Assignment-15

Lab 15 – Backend API Development: Creating RESTful Services with AI

NAME: ROHITH INNAMURI

Htno : 2503A52L05

BATCH:16

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

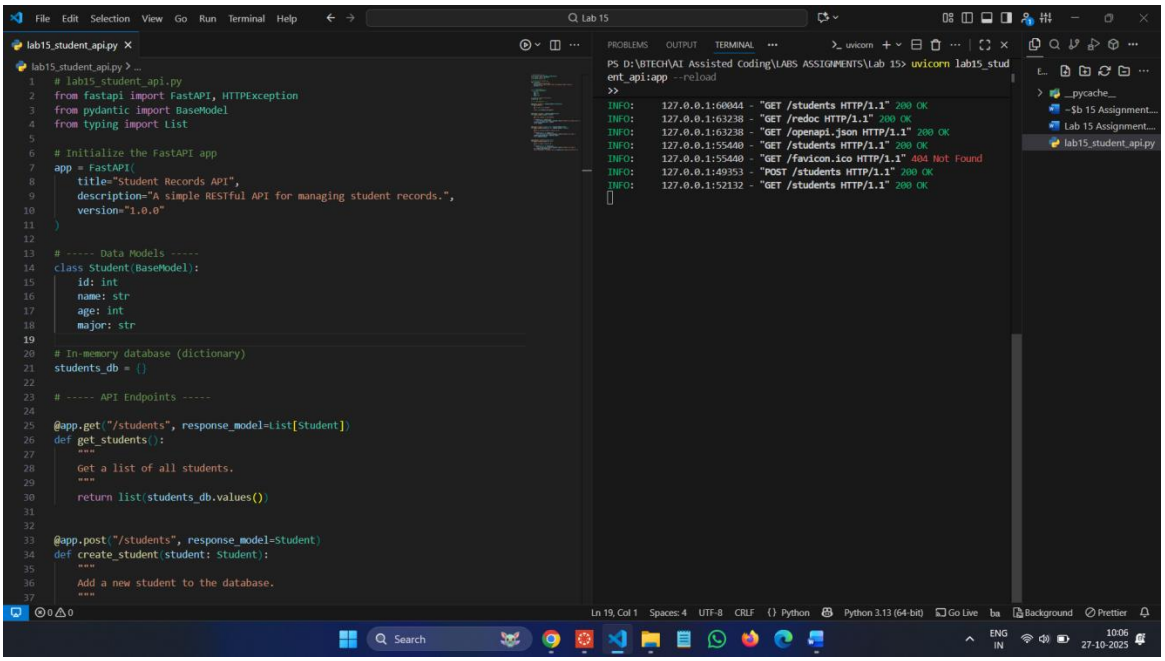
o PUT /students/{id} → Update student details

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.

Code :



Swagger UI: http://127.0.0.1:8000/docs

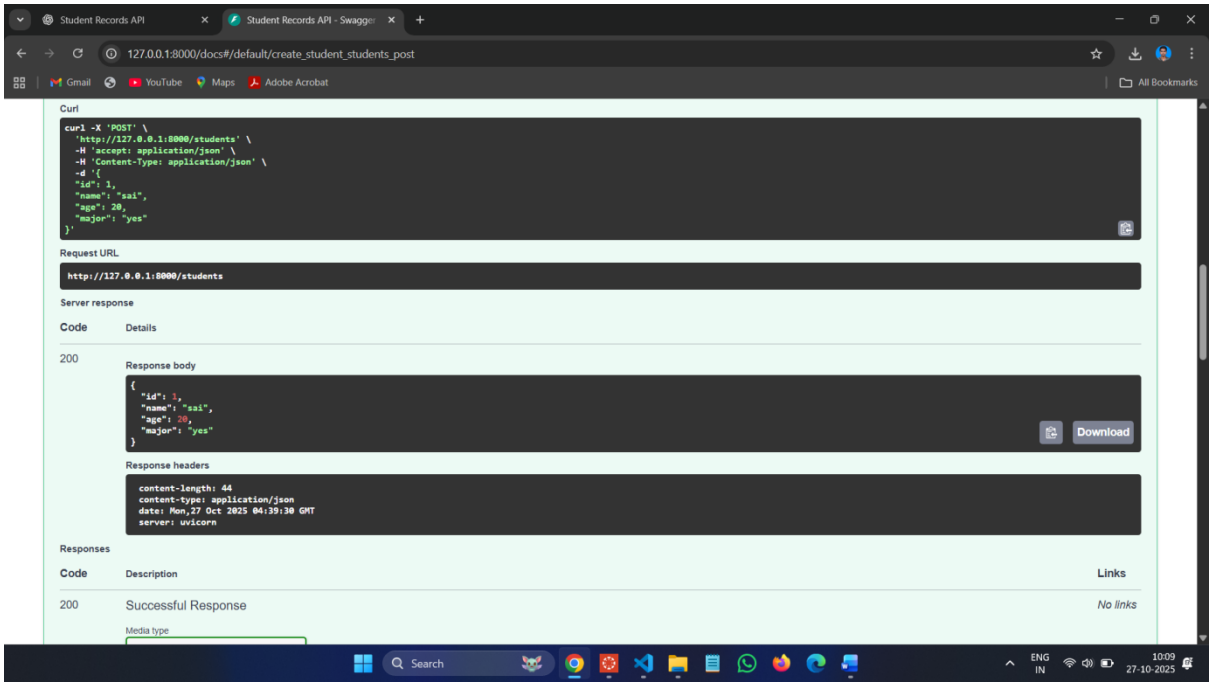
Commands :

pip install fastapiuvicorn

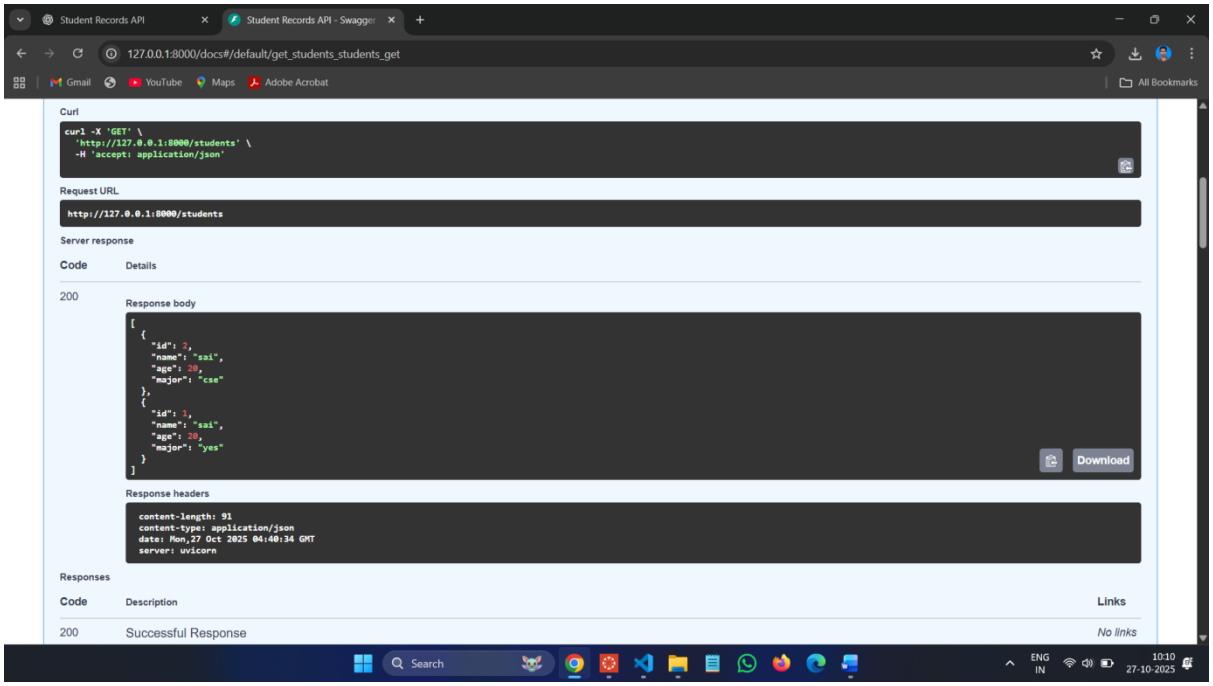
uvicorn lab15_student_api:app –reload

Output:

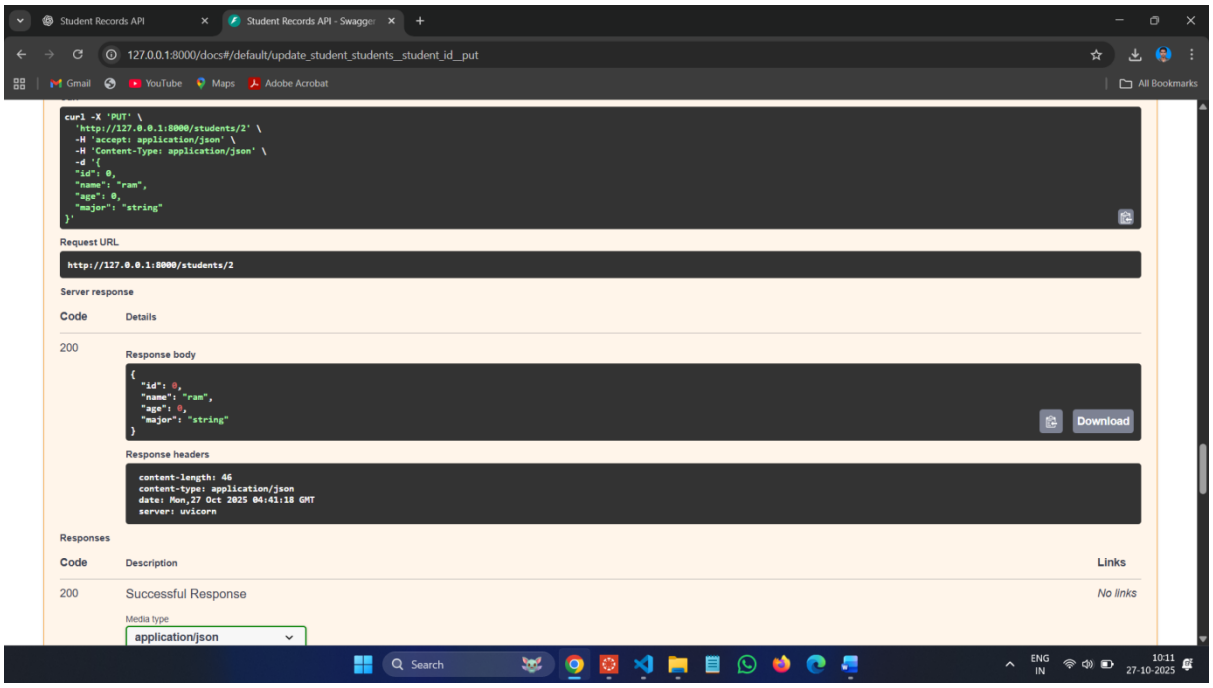
POST :



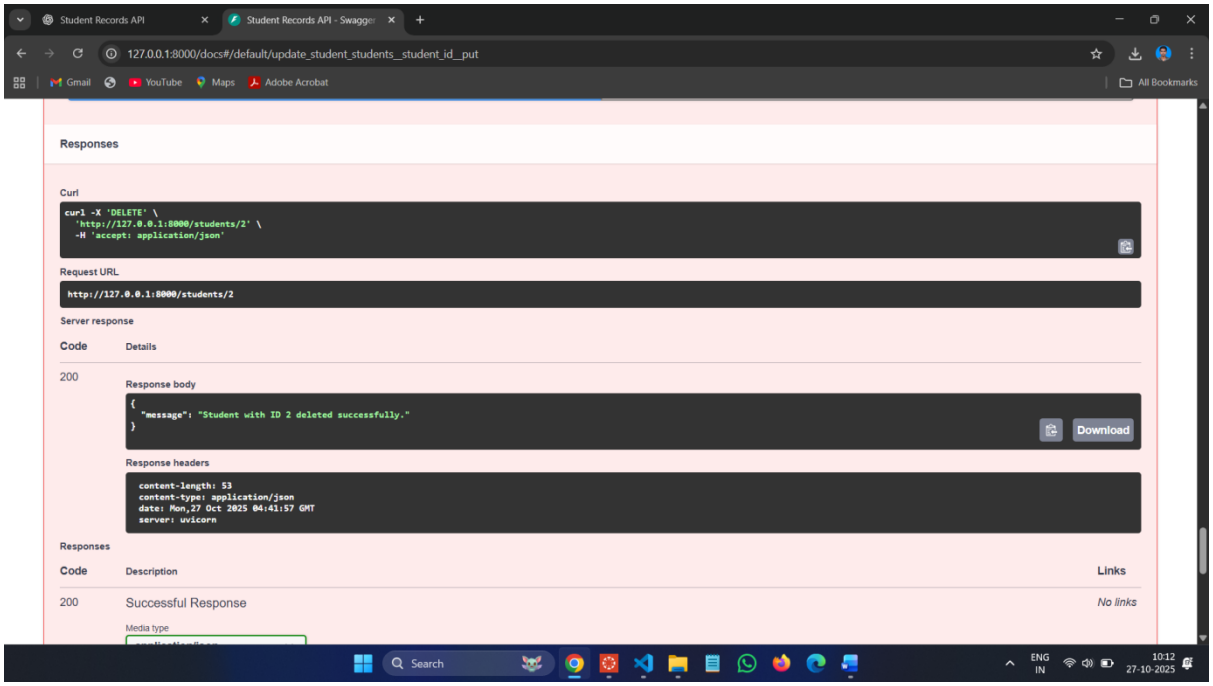
GET :



PUT :



DELETE :



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

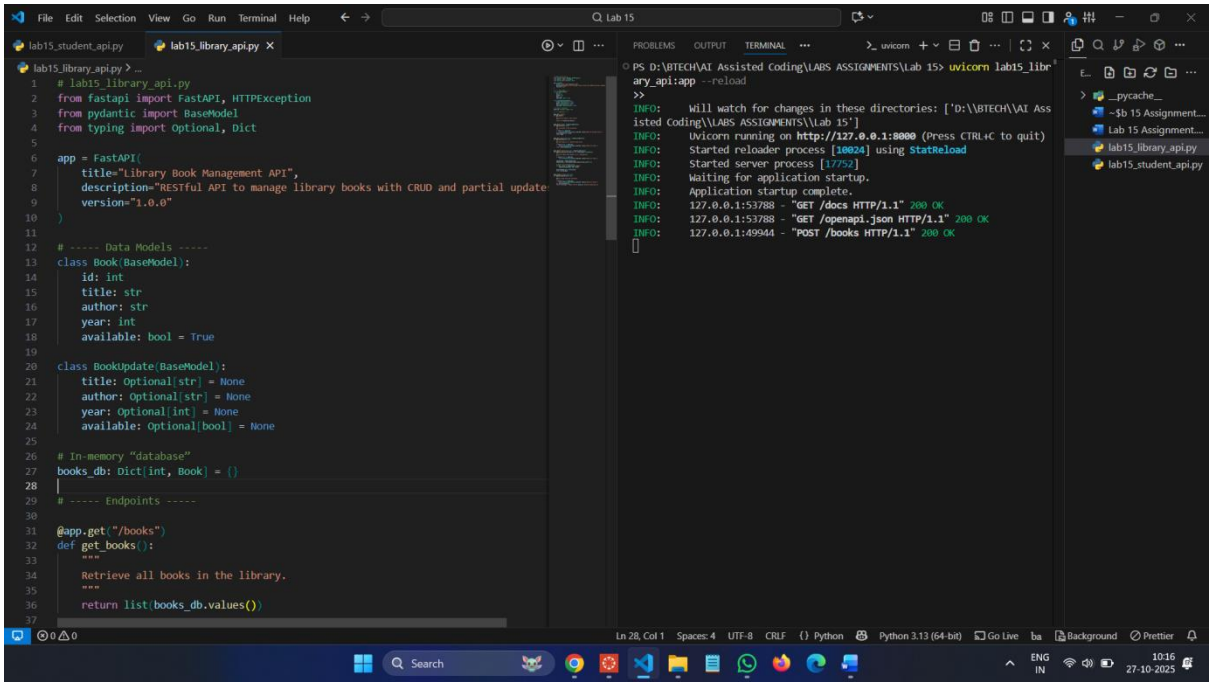
o GET /books/{id} → Get details of a specific book

