AI ASSISTED CODING

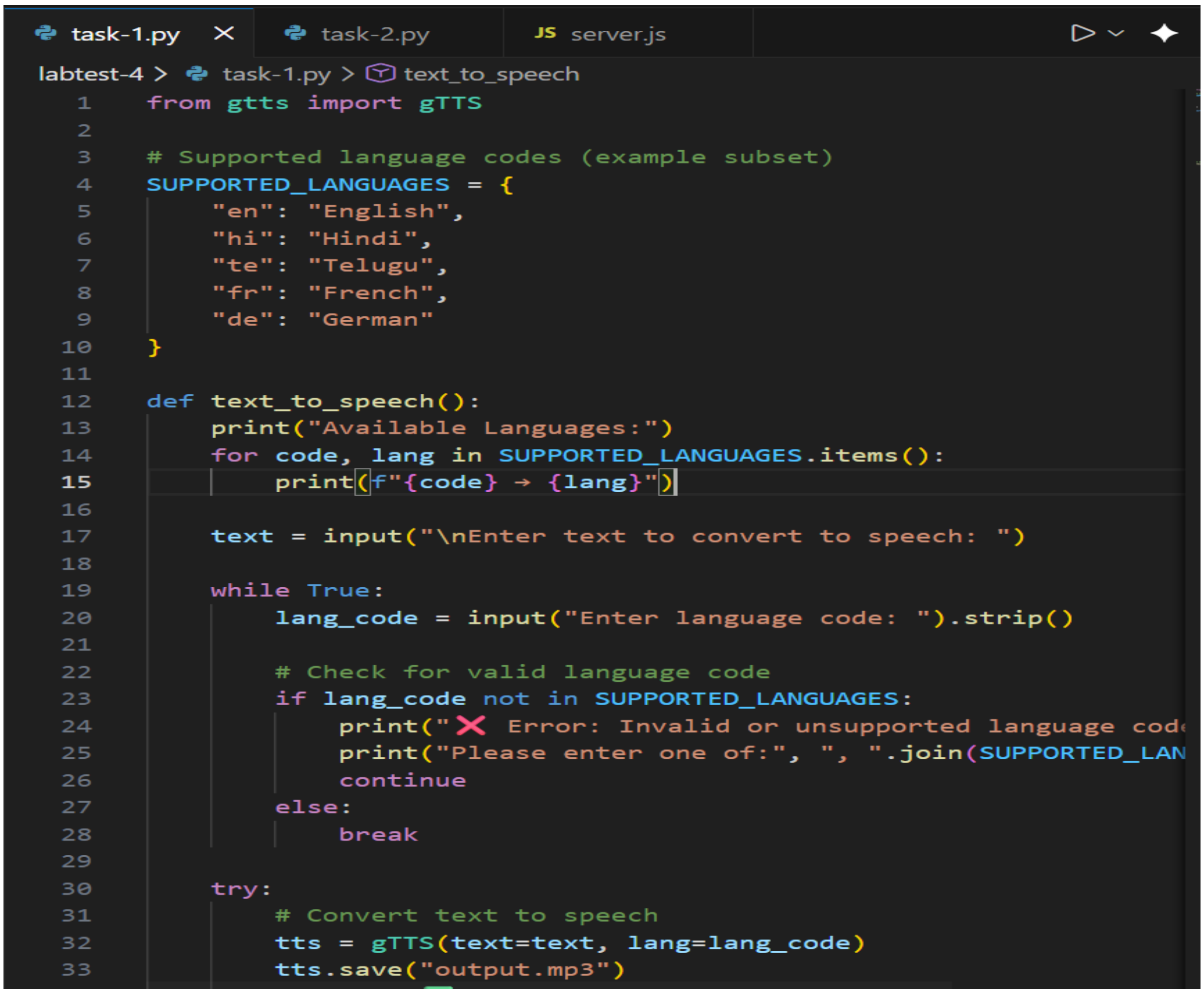
LAB TEST-04

ROLLNO:2403A52082

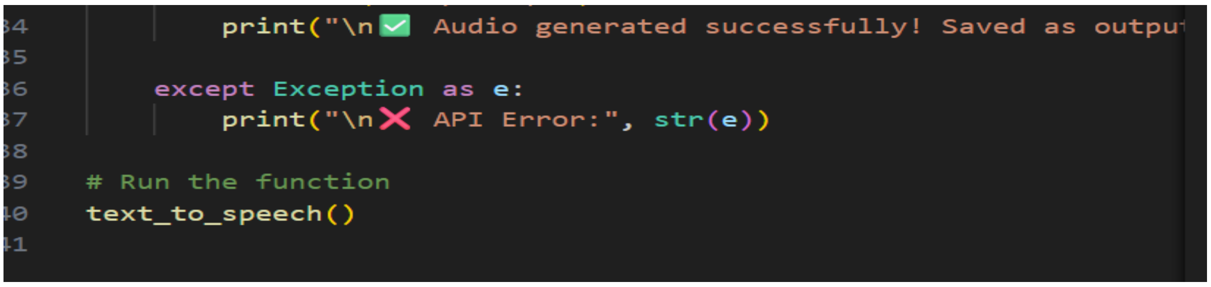
BATCH-04

Task-1: Write a program that connects to any Text-to-Speech API (such as Google Cloud TTS, Azure TTS, OpenAI TTS, or any free TTS API). Your program should detect and properly handle cases where the user enters an invalid or unsupported language code.

Code:

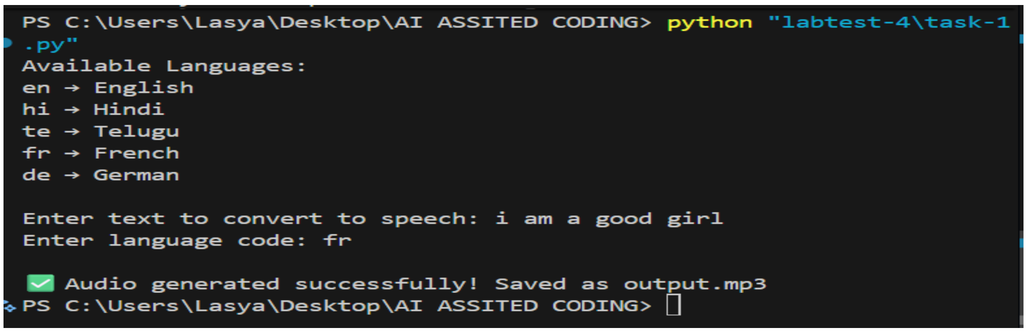


```python
from gtts import gTTS

# Supported language codes (example subset)
SUPPORTED_LANGUAGES = {
    "en": "English",
    "hi": "Hindi",
    "te": "Telugu",
    "fr": "French",
    "de": "German"
}

def text_to_speech():
    print("Available Languages:")
    for code, lang in SUPPORTED_LANGUAGES.items():
        print(f"{code} → {lang}")

    text = input("\nEnter text to convert to speech: ")

    while True:
        lang_code = input("Enter language code: ").strip()

        # Check for valid language code
        if lang_code not in SUPPORTED_LANGUAGES:
            print("❌ Error: Invalid or unsupported language cod
            print("Please enter one of:", ", ".join(SUPPORTED_LAN
            continue
        else:
            break

    try:
        # Convert text to speech
        tts = gTTS(text=text, lang=lang_code)
        tts.save("output.mp3")
```



```python
        print("\n✅ Audio generated successfully! Saved as outpu

    except Exception as e:
        print("\n❌ API Error:", str(e))

# Run the function
text_to_speech()
```

Output:



```
PS C:\Users\Lasya\Desktop\AI ASSITED CODING> python "labtest-4\task-1.py"
Available Languages:
en → English
hi → Hindi
te → Telugu
fr → French
de → German

Enter text to convert to speech: i am a good girl
Enter language code: fr

✅ Audio generated successfully! Saved as output.mp3
PS C:\Users\Lasya\Desktop\AI ASSITED CODING> 
```

