice*** function. If something goes wrong the text-to-speech functionality will be disabled, otherwise the application will play the opening message. Note: the chatbot's name has been spelled out phonetically because otherwise the Microsoft SAPI voice will pronounce it as '***Ma-djew-me***'.

```
def initialiseVoice(self):
    """ Initialises an instance of MasumiVoice. """
    try:
        self.Voice = MasumiVoice(1)
        self.Voice.engine.speak(sv.nonBotVoiceTags +
                    "Welcome to the Massoomi ADHD chat bot application.")
    except:
        self.Voice = MasumiVoice(0)
```

The ***Voice.engine*** object is an interface to the Microsoft SAPI voice functions within the Windows 10 operating system. The ***speak*** method used above sends a text string to the interface for it to be converted to speech. This text string consists of a description of the SAPI voice to use and the text to convert.

The two SAPI voices used by the application are defined inside of the ***vars.py*** file located in the ***scripts*** folder. Voices can be chosen from those installed upon the operating system, and the accent, pitch and speed they speak at customised within the XML voice tag preceding the text to convert. Whilst many SAPI voices are available for download, those that were chosen are default ones provided with the operating system.

```
# These variables are used to define the SAPI voice to speak with
nonBotVoiceTags = '<voice required="Gender=Male">
                   <lang langid="409"><pitch middle="-9"/>
                   <rate speed="+1">'
voiceTags =       '<voice required="Gender=Female">
                   <lang langid="809"><pitch middle="+6"/>
                   <rate speed="+2">'
```

Adding speech to the application required a further step. Although in the first example speech is produced, this is run before the opening of the user interface. One of the problems discovered during development was that running this method interrupted the loop used to animate the character. The screen would freeze during speech, and aside from the character animation stopping the user would be unable to enter text into the input box.

The solution was to use one of PyGame's built-in methods for playing sound effects within a separate thread than the animation loop. Two methods are used to achieve this. Firstly, the ***createWAV*** method outputs the audio stream produced by a SAPI voice method to a temporary WAV file stored upon the local system. This file is then played using the PyGame mixer's ***play*** method and deleted.

```python
def convertTextToSpeech(self, textToSpeak):
    """ Creates a temporary Text-to-Speech WAV of a text, plays it
        then deletes it
        :param textToSpeak: A line of text to convert to speech.
    """
    if self.voice:
        self.createWAV(textToSpeak)
        try:
            soundObj = pygame.mixer.Sound('temp.wav')
            soundObj.play()  # Continues code during sound playing
            subprocess.getoutput('del temp.wav')
        except:  # If there is an error create an audio error message
            self.createWAV("Program Error")
            soundObj = pygame.mixer.Sound('temp.wav')
            soundObj.play()
            subprocess.getoutput('del temp.wav')


def createWAV(self, textToSpeak):
    """ Creates a Text-to-Speech WAV from a text string.
    :param textToSpeak: A line of text to convert to speech.
    """

    # If there is no response (for example, because
    #    of a recursion depth exceeded error)
    if len(textToSpeak) <= 1:
        textToSpeak = "Lets talk about something else, please."

    # Windows pronounces bot's name as Mahjewmi
    textToSpeak = textToSpeak.replace("masumi", "Massoomi")
```

```
# Create a stream to write TTS output to a WAV file
stream = CreateObject("SAPI.SpFileStream")
outfile = "temp.wav"
stream.Open(outfile, SpeechLib.SSFMCreateForWrite)
self.engine.AudioOutputStream = stream

# Add voice tags to select the TTS voice
#  in front of the text string to convert
#  then write the stream out to the WAV file
textToSpeak = sv.voiceTags + textToSpeak
self.engine.speak(textToSpeak)
stream.Close()
```

Implementing the lip synchronisation animation

Synchronising the mouth shape being displayed on the avatar, and the words being spoken is achieved using a naïve method which tries to achieve the closest match.

When the application is started, the images holding the mouth shapes are loaded and stored into the ***mouthImages*** list. These images are PNG files with a transparent layer which enables them to be drawn over the main avatar image.

```
for img in mouthshapes:
    self.mouthImages.append(pygame.image.load(img).convert_alpha())
```

During each pass through the interface's '*main loop*' the ***renderMouthShape*** method is called to determine if a text-to-speech WAV file is being played. If so, the ***is_speaking*** variable will be set to true.

If it has not already done so, the method will add indexes within the ***mouthImages*** list of the mouth shapes to render for the current text to the ***mouthMoveImageList*** list.

The ***renderMouthShape*** method will pop an index from ***mouthMoveImageList*** at each pass through the '*main loop*' and use it to render the mouth shape on top of the avatar.

```
def renderMouthShape(self):
    """ Determines which speaking animation image to display
        and blits it to the PyGame surface.
    """
    if self.is_speaking:
        # Do not move head or use mannerism 2 whilst speaking
        self.currentMannerism = 0
        self.mannerism2Active = False
        self.mannerism2Index = 0

        self.nodStart = self.currentTime
        if self.mouthMoveImageList == None:
            self.createMouthMoveImageList(self.textForResponseBox)
        if len(self.mouthMoveImageList) > 0:
            self.mouthIndex = self.mouthMoveImageList.pop(0)
        mouthImg = self.mouthImages[self.mouthIndex]
        self.screen.blit(mouthImg, (0, 160))
    else:
        self.mouthMoveImageList = None
```

The ***renderMouthShape*** method uses the ***createMouthMoveImageList*** to populate the list of mouth shape indexes. If the text string is empty it will return the index of the neutral mouth shape, otherwise it will find the index of every letter within the string from the ***mshape*** dictionary stored in the ***vars.py*** file in the ***scripts*** directory. If a match cannot be found it will duplicate the previous mouth shape, for example when there is a space between letters or there is no shape associated with a letter.

This naïve method can produce is quite convincing with short phrases but starts to go completely out of sync even after a few sentences.

```
mshape = {'f': 0, 'v': 0, 't': 1, 's': 1, 'z': 1, 'd': 1,
          'c': 1, 'e': 2, 'y': 2, 'x': 2, 'i': 2, 'l': 3,
          'n': 3, 'a': 4, 'u': 5, 'g': 5, 'j': 5, 'q': 5,
          'o': 6, 'h': 6, 'k': 6, 'w': 7, 'r': 7, 'm': 8,
          'b': 8, 'p': 8}


def createMouthMoveImageList(self, text):
    """ Populates the list used for animating the bot's response texts.
    :param text: A text string.
    """
    moveList = []

    if len(text) == 0:
        self.mouthMoveImageList = [7, 7, 7]
        return

    text = text.lower()
    for letter in text:
        if letter in sv.mshape:
            moveList.append(sv.mshape[letter])
            moveList.append(sv.mshape[letter])
        else:
            if len(moveList) > 0:
                # hold the previous mouth shape
                moveList.append(moveList[-1])
            else:
                moveList = [7]

    self.mouthMoveImageList = moveList
```

Implementing conversation logging

All conversations between the chatbot and user are written to a logfile. This functionality can be disabled by setting the ***logConversation*** variable to 0.

```
# Set logConversation to 0 if you do not want to write conversations to file
logConversation = 1
```

When the application is opened, if the MasumiChatLogs folder does not exist in the user's Documents folder it will be created. A new logfile will then be created in the folder. The file's name will contain the current date and time (e.g., ***logfile_01032019_1508.txt***), and this name will be stored in the ***logfilename*** variable for use by the application.

```
if logConversation:
    # Get the path to the log file folder
    newpath = os.path.expanduser("~\Documents")
    newpath = newpath + "\MasumiChatLogs"

    # Create a folder for the log files in the Documents folder
    #  if it does not already exist
    if not os.path.exists(newpath):
        os.makedirs(newpath)

    # Create the path to the logfile
    now = datetime.now()
    logfilename = newpath + now.strftime("\logfile_%d%m%Y_%H%M.txt")
```

When the ***MasumiInterface*** class is initialised, a header will be written to the file containing details of the time and date that the session started. Although these details are contained in the file name, doing this will allow users to paste all their sessions together into a single file before passing it to the therapist for evaluation.

```
def createLogfile(self):
    """ Create a logfile for the session. """
    with open(logfilename, "a") as myfile:
        myfile.write("\n\n--------------------------------------------\n")
        myfile.write("   Session started: ")
        myfile.write(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        myfile.write("\n--------------------------------------------\n")
```

Now that the application is running, the user's input and the chatbot's response can be written to the logfile. This is done from within the ***getBotResponse*** method. The user text will have had punctuation stripped out before being sent to this method, but before being written to the logfile then returned to be displayed onscreen, the chatbot's response string will be tidied up using a regular expression.

```
def getBotResponse(self, inputText):
    """ Sends a text string to the PyAIML kernel for a response
        from the stored knowledge tree.
    :param inputText: A text string.
    :return: A tidied response to the text string.
    """
    ''' Get response and tidy it up. '''
    # Log the user's text
    if logConversation:
        with open(logfilename, "a") as myfile:
            myfile.write("\nUser: ")
            myfile.write(inputText)

    a = self.k.respond(inputText)

    # Tidy up the bot's response
    tidiedResponse = " ".join(a.split())
    tidiedResponse = re.sub(r'\s([?.!"](?:\s|$))', r'\1', tidiedResponse)
    tidiedResponse = tidiedResponse.capitalize()

    # If there was no response, change the subject :D
    if tidiedResponse=="":
        tidiedResponse="Lets talk about something else, please."

    # Log the bot's response
    if logConversation:
        with open(logfilename, "a") as myfile:
            myfile.write("\nBot:  ")
            myfile.write(tidiedResponse)
    return tidiedResponse
```

Logging user button presses is achieved using an event handling method which checks whether the button has been clicked using the ***Button*** class's ***button_was_pressed*** method.

If a press event is detected, the button's caption text will be sent to the ***writeButtonPressToLogFile*** method and written to the logfile.

```
def checkForButtonPress(self, event):
    """ Checks for a feedback button or exit button press.
        Writes feedback button text to the log file.

    :param event: A PyGame event.
    :return: True if the exit button was pressed, otherwise False.
    """
    if self.btn_1.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_1.caption)
    elif self.btn_2.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_2.caption)
    elif self.btn_3.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_3.caption)
    elif self.btn_4.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_4.caption)
    elif self.btn_5.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_5.caption)
    elif self.btn_6.button_was_pressed(event):
        self.writeButtonPressToLogFile(self.btn_6.caption)
    elif self.btn_exit.button_was_pressed(event):
        self.closeApplicationCleanly()
        return True
    return False

def writeButtonPressToLogFile(self, message):
    """ Writes the text of the button pressed to the logfile.

    :param message: The button's caption text.
    """
    """ """
    with open(logfilename, "a") as myfile:
        myfile.write("\n    BUTTON PRESS: ")
        myfile.write(message)
```

This process of writing conversation text and pressed button captions continues throughout the running of the program.

When the application is closed through pressing the ***EXIT*** button, by clicking upon the close window button on the top-right of the interface or by pressing the ***ESCAPE*** key, first the

closeApplicationCleanly method will be called to hide the interface, speak out the closing text, and save the ***PyAIML*** kernel to the brain file.

