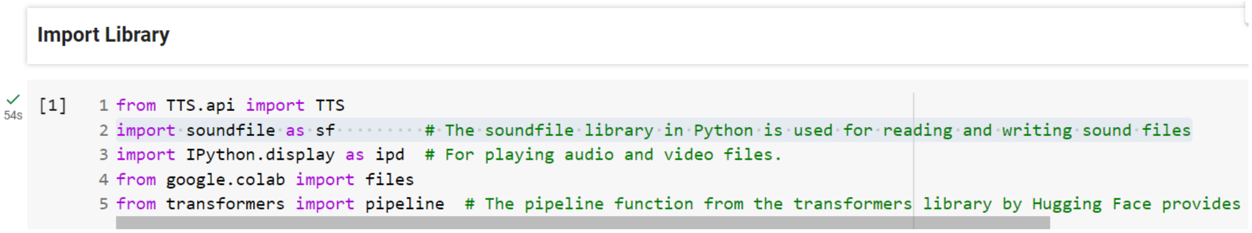
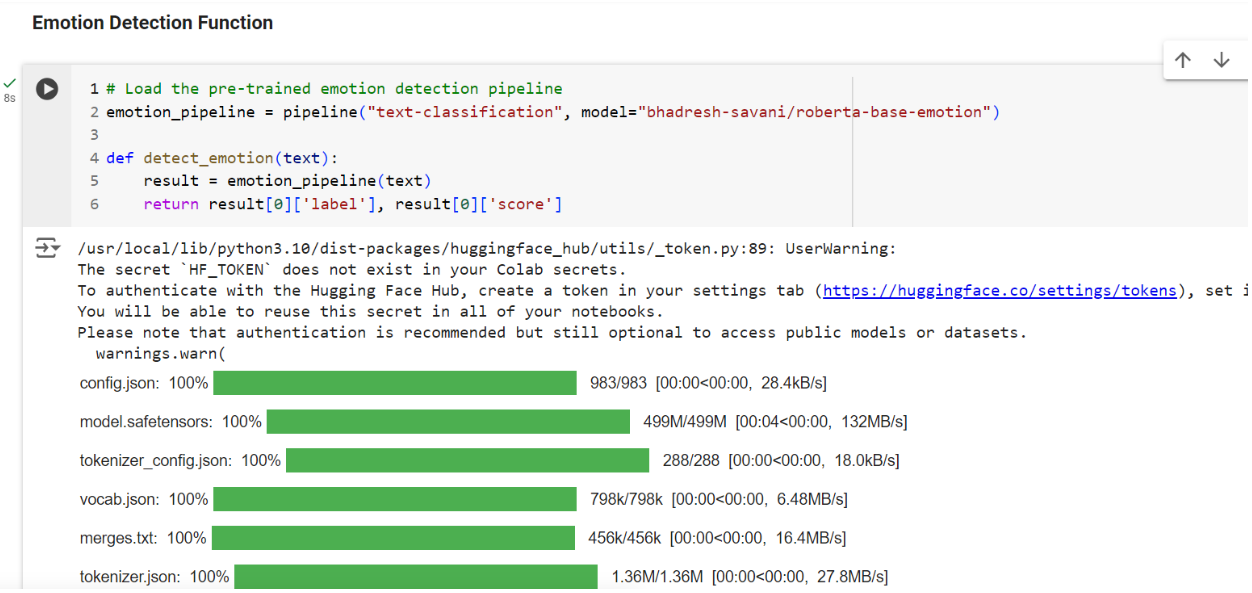
Required Library: The following are necessary libraries for this project

1. **TTS API:** This is an open-source library developed by Coqui for text-to-speech synthesis. It allows you to use pre-trained TTS models or train your models to convert text into spoken audio.
2. The model is trained on the XTTS_V2 dataset, which contains 17 languages.
3. The required model is tactron2, but it is trained on the LJ_speech dataset, which only contains English.
4. **Transformer:** Developed by Hugging Face, this library provides a wide range of pre-trained models for various natural language processing (NLP) tasks, including text classification, translation, and text generation. It's commonly used for models based on the Transformer architecture, such as BERT, GPT, and T5.
5. **IPython.display:** This is used for playing audio and video files.
6. **Soundfile :** The soundfile library in Python is used for reading and writing sound files.