Explanation: **Import gTTS**

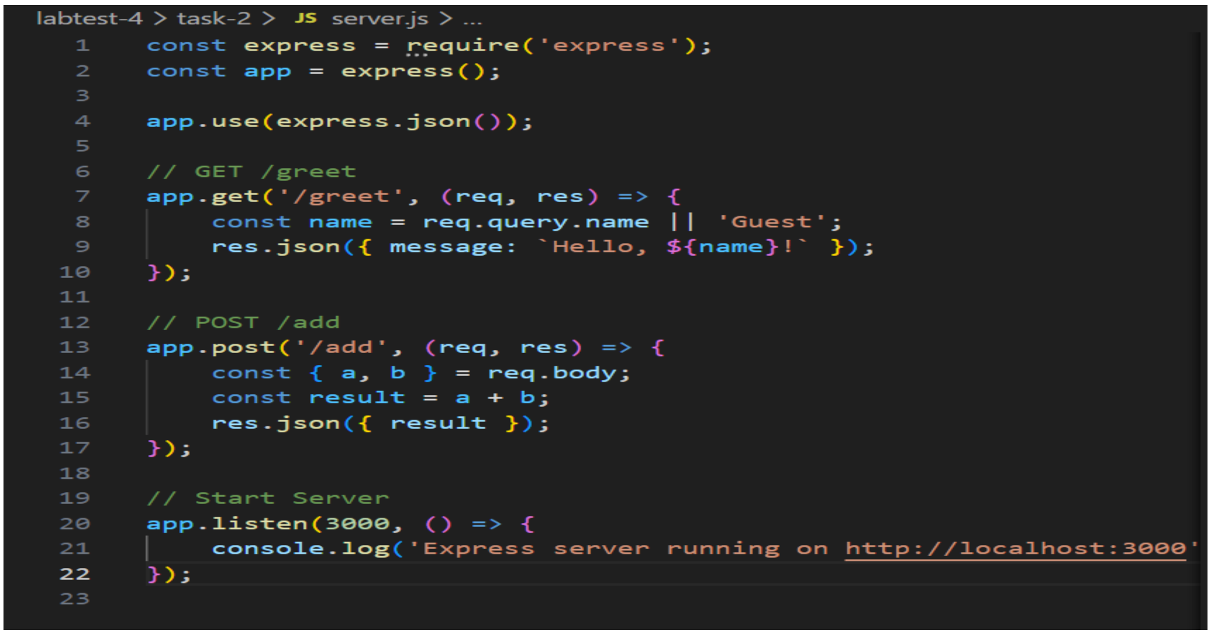
- from gtts import gTTS — imports the Google Text-to-Speech library.
- Requires internet connection to fetch audio from Google's API.

2. **Supported Languages Dictionary**
   - SUPPORTED_LANGUAGES — maps language codes (e.g., "en", "hi") to human-readable names.
   - Example: "en": "English", "hi": "Hindi", "te": "Telugu".
   - Limits available options for user selection.
3. **Display Available Languages**
   - Prints all supported language codes and names.
   - Format: en → English, hi → Hindi, etc.
   - Helps user choose a valid code.
4. **Get Text Input**
   - text = input("\nEnter text to convert to speech: ") — prompts user for text.
   - Stores the input string.
5. **Validate Language Code (While Loop)**
   - while True: — loops until a valid

Task-2: Convert the given Python Flask API code into an equivalent **Node.js Express** application. Maintain the same routes, HTTP methods, request handling, status codes, and JSON response structure.