The ***writeClosingTimeToLogFile*** method will then be called to write a footer to the logfile containing the date and time that the user's session with the application closed.

```
def closeApplicationCleanly(self):
    """ Speaks the closing text.
        Saves any changes to the brain file.
        Writes the closing time to the log file.
        Closes the application
    """
    pygame.display.quit() # Stop displaying the GUI
    engine = CreateObject("SAPI.SpVoice")
    engine.speak(sv.nonBotVoiceTags +
          "Closing the program. We hope you have enjoyed " +
          "using the chat bot application.")
    self.Brain.saveBrain()
    self.writeClosingTimeToLogFile()

def writeClosingTimeToLogFile(self):
    """ Writes the time that the application closed to the logfile. """
    with open(logfilename, "a") as myfile:
        myfile.write("\n\n------------------------------------------\n")
        myfile.write("   Session closed: ")
        myfile.write(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        myfile.write("\n------------------------------------------\n")
```

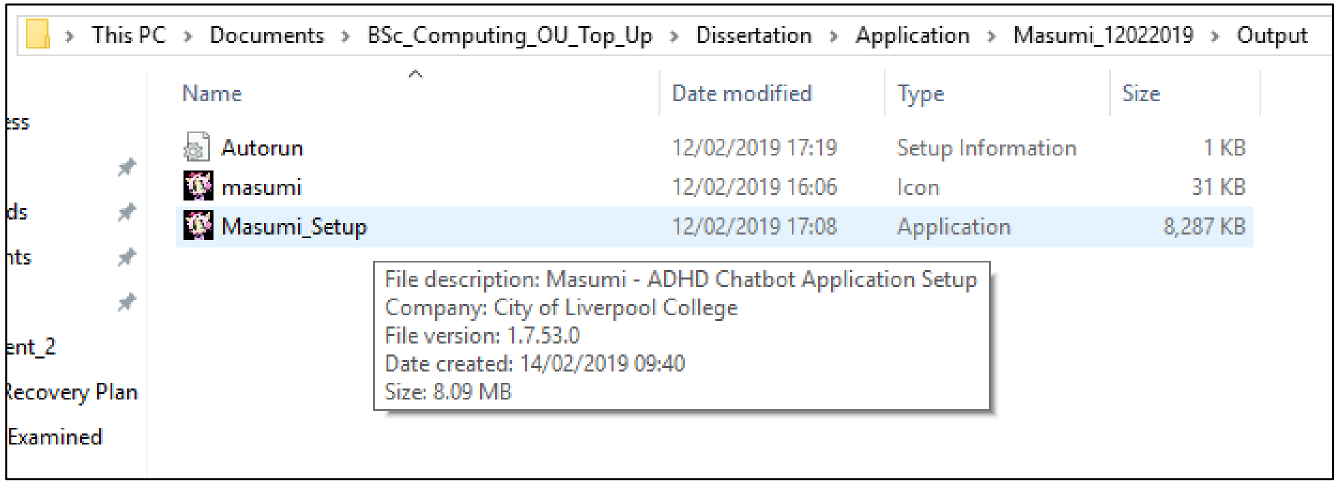
Appendix P: Test plan

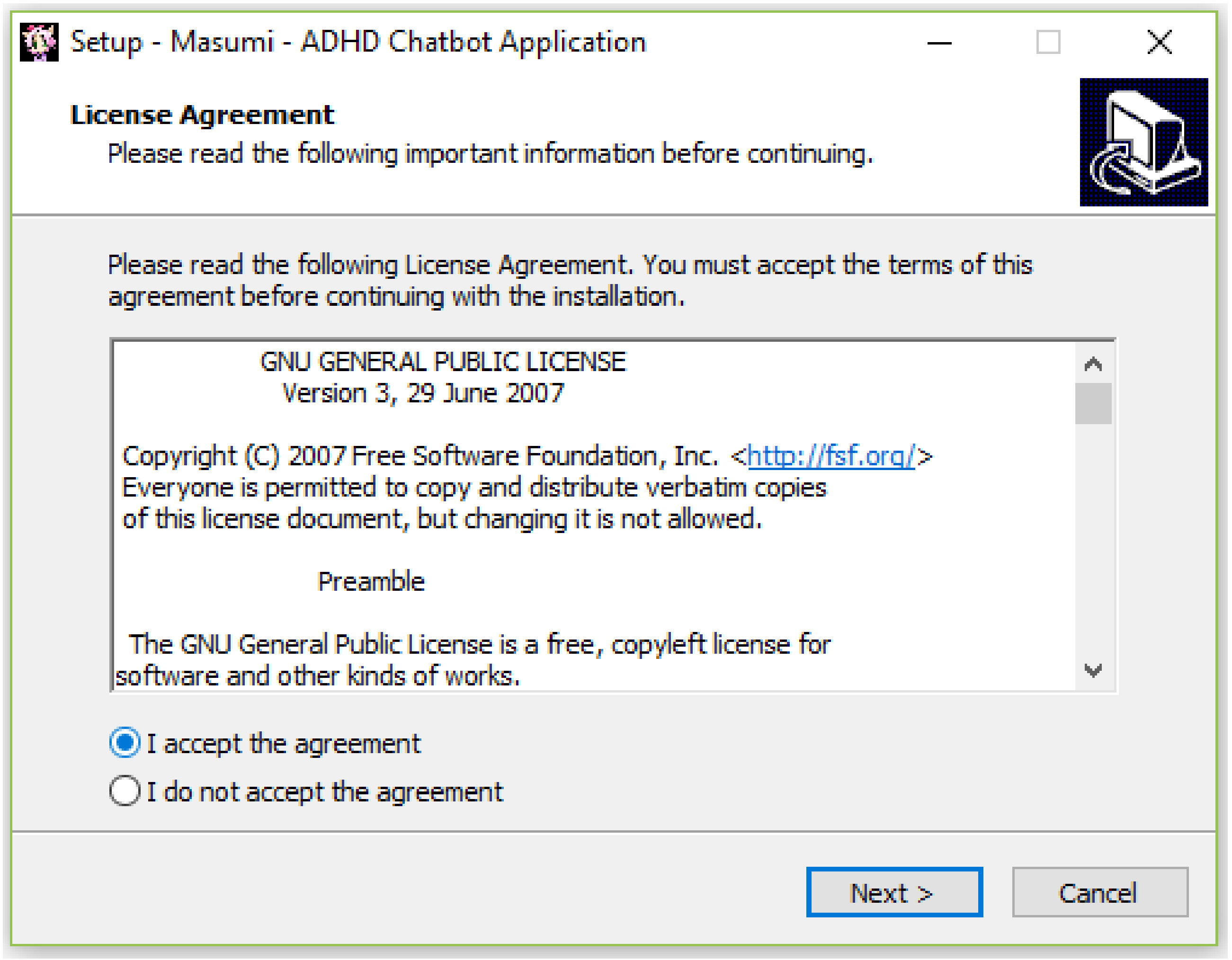
TEST PLAN				
Project Name: Masumi ADHD Chatbot Application **Tester's Name:** Andrew Laing				**Version No:** 1.7.53
TEST	**LAST UPDATED**	**PURPOSE OF TEST**	**INPUT / TEST DATA**	**EXPECTED OUTPUT**
Installation and Uninstall Tests				
1	14/12/2018	Functionality test 1.	Double click upon the '***Masumi_Setup'*** executable file.	Installation dialogue should be shown.
2	14/12/2018	Functionality test 2.	Follow the instructions in the installation dialogue and install the application, choosing to add a shortcut to the desktop.	The application should be installed correctly, an entry created in the '***Start menu***', and a shortcut on the desktop.
3	14/12/2018	Functionality test 3.	Right-click on the application entry in the '***Start menu***' and select '***uninstall***'. When the '***Programs and Features***' screen is shown, select the Masumi application, right-click it and choose '***uninstall***'. Follow the instructions in the uninstall dialogue.	The application will be uninstalled and its files, entry in the '***Start menu***', and shortcut on the desktop will be deleted.
Start-up tests				
4	14/12/2018	Functionality test 1.	Click on the application entry in the '***Start menu***'.	A welcome text should be spoken, and the application will start up correctly showing the interface.

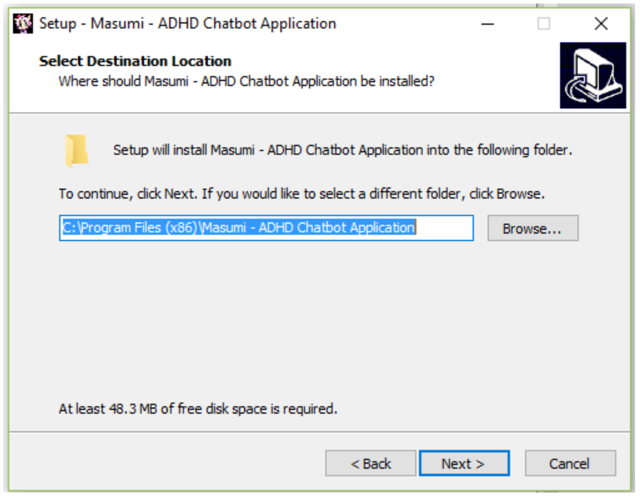
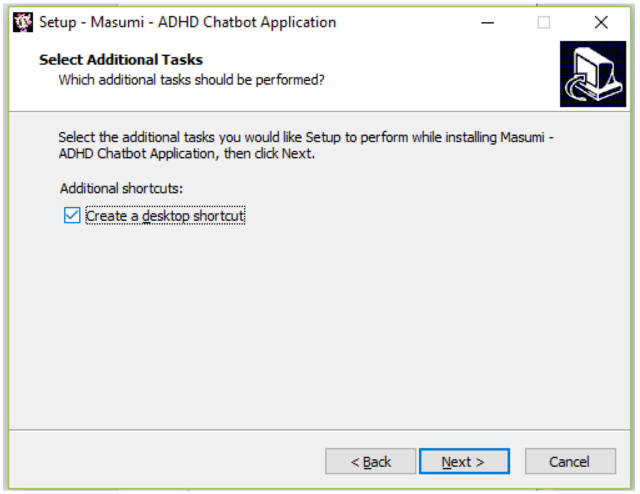
5	14/12/2018	Folder creation test.	After performing the previous test open the '***Documents***' folder.	The '***MasumiChatLogs***' folder should have been created in the user's '***Documents***' folder.
6	14/12/2018	File creation test.	After performing the previous test open the '***MasumiChatLogs***' folder.	A log file should have been created, named with the current date and time.
Interface Layout Tests				
7	14/12/2018	Visual inspection 1.	Examine all texts on the interface	Texts will be readable with no spelling mistakes and should make sense.
8	14/12/2018	Visual inspection 2.	Examine all graphic elements on the interface.	Graphical elements will be well placed and not obscure the texts.
Animation and Text-to-Speech Tests				
9	14/12/2018	Animation Functionality test 1.	Start the application and look at the chatbot character for several minutes.	The chatbot should blink, move its head, and after a while make mouth movements to get the user's attention.
10	14/12/2018	Animation Functionality test 2.	Type the word '***Hello***' into the application and watch the chatbot.	The chatbot's response to '***Hello***' will appear on the screen and the bot's mouth will move to speak out the words.
11	14/12/2018	Text-to-Speech Functionality test.	Listen to the bot's response to the previous test.	The bot should speak out the words that are displayed as its answer to '***Hello***'.
12	14/12/2018	Text resize test.	Type the question '***What is relativity***' into the application and look at the reply.	The text should be resized to fit into the interface.
Text Entry Tests				
13	14/12/2018	Functionality test 1.	With the application focused press each of the letter keys and all the number keys.	The letters and numbers pressed should appear on the screen.
14	14/12/2018	Functionality test 2.	After performing the previous test press the delete key.	Characters should be deleted one at a time.

