TEXT TO SPEECH APPLICATION

Title Page

- **Title:** Text to Speech Application
- **Name:** Roshika R
- **Internship Period:** June 2024 - July 2024
- **Company:** Systecks Solutions Virtual internship
- **Department:** Software Development
- **Submission Date:** July 8, 2024

Table of Contents

Acknowledgements

I would like to express my gratitude to Systecks Solutions Virtual internship, for invaluable guidance and support.

Introduction

The Text to Speech (TTS) application is designed to convert written text into spoken words. This technology is useful in various fields such as accessibility for the visually impaired, language learning, and automated customer service. This project leverages the Python programming language and its libraries to create a user-friendly TTS application with a graphical user interface (GUI).

Objective

The main objective of this project is to develop a TTS application that allows users to input text and hear it spoken aloud. The application should provide options to select voice gender, adjust speech rate, volume, and pitch, and offer playback controls such as play, pause, stop, and replay. Additionally, users should be able to download the generated speech as an MP3 file.

Literature Review

Text to Speech technology has been an area of active research and development for several decades. It involves the conversion of text into spoken voice output. Early TTS systems used concatenative synthesis, which involves stitching together pre-recorded speech segments. Modern TTS systems often use parametric synthesis or deep learning-based approaches to generate more natural-sounding speech.

Methodology

Tools and Technologies Used

- **Programming Language:** Python
- **Libraries:** Tkinter, Pyttsx3, PIL (Pillow), Pygame
- **Integrated Development Environment (IDE):** Visual Studio Code, PyCharm

System Architecture

The system architecture consists of the following components:

1. **GUI:** Built using Tkinter, providing an interface for user interaction.
2. **Speech Engine:** Pyttsx3 library for text to speech conversion.
3. **Audio Playback:** Pygame library for controlling audio playback.
4. **File Handling:** Standard Python libraries for saving and retrieving audio files.

Functionality and Features

- **Text Input:** Text area for users to input the text to be converted to speech.
- **Voice Selection:** Combobox to select the gender of the voice (Male/Female).
- **Speech Control:** Options to adjust speech rate, volume, and pitch.
- **Playback Control:** Buttons to play, pause, stop, and replay the audio.
- **Download Option:** Button to download the generated speech as an MP3 file.

Implementation

Imports and Initialization

```
import tkinter as tk
from tkinter import *
from tkinter.ttk import Combobox
from tkinter import filedialog
from PIL import Image, ImageTk
import pyttsx3
import os
import pygame
import time

root = Tk()
combobox = Combobox(root)
root.title("Text to Speech")
root.geometry("900x450+200+200")
root.resizable(False, False)
root.configure(bg='#a5a4a4')

engine = pyttsx3.init()
is_paused = False
is_stopped = True
audio_file = 'text.mp3'

pygame.mixer.init()
Function Definitions:
def select_voice(gender):
    voices = engine.getProperty('voices')
```

