AI ASSISTED CODING

NAME : Rokalboyana Varshith

ROLL NO: 2403A510D7

ASSIGNMENT : 15.1

#TASK-1

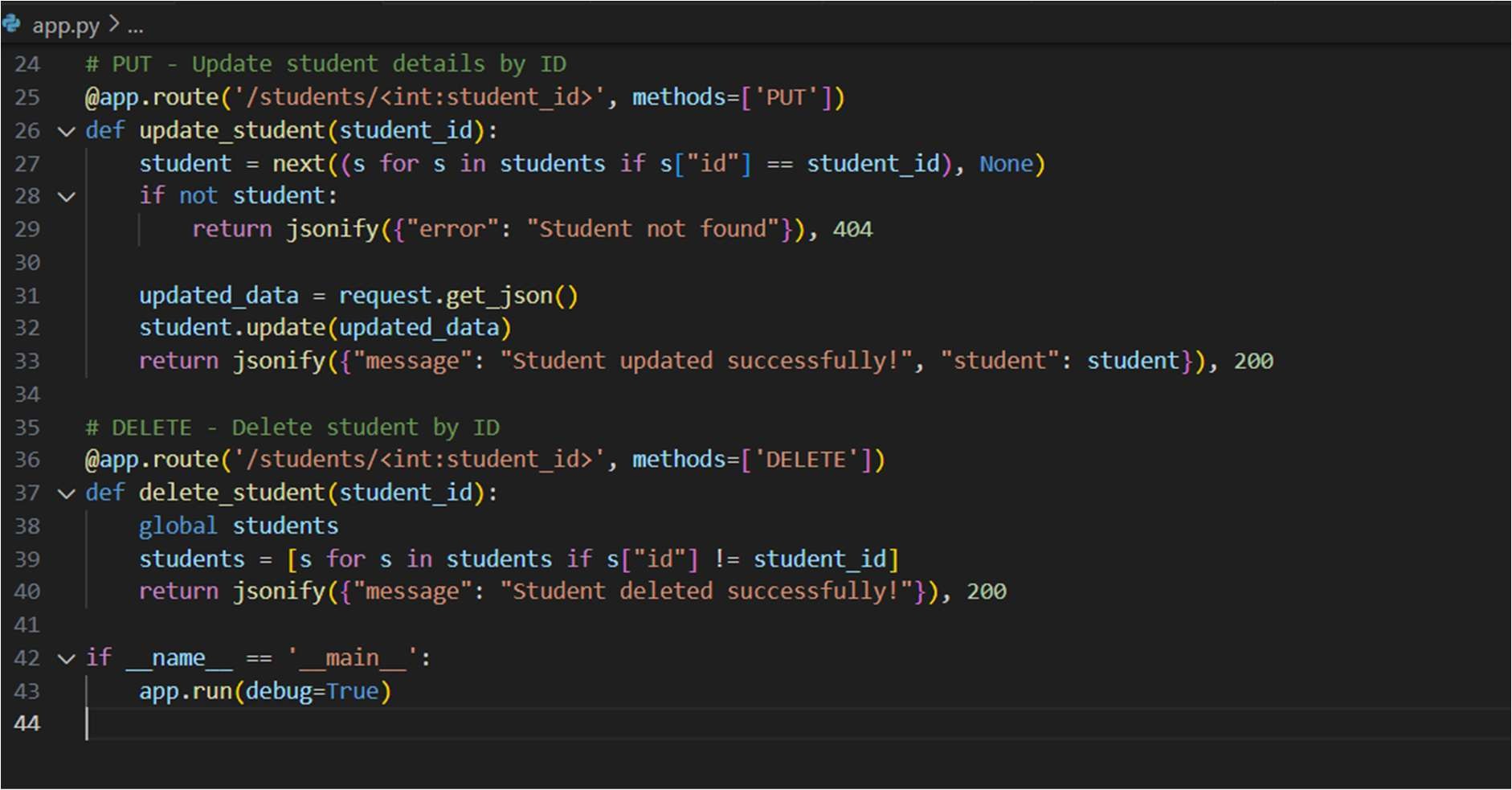
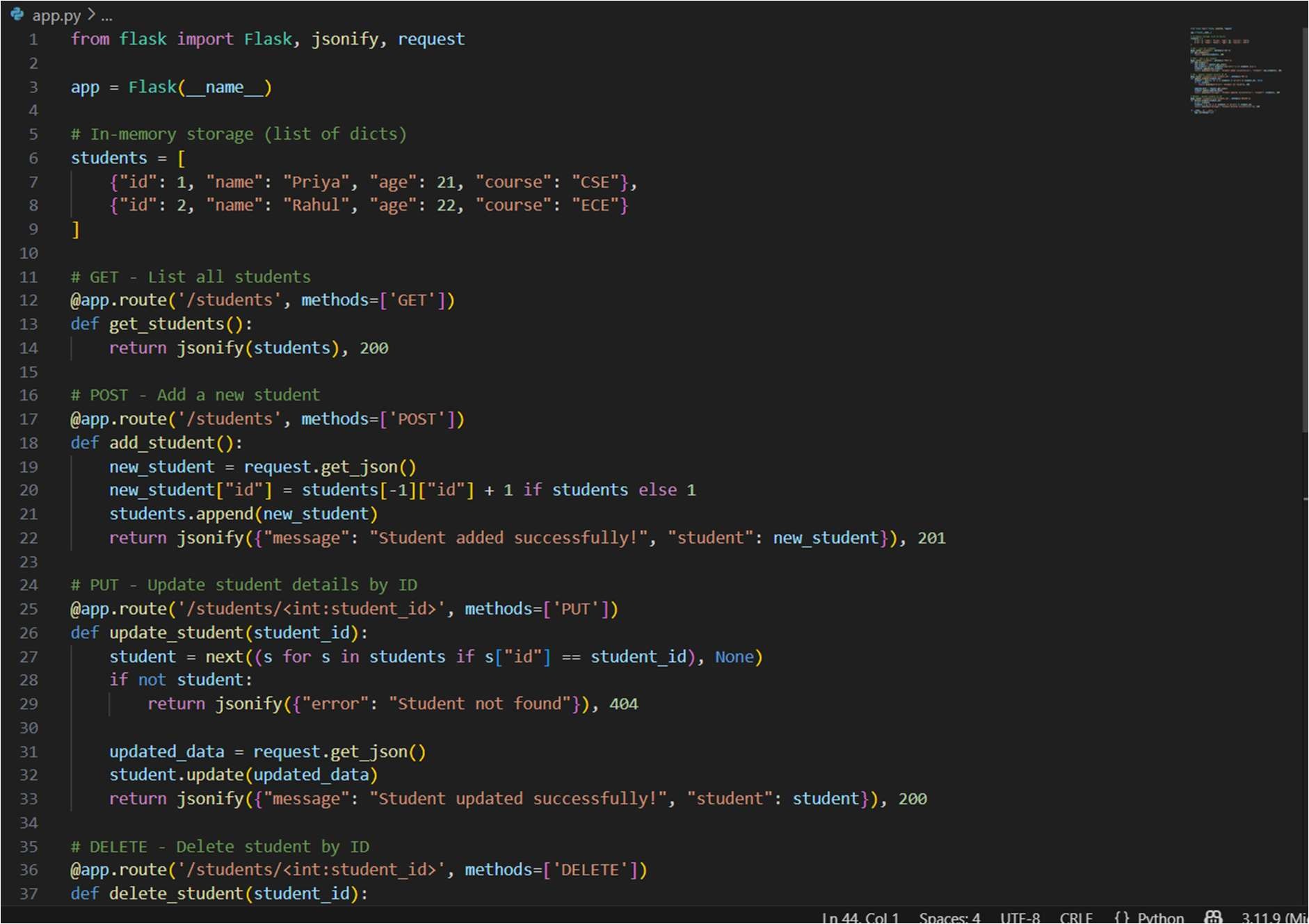
PROMPT: Build a RESTful API for managing student records.