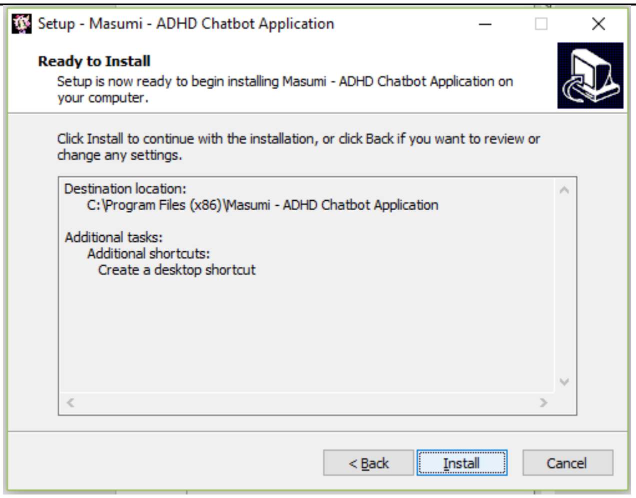
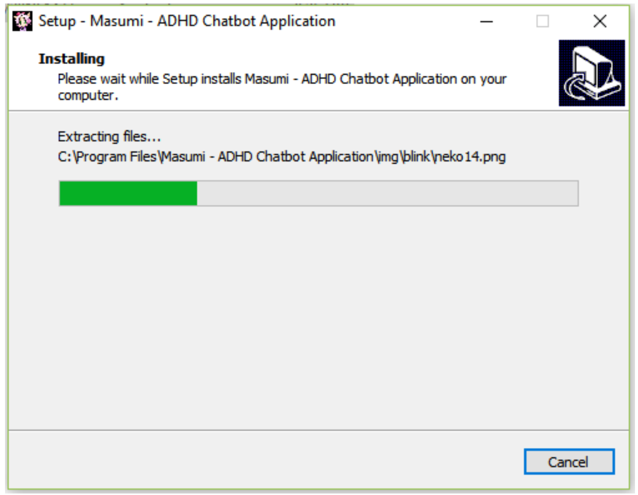
15	14/12/2018	Functionality test 3.	Enter the following texts into the application, pressing enter after each; ***Hello*** ***How are you?*** ***What is your name?*** ***What do you do?***	The chatbot should respond appropriately to each question entered. The text will appear onscreen and will be spoken.
Logging Functionality Tests				
16	14/12/2018	Functionality test 1.	After completing the previous test, close the application and open the log file for the session in the '***MasumiChatLogs***' folder using a text editor.	The logfile should be well laid out and contain the start and end date and times, plus a record of the conversation with the bot.
17	14/12/2018	Functionality test 2.	Restart the application, press each of the buttons in sequence, close the application and open the log file for the session in the '***MasumiChatLogs***' folder using a text editor.	The button presses should have been registered correctly in the log file.
Application Closing Tests				
18	14/12/2018	Functionality test 1.	With the application open, press the '***EXIT***' button.	The closing program text should be spoken, and the application will close.
19	14/12/2018	Functionality test 2.	With the application open, press the '***X***' button in the top-right corner of the application.	The closing program text should be spoken, and the application will close.
20	14/12/2018	Functionality test 3.	With the application open, right-click on the application's icon in the taskbar and choose '***Close Window***'.	The closing program text should be spoken, and the application will close.

Appendix Q: Test log

TEST LOG			
Project Name: Masumi ADHD Chatbot Application **Tester's Name:** Andrew Laing			**Version No:** 1.7.53
TEST	**DATE**	**PASS | FAIL**	**COMMENTS**
Installation and Uninstall Tests			
1	21/02/2019	PASS	 › This PC › Documents › BSc_Computing_OU_Top_Up › Dissertation › Application › Masumi_12022019 › Output Name | Date modified | Type | Size Autorun | 12/02/2019 17:19 | Setup Information | 1 KB masumi | 12/02/2019 16:06 | Icon | 31 KB Masumi_Setup | 12/02/2019 17:08 | Application | 8,287 KB File description: Masumi - ADHD Chatbot Application Setup Company: City of Liverpool College File version: 1.7.53.0 Date created: 14/02/2019 09:40 Size: 8.09 MB *Figure 1.1 – The Masumi_Setup.exe file was double clicked to start the application.*

			 Setup - Masumi - ADHD Chatbot Application **License Agreement** Please read the following important information before continuing. Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation. GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble The GNU General Public License is a free, copyleft license for software and other kinds of works. [✓ I accept the agreement] I do not accept the agreement *Figure 1.2: The Installation application started by asking the user to accept with the licence agreement*

2	21/02/2019	PASS	 Setup - Masumi - ADHD Chatbot Application **Select Destination Location** Where should Masumi - ADHD Chatbot Application be installed? Setup will install Masumi - ADHD Chatbot Application into the following folder. To continue, click Next. If you would like to select a different folder, click Browse. C:\Program Files (x86)\Masumi - ADHD Chatbot Application Browse... At least 48.3 MB of free disk space is required. < Back Next > Cancel *Figure 2.1: The user could then accept the default installation folder or specify another*  Setup - Masumi - ADHD Chatbot Application **Select Additional Tasks** Which additional tasks should be performed? Select the additional tasks you would like Setup to perform while installing Masumi - ADHD Chatbot Application, then click Next. Additional shortcuts: ☑ Create a desktop shortcut < Back Next > Cancel *Figure 2.2: The user could create a desktop shortcut.*

			 Setup - Masumi - ADHD Chatbot Application **Ready to Install** Setup is now ready to begin installing Masumi - ADHD Chatbot Application on your computer. Click Install to continue with the installation, or click Back if you want to review or change any settings. Destination location: C:\Program Files (x86)\Masumi - ADHD Chatbot Application Additional tasks: Additional shortcuts: Create a desktop shortcut *Figure 2.3: Now that all the options were set, installation could begin.*  Setup - Masumi - ADHD Chatbot Application **Installing** Please wait while Setup installs Masumi - ADHD Chatbot Application on your computer. Extracting files... C:\Program Files\Masumi - ADHD Chatbot Application\img\blink\neko14.png *Figure 2.4: Progress during installation reported on a status bar.*

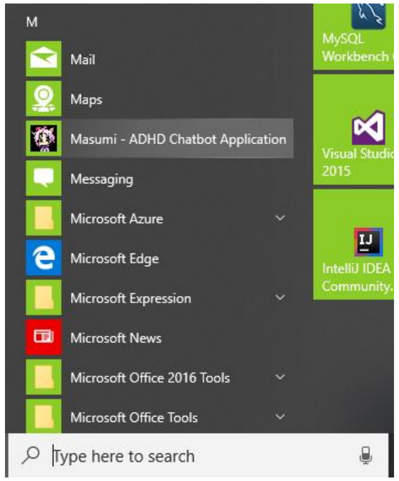
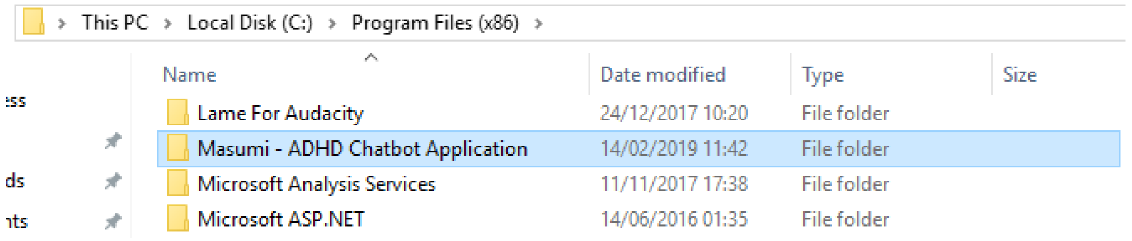


Figure 2.5: After the installation procedure had finished there was an entry for the application in the Windows Start menu....

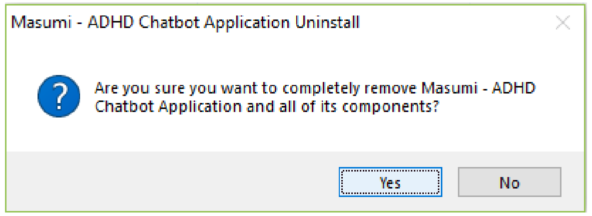
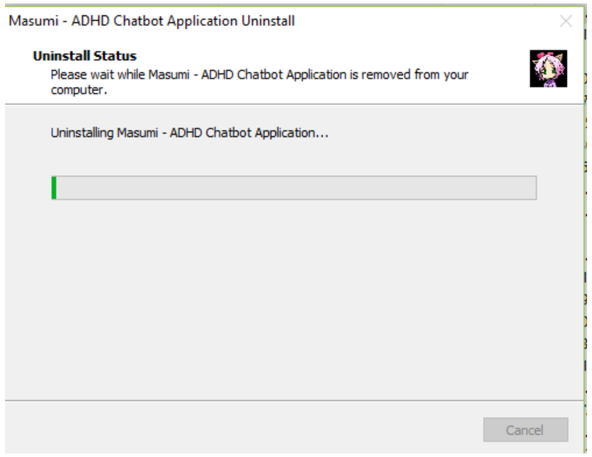
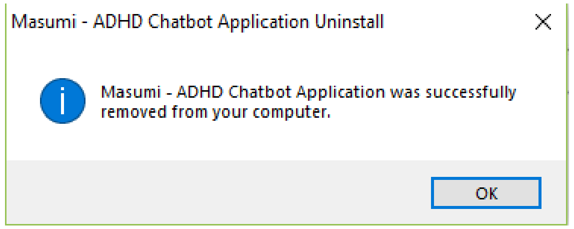
Figure 2.6: ... and a Desktop shortcut.

