Name: Fatima Zahid
Roll Number: BSCS51F22S010
Group Assignment: TTS Module

TTS Module – Project Progress Summary

(Google TTS + Flask)

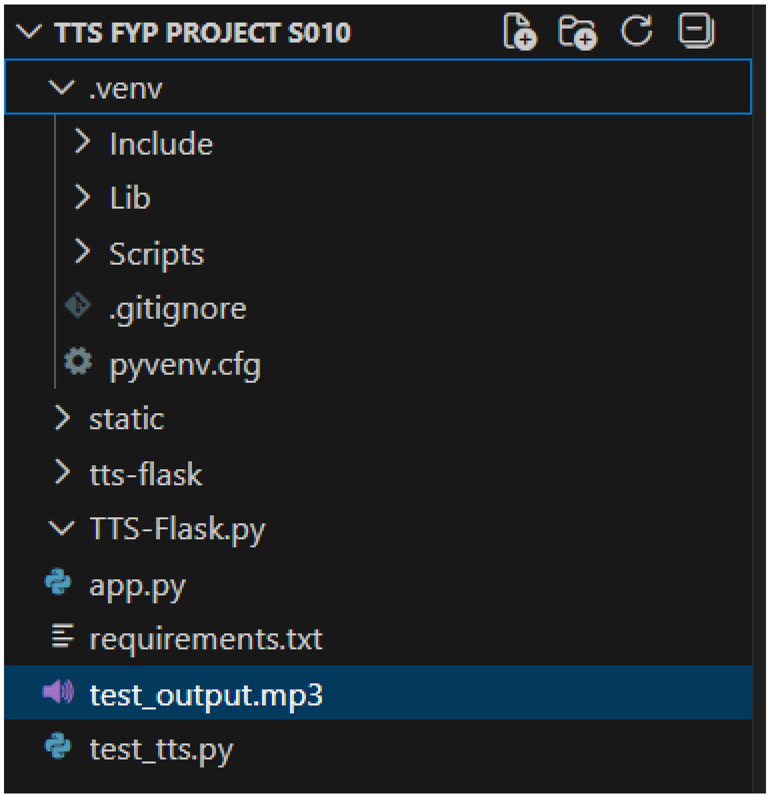
1. Project Setup

- Created a new project folder in VS Code:
  `D:/TTS FYP PROJECT S010`

- Created and activated a **Python virtual environment (.venv)**.

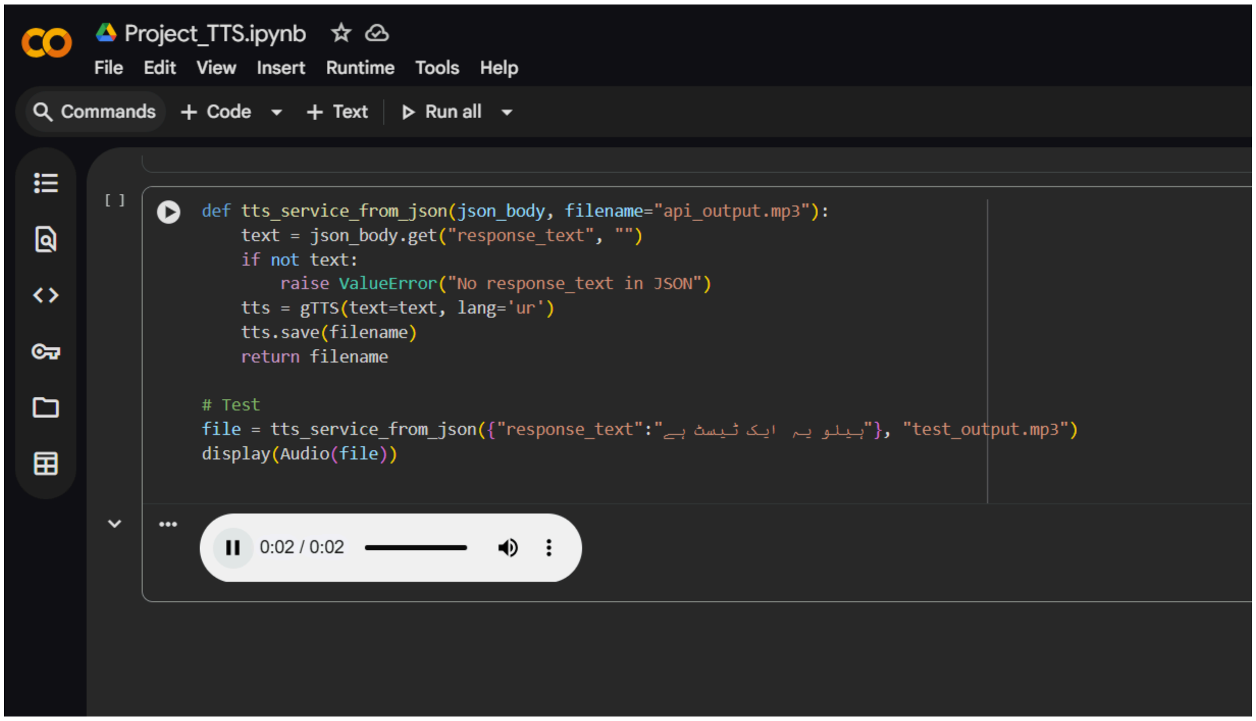
- Installed all required libraries through a `requirements.txt` file, including:
  `gTTS (Google Text-to-Speech)`
  `Flask (for API backend)`
  `Pydub (for audio processing)`

