AI ASSISTED CODING LAB

ASSIGNMENT-15.2

ENROLLMENT NO:2503A51L14

BATCH NO: 19

NAME: ROHITH GOPAGANI

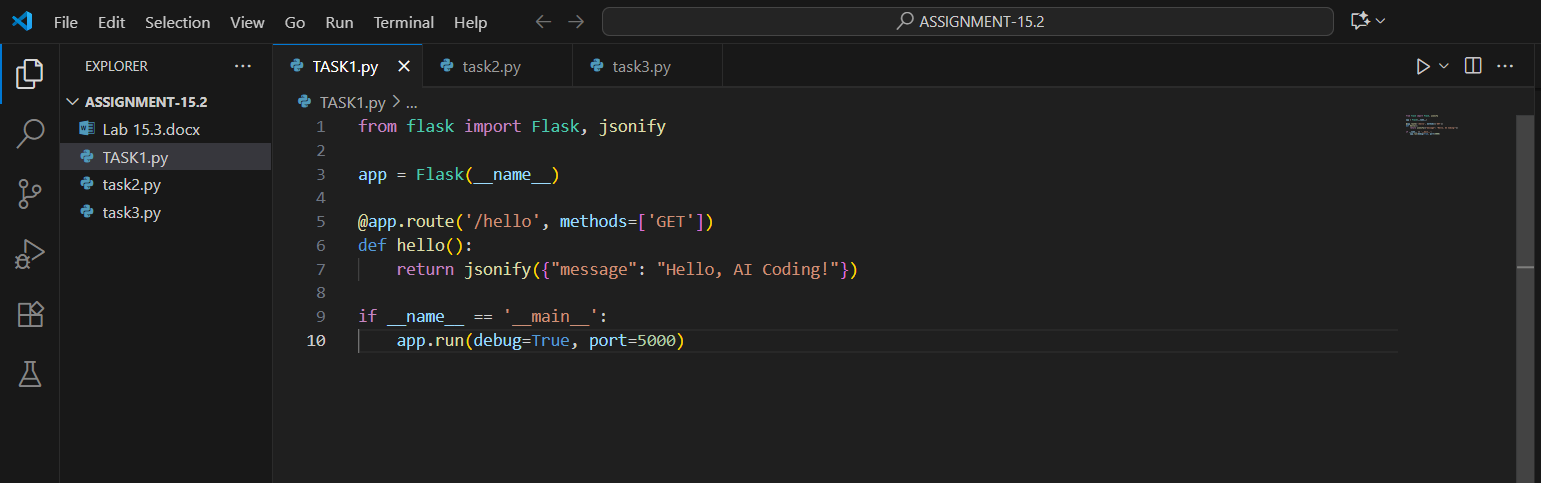
TASK DESCRIPTION 1 :Basic REST API Setup

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

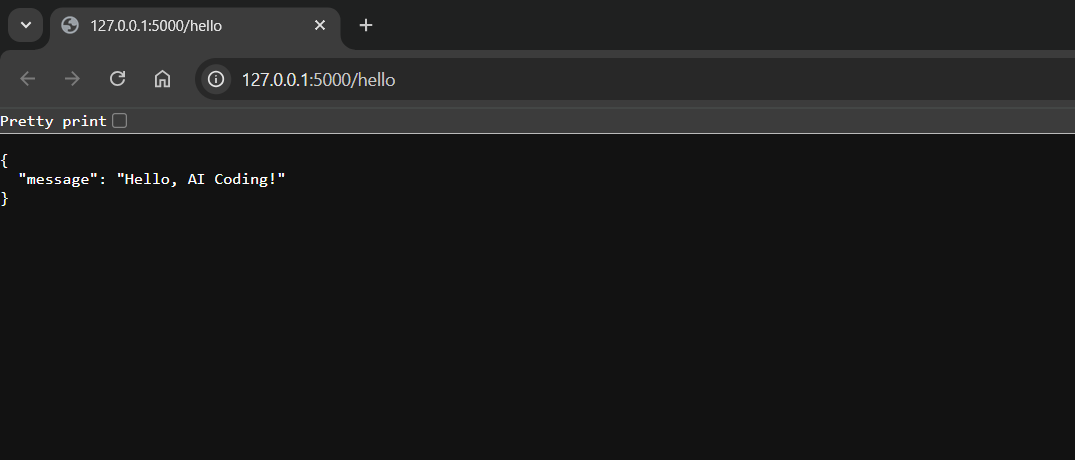
GET /hello → returns {"message": "Hello, AI Coding!"}

PROMPT :

Ask the AI to create a simple Flask REST API with one route, GET /hello, that returns a JSON message {"message": "Hello, AI Coding!"}, including all necessary setup and imports.

