PROJECT REPORT

On

Text-to-Speech and YouTube Transcript for Note-Taking Enhancement in Various Applications

(CSE IVth Semester Mini project)

2023-2024

Submitted to:

Ms. Shagun Dasawat
(Graphic Era Hill University)

Submitted by:

Mr. Shriyansh Negi
Roll. No :- 2219687

Class Roll No. :- 65

Section:- F1

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

CERTIFICATE

This is to certify that **SHRIYANSH NEGI** successfully completed the project titled "**Text-to-Speech and YouTube Transcript for Note-Taking Enhancement in Various Applications"** as a part of the B. Tech CSE Mini Project at Graphic Era Hill University, Dehradun.

The project report has been examined and evaluated. It is hereby acknowledged that the work presented in this report is original and authentic to the best of our knowledge.

Date: 19.July.2024

ACKNOWLEDGMENT

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of this project and the preparation of this report.

First and foremost, I extend my heartfelt thanks to everyone who supervises, supports, and encourages throughout the duration of the project.

I would also like to acknowledge the support of my family and friends, who stood by me and provided encouragement throughout the project.

Thank you to everyone who played a role, big or small, in the completion of this project. Your support has been invaluable.

SHRIYANSH NEGI

(CSE Graphic Era Hill University, Dehradun)

Mr. Shriyansh Negi

University Roll No.- 2219687

Class Roll No. :- 64

Session: 2023-2024

GEHU, Dehradun

TABLE OF CONTENT

INTRODUCTION

In today's fast-moving world, where technology is super important, it's crucial to be efficient in how we take in information. These summary talks about two cool ideas to make things better for people: first, **automatically getting notes from videos**, and second**, letting messages in apps be turned into spoken words**. These ideas make it easier to take notes, avoid getting overwhelmed with too much info, and make digital interactions better for everyone.

Today, technology drives progress and innovation, shaping how we live and work. It's everywhere, making tasks easier and more efficient. One area ripe for improvement is note-taking. Let's explore how technology can revolutionize this aspect of our lives.
Imagine a scenario where students, professionals, or anyone seeking to absorb information from online content could seamlessly extract key points and insights from video lectures or presentations.

This is where the concept of automatically generating notes from videos comes into play. By including the transcript feature available on platforms like YouTube, users can **effortlessly convert spoken content into written text**, which can then be saved into a convenient file format.

This not only targets the note-taking process but also offers a **flexible solution for documentation purposes**. Additionally, for individuals who prefer reading over watching videos or those with accessibility needs, having access to transcripts provides an invaluable alternative method of consuming content while ensuring that no crucial information is missed.

Now, let's bump into the **potential of enhancing communication** experiences through speech-to-text technology within messaging applications. In today's digital age, we are overwhelmed by a never-ending flow of text-based communication across various platforms such as **WhatsApp**, **Instagram**, **Facebook**, and **Telegram**. People with busy schedules find the act of reading to be time-consuming, and this can lead to information overload

To address this challenge, imagine if users had the option to **transform incoming text messages into spoken words with a simple tap of a button**. This innovative feature, seamlessly integrated into messaging apps, would enable users to listen to the contents of their messages instead of reading them. By activating the text-to-speech function within the app settings, users can **effectively multitask** and **stay informed** while on the go, without the need to dedicate focused attention to reading lengthy messages. This not only enhances user convenience but also promotes accessibility and inclusivity by catering to individuals with visual impairments or those who prefer auditory learning modalities.

In summary, these ideas represent innovative solutions that harness the power of technology to optimize efficiency, streamline workflows, and enhance user experiences across various digital platforms. By embracing these advancements, we can unlock new possibilities for **learning**, **communication**, and **productivity** in the digital age.

INTRODUCTION TO PYTHON

Python is a **high-level**, **interpreted programming language** known for its readability, simplicity, and versatility. It was created by **Guido van Rossum** and first released in 1991. Python has since become one of the most popular programming languages in the world, used for a wide range of applications from web development to data science and artificial intelligence.

Key Features of Python:

1. **Readability and Simplicity**: Python's syntax is designed to be readable and straightforward, making it an excellent choice for beginners and experienced programmers alike.
2. **Interpreted Language**: Python is an interpreted language, which means that code is executed line by line. This allows for interactive testing and debugging of code.
3. **Dynamic Typing**: Python uses dynamic typing, meaning that variables do not need to be declared with a specific type, and the type can change as the program runs.
4. **Extensive Standard Library**: Python comes with a vast standard library that includes modules and packages for handling various tasks, from file I/O to web protocols and data manipulation.
5. **Cross-Platform**: Python is cross-platform and can run on various operating systems, including Windows, macOS, Linux, and more.

