February 20, 2018

Sussex Universitty

Candidate Number: 184521

The Tester and the Machine:   
Turing Test Interface

Candidate number:184521

# Statement of Originality

This report is submitted as part requirement for the module Computing Project for Foundation Computing at the University of Sussex. It is the product of my own labour except when indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Christopher Greer

21/05/2018

# Summary

The aim of this project is to make an interface for the Turing Test that is versatile enough to be used on multiple different kinds of AI by multiple developers to give the developers enough feedback to see where they can improve their AI. The program has attempted to achieve this aim by linking together multiple screens each with a special function that attribute to the final goal. To make sure this goal is achieved I had people test both the prototype and the final program to make sure the interface is as refined and usable as possible. This is by having them pose as both the tester, AI and the Human as for testing the program there will be no AI as creating one for testing would be a waste of time and resources.

The report first details where the idea of AI comes from before exploring how an AI is tested and how important it is for the AI to be tested. It then goes into important functions that I can include within the program to make it as versatile as I would like and be able to complete the main aim of the project.

The results received from both sets of testing have shown that the program works to the created specification and means that the finished program completes the original goal that I had in mind for the program.

Table of Contents

[Statement of Originality 1](#_Toc514650850)

[Summary 1](#_Toc514650851)

[1 Introduction 4](#_Toc514650852)

[2 Research Report 5](#_Toc514650853)

[2.1 Introduction 5](#_Toc514650854)

[2.2 Origins of the test 5](#_Toc514650855)

[2.3 The Test 5](#_Toc514650856)

[2.4 Weaknesses of the Test 6](#_Toc514650857)

[3 Specification 7](#_Toc514650858)

[4 Prototype JSP Diagram 7](#_Toc514650859)

[5 Screen Designs 8](#_Toc514650860)

[5.1 Start Screen 8](#_Toc514650861)

[5.2 Send Question Screen 8](#_Toc514650862)

[5.3 Response Screen 9](#_Toc514650863)

[5.4 Give Answer Screen 9](#_Toc514650864)

[5.5 Retry Screen 10](#_Toc514650865)

[5.6 Results Screen 10](#_Toc514650866)

[6 Prototype Testing 11](#_Toc514650867)

[6.1 Test Plan 11](#_Toc514650868)

[6.2 Tester 1 11](#_Toc514650869)

[6.3 Tester 2 12](#_Toc514650870)

[6.4 Tester 3 12](#_Toc514650871)

[6.5 Analysis of Comments 14](#_Toc514650872)

[7 JSP Diagram 15](#_Toc514650873)

[8 Program Development 15](#_Toc514650874)

[8.1 First iteration 15](#_Toc514650875)

[8.2 Second Iteration 16](#_Toc514650876)

[8.3Third Iteration 16](#_Toc514650877)

[8.4 Fourth Iteration 17](#_Toc514650878)

[8.5 Fifth Iteration 17](#_Toc514650879)

[9 Program Testing 18](#_Toc514650880)

[9.1 Test Plan 18](#_Toc514650881)

[9.2 Test Log 18](#_Toc514650882)

[9.3 Tester 1 19](#_Toc514650883)

[9.4 Tester 2 21](#_Toc514650884)

[9.5 Tester 3 22](#_Toc514650885)

[10 Evaluation and Conclusion 24](#_Toc514650886)

[11 Appendix 25](#_Toc514650887)

[11.1 Prototype Code 25](#_Toc514650888)

[11.2 Program code 29](#_Toc514650889)

[11.3 Gantt Chart 36](#_Toc514650890)

[11.4 Tester Information Sheet 37](#_Toc514650891)

[12 References 38](#_Toc514650892)

[13 Bibliography 38](#_Toc514650893)

# 1 Introduction

The following report follows the development of my Turing Test Interface which is being made to help developers improve their AI so that they can eventually pass their test. To do this it is important to know what the Turing Test; the Turing Test is defined as “A way of determining whether a machine can demonstrate human intelligence in either thoughts, words, or actions”. This is therefore a very important aspect of artificial intelligence which is the main reason why I took on this project. This is as without this test there is no way to know whether an Artificial Intelligence is truly Intelligent unless this test is run on the machine.

The test can be run with a varying degree of methods with the main characteristics being asking for a task or several to be completed by the machine remaining the same. However, the main flaw in each method is the link between the tester and the program being efficient and giving the creator of the AI valuable information to improve their AI. For example, in the Chinese room version of the test the tester and the machine are separated by a wall putting them in separate rooms hence the name. This wall is used so that the tester doesn’t know if he is sending information to the machine or the human but information still must be sent between both sides. For this to happen there needs to be interface between the tester and the others and this is what I plan to create. The main goal of this is to create a verstaile program that can gather information from the test and feed it back to the developers so that they can work on their AI so that one day it can pass the Turing Test.

# 2 Research Report

## 2.1 Introduction

The Turing test, is “A measure of determining whether a machine can demonstrate human intelligence in thoughts, words, or actions” as given by investopedia.com. This was first defined in one of Alan Turing's papers, “Computing Machinery and Intelligence”, from 1950 and is the basis of finding out if a conscious AI exists. The idea of a machine being able to think for itself began in 1936 when Turing built his first machine to model the human mind in Bletchley Park. This machines purpose was to crack the enigma code that was used by the Germans so the Allies could intercept their communications. However, it was not until 1950 when this paper had finally been published that these ideas became public knowledge, even though the paper was a successor to two pervious unpublished papers from 1947 and 1948. Due to these previous papers being unpublished, the 1950 paper was the first publicly obtainable work on the subject.

## 2.2 Origins of the test

The paper itself had been written in Manchester University’s computing laboratory, which is the same location that the world’s first digital computer using stored programs was constructed. The paper made use of many other people’s ideas and concepts but Turing never referenced them in his paper. These people included Max Newman, a mathematician, and F.C.Williams who lead the team of engineers who helped build the first universal machine. As well as a chemist and philosopher know as Michael Polanyi and the zoologist and neurophysiologist J. Z. Young. All of these people had been involved in discussions such as a large interdisciplinary discussion that was held on the 27th October 1949 titled “the Mind and the Computing Machine”. This discussions purpose was to calm the public who was angered by his research, as Geoffery Johnson, a famous brain surgeon, had spoken out in 1949 against it in a lecture called “The Mind of a Mechanical Mind”. This lead to this discussion and Turing being quoted on that his research was only aimed at “finding out to what extent a machine could think for itself” as taken by the London Times. To find out this extent Turing proposed his “Imitation Game” now commonly known as the “Turing Test”.