Import Library

```
from TTS.api import TTS
import soundfile as sf        # The soundfile library in Python is used for reading and writing sound files
import IPython.display as ipd  # For playing audio and video files.
from google.colab import files
from transformers import pipeline  # The pipeline function from the transformers library by Hugging Face provides
```

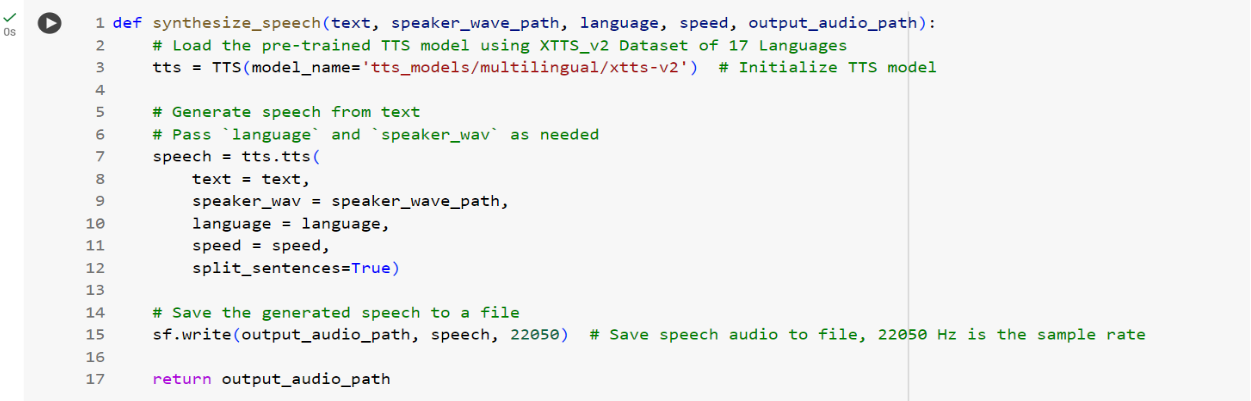


Emotion Detection Function

```
# Load the pre-trained emotion detection pipeline
emotion_pipeline = pipeline("text-classification", model="bhadresh-savani/roberta-base-emotion")

def detect_emotion(text):
    result = emotion_pipeline(text)
    return result[0]['label'], result[0]['score']
```

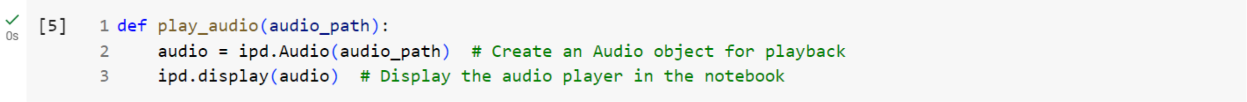
```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set i
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%  983/983 [00:00<00:00, 28.4kB/s]
model.safetensors: 100%  499M/499M [00:04<00:00, 132MB/s]
tokenizer_config.json: 100%  288/288 [00:00<00:00, 18.0kB/s]
vocab.json: 100%  798k/798k [00:00<00:00, 6.48MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 16.4MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 27.8MB/s]
```