Common Uses of Python:

1. **Web Development**: Python frameworks such as Django and Flask are widely used for developing web applications.
2. **Data Science and Machine Learning**: Libraries like NumPy, pandas, SciPy, scikit-learn, and TensorFlow make Python a powerful tool for data analysis, visualization, and machine learning.
3. **Automation and Scripting**: Python is often used for writing scripts to automate repetitive tasks and for system administration.
4. **Software Development**: Python is used in the development of various software applications, from desktop to enterprise-level applications.
5. **Game Development**: Libraries such as Pygame allow for the development of simple games and multimedia applications.

BASIC TERMINOLOGIES OF PYTHON

Variables: -Variables are fundamental to programming in Python. They act as containers for storing data values. In Python, you do not need to declare a variable's type explicitly; the interpreter infers it based on the assigned value. For example:

```
x = 10        # x is an integer
name = "Alice"  # name is a string
```

Data Types: - Python supports several built-in data types, which can be categorized as follows:

Numeric Types:

- Integer (int): Whole numbers, e.g., 5, -3.
- Float (float): Decimal numbers, e.g., 3.14, -0.001.

Text Type:

- String (str): A sequence of characters, e.g., "Hello, World!".

Boolean Type:

- Boolean (bool): Represents truth values, either True or False.

Collection Types:

- List (list): An ordered, mutable collection, e.g., [1, 2, 3].
- Tuple (tuple): An ordered, immutable collection, e.g., (1, 2, 3).
- Set (set): An unordered collection of unique elements, e.g., {1, 2, 3}. Dictionary (dict): A collection of key-value pairs, e.g., {"name": "Alice", "age": 25}.

Script: - A Python script is a file containing Python code that can be executed. Scripts typically have a .py extension. They can be run in various environments, such as command line, IDEs, or Jupyter notebooks.

```
# This is a simple script
print("Hello, World!")
```

Interpreter:- The Python interpreter is a program that reads and executes Python code. It can be run in interactive mode (REPL) or script mode. The interpreter converts Python code into machine code that the computer can understand.

Program: - A program is a collection of instructions that perform a specific task. In Python, programs can range from simple scripts to complex applications.

Function: - Functions are reusable blocks of code that perform a specific task. They can take inputs (parameters) and return outputs. Python has built-in functions, but you can also define your own using the def keyword.

```
def greet(name):
    return f"Hello, {name}!"
```

Module: - A module is a file containing Python code that can define functions, classes, and variables. Modules promote code reuse and organization. You can import a module into your script using the import statement.

```
import math
print(math.sqrt(16))  # Outputs: 4.0
```

Class: - A class is a blueprint for creating objects. It encapsulates data for the object and methods to manipulate that data. Classes are the foundation of object-oriented programming in Python.

```
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        return "Woof!"
```

Object: - An object is an instance of a class. It contains attributes (data) and methods (functions) defined by its class. Objects allow for encapsulation and modularity in programming.

```
my_dog = Dog("Buddy")
print(my_dog.bark())  # Outputs: Woof!
```

METHODOLOGY

The programs utilize various libraries of **Python** (functionality), **Tkinter** (GUI), **exception handling** (managing exceptions), **basic Python**, and **file handling concepts**.
Let's discuss and understand these.

PYTHON LIBRARIES: -

- Python is renowned for its wide collection of libraries that offer pre-written code to perform specific task. In this we don't have to define or declare, we can just call the different function for different tasks as per the requirements.
- Libraries like: - Google_trans, Google Text-to-speech, playsound, etc.

EXCEPTION HANDLING CONCEPT: -

- It concepts for dealing with different types of exceptions that can't be handle locally.
- These are some statements which are used to handle the exception are: - 'try', 'except', 'finally', etc.
- These are used so that error not be show by the program in case of failure. Instead of showing error in the program, the exception handler transfers the control to where the error can be handled using catch and try block/statements.
- Some types of error are: - Value Error, Type Error, 'IOError', etc and these errors are handled appropriately to prevent unexpected termination

BASIC PYTHON CONCEPTS: -

- The program uses all basic concepts of python to achieve its objectives.
- Variables, data type, etc are used to store and manipulation data.
- Control flow statement like if, else, elif, for, and while are used for decision making and iteration.
- Functions are defined and declared to encapsulate the code and to the reusability.

CONCEPT OF FILE HANDLING: -

- File handling plays important role for reading and writing of/to files and interaction with external resources.
- File handling operation like: - writing, closing files, reading, opening etc. are performed using some in-built functions and methods.