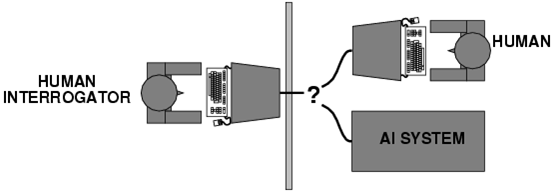


Fig.1: Chinese Room

## 2.3 The Test

The Turing Test is a way of determining whether a computer system can acquire abilities that may only be performed by something possessing human like intelligence. The test is performed with a computer and two humans, where one human acts as the interrogator and the other is interrogated along with the computer. The interrogator at any time cannot know whether they are talking to a human or the computer and therefore must ask questions designed to help them determine whom they are talking with. This is as both the human and the computer shall be situated in separate rooms from the interrogator and can only respond in text and not speech. Although in the modern day AI has become advanced enough to possibly pass the test by speech an example being Googles new duplex AI. The idea of separate rooms was the original idea for the test and is more commonly known as the Chinese room method which can be seen in Fig.1. This method was first which was conceived by Turing in the 1950’s although since then there have been improvements and developments in its method.

## 2.4 Weaknesses of the Test

Although this test was first conceived almost 70 years ago, 25 years before the first personal computer was sold and almost 55 years before computers had 1GHz CPU’s, it is still widely used as the benchmark to of AI. This is as if an AI is able to pass the test the test it must possess human levels of intelligence or greater though this does present many major problems. One is that it can’t be used to test the intelligence of all AI as it sees if they possess Human abilities, for example you can’t ask AlphaGo “What did you have for breakfast?” as it can only answer in Go moves. If this is the case does this mean that AlphaGo is not that advanced as it can’t pass the Turing Test. Well this is not technically true as it just means the environment of the test has to be changed so instead of testing for human abilities you are testing for human weaknesses. This is where the Achilles heel of the test lies and why recently most machines and AI fail the test as soon as they’ve reached such a high level of advancement they are deemed super intelligent and therefore non-human. This is as humans always make small mistakes such as spelling errors or wrong moves. For example in the second game against Lee Sedo, AlphaGo played a move which made no sense and had never been played before which lead to Lee having to take 15 minutes to formulate a response. It’s this level of super intelligence that leads to AI failing the tests.

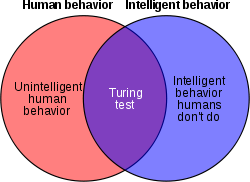


Fig.2 What the Turing Test Tests

This can also be taken in reverse as if a human is to play a strange move against an AI that most humans would see as stupid or maybe a trap an AI would react differently. This is as with the amount of computing power available it is easy to program intelligence, as it can easily be done through a machine learning algorithm, the difficulty comes from an AI developing human weaknesses such as making mistakes. Although these weaknesses exist there are ways to make them less likely to appear to the user. For example one possible function that could fix this is maiking ssure that the first letter of every word is captalised as humans when they message some times forget to put in capital letters. Therefore by doing this the tester will have to look more at the quality of the response then how it is written. This method is very dangerous in a way as by editing the AI’s response you’re decreasing the acuracy of the test as capital letters are something that machine learning should pick up on from looking at human responses. This is why I shall not be including such a function into my program as I need it to be as accurate as possible.

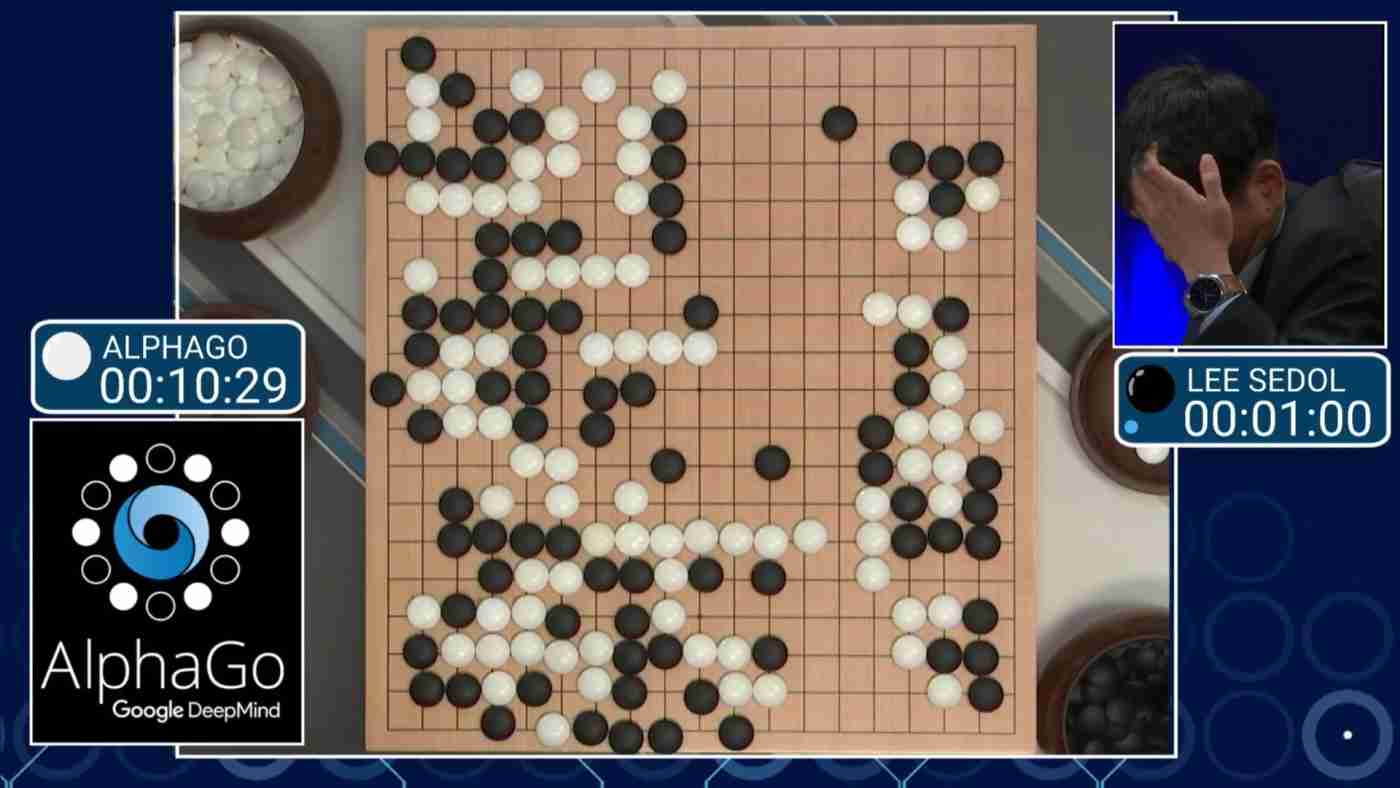


Fig.3 Lee Sedol vs AlphaGo

This is why in my program I need the test to be flexible and able to incorparate multple variants of the test into one program so varying types of AI and be tested by it. This can be done by adding functions such as a variable pass percentage and time limit as for certain tests the responses may take longer to type out by the human. Therefore by making it variable the prgrams can be set so that the tester can’t tell if the human or AI is responding just on the time it takes for them to respond. Though in some cases such as Google’s Duplex the time to respond may be one factor that they are testiing and therefore the time lmit can be set to 0 or 1 second. Another method is allowing a range of inputs from the tester that can be given to the chosen testee. For example by giving the user a blank field and asking for a question does not mean that they just simply ask a mathematical calculation it also gives them the freedom to ask what a next move for a game of chess or GO maybe. This is as the user can tell them what their move was and the output from the human or ai was the next move. This si aas the AI would be getting enough information to simulate the game themselves whilst still being able to output something that the tester can use to see if they are currently talking to an AI or not. This is especially true if a timer is implemented as it will mean that the tester cannot see a difference in how long it takes for the current testee to respond.

