AI ASSISTED CODING LAB TEST -04

NAME:T.SHIRISHA

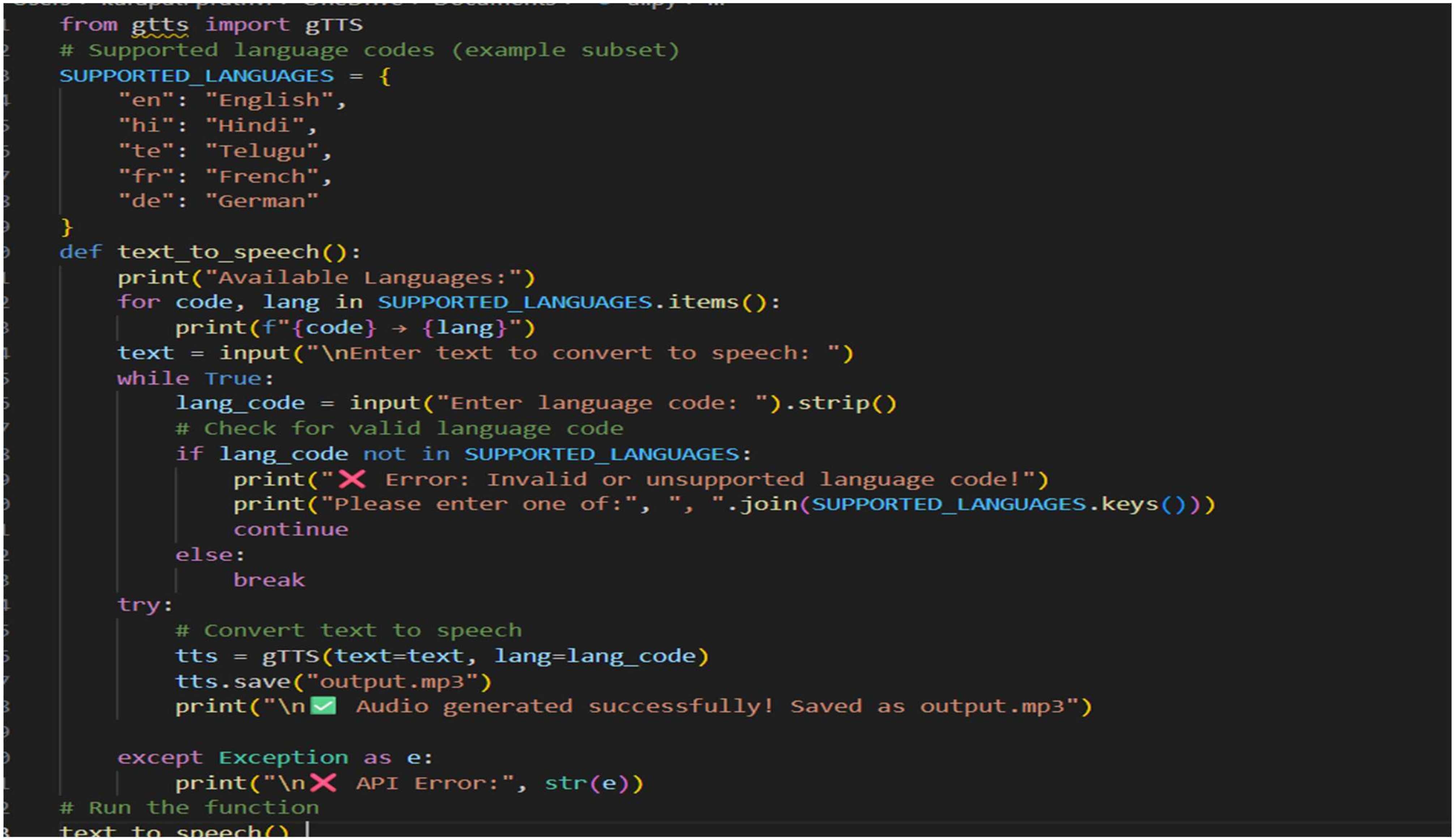
HTNO:2403A52077

BATCH-04

