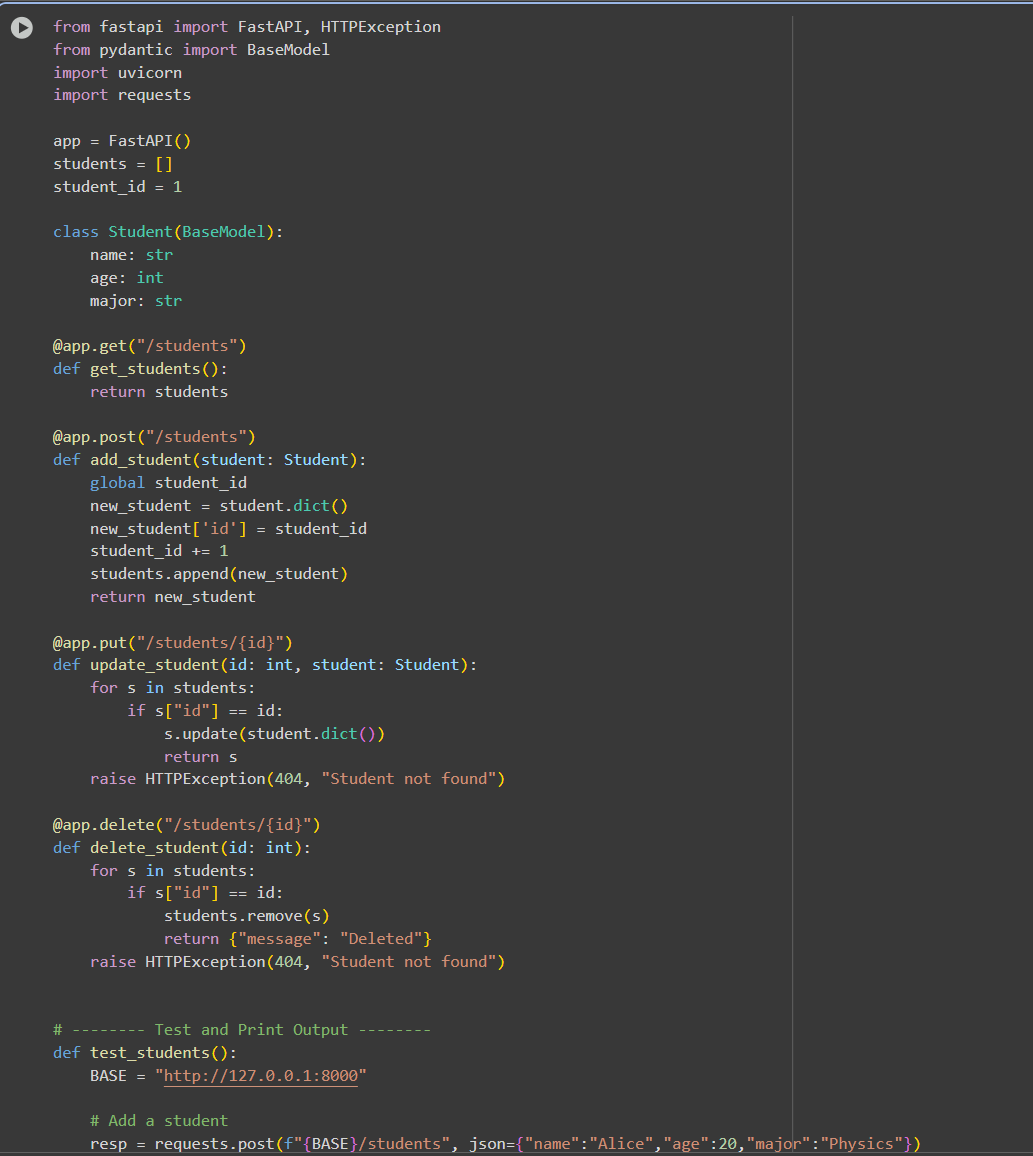