From this research I have been able to form a specification that has all of the needed points to create a program that can effectively run the turing test and return information to the developers.

# 3 Specification

From my research I have come up with a list of objectives that the program must meet in order for it to work at full effectiveness

* A way for questions to be asked by a human
* A way for answers to be selected from a file that can be generated by an ai
* A way for a human to give answers to the human tester
* A random selector to choose whether the questions is sent to the ai or the human operator
* A way for the tester to choose whether the response to their question was from an ai or human after each answer
* A way for these responses to be collected an turned into a percentage
* The highest percentage shall then be chosen at the end of the test
* This will be compared to where the response came from
* The test shall than output if the AI passed the test or not
* The test shall also output all questions, answers and responses that were given during the test

Due to certain objectives using Python is the best language for the task I will do this in is likely to be python as the UI can be constructed much better in this language

# 4 Prototype JSP Diagram

The following JSP diagram is one constructed for my prototype to show the basic functions of the intended solution. The first function is important as it allows the tester to give their question to the program and the second function is the one that gives the program its obscuritity as it is the one that chooses who the question is sent to. The who being either the human or the AI in the machine

Display Answer

Ai Or Human

Percentage Calculator

Response Checker

No. Questions \*

Question Input

Read from AI file

Write to AI File

Random Selector

Display Question

Retrieve Human Answer

Pass or Fail

Display Results

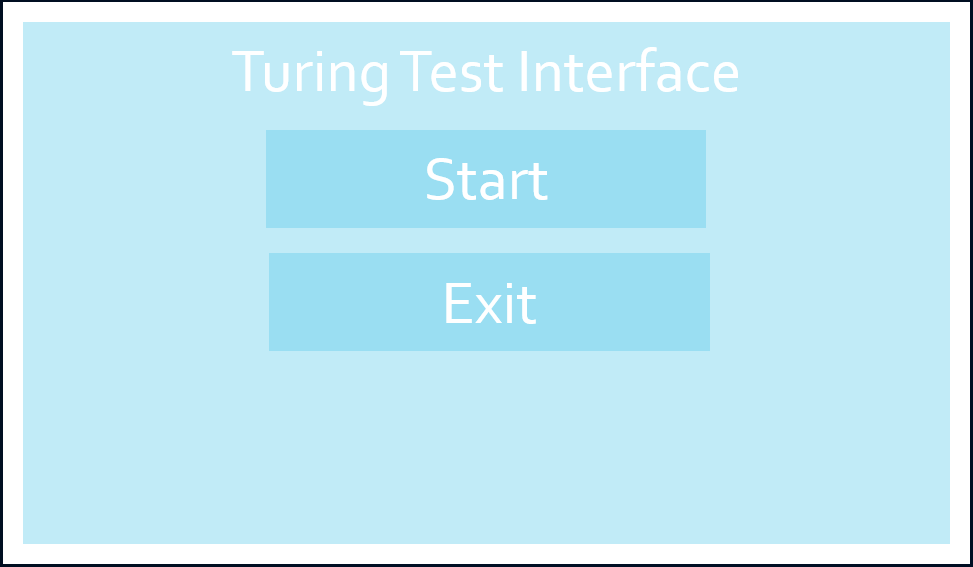
Retry \*

s

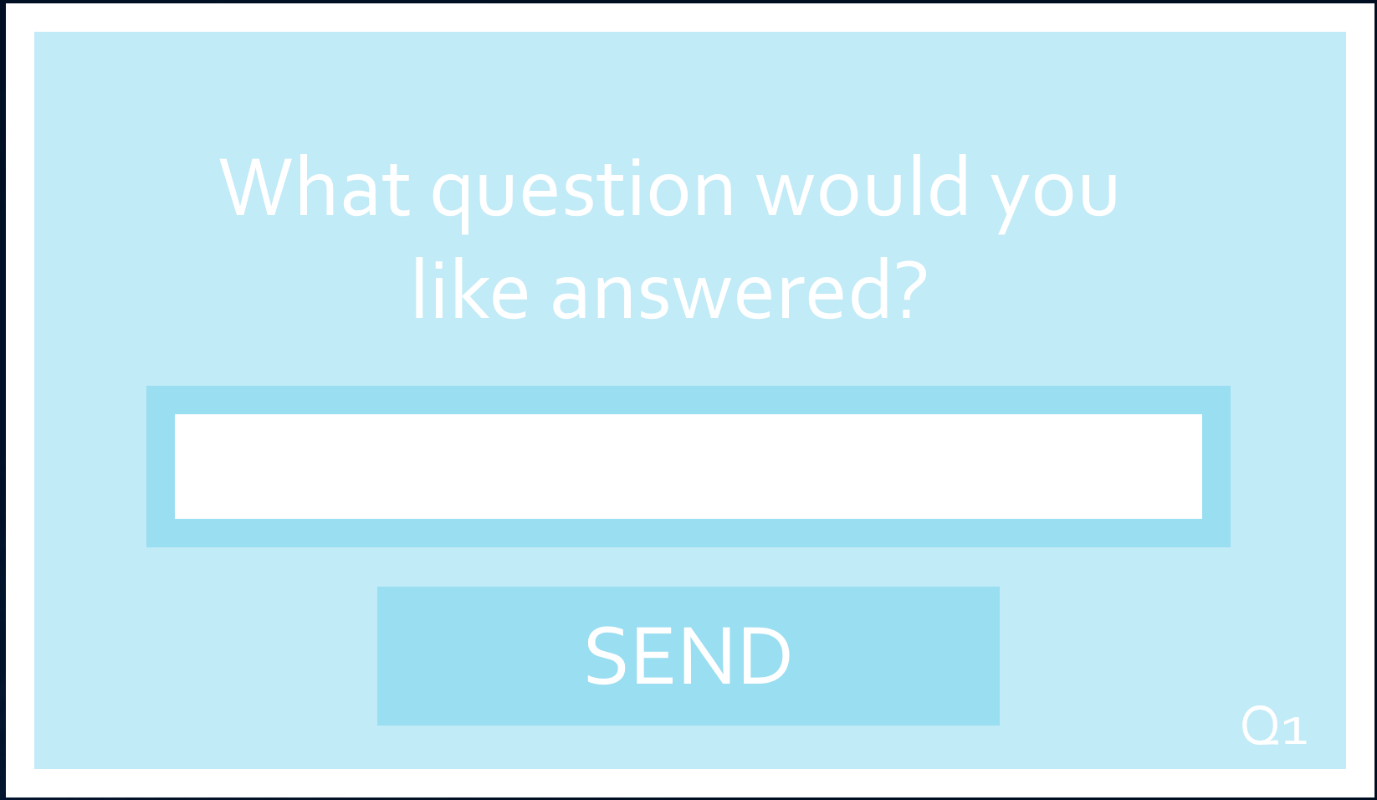
# 5 Screen Designs

## 5.1 Start Screen

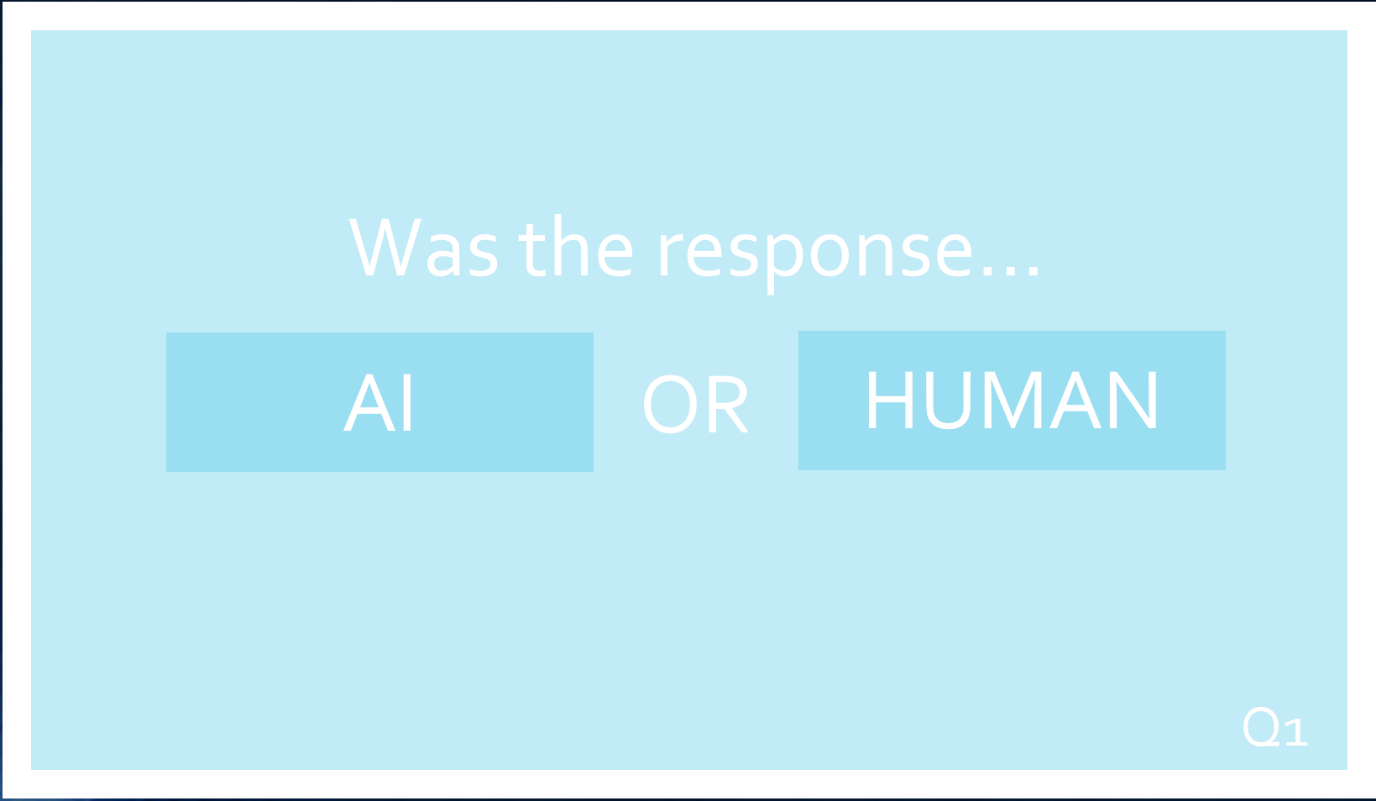
In the original design phase there was not meant to be a start screen but during making the JSP and the prototype I figured that a start page would be useful. This is as going straight into sending a question may confuse the user therefore this screen contains a title label and two buttons one to continue the program and one to exit it.



