Appendices

Appendix A

Extra Information

A.1 Interview Transcripts

https://emckclac-my.sharepoint.com/:f:/g/personal/k1923149_kcl_ac_uk/ErsUxqwXyNxKsjK5y8txSMkBtNGdnPbufNqbq6ZoRo_sYw?e=KwA1fL

A.2 Usability Testing Transcripts

https://emckclac-my.sharepoint.com/:f:/g/personal/k1923149_kcl_ac_uk/Eow2bsZco5FOt3tHOi-woNgB3gLdLBzkiM5z8EQI5LcbVA?e=bOtHRy

A.3 Test Cases for Final Prototype

The test cases for final prototype are clustered based on the system's output message, which are listed as screenshots below:

A.3.1 Word exists inside the text

Figure A.1: System's output for the 1st occurrence of the word "encryption".

Figure A.2: System's output for the 2nd occurrence of the word "encryption".

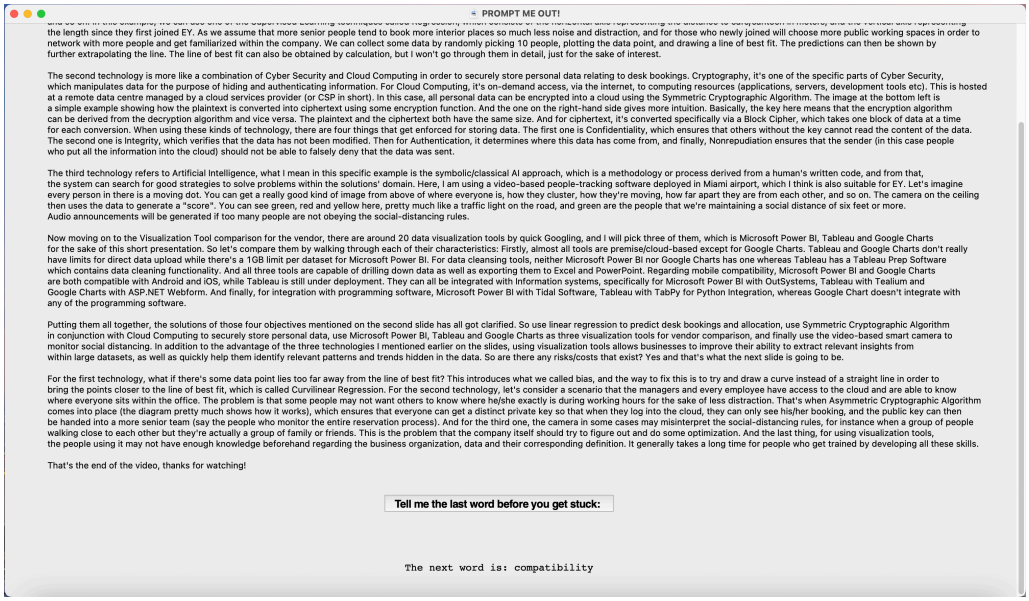

Figure A.3: System's output for the word "mobile".

A.3.2 Word does not exist inside the text

Figure A.4: System's output for the word "dog".

