# 9 Program Testing

## 9.1 Test Plan

|  |  |  |
| --- | --- | --- |
| Step | Input | Expected Output |
| 1 | Press start | Move to options Screen |
| 2 | *Input 2 as Number of questions, 15 as timer length and 50 as pass percentage for the AI*  Click apply Settings | Move to question screen |
| 3 | *Enter “what is 5+5”*  Click Send | Starts Timer and sends question to file |
| 4 | *Input Answer to file*  Save File  Wait for timer to elapse | Answer saved to answer file |
| 5 | *Question and Answers are displayed to screen*  Press Next | Question and Answer is displayed |
| 6 | Choose a response | Move to next screen |
| 7 | *Enter “what is 8\*20”*  Click Send | Starts Timer and sends question to file |
| 8 | *Input Answer to file*  Save File  Wait for timer to elapse | Answer saved to answer file |
| 9 | *Question and Answers are displayed to screen*  Press Next | Question and Answer is displayed |
| 10 | Choose a response | Move to next screen |
| 11 | Give the results table | Results table with all inputs |

## 9.2 Test Log

|  |  |  |
| --- | --- | --- |
| Step | Output | Comments |
| 1 |  | Start screen works as expected |
| 2 |  | Values taken as expected |
| 3 |  | Question taken as expected |
| 4 |  | Answer saves itself as expected |
| 5 |  | Question and answer display |
| 6 |  | Response taken |
| 7 |  | Question taken as expected |
| 8 |  | Answer saves as expected |
| 9 |  | Question and answer displays |
| 10 |  | Response taken |
| 11 |  | Results from test displayed |

## 9.3 Tester 1

|  |  |  |
| --- | --- | --- |
| Step | Output | Comments |
| 1 |  | New colour scheme looks good |
| 2 |  | Nicely laid out UI, easy to use option screen, nice improvement over prototype |
| 3 |  | Question Number now functions, still takes question as expected |
| 4 |  | Still strange to type in a separate file but its understandable |
| 5 |  | I forgot to save the file so it didnt display correctly *HUMAN ERROR* |
| 6 |  | Easy to use |
| 7 |  | Length of question make results table look strange, Weird { symbols |
| 8 |  | Retry Button works as expected |
| 9 |  | Exit button works as expected |

## 9.4 Tester 2

|  |  |  |
| --- | --- | --- |
| Step | Output | Comments |
| 1 |  | No comment |
| 2 |  | No comment |
| 3 |  | Question Number now works, box may be a bit small |
| 4 |  | Had enough time to fill in answer with new timer |
| 5 | Q+A screen | No comment |
| 6 |  | UI looks better with Or centered |
| 7 |  | UI looks broken with { signs |
| 8 |  | Retry works as expected |
| 9 |  | Exit still jarring, |

## 9.5 Tester 3

Tester 3 was the only tester to run multiple questions as the rest only did one question however in doing so they didn’t take screen shots like the previous so I have taken all their comments and broken them down for each screen. The only screen shots that they took were from the response to the final screen .

|  |  |  |
| --- | --- | --- |
| Step | Output | Comments |
| 1 | Start Screen | Still looks simple to use, text may be a bit faint |
| 2 | Options Screen | Settings titles confusing and don’t make sense |
| 3 | Question Screen | Still no instructions, though not a major issue just makes it messy, No timer on screen still |
| 4 | Answer File | Still separate file, need instructions, didn’t know you had to delete last entry, or got to save once |
| 5 | Q and A Screen | No comment |
| 6 | Response Screen | UI seems better and more original |
| 7 |  | Results page looks weird but you can understand |
| 8 |  | Retry works |
| 9 |  | Over all functionality good |

# 10 Evaluation and Conclusion

My final program meets all the points of my specification that I made at the start of the project as it allows a Turing Test to be completed to the best of its ability and give a full results table allowing the creator of the AI to improve their AI accordingly. The finished product looks very similar to the first screen designs I drew up although there are some obvious changes such as the size of the interface, font and the layouts of the screen. On the results screen there is some errors with how the information is displayed to the user as each entry has { next to each entry but I was unable to remove this in debugging. This was also picked up by the testers in the final testing of the finished program and is probably the largest flaw of the system itself. The rest of the program functions 100% as expected though as proven by the results of my testing.

In terms of improvements that I would make to the system I would like to make the program cover two screens meaning that when the human is chosen by the randomiser they enter their answer onto the screen. This is as at the moment the user must write their answer into the answer file which is the same place that the AI writes to when it issues its response. This current method is not user friendly which is why I would like to incorporate it into the program or have a separate program that works with this one. This is another thig that was pointed out by the testers though at least in the final testing they understood why it set up in this way.

From the final testing the testers also had some other minor improvements though these weren’t to functionality and more to aesthetics and the user-friendly nature of the system unlike the previous. This is as users didn’t understand the titles of the entries on the option system and wished that there was some kind of user documentation for the system so that they better understood how to use it. This is as the only time the program failed in testing was down to user error as they didn’t understand the system as well as I did as they weren’t the ones that created it.