o PATCH /books/{id} → Update partial book details (e.g., availability)

o DELETE /books/{id} → Remove a book

• Implement error handling for invalid requests.

Code :

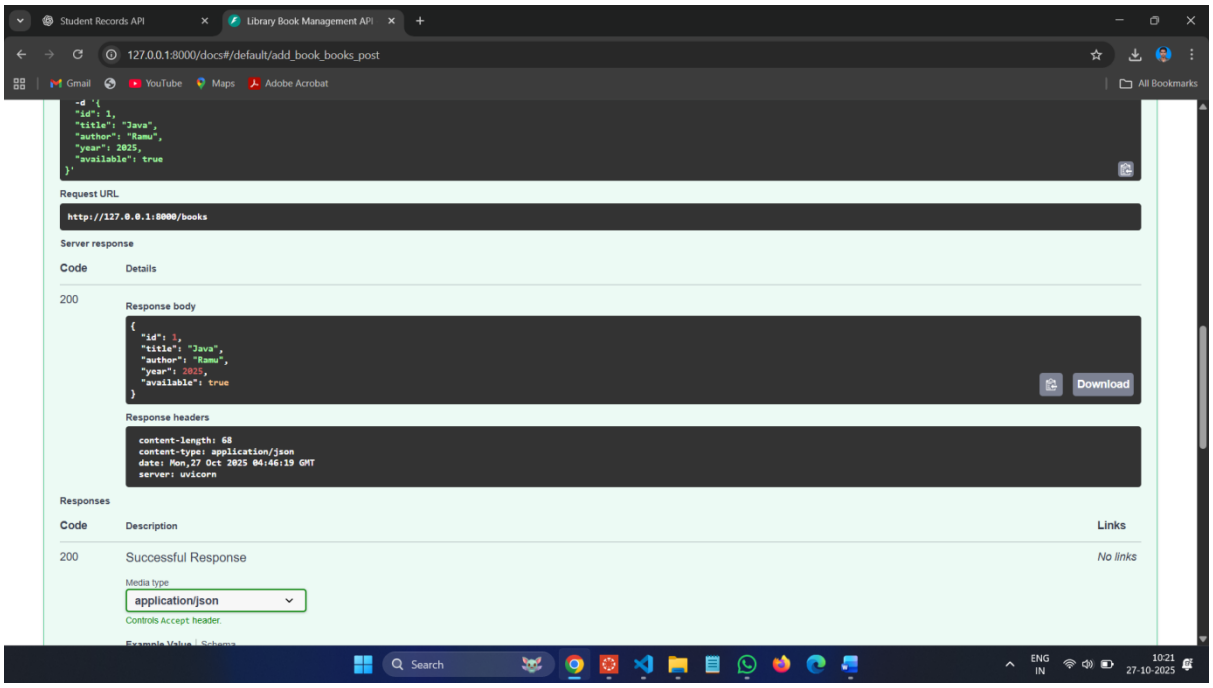


Commands :