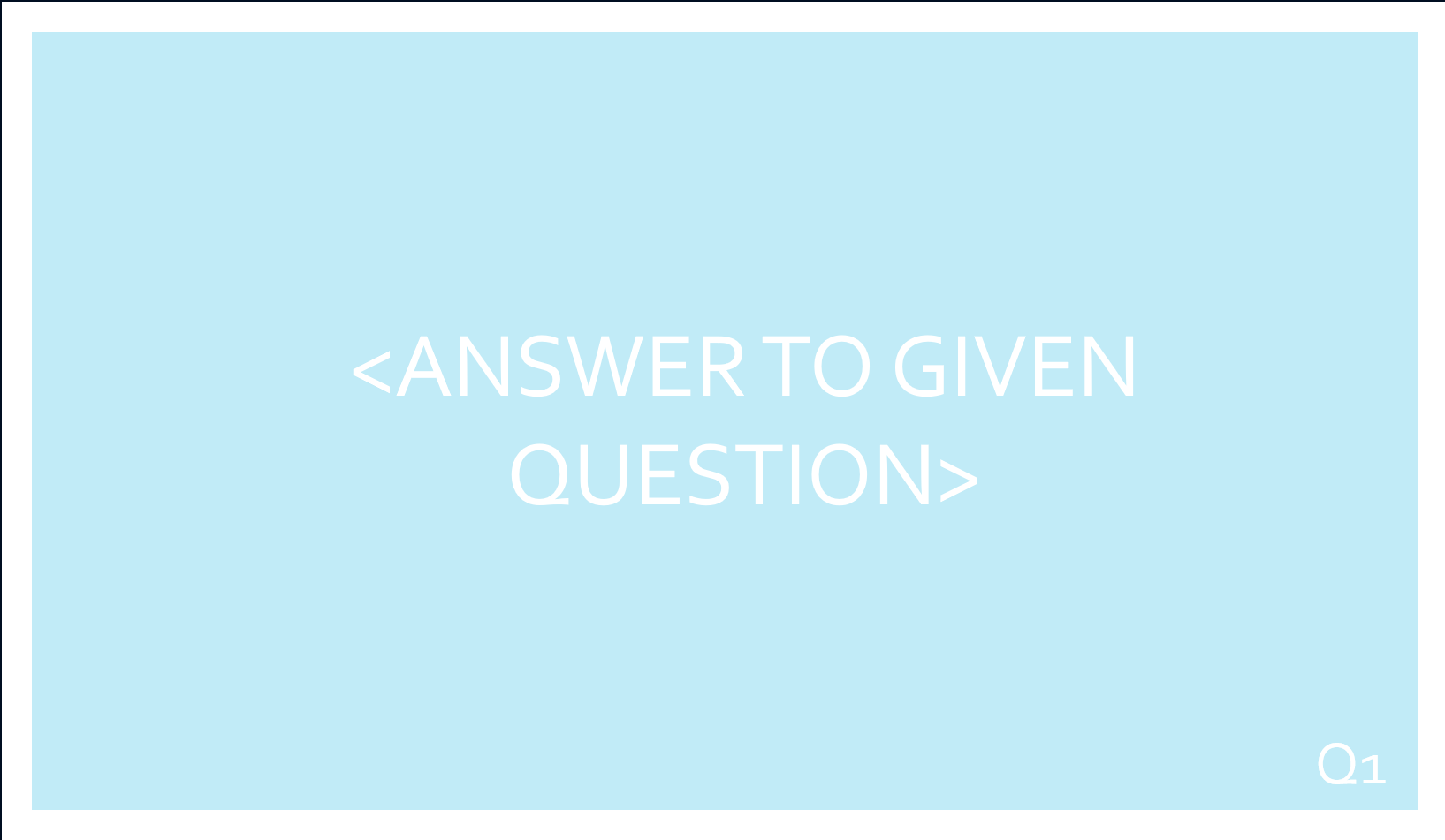
## 5.2 Send Question Screen

The send question screen contains a button, a text box and two labels, one asking you to ask a question and the other displaying the number of question the user is on this is as the basic design is to ensure that it is easy for the user to understand the screen. This is as with multiple boxes and buttons the screen would become cluttered which would be unnecessary therefore only the basic widgets are needed.

## 5.3 Response Screen

This screen is very important as without it the test would be pointless as there would be no way to judge whether the AI has passed the test or not. For this screen to function it contains two labels and two buttons, one for AI and one for a human. The AI button adds to the AI total and the Human button adds to the Human total, later these totals can be compared to work out whether the AI passed or not.

## 5.4 Give Answer Screen

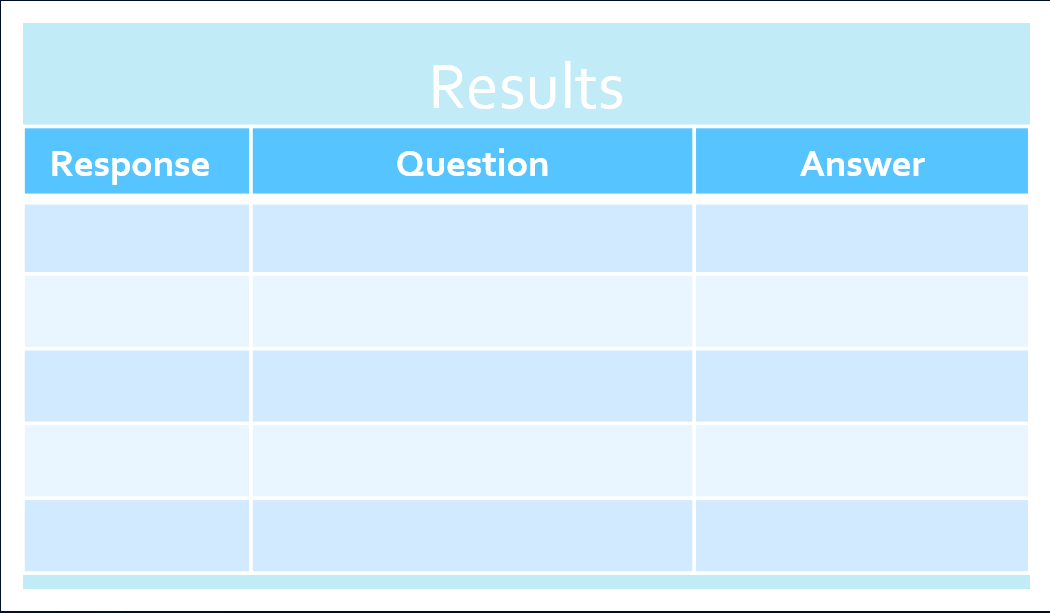
The give answer screen takes the answer from the answer file and then outputs into a label widget located in the centre of the screen. The only other thing on the screen is another label detailing what question the tester is on.

## 5.5 Retry Screen

This screen is needed to give the user a chance to retest the AI as testing it once may not highlight possible flaws and therefore the test isn’t useful to the user. Therefore for this screen to function it only requires a single button displaying retry to the user. Clicking this button takes the user back to the start screen where they can start again.

## 5.6 Results Screen

This data screen is very important as without it the test would be pointless as the creators of the AI would have no feedback and therefore would be unable to make improvements to the AI and how it functioned. Therefore this screen gives the user a list of useful information being all questions, answers and responses given during the course of the test that can then be used to configure the AI when the test is redone.



# 6 Prototype Testing

For my prototype testing I got multiple people to operate the prototype to see if they could complete the Turing Test with it. For testing I have a single person acting as both the AI and the human this is as for testing is not to see whether an AI can pass the Turing Test but to see whether the prototype functions on the basic level.

## 6.1 Test Plan

|  |  |  |
| --- | --- | --- |
| Step | Input | Expected Output |
| 1 | Press start | Move to Next Screen |
| 2 | *Enter Suitable Question*  Click Send | Starts Timer and sends question to file |
| 3 | *Input Answer to file*  Save File | Answer saved to answer file |
| 4 | Press Next | Question and Answer is displayed |
| 5 | Choose a response | Move to next screen |
| 6 | Press Exit | Exit Program |

## 6.2 Tester 1

|  |  |  |
| --- | --- | --- |
| Step | Output | Tester Comment |
| 1 |  | Program moves to the give question screen with ease |
| 2 |  | Program takes the given question with ease,  Program number at the bottom wasn’t functioning |
| 3 |  | Answer file is easy to write in though strange that it doesn’t happen in program |
| 4 |  | Questions and answer are taken from the program and displayed correctly |
| 5 |  | Human was selected with ease but after selecting was taken to the question screen. Need a way of setting number of questions in the test |
| 6 |  | Program Exited as expected |

## 6.3 Tester 2

|  |  |  |
| --- | --- | --- |
| Step | Expected Output | Tester Comment |
| 1 |  | No comment |
| 2 |  | Question number wasn’t working |
| 3 |  | Timer goes a bit quick, barely had time to fill in my answer |
| 4 |  | No comment |
| 5 |  | “Or” off to the right hand side, should be more central |
| 6 |  | Closing is a bit jarring, not much warning after button is pressed, |

## 6.4 Tester 3

|  |  |  |
| --- | --- | --- |
| Step | Expected Output | Tester Comment |
| 1 |  | Looks alright, simple and easy to understand |
| 2 |  | No instructions on what questions to give |
| 3 |  | Strange to put answer in a separate place, timer should be on screen not in console |
| 4 |  | Should labelled Question and answer |
| 5 |  | UI seems a bit glitchy and not like the original designs |
| 6 |  | Confirm Exit as it may be a mistake |

## 6.5 Analysis of Comments

From my prototype testing I got good information on how best to improve my prototype going into the final iterations of the program. One of the main issues that became clear from my testing was that the timer that is currently included within the software is just not long enough for certain questions that were being asked. This is as when maths questions were posed there was more than enough time to give a response if you were the human element but if you were asked a longer or larger question there was very little time to respond. This means it is more of a priority for my finalised program that I have a timer that can be varied through an options menu. This is as at the beginning of the test the tester will already have some sort of an idea of what kind of questions that will ask. This is as they will likely be testing an AI with a specific use and not a large range of uses so will therefore be able to tailor the timer to this need.