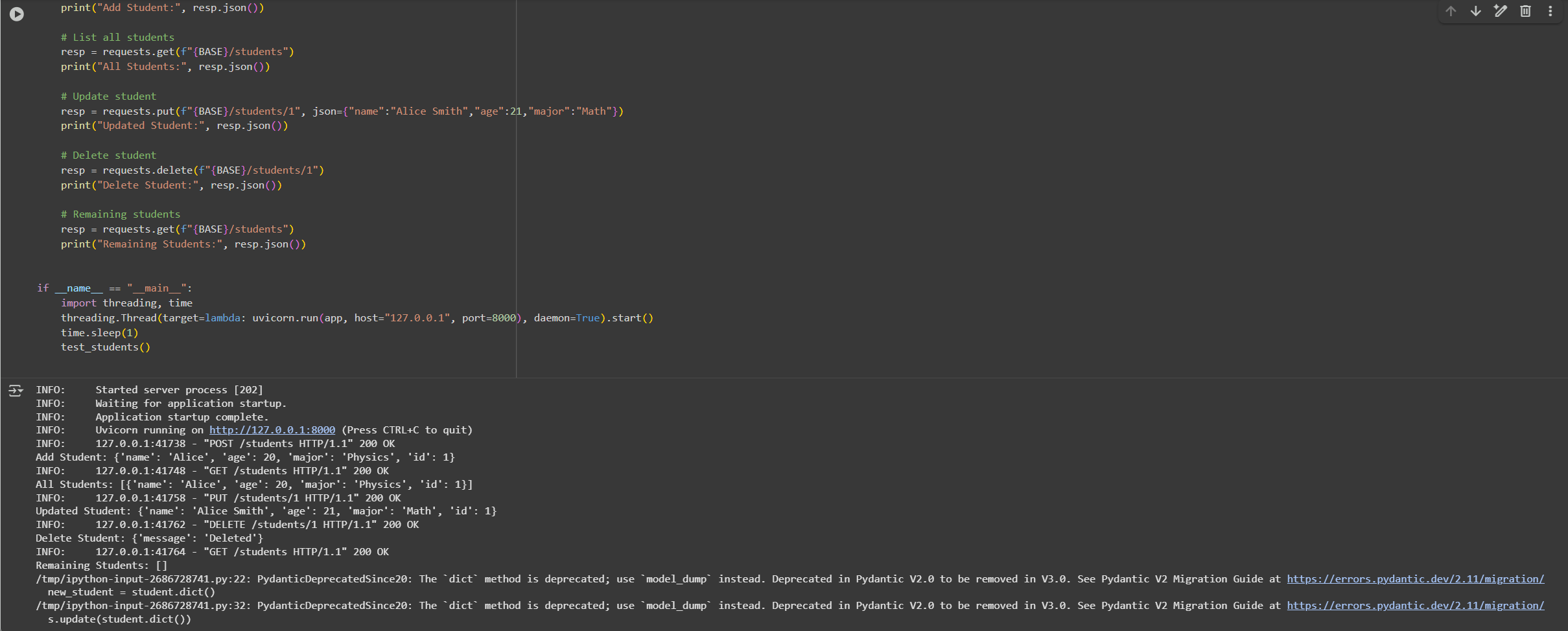
**Assignment 15.1**