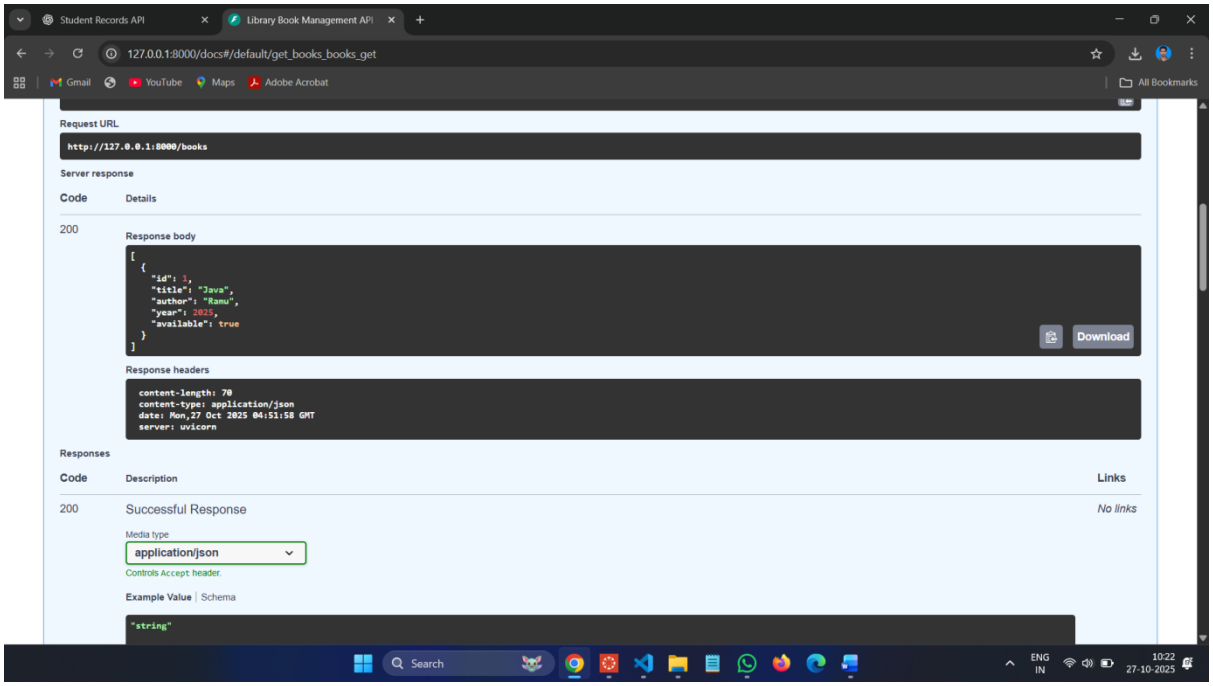
uvicorn lab15_library_api:app –reload

Swagger UI:http://127.0.0.1:8000/docs

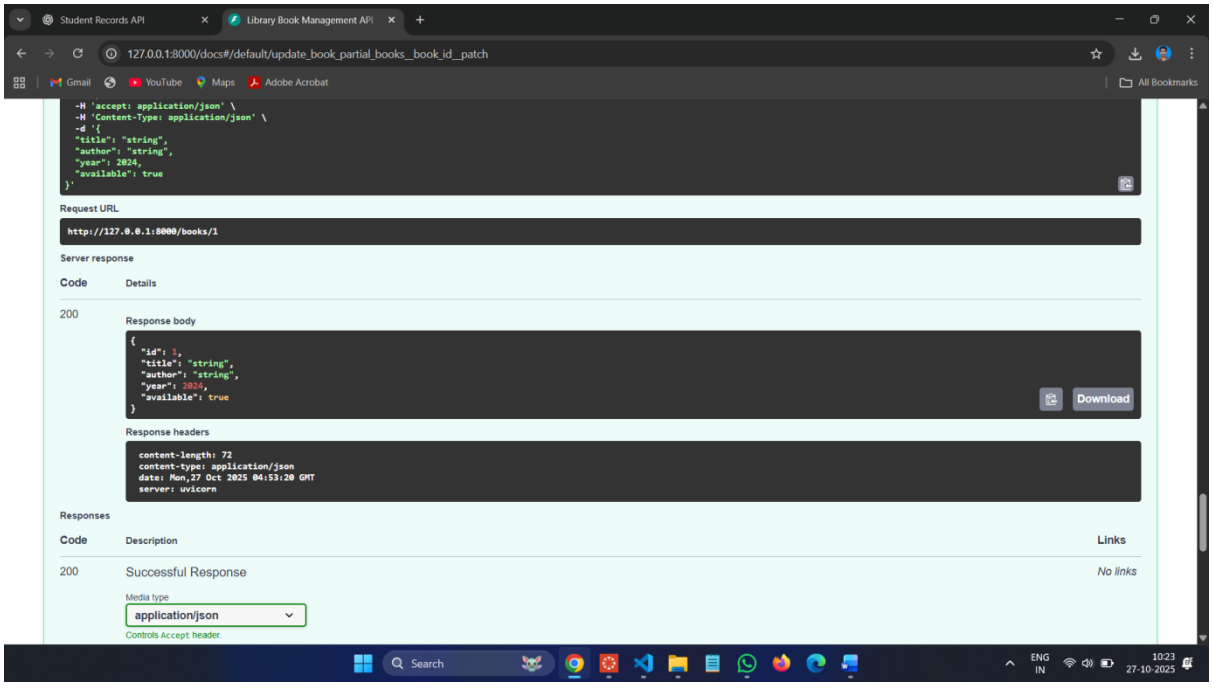
POST :



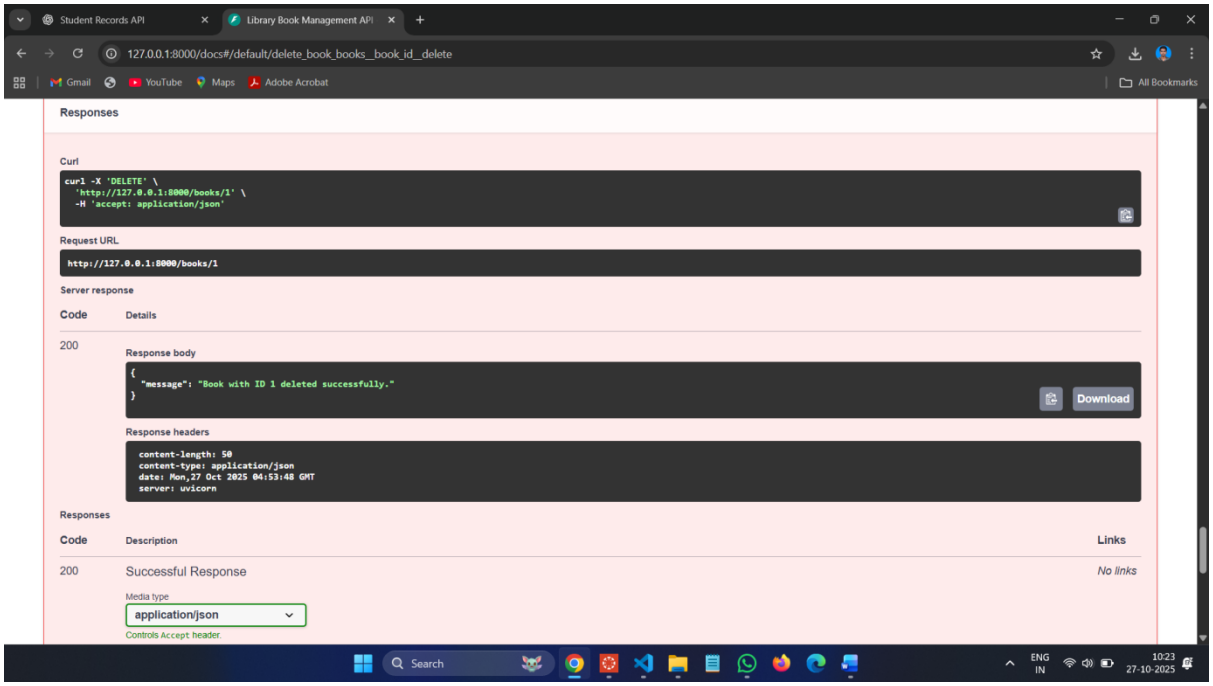
GET :


Student Records API
Library Book Management API
127.0.0.1:8000/docs#/default/get_books_books_get
Request URL
http://127.0.0.1:8000/books
Server response
Code
Details
200
Response body
{
"id": 1,
"title": "Java",
"author": "Ramu",
"year": 2025,
"available": true
}
]
Download
Response headers
content-length: 70
content-type: application/json
date: Mon,27 Oct 2025 04:51:58 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
"string"
Search
ENG
IN
10:22
27-10-2025

PUT :


Student Records API
Library Book Management API
127.0.0.1:8000/docs#/default/update_book_partial_books__book_id__patch
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"title": "string",
"author": "string",
"year": 2024,
"available": true
}'
Request URL
http://127.0.0.1:8000/books/1
Server response
Code
Details
200
Response body
{
"id": 1,
"title": "string",
"author": "string",
"year": 2024,
"available": true
}
Download
Response headers
content-length: 72
content-type: application/json
date: Mon,27 Oct 2025 04:53:20 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Search
ENG
IN
10:23
27-10-2025

DELETE :



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

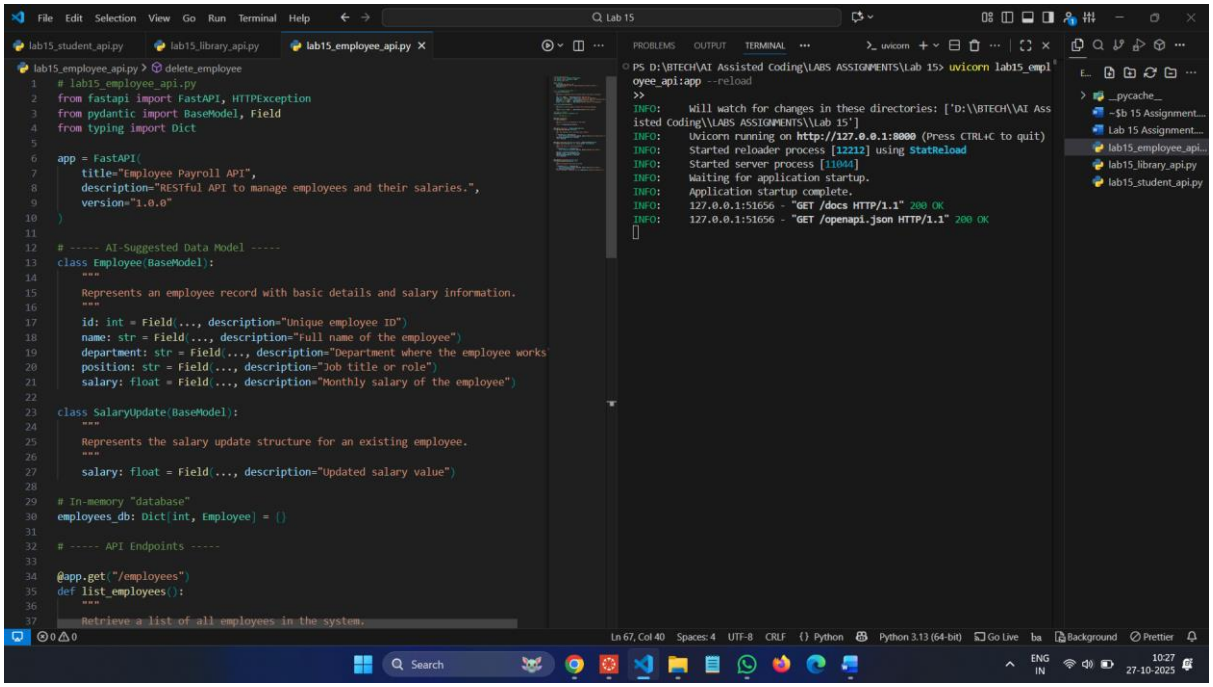
o DELETE /employees/{id} → Remove employee from system

- **Use AI to:**

o Suggest data model structure.

o Add comments/docstrings for all endpoints.

