JOHN

VIRTUAL ENGLISH TEACHER: GRAMMAR & PRONUNCIATION

Mohammed S. Obeidat & Abdullah Al Ajami | CS499B | First Semester 2022/2023

Computer Sciences Department

Faculty of Information Technology and Computer Science

Yarmouk University

Irbid, Jordan

Graduation Project Report

BSc Project

CS Department

Project ID: 1 (ALPHA)

JOHN

Virtual English Teacher

Mohammed S. Obeidat 2020901023

Abdullah Al Ajami 2019801014

Department of Computer Science

Faculty of Information Technology and Computer Sciences

Yarmouk University, Jordan

Contact Information

This project report is submitted to the Department of Computer Science at Yarmouk University in partial fulfillment of the requirements for the degree of Bachelor of Information Technology in Computer Science.

Author(s):

Mohammed S. Obeidat 2020901023

Abdullah Al Ajami 2019801014

Address: Irbid / Kuwait

E-mail: mohdsameer14.ms@gmail.com

University supervisor(s):

Dr. Ameera Jaradat

Department of Computer Science

Department of Computer Science
Faculty of Information Technology
and Computer Science
Yarmouk University
Jordan

Internet: http://yu.edu.jo
Phone: +962 2 72711111 Ext. 6710
Fax : +962 2 7211111

Intellectual Property Right Declaration

This is to declare that the work under the supervision of Dr. Ameera Jaradat having title " John" carried out in partial fulfillment of the requirements of Bachelor of Information Technology in Computer Science, is the sole property of the Yarmouk University and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor.

This above statement applies to all students and faculty members.

Date: 11/01/2023

Author(s):

Name: Mohammed S. Obeidat *Signature: _____*

Name: Abdullah Al Ajami *Signature: _____*

Supervisor(s):

Name: Dr. Ameera Jaradat Signature: _____

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the supervision of Dr. Ameera Jaradat having title “John” is the sole contribution of the author(s) and no part hereof has been reproduced illegally (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: 11/01/2023

Author(s):

Name: Mohammed S. Obeidat *Signature:* _____

Name: Abdullah Al Ajami *Signature:* _____

ACKNOWLEDGMENTS

This work is dedicated to my dear parents, the most loving in this world.

As the developer of this program, I would like to express my sincere gratitude to my parents for their constant support and encouragement throughout my journey. Their unwavering belief in me has been a driving force behind my success. I would also like to thank my university professors for imparting their invaluable knowledge and wisdom. Their guidance and insights have been instrumental in shaping my technical and analytical skills. I would like to extend a thank you to the OpenAI team as well for providing the opportunity to train the language model and make this program possible. Finally, I would like to acknowledge the efforts of the entire development team for their hard work and dedication in bringing this program to life.

-Mohammed S. Obeidat.

ABSTRACT

This program features a cutting-edge graphical user interface and offers a sleek and innovative voice-controlled interface for natural language processing tasks, including grammar correction. With this program, users can simply speak their thoughts and have their sentences corrected in real-time with stunning accuracy. The program uses the latest in speech recognition technology and machine learning algorithms to transcribe audio input from a microphone, recognize the user's speech, and make grammatical corrections using the OpenAI API. The corrected speech is then synthesized using the Google Text-to-Speech API and played back to the user with crystal-clear sound, providing a truly immersive experience. The program is designed to continuously listen to the user's speech and make corrections until the user says "exit". This program is not only practical, but also visually stunning, making it a must-have tool for anyone looking to improve their language skills.

Keywords: Voice recognition, Text to speech, Grammar correction, Graphical user interface, User-friendly, Intuitive, Innovative, Cutting-edge technology, Streamlined, Effortless interaction

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 Purpose of the Project

The purpose of this program is to take in speech input from the user through a microphone, correct the grammar in the recognized speech using OpenAI, and output the corrected sentence in speech form. The first script continuously listens for speech input and recognizes it using the Speech Recognition library and Google's speech recognition API. The second script takes a one-time speech input, converts it to text, corrects the grammar using OpenAI, and outputs the corrected sentence in speech form using the gTTS library and the playsound library.