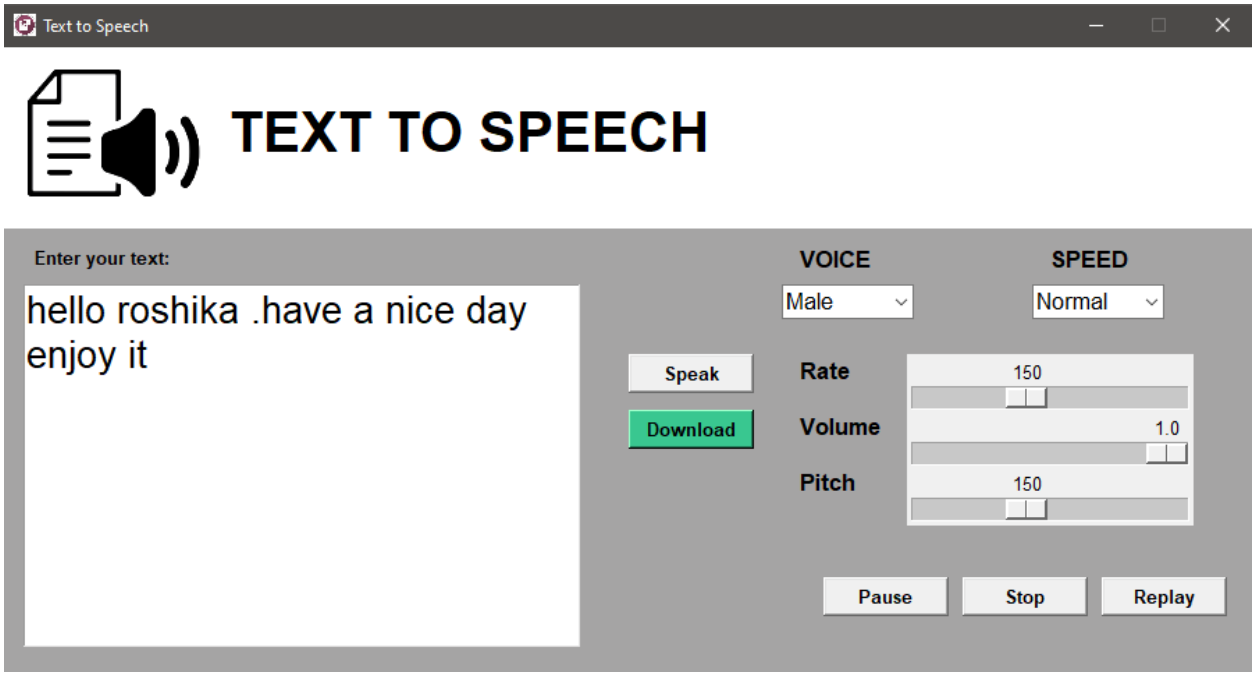
```
    for voice in voices:
        if gender == 'Male' and 'male' in voice.name.lower():
            engine.setProperty('voice', voice.id)
            return
        elif gender == 'Female' and 'female' in voice.name.lower():
            engine.setProperty('voice', voice.id)
            return

def set_speech_properties(gender, speed, rate, volume, text):
    select_voice(gender)
    engine.setProperty('rate', rate)
    engine.setProperty('volume', volume)

    if speed == "Fast":
        engine.setProperty('rate', 250)
    elif speed == 'Normal':
        engine.setProperty('rate', 150)
    else:
        engine.setProperty('rate', 60)

    engine.save_to_file(text, audio_file)
    engine.runAndWait()

def play_audio():
    pygame.mixer.music.load(audio_file)
    pygame.mixer.music.play()
```

```
def speaknow():
    global is_paused, is_stopped
    text = text_area.get(1.0, END).strip()
    gender = gender_combobox.get()
    speed = speed_combobox.get()
    rate = rate_slider.get()
    volume = volume_slider.get()

    if text:
        is_paused = False
        is_stopped = False
        set_speech_properties(gender, speed, rate, volume, text)
        play_audio()

def pause():
    global is_paused
    if not is_paused:
        pygame.mixer.music.pause()
        is_paused = True
    else:
        pygame.mixer.music.unpause()
        is_paused = False

def stop():
    global is_paused, is_stopped
```

```
    pygame.mixer.music.stop()
    is_paused = False
    is_stopped = True

def replay():
    if not pygame.mixer.music.get_busy():
        return
    current_time = pygame.mixer.music.get_pos() / 1000  # in seconds
    start_time = max(0, current_time - 15)
    pygame.mixer.music.play(start=start_time)
    time.sleep(15)  # wait for 15 seconds of playback
    pygame.mixer.music.stop()

def download():
    text = text_area.get(1.0, END).strip()
    gender = gender_combobox.get()
    speed = speed_combobox.get()
    rate = rate_slider.get()
    volume = volume_slider.get()

    if text:
        path = filedialog.askdirectory()
        if path:
            os.chdir(path)
            engine.setProperty('rate', rate)
            engine.setProperty('volume', volume)
```

```
    if speed == "Fast":
        engine.setProperty('rate', 250)
    elif speed == 'Normal':
        engine.setProperty('rate', 150)
    else:
        engine.setProperty('rate', 60)

    engine.save_to_file(text, 'text.mp3')
    engine.runAndWait()
```

