

AI ASSISTED CODING

Program :B.tech(CSE)
Name of Student :ANANTHA MANIDEEP
Enrollment No. :2403A52078
Batch No. :02

LAB ASSIGNMENT-15.3

TASK DESCRIPTION-1:

Basic REST API Setup

Task: Ask AI to generate a Flask REST API with one route:

GET /hello → returns {"message": "Hello, AI Coding!"}

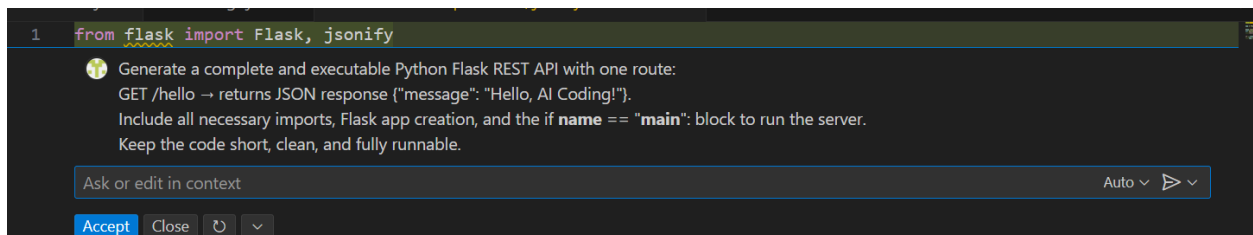
PROMPT:

Generate a complete and executable Python Flask REST API with one route:

GET /hello → returns JSON response {"message": "Hello, AI Coding!"}.

Include all necessary imports, Flask app creation, and the if `__name__ == "__main__"`: block to run the server.

Keep the code short, clean, and fully runnable.



The screenshot shows an AI coding assistant interface. At the top, a code snippet is displayed: `1 from flask import Flask, jsonify`. Below this, a prompt is shown: "Generate a complete and executable Python Flask REST API with one route: GET /hello → returns JSON response (\"message\": \"Hello, AI Coding!\"). Include all necessary imports, Flask app creation, and the if `name == \"main\"`: block to run the server. Keep the code short, clean, and fully runnable." The prompt is enclosed in a box with a blue border. Below the prompt, there is a text input field with the placeholder "Ask or edit in context" and a button labeled "Auto" with a dropdown arrow. At the bottom of the interface, there are buttons for "Accept", "Close", and a refresh icon.

CODE:

```
C: > Users > deeks > success2.py > ...
1  from flask import Flask, jsonify
2
3  app = Flask(__name__)
4
5  @app.route('/hello', methods=['GET'])
6  def hello():
7      return jsonify({"message": "Hello, AI Coding!"})
8
9  if __name__ == '__main__':
10     app.run(debug=True)
```

OUTPUT:

```
10.3.60.4 - - [05/Nov/2025 13:17:53] "GET / HTTP/1.1" 404 -
10.3.60.4 - - [05/Nov/2025 13:17:55] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Nov/2025 13:18:47] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [05/Nov/2025 13:18:47] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Nov/2025 13:18:50] "GET / HTTP/1.1" 404 -
```

TASK DESCRIPTION-2:

CRUD Operations (Students API)

Task:

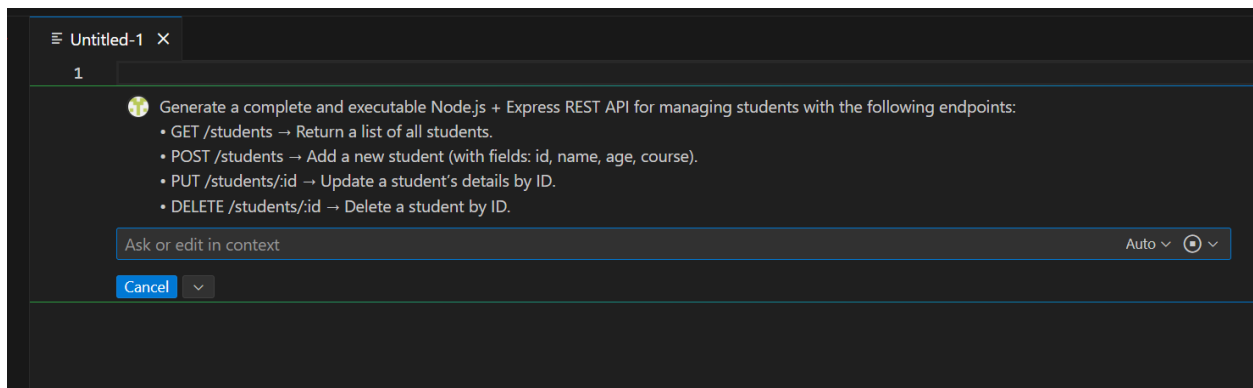
Use AI to build REST endpoints for a Student API:

- GET /students → List all students.
- POST /students → Add a new student.
- PUT /students/<id> → Update student details.
- DELETE /students/<id> → Delete a student.

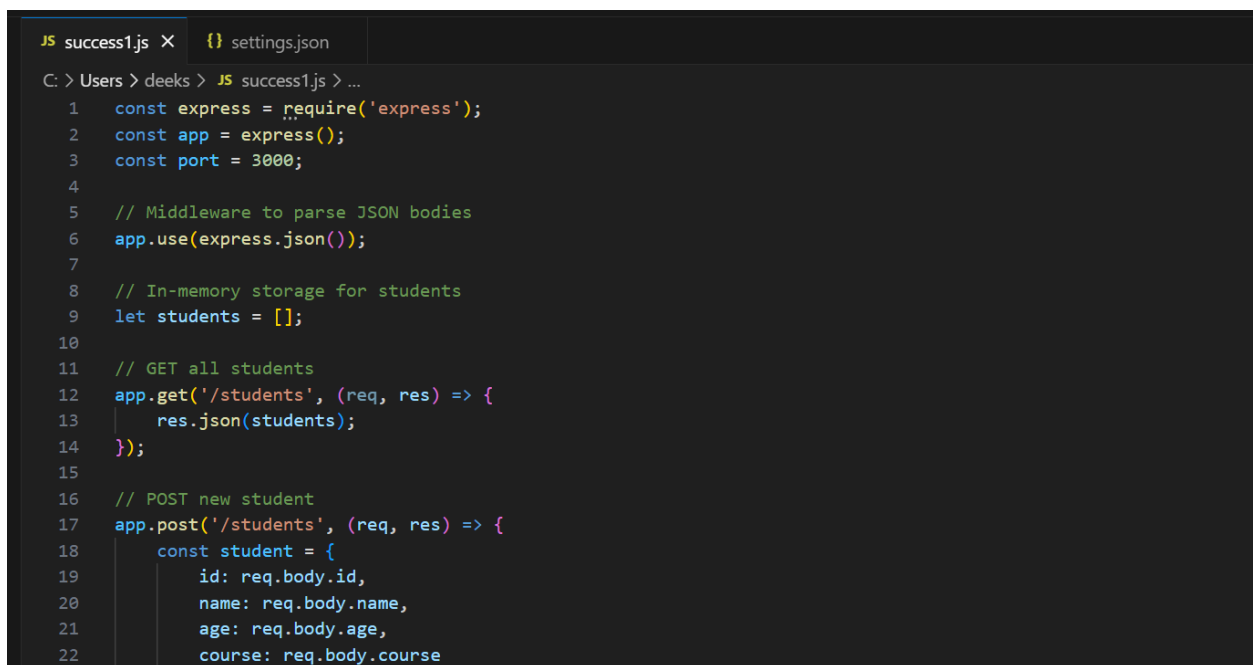
PROMPT:

Generate a complete and executable Node.js + Express REST API for managing students with the following endpoints:

- GET /students → Return a list of all students.
- POST /students → Add a new student (with fields: id, name, age, course).
- PUT /students/:id → Update a student's details by ID.
- DELETE /students/:id → Delete a student by ID.