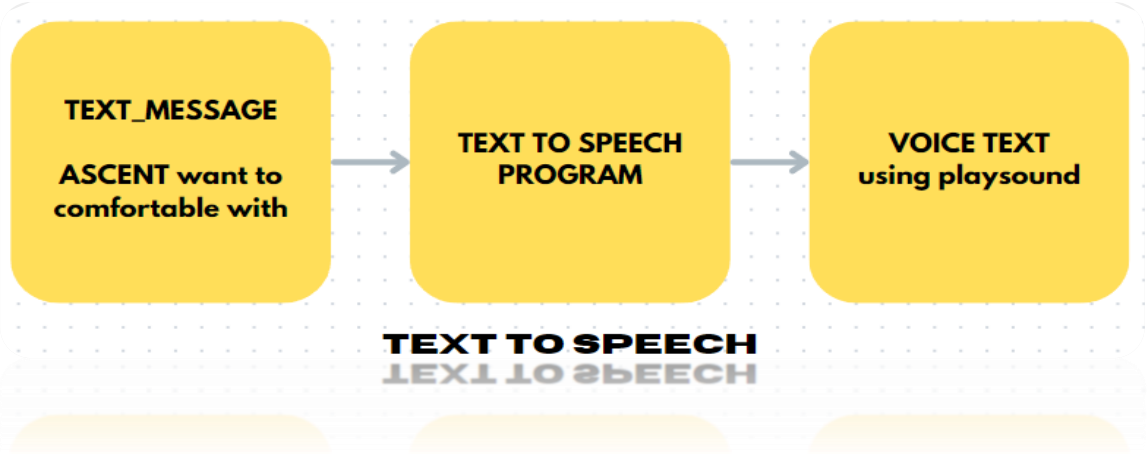
- It can process different formats as per the program’s requirements.

WORKING WITH TKINTER LIBRARIES: -

- The **Tkinter library** is responsible for developing the GUI (**Graphical User Interface**). It improves the user experience while working with the application.
- Concept of **Creating a Menu Bar**: This improves the user experience while using the application.
- Connecting the button with the functions defined by the user so that they can provide **functionality** to the button for which the GUI is designed.

UNDERSTANDING PROJECT MECHANICS

This project consists of **two** features for the user, which also assist in **advancing** their **experience**.



Importing Libraries:

Firstly, we import the necessary libraries to perform our desired tasks. For instance:

- Playsound
- Google_trans_new
- Other user-specified libraries

User Input:

- The program prompts the user to input the text they wish to convert to voice format. After entering the text, if the user does not want to enter anything else, they simply type "done" at the end.

Terminating Input:

- The input function terminates, and the entered text is stored in a variable for further processing.

Language Options Menu:

- Next, the program displays a menu with different language options for the accent in which the text will be spoken.

Function Call:

- After the user selects an accent, the variable containing the text and the chosen accent are passed as arguments to the function named text_to_speech.

Playing the Sound:

- Finally, using the playsound library, the program speaks the text in the selected accent, providing an auditory experience for the user. This structured approach makes it easier to understand the flow of the program and the tasks it performs.

SNIPPET CODE: -

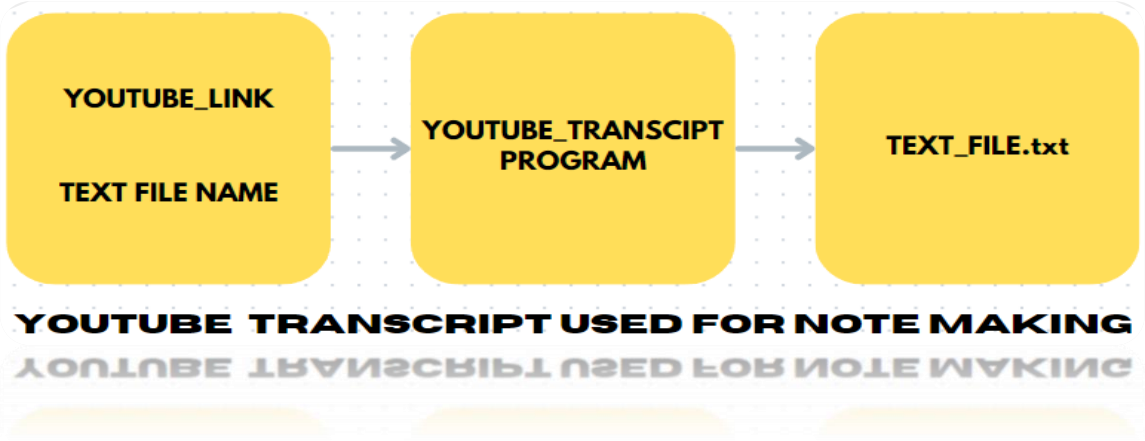
```
def text_to_voice():
    input_text = fn.get()
    text = input_text.strip()
    filename = fl_t.get().strip() + ".mp3"
    if not text:
        tkinter.messagebox.showerror("Error", "Input text cannot be empty.")
        return

    try:
        # Create a gTTS object with the input text
        tts = gTTS(text=text, lang="en", slow=False)
        tts.save(filename)  # Save the audio file

        # Check if the file exists before trying to play it
        if os.path.exists(filename):
            playsound(filename)  # Play the saved audio file
        else:
            tkinter.messagebox.showerror("Error", "Audio file was not created successfully.")
    except Exception as e:
        tkinter.messagebox.showerror("Error", f"Failed to play audio: {str(e)}")
```