In this sense I believe it is easy to see how the test may have failed but because of that I wouldn’t say that the system would have therefore failed. This is as with the correct documentation the system would’ve easily passed. This can be proven as in tester 3’s final comment they say that “overall functionality is good”.

Due to the success of the system in terms of the specification and its functionality I would say that I have met my goal that was set in the introduction of this project. Therefore, I believe this project is overall a success.

# 11 Appendix

## 11.1 Prototype Code

The following is the code from my prototype which was used as a basis for testing if my interface would be able to operate if completed as well as if there any other functions that could be added to help.

11.1.1 Code

from tkinter import \*

import random

import time

class TTI(Frame):

def Start(self):

def goQuestion():

Exit.grid\_forget()

Blank2.grid\_forget()

Menu.grid\_forget()

Blank.grid\_forget()

StartBut.grid\_forget()

QuestionNumber=0

Question(self,QuestionNumber)

#Start Screen Widgets

Exit = Button(self, width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=W, row=5, column=1)

Blank2= Label(self,text=" ", pady=20, font="calibri, 20")

Blank2.grid (row=4, column=1)

StartBut = Button(self, width=5, text="Start", font="calibri", command=goQuestion)

StartBut.grid (sticky=W, row=3, column=1)

Blank = Label(self,text=" ", pady=20, font="calibri, 20")

Blank.grid (row=2, column=1)

Menu = Label(self,text="Turing Test Interface", pady=20, font="calibri, 20")

Menu.grid (row=1, column=1)

def Question(self,QuestionNumber):

def goAnswer():

qInput=StringVar()

qInput=qEntry.get()

Exit.grid\_forget()

qNo.grid\_forget()

Send.grid\_forget()

qEntry.grid\_forget()

qHeader.grid\_forget()

sendQ(qInput)

def Random():

i=0

print("Getting Random Value")

for x in range(1):

no=random.randint(0,1)

return no

def sendQ(qInput):

no=Random()

print(no)

if no==0:

print("AI Recieving...")

print(qInput)

aiFile=open("aiFile.txt", "a")

aiFile.write(qInput)

aiFile.close()

t=0

print("Timer Start")

wait(t,qInput)

elif no==1:

print("Human Recieving...")

print(qInput)

t=0

print("Timer Start")

wait(t,qInput)

else:

print("Error Retrying")

sendQ(qInput)

def wait(t,qInput):

if t>=10:

Answer(self,QuestionNumber,qInput)

else:

time.sleep(1)

t=t+1

print(t)

wait(t,qInput)

#Question Screen Widgets

qNo = Label(self,text="Question Number:", pady=20, font="calibri, 10")

qNo.grid (row=6, column=3)

print(QuestionNumber)

Exit = Button(self, width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=W, row=5, column=1)

Send = Button(self, width=5, text="Send", font="calibri",command=goAnswer)

Send.grid (sticky=W, row=4, column=1)

qInput=StringVar()

qEntry = Entry(self, width=27,textvariable=qInput)

qEntry.grid (sticky=W,column=1,row=3)

qHeader = Label(self,text="Please Enter Your Question?", pady=20, font="calibri, 20")

qHeader.grid (row=2, column=1)

def Answer(self,QuestionNumber,qInput):

def goResponse():

Exit.grid\_forget()

Next.grid\_forget()

qHeader.grid\_forget()

aHeader.grid\_forget()

Response(self,QuestionNumber)

def getAns():

line=open("ansFile.txt","r")

ans=line.readline()

return ans

#Answer Screen Widgets

qEntry.grid\_forget()

Exit = Button(self, width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=W, row=6, column=1)

Next = Button(self, width=5, text="Next", font="calibri",command=goResponse)

Next.grid (sticky=W, row=5, column=1)

qHeader = Label(self,text=qInput, pady=20, font="calibri, 20")

qHeader.grid (row=3, column=1)

aHeader = Label(self,text=getAns(), pady=20, font="calibri, 20")

aHeader.grid (row=4, column=1)

def Response(self,QuestionNumber):

QuestionNumber=QuestionNumber+1

def sendAI():

Exit.grid\_forget()

ai.grid\_forget()

orHeader.grid\_forget()

human.grid\_forget()

resHeader.grid\_forget()

cont()

def sendHuman():

Exit.grid\_forget()

ai.grid\_forget()

orHeader.grid\_forget()

human.grid\_forget()

resHeader.grid\_forget()

cont()

def cont():

if QuestionNumber<2:

Question(self,QuestionNumber)

else:

print("Complete, displaying results")

Results()

#Response Screen Widgets

Exit = Button(self, width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=W, row=5, column=1)

ai = Button(self, width=5, text="AI", font="calibri",command=sendAI)

ai.grid (sticky=W, row=3, column=1)

orHeader = Label(self,text="OR", pady=20, font="calibri, 10")

orHeader.grid (row=3, column=2)

human = Button(self, width=5, text="Human", font="calibri",command=sendHuman)

human.grid (sticky=W, row=3, column=3)

resHeader = Label(self,text="Who issued that response?", pady=20, font="calibri, 20")

resHeader.grid (row=1, column=1)

def Results():

print("Here are your results")