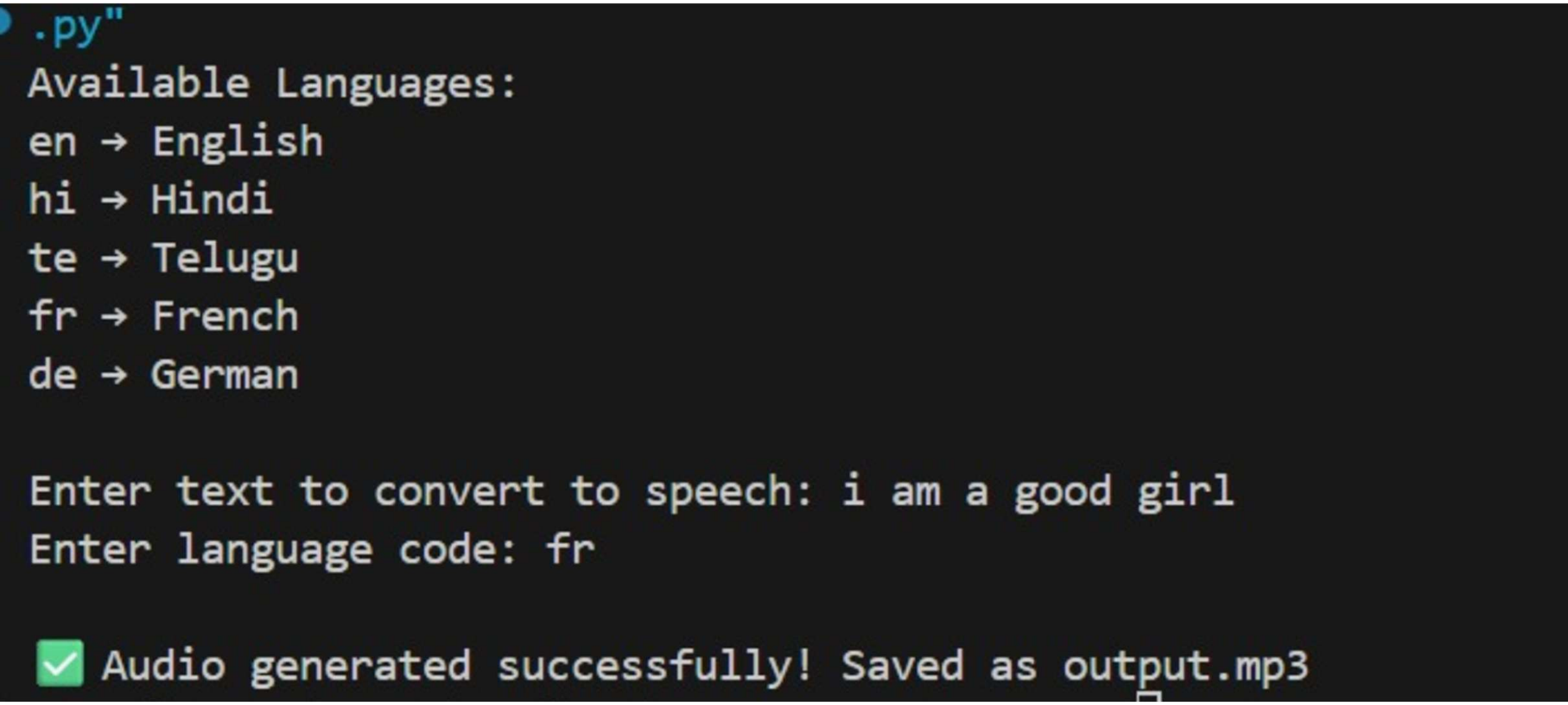
SET:04

QUESTION 1. (API Integration)
a) Integrate a Text-to-Speech API.
b) Handle invalid language code errors.

