# **ASSIGNMENT-15.1**

**NAME : B.Divija**

**HALL TICKET NO : 2403A51302**

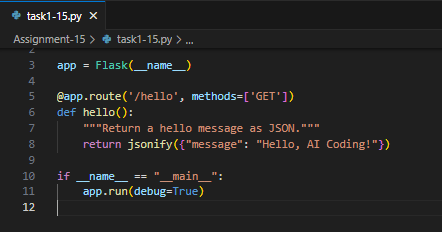
**BATCH : 13**

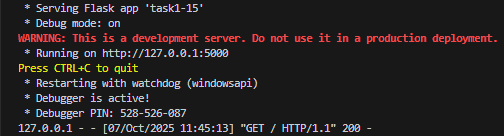
**Task-1 Description:** Basic REST API Setup  
Task: Ask AI to generate a Flask REST API with one route: GET /hello → returns {"message": "Hello, AI Coding!"}

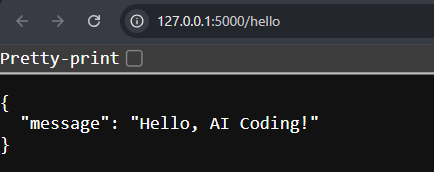
**Prompt:** Generate a Flask REST API with one route: GET /hello → returns {"message": "Hello, AI Coding!"}

**Code**

**Generated:**

****

**Output:**

****

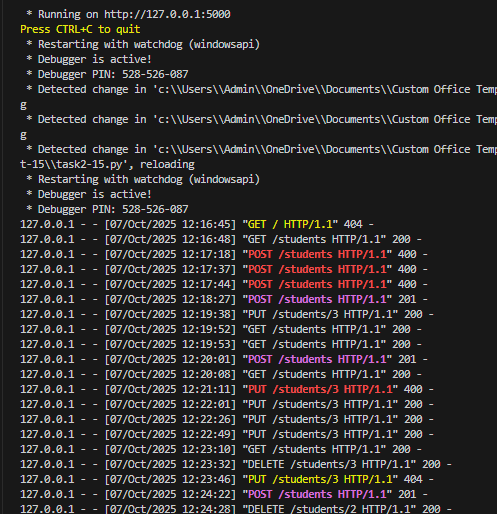
**Observation:** The Flask REST API was successfully created with a single route /hello. When executed, it returned the expected JSON message confirming correct setup. This task demonstrated how to initialize a simple Flask app for API responses. Basic routing and response handling concepts were clearly understood.

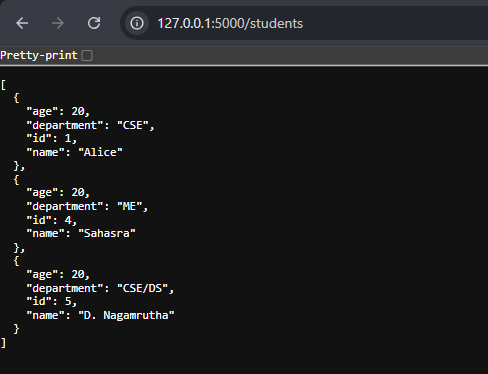
**Task-2 Description:** CRUD Operations (Students API)  
Task: Use AI to build REST endpoints for a Student API:  
• GET /students → List all students.  
• POST /students → Add a new student.  
• PUT /students/<id> → Update student details.  
• DELETE /students/<id> → Delete a student.