Instructions:

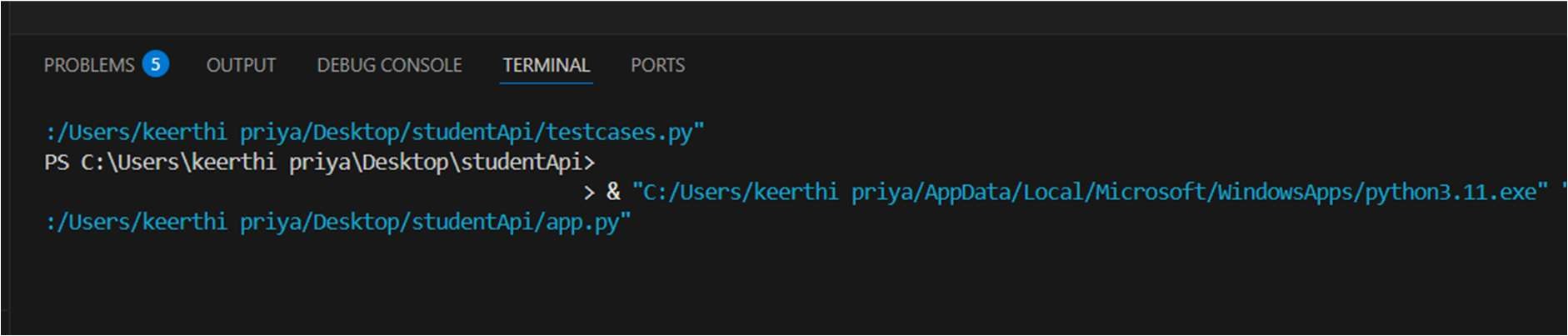
• Endpoints required:

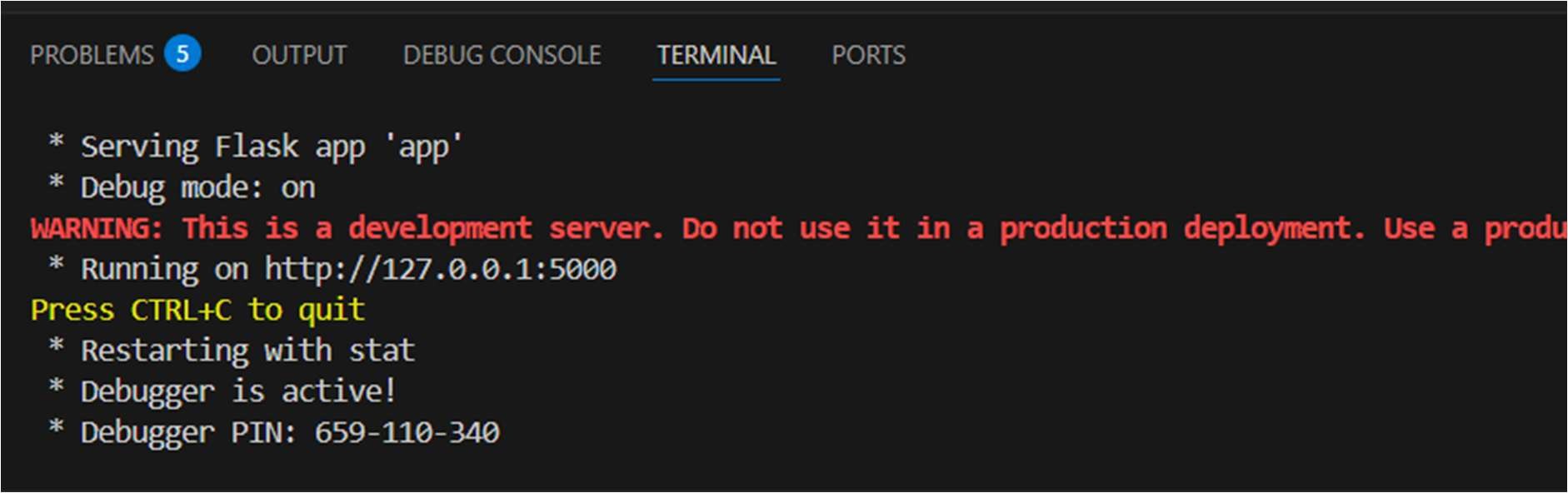
o GET /students → List all students o POST /students → Add a new student o PUT /students/{id} → Update student details o DELETE /students/{id} → Delete a student record

* Use an in-memory data structure (list or dictionary) to store records.
* Ensure API responses are in JSON format. Expected Output:
* Working API with CRUD functionality for student records CODE:



OUTPUT :





OBSERVATION:

Flask server starts successfully on http://127.0.0.1:5000.

Returns all student records as a JSON array.

Initial data is the hardcoded list of students.

HTTP status code: 200 OK.

All student records are stored in a Python list.