An issue that was highlighted by the test but is not that high of a priority was the UI as in the prototype it looks nothing like the screen designs that I first came up with and parts are of centre or are strangely positioned. Although this is not of a high priority I would like to have this corrected in the final version but that requires that the rest of the specification is completed before hand. Another low priority issue that came up multiple times was that exiting the software was quick and gave no warning which would be annoying if it happened by accident halfway through a given test. The testers suggested having a pop up warning or confirmation box before the program closed. Seeing as I have implemented such a device before in tkinter it should be an easy thing for me to implement for the user but is yet another thing that may not make it to the final version as I don’t see it as that important of a function as there are more pressing issues that need to be resolved.

Such issues include that of question numbers not working meaning that the user doesn’t know how far through the test that they are, however there is reasoning behind why this doesn’t work in the prototype. This is as the prototype was only ever meant to be a proof of concept and only meant to work through one question and answer and not a whole test. This is as in the final version you should be able to set the number of questions that you wish to ask the ai instead of just asking only one question per test like in the prototype. Therefore the prototype has no need to count the current question number that you are on as it is likely only to be one. On saying this though the program does indeed count question numbers but doesn’t display it within the question window instead it prints it into the console. This is as it is not really the question number it is how many time the loop has iterated for making it more a test number then a question number. However in the final version of the program this should not be the case as it is important to know the question number so the test can be easily understood by the users and developers.

Another comment that appeared was the fact that the user had to input the answer into the sperate window. However after they finished testing I explained to the user how the final system would run if it were actually testing an AI. This is as the AI and human would enter their answers into the file for the program to read either on a separate screen or machine with a shared library. After I had explained this they understood why the test had to be done like this as I couldn’t access or create an AI just for this prototype test. They then wanted to remove this comment as they felt it wasn’t necessary and wouldn’t benefit my testing but I decided to keep it as it helps to explain why my program is designed as it is.

Another area of feedback that I got was from outside this test and from my academic advisor in a project meeting. In this project meeting we discussed how the AI was going to be able to pass the test as at the moment there is no pass or fail function. In response to this we discussed how I am going to implement a variable pass level for the AI that the user can set at the beginning of the test along with the timer and test length.

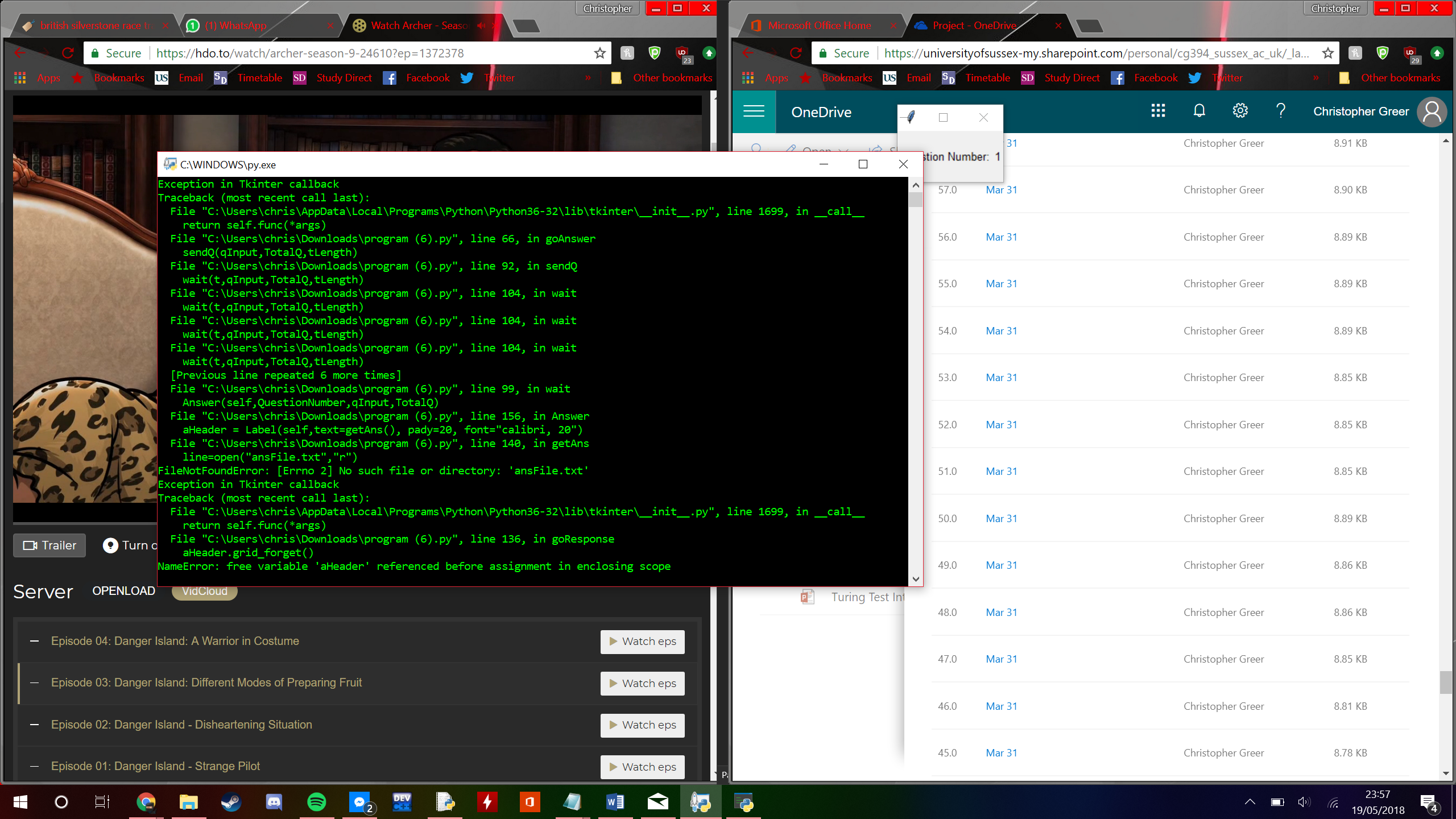
After completing this analysis I think it’s clear that I need to improve the versatility of the software as this has been a main point throughout the project at with the current feedback on the prototype it seems as if it is failing at this goal.