Define the synthesize_speech Function



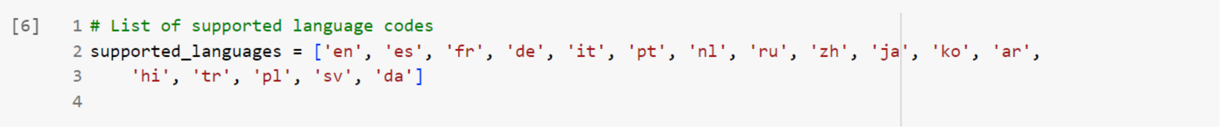
```python
def synthesize_speech(text, speaker_wave_path, language, speed, output_audio_path):
    # Load the pre-trained TTS model using XTTS_v2 Dataset of 17 Languages
    tts = TTS(model_name='tts_models/multilingual/xtts-v2')  # Initialize TTS model

    # Generate speech from text
    # Pass `language` and `speaker_wav` as needed
    speech = tts.tts(
        text = text,
        speaker_wav = speaker_wave_path,
        language = language,
        speed = speed,
        split_sentences=True)

    # Save the generated speech to a file
    sf.write(output_audio_path, speech, 22050)  # Save speech audio to file, 22050 Hz is the sample rate

    return output_audio_path
```

Define play_audio function, when it call it play the audio of text

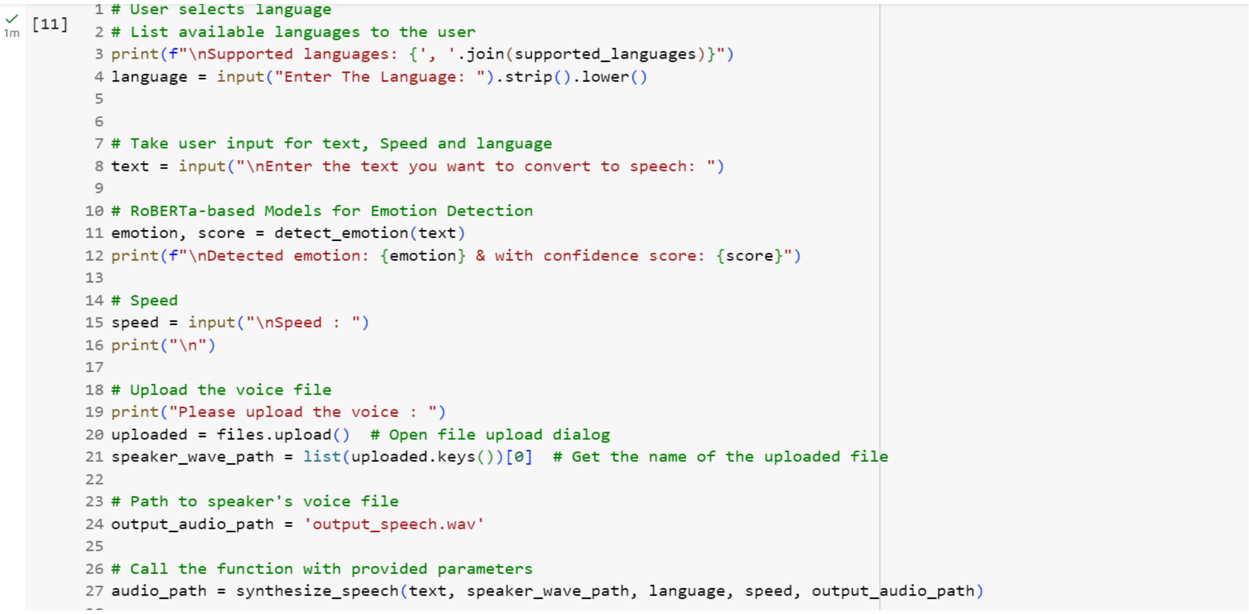


```python
def play_audio(audio_path):
    audio = ipd.Audio(audio_path)  # Create an Audio object for playback
    ipd.display(audio)  # Display the audio player in the notebook
```

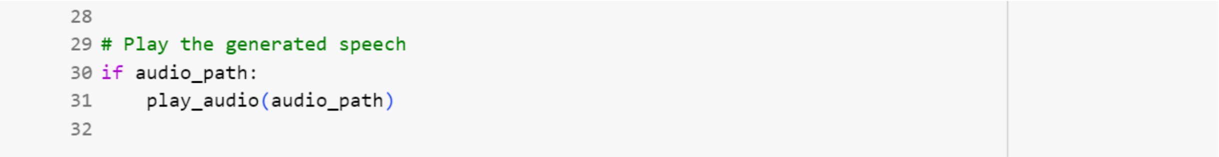
Supported Parameters



```python
# List of supported language codes
supported_languages = ['en', 'es', 'fr', 'de', 'it', 'pt', 'nl', 'ru', 'zh', 'ja', 'ko', 'ar',
    'hi', 'tr', 'pl', 'sv', 'da']