- `Requests (for sending/receiving HTTP requests)`



2. Implemented Google Text-to-Speech (gTTS)

- Wrote a small Python script to test Google gTTS.
- Successfully generated audio directly from text.
- Verified that MP3 files were created inside the project folder.

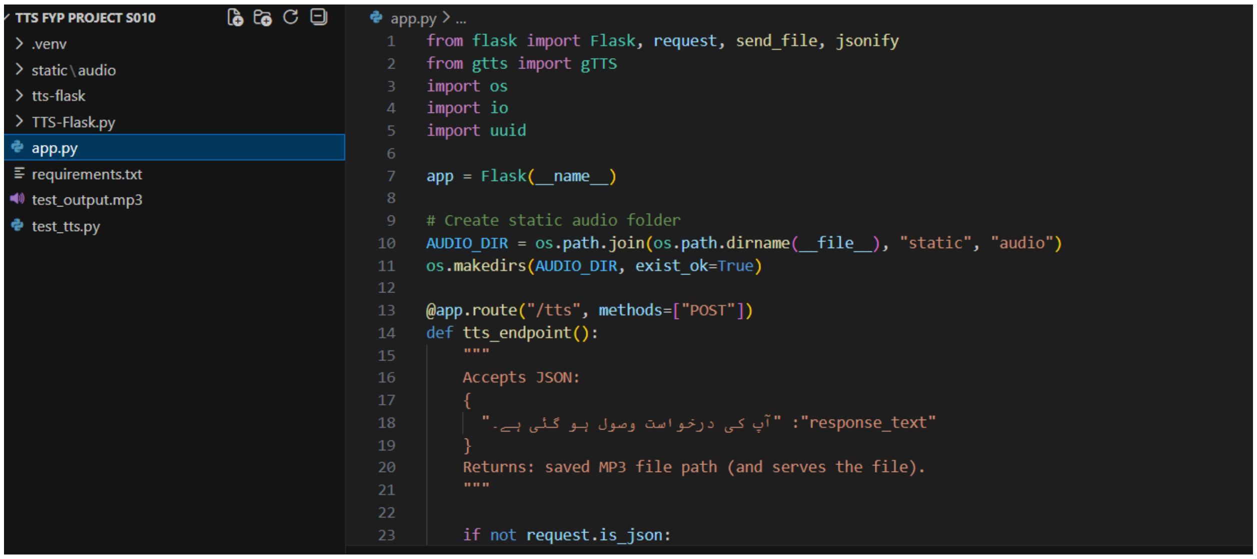

Project_TTS.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```python
def tts_service_from_json(json_body, filename="api_output.mp3"):
    text = json_body.get("response_text", "")
    if not text:
        raise ValueError("No response_text in JSON")
    tts = gTTS(text=text, lang='ur')
    tts.save(filename)
    return filename