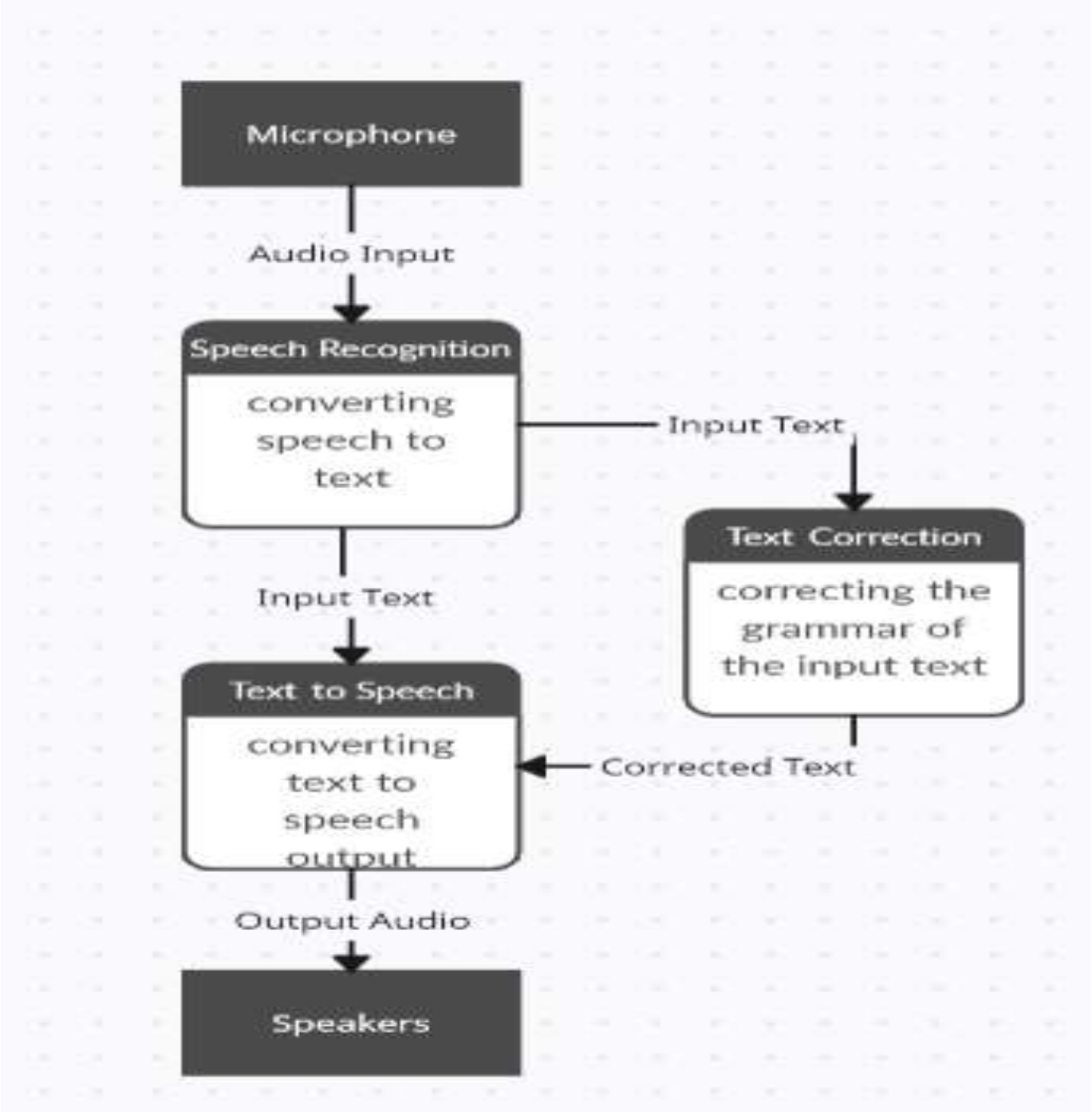
1.2 Description of functionality

This program is a combination of two scripts that offer various functionalities. The first script provides a speech recognition and text-to-speech conversion feature using the speech_recognition and win32com library. The user can speak commands that are then recognized by the program and either converted to text or executed as a system command. If the user says "exit", the program will end and say goodbye. The second script provides a grammar correction feature using the openai API and gTTS library. The user can speak a sentence, which will then be transcribed to text and passed to the openai API to correct any grammatical errors. The corrected sentence is then converted to speech and played back to the user.