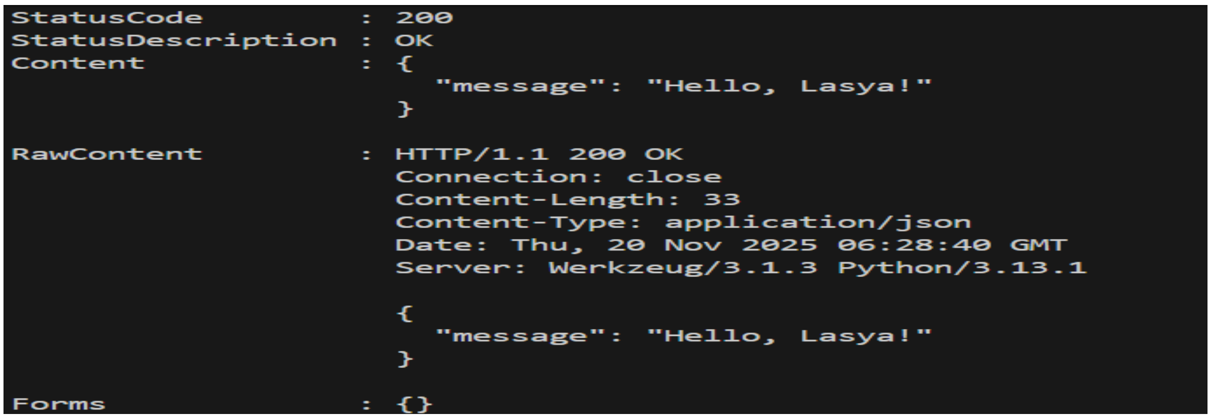
b) Provide a method to **test endpoint equivalence** between the Flask version and the Express version.

Code:

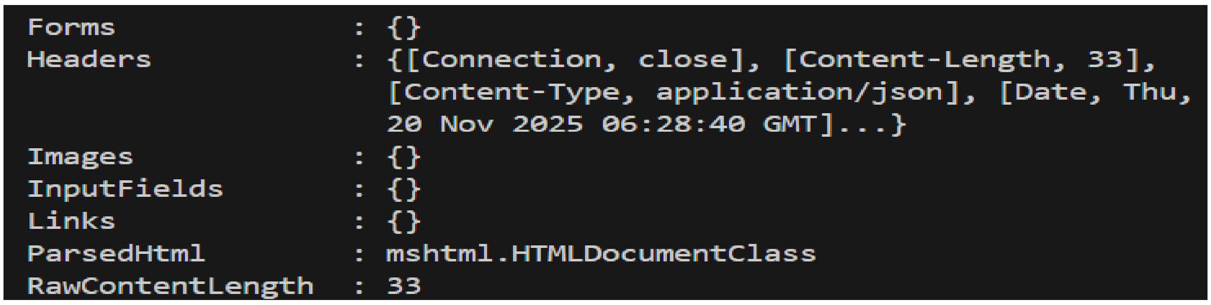


```
labtest-4 > task-2 > JS server.js > ...
const express = require('express');
const app = express();

app.use(express.json());

// GET /greet
app.get('/greet', (req, res) => {
    const name = req.query.name || 'Guest';
    res.json({ message: `Hello, ${name}!` });
});

// POST /add
app.post('/add', (req, res) => {
    const { a, b } = req.body;
    const result = a + b;
    res.json({ result });
});

// Start Server
app.listen(3000, () => {
    console.log('Express server running on http://localhost:3000'
});
```

Output:



```
StatusCode        : 200
StatusDescription : OK
Content           : {
                      "message": "Hello, Lasya!"
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 33
                    Content-Type: application/json
                    Date: Thu, 20 Nov 2025 06:28:40 GMT
                    Server: Werkzeug/3.1.3 Python/3.13.1

                    {
                      "message": "Hello, Lasya!"
                    }
Forms             : {}
```



```
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 33],
                    [Content-Type, application/json], [Date, Thu,
                    20 Nov 2025 06:28:40 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 33
```

- Explanation: File: an Express.js HTTP server (server.js).
- Middleware: app.use(express.json()) parses incoming JSON request bodies.
- Endpoints:
    - GET /greet — optional query parameter name; responds with JSON { "message": "Hello, <name>!" } (defaults to "Guest").

    - POST /add — expects JSON body with numeric fields a and b; responds with JSON { "result": a + b }.

- Server: listens on port 3000 and logs the URL.

Notes / quick improvements (short):

- Add input validation (check a and b are numbers) and error handling.
- Consider CORS or rate-limiting if used from browsers/production.
- Add logging and process management (pm2) for production use.