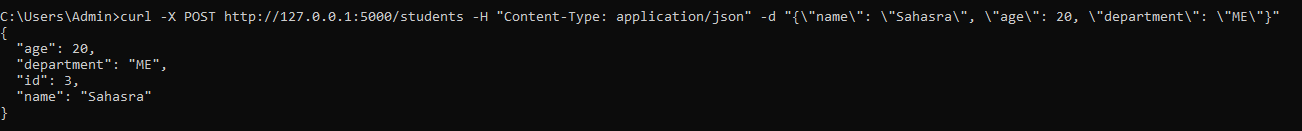
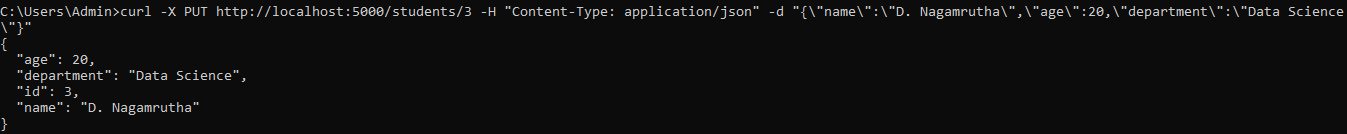
**Prompt:** Build REST endpoints for a Student API:  
• GET /students → List all students.  
• POST /students → Add a new student.  
• PUT /students/<id> → Update student details.  
• DELETE /students/<id> → Delete a student.

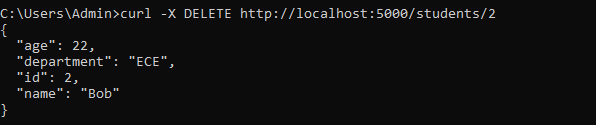
**Code Generated:**



**Output:**



**Commands Used:**

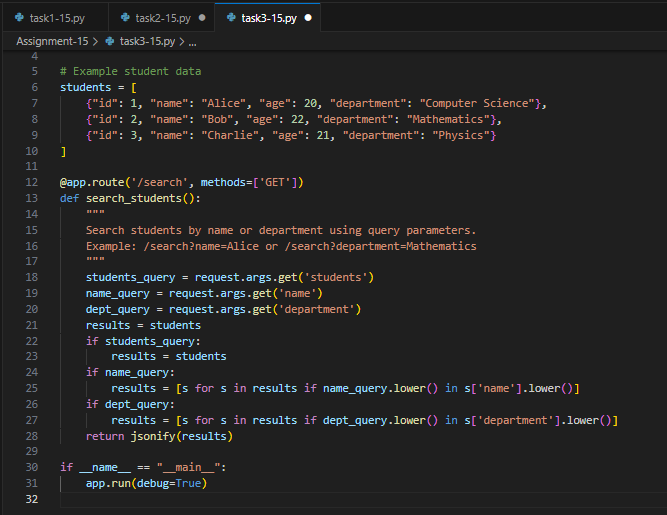


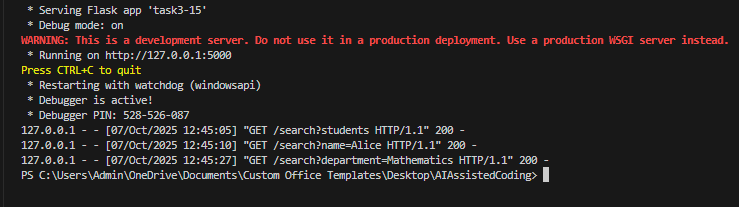
**Observation:** A complete CRUD-based Student API was implemented using Flask. Endpoints for GET, POST, PUT, and DELETE worked as intended. Data could be created, viewed, modified, and removed successfully. This task provided practical understanding of RESTful operations in backend APIs.

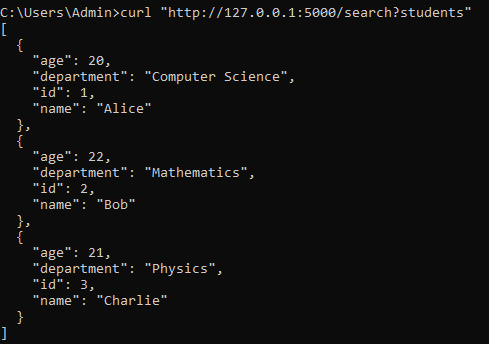
**Task-3 Description:** API with Query Parameters  
Task: Ask AI to generate a REST API endpoint.