#TASK-2

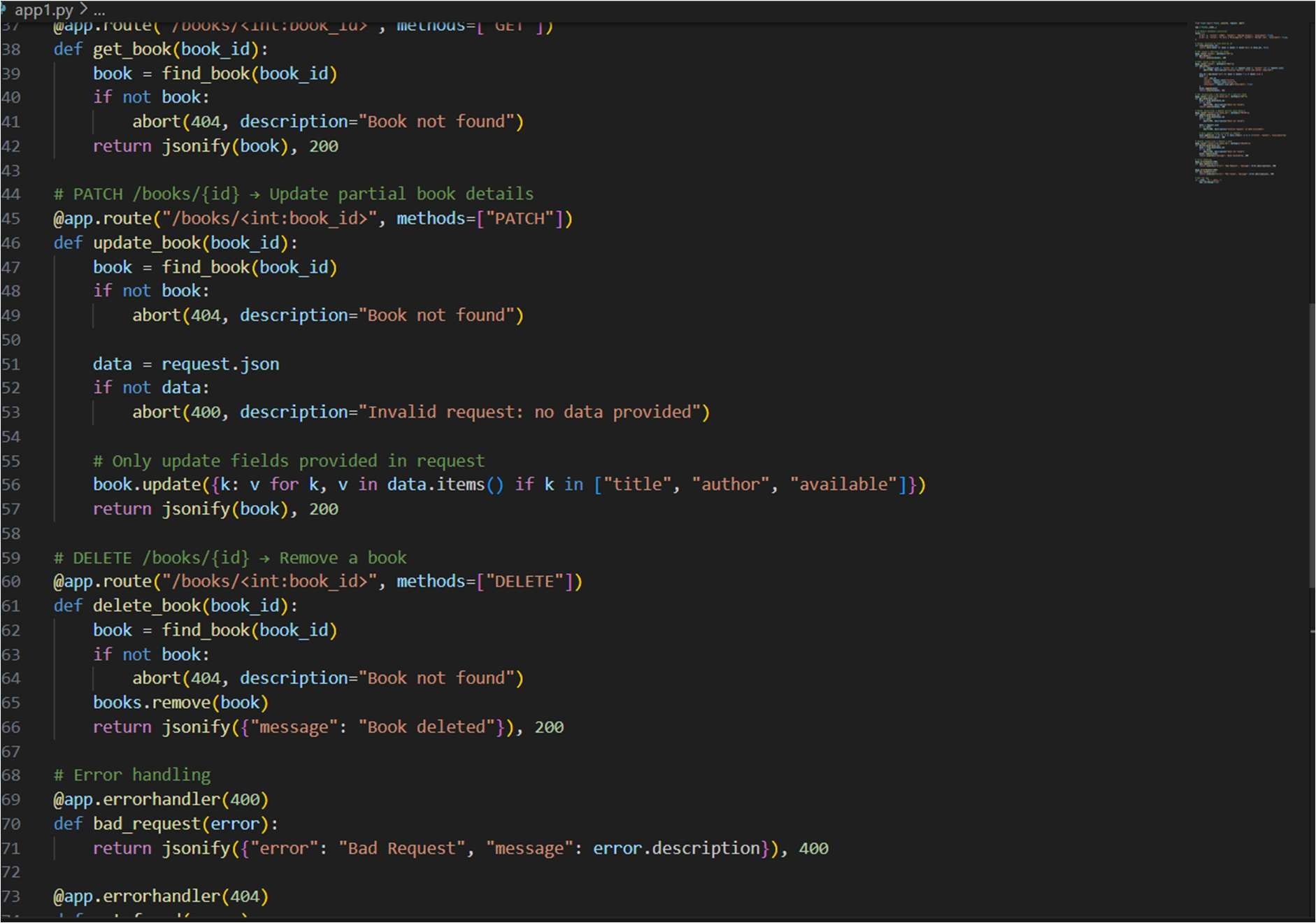
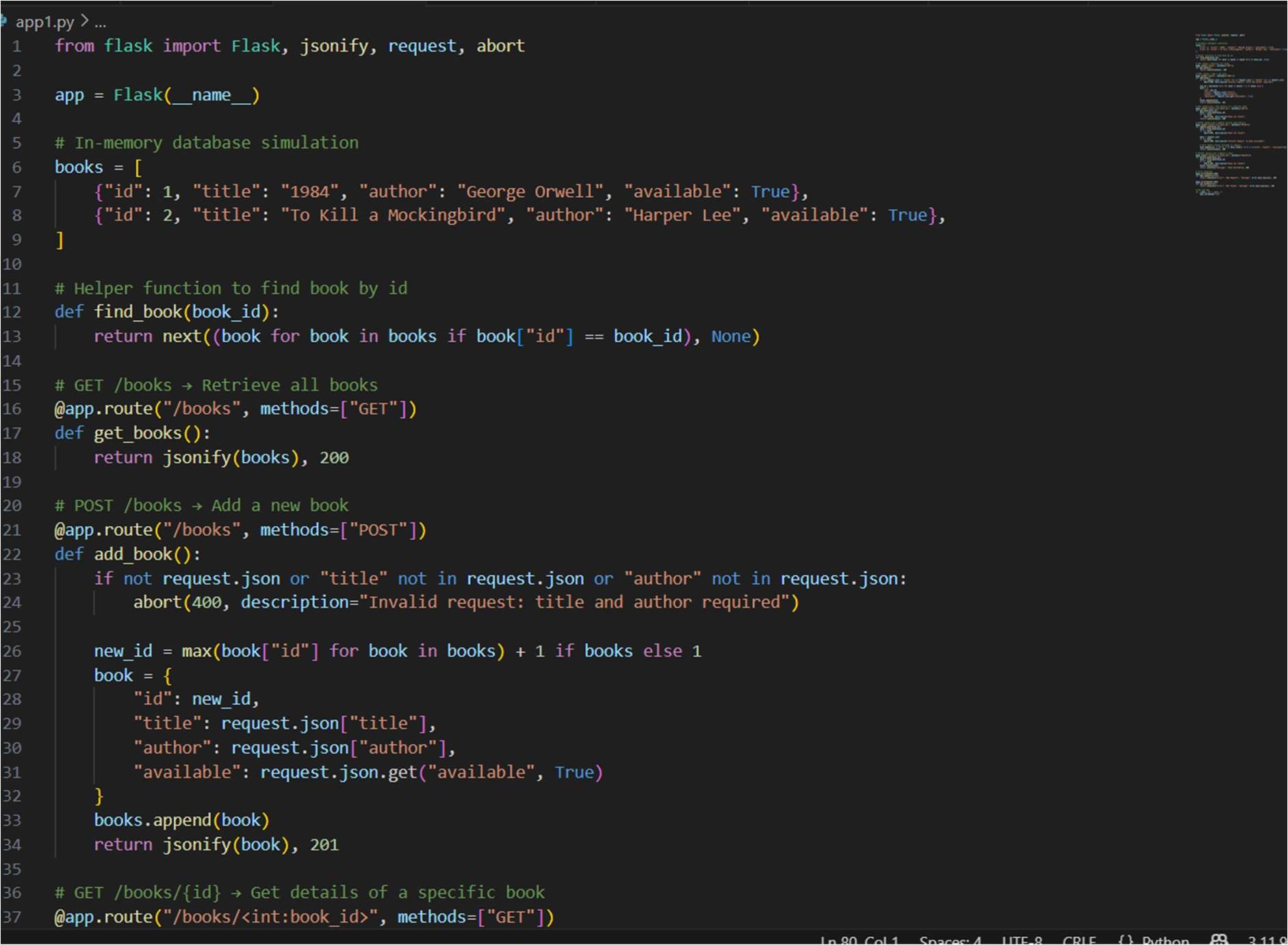
PROMPT:

Develop a RESTful API to handle library books.

