PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

NAME : **J.Abishek**

123012019002

M.Gokul

123012019011

S.Elaikiya

123011019009

CLASS : **B.TECH CSE (AI & ML)**
II yr B sec

COURSE NAME : **Introduction to AI**

SUBJECT CODE. : **XCSHA1**

Enhanced Assistant with Text-to-Speech (TTS)

---

1. Introduction

The program is designed to function as an enhanced assistant that provides various utilities like fetching the current date and time, performing web searches, retrieving Wikipedia topic details, performing basic arithmetic calculations, and utilizing text-to-speech (TTS) capabilities. The program employs the `pyttsx3` library for speech synthesis, which reads out the results of operations in a selected language. The graphical user interface (GUI) is built using `tkinter`, making it user-friendly and interactive.

---

2. Key Features

1. **Date and Time Retrieval**
   - The program retrieves the current date and time and displays it in the output box.
   - The assistant also speaks the current date and time out loud.
2. **Web Search**
   - Allows users to input a search query, and the program performs a web search using Google.
   - The assistant then reads the search query aloud after displaying the result.
3. **Wikipedia Topic Details**
   - The program fetches a summary of a given Wikipedia topic using the Wikipedia API.
   - The details of the topic are displayed in the output box, and the assistant speaks the text out loud.
4. **Basic Calculator**
   - The program evaluates simple mathematical expressions input by the user (e.g., `5+3*2`).
   - The result is displayed, and the assistant reads out the calculated result.
5. **Text-to-Speech**
   - A general Text-to-Speech feature where any user input text is spoken aloud.
   - The user can select the language for TTS (currently supporting "English").

---

3. Functional Details

- **GUI Setup**: The GUI layout includes an input field for the user to enter commands or queries. It contains buttons for triggering different functionalities like fetching date/time, performing web searches, fetching topic details, calculating expressions, and using text-to-speech. An output box displays the results, and a footer provides credits.
- **Text-to-Speech**: The `pyttsx3` engine is used to convert text into speech. The user can select the language from a dropdown menu. Currently, only "English" is supported in terms of voice selection, but the program is extensible to include more languages.

Program

```
import datetime
import webbrowser
import requests
import pyttsx3
from tkinter import *
from tkinter import messagebox, ttk

# Initialize TTS engine
engine = pyttsx3.init()
available_languages = {
   "English": "com.apple.speech.synthesis.voice.alex",

}

# Function to get current date and time
def get_date_time():
   now = datetime.datetime.now()
   text = f"Current Date & Time: {now.strftime('%Y-%m-%d %H:%M:%S')}"
   output_box.delete(1.0, END)
   output_box.insert(END, text + "\n")  # First print the output
   root.after(500, lambda: speak_text(text))  # Then speak after a slight delay


# Function to search the web
def open_web_search():
   query = user_input.get()
   if query:
      webbrowser.open(f"https://www.google.com/search?q={query}")
      text = f"Searching the web for: {query}"
      output_box.delete(1.0, END)
      output_box.insert(END, text + "\n")  # First print the output
      root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
   else:
      messagebox.showwarning("Input Error", "Please enter a query to search.")
```

```
# Function to fetch topic details from Wikipedia
def get_topic_details():
    topic = user_input.get()
    if topic:
        try:
            topic_formatted = topic.replace(" ", "_")
            url = f"https://en.wikipedia.org/api/rest_v1/page/summary/{topic_formatted}"
            response = requests.get(url)
            if response.status_code == 200:
                data = response.json()
                title = data.get("title", "Unknown Topic")
                extract = data.get("extract", "No details available.")
                text = f"Title: {title}\n\nDetails: {extract}"
                output_box.delete(1.0, END)
                output_box.insert(END, text + "\n")  # First print the output
                root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
            elif response.status_code == 404:
                text = f"Error: No details found for '{topic}'. Try another topic or search online."
                output_box.delete(1.0, END)
                output_box.insert(END, text + "\n")  # First print the output
                root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
            else:
                text = "Error: Unable to fetch topic details."
                output_box.delete(1.0, END)
                output_box.insert(END, text + "\n")  # First print the output
                root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
        except Exception as e:
            text = f"Error: {str(e)}"
            output_box.delete(1.0, END)
            output_box.insert(END, text + "\n")  # First print the output
            root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
    else:
        messagebox.showwarning("Input Error", "Please enter a topic to get details.")


# Function to calculate basic arithmetic
def calculator():
    expression = user_input.get()
    try:
        result = eval(expression)
        text = f"Result of {expression} = {result}"
        output_box.delete(1.0, END)
        output_box.insert(END, text + "\n")  # First print the output
        root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
    except Exception as e:
        text = f"Error: Invalid expression. Details: {str(e)}"
        output_box.delete(1.0, END)
        output_box.insert(END, text + "\n")  # First print the output
        root.after(500, lambda: speak_text(text))  # Then speak after a slight delay
```

```
# Text-to-Speech Functionality
def speak_text(text):
    selected_language = language_selector.get()
    if selected_language in available_languages:
        try:
            # Set the language for TTS
            engine.setProperty("voice", available_languages[selected_language])
            engine.say(text)
            engine.runAndWait()
        except Exception as e:
            messagebox.showerror("TTS Error", f"Error setting language: {str(e)}")
    else:
        messagebox.showerror("Language Error", "Selected language is not supported.")


# GUI setup
root = Tk()
root.title("Enhanced Assistant with TTS")
root.geometry("600x700")
root.resizable(False, False)

# Header
header_label = Label(root, text="Enhanced Assistant with TTS", font=("Arial", 18, "bold"),
bg="lightblue", pady=10)
header_label.pack(fill=X)

# Labels and Input Fields
Label(root, text="Enter Input:", font=("Arial", 12)).pack(pady=5)
user_input = Entry(root, width=50, font=("Arial", 12))
user_input.pack(pady=5)

# Dropdown for language selection
Label(root, text="Select Language for TTS:", font=("Arial", 12)).pack(pady=5)
language_selector = ttk.Combobox(root, values=list(available_languages.keys()), font=("Arial", 12),
state="readonly")
language_selector.set("English")  # Default language
language_selector.pack(pady=5)

# Buttons for Functionalities
Button(root, text="Get Date & Time", font=("Arial", 12), command=get_date_time,
bg="lightgreen").pack(pady=5)
Button(root, text="Web Search", font=("Arial", 12), command=open_web_search,
bg="lightblue").pack(pady=5)
Button(root, text="Get Topic Details", font=("Arial", 12), command=get_topic_details,
bg="lightyellow").pack(pady=5)
Button(root, text="Calculator (e.g., 5+3*2)", font=("Arial", 12), command=calculator,
bg="lightpink").pack(pady=5)
Button(root, text="Text-to-Speech", font=("Arial", 12), command=lambda:
speak_text(user_input.get()), bg="lightgray").pack(pady=5)
```

```
# Output Box
Label(root, text="Output:", font=("Arial", 12)).pack(pady=5)
output_box = Text(root, width=70, height=10, font=("Arial", 12), wrap=WORD)
output_box.pack(pady=10)

# Footer
footer_label = Label(root, text="Assistant with TTS - By You", font=("Arial", 10), bg="lightblue",
pady=5)
footer_label.pack(fill=X)

# Run the GUI application
root.mainloop()
```