LIBRARIES USED:

```
import speech_recognition as sr
from google_trans_new import google_translator
from gtts import gTTS
from playsound import playsound
```



Importing Libraries:

- Firstly, the program imports necessary libraries such as **youtube_transcript_api**. This library is crucial for fetching transcripts from YouTube videos.
- Other essential libraries for file handling and exception management might also be imported to ensure smooth operation.

User Input for YouTube Link:

- The program prompts the user to input the **YouTube video link**. This input is stored in a variable. This step is essential as the link provided will be used to fetch the transcript of the video.

User Input for File Name:

- Next, the program asks the user to input the **name of the file** where the transcript will be saved. This name is stored in another variable.
- This step allows users to specify how they want their transcript file to be named and stored.

Exception Handling for URL Validation:

- Before proceeding with fetching the transcript, the program employs exception handling to validate the YouTube link entered by the user. If the link is invalid or if there's an issue with fetching the transcript, an exception is thrown.
- This ensures that the program can gracefully handle errors and provide feedback to the user.

Fetching the Transcript:

- Using the **youtube_transcript_api**, the program fetches the transcript of the video from the provided YouTube link.
- This API call retrieves the transcript data in a structured format, typically as a list of dictionaries where each dictionary contains a segment of the transcript along with its start time and duration.

Writing the Transcript to a File:

- With the help of file handling concepts, the program writes the fetched transcript into the file specified by the user. The program opens the file in write mode and iterates through the transcript data, writing each segment of the transcript into the file in a readable format.
- This step ensures that the user's transcript is saved accurately and can be accessed later.

SNIPPET CODE: -

```
def youtube():
    video_link = jn.get()
    filename = fl.get().strip() + ".docx"

    if not video_link.strip():
        tkinter.messagebox.showerror("ALERT WINDOW", "VIDEO LINK FIELD CAN'T BE EMPTY....\n\nKINDLY FILL THE VIDEO LINK FIELD")
    elif not filename.strip():
        tkinter.messagebox.showerror("ALERT WINDOW", "FILE NAME FIELD CAN'T BE EMPTY...\n\nKINDLY FILL THE FILE NAME")
    else:
        try:
            srt = YouTubeTranscriptApi.get_transcript(video_link)

            doc = Document()
            doc.add_heading('Notes', level=1)

            for item in srt:
                text = item["text"]
                doc.add_paragraph(text)

            doc.save(filename)
            tkinter.messagebox.showinfo("ALERT WINDOW", "SUCCESSFULLY DOWNLOADED !!!!")
        except Exception as e:
            if "Video unavailable" in str(e):
                tkinter.messagebox.showerror("Error", "Video not found. Please enter the correct link.")
            else:
                tkinter.messagebox.showerror("Error", str(e))
```

LIBRARIES USED:

```
from youtube_transcript_api import YouTubeTranscriptApi
import os
from docx import Document
```

POTENTIAL ADVANCEMENT

- **Limited ascent**: - While the current implementation may have limited accents, there is potential to expand this feature by including a wider range of accents. This could provide users with a more diverse and natural-sounding experience, akin to human speech rather than machine-generated voices.

- **Converting API**:- Once converted into an API (Application Programming Interface), the functionality of the project can be accessed and utilized by different applications and mobile phones. This opens up opportunities for widespread adoption and integration into various platforms and devices, enhancing accessibility and usability.

- **Collaboration with AI and ML**: - Collaborating the project with Artificial Intelligence (AI) and Machine Learning (ML) technologies could further enhance its capabilities. By leveraging AI and ML algorithms, the project could adapt and improve over time, learning from user interactions and feedback to deliver more accurate and personalized speech synthesis. Additionally, AI-powered features such as voice recognition and natural language processing could be integrated to enable more advanced functionalities and interactions.

- **Offline Functionality: -** Local Processing: Enable offline text-to-speech conversion to allow users to access the service without an internet connection.

- **Community Contributions: -** Open Source Development: Consider making the project open source to encourage community contributions and collaborative improvements.

REFERENCE

- https://www.youtube.com/watch?v=-GhzpvvIXlM&list=PLS1QulWo1RIY6fmY_iTjEhCMsdtAjgbZM
- https://www.youtube.com/watch?v=A_F8W3VE3k4
- https://www.geeksforgeeks.org/libraries-in-python/
- https://www.youtube.com/watch?v=eVX0QrvjA5M