Code :

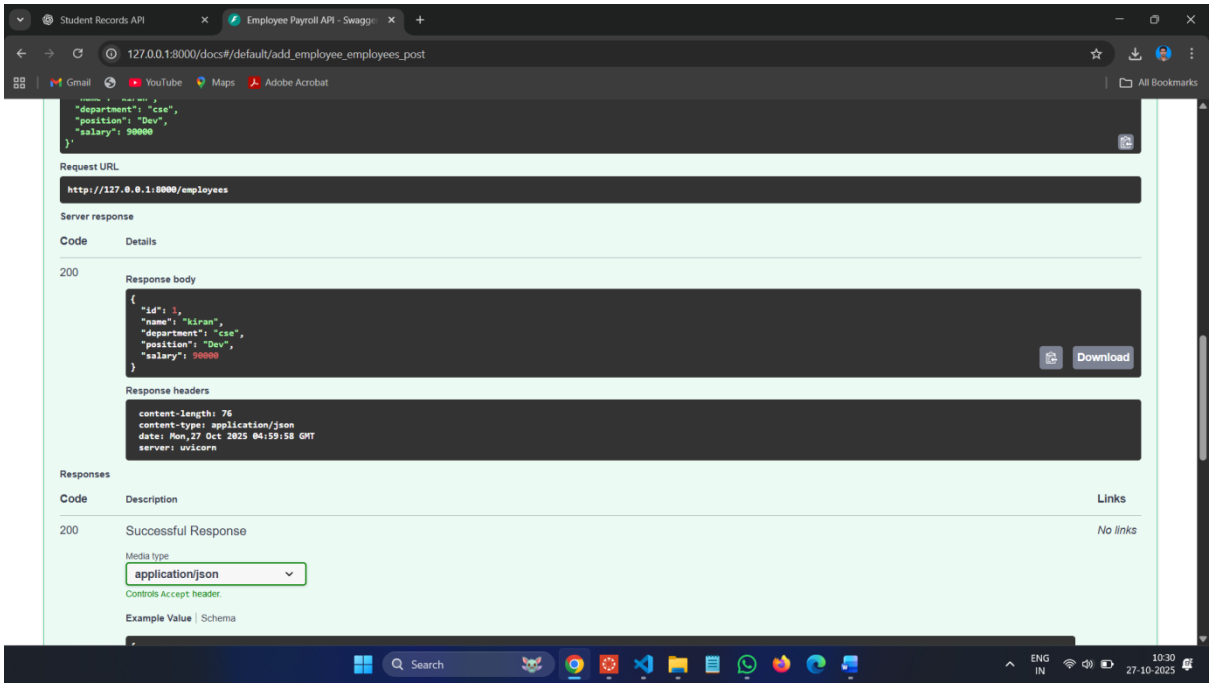


Commands :