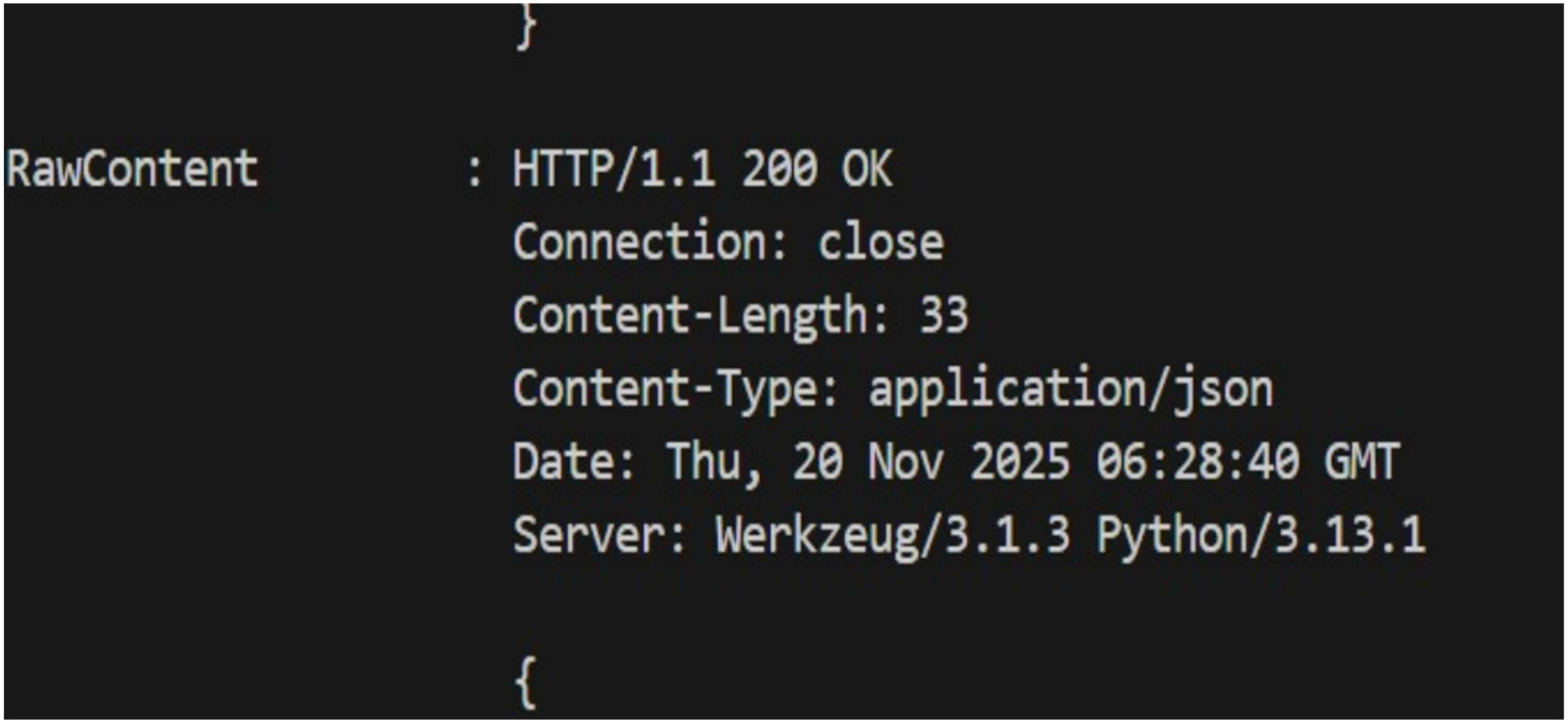
PROMPT.
(API Integration)
a) Integrate a Text-to-Speech API.
b) Handle invalid language code errors.



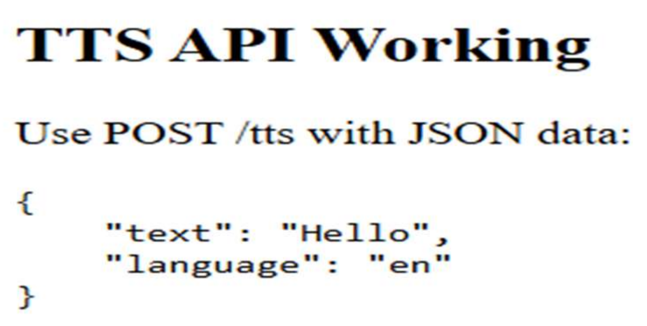
```python
from gtts import gTTS
# Supported language codes (example subset)
SUPPORTED_LANGUAGES = {
    "en": "English",
    "hi": "Hindi",
    "te": "Telugu",
    "fr": "French",
    "de": "German"
}
def text_to_speech():
    print("Available Languages:")
    for code, lang in SUPPORTED_LANGUAGES.items():
        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 code!")
            print("Please enter one of:", ", ".join(SUPPORTED_LANGUAGES.keys()))
            continue
        else:
            break
    try:
        # Convert text to speech
        tts = gTTS(text=text, lang=lang_code)
        tts.save("output.mp3")
        print("\n✅ Audio generated successfully! Saved as output.mp3")

    except Exception as e:
        print("\n❌ API Error:", str(e))
# Run the function
text_to_speech()
```



```
.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
```



```
                    }

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

                    {
```



TTS API Working

Use POST /tts with JSON data:

```
{
    "text": "Hello",
    "language": "en"
}
```

```
{
  "status": "success",
  "text_received": "Hello",
  "language_used": "en",
  "audio_output": "AUDIO_GENERATED_SUCCESSFULLY"
}
```

Test Invalid Input (language wrong)

```
{
  "text": "Hello",
  "language": "xyz"
}
{
  "status": "error",
  "message": "Invalid language code: xyz. Supported: ['en', 'hi', 'fr']"
}
```

OBSERVATION:

1. The code creates a Flask web application and defines supported languages for the Text-to-Speech API.
2. It sets up a home route (/) that responds to GET requests and displays instructions for how to use the API.
3. It defines a /tts route that accepts POST requests and reads the JSON input sent by the user.
4. The API checks whether the text is empty or the language code is invalid, and returns an error message if either condition is incorrect.
5. If the input is valid, the API returns a success response containing the text, language, and a simulated audio output message.