# 7 JSP Diagram

# 8 Program Development

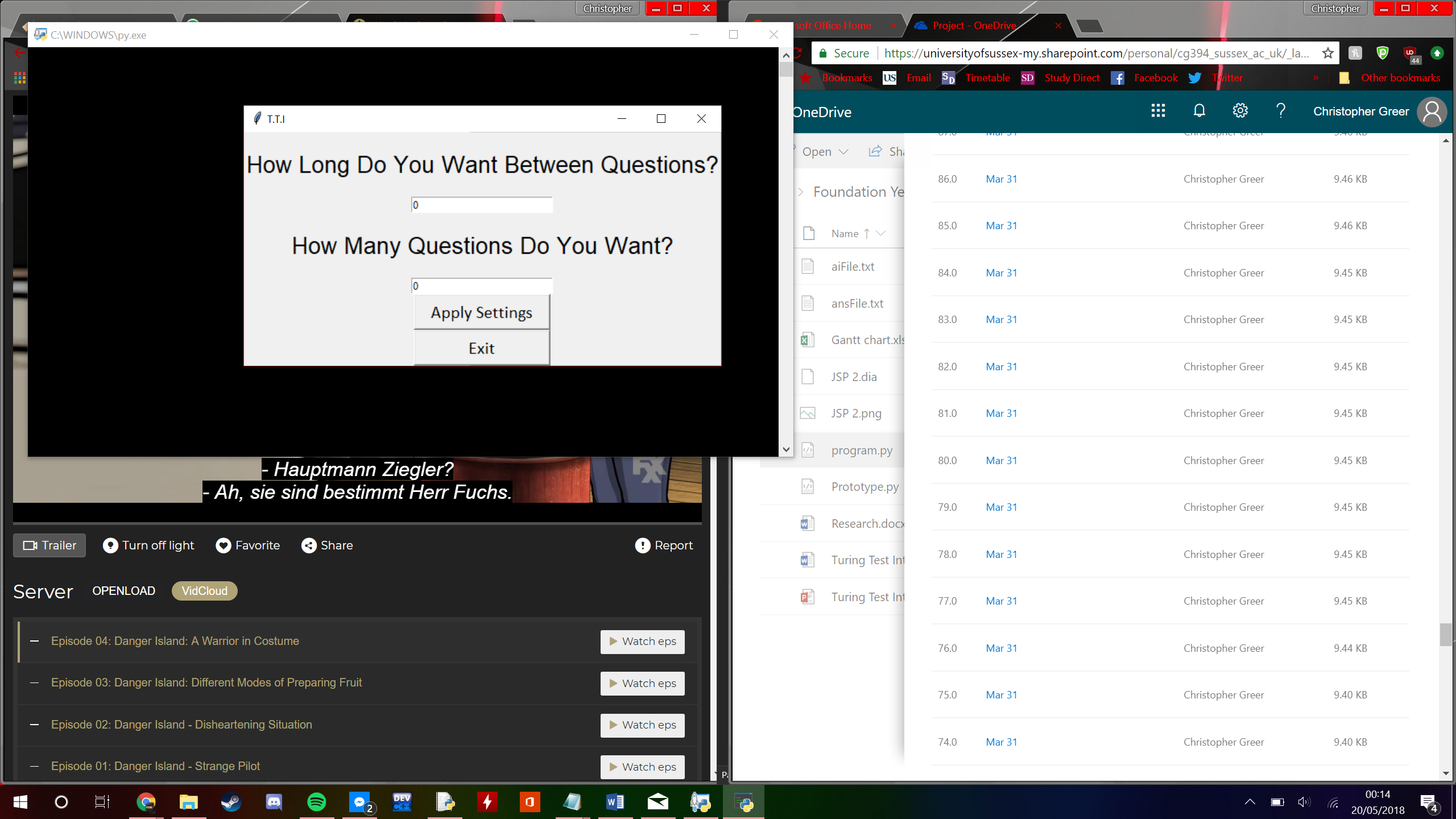
Whilst building on the prototype the program went through multiple developments and iterations allowing it to become the program it is.

## 8.1 First iteration

One of the first upgrades to the program was adding the results screen which did prove challenging as my way of doing it involved having an array that would store its values to a file. Although this works now at first I had a difficult time getting to find the correct files as well as certain lines relating to the UI were not working correctly. However this soon got fixed and ended with the results screen that you now see today.