CODE GENERATED :

OUTPUT :



OBSERVATION :

The AI successfully generated a basic Flask REST API with the required setup and imports. The code included a single route, GET /hello, which correctly returned the JSON response {"message": "Hello, AI Coding!"} when accessed. This demonstrates that the AI can accurately understand and implement simple REST API instructions using Flask.

TASK DESCRIPTION 2 :

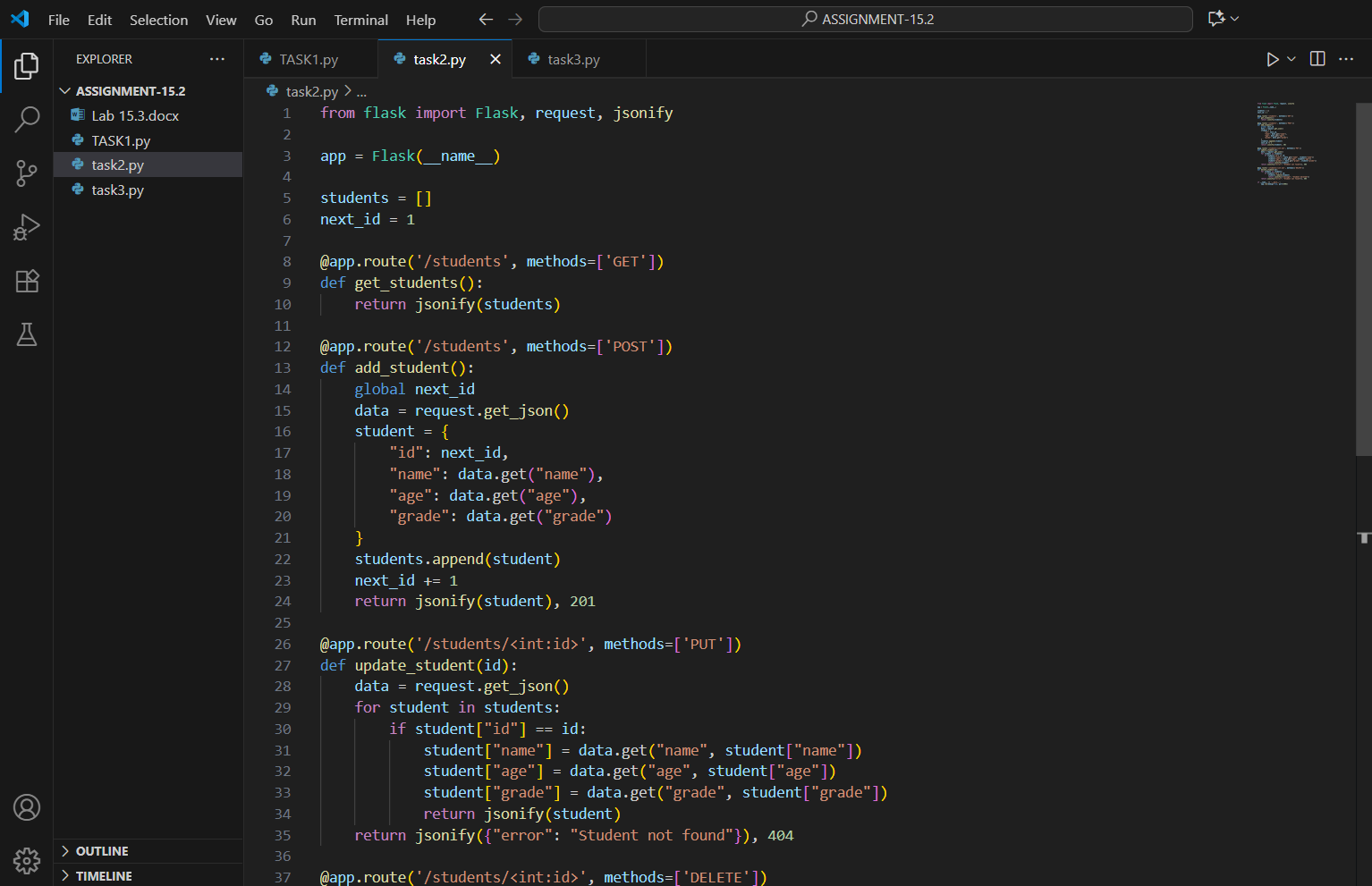
Use AI to build REST endpoints for a **Student API**:

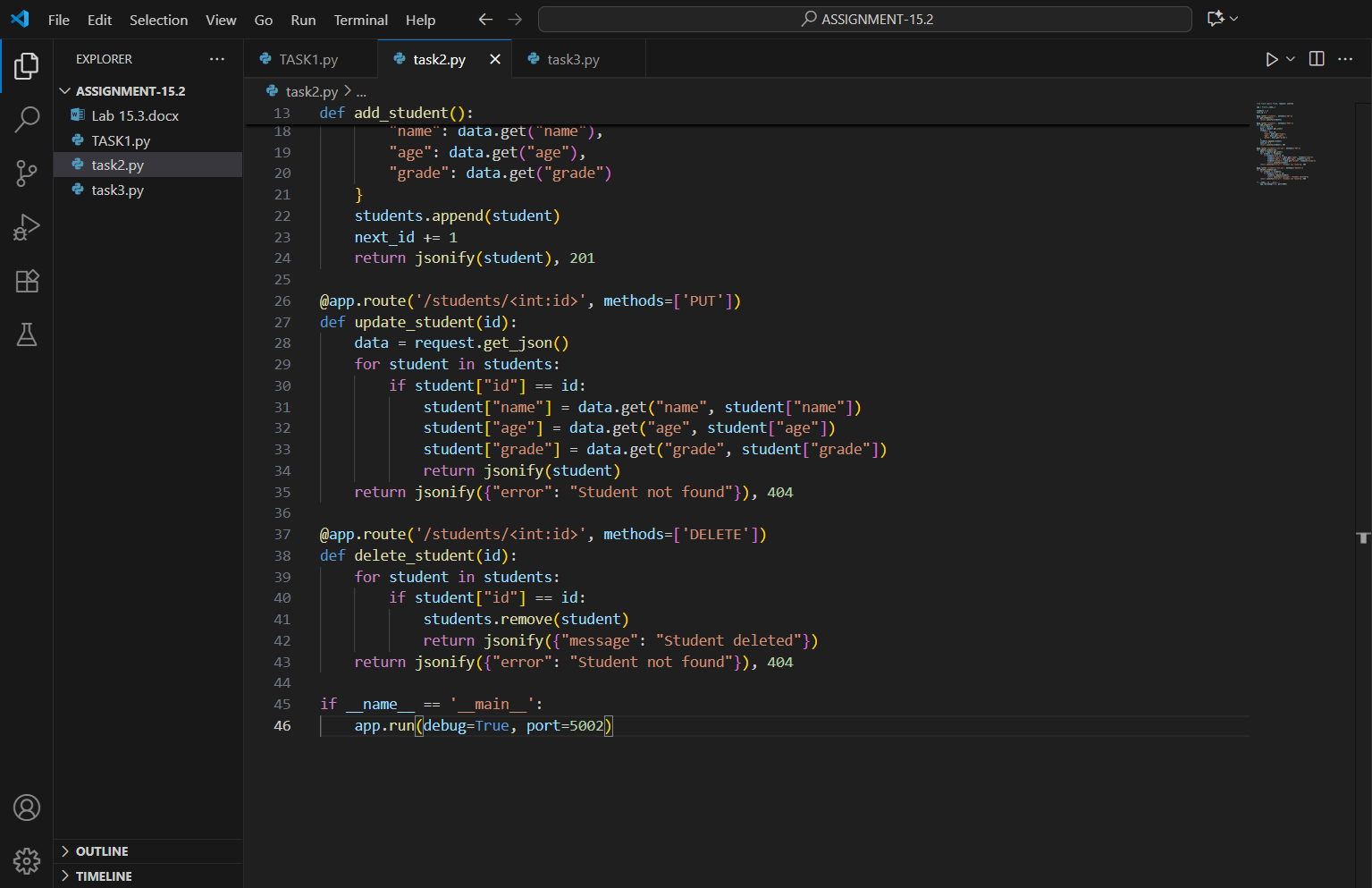
* GET /students → List all students.
* POST /students → Add a new student.
* PUT /students/<id> → Update student details.
* DELETE /students/<id> → Delete a student.