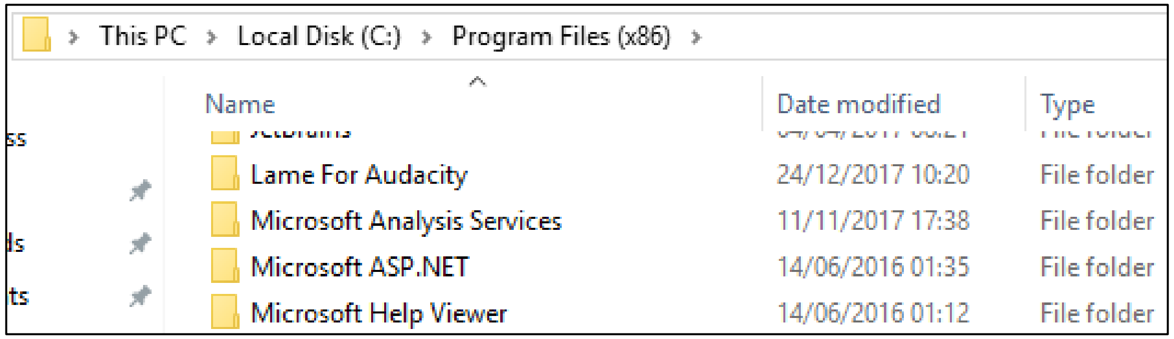
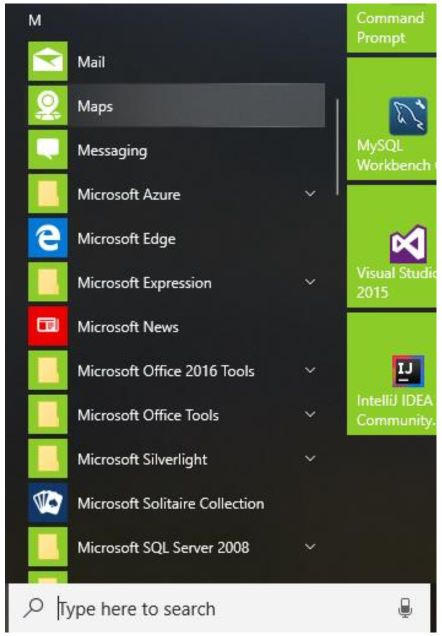


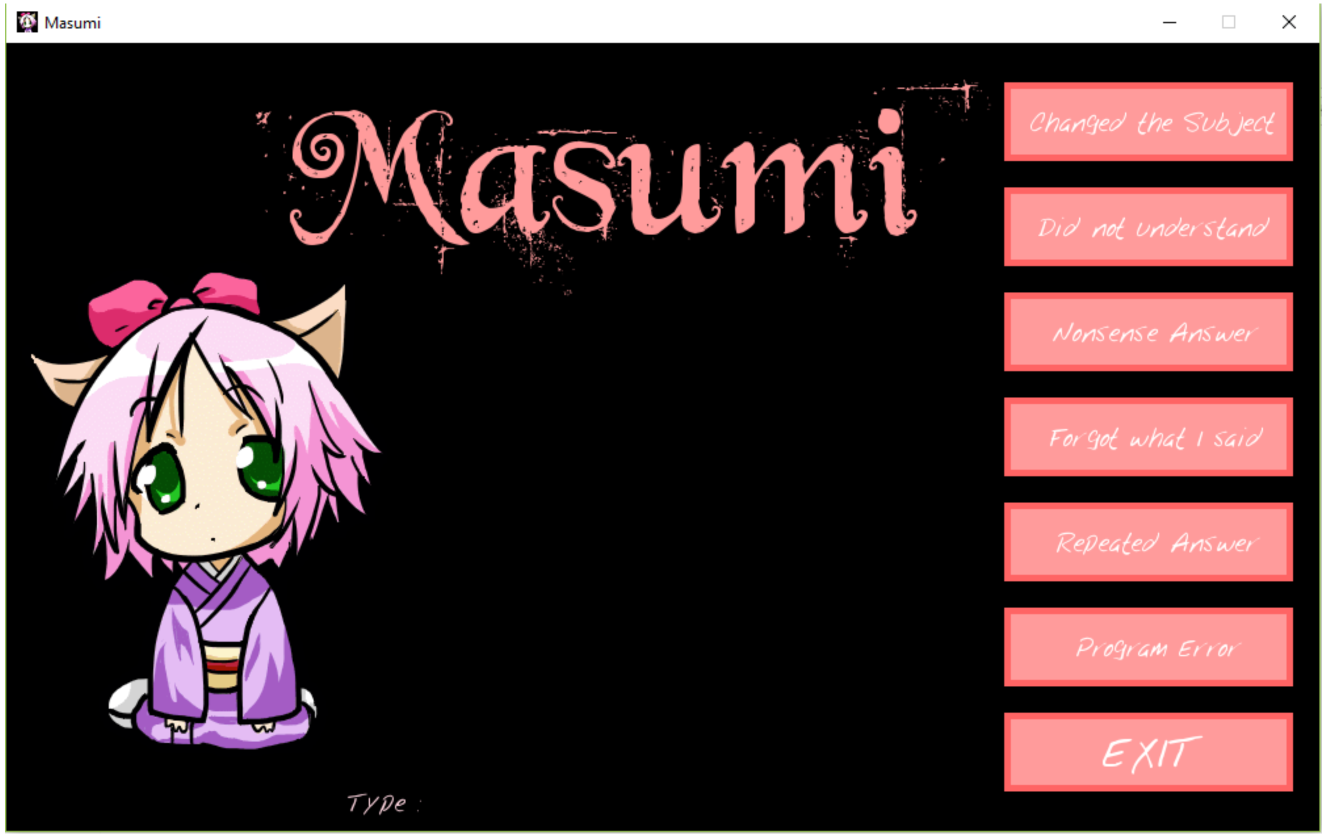
This PC > Local Disk (C:) > Program Files (x86) >

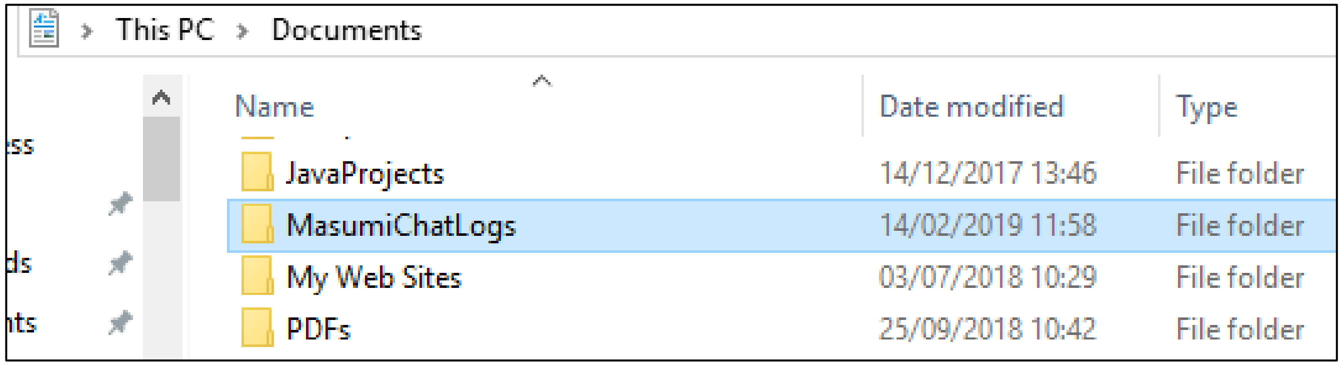
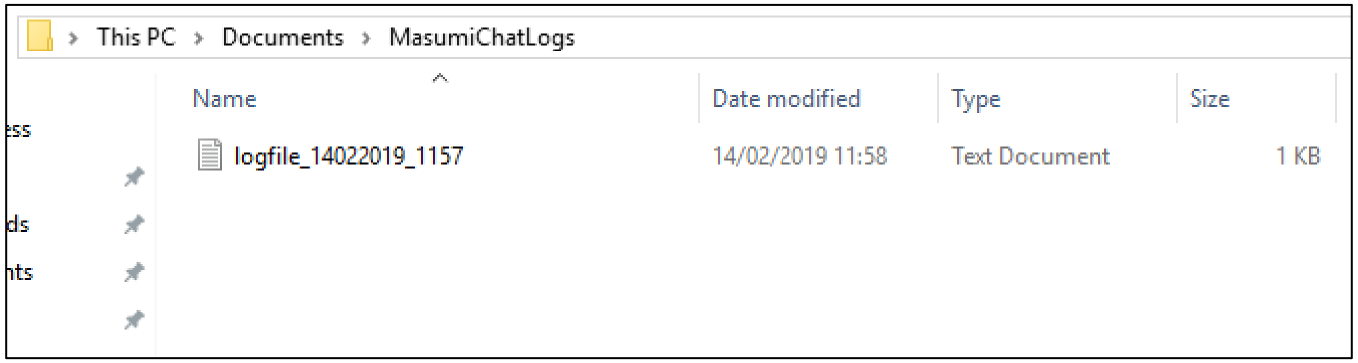
Name	Date modified	Type	Size
Lame For Audacity	24/12/2017 10:20	File folder	
Masumi - ADHD Chatbot Application	14/02/2019 11:42	File folder	
Microsoft Analysis Services	11/11/2017 17:38	File folder	
Microsoft ASP.NET	14/06/2016 01:35	File folder	

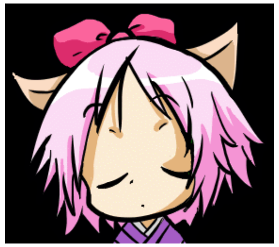
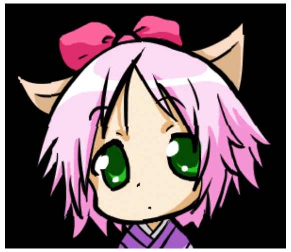
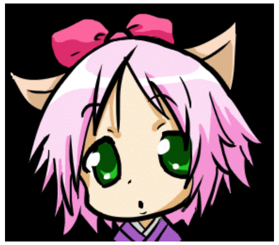
Figure 2.7: The application was installed to the default program files folder.

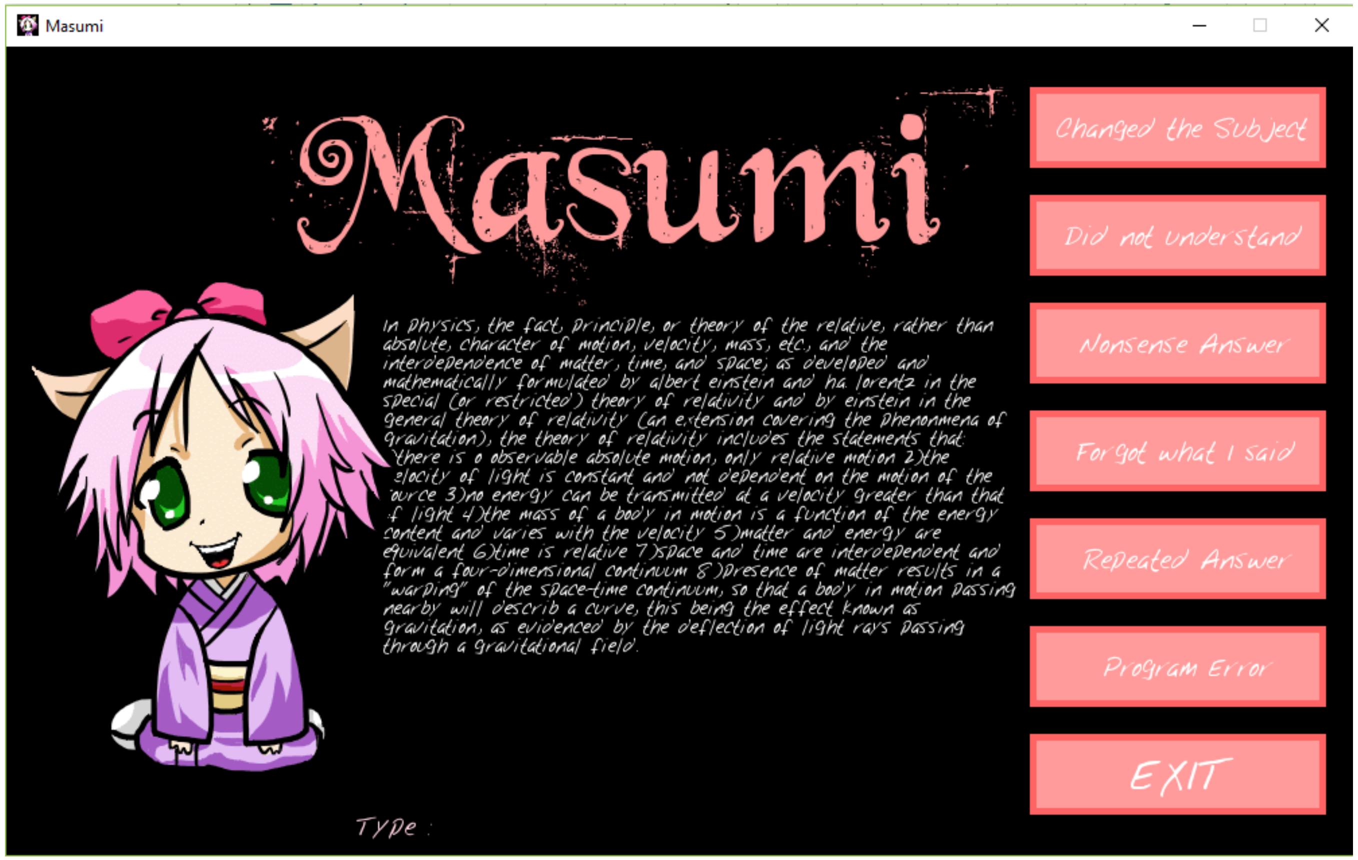
3	21/02/2019	PASS	 Masumi - ADHD Chatbot Application Uninstall Are you sure you want to completely remove Masumi - ADHD Chatbot Application and all of its components? Yes No *Figure 3.1: The Uninstall application first asked for user confirmation.*  Masumi - ADHD Chatbot Application Uninstall **Uninstall Status** Please wait while Masumi - ADHD Chatbot Application is removed from your computer. Uninstalling Masumi - ADHD Chatbot Application... Cancel *Figure 3.2: The user was kept informed of progress during the uninstall procedure.*  Masumi - ADHD Chatbot Application Uninstall Masumi - ADHD Chatbot Application was successfully removed from your computer. OK *Figure 3.3: The application was successfully removed from the computer.*