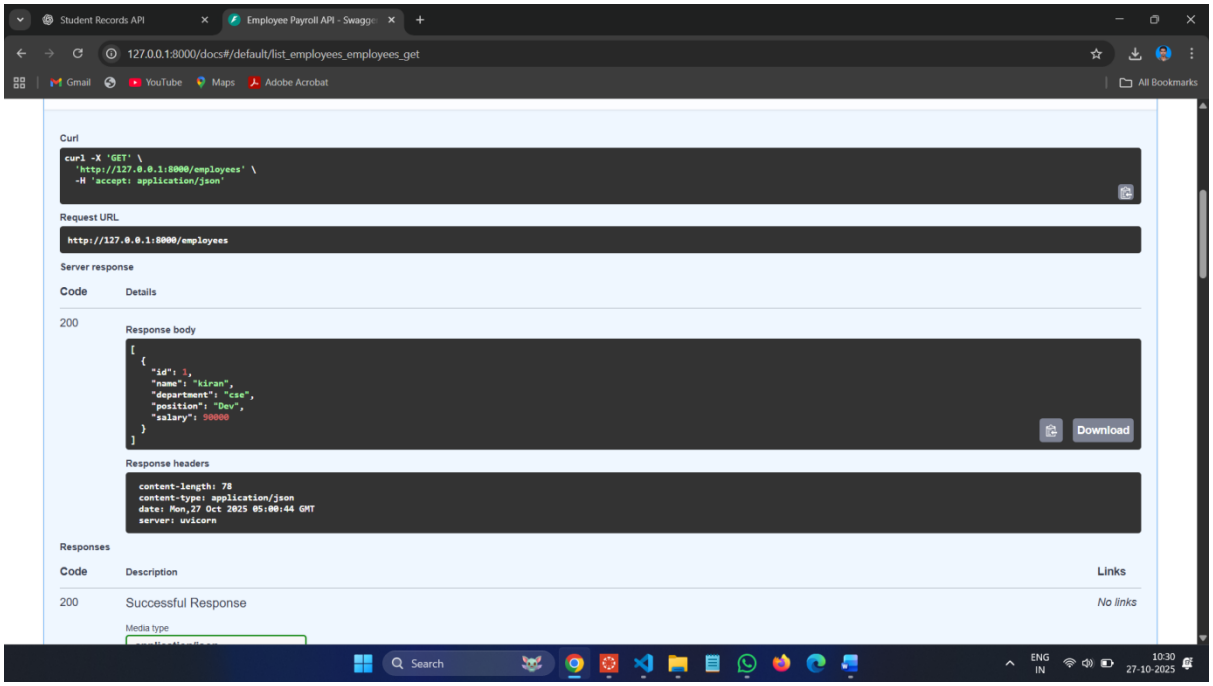
uvicorn lab15_employee_api:app –reload

Swagger UI: http://127.0.0.1:8000/docs

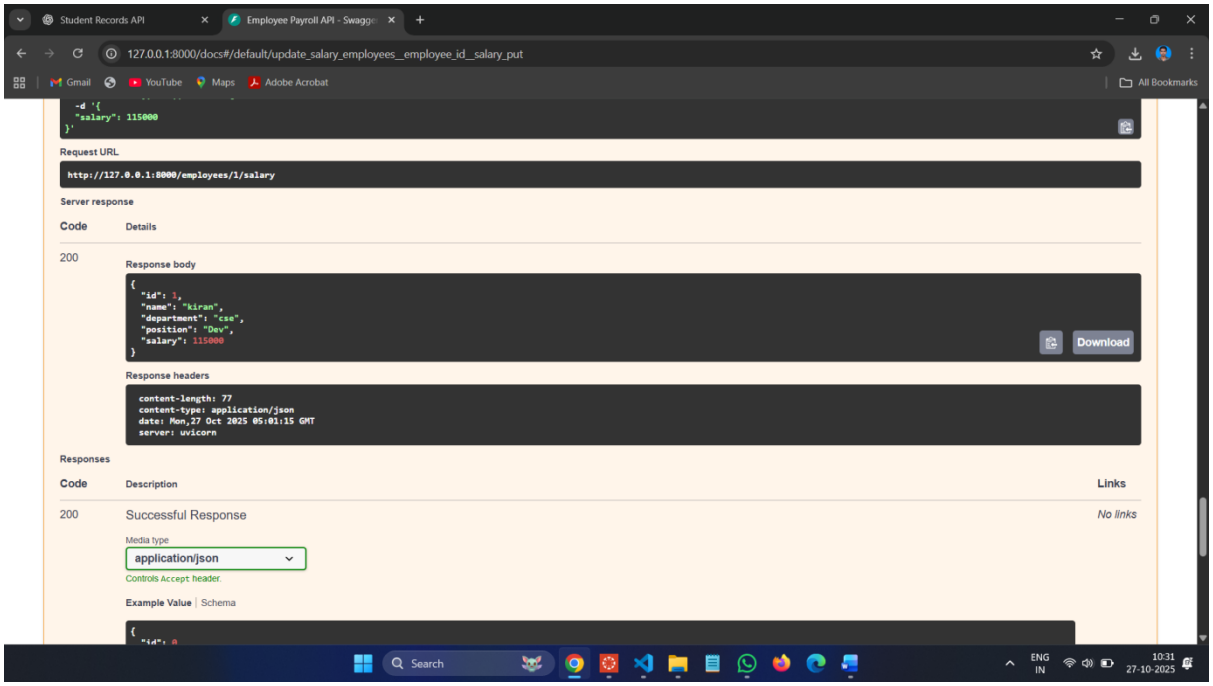
POST :



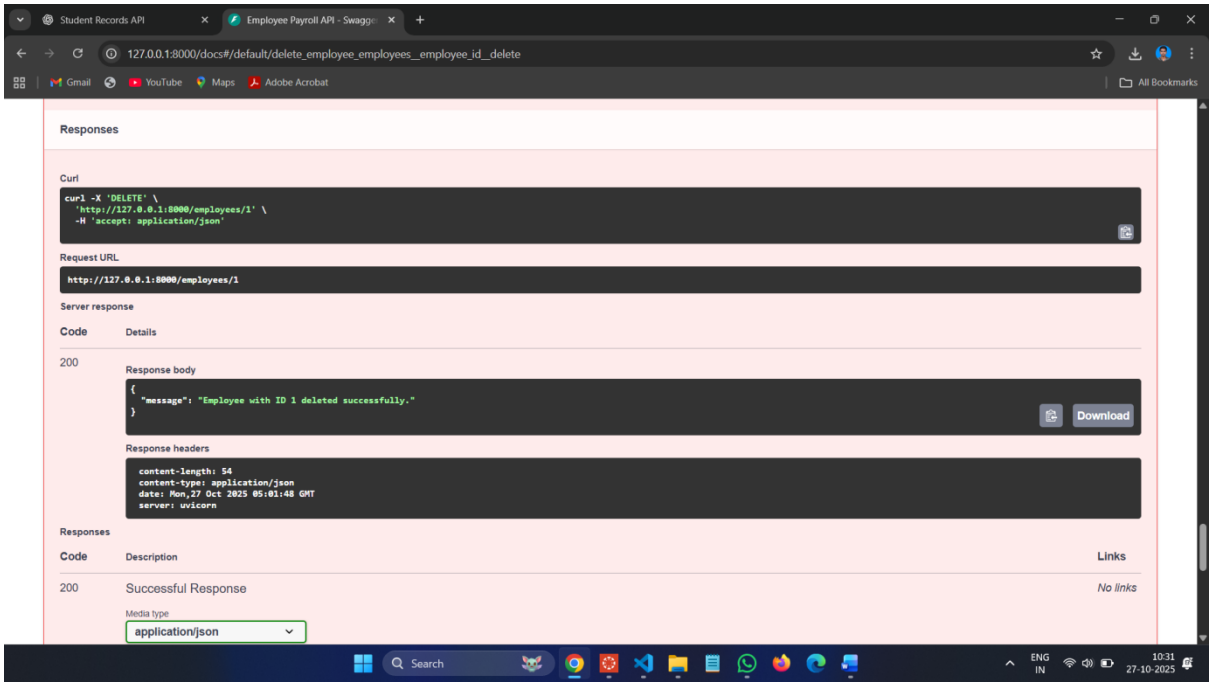
GET :


Student Records API
Employee Payroll API - Swagger
127.0.0.1:8000/docs#/default/list_employees_employees_get
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/employees' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/employees
Server response
Code
Details
200
Response body
[
{
"id": 1,
"name": "kiran",
"department": "cse",
"position": "Dev",
"salary": 90000
}
]
Download
Response headers
content-length: 78
content-type: application/json
date: Mon,27 Oct 2025 05:00:44 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type

PUT :


Student Records API
Employee Payroll API - Swagger
127.0.0.1:8000/docs#/default/update_salary_employees__employee_id__salary_put
-d '{
"salary": 115000
}'
Request URL
http://127.0.0.1:8000/employees/1/salary
Server response
Code
Details
200
Response body
{
"id": 1,
"name": "kiran",
"department": "cse",
"position": "Dev",
"salary": 115000
}
Download
Response headers
content-length: 77
content-type: application/json
date: Mon,27 Oct 2025 05:01:15 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value | Schema

DELETE :



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)

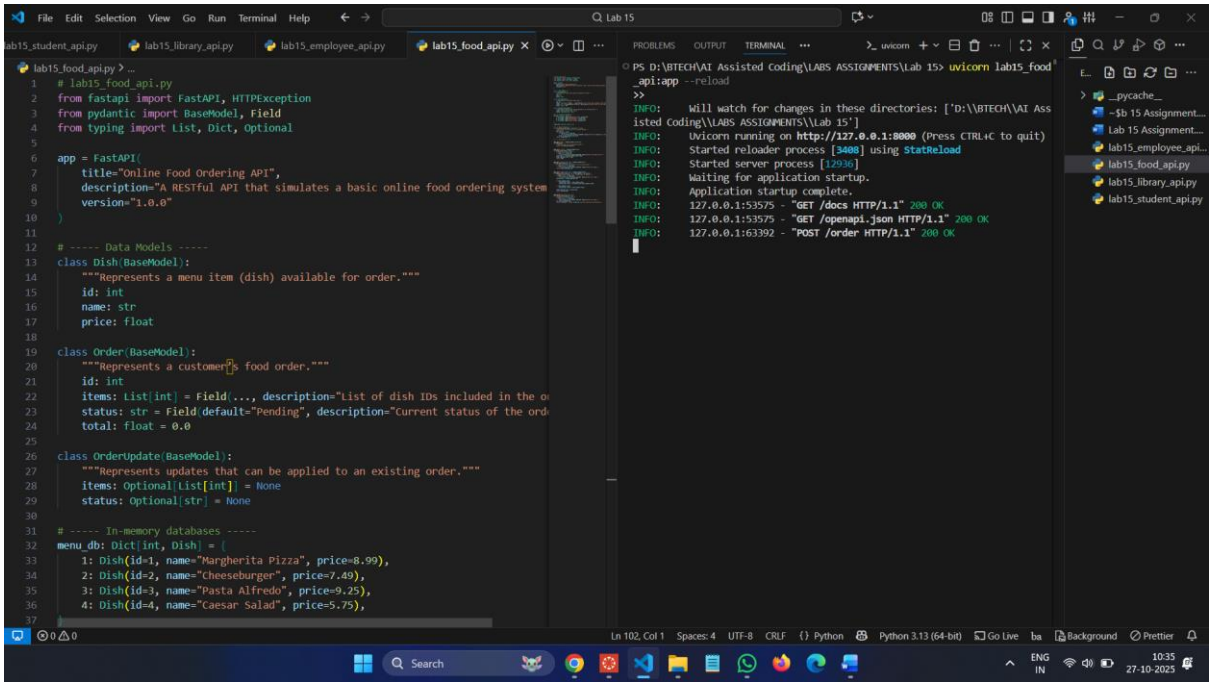
o DELETE /order/{id} → Cancel an order

- **AI should generate:**

o REST API code

o Suggested improvements (like authentication, pagination)