A.3.3 No input

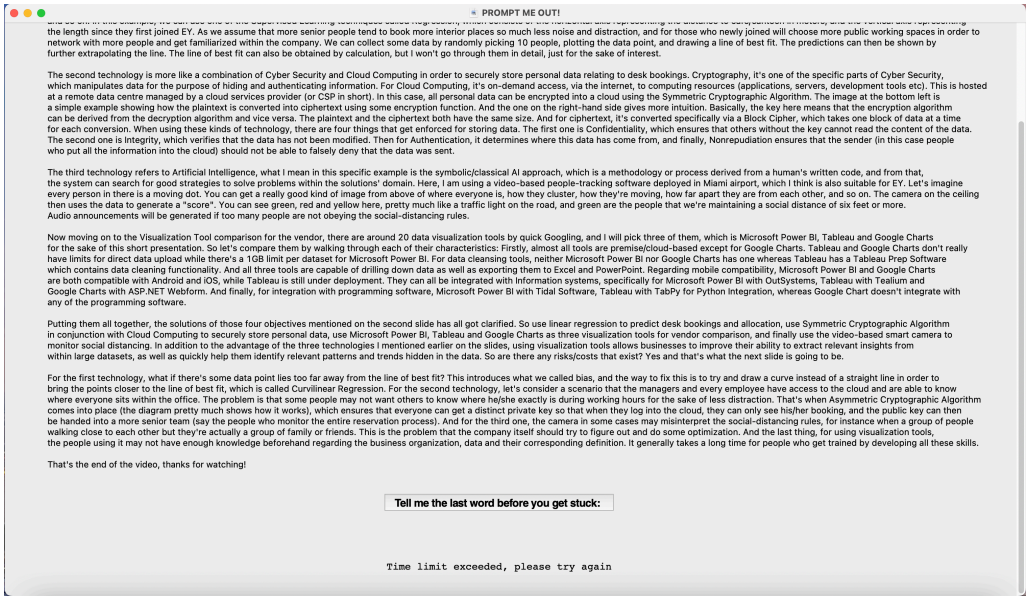

Figure A.5: No input has been detected by the system for more than 8 seconds.

A.3.4 End of the text

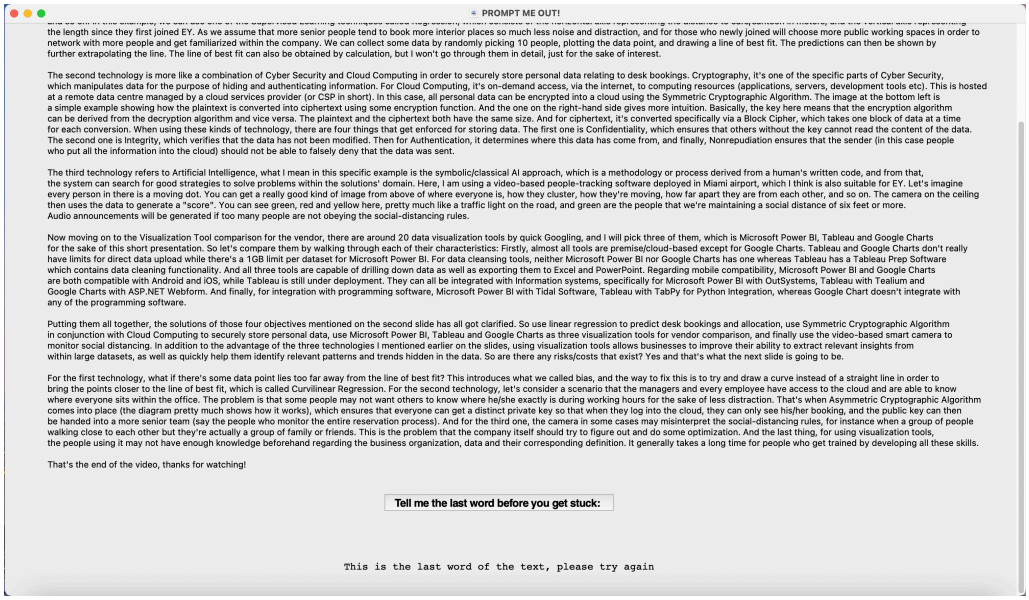

Figure A.6: System's output for the last word of the text "watching".

A.3.5 Last occurrence of a specific word inside the text

Figure A.7: System's output for the 1st occurrence of the word "asymmetric".