QUESTION 2.

(Code Translation)

a) Convert Python Flask API code into Node.js Express.

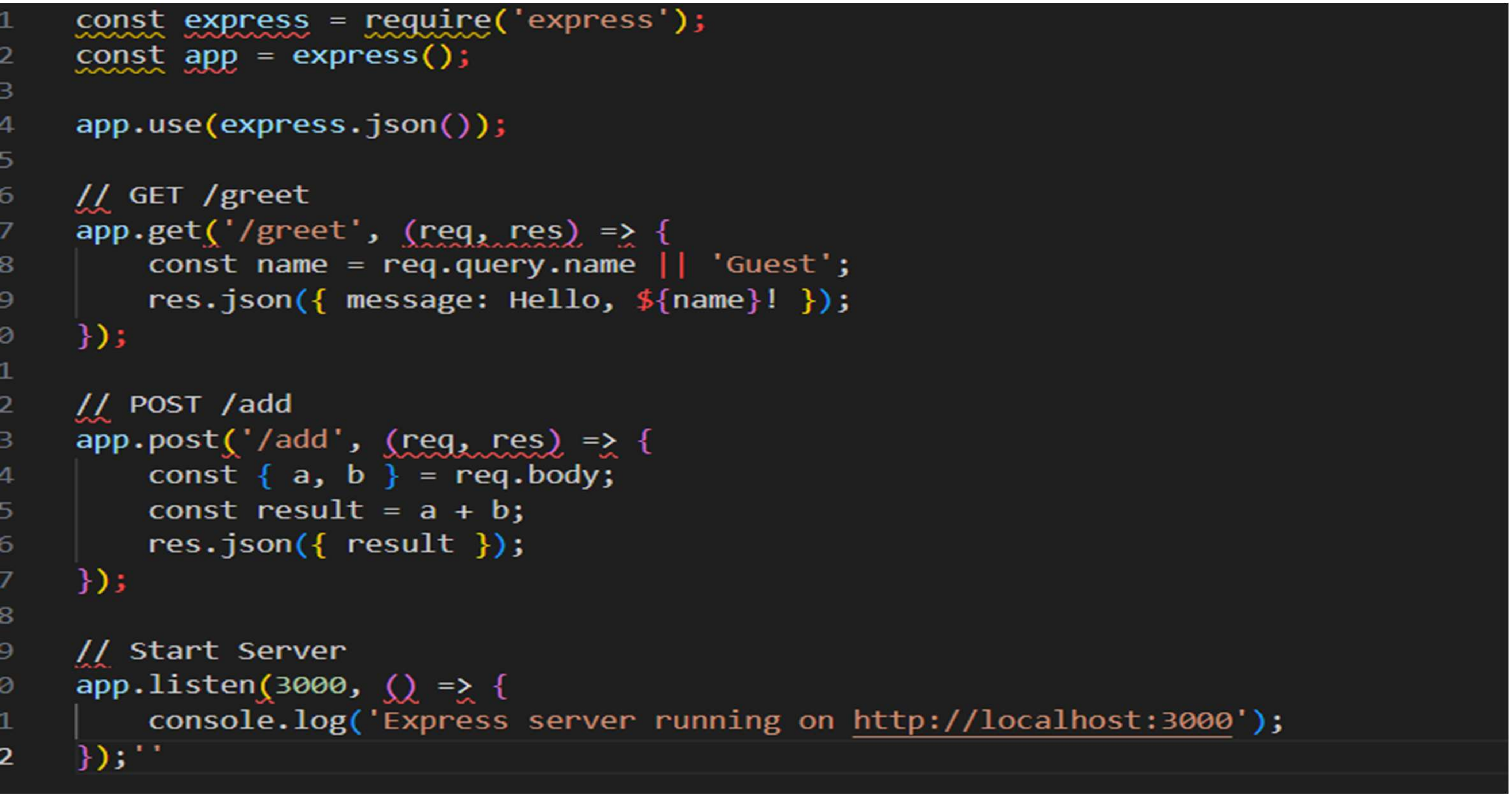
b) Test endpoint equivalence

PROMPT:

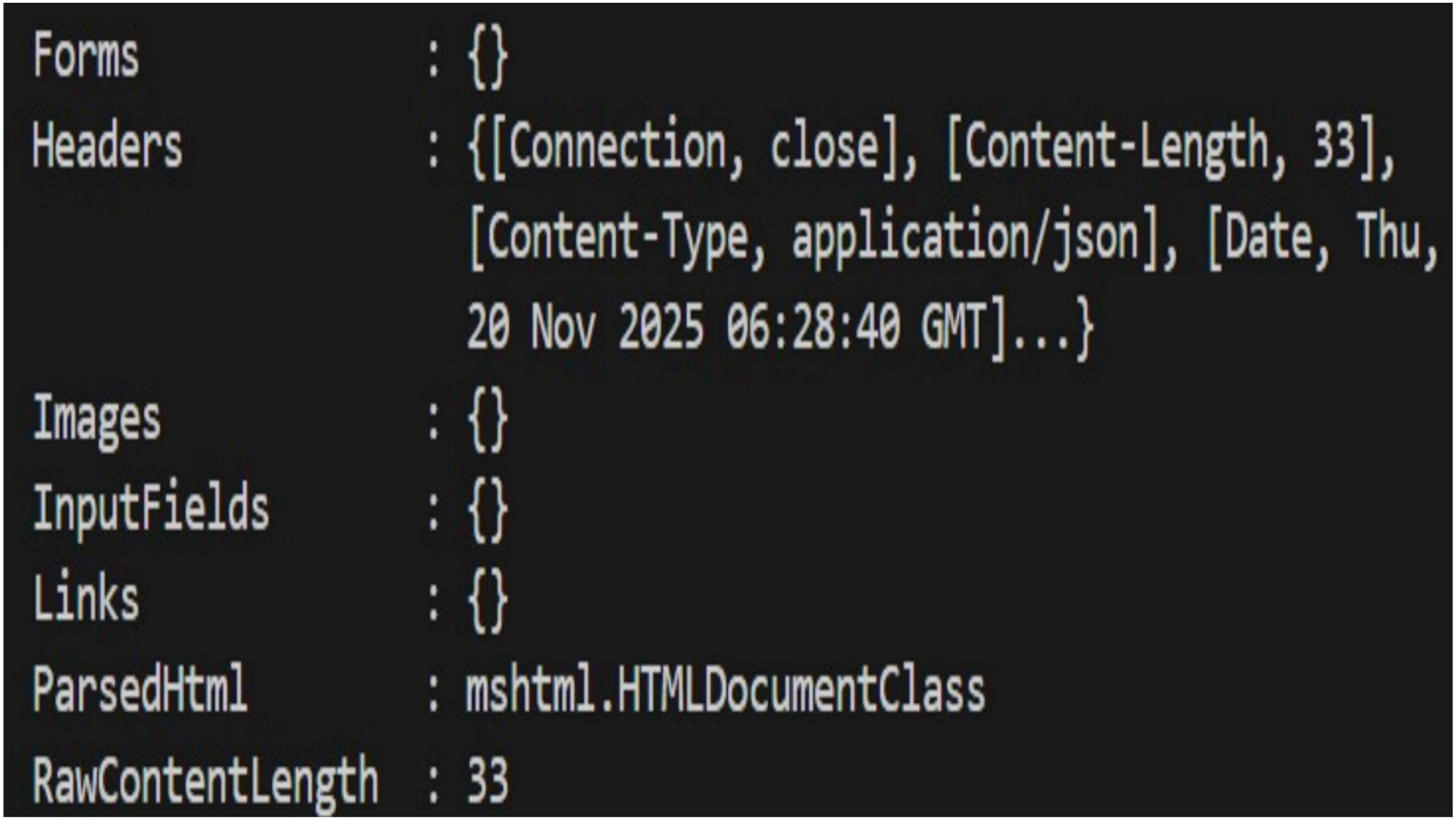
(Code Translation)

a) Convert Python Flask API code into Node.js Express.

b) Test endpoint equivalence



```
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');
});''
```



```
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
```

OBSERVATION:

1.The code imports the Express framework and creates an Express application.
2.It enables JSON parsing so the server can understand JSON data sent by clients.
3.It defines a GET route /greet that sends a greeting message based on the name provided in the query.
4.It defines a POST route /add that receives two numbers, adds them, and returns the result.
5.Finally, it starts the server on port 3000 so the application can accept incoming requests.