PROMPT :

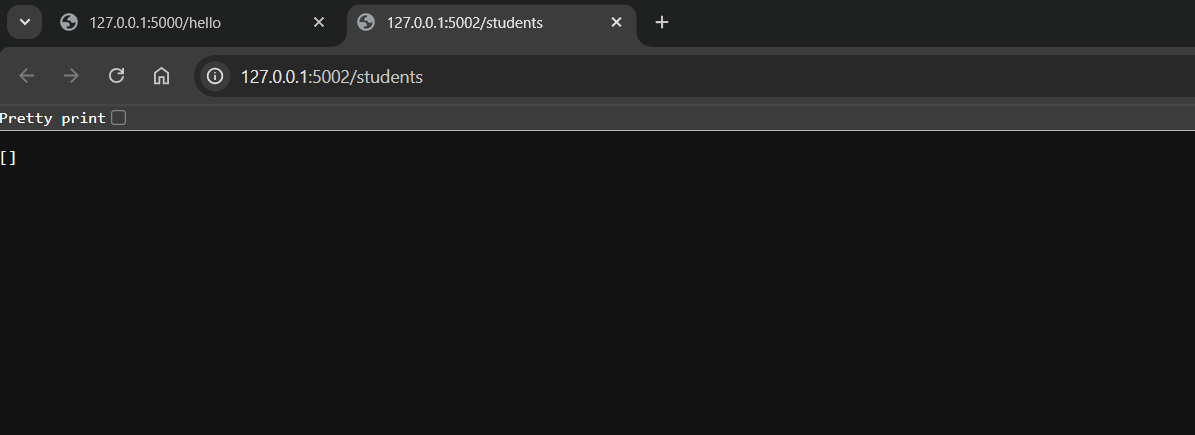
Ask the AI to build a Flask REST API for managing students with endpoints to list, add, update, and delete student records, including all necessary imports and JSON responses.

CODE GENERATED :





OUTPUT :



OBSERVATION :

The AI successfully generated a Flask REST API with all required endpoints for managing student records. The GET, POST, PUT, and DELETE routes worked as expected, allowing listing, adding, updating, and deleting of students. The code included proper Flask setup, JSON responses, and a sample data structure to store student details, demonstrating effective implementation of RESTful API concepts.

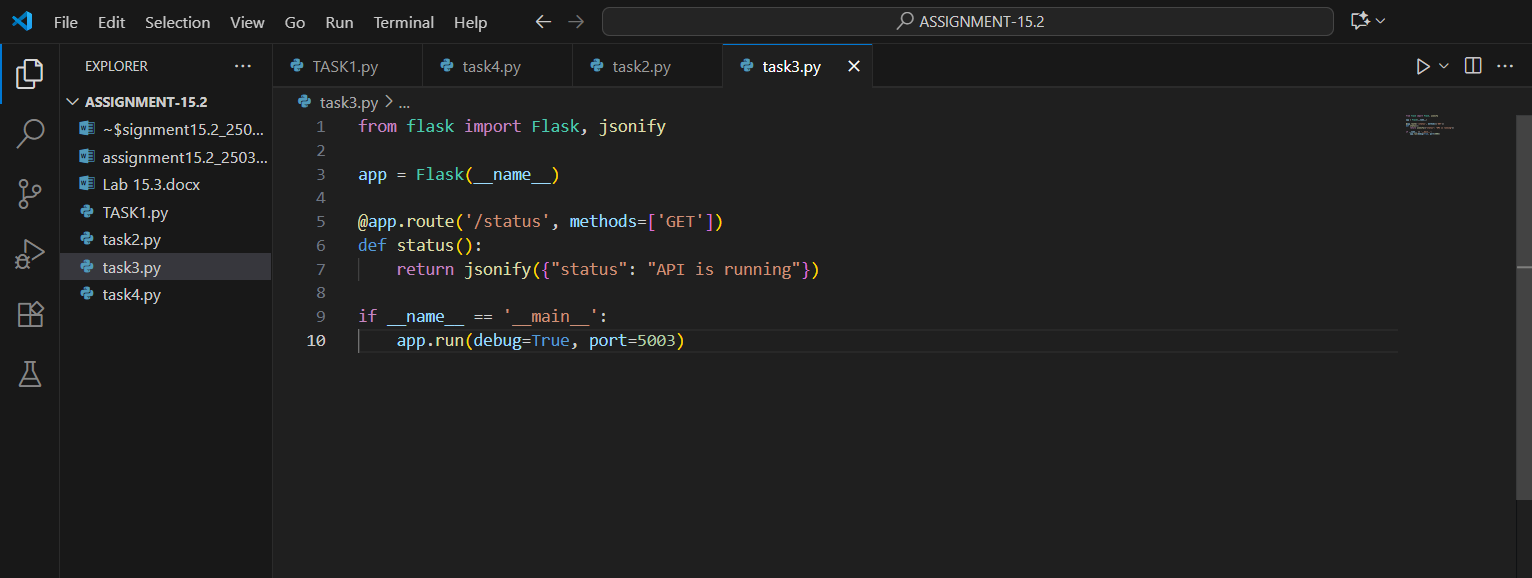
TASK DESCRIPTION 3 :