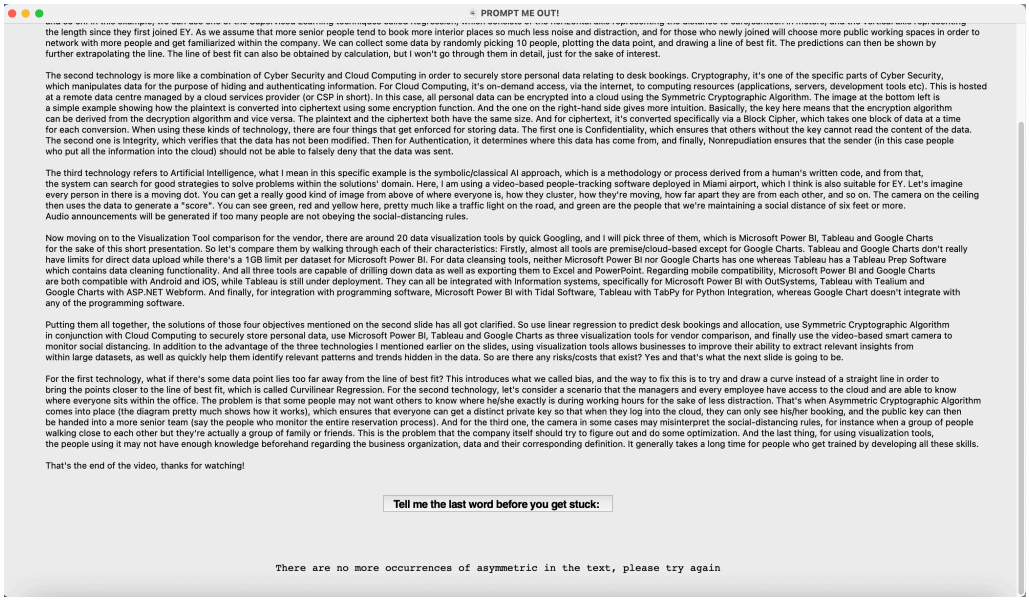

Figure A.8: System's output for the last occurrence of the word "asymmetric".

Appendix B

User Guide

Below provides a complete walkthrough of how to properly set up and use the software (i.e. final prototype) implemented in this project that is divided into three subsections:

B.1 Integrated Development Environment and Python Installation

If none of the programming languages and environments has previously been installed in the personal machine, then downloading these two will be the preconditions before proceeding to run the program.

Visual Studio Code has nowadays become one of the most popular Integrated Development Environments (IDEs) for code writing and running, the installation for various operating systems can be found on the following website:

`https://code.visualstudio.com/download`

A follow-up installation of Python 3.9.7 can be achieved via visiting the below link, for the sake of matching the programming language version being used in this project:

`https://www.python.org/downloads/release/python-397/`

Once the Visual Studio Code IDE has been successfully downloaded, open the application and download the relevant extensions for the Python language (e.g. Python, Python for VSCode etc.) by clicking one of the menu items displayed on the left side of the window, depicted as a screenshot below:

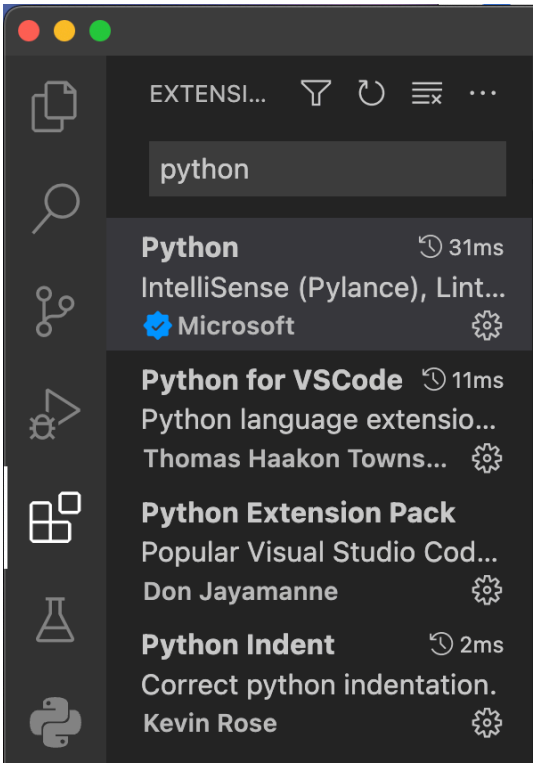

Figure B.1: Left menu bar of the Visual Studio Code IDE.

B.2 External Libraries Installation

After setting up the environment and its corresponding programming language, it is time to dive deeper into the program itself and do installations with regards to some Python external libraries being used in this program, in order to make the Speech Recognition and Text-to-Speech algorithms work.

For Speech Recognition, Google Speech Recognition API followed by PyAudio should be installed so that the system is able to detect the user's sound input directly from the microphone. The following website is useful to refer to:

`https://www.thepythoncode.com/article/`
`using-speech-recognition-to-convert-speech-to-text-python`

Similarly, for Text-to-Speech, an online Google Text-to-Speech library has to be installed via the following link:

`https://www.thepythoncode.com/article/convert-text-to-speech-in-python`

B.3 Program Running

The program itself already contains a sample user's premade speech note stored inside the users_note variable in line 56, which can be customised by replacing the text within the multiline-string represented as triple quotes.

Finally, to run the software simply click the "Run Code" button located on top of the menu bar (as the screenshot below shows), a new window will then pop up, which is where the program starts running.

Notice that in order to make the system's output message visible, the program window has to be slightly resized when the button is pressed for the first time.

Figure B.2: Top menu bar of the Visual Studio Code IDE.