Chapter 2 SYSTEM ANALYSIS

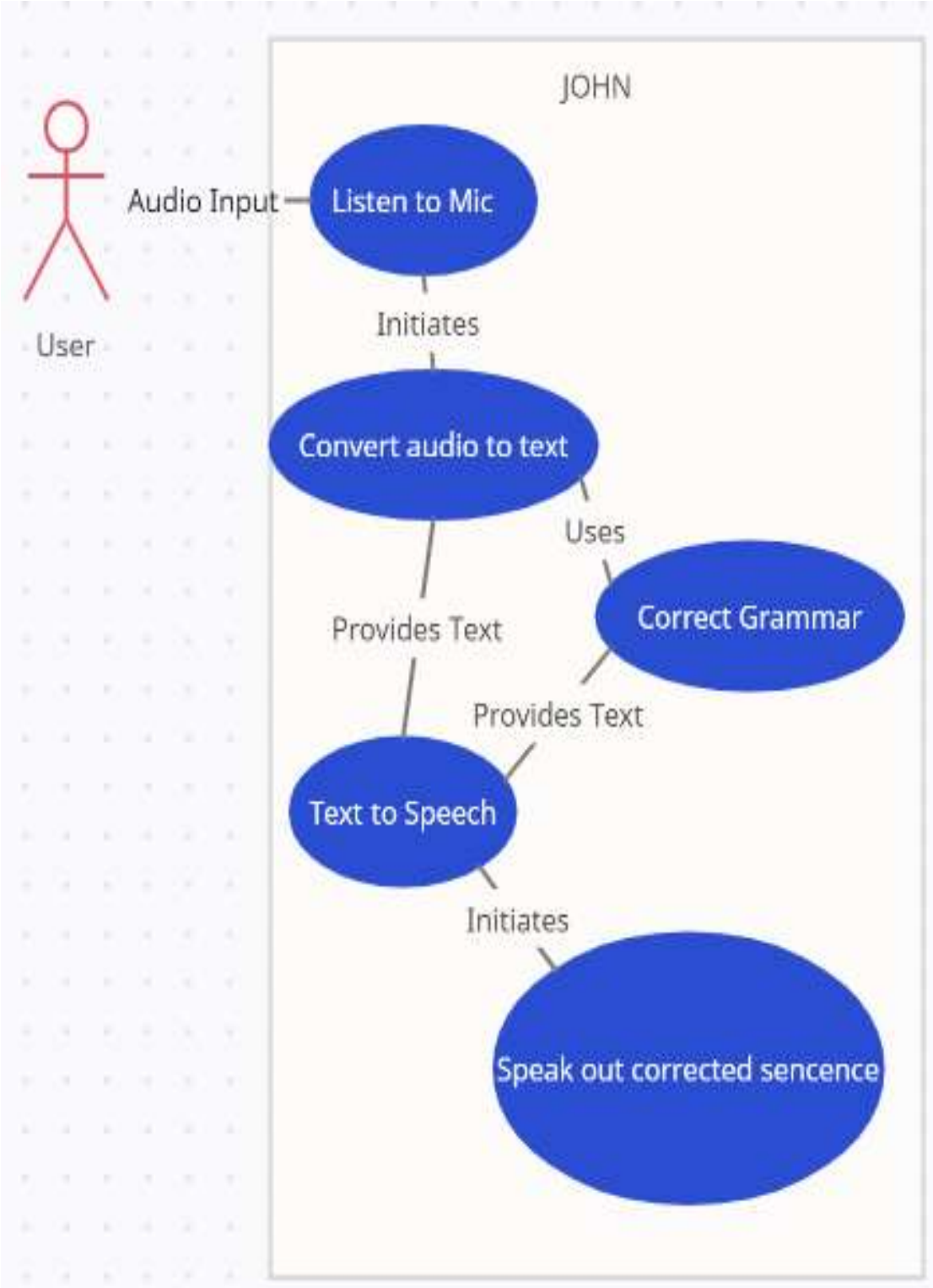
2.1 Data Analysis

2.1.1 Data flow diagram



2.1.2 System requirements

2.1.2.1 Clients, customers, and users (Use-case Diagram)



2.1.2.2 Functional and data requirements

The functional requirements of this program are:

1-To listen to the user's voice input through a microphone

2-To convert the audio input to text using Google's speech recognition API

3-To correct the grammar of the text input using OpenAI's language model API

4-To save the corrected sentence as an MP3 audio file

5-To play the corrected sentence audio file

The data requirements are:

1-Audio input from the microphone.

2-Text representation of the audio input.

3-Corrected text representation of the input.

4-MP3 audio file of the corrected sentence.

2.1.2.3 Non-functional requirements

1. Performance: The program should perform efficiently, providing real-time correction of grammar in a prompt manner.
2. Reliability: The program should be reliable and produce consistent results in terms of grammar correction.
3. Usability: The program should be easy to use, with a clear and simple interface for inputting and outputting text.
4. Scalability: The program should be able to handle increasing amounts of input data as its user base grows.
5. Security: The program should be secure, protecting sensitive information such as API keys and user data.
6. Maintainability: The program should be maintainable, allowing for easy updating and fixing of bugs as they are discovered.

2.1.2.3.1 Look and feel requirements.

This product shall provide an easy to use Graphical User Interface.

This product shall provide separate buttons for each functionality.

This product shall have a colorful appearance.

This product shall be attractive to all audiences.

This product shall be easy to use for non-proficient English speakers.

2.1.2.3.2 Usability requirements

1. Ease of use: The program should be designed in a way that it is easy to use and understand.
2. User-friendly interface: The interface should be visually appealing and user-friendly, with clear and intuitive navigation.
3. Accessibility: The program should be accessible to people with disabilities, following accessibility guidelines.
4. Error handling: The program should handle errors in a way that is clear and understandable for the user.
5. Performance: The program should be fast and responsive, with a quick response time for user interactions.
6. Help and support: The program should provide help and support options for users who need assistance.

2.1.2.3.3 Security requirements

it is recommended to keep the OpenAI API key secure and prevent unauthorized access. Additionally, while using speech recognition and text to speech APIs, it's crucial to ensure that the data processed and transmitted is protected against unauthorized access, theft, or modification.