Code :

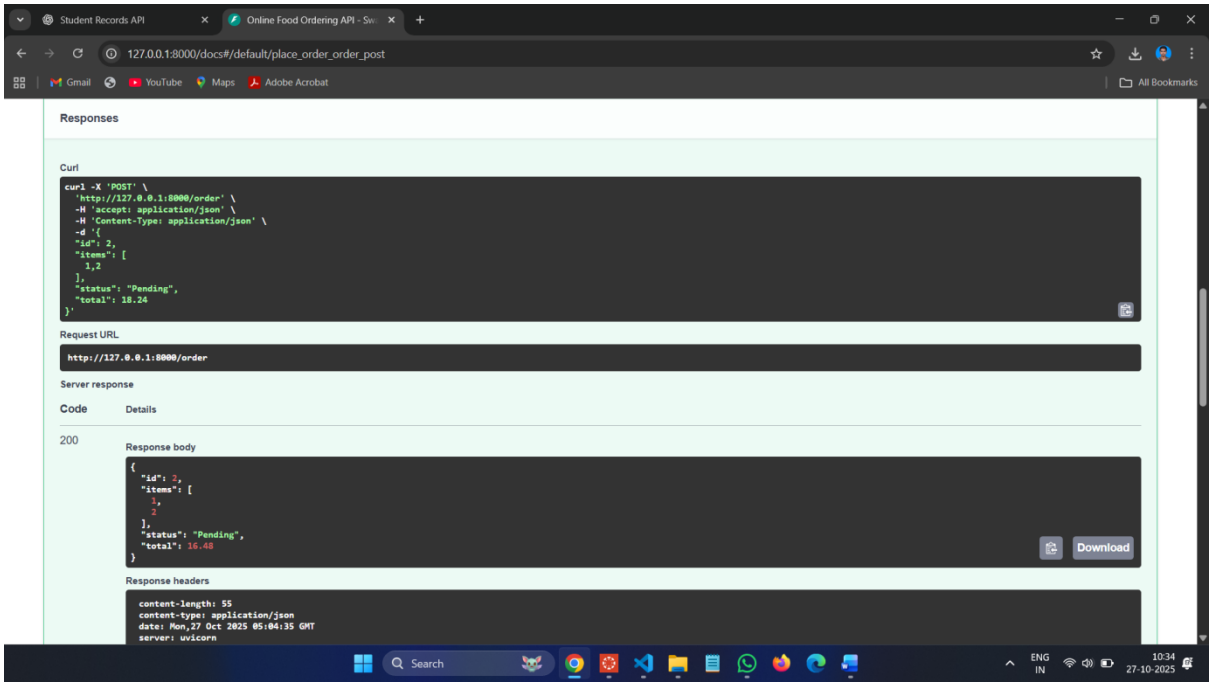


Commands :

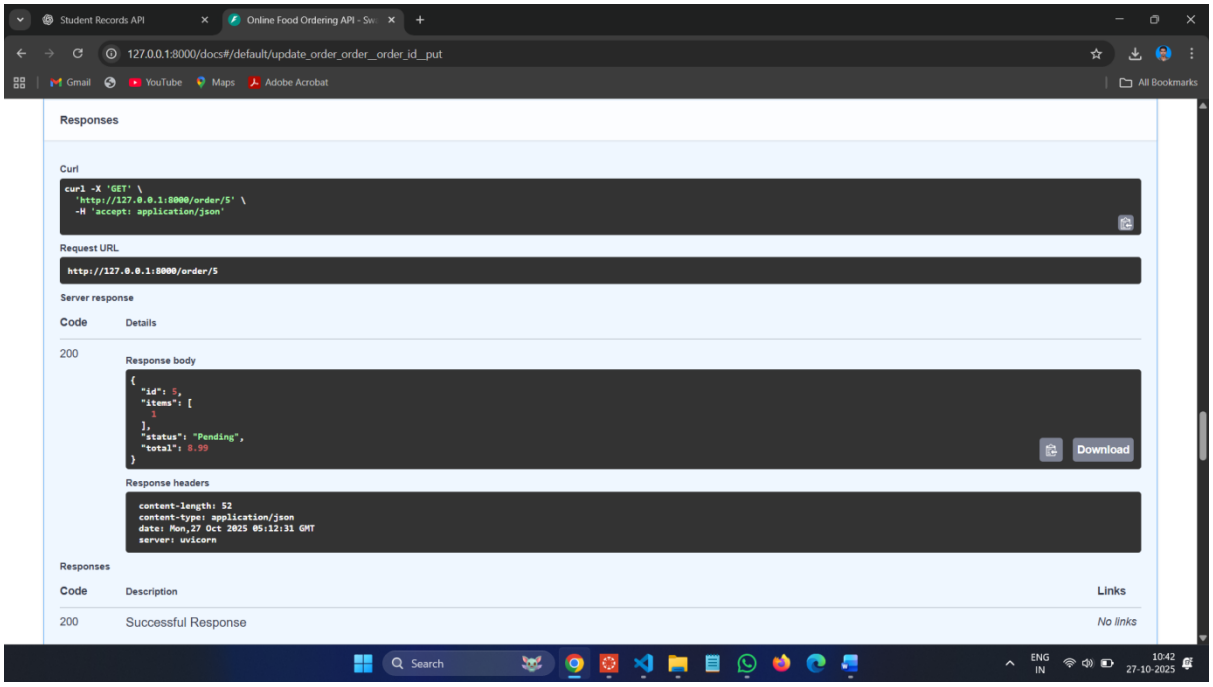
uvicorn lab15_food_api:app --reload

Swagger UI: http://127.0.0.1:8000/docs

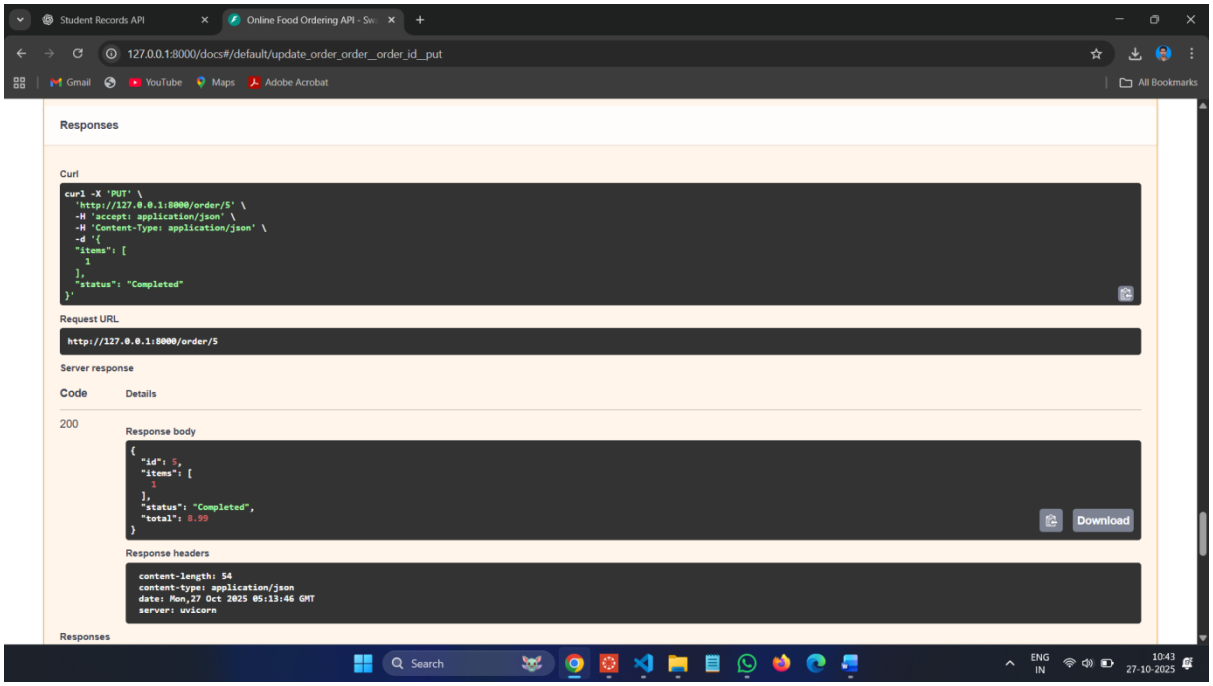
POST :