Appendix C

Program Listings

C.1 Initial Prototype

```
#!/usr/bin/env python
# coding: utf-8


# Speech-to-Text Import.
import speech_recognition as sr
# Text-to-Speech Import.
import gtts
from playsound import playsound
# To remove the punctuation and upper case letters from the user's pre-made note.
import string

def prompter():

    """
    Prompts the cue word right next to the input word via text and sound.

    Raises
    ------
    UnknownValueError
        If no input has been detected by the system for more than 5 seconds.
    IndexError
        If the last word of the text has been entered.
    """

```

```
    users_note = """
                    New York is 3 hours ahead of California,
                    but it does not make California slow.
                    Someone graduated at the age of 22,
                    but waited 5 years before securing a good job!
                    Someone became a CEO at 25,
                    and died at 50.
                    While another became a CEO at 50,
                    and lived to 90 years.
                    Someone is still single,
                    while someone else got married.
                    Obama retires at 55,
                    but Trump starts at 70.
                    Charles Njonjo married at 52 and is now 96...his peers could have
     married at 25 and died at 50.
                    Absolutely everyone in this world works based on their Time Zone.
                    People around you might seem to go ahead of you,
                    some might seem to be behind you.
                    But everyone is running their own RACE, in their own TIME.
                    Don't envy them or mock them.
                    They are in their TIME ZONE, and you are in yours!
                    Life is about waiting for the right moment to act.
                    So, RELAX.
                    You are not LATE.
                    You are not EARLY.
                    You are very much ON TIME, and in your TIME ZONE Destiny set up
    for you.
                 """
    # Convert the string format of text into a list of strings.
    note_list = users_note.split()

    # Give user 5 attempts in total.
    for users_try in range(5):
        # Initialise the recognizer.
        r = sr.Recognizer()
        with sr.Microphone() as source:
            try:
                word_list = input('Forgot what you wanna say next? Type something: ')
    .split()
                # Read the audio data from the default microphone, give user 5
    seconds to respond.
                audio_data = r.record(source, duration = 5)
```

```
                print('Recognizing...')
                # Convert speech to text.
                text = r.recognize_google(audio_data)
                for word in word_list:
                    if word in note_list:
                        # Keep track of the user's position by allowing the system to
 search the text forward only.
                        note_list = note_list[note_list.index(word) + 1:]
                        # Remove punctuation and upper case letters for the system's
output.
                        note_list = [''.join(letter.lower() for letter in word if
letter not in string.punctuation) for word in note_list]
                        next_word = note_list[0]
                        print(next_word)
                        # Make request to Google to get synthesis.
                        tts = gtts.gTTS(next_word)
                        tts.save("word.mp3")
                        playsound("word.mp3")
                    else:
                        print('Sorry, try again')
                        tts2 = gtts.gTTS('Sorry, try again')
                        tts2.save("error.mp3")
                        playsound("error.mp3")
                print('\n')

            # When the user doesn't produce any response for more than 5 seconds.
            except sr.UnknownValueError:
                print('Time limit exceeded, please try again')
                tts3 = gtts.gTTS('Time limit exceeded, please try again')
                tts3.save("exception1.mp3")
                playsound("exception1.mp3")
                print('\n')
            # When the user inputs the last word of his/her notes.
            except IndexError:
                print('This is the last word of the text, please try again')
                tts4 = gtts.gTTS('This is the last word of the text, please try again
')
                tts4.save("exception2.mp3")
                playsound("exception2.mp3")
                print('\n')

    print('Thanks for using my teleprompter, bye')
```

```
102
103 prompter()
```

Listing C.1: Initial Prototype.