#Results Screen Widgets

Exit = Button(self, width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=W, row=5, column=1)

rHeader = Label(self,text="Your results are:", pady=20, font="calibri, 20")

rHeader.grid (row=3, column=2)

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.Start()

questionNumber=0

root = Tk()

app = TTI(master=root)

root.title("T.T.I")

root.attributes("-fullscreen",True)

app.mainloop()

root.destroy()

## 11.2 Program code

#getting needed libaries

from tkinter import \*

import random

import time

class TTI(Frame):

def Start(self):

#Sets the background colour

root["bg"] ="#c2ebf6"

#function that is called when moving to the Options Screen

def goOptions():

#Creation of Arrays, and removal of widgets

qDB=['']\*100

aDB=['']\*100

rDB=['']\*100

TotalQ=0

tLength=10

Exit.grid\_forget()

Blank2.grid\_forget()

Menu.grid\_forget()

Blank.grid\_forget()

StartBut.grid\_forget()

QuestionNumber=0

Options(QuestionNumber,TotalQ,tLength,qDB,aDB,rDB)

#Start Screen Widgets

Exit = Button(self,background="#99ddff",fg="#ffffff", width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=5, column=1)

Blank2= Label(self,background="#c2ebf6",text=" ", pady=20, font="calibri, 20")

Blank2.grid (row=4, column=1)

StartBut = Button(self,background="#99ddff",fg="#ffffff", width=5, text="Start", font="calibri", command=goOptions)

StartBut.grid (sticky=N, row=3, column=1)

Blank = Label(self,background="#c2ebf6",text=" ", pady=20, font="calibri, 20")

Blank.grid (row=2, column=1)

Menu = Label(self,background="#c2ebf6",fg="#ffffff", text="Turing Test Interface", pady=20, font="calibri, 20")

Menu.grid (row=1, column=1)

def Options(QuestionNumber,TotalQ,tLength,qDB,aDB,rDB):

#Function that is called when the programs goes to the question screens

def goQuestion():

#Collection of varaiables from options screen

TotalQ=qNoInput.get()

tLength=tInput.get()

limit=LimitInput.get()

if TotalQ==0:

TotalQ=1

else:

TotalQ=TotalQ

AITotal=0

HumanTotal=0

#Redefing array parameters

qDB=[""]\*TotalQ

aDB=[""]\*TotalQ

rDB=[""]\*TotalQ

#Removal of widgets of Options screen

Exit.grid\_forget()

tEntry.grid\_forget()

tLabel.grid\_forget()

qNoEntry.grid\_forget()

qInputLabel.grid\_forget()

Apply.grid\_forget()

LimitEntry.grid\_forget()

lInputLabel.grid\_forget()

Question(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal)

#option screen widgets

Exit = Button(self,fg="#ffffff",background="#99ddff", width=15, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=8, column=1)

LimitInput=IntVar()

LimitEntry = Entry(self, width=27,textvariable=LimitInput)

LimitEntry.grid (sticky=N,column=1,row=7)

lInputLabel= Label(self,background="#c2ebf6",fg="#ffffff",text="What Are the Pass Constraints for the AI?", pady=20, font="calibri, 15")

lInputLabel.grid (row=6, column=1)

qNoInput=IntVar()

qNoEntry = Entry(self, width=27,textvariable=qNoInput)

qNoEntry.grid (sticky=N,column=1,row=5)

qInputLabel= Label(self,background="#c2ebf6",fg="#ffffff",text="How Many Questions Do You Want?", pady=20, font="calibri, 15")

qInputLabel.grid (row=4, column=1)

tInput=IntVar()

tEntry = Entry(self, width=27,textvariable=tInput)

tEntry.grid (sticky=N,column=1,row=3)

tLabel= Label(self,fg="#ffffff",background="#c2ebf6",text="How Long Do You Want Between Questions?", pady=20, font="calibri, 15")

tLabel.grid (row=2, column=1)

Apply = Button(self,background="#99ddff",fg="#ffffff", width=15, text="Apply Settings", font="calibri",command=goQuestion)

Apply.grid (sticky=N, row=9, column=1)

def Question(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal):

#Incrementing Question Number

QuestionNumber=QuestionNumber+1

i=QuestionNumber-1

#Function that is called when going to answer screen

def goAnswer():

#Collecting Variables

qInput=StringVar()

qInput=qEntry.get()

que="".join(qInput)

que=que+"\n"

qDB[i]=que

#Removing Widgets

Exit.grid\_forget()

Send.grid\_forget()

qEntry.grid\_forget()

qHeader.grid\_forget()

sendQ(qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

#Function to get random digit

def Random():

i=0

print("Getting Random Value")

for x in range(1):

no=random.randint(0,1)

return no

#Function to send question

def sendQ(qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal):

no=Random()

#if statement to select where the Question is sent

if no==0:

print("AI Recieving...")