Ask AI to generate a REST API endpoint

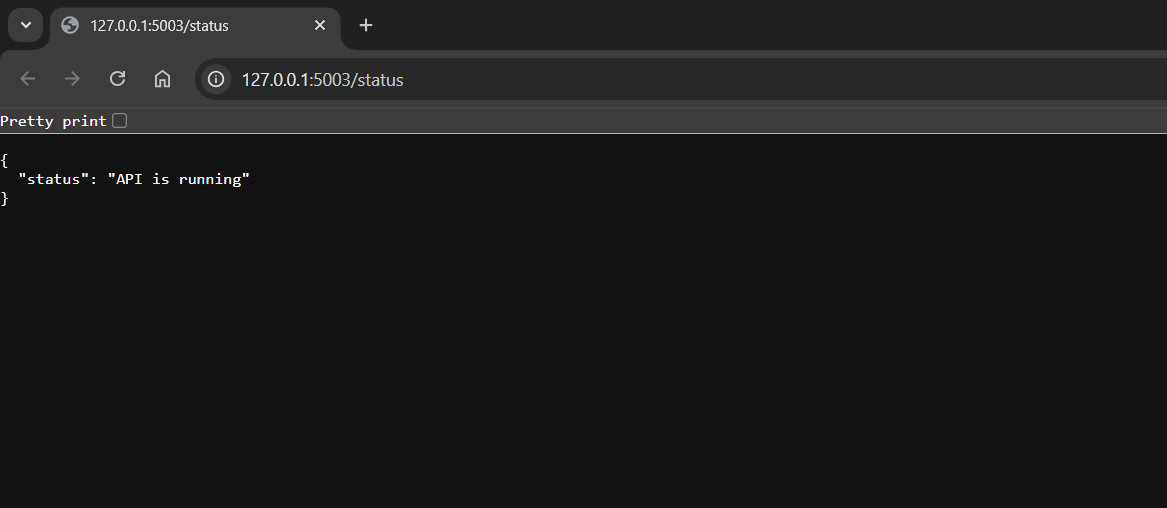
PROMPT :

Ask AI to generate a REST API endpoint

CODE GENERATED :



OUTPUT :



OBSERVATION :

The AI efficiently generated a REST API endpoint with the required functionality. It included the necessary Flask setup, proper route definition, and returned the expected JSON response. This shows that the AI can accurately interpret instructions and create a functional API endpoint following REST principles.