Student Records API
Online Food Ordering API - Sw
127.0.0.1:8000/docs#/default/place_order_order_post
Gmail
YouTube
Maps
Adobe Acrobat
All Bookmarks
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/order' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"id": 2,
"items": [
1,2
],
"status": "Pending",
"total": 18.24
}'
Request URL
http://127.0.0.1:8000/order
Server response
Code
Details
200
Response body
{
"id": 2,
"items": [
1,
2
],
"status": "Pending",
"total": 16.48
}
Download
Response headers
content-length: 55
content-type: application/json
date: Mon,27 Oct 2025 05:04:35 GMT
server: uvicorn
Search
ENG
IN
10:34
27-10-2025

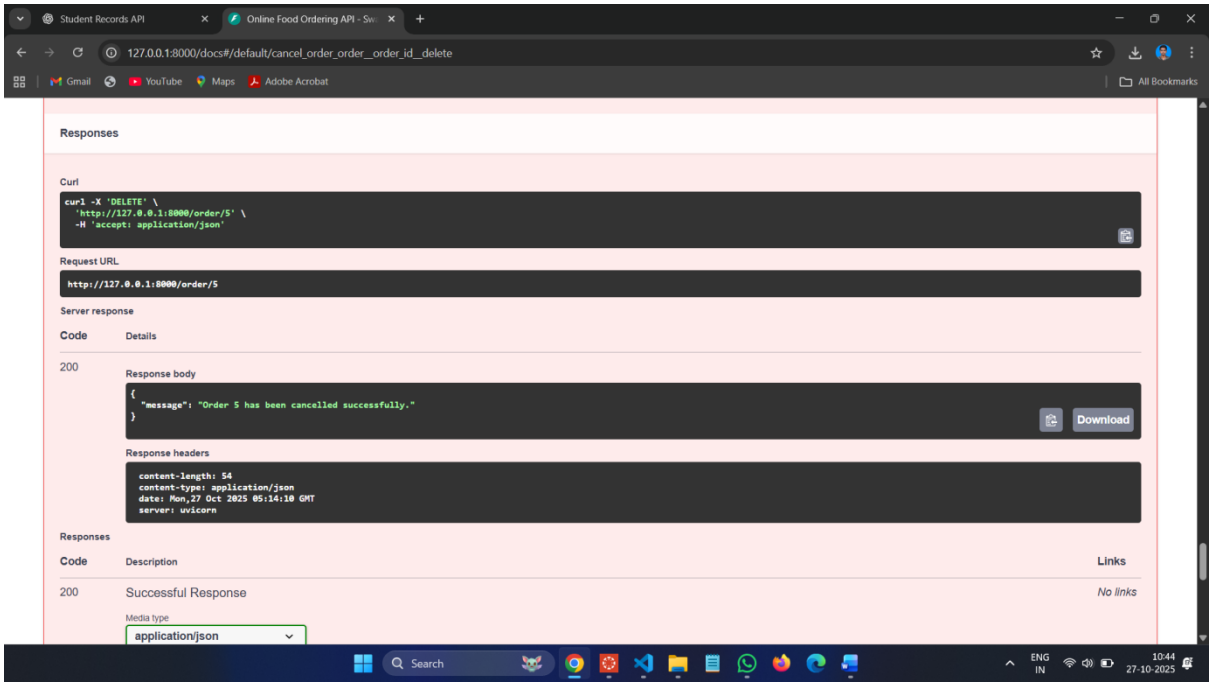
GET :


Student Records API
Online Food Ordering API - Sw
127.0.0.1:8000/docs#/default/update_order_order__order_id__put
Gmail
YouTube
Maps
Adobe Acrobat
All Bookmarks
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/order/5' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/order/5
Server response
Code
Details
200
Response body
{
"id": 5,
"items": [
1
],
"status": "Pending",
"total": 8.99
}
Download
Response headers
content-length: 52
content-type: application/json
date: Mon,27 Oct 2025 05:12:31 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Search
ENG
IN
10:42
27-10-2025

PUT :


Student Records API
Online Food Ordering API - Sw
127.0.0.1:8000/docs#/default/update_order_order__order_id__put
Gmail
YouTube
Maps
Adobe Acrobat
All Bookmarks
Responses
Curl
curl -X 'PUT' \
'http://127.0.0.1:8000/order/5' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"items": [
1
],
"status": "Completed"
}'
Request URL
http://127.0.0.1:8000/order/5
Server response
Code
Details
200
Response body
{
"id": 5,
"items": [
1
],
"status": "Completed",
"total": 8.99
}
Download
Response headers
content-length: 54
content-type: application/json
date: Mon,27 Oct 2025 05:13:46 GMT
server: uvicorn
Responses
Search
ENG
IN
10:43
27-10-2025

DELETE :


Student Records API
Online Food Ordering API - Sw
127.0.0.1:8000/docs#/default/cancel_order_order__order_id__delete
Gmail
YouTube
Maps
Adobe Acrobat
All Bookmarks
Responses
Curl
curl -X 'DELETE' \
'http://127.0.0.1:8000/order/5' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/order/5
Server response
Code
Details
200
Response body
{
"message": "Order 5 has been cancelled successfully."
}
Download
Response headers
content-length: 54
content-type: application/json
date: Mon,27 Oct 2025 05:14:10 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Search
ENG
IN
10:44
27-10-2025

Observations:

1. **RESTful Design Principles:**
   Each task followed REST standards — using appropriate HTTP methods (GET, POST, PUT, PATCH, DELETE) and clear URL structures for resources like /students, /books, /employees, and /order.
2. **AI-Assisted Code Generation:**
   AI tools were effectively used to generate data models, endpoint structures, and documentation automatically, saving development time and ensuring consistency.
3. **CRUD Operations:**
   All four APIs successfully implemented CRUD functionality:
   - Create: POST requests
   - Read: GET requests
   - Update: PUT/PATCH requests
   - Delete: DELETE requests
4. **Error Handling:**
   Each API handled invalid requests gracefully using HTTPException with proper status codes and messages.
5. **Auto Documentation:**
   FastAPI automatically provided API documentation through /docs (Swagger UI) and /redoc (ReDoc), fulfilling the lab's documentation objective.
6. **Partial Updates & Real-Time Design:**
   - Task 2 introduced partial updates (PATCH) for library books.
   - Task 4 simulated a real-world food ordering system with realistic order tracking and AI improvement suggestions.

7. **AI-Driven Improvements:**
   Suggested features such as authentication, pagination, real-time updates (via WebSockets), and database integration demonstrate how AI can enhance backend architecture planning.