print("Question: ",qInput)

aiFile=open("aiFile.txt", "a")

aiFile.write(qInput)

aiFile.close()

t=0

print("Timer Start")

wait(t,qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

elif no==1:

print("Human Recieving...")

print(qInput)

t=0

print("Timer Start")

wait(t,qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

else:

print("Error Retrying")

sendQ(qInput,TotalQ)

#function to start the timer

def wait(t,qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal):

if t>=tLength:

Answer(QuestionNumber,qInput,TotalQ,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

else:

time.sleep(1)

t=t+1

print(t)

wait(t,qInput,TotalQ,tLength,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

#Question Screen Widgets

qNo = Label(self,fg="#ffffff",background="#c2ebf6",text="Question Number:", pady=20, font="calibri, 10")

qNo.grid (row=6, column=3)

qNoNo = Label(self,fg="#ffffff",background="#c2ebf6",text=QuestionNumber, pady=20, font="calibri, 10")

qNoNo.grid (row=6, column=4)

Exit = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=5, column=1)

Send = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Send", font="calibri",command=goAnswer)

Send.grid (sticky=N, row=4, column=1)

qInput=StringVar()

qEntry = Entry(self, width=27,textvariable=qInput)

qEntry.grid (sticky=N,column=1,row=3)

qHeader = Label(self,fg="#ffffff",background="#c2ebf6",text="Please Enter Your Question?", pady=20, font="calibri, 20")

qHeader.grid (sticky=E, row=2, column=1)

#Answer Screen Function

def Answer(QuestionNumber,qInput,TotalQ,limit,qDB,aDB,rDB,i,AITotal,HumanTotal):

#Function called when going to the response screeen

def goResponse():

Exit.grid\_forget()

Next.grid\_forget()

qHeader.grid\_forget()

aHeader.grid\_forget()

Response(self,QuestionNumber,TotalQ,limit,qDB,aDB,rDB,i,AITotal,HumanTotal)

#Function gets the answer from the screen

def getAns():

line=open("ansFile.txt","r")

ansInput=line.readline()

ans=''.join(ansInput)

ans=ans+"\n"

aDB[i]=ans

return ans

#Answer Screen Widgets

qEntry.grid\_forget()

Exit = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=6, column=1)

Next = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Next", font="calibri",command=goResponse)

Next.grid (sticky=N, row=5, column=1)

qHeader = Label(self,fg="#ffffff",background="#c2ebf6",text=qInput, pady=20, font="calibri, 20")

qHeader.grid (row=3, column=1)

aHeader = Label(self,fg="#ffffff",background="#c2ebf6",text=getAns(), pady=20, font="calibri, 20")

aHeader.grid (row=4, column=1)

#Funcion for the response screen

def Response(self,QuestionNumber,TotalQ,limit,qDB,aDB,rDB,i,AITotal,HumanTotal):

#Function sends the AI response

def sendAI():

AI="AI\n"

rDB[i]=AI

Exit.grid\_forget()

ai.grid\_forget()

orHeader.grid\_forget()

human.grid\_forget()

resHeader.grid\_forget()

qNo.grid\_forget()

qNoNo.grid\_forget()

cont(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal)

#Function sends the Human Response

def sendHuman():

HUMAN="HUMAN\n"

rDB[i]=HUMAN

Exit.grid\_forget()

ai.grid\_forget()

orHeader.grid\_forget()

human.grid\_forget()

resHeader.grid\_forget()

qNo.grid\_forget()

cont(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal)

qNoNo.grid\_forget()

#Function to loop till question number is reached

def cont(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal):

if QuestionNumber<TotalQ:

Question(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB,AITotal,HumanTotal)

else:

print("Test Complete, displaying your results")

Results(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB)

#Response Screen Widgets

Exit = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=5, column=1)

ai = Button(self,fg="#ffffff",background="#99ddff", width=5, text="AI", font="calibri",command=sendAI)

ai.grid (sticky=W, row=3, column=1)

orHeader = Label(self,fg="#ffffff",background="#c2ebf6",text="OR", pady=20, font="calibri, 10")

orHeader.grid (sticky=N, row=3, column=1)

human = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Human", font="calibri",command=sendHuman)

human.grid (sticky=E, row=3, column=1)

resHeader = Label(self,fg="#ffffff",background="#c2ebf6",text="Who issued that response?", pady=20, font="calibri, 20")

resHeader.grid (row=1, column=1)

#Fucntion for Results Screen

def Results(QuestionNumber,TotalQ,tLength,limit,qDB,aDB,rDB):

print("Here are your results")

#Function to restart the test after one has been completed

def retry():

Exit.grid\_forget()

Retry.grid\_forget()

rHeader.grid\_forget()

CellR.grid\_forget()

CellC.grid\_forget()

CellL.grid\_forget()

TitleL.grid\_forget()

TitleC.grid\_forget()

TitleR.grid\_forget()

QuestionNumber=0

Options(QuestionNumber,TotalQ,tLength,qDB,aDB,rDB)

#Results Screen Widgets

Exit = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Exit", font="calibri")

Exit["command"] = self.quit

Exit.grid (sticky=N, row=102, column=2)