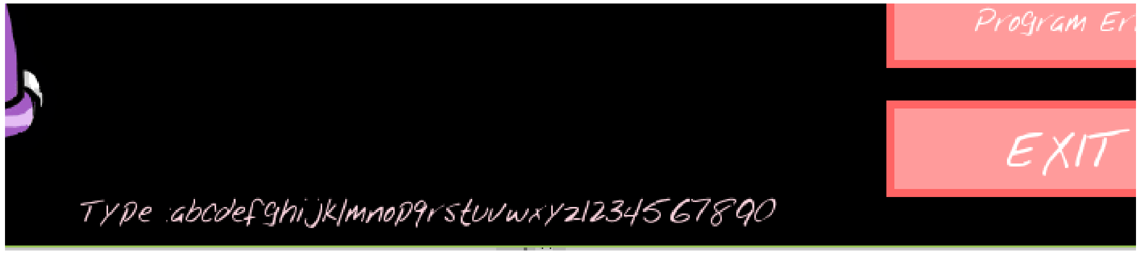
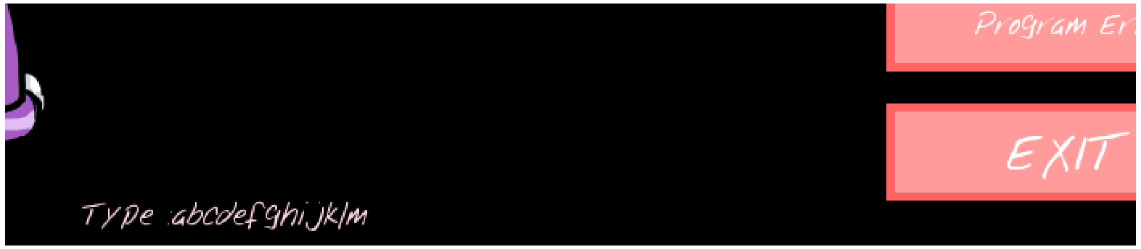
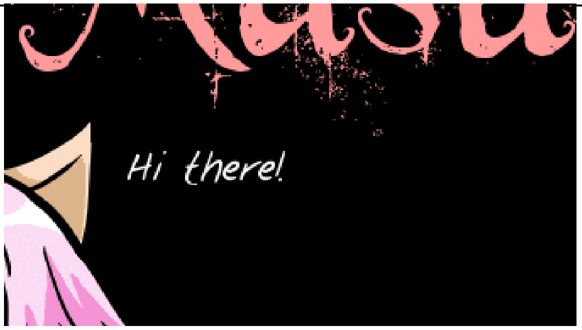
			 This PC > Local Disk (C:) > Program Files (x86) > Name | Date modified | Type Lame For Audacity | 24/12/2017 10:20 | File folder Microsoft Analysis Services | 11/11/2017 17:38 | File folder Microsoft ASP.NET | 14/06/2016 01:35 | File folder Microsoft Help Viewer | 14/06/2016 01:12 | File folder *Figure 3.4: The folder created for the application was successfully removed from the computer.*  M Mail Maps Messaging Microsoft Azure Microsoft Edge Microsoft Expression Microsoft News Microsoft Office 2016 Tools Microsoft Office Tools Microsoft Silverlight Microsoft Solitaire Collection Microsoft SQL Server 2008 Command Prompt MySQL Workbench Visual Studio 2015 IntelliJ IDEA Community Type here to search *Figure 3.5: The Windows Start menu entry and Desktop shortcut were successfully removed from the computer.*

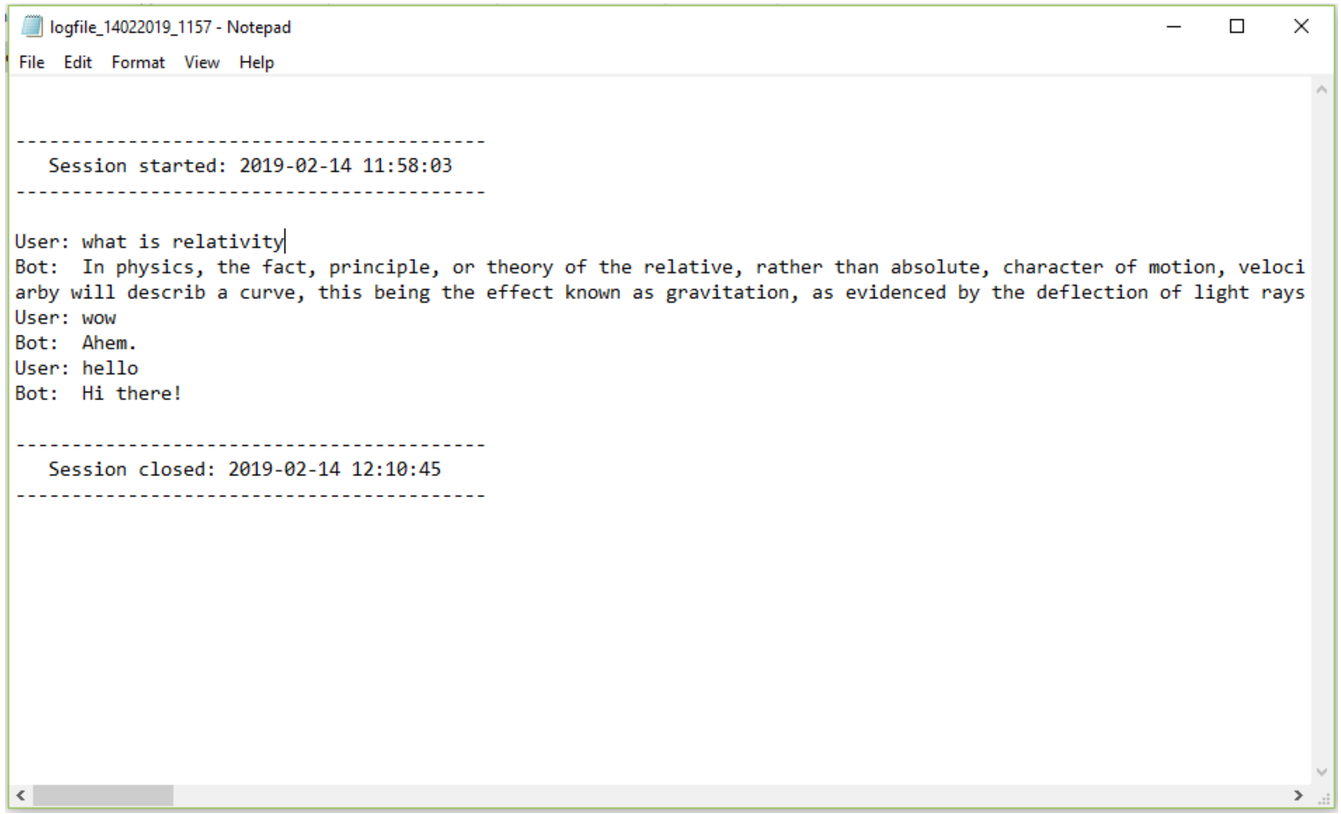
Start-up tests			
4	21/02/2019	PASS	 Masumi Masumi Changed the Subject Did not understand Nonsense Answer Forgot what I said Repeated Answer Program Error EXIT Type : *Figure 4.1: The application started correctly. First the welcome text was spoken, then the interface was displayed.*