CODE:



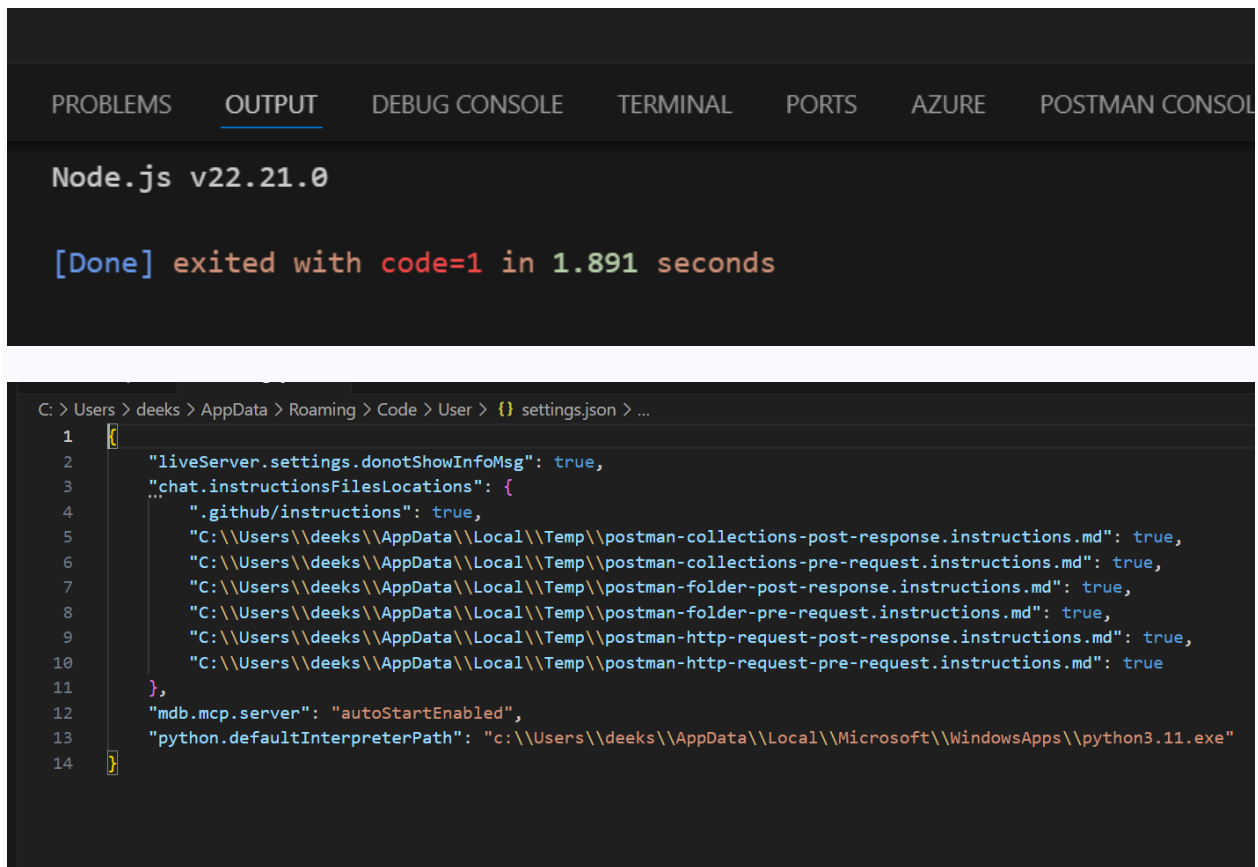
```
JS success1.js X {} settings.json
C: > Users > deeks > JS success1.js > ...
17 app.post('/students', (req, res) => {
18   const student = {
23   };
24   students.push(student);
25   res.status(201).json(student);
26 });
27
28 // PUT update student
Complexity is 4 Everything is cool!
29 app.put('/students/:id', (req, res) => {
30   const id = req.params.id;
31   const index = students.findIndex(s => s.id === id);
32
33   if (index === -1) {
34     return res.status(404).json({ message: "Student not found" });
35   }
36
37   students[index] = {
38     id: id,
39     name: req.body.name,
40     age: req.body.age,
41     course: req.body.course
```

```
JS success1.js X {} settings.json
C: > Users > deeks > JS success1.js > ...
29 app.put('/students/:id', (req, res) => {
41   course: req.body.course
42   };
43
44   res.json(students[index]);
45 });
46
47 // DELETE student
Complexity is 5 Everything is cool!
48 app.delete('/students/:id', (req, res) => {
49   const id = req.params.id;
50   const index = students.findIndex(s => s.id === id);
51
52   if (index === -1) {
53     return res.status(404).json({ message: "Student not found" });
54   }
55
56   students = students.filter(s => s.id !== id);
57   res.status(204).send();
58 });
59
```

```
});

// Start server
v app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

OUTPUT:



The screenshot shows the Visual Studio Code interface. The top bar has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, AZURE, and POSTMAN CONSOLE. The OUTPUT window is active, displaying the message: `Node.js v22.21.0` and `[Done] exited with code=1 in 1.891 seconds`. Below the output window, a code editor shows the contents of `settings.json` in a dark theme. The file path is `C:\Users\deeks\AppData\Roaming\Code\User\settings.json`. The code is a JSON object with the following structure:

```
1 {
2   "liveServer.settings.donotShowInfoMsg": true,
3   "chat.instructionsFilesLocations": {
4     ".github/instructions": true,
5     "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-collections-post-response.instructions.md": true,
6     "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-collections-pre-request.instructions.md": true,
7     "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-folder-post-response.instructions.md": true,
8     "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-folder-pre-request.instructions.md": true,
9     "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-http-request-post-response.instructions.md": true,
10    "C:\\Users\\deeks\\AppData\\Local\\Temp\\postman-http-request-pre-request.instructions.md": true
11  },
12   "mdb.mcp.server": "autoStartEnabled",
13   "python.defaultInterpreterPath": "c:\\Users\\deeks\\AppData\\Local\\Microsoft\\WindowsApps\\python3.11.exe"
14 }
```

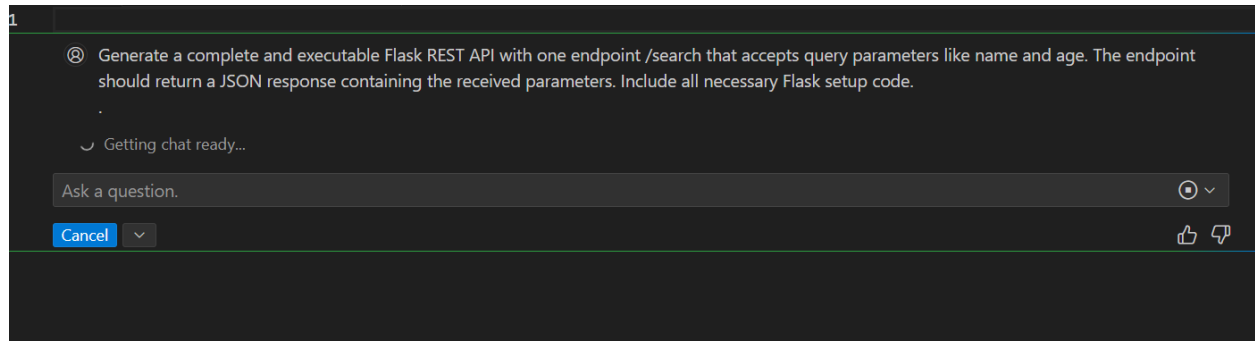
TASK DESCRIPTION-3:

API with Query Parameters

Task: Ask AI to generate a REST API endpoint

PROMPT:

Generate a complete and executable Flask REST API with one endpoint `/search` that accepts query parameters like name and age. The endpoint should return a JSON response containing the received parameters. Include all necessary Flask setup code.



CODE:

```
from flask import Flask, request, jsonify

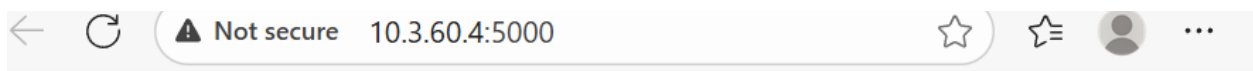
app = Flask(__name__)

@app.route('/search', methods=['GET'])
def search():
    name = request.args.get('name')
    age = request.args.get('age')
    if age is not None:
        try:
            age = int(age)
        except ValueError:
            # keep as string if it cannot be converted
            pass
    return jsonify({
        "name": name,
        "age": age,
        "query": dict(request.args) # echo all received query parameters
    })

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

OUTPUT:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE POSTMAN CONSOLE Filter Code
[Running] python -u "c:\Users\deeks\success3.py"
* Serving Flask app 'success3'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.3.60.4:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 928-731-464
```



Not Found

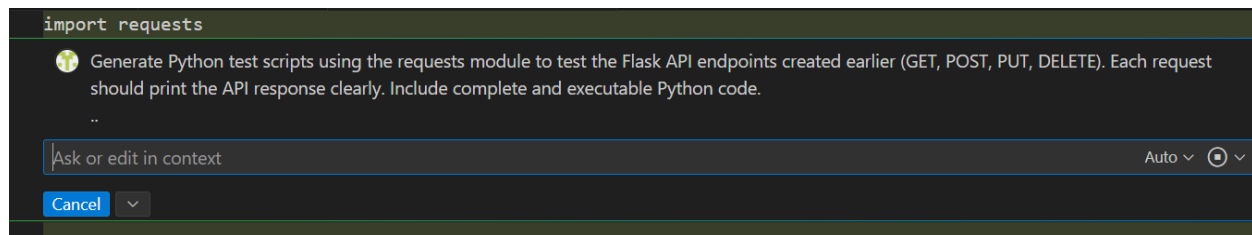
TASK DESCRIPTION-4:

Integration & Testing

Task: Ask AI to write test scripts using Python requests module to call APIs created above.

PROMPT:

Generate Python test scripts using the requests module to test the Flask API endpoints created earlier (GET, POST, PUT, DELETE). Each request should print the API response clearly. Include complete and executable Python code.



CODE:

```
import requests

BASE_URL = 'http://localhost:5000/api' # Change this to your actual API URL

def test_get():
    response = requests.get(f'{BASE_URL}/endpoint')
    print('GET Response:', response.json())

def test_post():
    data = {'key': 'value'} # Replace with your actual data
    response = requests.post(f'{BASE_URL}/endpoint', json=data)
    print('POST Response:', response.json())

def test_put():
    data = {'key': 'new_value'} # Replace with your actual data
    response = requests.put(f'{BASE_URL}/endpoint/1', json=data) # Change '1' to the ID you want to update
    print('PUT Response:', response.json())

def test_delete():
    response = requests.delete(f'{BASE_URL}/endpoint/1') # Change '1' to the ID you want to delete

def test_delete():
    response = requests.delete(f'{BASE_URL}/endpoint/1') # Change '1' to the ID you want to delete
    print('DELETE Response:', response.status_code)

if __name__ == '__main__':
    test_get()
    test_post()
    test_put()
    test_delete()
```

OUTPUT:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE POSTMAN CONSOLE

PS C:\Users\deeks> & 'C:\Users\deeks\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\python.exe' 'c:\Users\deeks\.vscode\extensions\ms-python.debugpy-2025.15.2025110302-win32-x64\bundled\libs\debugpy\launcher' '52314' '--' ''
Python 3.12.10 (tags/v3.12.10:0cc8128, Apr 8 2025, 12:21:36) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
Ctrl click to launch VS Code Native REPL
>>>
>>> □
```


--- Thank You---