```

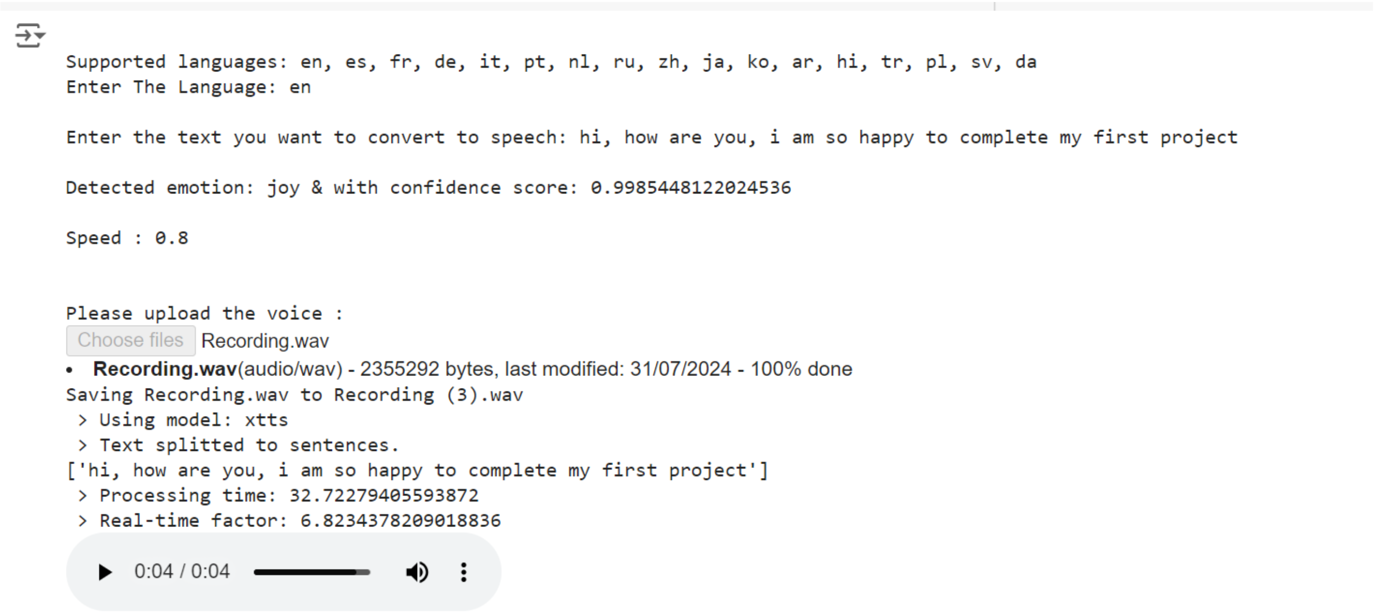


```python
# User selects language
# List available languages to the user
print(f"\nSupported languages: {', '.join(supported_languages)}")
language = input("Enter The Language: ").strip().lower()


# Take user input for text, Speed and language
text = input("\nEnter the text you want to convert to speech: ")

# RoBERTa-based Models for Emotion Detection
emotion, score = detect_emotion(text)
print(f"\nDetected emotion: {emotion} & with confidence score: {score}")

# Speed
speed = input("\nSpeed : ")
print("\n")

# Upload the voice file
print("Please upload the voice : ")
uploaded = files.upload()  # Open file upload dialog
speaker_wave_path = list(uploaded.keys())[0]  # Get the name of the uploaded file

# Path to speaker's voice file
output_audio_path = 'output_speech.wav'

# Call the function with provided parameters
audio_path = synthesize_speech(text, speaker_wave_path, language, speed, output_audio_path)
```



```python

# Play the generated speech
if audio_path:
    play_audio(audio_path)