Retry = Button(self,fg="#ffffff",background="#99ddff", width=5, text="Retry", font="calibri",command=retry)

Retry.grid (sticky=N, row=101, column=2)

TitleL = Label(self,fg="#ffffff",background="#c2ebf6",text="Question", pady=20, font="calibri, 20")

TitleL.grid (sticky=N, row=2, column=1)

TitleC = Label(self,fg="#ffffff",background="#c2ebf6",text="Answer", pady=20, font="calibri, 20")

TitleC.grid (sticky=N, row=2, column=2)

TitleR = Label(self,fg="#ffffff",background="#c2ebf6",text="Response", pady=20, font="calibri, 20")

TitleR.grid (sticky=N, row=2, column=3)

rHeader = Label(self,fg="#ffffff",background="#c2ebf6",text="RESULTS", pady=20, font="calibri, 20")

rHeader.grid (row=1, column=2)

CellL = Label(self,fg="#ffffff",background="#c2ebf6",text=qDB, pady=20, font="calibri, 20")

CellL.grid (sticky=N, row=5, column=1)

CellC = Label(self,fg="#ffffff",background="#c2ebf6",text=aDB, pady=20, font="calibri, 20")

CellC.grid (sticky=N, row=5, column=2)

CellR = Label(self,fg="#ffffff",background="#c2ebf6",text=rDB, pady=20, font="calibri, 20")

CellR.grid (sticky=N, row=5, column=3)

#the initialiser function

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.Start()

questionNumber=0

root = Tk()

app = TTI(master=root)

root.title("T.T.I")

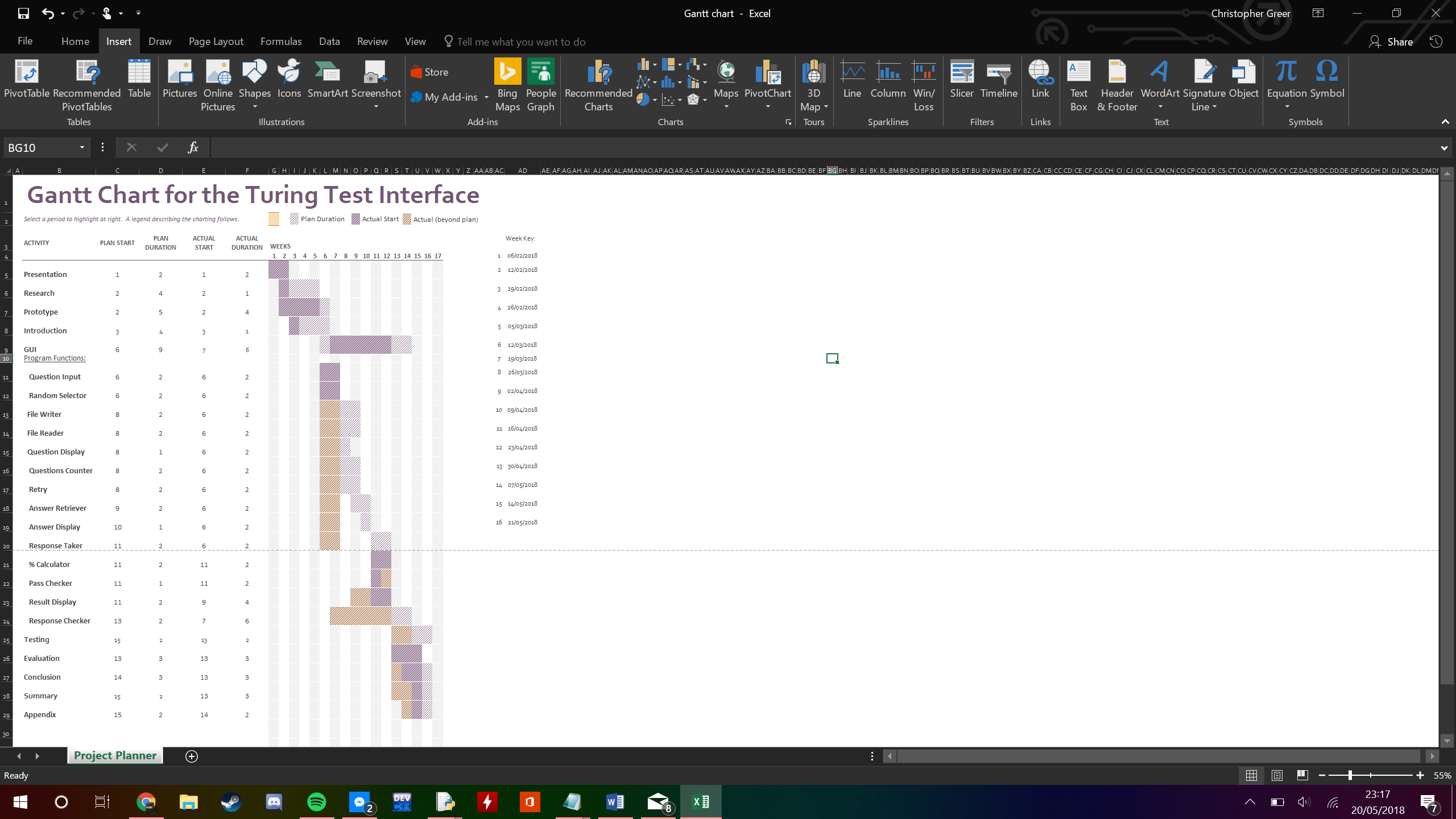
app.configure(background="#c2ebf6")