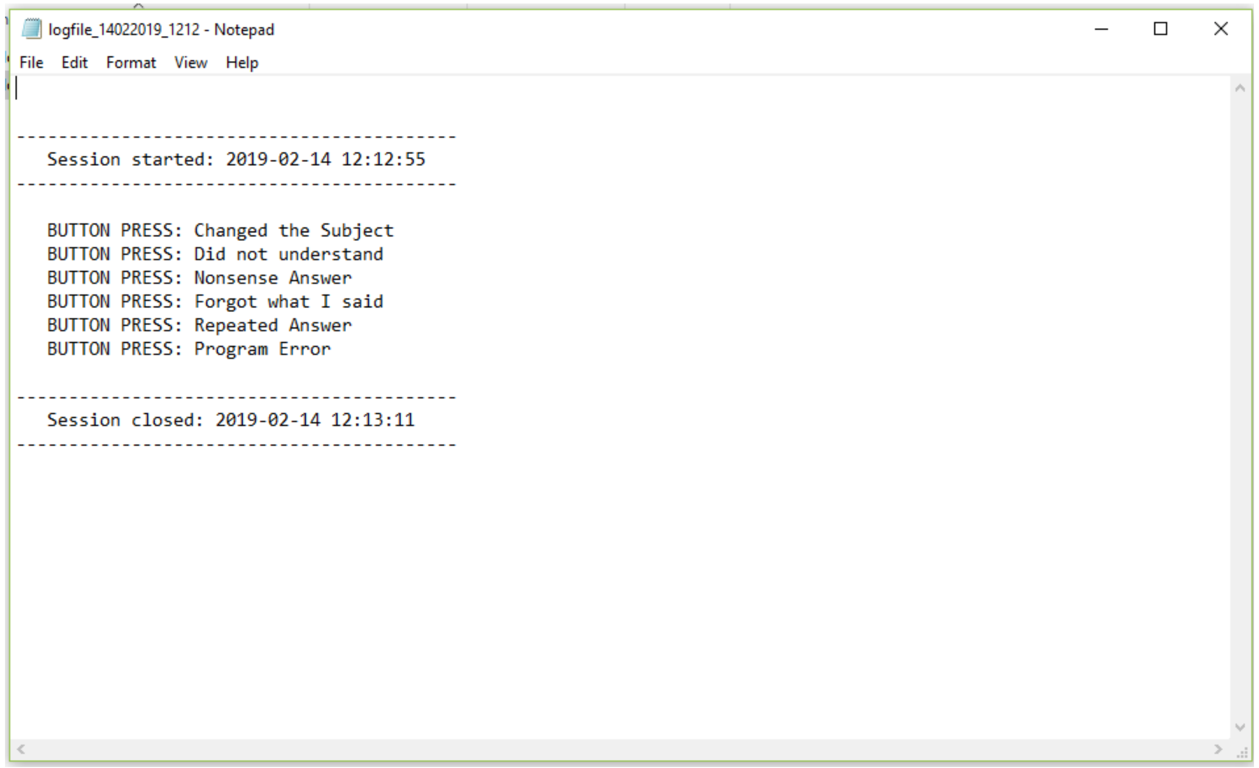
5	21/02/2019	PASS	 This PC › Documents Name | Date modified | Type JavaProjects | 14/12/2017 13:46 | File folder MasumiChatLogs | 14/02/2019 11:58 | File folder My Web Sites | 03/07/2018 10:29 | File folder PDFs | 25/09/2018 10:42 | File folder *Figure 5.1: The MasumiChatLogs folder was created as expected*
6	21/02/2019	PASS	 This PC › Documents › MasumiChatLogs Name | Date modified | Type | Size logfile_14022019_1157 | 14/02/2019 11:58 | Text Document | 1 KB *Figure 6.1: A correctly named log file was created as expected.*
Interface Layout Tests			
7	21/02/2019	PASS	All texts were readable, understandable, and there were no typos. (See Figure 4.1)

8	21/02/2019	PASS	All graphic elements were well placed and did not obscure the texts. (See Figure 4.1 and Figure 12.1)
Animation and Text-to-Speech Tests			
9	21/02/2019	PASS	 *Figure 9.1: The blink animation worked as expected.*  *Figure 9.2: The head movement animation worked as expected.*  *Figure 9.3: The mouth movement animation worked as expected.*

10	21/02/2019	PASS	The chatbot's response appeared on the interface as it should (See figure 15.1)
11	21/02/2019	PASS	The Text-to-Speech functionality worked as it should.
12	21/02/2019	PASS	 Masumi Masumi Changed the Subject Did not understand Nonsense Answer Forgot what I said Repeated Answer Program Error EXIT Type : in physics, the fact, principle, or theory of the relative, rather than absolute, character of motion, velocity, mass, etc., and the interdependence of matter, time, and space; as developed and mathematically formulated by albert einstein and ha. lorentz in the special (or restricted) theory of relativity and by einstein in the general theory of relativity (an extension covering the phenonmena of gravitation), the theory of relativity includes the statements that: 'there is o observable absolute motion, only relative motion 2)the elocity of light is constant and not dependent on the motion of the ource 3)no energy can be transmitted at a velocity greater than that f light 4)the mass of a body in motion is a function of the energy content and varies with the velocity 5)matter and energy are equivalent 6)time is relative 7)space and time are interdependent and form a four-dimensional continuum 8)presence of matter results in a "warping" of the space-time continuum, so that a body in motion passing nearby will describ a curve, this being the effect known as gravitation, as evidenced by the deflection of light rays passing through a gravitational field. *Figure 12.1: The text resized correctly to fit onto the interface.* Note: The avatar slightly obscures some of the text. This can easily be remedied by removing some of the black backgrounds on the transparent png avatar files as the character's hair itself does not obscure the text.

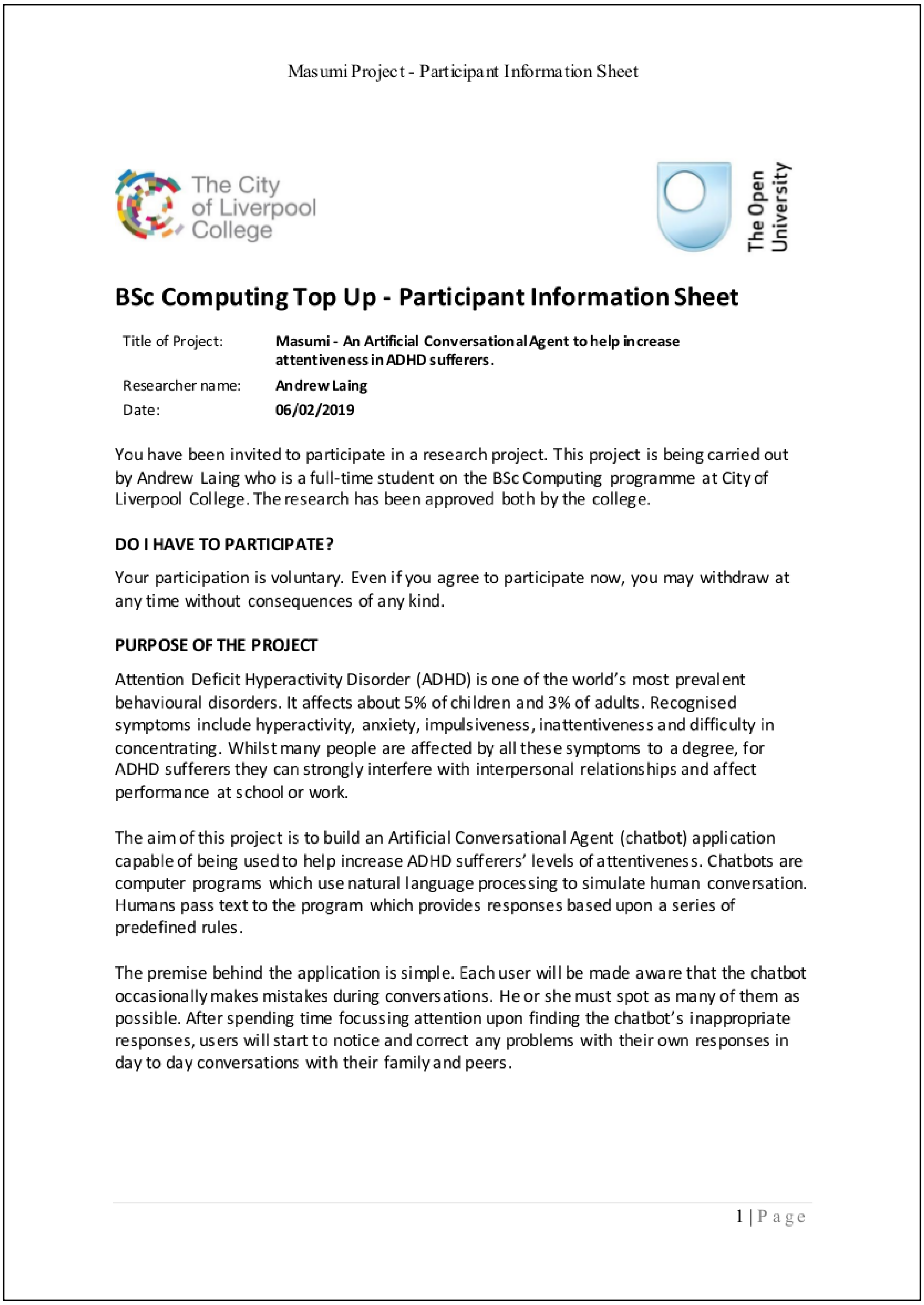
Text Entry Tests			
13	21/02/2019	PASS	 Type abcdefghijklmnopqrstuvwxyz1234567890 *Figure 13.1: When the keys were pressed the characters appeared onscreen as expected.*
14	21/02/2019	PASS	 Type abcdefghijklm *Figure 14.1: The delete button worked as expected.*
15	21/02/2019	PASS	 Hi there! *Figure 15.1: The chatbot responded as expected.*

Logging Functionality Tests			
16	21/02/2019	PASS	 logfile_14022019_1157 - Notepad ``` --- Session started: 2019-02-14 11:58:03 --- User: what is relativity Bot: In physics, the fact, principle, or theory of the relative, rather than absolute, character of motion, veloci arby will describ a curve, this being the effect known as gravitation, as evidenced by the deflection of light rays User: wow Bot: Ahem. User: hello Bot: Hi there! --- Session closed: 2019-02-14 12:10:45 --- ``` *Figure 16.1: The logfile was formatted correctly and contained a record of the conversation.*

17	21/02/2019	PASS	 logfile_14022019_1212 - Notepad
			`------------------------------------------`
			`Session started: 2019-02-14 12:12:55`
			`------------------------------------------`
			`BUTTON PRESS: Changed the Subject`
			`BUTTON PRESS: Did not understand`
			`BUTTON PRESS: Nonsense Answer`
			`BUTTON PRESS: Forgot what I said`
			`BUTTON PRESS: Repeated Answer`
			`BUTTON PRESS: Program Error`
			`------------------------------------------`
			`Session closed: 2019-02-14 12:13:11`
			`------------------------------------------`
			Figure 17.1: The button presses were registered in the logfile as expected.

Application Closing Tests			
18	21/02/2019	PASS	The application closed as it should do.
19	21/02/2019	PASS	The application closed as it should do.
20	21/02/2019	PASS	The application closed as it should do.

Appendix R: User testing participation forms



Masumi Project - Participant Information Sheet

The City of Liverpool College

The Open University

BSc Computing Top Up - Participant Information Sheet

Title of Project:	**Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers.**
Researcher name:	**Andrew Laing**
Date:	**06/02/2019**

You have been invited to participate in a research project. This project is being carried out by Andrew Laing who is a full-time student on the BSc Computing programme at City of Liverpool College. The research has been approved both by the college.

DO I HAVE TO PARTICIPATE?

Your participation is voluntary. Even if you agree to participate now, you may withdraw at any time without consequences of any kind.