GUI Design




```
image_icon = PhotoImage(file="download.png")
root.iconphoto(False, image_icon)

Top_frame = Frame(root, bg="white", width=900, height=130)
```

```
Top_frame.place(x=0, y=0)


Logo = PhotoImage(file="new.png")

Label(Top_frame, image=Logo, bg="white").place(x=5, y=5)

Label(Top_frame, text="TEXT TO SPEECH", font="arial 30 bold", bg="white",
fg="black").place(x=160, y=35)


Label(root, text="Enter your text:", font="arial 10 bold", bg="#a5a4a4", fg="black").place(x=20, y=140)


text_area = Text(root, font="Roboto 20", bg="white", relief=GROOVE, wrap=WORD)

text_area.place(x=15, y=170, width=400, height=260)


Label(root, text="VOICE", font="arial 13 bold", bg="#a5a4a4", fg="black").place(x=570, y=140)

Label(root, text="SPEED", font="arial 13 bold", bg="#a5a4a4", fg="black").place(x=750, y=140)


gender_combobox = Combobox(root, values=['Male', 'Female'], font="arial 13", state='readonly',
width=8)

gender_combobox.place(x=560, y=170)

gender_combobox.set('Male')


speed_combobox = Combobox(root, values=['Fast', 'Normal', 'Slow'], font="arial 13", state='readonly',
width=8)

speed_combobox.place(x=740, y=170)

speed_combobox.set('Normal')


btn = Button(root, text="Speak", compound=LEFT, width=10, font="arial 10 bold",
command=speaknow)

btn.place(x=450, y=220)
```

```
pause_btn = Button(root, text="Pause", compound=LEFT, width=10, font="arial 10 bold", command=pause)

pause_btn.place(x=590, y=380)


stop_btn = Button(root, text="Stop", compound=LEFT, width=10, font="arial 10 bold", command=stop)

stop_btn.place(x=690, y=380)


replay_btn = Button(root, text="Replay", compound=LEFT, width=10, font="arial 10 bold", command=replay)

replay_btn.place(x=790, y=380)


save = Button(root, text="Download", compound=LEFT, width=10, bg="#39c790", font="arial 10 bold", command=download)

save.place(x=450, y=260)
```

6. Testing

Testing Process

Testing the Text to Speech (TTS) application involved several key scenarios to ensure functionality and usability across different parameters.

1. **Input Validation Testing:**
    - **Objective:** Ensure the application handles various types of input, including text length, special characters, and formatting.
    - **Method:** Input different types of texts into the application, such as short sentences, paragraphs, and texts with special characters.
    - **Result:** The application successfully converts different inputs into speech without errors.
2. **Voice and Audio Quality Testing:**
    - **Objective:** Evaluate the quality of speech synthesis for different voices, speeds, volumes, and pitches.
    - **Method:** Adjust voice settings and listen to the synthesized speech.
    - **Result:** Speech quality is clear and natural across different settings, providing satisfactory user experience.
3. **Playback Control Testing:**

    - **Objective:** Test the functionality of playback controls (play, pause, stop, replay).
    - **Method:** Trigger each control during speech synthesis and observe the application's response.
    - **Result:** Controls function correctly, allowing users to pause, resume, stop, and replay speech as intended.
4. **Download Functionality Testing:**
    - **Objective:** Verify the download feature for saving synthesized speech as an audio file.
    - **Method:** Input text, adjust settings, and download the synthesized speech.
    - **Result:** The downloaded audio file is saved in the specified location and plays back correctly outside the application.

Test Results

- **Input Validation:** All types of text inputs were successfully processed without causing application errors.
- **Voice and Audio Quality:** Speech synthesis quality met expectations across different voice settings.
- **Playback Controls:** All controls responded promptly and accurately during speech synthesis.
- **Download Functionality:** The download feature reliably saved synthesized speech as audio files.

7. Results

Project Outcomes

The Text to Speech (TTS) application achieved the following outcomes:

- **Successful Development:** The application was successfully developed using Python and Tkinter for the GUI, Pyttsx3 for text-to-speech conversion, and Pygame for audio playback.
- **Functional Features:** Implemented features include text input, voice selection (male/female), speed adjustment (fast/normal/slow), volume control, pitch adjustment, playback controls (play, pause, stop, replay), and download functionality.
- **User Experience:** The application provides a user-friendly interface with intuitive controls for adjusting speech settings and managing playback.

8. Conclusion

The Text to Speech (TTS) application demonstrates effective implementation of text-to-speech technology, providing a practical tool for converting written text into spoken words. Through the development process, valuable insights were gained into software design, GUI development, speech synthesis techniques, and audio management. The project successfully met its objectives

of creating a functional and user-friendly application suitable for various applications including accessibility tools, language learning aids, and automated systems.