

Assignment-15

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

NAME: BHUVANESHWAR REDDY

Htno : 2403A52416

BATCH:

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 :**

The screenshot shows a code editor with a Python file named `lab15_student_api.py`. The code defines a FastAPI application for managing student records. It includes a simple database represented as a dictionary of `Student` objects. The API endpoints include `/students` for listing, creating, updating, and deleting students. The terminal window shows the application running on port 8000, with log messages indicating successful requests for each endpoint. The system tray at the bottom right shows the date and time as 27-10-2025.

```
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 15> uvicorn lab15_student_api:app --reload
INFO: 127.0.0.1:60044 - "GET /students HTTP/1.1" 200 OK
INFO: 127.0.0.1:63238 - "GET /redoc HTTP/1.1" 200 OK
INFO: 127.0.0.1:63238 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:55408 - "GET /students HTTP/1.1" 200 OK
INFO: 127.0.0.1:55408 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:49553 - "POST /students HTTP/1.1" 200 OK
INFO: 127.0.0.1:52132 - "GET /students HTTP/1.1" 200 OK
```

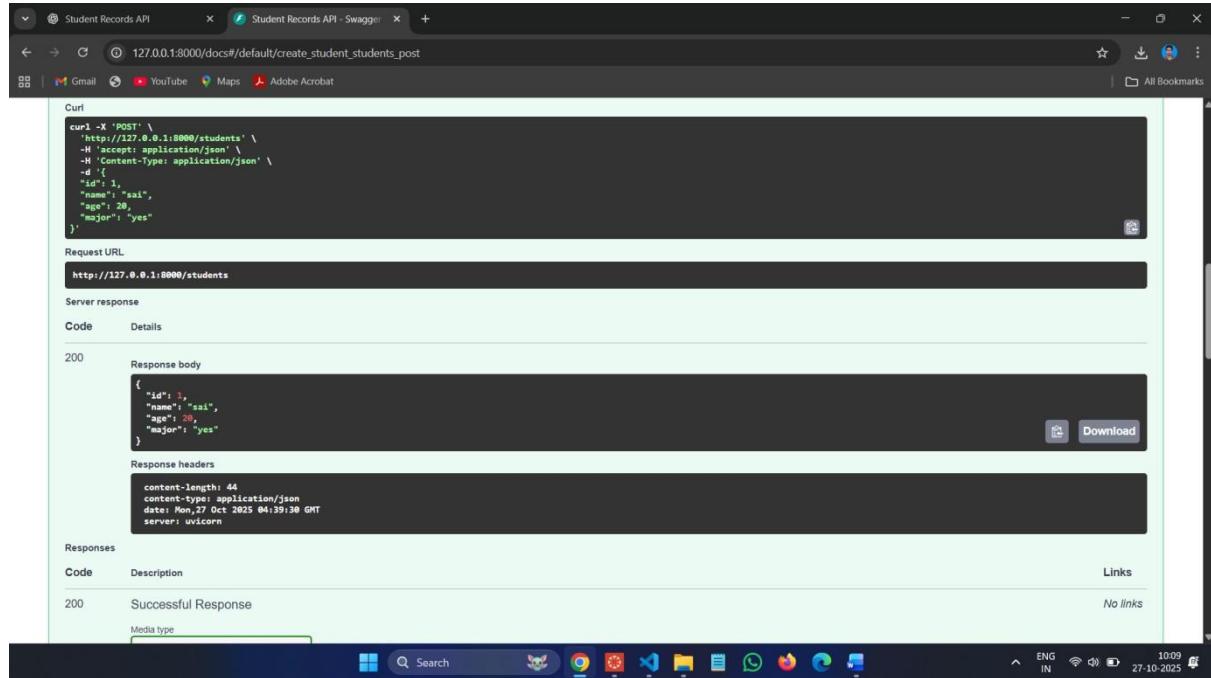
Swagger UI: <http://127.0.0.1:8000/docs>

Commands :

pip install fastapiuvicorn

uvicorn lab15_student_api:app –reload **Output:**

POST :



Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/students' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "name": "sai",
    "age": 20,
    "major": "yes"
}'
```

Request URL

`http://127.0.0.1:8000/students`

Server response

Code Details

200 Response body