```

Test Model in English language to generate speech from text:

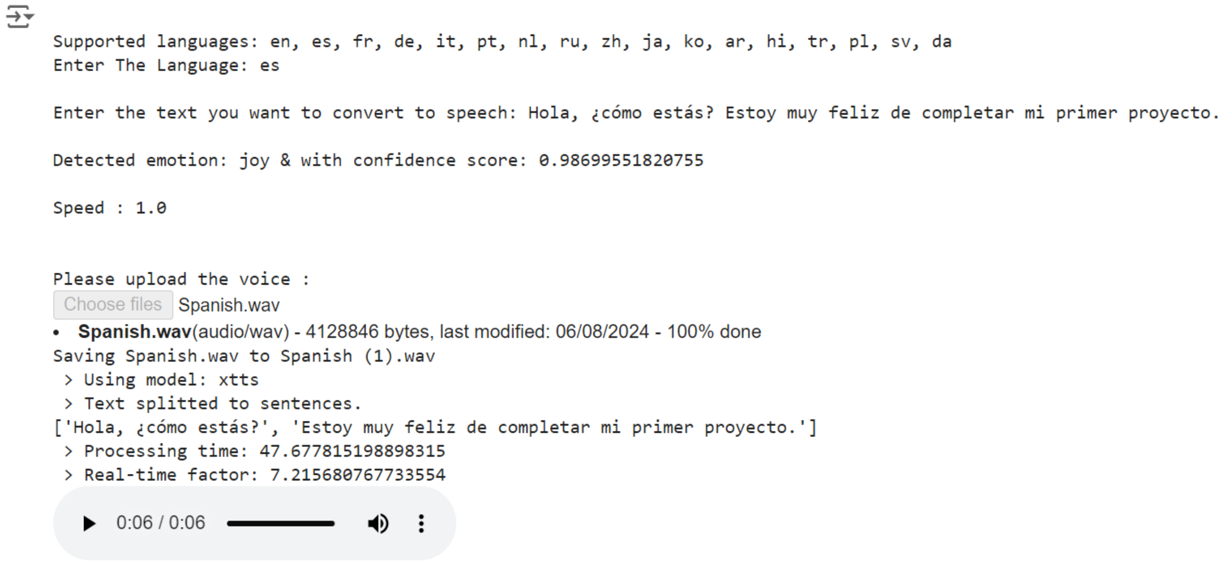


```
Supported languages: en, es, fr, de, it, pt, nl, ru, zh, ja, ko, ar, hi, tr, pl, sv, da
Enter The Language: en

Enter the text you want to convert to speech: hi, how are you, i am so happy to complete my first project

Detected emotion: joy & with confidence score: 0.9985448122024536

Speed : 0.8


Please upload the voice :
```

- **Recording.wav**(audio/wav) - 2355292 bytes, last modified: 31/07/2024 - 100% done

```
Saving Recording.wav to Recording (3).wav
 > Using model: xtts
 > Text splitted to sentences.
['hi, how are you, i am so happy to complete my first project']
 > Processing time: 32.72279405593872
 > Real-time factor: 6.8234378209018836
```

Generate Speech in Spanish from Spanish Text:



```
Supported languages: en, es, fr, de, it, pt, nl, ru, zh, ja, ko, ar, hi, tr, pl, sv, da
Enter The Language: es

Enter the text you want to convert to speech: Hola, ¿cómo estás? Estoy muy feliz de completar mi primer proyecto.

Detected emotion: joy & with confidence score: 0.98699551820755

Speed : 1.0


Please upload the voice :
```

- **Spanish.wav**(audio/wav) - 4128846 bytes, last modified: 06/08/2024 - 100% done

```
Saving Spanish.wav to Spanish (1).wav
 > Using model: xtts
 > Text splitted to sentences.
['Hola, ¿cómo estás?', 'Estoy muy feliz de completar mi primer proyecto.']
 > Processing time: 47.677815198898315
 > Real-time factor: 7.215680767733554
```