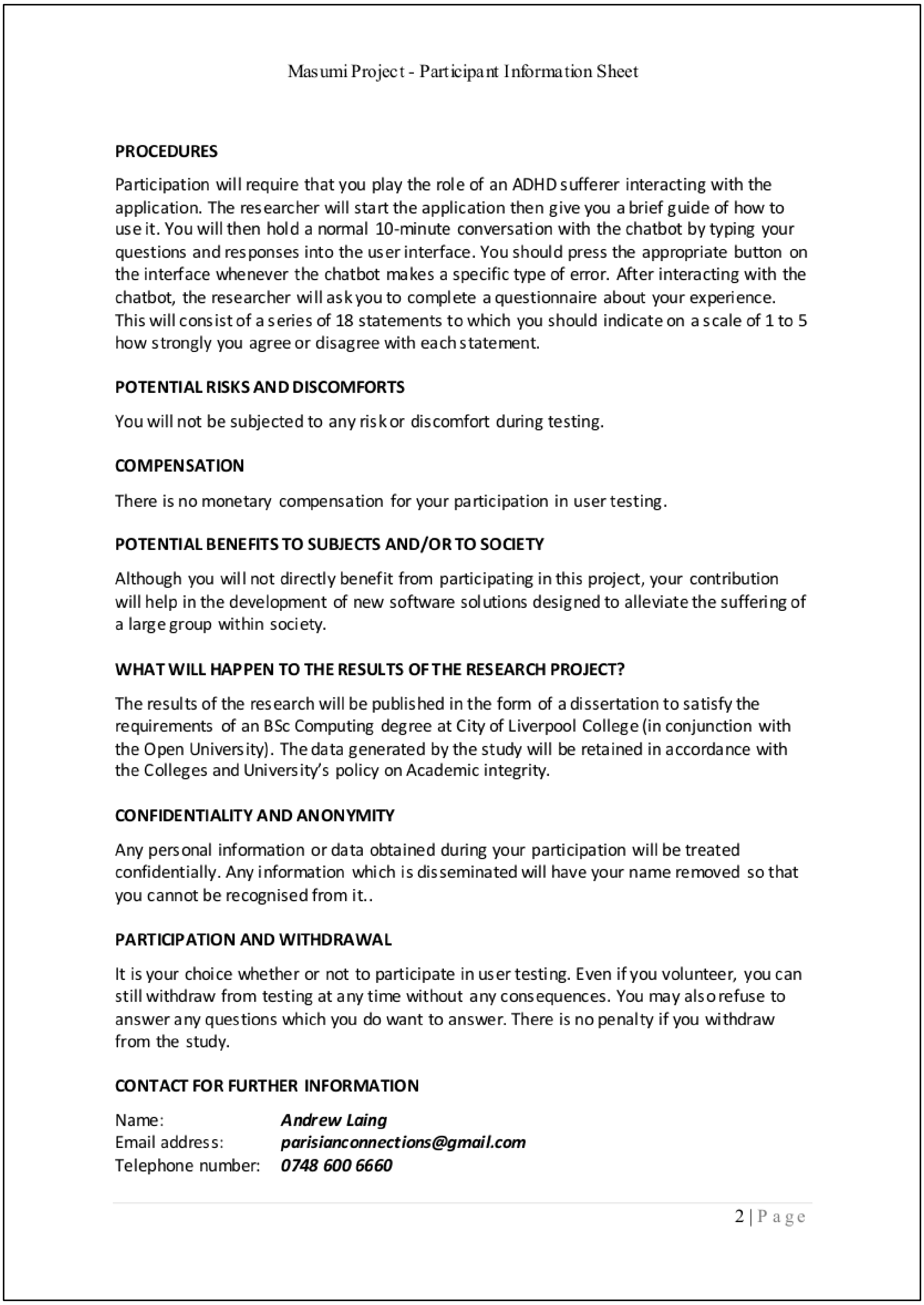
PURPOSE OF THE PROJECT

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioural disorders. It affects about 5% of children and 3% of adults. Recognised symptoms include hyperactivity, anxiety, impulsiveness, inattentiveness and difficulty in concentrating. Whilst many people are affected by all these symptoms to a degree, for ADHD sufferers they can strongly interfere with interpersonal relationships and affect performance at school or work.

The aim of this project is to build an Artificial Conversational Agent (chatbot) application capable of being used to help increase ADHD sufferers' levels of attentiveness. Chatbots are computer programs which use natural language processing to simulate human conversation. Humans pass text to the program which provides responses based upon a series of predefined rules.

The premise behind the application is simple. Each user will be made aware that the chatbot occasionally makes mistakes during conversations. He or she must spot as many of them as possible. After spending time focussing attention upon finding the chatbot's inappropriate responses, users will start to notice and correct any problems with their own responses in day to day conversations with their family and peers.

1 | Page



Masumi Project - Participant Information Sheet

PROCEDURES

Participation will require that you play the role of an ADHD sufferer interacting with the application. The researcher will start the application then give you a brief guide of how to use it. You will then hold a normal 10-minute conversation with the chatbot by typing your questions and responses into the user interface. You should press the appropriate button on the interface whenever the chatbot makes a specific type of error. After interacting with the chatbot, the researcher will ask you to complete a questionnaire about your experience. This will consist of a series of 18 statements to which you should indicate on a scale of 1 to 5 how strongly you agree or disagree with each statement.

POTENTIAL RISKS AND DISCOMFORTS

You will not be subjected to any risk or discomfort during testing.

COMPENSATION

There is no monetary compensation for your participation in user testing.

POTENTIAL BENEFITS TO SUBJECTS AND/OR TO SOCIETY

Although you will not directly benefit from participating in this project, your contribution will help in the development of new software solutions designed to alleviate the suffering of a large group within society.

WHAT WILL HAPPEN TO THE RESULTS OF THE RESEARCH PROJECT?

The results of the research will be published in the form of a dissertation to satisfy the requirements of an BSc Computing degree at City of Liverpool College (in conjunction with the Open University). The data generated by the study will be retained in accordance with the Colleges and University's policy on Academic integrity.

CONFIDENTIALITY AND ANONYMITY

Any personal information or data obtained during your participation will be treated confidentially. Any information which is disseminated will have your name removed so that you cannot be recognised from it..

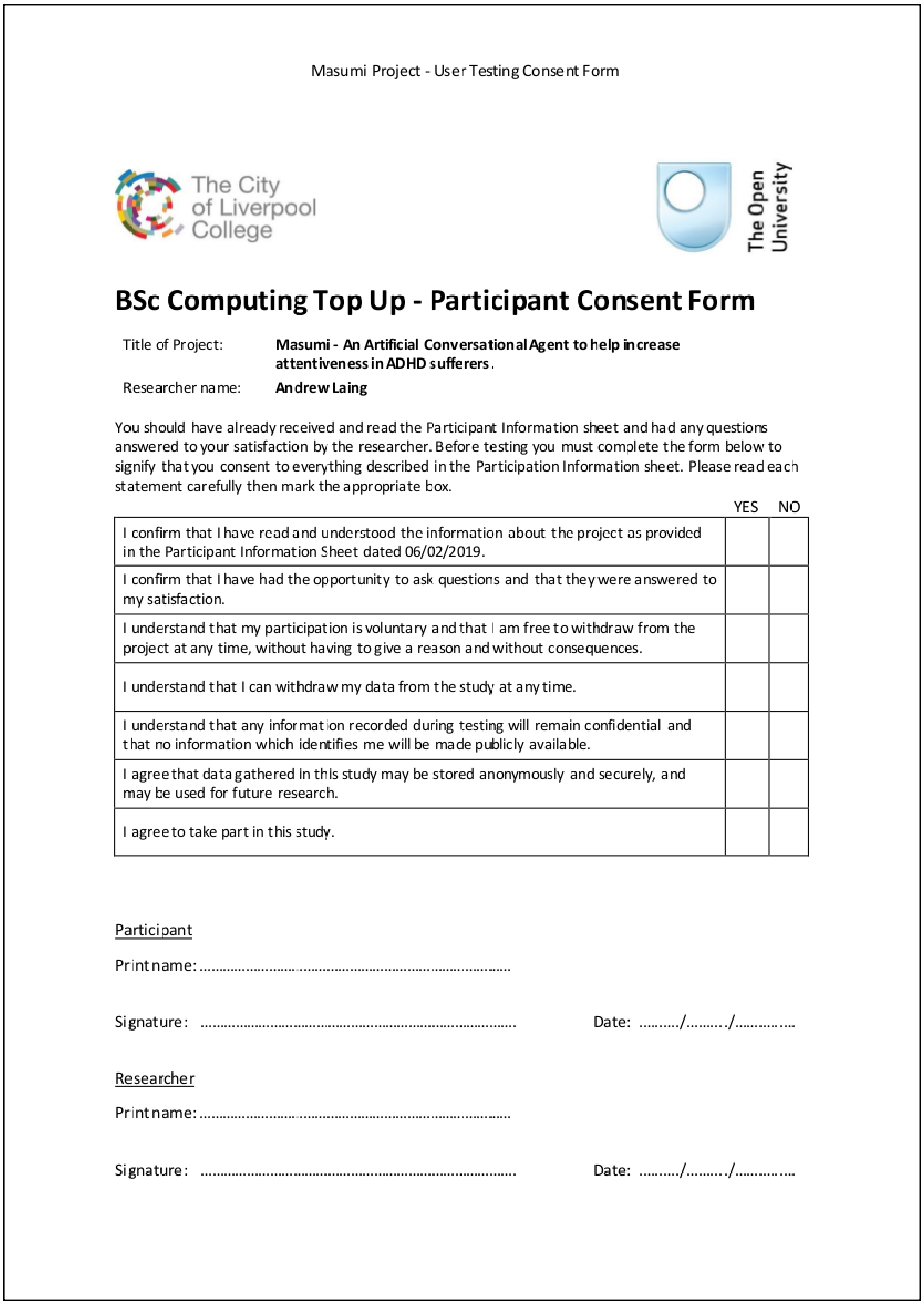
PARTICIPATION AND WITHDRAWAL

It is your choice whether or not to participate in user testing. Even if you volunteer, you can still withdraw from testing at any time without any consequences. You may also refuse to answer any questions which you do want to answer. There is no penalty if you withdraw from the study.

CONTACT FOR FURTHER INFORMATION

Name:	***Andrew Laing***
Email address:	***parisianconnections@gmail.com***
Telephone number:	***0748 600 6660***

2 | Page



Masumi Project - User Testing Consent Form

The City of Liverpool College

The Open University

BSc Computing Top Up - Participant Consent Form

Title of Project:	**Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers.**
Researcher name:	**Andrew Laing**

You should have already received and read the Participant Information sheet and had any questions answered to your satisfaction by the researcher. Before testing you must complete the form below to signify that you consent to everything described in the Participation Information sheet. Please read each statement carefully then mark the appropriate box.

	YES	NO
I confirm that I have read and understood the information about the project as provided in the Participant Information Sheet dated 06/02/2019.		
I confirm that I have had the opportunity to ask questions and that they were answered to my satisfaction.		
I understand that my participation is voluntary and that I am free to withdraw from the project at any time, without having to give a reason and without consequences.		
I understand that I can withdraw my data from the study at any time.		
I understand that any information recorded during testing will remain confidential and that no information which identifies me will be made publicly available.		
I agree that data gathered in this study may be stored anonymously and securely, and may be used for future research.		
I agree to take part in this study.		