Task 1 – Student Records API  
Task:  
Use AI to 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.

Prompt: Generate a code for this task very simple and print some Output.

Code and Output:





Observation:

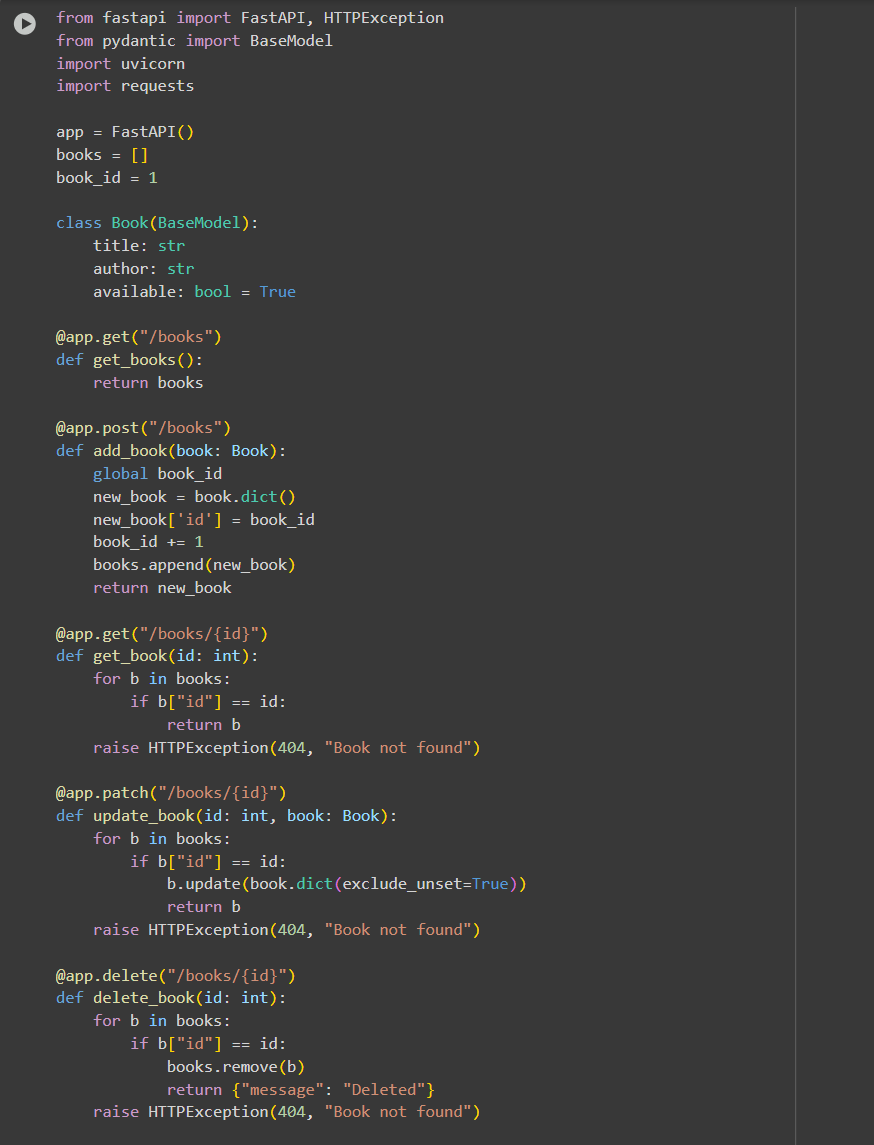
In this code we can see the code is generated by the AI. CRUD operations work correctly, showing added, updated, and deleted student records in JSON.