```
{ "id": 1, "name": "sai", "age": 20, "major": "yes" }
```

Download

Response headers

```
content-length: 44
content-type: application/json
date: Mon, 27 Oct 2025 04:39:30 GMT
server: uvicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type

GET :

GET :

The screenshot shows a browser window with the title "Student Records API - Swagger". The URL in the address bar is "127.0.0.1:8000/docs#/default/get_students_students_get". The page displays a curl command to fetch student data, the request URL (http://127.0.0.1:8000/students), and the server response. The response body contains two student records with IDs 2 and 3. The response headers include content-length, content-type, date, and server. The status code is 200, and the description is "Successful Response".

```
curl -X 'GET' \
'http://127.0.0.1:8000/students' \
-H 'accept: application/json'
```

Request URL
http://127.0.0.1:8000/students

Server response

Code	Details						
200	<p>Response body</p> <pre>[{ "id": 2, "name": "sai", "age": 28, "major": "cse" }, { "id": 3, "name": "sai", "age": 28, "major": "yes" }]</pre> <p>Response headers</p> <pre>content-length: 31 content-type: application/json date: Mon, 27 Oct 2025 04:40:34 GMT server: unicorn</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Successful Response</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	Successful Response	No links
Code	Description	Links					
200	Successful Response	No links					

10:10 27-10-2025 ENG IN

PUT :

The screenshot shows a browser window with the title "Student Records API - Swagger". The URL in the address bar is "127.0.0.1:8000/docs#/default/update_student_students_student_id_put". The page displays a curl command to update a student record with ID 2, the request URL (http://127.0.0.1:8000/students/2), and the server response. The response body shows the updated student record with ID 2, name ram, age 0, and major string. The response headers include content-length, content-type, date, and server. The status code is 200, and the description is "Successful Response". The media type is listed as application/json.

```
curl -X 'PUT' \
'http://127.0.0.1:8000/students/2' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{  
  "id": 2,  
  "name": "ram",  
  "age": 0,  
  "major": "string"  
}'
```

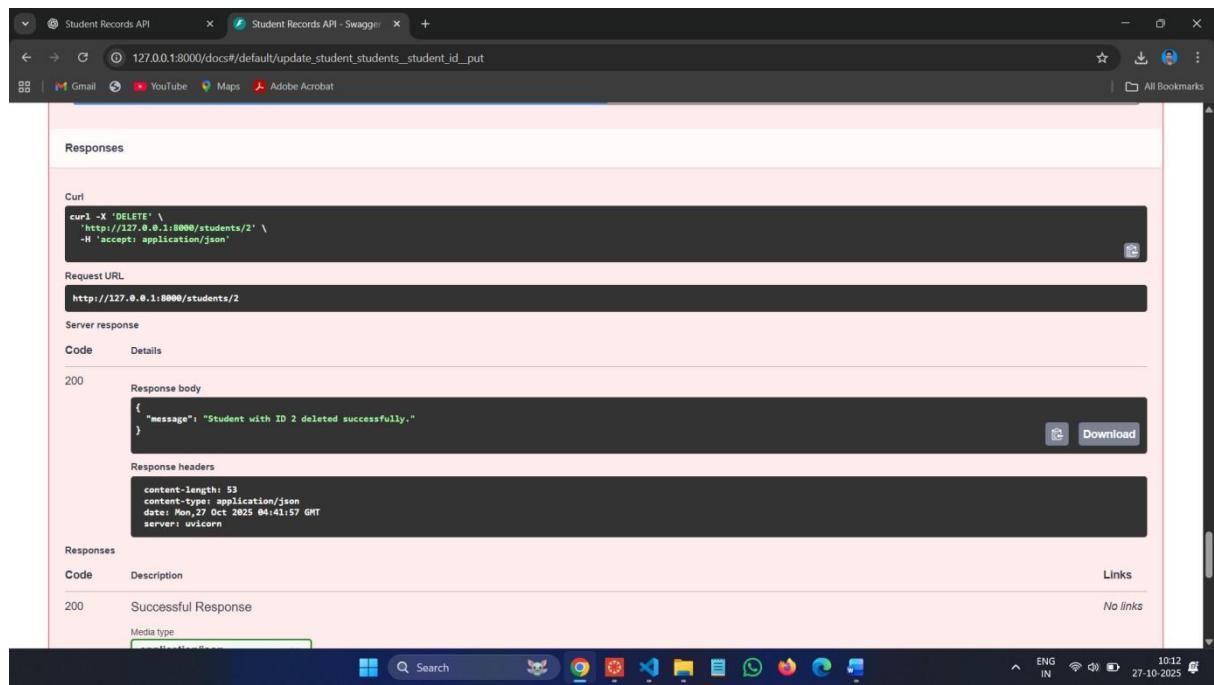
Request URL
http://127.0.0.1:8000/students/2