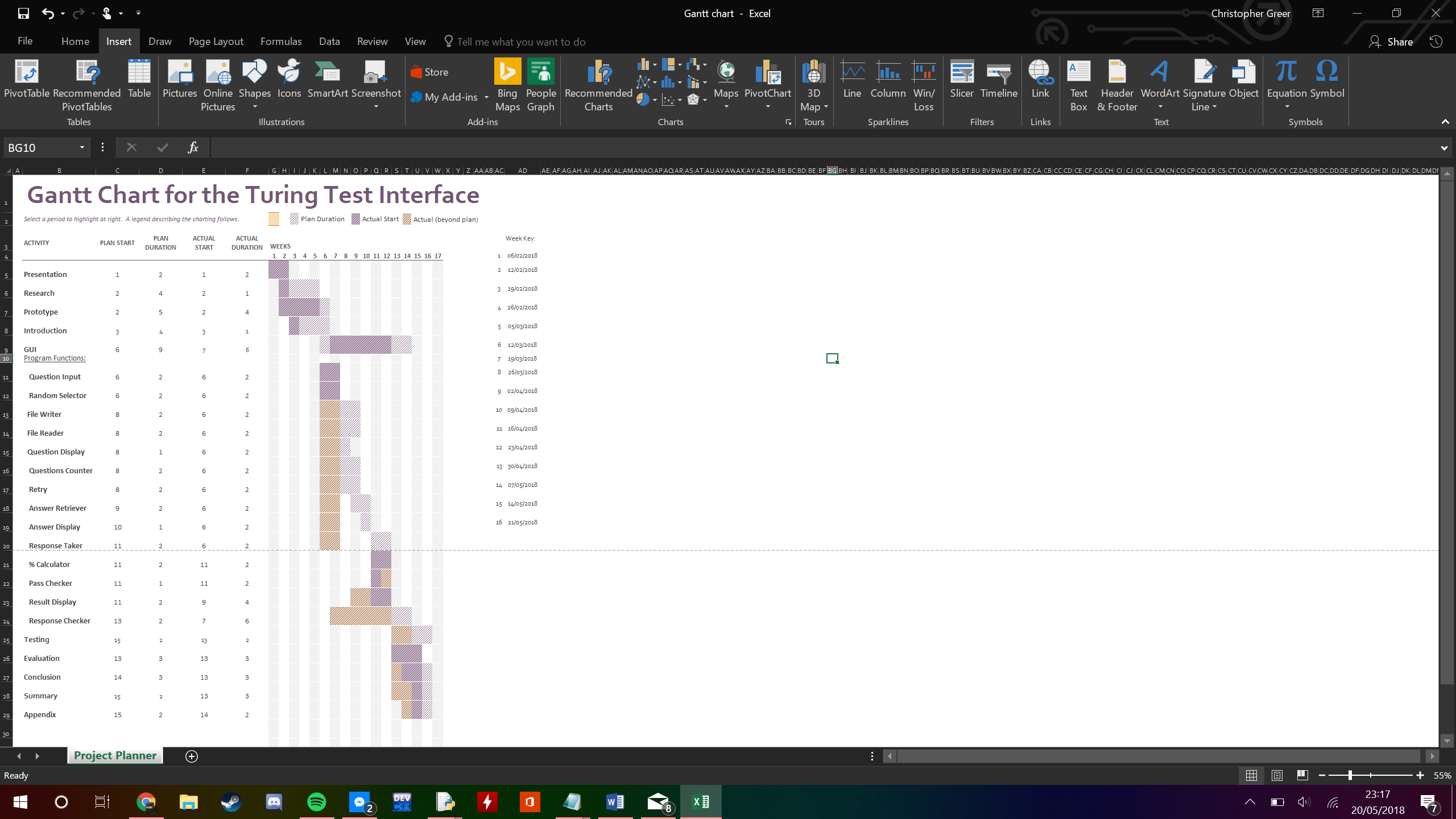
root.attributes("-fullscreen",True)