4. Example Outputs

1. Current Date & Time

- **Input**: User presses the "Get Date & Time" button.
- **Output Display**:

  ```
  Current Date & Time: 2024-11-21 15:45:32
  ```

- **Text-to-Speech Output**: The assistant reads: "Current Date and Time: 2024-11-21 15:45:32"

2. Web Search

- **Input**: User enters "Python programming" and clicks the "Web Search" button.
- **Output Display**:

  ```
  Searching the web for: Python programming
  ```

- **Text-to-Speech Output**: The assistant reads: "Searching the web for Python programming."

3. Wikipedia Topic Details

- **Input**: User enters "Artificial Intelligence" and clicks the "Get Topic Details" button.
- **Output Display**:

  ```
  Title: Artificial intelligence

  Details: Artificial intelligence (AI) is intelligence demonstrated by
  machines, in contrast to the natural intelligence displayed by humans
  and animals.
  ```

- **Text-to-Speech Output**: The assistant reads: "Title: Artificial intelligence. Details: Artificial intelligence (AI) is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and animals."

4. Calculator

- **Input**: User enters "5+3*2" and clicks the "Calculator" button.
- **Output Display**:

```
Result of 5+3*2 = 11
```

- **Text-to-Speech Output**: The assistant reads: "Result of 5+3*2 equals 11."

5. Text-to-Speech (TTS)

- **Input**: User enters "Hello, how are you?" and clicks the "Text-to-Speech" button.
- **Output Display**:

```
Hello, how are you?
```

- **Text-to-Speech Output**: The assistant reads: "Hello, how are you?"

---

5. Challenges & Considerations

- **Language Support**: Currently, only English is supported. To improve this, the program can be expanded by adding more language options and voices available in `pyttsx3`.
- **Error Handling**: The program does a good job of catching errors in most scenarios, such as invalid expressions for the calculator and failed API requests. However, users should be made aware of possible errors (e.g., network issues) with more descriptive error messages.
- **Text Length**: When fetching long Wikipedia summaries or dealing with large outputs, the assistant reads the text as a whole. This can cause problems with long texts. A possible enhancement could be splitting the text into chunks and reading them one after another.

---

6. Future Improvements

- **Multi-language Support**: Extend the available voices and languages in the Text-to-Speech section to cater to more users.
- **Interactive Features**: Integrate more interactive features such as a to-do list, calendar integration, or a reminder system.
- **More Robust Error Handling**: Add better handling for situations where the API is down, or the text-to-speech engine is unavailable.
- **Advanced GUI Design**: Enhance the layout, perhaps adding tabs or collapsible panels to better organize features.

---

7. Conclusion

This program provides a comprehensive set of tools for enhancing productivity and interaction with the user, using text-to-speech and a variety of utilities such as web search, date/time retrieval, Wikipedia topic fetching, and basic arithmetic calculation. It combines accessibility through TTS with powerful APIs, making it a valuable tool for users looking for a multi-functional assistant.