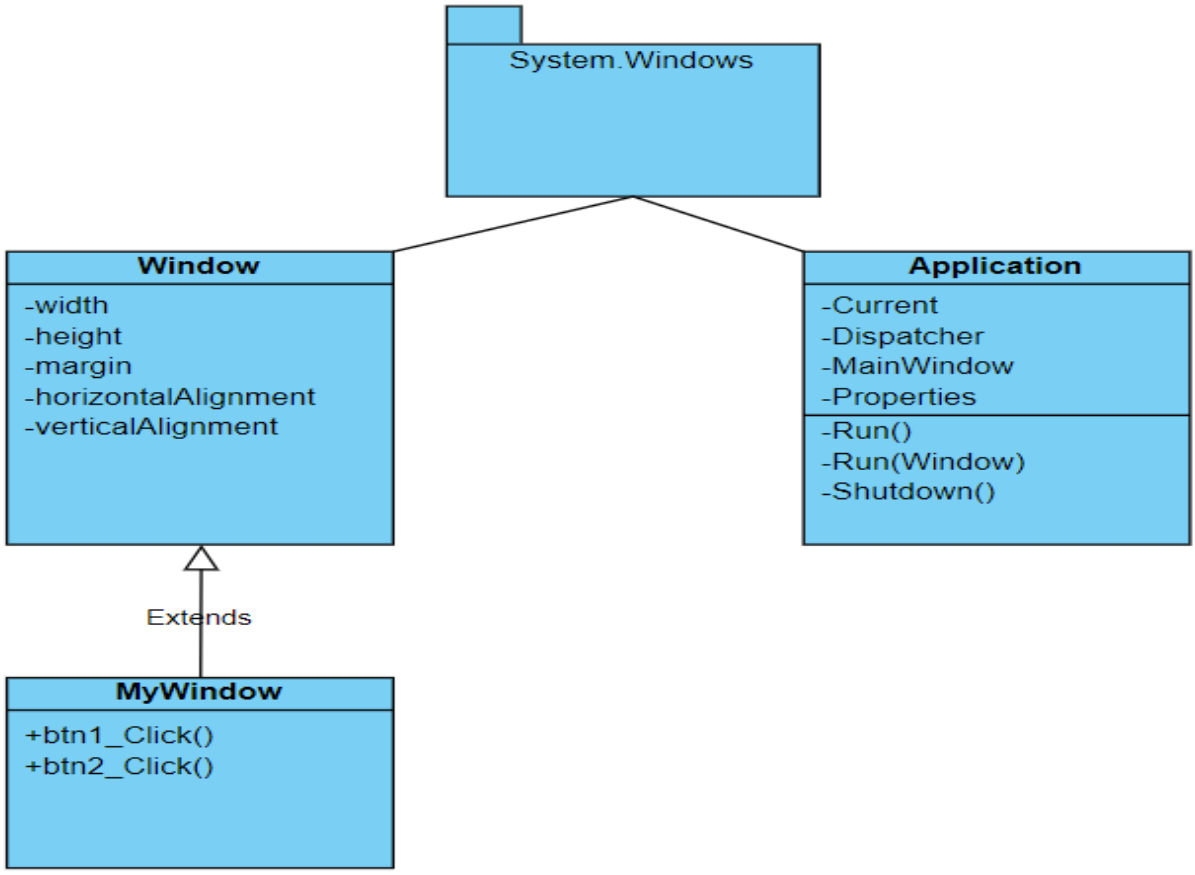
2.1.2.3.4 Performance requirement

the program should have a fast response time, low latency in processing audio and converting it to text, and high accuracy in grammar correction. The program should also be able to handle large amounts of data efficiently, while maintaining stability and reliability.

2.1.2.3.5 Portability requirements

Due to the use of Python in the development of this program, currently it is only able to run on Windows, Linux and macOS.

2.1.3 Class Diagram



2.1.4 Proposed Solutions

1. Voice recognition: The script provides a solution for recognizing and transcribing the user's voice input.
2. Text-to-speech conversion: The script provides a solution for converting text to speech using the win32com library.
3. Voice input to text conversion: The script uses Google's speech recognition API to convert voice input to text.
4. Continuous listening: The script provides a solution for continuously listening to the user's voice input until the user says "exit" or a keyboard interrupt occurs.
5. Grammar correction: The script provides a solution for correcting the grammar of the user's voice input using OpenAI's API.
6. Audio output: The script provides a solution for playing the speech version of the corrected sentence using the playsound library.

2.1.5 Alternative Solutions

1. Using other speech recognition libraries such as CMU Sphinx or Kaldi instead of Google's speech recognition API.
2. Using a different language model for grammar correction other than OpenAI's, for example, a rule-based model or a neural network-based model.
3. Integrating a different text-to-speech library such as pyttsx or nsss instead of Google Text-to-Speech API to generate the audio output.
4. Implementing a different approach for error handling, such as using a logging library to log errors, or utilizing error-handling middleware to catch and handle exceptions.
5. Using a different method for playing the audio output, for example, using PyAudio instead of playsound library.

Chapter 3 DESIGN CONSIDERATIONS

3.1 Design Constraints

3.1.1 Hardware and software environment :

Hardware Requirements:

1. Processor: A modern processor with a clock speed of at least 1 GHz
2. Memory (RAM): At least 4 GB of RAM.
3. Storage: A minimum of 20 GB of free storage space.
4. Operating System: A recent version of Windows, macOS, or Linux.
5. A microphone to capture the user's speech
6. A computer running Python 3 with the required libraries: speech_recognition, win32com, pyaudio openai, gtts, and playsound installed.
7. An internet connection to access the Google Speech API.