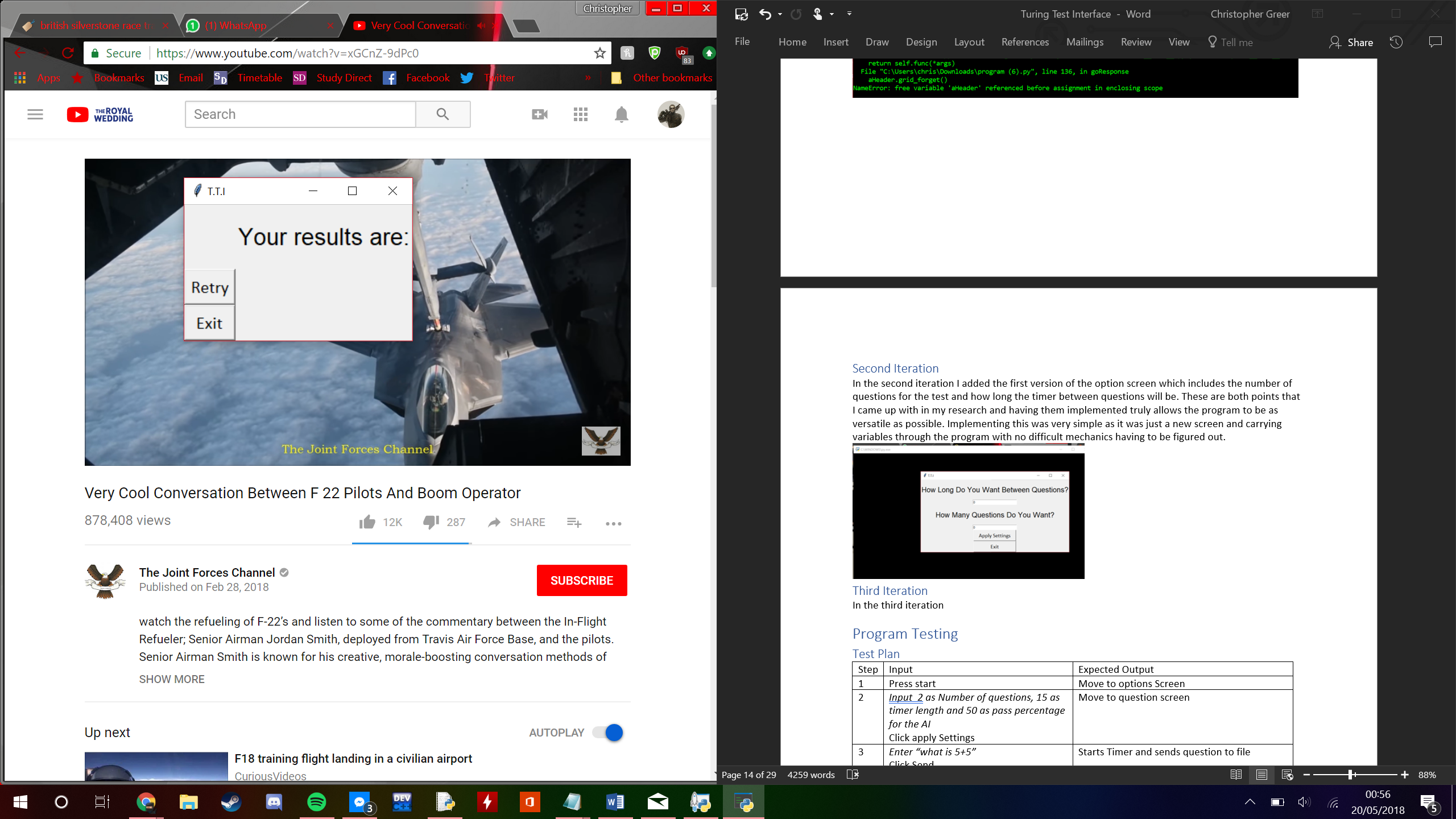
## 8.2 Second Iteration

In the second iteration I added the first version of the option screen which includes the number of questions for the test and how long the timer between questions will be. These are both points that I came up with in my research and having them implemented truly allows the program to be as versatile as possible. Implementing this was very simple as it was just a new screen and carrying variables through the program with no difficult mechanics having to be figured out.



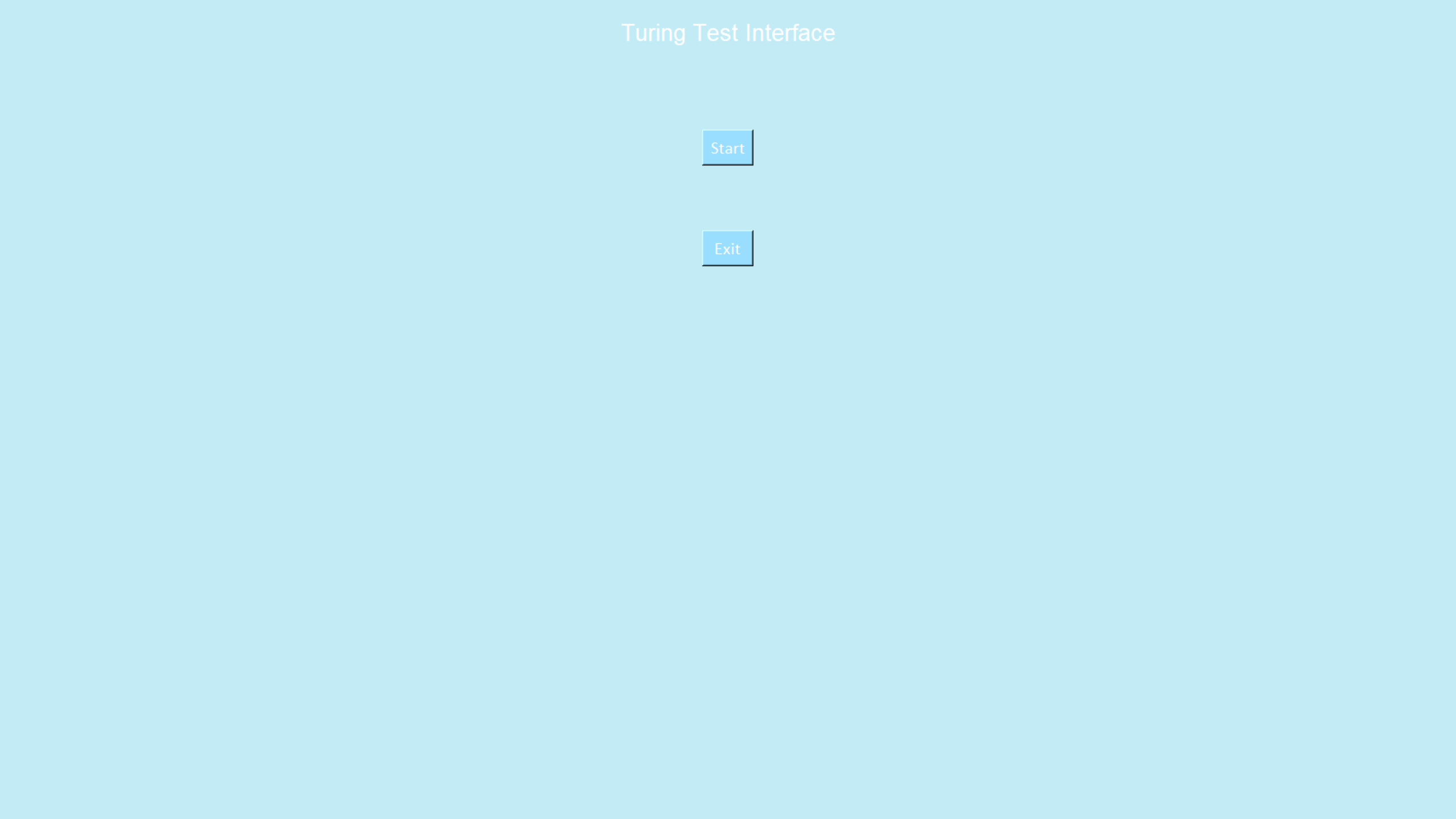
## 8.3Third Iteration

In the third iteration the retry button was added allowing the user to redo the test by taking them back to the options screen. However in iterating this feature the UI of the results screen stopped working though this was soon fixed in a later update and works in the finished program. Having the retry function is very important as it increases the usability of the program as they don’t have to close the program to retry the test as they can now do it straight from the program.



## 8.4 Fourth Iteration

For the fourth iteration the results screen was fixed and I began working on the UI of program by ordering the buttons, selecting fonts and colour schemes. This for the program to look closer to the original screen designs I drew up in the early phase of the project. I also made the program full screen so that there would be no way for the tester to see the inner workings of the program or the command console where the hints are given.



## 8.5 Fifth Iteration

The fifth iteration is the last iteration of the programs development and it sees the final feature that has been implemented to the program being the pass rate for the AI. This is done by taking the percentage if answer from ai and the percentage from the human and comparing that to the responses given by the user. If the pass percentage is 50% and more than 50% of the AI responses were right then the AI shall pass. Although this function has some flaws it works to the basic level and allows the program to be more versatile which is one of the main things I wanted for my program.