Server response

Code	Details						
200	<p>Response body</p> <pre>{ "id": 2, "name": "ram", "age": 0, "major": "string" }</pre> <p>Response headers</p> <pre>content-length: 46 content-type: application/json date: Mon, 27 Oct 2025 04:41:18 GMT server: unicorn</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Successful Response</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	Successful Response	No links
Code	Description	Links					
200	Successful Response	No links					

10:11 27-10-2025 ENG IN

DELETE :



Task 2 – Library Book Management API Task:

Develop a RESTful API to handle library books.

Instructions:

- Endpoints required:
 - GET /books → Retrieve all books
 - POST /books → Add a new book
 - GET /books/{id} → Get details of a specific book
 - PATCH /books/{id} → Update partial book details (e.g., availability)
 - DELETE /books/{id} → Remove a book
- Implement error handling for invalid requests.

Code :

```

lab15_library_api.py
1 # lab15_library_api.py
2 from fastapi import FastAPI, HTTPException
3 from pydantic import BaseModel
4 from typing import Optional, Dict
5
6 app = FastAPI(
7     title="Library Book Management API",
8     description="RESTful API to manage library books with CRUD and partial update",
9     version="1.0.0"
10 )
11
12 # ----- Data Models -----
13 class Book(BaseModel):
14     id: int
15     title: str
16     author: str
17     year: int
18     available: bool = True
19
20 class BookUpdate(BaseModel):
21     title: Optional[str] = None
22     author: Optional[str] = None
23     year: Optional[int] = None
24     available: Optional[bool] = None
25
26 # In-memory "database"
27 books_db: Dict[int, Book] = {}
28
29 # ----- Endpoints -----
30
31 @app.get("/books")
32 def get_books():
33     """
34     Retrieve all books in the library.
35     """
36     return list(books_db.values())

```

PS D:\BTECH\VAI Assisted Coding\LABS ASSIGNMENTS\Lab 15> uvicorn lab15_library_api:app --reload

>>> INFO: Will watch for changes in these directories: ['D:\\BTECH\\VAI Assisted Coding\\LABS ASSIGNMENTS\\Lab 15']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [10004] using StatReload
INFO: Started server process [17792]
INFO: Application startup complete.
INFO: Waiting for application startup.
INFO: 127.0.0.1:53788 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:53788 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:49944 - "POST /books HTTP/1.1" 200 OK

Commands :

uvicorn lab15_library_api:app --reload

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

POST :

-d '{
" id": 1,
" title": "Java",
" author": "Ramu",
" year": 2025,
" available": true
}'

Request URL:
http://127.0.0.1:8000/books

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 1, "title": "Java", "author": "Ramu", "year": 2025, "available": true }</pre> <p>Download</p> <p>Response headers</p> <pre>content-length: 68 content-type: application/json date: Mon, 27 Oct 2025 04:46:19 GMT server: uvicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header

Example Value | Schema

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: unicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type application/json

Controls Accept header

Example Value Schema

```
"string"
```

PUT :

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 04:53:28 GMT
server: unicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type application/json

Controls Accept header

DELETE :

The screenshot shows a browser window with the following details:

- Curl:** curl -X 'DELETE' '\<http://127.0.0.1:8000/books/1>' \-H 'accept: application/json'
- Request URL:** <http://127.0.0.1:8000/books/1>
- Server response:**
 - Code:** 200
 - Response body:** {"message": "Book with ID 1 deleted successfully."}
 - Response headers:** content-length: 50, content-type: application/json, date: Mon, 27 Oct 2025 04:53:48 GMT, server: unicorn
- Responses:**

Code	Description	Links
200	Successful Response	No links

Task 3 – Employee Payroll API Task:

Create an API for managing employees and their salaries.