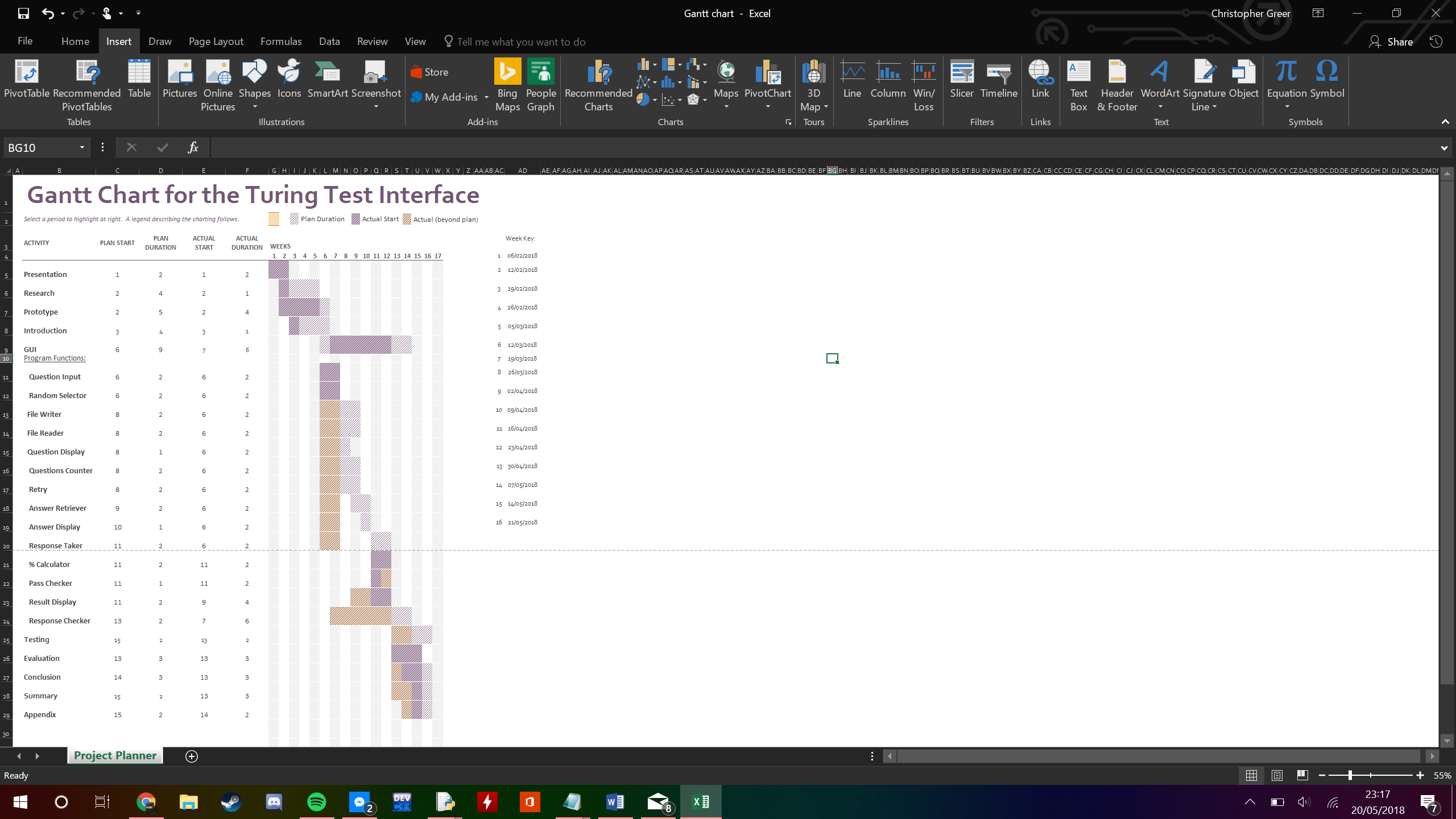
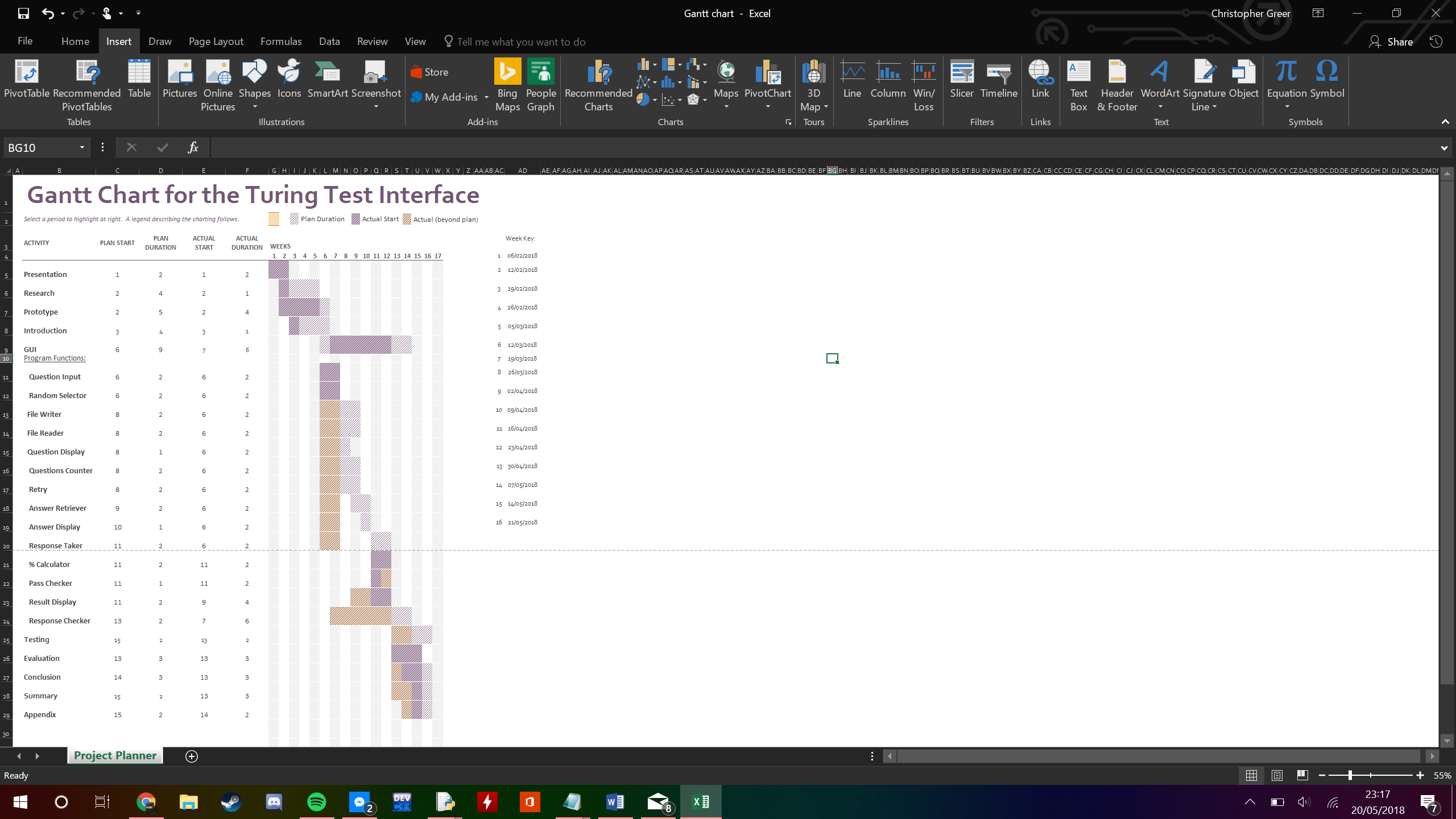
app.mainloop()