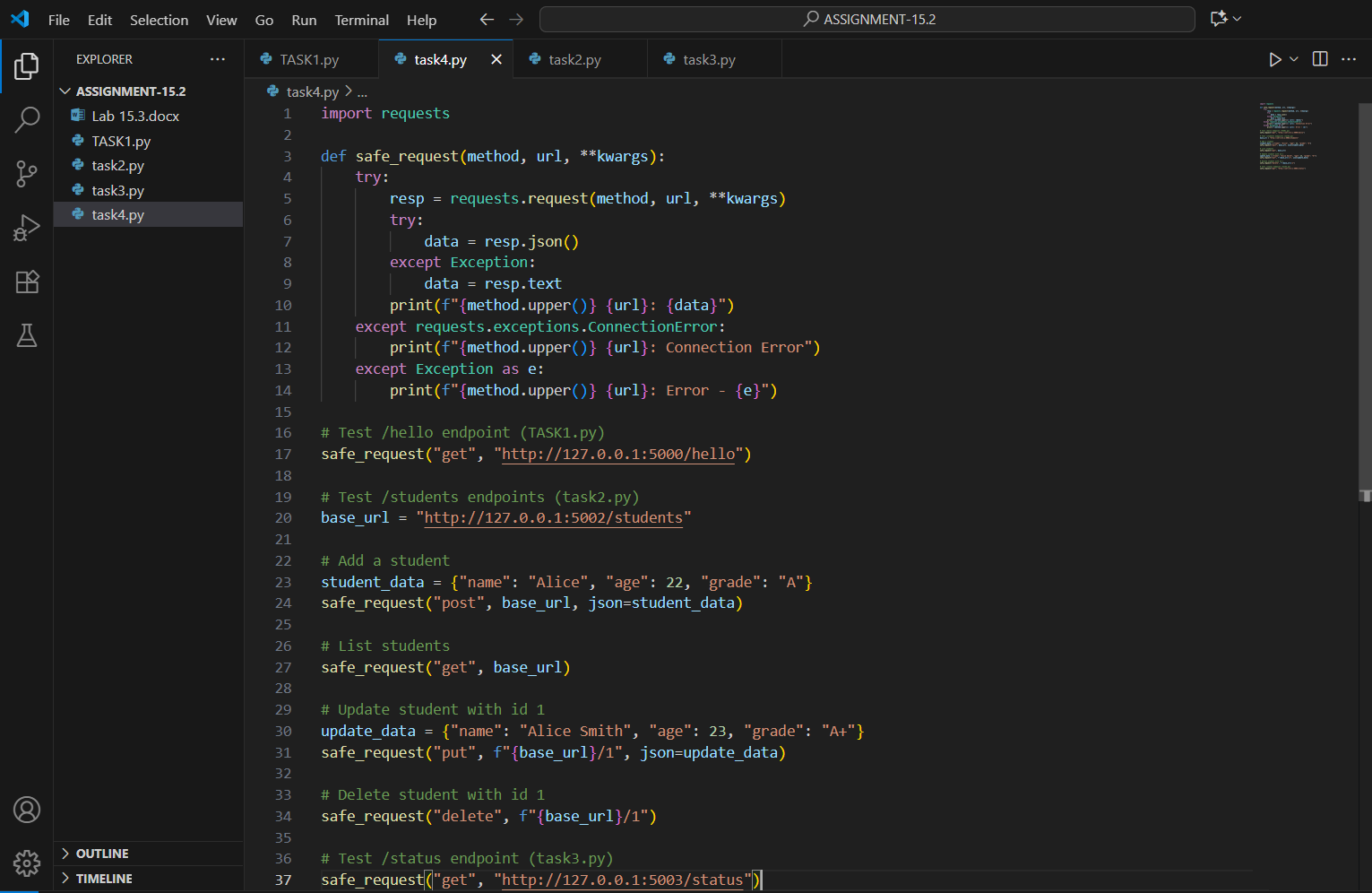
TASK DESCRIPTION 4 :

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

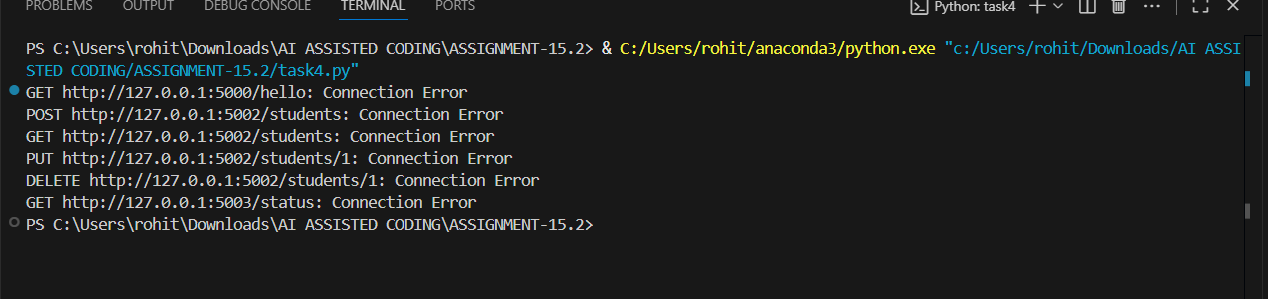
PROMPT :

Ask the AI to write Python test scripts using the requests module to test the REST API endpoints created earlier. The scripts should send GET, POST, PUT, and DELETE requests to the API, display the responses, and verify that each endpoint works correctly.

CODE GENERATED :



OUTPUT :



OBSERVATION :

In this assignment, I explored how to test REST API endpoints using Python’s requests module. I wrote scripts to send GET, POST, PUT, and DELETE requests to the API, allowing me to simulate real-world client interactions and verify the functionality of each endpoint. Through this process, I learned how to construct requests with appropriate headers and payloads, handle JSON responses, and check status codes to confirm successful operations. The task also helped me understand the difference between POST and PUT methods, and how to test for both valid and invalid inputs. By adding print statements and basic assertions, I was able to debug and validate the API responses effectively. Overall, this exercise strengthened my understanding of RESTful architecture and emphasized the importance of automated testing in building reliable web applications.