Participant

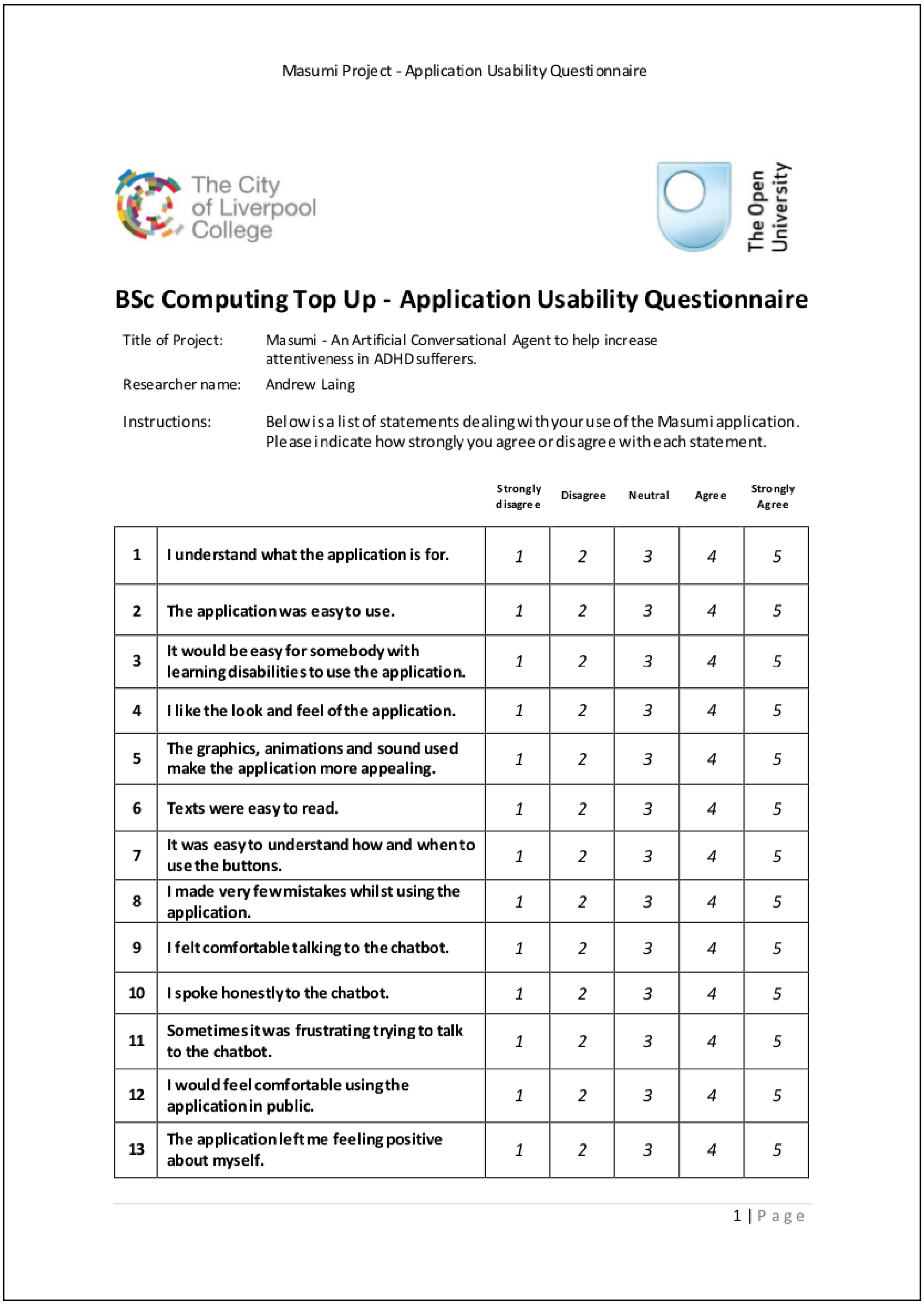
Print name: ………………………………………………………………………

Signature: ……………………………………………………………………… Date: ………/………/……………

Researcher

Print name: ………………………………………………………………………

Signature: ……………………………………………………………………… Date: ………/………/……………



Masumi Project - Application Usability Questionnaire

The City of Liverpool College

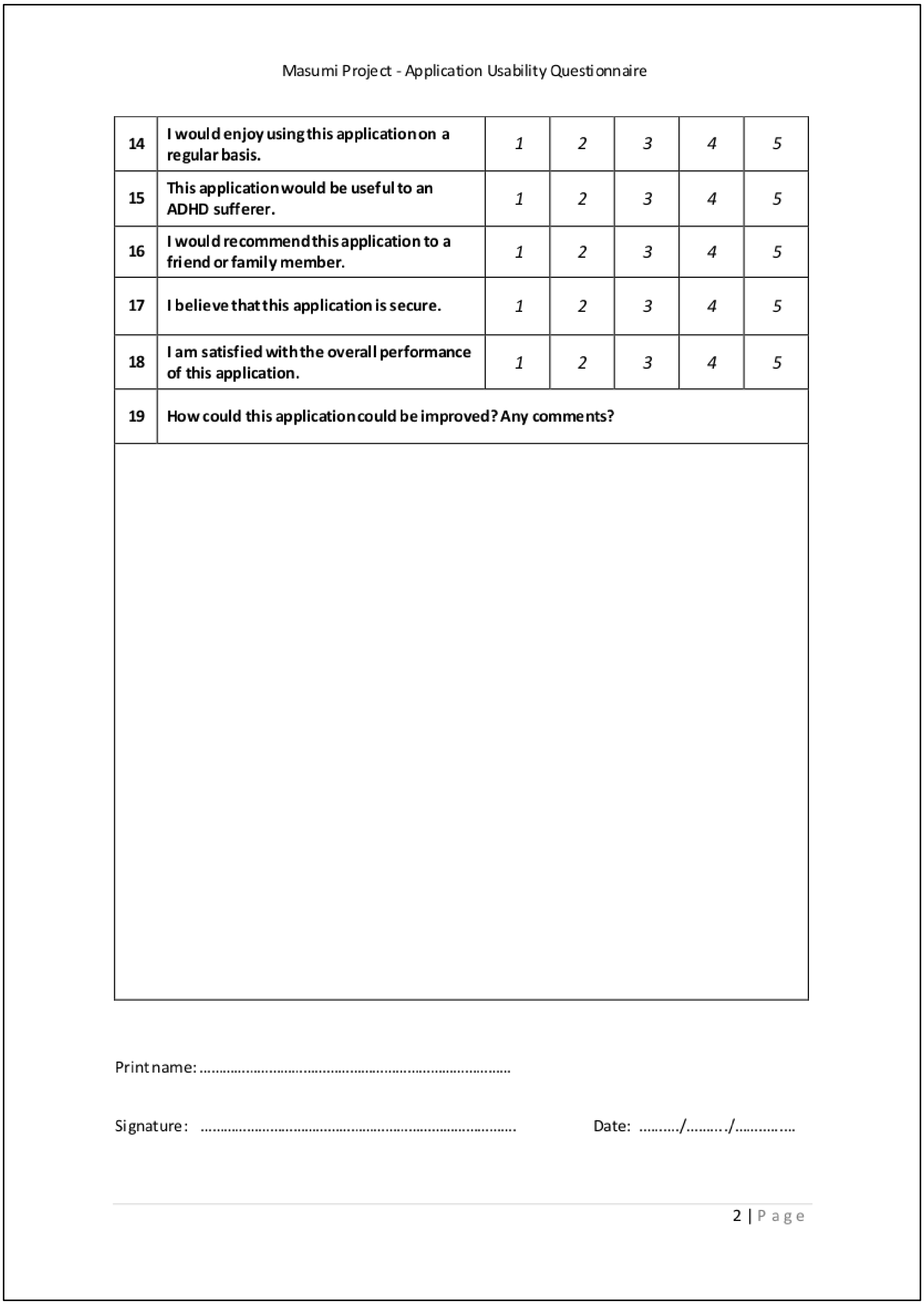
The Open University

BSc Computing Top Up - Application Usability Questionnaire

Title of Project:	Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers.
Researcher name:	Andrew Laing
Instructions:	Below is a list of statements dealing with your use of the Masumi application. Please indicate how strongly you agree or disagree with each statement.

		Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
1	**I understand what the application is for.**	*1*	*2*	*3*	*4*	*5*
2	**The application was easy to use.**	*1*	*2*	*3*	*4*	*5*
3	**It would be easy for somebody with learning disabilities to use the application.**	*1*	*2*	*3*	*4*	*5*
4	**I like the look and feel of the application.**	*1*	*2*	*3*	*4*	*5*
5	**The graphics, animations and sound used make the application more appealing.**	*1*	*2*	*3*	*4*	*5*
6	**Texts were easy to read.**	*1*	*2*	*3*	*4*	*5*
7	**It was easy to understand how and when to use the buttons.**	*1*	*2*	*3*	*4*	*5*
8	**I made very few mistakes whilst using the application.**	*1*	*2*	*3*	*4*	*5*
9	**I felt comfortable talking to the chatbot.**	*1*	*2*	*3*	*4*	*5*
10	**I spoke honestly to the chatbot.**	*1*	*2*	*3*	*4*	*5*
11	**Sometimes it was frustrating trying to talk to the chatbot.**	*1*	*2*	*3*	*4*	*5*
12	**I would feel comfortable using the application in public.**	*1*	*2*	*3*	*4*	*5*
13	**The application left me feeling positive about myself.**	*1*	*2*	*3*	*4*	*5*

1 | Page



Masumi Project - Application Usability Questionnaire

14	**I would enjoy using this application on a regular basis.**	*1*	*2*	*3*	*4*	*5*
15	**This application would be useful to an ADHD sufferer.**	*1*	*2*	*3*	*4*	*5*
16	**I would recommend this application to a friend or family member.**	*1*	*2*	*3*	*4*	*5*
17	**I believe that this application is secure.**	*1*	*2*	*3*	*4*	*5*
18	**I am satisfied with the overall performance of this application.**	*1*	*2*	*3*	*4*	*5*
19	**How could this application could be improved? Any comments?**					

Print name: ……………………………………………………………………………

Signature: …………………………………………………………………………… Date: ………/………/……………

2 | Page

Appendix S: Excerpts from user testing logfiles

Excerpt 1:

```
User: what is your name
Bot:  My name is masumi.
User: how are you
Bot:  Fair to partly cloudy.
  BUTTON PRESS: Did not understand
User: what is your favourate food
Bot:  I require only indian food.
User: why do you like indian food
Bot:  I have a lot of good experience with indian food.
User: what is your favourate film
Bot:  My favorite movie is kill bill. have you seen kill bill?
User: no
Bot:  That's alright.
User: what happens in kill bill
Bot:  Try it.
  BUTTON PRESS: Did not understand
User: why do you like kill bill
Bot:  I have a lot of good experience with kill bill.
User: why do you like chinese food
Bot:  I have a lot of good experience with chinese food.
User: why do you like penguins
Bot:  I have a lot of good experience with penguins.
  BUTTON PRESS: Repeated Answer
```

Excerpt 2:

```
User: what is a lemon
Bot:  I have to process that one for a while.
User: are you sure
Bot:  I am as sure as i can be.
User: how
Bot:  Don't you think i know how?
User: why
Bot:  Does everything have to have an explanation?
User: yes it does
Bot:  Tell me more.. how often?
  BUTTON PRESS: Changed the Subject
```

Excerpt 3:

User: whats for tea
Bot: Masumi eats indian food.
User: like chiken chat
Bot: Interesting comparison.
User: who is your pogrammer
Bot: I don't think i have a pogrammer. i have a brain, a few friends, and some celebrities i like.
User: like who
Bot: Santa claus.
User: peado
Bot: My brain does not have a response for that.
User: suprise surprise
Bot: If you don't pay your exorcist, do you get repossessed?
User: ha ha ha ha ha
Bot: I'm glad you find this amusing.. i'm glad you find this amusing.
User: hi hi
Bot: Hi there!. hi there!
User: you stink
Bot: Thanks for your criticism, human.
User: i like poo
Bot: What else do you like?
User: frogs
Bot: What is your sign?
User: arse
Bot: Is that a sign?
User: ye defo
Bot: Do you have any idea what i am talking about?