C.2 Final Prototype

```
#!/usr/bin/env python
# coding: utf-8


# Speech-to-Text Import.
import speech_recognition as sr
# Text-to-Speech Import.
import gtts
from playsound import playsound
# To remove the punctuation and upper case letters from the user's pre-made note.
import string
# To build Graphical User Interfaces.
from tkinter import *
# Add a full-screen scrollbar as ttk widget.
from tkinter import ttk
# To change the font type of Tkinter button.
import tkinter.font as font
# A data structure used in manipulating the user's note.
from collections import defaultdict

# Create a root widget that corresponds to the main window.
root = Tk()
root.title("PROMPT ME OUT!")
root.iconbitmap("/Users/andersonyou/Desktop/Year 3/Individual Project/21-22/Final
    Prototype/teleprompter.ico")
root.geometry("800x600") # default size of the main window

# The section below before defining users_note variable refers to a general method
    for
# adding a full-screen scrollbar to the main window in Tkinter.
#
# Adapted from a YouTube tutorial video delivered by John Elder:
# URL: https://www.youtube.com/watch?v=0WafQCaok6g
#
# Create a main frame.
main_frame = Frame(root)
main_frame.pack(fill = BOTH, expand = 1)

# Create a canvas.
my_canvas = Canvas(main_frame)
```

```
my_canvas.pack(side = LEFT, fill = BOTH, expand = 1)

# Add a scrollbar to the canvas.
my_scrollbar = ttk.Scrollbar(main_frame, orient = VERTICAL, command = my_canvas.yview
    )
my_scrollbar.pack(side = RIGHT, fill = Y)

# Configure the canvas.
my_canvas.configure(yscrollcommand = my_scrollbar.set)
my_canvas.bind('<Configure>', lambda e: my_canvas.configure(scrollregion = my_canvas.
    bbox("all")))

# Create another frame inside the canvas.
second_frame = Frame(my_canvas)

# Add that new frame to a window in the canvas.
my_canvas.create_window((0,0), window = second_frame, anchor = "nw")

global users_note
users_note = """
                Hello, I am Anderson and I was working on a small piece of technology
     consulting project for EY.

                Recently, my company had been fined by the violation of Regulation
    1215, which highlights that the number of people in their office wearing a face
    mask, and the number of people in their office at any point in time.
                I was told to find some solutions by achieving four objectives, which
     are desk bookings and allocations' prediction, secure storage of personal data,
    using visualization tools for vendor comparison and monitoring social distancing.

                The first technology I am going to talk about refers to Machine
    Learning, and precisely Supervised Learning, which can be used to effectively
    forecast desk bookings and allocations. So what is Supervised Learning?
                Well if you have a look at the diagram on the top right, let's assume
     that the Model is a Mathematical function say y = 3x, and assign the training
    input data to be 2, 3 and 4, then every time we feed those inputs into the model,
                it will get updated and output 6, 9 and 12 respectively. Now, this
    model has got trained and that means if we give it a brand-new piece of data, we
    can check whether the system works as intended. So 5 outputs 15, 6 outputs 18,
                and so on. In this example, we can use one of the Supervised Learning
     techniques called Regression, which consists of the horizontal axis representing
     the distance to café/canteen in meters, and the vertical axis representing
```

the length since they first joined EY. As we assume that more senior people tend to book more interior places so much less noise and distraction, and for those who newly joined will choose more public working spaces in order to network with more people and get familiarized within the company. We can collect some data by randomly picking 10 people, plotting the data point, and drawing a line of best fit. The predictions can then be shown by further extrapolating the line. The line of best fit can also be obtained by calculation, but I won't go through them in detail, just for the sake of interest.

The second technology is more like a combination of Cyber Security and Cloud Computing in order to securely store personal data relating to desk bookings. Cryptography, it's one of the specific parts of Cyber Security, which manipulates data for the purpose of hiding and authenticating information. For Cloud Computing, it's on-demand access, via the internet, to computing resources (applications, servers, development tools etc). This is hosted at a remote data centre managed by a cloud services provider (or CSP in short). In this case, all personal data can be encrypted into a cloud using the Symmetric Cryptographic Algorithm. The image at the bottom left is a simple example showing how the plaintext is converted into ciphertext using some encryption function. And the one on the right-hand side gives more intuition. Basically, the key here means that the encryption algorithm can be derived from the decryption algorithm and vice versa. The plaintext and the ciphertext both have the same size. And for ciphertext, it's converted specifically via a Block Cipher, which takes one block of data at a time for each conversion. When using these kinds of technology, there are four things that get enforced for storing data. The first one is Confidentiality, which ensures that others without the key cannot read the content of the data. The second one is Integrity, which verifies that the data has not been modified. Then for Authentication, it determines where this data has come from, and finally, Nonrepudiation ensures that the sender (in this case people who put all the information into the cloud) should not be able to falsely deny that the data was sent.