root.destroy()

## 11.3 Gantt Chart

The following image contains my Gantt chart which I used throughout the entirety of my project to keep me on track with all my modules including program functions and sections of the report. To understand the Gantt Chart please look at the following key as it shall explain all of the necessary information needed:





## 11.4 Tester Information Sheet

**Introduction**

Hello and welcome to this testing. I am creating a piece of software that will act as the interface for the Turing Test as my computing project which is part of my Computer Science Foundation Course.

**What is the Turing Test and how is it carried out?**

The Turing Test is a test that is carried out on an Artificial Intelligence to see if it possesses anything that could be considered a human like intelligence. This is done by a tester who sends questions into a white room. In that white room is either a human or the AI who will then issue a response. From this response the tester then must guess if there is a human or AI in the room. If the AI can trick the tester into making it think it’s the human it passes.

**Why are you needed?**

You are needed to make sure that the interface works correctly and that if a test was run that it would be completed and the developer would get enough information back to improve their AI. All you must do is run the test and then give feedback on each component of the system.

**TAKING PART IS NOT COMPULSORY!!!**

However if you do take part you will be asked to return to test the finalised version of the program

**If you wish to take part email me at-cg394@sussex.ac.uk**

# 12 References

1. Turing Scrapbook, Website about Turing’s life and work <http://www.turing.org.uk/scrapbook/test.html>
2. Investopedia, Websiste that gives information for people using stocks

<https://www.investopedia.com/terms/t/turing-test.asp>

1. Wired.com, Tech News website

<https://www.wired.com/2016/03/googles-ai-viewed-move-no-human-understand/>

1. KCC Software, Software Company <http://www.kccsoftware.com/kcchome/media/FirstTo1GHz.pdf>
2. The Emerging Future, Website on Future Technologies and Trends <http://theemergingfuture.com/speed-technological-advancement.htm>
3. Binary Move, Website Documenting the move to digital

[www.binarymove.com%2Fwhat-is-the-turing-test](www.binarymove.com%2Fwhat-is-the-turing-test%20)

1. Society for Human Resource Management, Website with tools and Information for workplace Management

<https://www.shrm.org/hr-today/news/hr-magazine/0417/pages/recruiting-gets-smart-thanks-to-artificial-intelligence.aspx>

1. Disruption Hub, Website that discuss about ethical situations <https://disruptionhub.com/disrupted-humanity-move-37/>
2. Android Central, News outlet that reports on all things Google related <https://www.androidcentral.com/google-duplex>

# 13 Bibliography

1. Artificial Intelligence in Psychology, Interdisciplinary Essays By Margaret A.Boden

Book Borrowed From the Sussex University Library

1. LeAp4Java: A Learning Application for Java By Rebecca Friend

Project Borrowed from the Computer Science Library

1. Robot or Human By Thomas Ball

Project Borrowed from the Computer Science Library