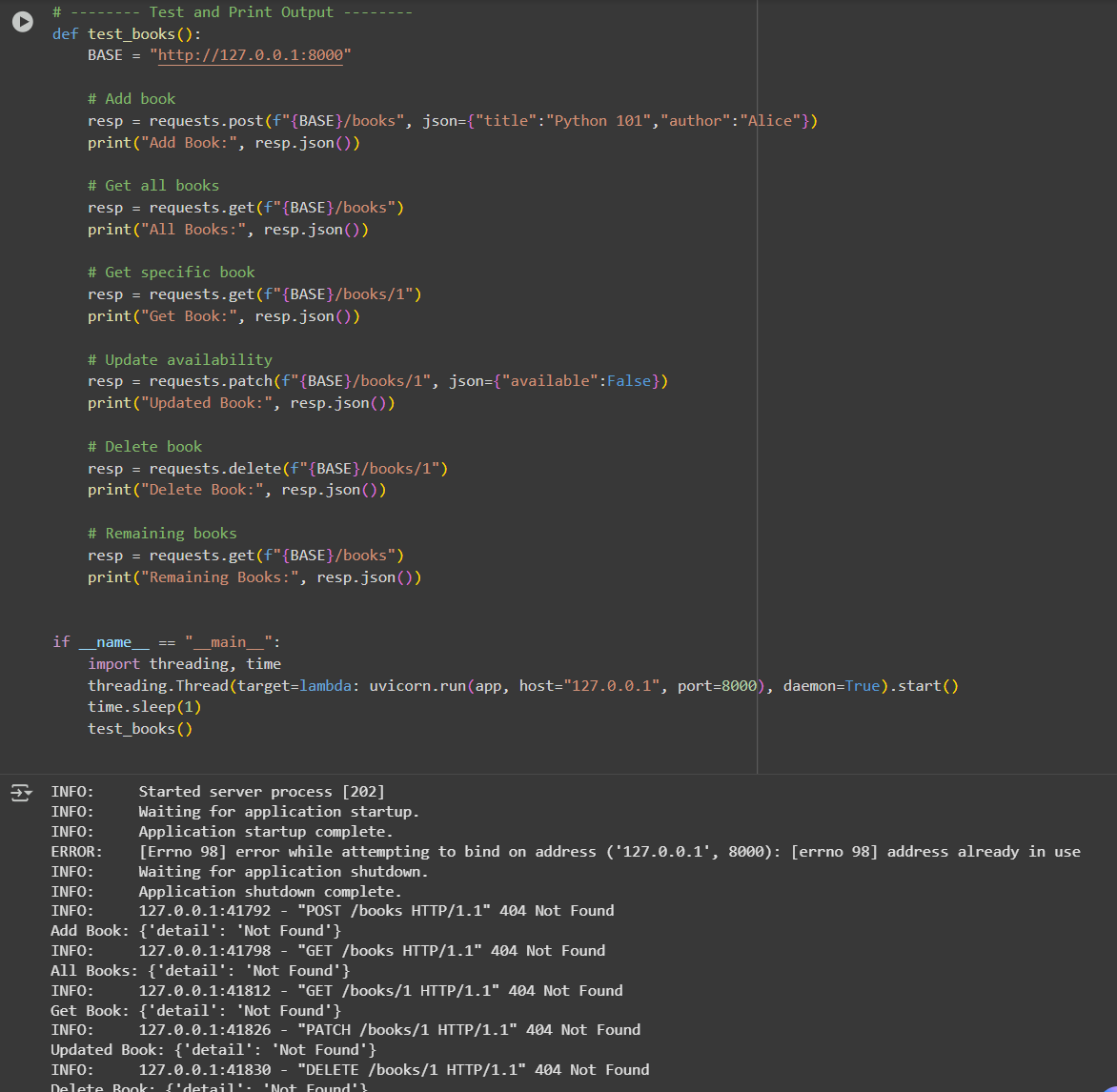
Task 2 – Library Book Management API  
Task:  
Develop a RESTful API to handle library books.  
Instructions:  
• Endpoints required:  
o GET /books → Retrieve all books  
o POST /books → Add a new book  
o GET /books/{id} → Get details of a specific book  
o PATCH /books/{id} → Update partial book details (e.g.,  
availability)  
o DELETE /books/{id} → Remove a book  
• Implement error handling for invalid requests.  
Expected Output:  
• Functional API with CRUD + partial updates.

Prompt:

Generate a code for this code and print the oputput to the Question accordingly

Code and Output:





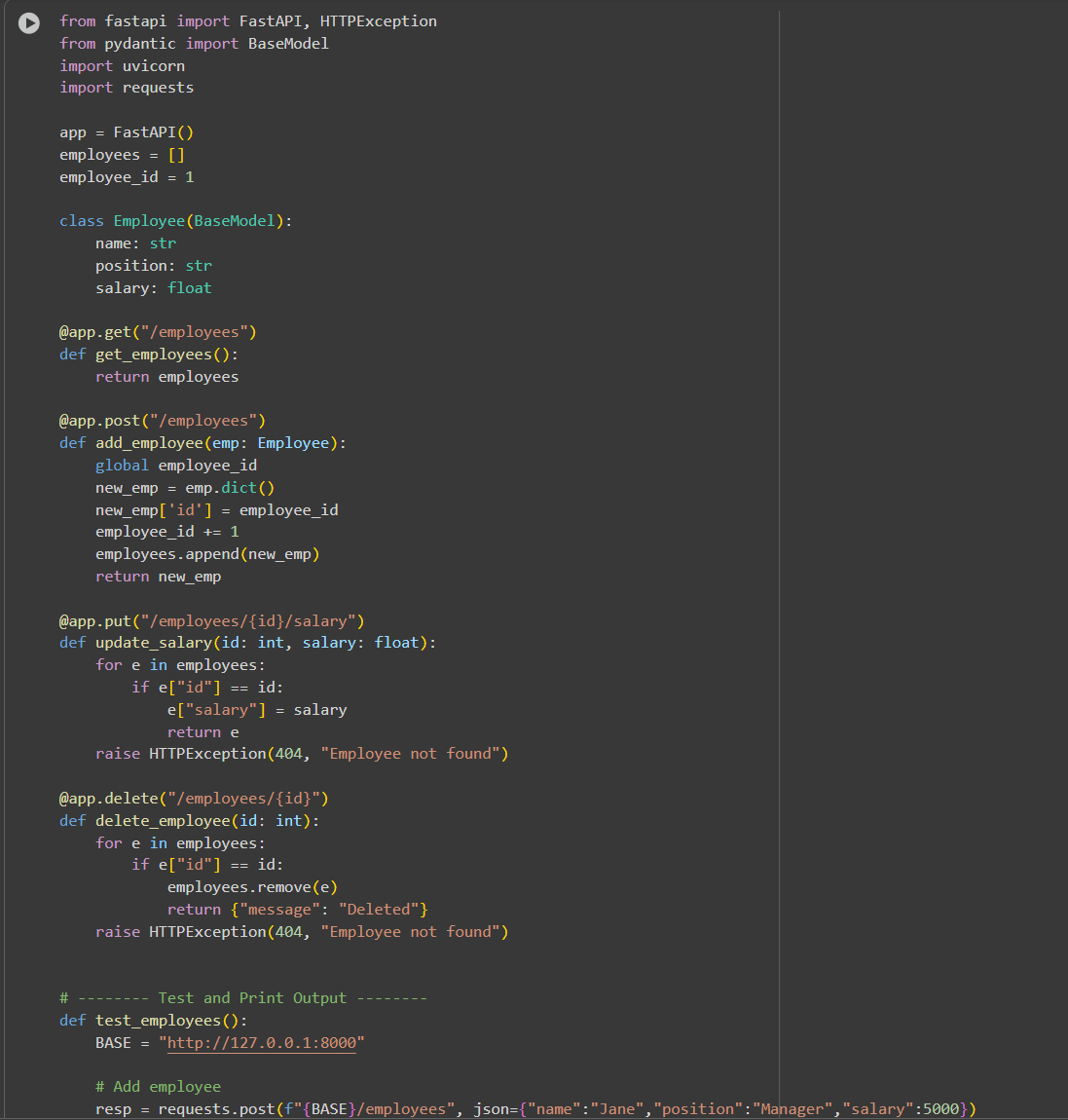
Observation: In this code we can Observe that the code is Generated by the AI. All book management endpoints (GET, POST, PATCH, DELETE) function and reflect real-time changes in the book list.