Software Requirements:

1. Python 3.x
2. speech_recognition library (can be installed using pip install speech_recognition)
3. win32com library (can be installed using pip install pywin32)
4. openai library (can be installed using pip install openai)
5. gTTS library (can be installed using pip install gtts)
6. playsound library (can be installed using pip install playsound)
7. Internet connection for Google Speech Recognition API and OpenAI API.

3.1.2 End user characteristics :

Anyone looking to hone their English skills through conversation. The users should be comfortable with using a graphical user interface and have a basic understanding of using computer programs.

3.2 Architectural Strategies

3.2.1 Algorithm to be used

This program uses the following algorithms:

1. Speech recognition: The Speech Recognition library is used to recognize speech from a microphone and convert it to text. Google's speech recognition API is also used to recognize the speech.
2. Text-to-Speech conversion: The gTTS library is used to convert text to speech and output the corrected sentence in speech form.
3. Grammar correction: The OpenAI API is used to correct the grammar in the recognized speech input. The OpenAI API uses the "text-davinci-002" engine, which is based on the GPT-3 language model, to correct the grammar.

3.2.2 Reuse of existing software components:

In this program, the following existing software components were reused:

1. Speech Recognition library (sr): The Speech Recognition library is used to recognize speech from a microphone and convert it to text.
2. OpenAI API: The OpenAI API is used to correct the grammar in the recognized speech input.
3. gTTS library: The gTTS library is used to convert text to speech and output the corrected sentence in speech form.
4. playsound library: The playsound library is used to play the MP3 file that contains the corrected sentence in speech form.
5. SAPI.SpVoice: The SAPI.SpVoice module from the win32com library is used to play speech on a Windows operating system.

3.2.4 Development method:

This program was developed in accordance with the Agile method.

3.2.5 Future enhancements/plans

1. Integration of a Natural Language Processing (NLP) model to better understand user inputs and respond accordingly.
2. Adding the ability to interact with the user in a more conversational manner, like asking follow-up questions or clarifying instructions.
3. Implementing multi-language support, allowing users to interact in their preferred language.
4. Incorporating advanced grammar correction algorithms to better handle complex sentences and cases.
5. Adding a Text-to-Speech (TTS) engine that can convert text into multiple languages.
6. Adding the ability to save corrected sentences as text or audio files, allowing users to access the corrected content at a later time.

Chapter 4 SYSTEM DESIGN

4.1 Detailed System Design

4.1.1 Detailed component description

1. Speech Recognition: The program uses the speech_recognition library to capture and recognize speech from the user through the microphone. The speech_recognition library is used to convert audio signals to text.
2. Text-to-Speech Conversion: The program uses the gTTS (Google Text-to-Speech) library to convert text to speech and play it back to the user. The gTTS library generates an MP3 file that is played using the playsound library.
3. Grammar Correction: The program uses the OpenAI API to perform grammar correction on the text generated from the speech recognition. The OpenAI API is used to generate a corrected version of the text, which is then played back to the user using the text-to-speech conversion.
4. User Input: The program continuously listens for user input through the microphone and performs actions based on the input. The user input is processed by the speech recognition component to generate text, which is then used for further processing.
5. speech_recognition library: This library is used to recognize speech from audio files or live audio streams.
6. win32com library: This library is used to perform Text-to-Speech (TTS) conversion in the first script.
7. os library: This library is used to execute shell commands in the first script.
8. openai library: This library is used to perform grammar correction in the second script.
9. gtts library: This library is used to convert text to speech in the second script.
10. playsound library: This library is used to play an audio file in the second script.

Chapter 5 Error Handling

The error handling of the program has been implemented with try-except blocks. The try block contains the code that may raise exceptions, and the except block contains the code that will be executed in case of an exception. In this program, the try block contains the code for recognizing speech using Google's speech recognition API and converting the recognized speech to text.

If there is an error in accessing the API, such as an API key error or a network error, a "API unavailable/unresponsive" message will be printed. If the speech could not be recognized by the API, a "Unable to recognize speech" message will be printed. This way, the program can handle exceptions gracefully and provide informative error messages to the user, making it easier to debug and improve the program.