# Test
file = tts_service_from_json({"response_text":"ہیلو یہ ایک ٹیسٹ ہے"}, "test_output.mp3")
display(Audio(file))
```

```
0:02 / 0:02
```

3. Created Flask API for TTS

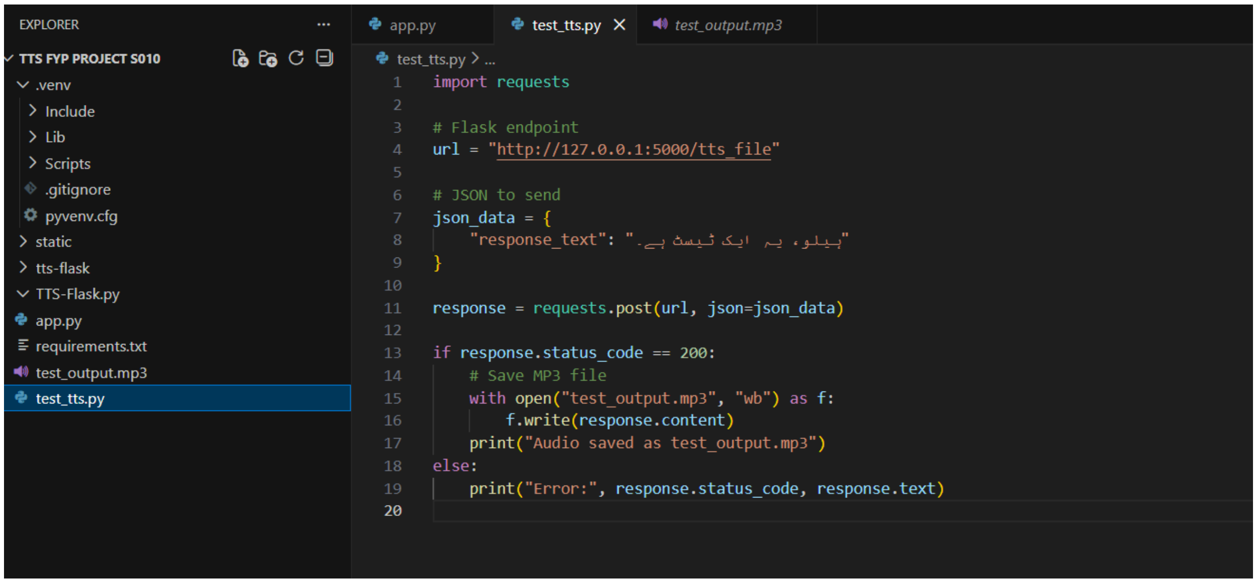
- Built a Flask backend `app.py` that:
    1. Accepts POST requests with JSON input → `{"response_text": "<text>"}`
    2. Generates speech audio using gTTS
    3. Returns the MP3 file as an API response.



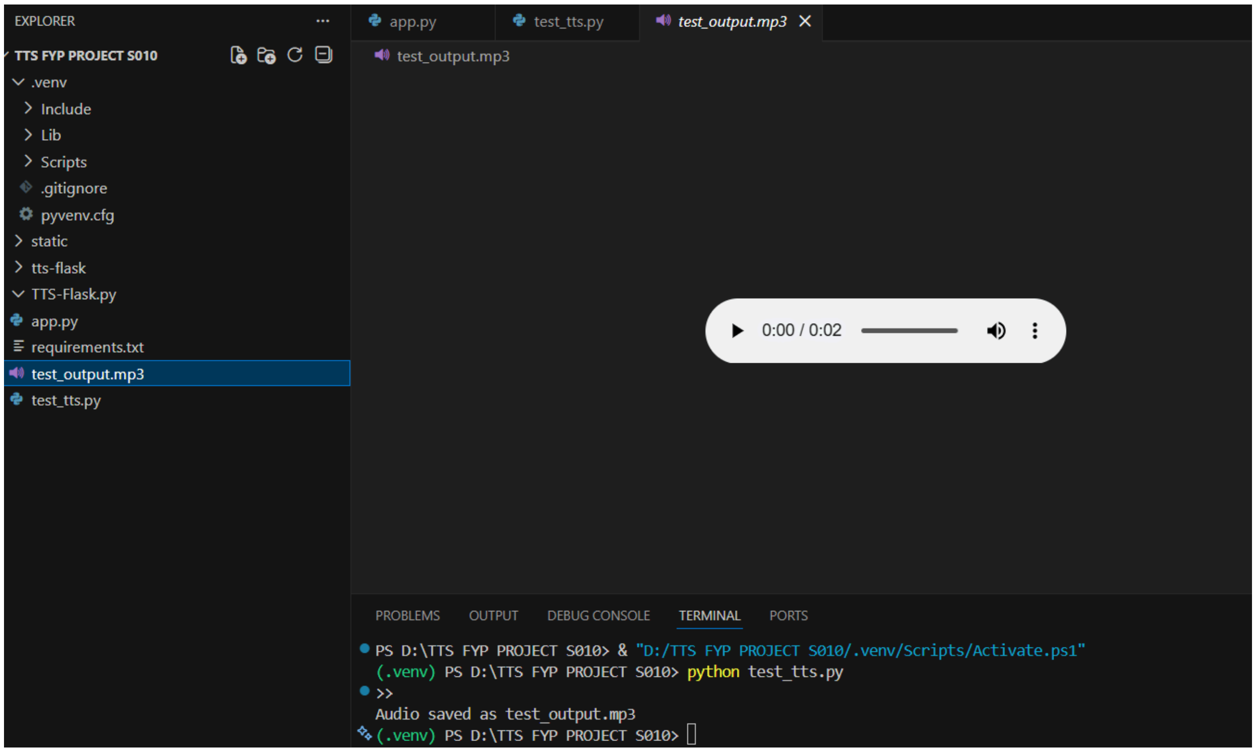
```python
from flask import Flask, request, send_file, jsonify
from gtts import gTTS
import os
import io
import uuid