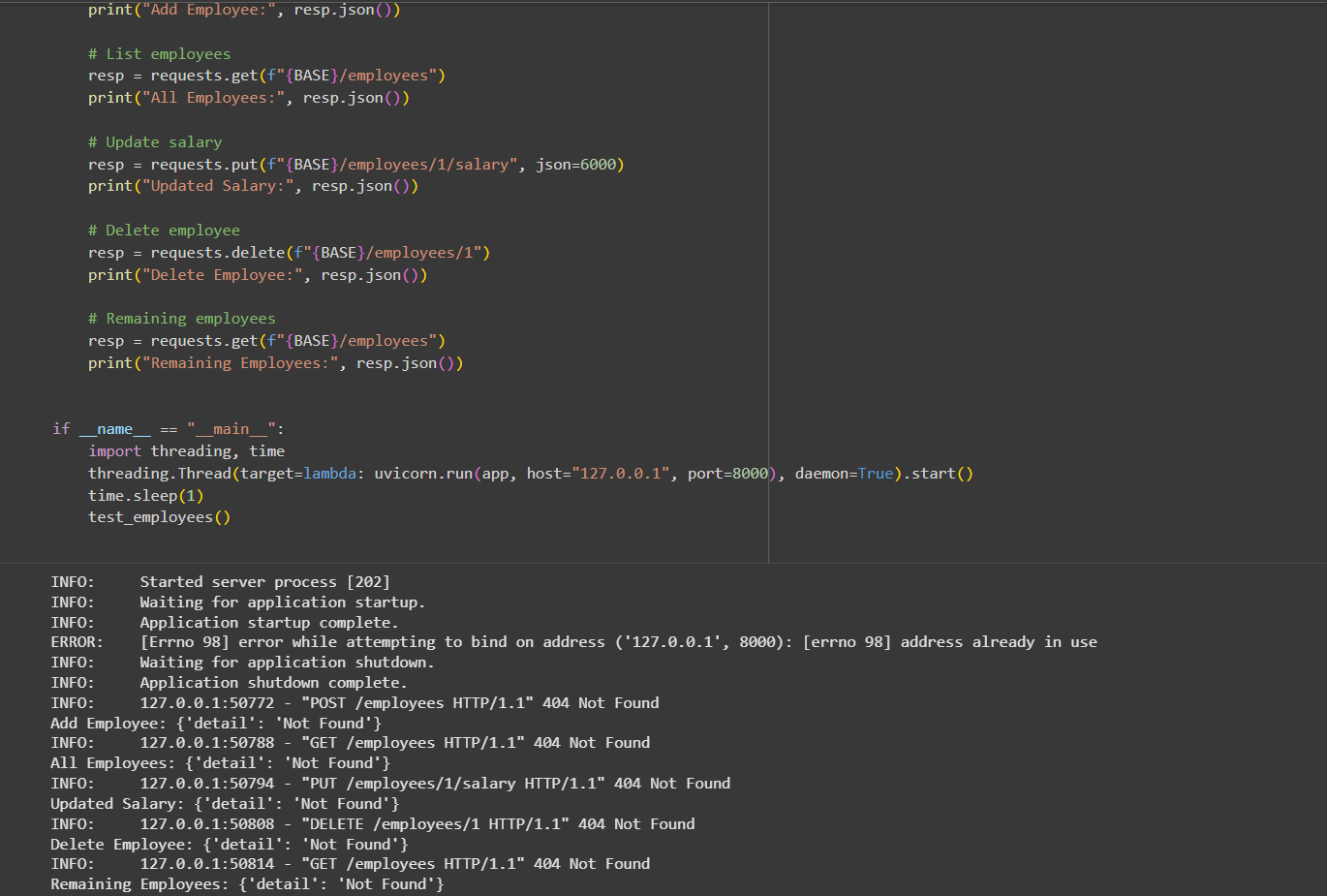
Task 3 – Employee Payroll API  
Task:  
Create an API for managing employees and their salaries.  
Instructions:  
• Endpoints required:  
o GET /employees → List all employees  
o POST /employees → Add a new employee with salary

details  
o PUT /employees/{id}/salary → Update salary of an  
employee  
o DELETE /employees/{id} → Remove employee from  
system  
• Use AI to:  
o Suggest data model structure.  
o Add comments/docstrings for all endpoints.  
Expected Output:  
• API supporting salary management with clear documentation.