5.1 Description of error types

1. API unavailable/unresponsive: This error occurs when the Google Speech Recognition API is not accessible.
2. Unable to recognize speech: This error occurs when the speech recognition module is unable to recognize the audio input from the microphone.
3. Exception at runtime: This error occurs when there is an exception during the execution of the program that is not handled by the error handling mechanism.

5.2 Troubleshooting tips

1. Check if the inputs are entered correctly and in the right format
2. Verify that the program has the necessary permissions to run on your device
3. Make sure that all the dependencies and requirements are installed and up to date
4. Restart the program and try again
5. Check the logs for any error messages
6. If the issue persists, try reinstalling the program
7. If the problem still cannot be resolved, reach out to the support team for assistance.

Chapter 6 IMPLEMENTATION &VALIDATION

1. Install the required libraries and dependencies such as SpeechRecognition and win32com.
2. Test the microphone input and make sure it is working properly.
3. Test the text-to-speech functionality and make sure the output is clear and accurate.
4. Test the speech recognition functionality and make sure the system is able to accurately transcribe what is being said.
5. Test the exit function and make sure the program terminates as expected when the user says "exit".
6. Test the program in different noise environments to ensure it is able to adjust to ambient noise levels.
7. Test the OpenAI functionality and make sure the system is able to correct the grammar of the input.
8. Test the saved audio file to make sure it plays correctly and the output is clear and accurate.
9. Test the program with different accents to make sure it is able to transcribe speech accurately in various accents.

Appendix A CODE

GUI:

```
import wpf
import subprocess
import os
from System.Windows import Application, Window
class MyWindow(Window):
def __init__(self):
wpf.LoadComponent(self, 'GRAD2.xaml')
def btn1_Click(self, sender, e):
subprocess.call(["python", "script1.py"])
pass
def btn2_Click(self, sender, e):
subprocess.call(["python", "script2.py"])
pass
if __name__ == '__main__':
Application().Run(MyWindow())
```

==========

Script 1:

```
import speech_recognition as sr
import os
r = sr.Recognizer()
def speak_text_cmd(cmd):
from win32com.client import Dispatch
speak = Dispatch("SAPI.SpVoice")
speak.Speak(cmd)
while True:
    try:
    with sr.Microphone() as source:
    r.adjust_for_ambient_noise(source, duration=0.2)
    audio = r.listen(source)
    MyText = r.recognize_google(audio)
    MyText = MyText.lower()
    print(f"You said: {MyText}\n")
    if "exit" in MyText:
        speak_text_cmd("It was nice talking to you, Goodbye!")
        break
    else:
        speak_text_cmd("You said: ")
        speak_text_cmd(MyText)

    except sr.RequestError as e:
    print("API unavailable/unresponsive")
    except sr.UnknownValueError:
    print("Unable to recognize speech")
```

==========

Script 2:

```
import speech_recognition as sr
import openai
import gtts
from playsound import playsound
openai.api_key = "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"
def grammar_correct(sentence):
    completions = openai.Completion.create(
        engine="text-davinci-002",
        prompt=(f"Please correct the grammar of this sentence: {sentence}"),
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.5,)
    message = completions.choices[0].text
    return message
r = sr.Recognizer()
with sr.Microphone() as source:
print("Say something!")
audio = r.listen(source)
voice_input = r.recognize_google(audio)
corrected_sentence = grammar_correct(voice_input)
print(f"Original sentence: {voice_input}")
print(f"Corrected sentence: {corrected_sentence}")
tts = gtts.gTTS(corrected_sentence)
tts.save("corrected_sentence.mp3")
playsound("corrected_sentence.mp3")
```

==========

==========

Appendix References

speech_recognition: https://pypi.org/project/SpeechRecognition/

win32com: https://docs.microsoft.com/en-us/windows/win32/com/

os: https://docs.python.org/3/library/os.html

openai: https://beta.openai.com/docs/

gtts: https://pypi.org/project/gTTS/

playsound: https://pypi.org/project/playsound/

ironpython: https://ironpython.net/documentation/