Instructions:

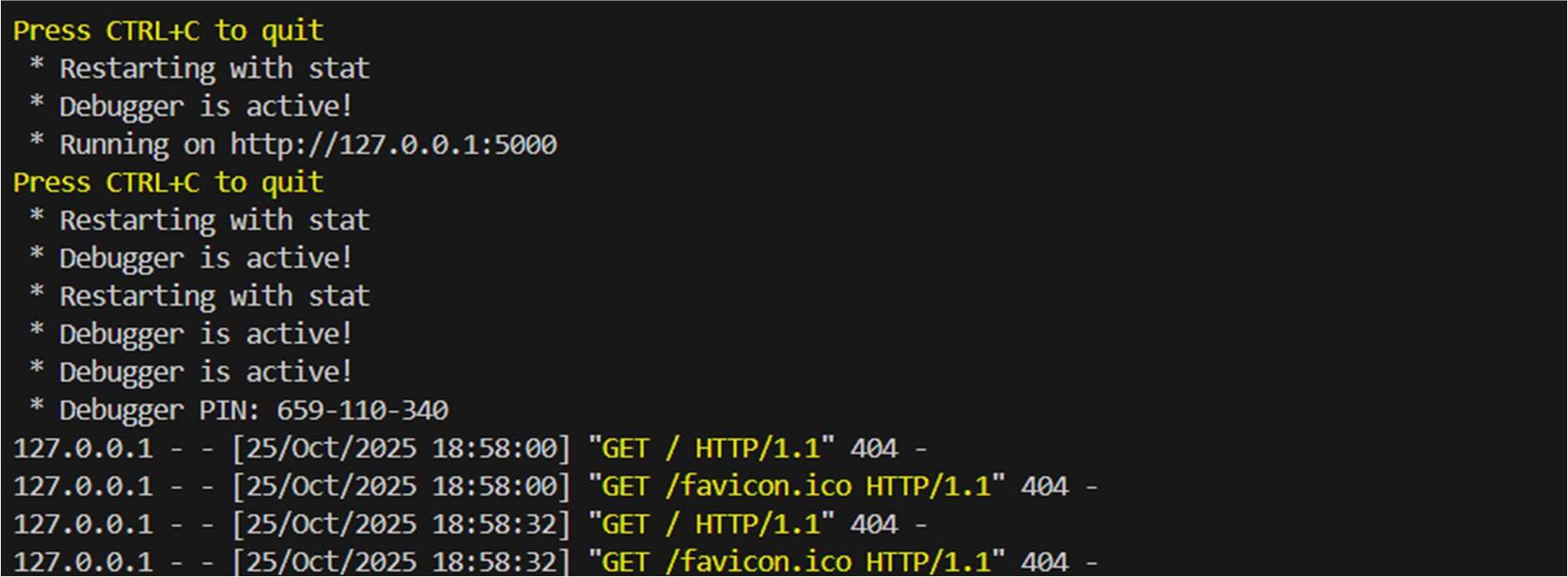
• Endpoints required: o GET /books → Retrieve all books o POST /books → Add a new book o GET /books/{id} → Get details of a specific book o PATCH /books/{id} → Update partial book details (e.g., availability)

o DELETE /books/{id} → Remove a book • Implement error handling for invalid requests CODE:





OUTPUT:



OBSERVATION:

Adds a new book to the in-memory list.

Automatically generates id.

Optional field available defaults to True if not provided.

Response includes the new book object

#TASK-3

PROMPT :

Create an API for managing employees and their salaries.

Instructions:

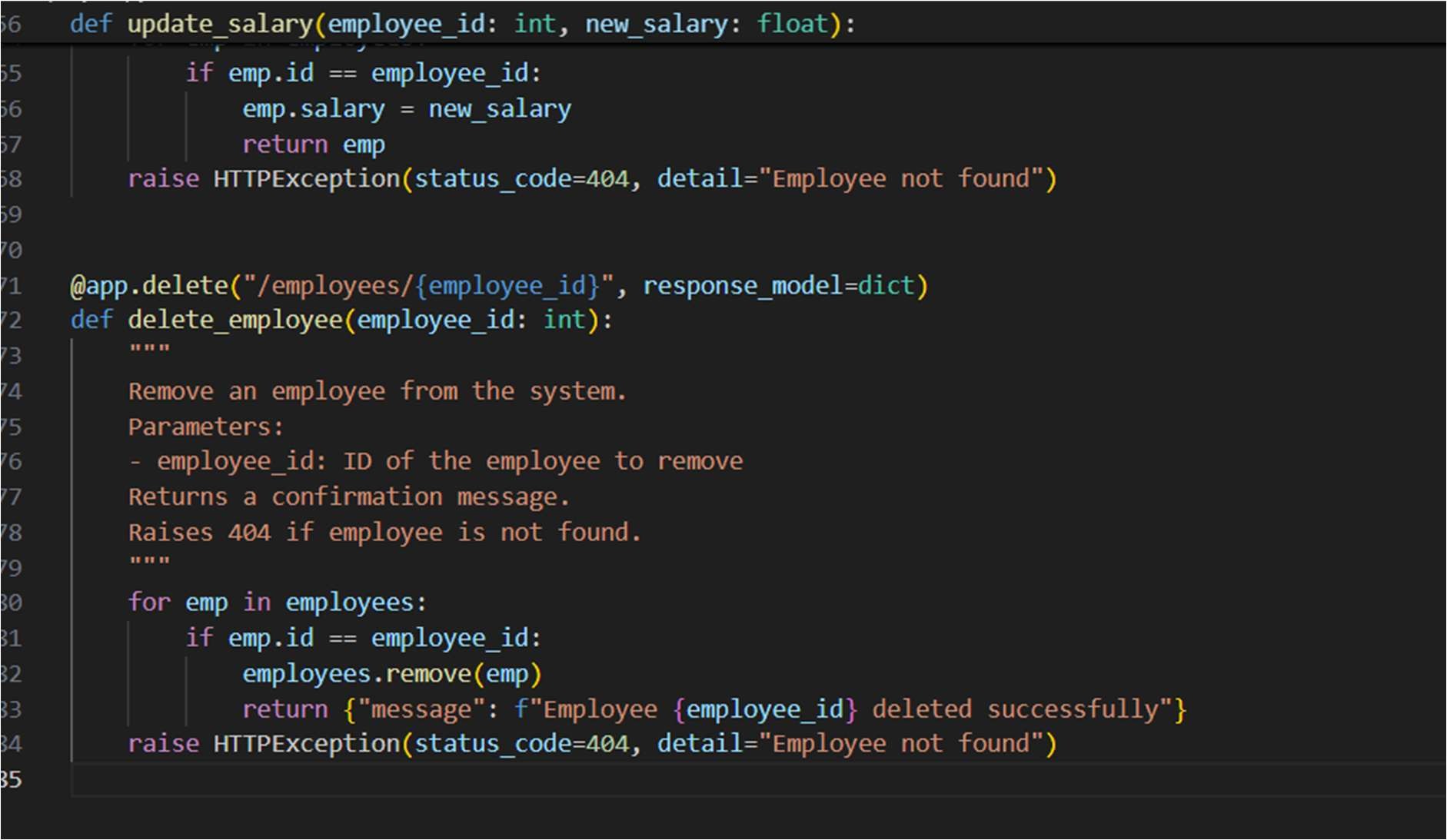
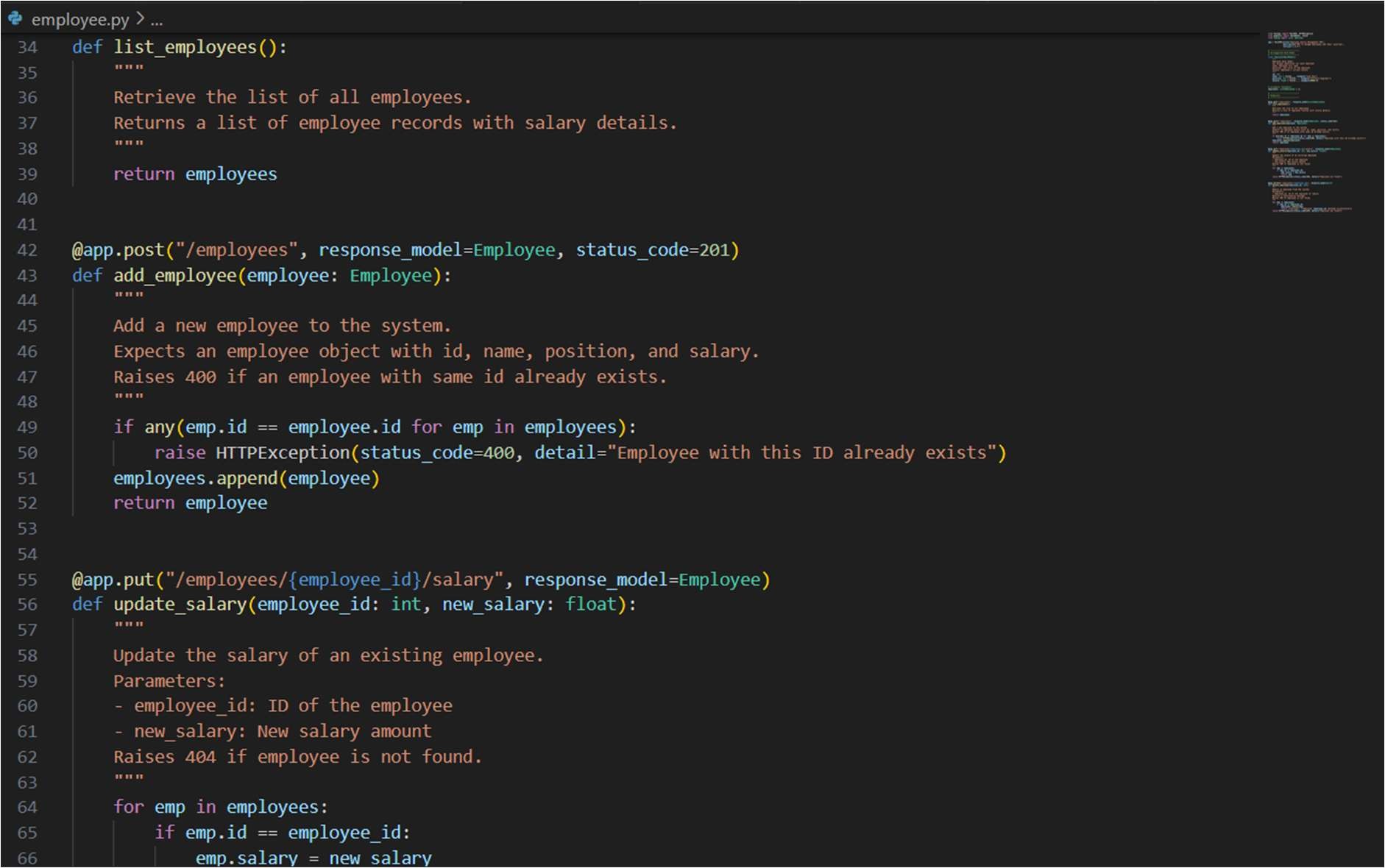
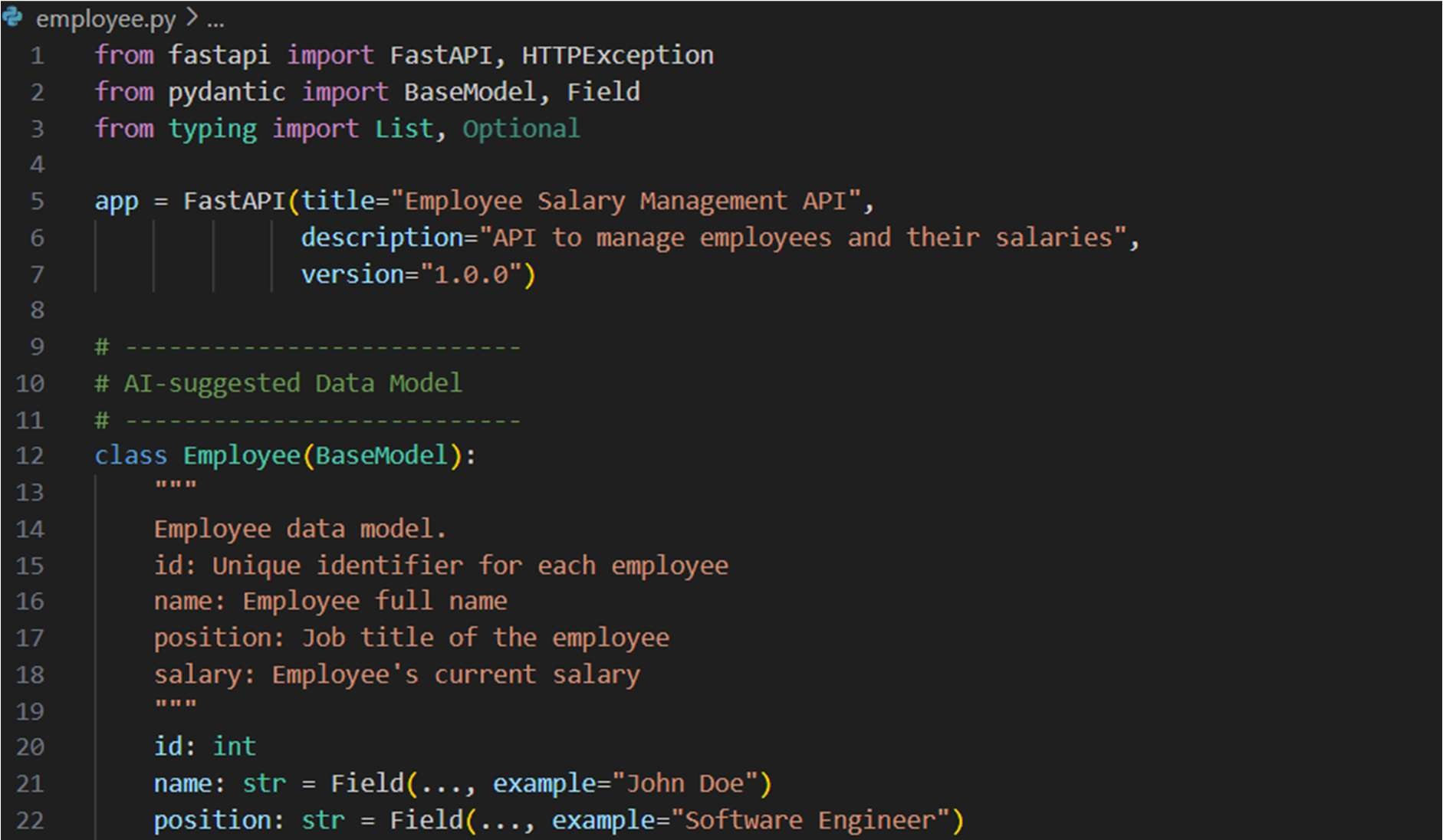
• Endpoints required:

* GET /employees → List all employees o POST /employees → Add a new employee with salary details
* PUT /employees/{id}/salary → Update salary of an employee
* DELETE /employees/{id} → Remove employee from system

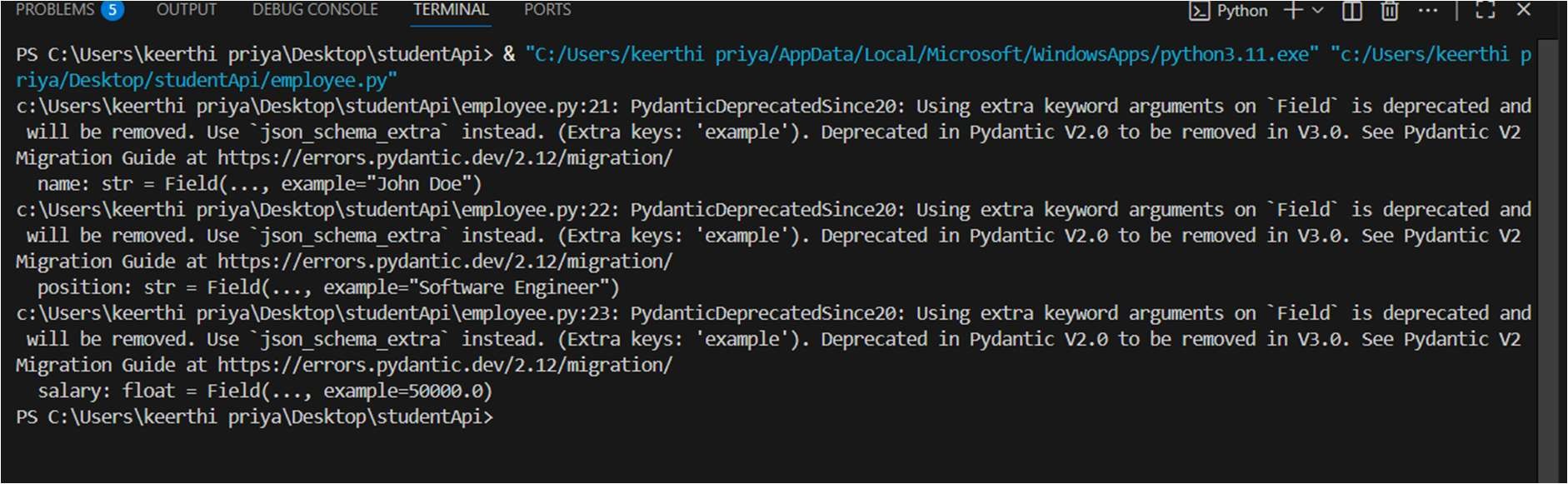
• Use AI to:

* Suggest data model structure.
* Add comments/docstrings for all endpoints

CODE:



OUTPUT:



OBSERVATION:

Employees are stored in a Python list (employees: List[Employee]).

Data is volatile — restarting the server resets the list.

Supports multiple employees with unique IDs.