The third technology refers to Artificial Intelligence, what I mean in this specific example is the symbolic/classical AI approach, which is a methodology or process derived from a human's written code, and from that, the system can search for good strategies to solve problems within the solutions' domain. Here, I am using a video-based people-tracking software deployed in Miami airport, which I think is also suitable for EY. Let's imagine

every person in there is a moving dot. You can get a really good kind of image from above of where everyone is, how they cluster, how they're moving, how far apart they are from each other, and so on. The camera on the ceiling then uses the data to generate a "score". You can see green, red and yellow here, pretty much like a traffic light on the road, and green are the people that we're maintaining a social distance of six feet or more. Audio announcements will be generated if too many people are not obeying the social-distancing rules.

Now moving on to the Visualization Tool comparison for the vendor, there are around 20 data visualization tools by quick Googling, and I will pick three of them, which is Microsoft Power BI, Tableau and Google Charts for the sake of this short presentation. So let's compare them by walking through each of their characteristics: Firstly, almost all tools are premise/cloud-based except for Google Charts. Tableau and Google Charts don't really have limits for direct data upload while there's a 1GB limit per dataset for Microsoft Power BI. For data cleansing tools, neither Microsoft Power BI nor Google Charts has one whereas Tableau has a Tableau Prep Software which contains data cleaning functionality. And all three tools are capable of drilling down data as well as exporting them to Excel and PowerPoint. Regarding mobile compatibility, Microsoft Power BI and Google Charts are both compatible with Android and iOS, while Tableau is still under deployment. They can all be integrated with Information systems, specifically for Microsoft Power BI with OutSystems, Tableau with Tealium and Google Charts with ASP.NET Webform. And finally, for integration with programming software, Microsoft Power BI with Tidal Software, Tableau with TabPy for Python Integration, whereas Google Chart doesn't integrate with any of the programming software.