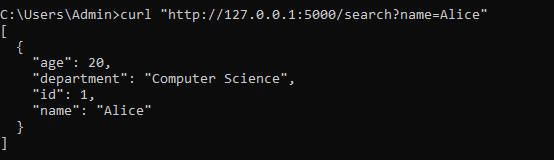
**Prompt:** Generate a REST API endpoint with Working search function with query param handling.

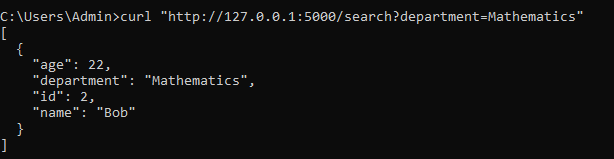
**Code Generated:**



**Output:**

**Query to get all students:**

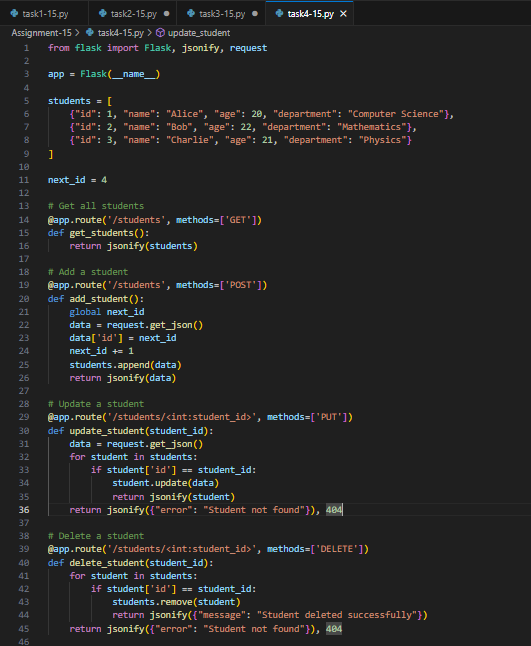
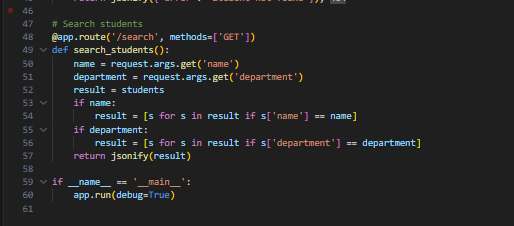
**Query to get student by name:**

**Query to get student by department:**

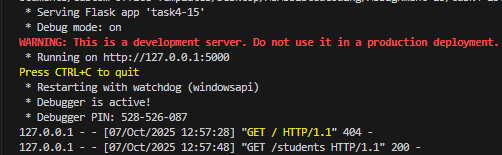
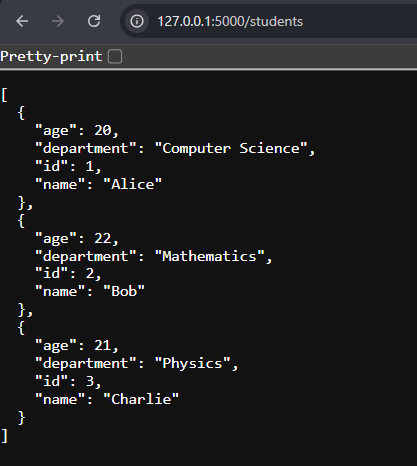
**Observation:** The API was enhanced with query parameter handling for search functionality.  
Users could retrieve students by name or department efficiently. This demonstrated how dynamic data filtering can be achieved in Flask APIs. Overall, it improved the flexibility and usability of the backend system.

**Task-4 Description:** – Integration & Testing  
Task: Ask AI to write test scripts using Python requests module to call APIs created above.

**Prompt:** Write test scripts using python requests module to call APIs created before/above.

**Code Generated:**

**Output:**

**Observation:** Python requests module was used to test all the API endpoints. Each request returned appropriate JSON responses confirming functionality. The testing scripts validated both data integrity and route accuracy. This task reinforced API testing and integration verification skills.