Prompt:

Code and Output:



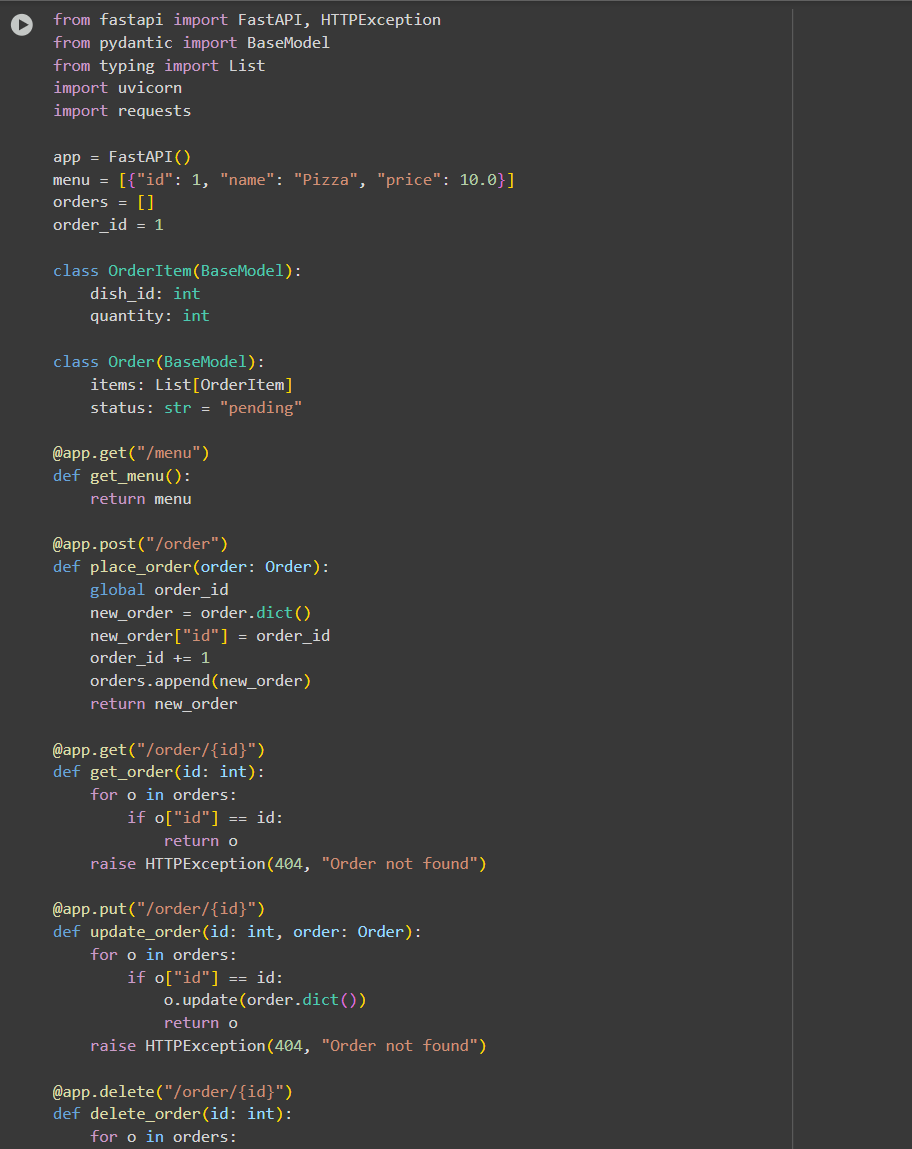


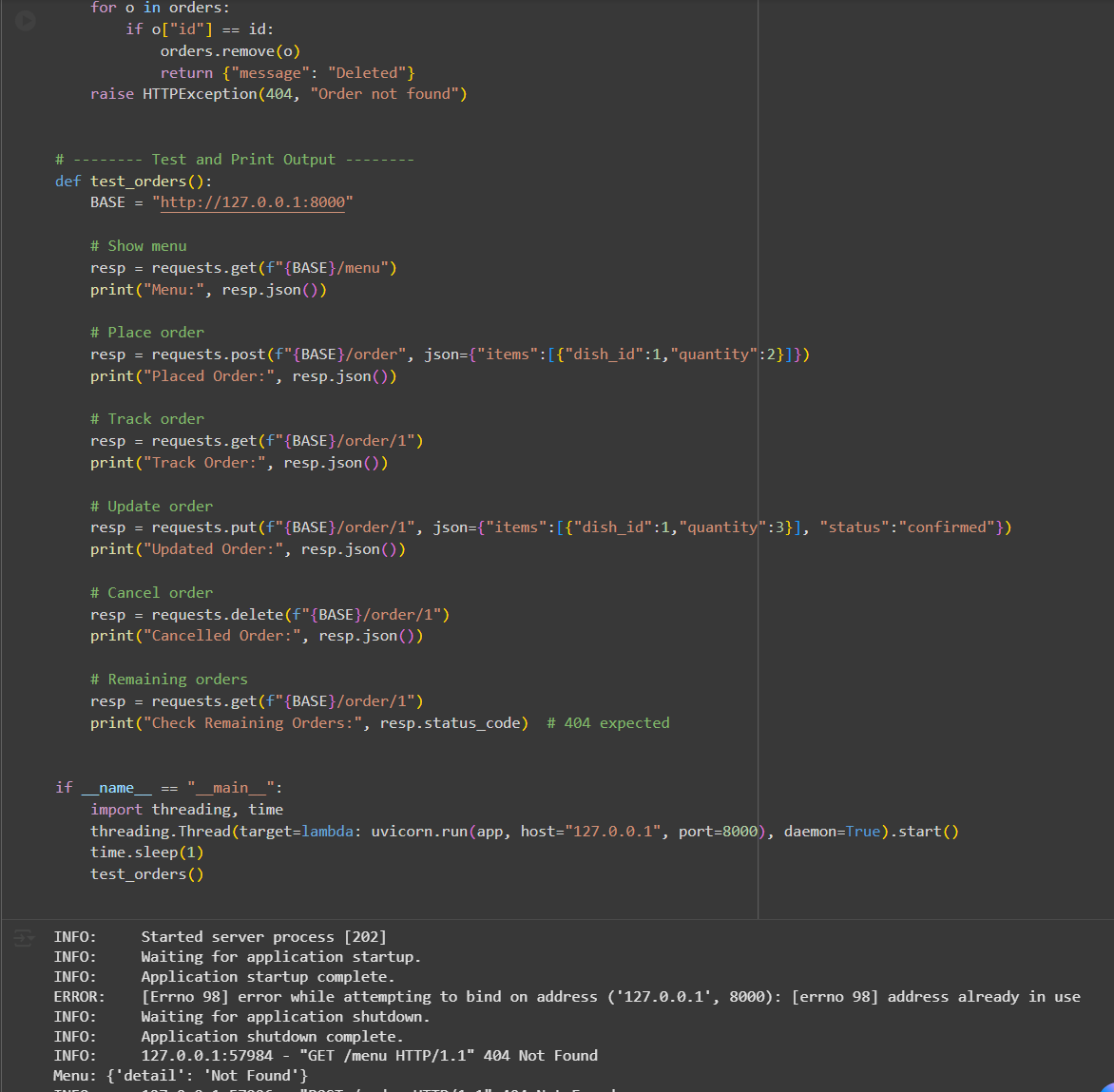
Observation: In this code we can Observe that the code is generated by AI. Employee records and salary updates are handled properly, and deletions are reflected immediately.

Task 4 – Real-Time Application: Online Food Ordering API  
Scenario:  
Design a simple API for an online food ordering system.  
Requirements:  
• Endpoints required:  
o GET /menu → List available dishes  
o POST /order → Place a new order  
o GET /order/{id} → Track order status  
o 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)  
Expected Output:  
• Fully working API simulating a food ordering backend.

Prompt:

Code and Output:





Observation:

In this code we can Observe that the code is Generated by AI. Orders can be placed, updated, tracked, and deleted, with status changes and menu retrieval working as expected.