app = Flask(__name__)

# Create static audio folder
AUDIO_DIR = os.path.join(os.path.dirname(__file__), "static", "audio")
os.makedirs(AUDIO_DIR, exist_ok=True)

@app.route("/tts", methods=["POST"])
def tts_endpoint():
    """
    Accepts JSON:
    {
      "response_text": "آپ کی درخواست وصول ہو گئی ہے۔"
    }
    Returns: saved MP3 file path (and serves the file).
    """

    if not request.is_json:
```

4. Testing TTS Locally

- Activated `.venv` and ran Flask using:
  `python app.py`
- Tested the API using:
  - **test_tts.py** helper script
- Successfully received MP3 files generated through the API.



```python
import requests

# Flask endpoint
url = "http://127.0.0.1:5000/tts_file"

# JSON to send
json_data = {
    "response_text": "ہیلو، یہ ایک ٹیسٹ ہے۔"
}

response = requests.post(url, json=json_data)

if response.status_code == 200:
    # Save MP3 file
    with open("test_output.mp3", "wb") as f:
        f.write(response.content)
    print("Audio saved as test_output.mp3")
else:
    print("Error:", response.status_code, response.text)
```

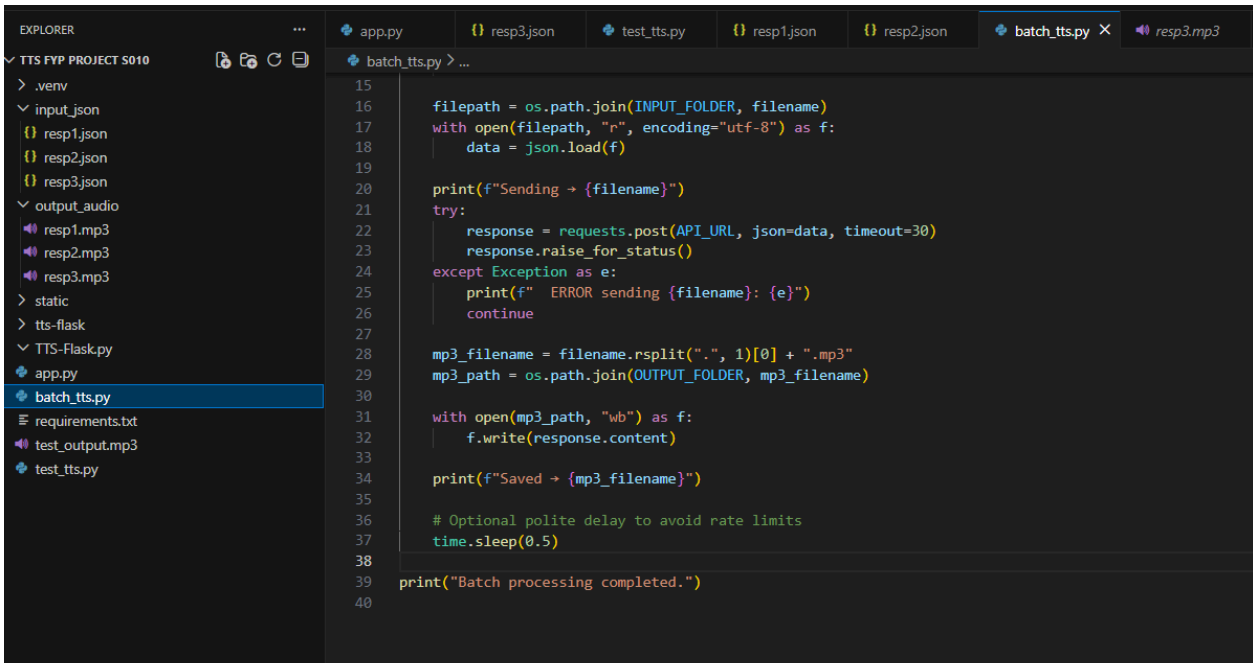


```
PS D:\TTS FYP PROJECT S010> & "D:/TTS FYP PROJECT S010/.venv/Scripts/Activate.ps1"
(.venv) PS D:\TTS FYP PROJECT S010> python test_tts.py
>>
Audio saved as test_output.mp3
(.venv) PS D:\TTS FYP PROJECT S010> 
```

5. Batch JSON Processing

- Implemented batch processing of multiple JSON files.

- Script sends each JSON to the Flask `/tts` endpoint and retrieves corresponding MP3 files.
- Verified end-to-end workflow: JSON → Flask API → MP3 output.



```python

    filepath = os.path.join(INPUT_FOLDER, filename)
    with open(filepath, "r", encoding="utf-8") as f:
        data = json.load(f)

    print(f"Sending → {filename}")
    try:
        response = requests.post(API_URL, json=data, timeout=30)
        response.raise_for_status()
    except Exception as e:
        print(f"  ERROR sending {filename}: {e}")
        continue

    mp3_filename = filename.rsplit(".", 1)[0] + ".mp3"
    mp3_path = os.path.join(OUTPUT_FOLDER, mp3_filename)

    with open(mp3_path, "wb") as f:
        f.write(response.content)

    print(f"Saved → {mp3_filename}")

    # Optional polite delay to avoid rate limits
    time.sleep(0.5)

print("Batch processing completed.")

```

Future course of actions:

1.Integration with Knowledge Graph (KG) & Natural Language Understanding (NLU):

- ○ Integrate TTS module with KG/NLU for dynamic text generation based on user queries.
- ○ This will enable context-aware responses and improve conversational capabilities.

2. Output Storage & Management:

- ○ Consider adding a database or metadata logging for easier retrieval and playback.

3. Automation & Deployment:

  - Run Flask server continuously or deploy using a production-ready WSGI server like Gunicorn.
  - Implement error handling for invalid JSON input and logging of TTS processing events.

4. **Additional Enhancements**:
  - Integrate with front-end or chatbot interface for seamless user interaction.