#TASK-4

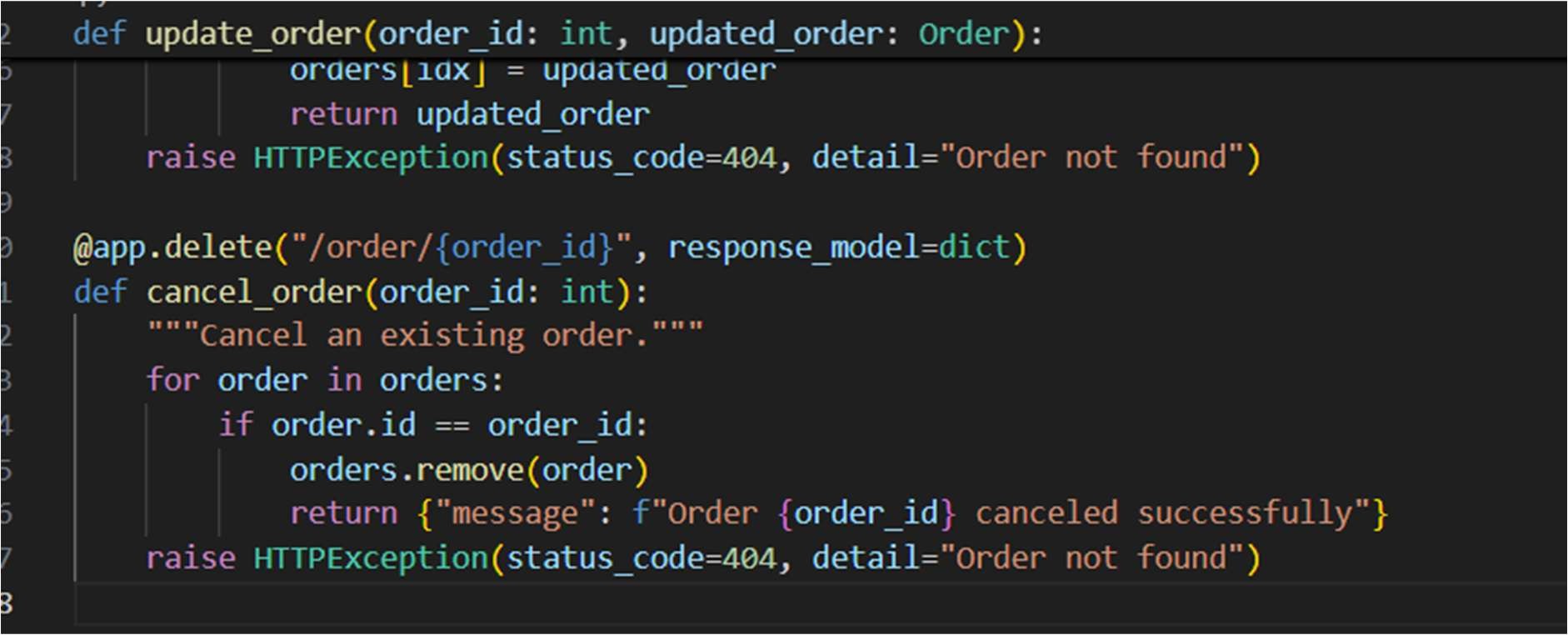
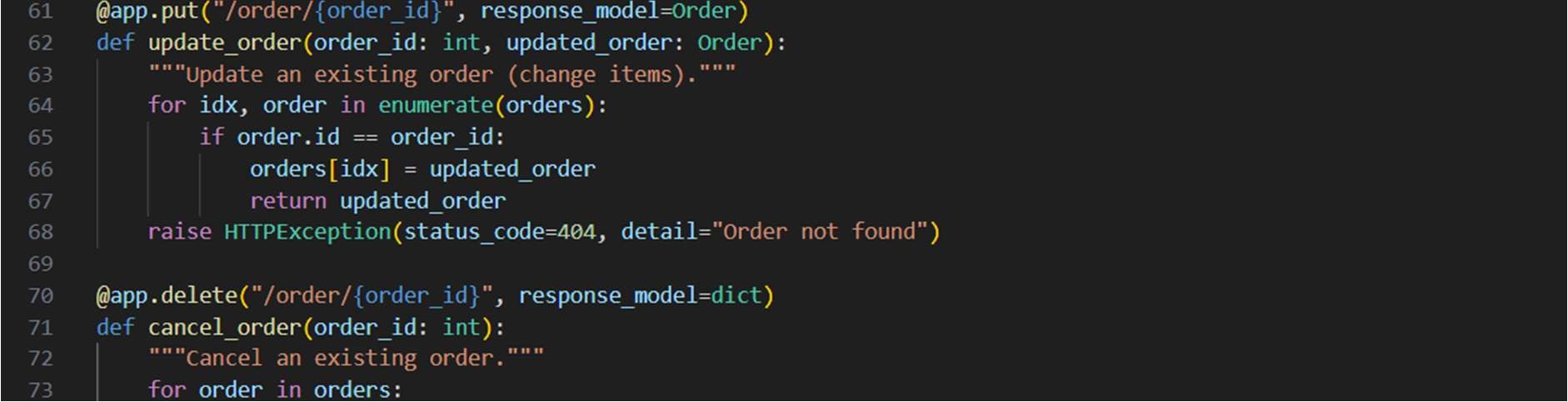
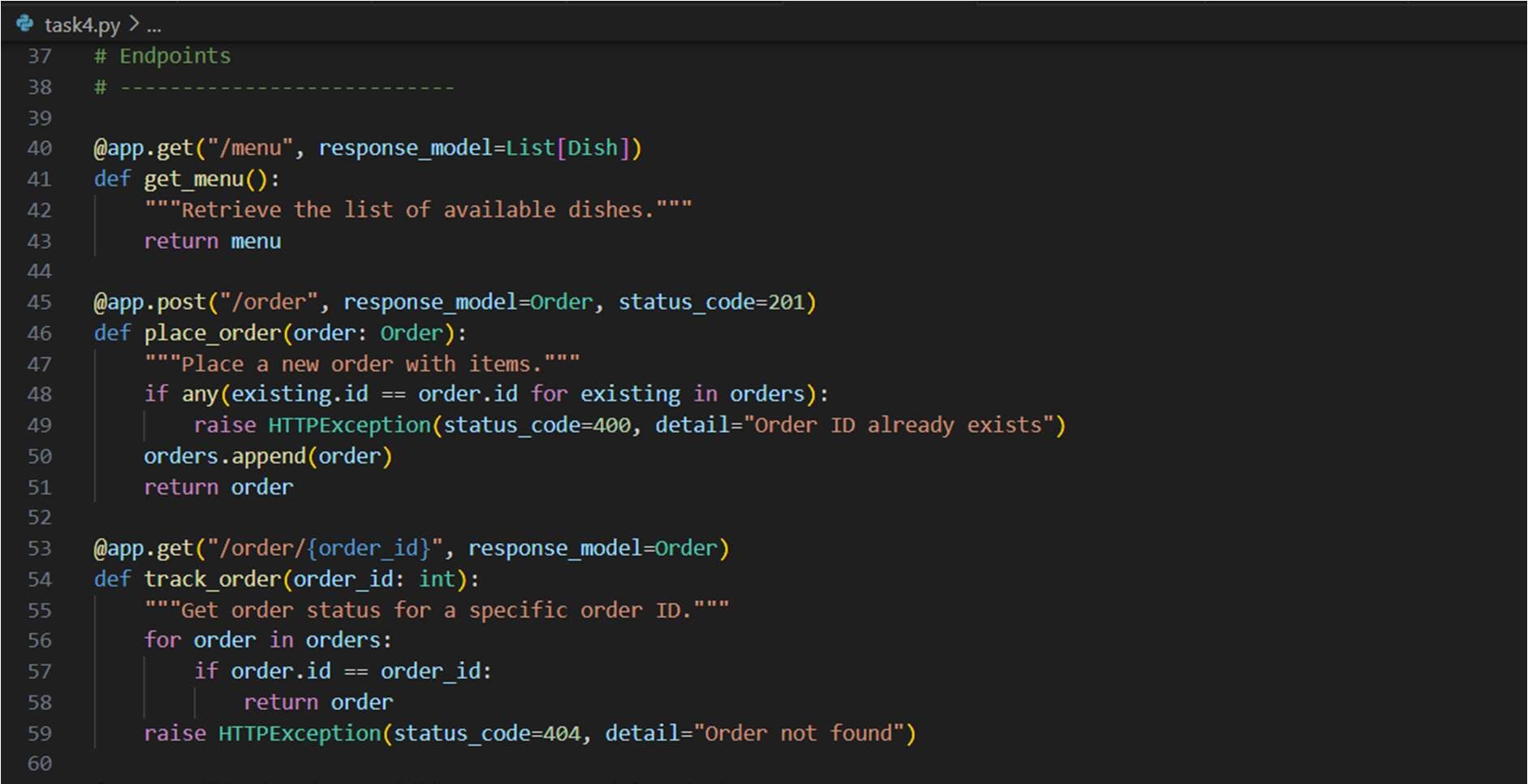
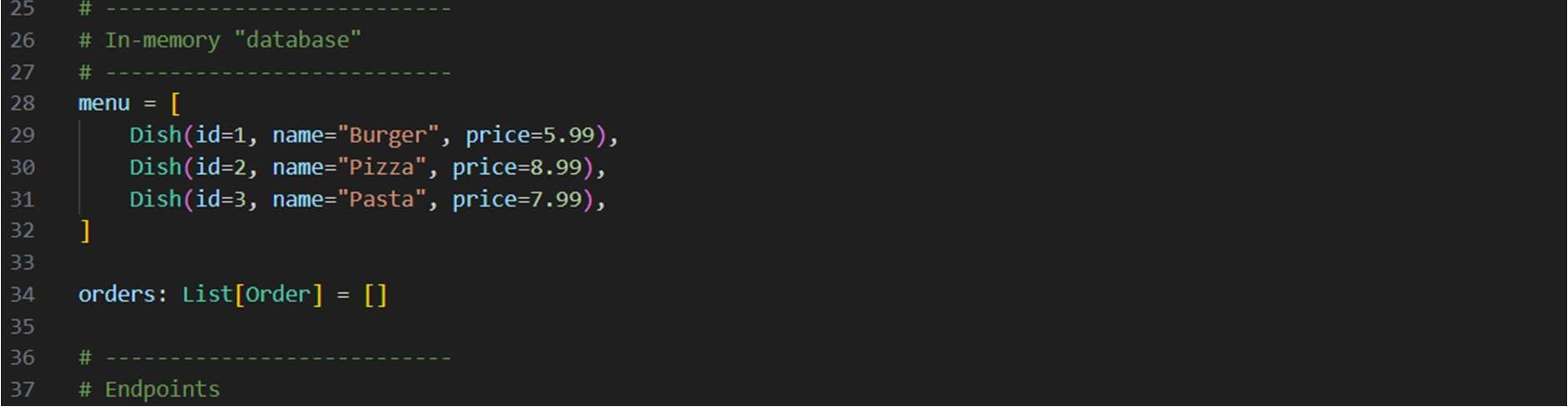
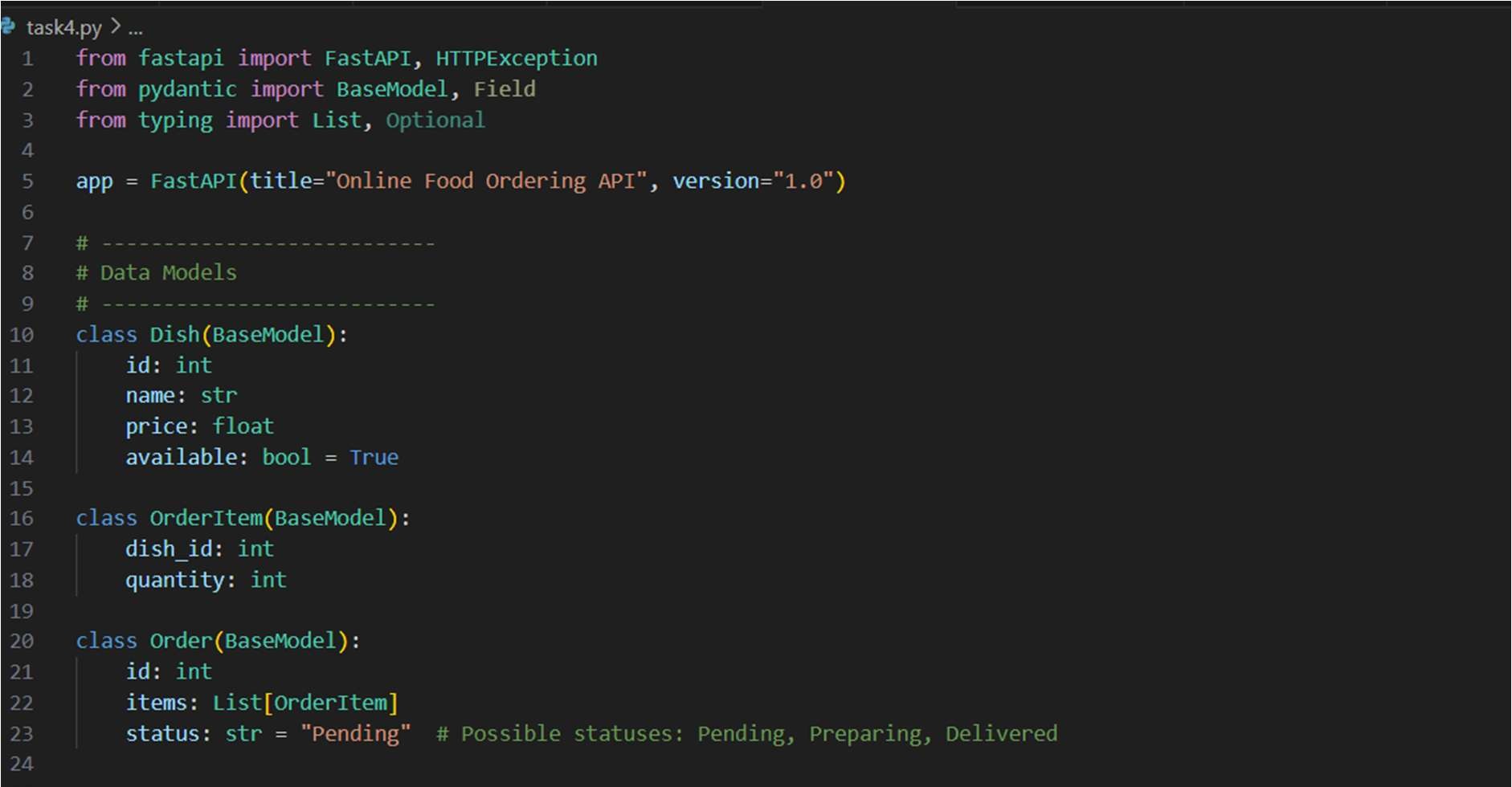
PROMPT:

Design a simple API for an online food ordering system.

Requirements:

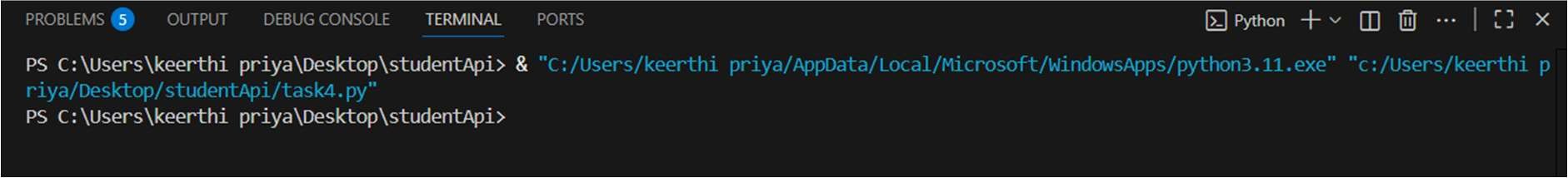
• Endpoints required:

* GET /menu → List available dishes o POST /order → Place a new order o GET /order/{id} → Track order status
* PUT /order/{id} → Update an existing order (e.g., change items) o DELETE /order/{id} → Cancel an order • AI should generate: o REST API code o Suggested improvements (like authentication, pagination) CODE:





OUTPUT:



OBSERVATION:

Menu Retrieval Works – GET /menu correctly lists all available dishes.

Order Placement Works – POST /order allows creating a new order with items.

Order Tracking Works – GET /order/{id} returns the correct order and status.

Order Update Works – PUT /order/{id} successfully updates order items.

Order Cancellation Works – DELETE /order/{id} removes the order and confirms deletion.