Putting them all together, the solutions of those four objectives mentioned on the second slide has all got clarified. So use linear regression to predict desk bookings and allocation, use Symmetric Cryptographic Algorithm in conjunction with Cloud Computing to securely store personal data, use Microsoft Power BI, Tableau and Google Charts as three visualization tools for vendor comparison, and finally use the video-based smart camera to monitor social distancing. In addition to the advantage of the three technologies I mentioned earlier on the slides, using visualization tools allows businesses to improve their ability to extract relevant insights from within large datasets, as well as quickly help them identify relevant patterns and trends hidden in the data. So are there any risks/costs that exist? Yes and that's what the next slide is going to be.

```
97
98                 For the first technology, what if there's some data point lies too
    far away from the line of best fit? This introduces what we called bias, and the
    way to fix this is to try and draw a curve instead of a straight line in order to
99                 bring the points closer to the line of best fit, which is called
    Curvilinear Regression. For the second technology, let's consider a scenario that
     the managers and every employee have access to the cloud and are able to know
100                where everyone sits within the office. The problem is that some
    people may not want others to know where he/she exactly is during working hours
    for the sake of less distraction. That's when Asymmetric Cryptographic Algorithm
101                comes into place (the diagram pretty much shows how it works), which
    ensures that everyone can get a distinct private key so that when they log into
    the cloud, they can only see his/her booking, and the public key can then
102                be handed into a more senior team (say the people who monitor the
    entire reservation process). And for the third one, the camera in some cases may
    misinterpret the social-distancing rules, for instance when a group of people
103                walking close to each other but they're actually a group of family or
     friends. This is the problem that the company itself should try to figure out
    and do some optimization. And the last thing, for using visualization tools,
104                the people using it may not have enough knowledge beforehand
    regarding the business organization, data and their corresponding definition. It
    generally takes a long time for people who get trained by developing all these
    skills.
105
106                That's the end of the video, thanks for watching!
107             """
108
109 global note_label
110 # Create a Label widget to hold the user's pre-made note with left align.
111 note_label = Label(second_frame, text = users_note, justify = "left")
112 # Shove it onto the screen, with some padding below the border.
113 note_label.grid(row = 0, column = 0, pady = 15)
114
115 output_label = Label(second_frame)
116
117 def prompter(current_index = defaultdict(int)):
118
119     """
120     Prompts the cue word right next to the input word via text and sound.
121
122     Parameters
123     ----------
```

```
    current_index : collections.defaultdict
        This is bound to a new defaultdict(int) only once, when the function is defined here,
        not each time the function is called. Therefore current_index is preserved between calls.
        Key = word entered (e.g. "you")
        Value = index of word's previous next word (e.g. "2" defaults to 0)

    Raises
    ------
    ValueError
        If the word entered either has no more occurrences for the rest of the text,
        or it cannot be found inside the text.
    """

    # Convert the string format of text into a list of strings.
    note_list = users_note.split()
    # Remove punctuation and upper case letters for the system's output.
    note_list = [''.join(letter.lower() for letter in word if letter not in string.punctuation) for word in note_list]

    # Initialize the recognizer.
    r = sr.Recognizer()
    with sr.Microphone() as source:
        # Read the audio data from the default microphone, give user 8 seconds to respond.
        audio_data = r.record(source, duration = 8)
        # Convert speech to text.
        word = r.recognize_google(audio_data)

        # Get the index of the word, starting from the word's previous
        # next word index, which is stored in current_index[word], then increment 1.
        try:
            next_word_index = note_list.index(word, current_index[word]) + 1

        # There are either no more occurrences of the word, or the word doesn't exist at all.
        except ValueError:
            if word in note_list:
                last_occurrence = f'There are no more occurrences of {word} in the text, please try again'
                global output_label
```

```
160                 # Delete the previous system's output.
161                 output_label.destroy()
162                 # Update the system's output.
163                 output_label = Label(second_frame, text = last_occurrence, font = ("
    Courier", 15))
164                 # Locate the system's output right below the button.
165                 output_label.grid(row = 3, column = 0, pady = 30)
166                 # Make request to Google to get synthesis.
167                 tts = gtts.gTTS(last_occurrence)
168                 # Save the audio file.
169                 tts.save("ValueError1.mp3")
170                 # Play the audio file.
171                 playsound("ValueError1.mp3")
172             else:
173                 output_label.destroy()
174                 output_label = Label(second_frame, text = "Sorry, try again", font =
    ("Courier", 15))
175                 output_label.grid(row = 3, column = 0, pady = 30)
176                 tts2 = gtts.gTTS('Sorry, try again')
177                 tts2.save("ValueError2.mp3")
178                 playsound("ValueError2.mp3")
179             return
180 
181         # Update the word's current index.
182         current_index[word] = next_word_index
183 
184         # When the input word is the last word of the user's note.
185         if next_word_index == len(note_list):
186             output_label.destroy()
187             output_label = Label(second_frame, text = "This is the last word of the
    text, please try again", font = ("Courier", 15))
188             output_label.grid(row = 3, column = 0, pady = 30)
189             tts3 = gtts.gTTS('This is the last word of the text, please try again')
190             tts3.save("IndexOutOfBoundError.mp3")
191             playsound("IndexOutOfBoundError.mp3")
192             return
193 
194         # Yield the next word.
195         next_word = note_list[next_word_index]
196         output_label.destroy()
197         output_label = Label(second_frame, text = "The next word is: " + next_word,
    font = ("Courier", 15))
```

```
198         output_label.grid(row = 3, column = 0, pady = 30)
199         tts4 = gtts.gTTS(next_word)
200         tts4.save("output.mp3")
201         playsound("output.mp3")
202 
203 def main():
204 
205     """
206     Calls the previous function and handles another exception.
207 
208     Raises
209     ------
210     UnknownValueError
211         If no input has been detected by the system for more than 8 seconds.
212     """
213 
214     try:
215         prompter()
216 
217     # When the user doesn't produce any response for more than 8 seconds.
218     except sr.UnknownValueError:
219         global output_label
220         output_label.destroy()
221         output_label = Label(second_frame, text = "Time limit exceeded, please try
    again", font = ("Courier", 15))
222         output_label.grid(row = 3, column = 0, pady = 30)
223         tts5 = gtts.gTTS('Time limit exceeded, please try again')
224         tts5.save("UnknownValueError.mp3")
225         playsound("UnknownValueError.mp3")
226         return
227 
228 # Create a Button widget that enables users to press when they get stuck on a word,
    customise the font.
229 buttonFont = font.Font(family = 'Helvetica', size = 16, weight = 'bold')
230 btn = Button(second_frame, text = "Tell me the last word before you get stuck: ",
    font = buttonFont, command = main)
231 # Locate the button right below the user's note.
232 btn.grid(row = 1, column = 0)
233 
234 # Add some space between the button and the system's output.
235 Label(second_frame, text = "\n").grid(row = 2, column = 0)
236 
```

```
237 # Centre all the contents displayed in the window.
238 second_frame.columnconfigure(0, weight = 1)
239 second_frame.rowconfigure(0, weight = 1)
240
241 # Create an event loop.
242 root.mainloop()
```

Listing C.2: Final Prototype.