Instructions:

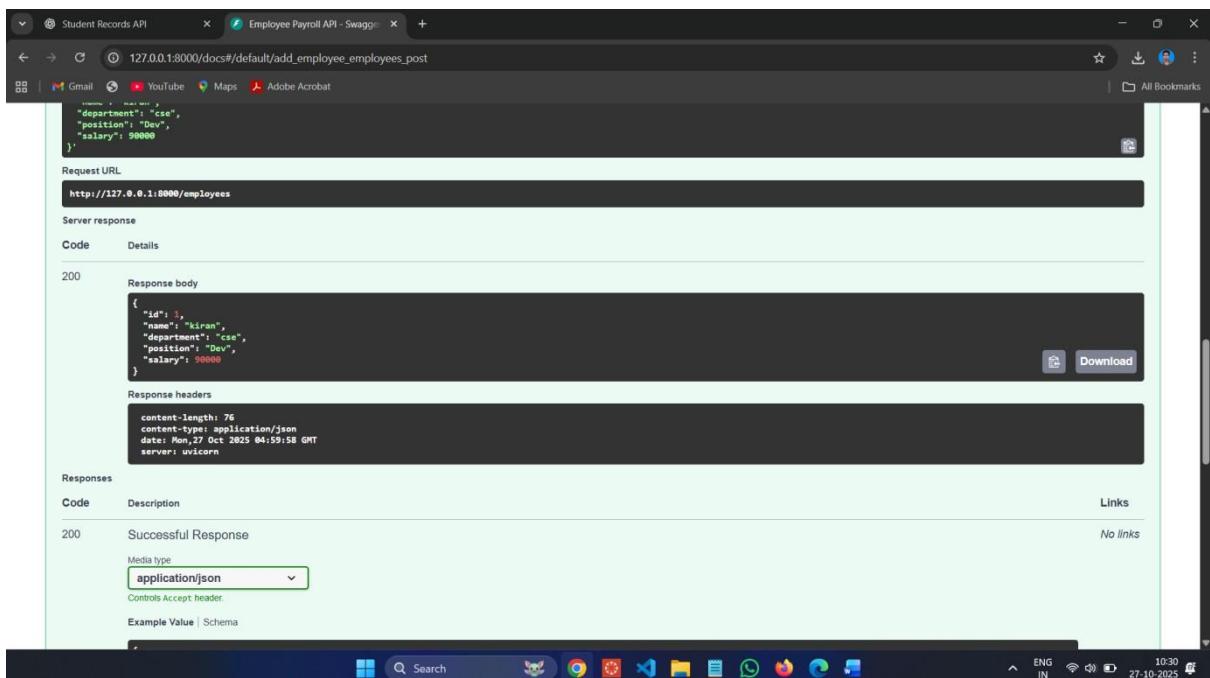
- Endpoints required:
 - GET /employees → List all employees
 - POST /employees → Add a new employee with salary details
 - PUT /employees/{id}/salary → Update salary of an employee
 - DELETE /employees/{id} → Remove employee from system
- Use AI to:
 - Suggest data model structure.
 - Add comments/docstrings for all endpoints. **Code :**

Commands :

```
uvicorn lab15_employee_api:app --reload
```

Swagger UI: <http://127.0.0.1:8000/docs> POST

6



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: 79
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

application/json

ENG IN 10:30 27-10-2025

PUT :

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

ENG IN 10:31 27-10-2025

DELETE :

The screenshot shows a browser window with two tabs: "Student Records API" and "Employee Payroll API - Swagger". The main content area is titled "Employee Payroll API - Swagger" and shows a DELETE operation for an employee record. The "Responses" section for a 200 status code includes a "Curl" command, a "Request URL" (http://127.0.0.1:8000/employees/1), and a "Server response" block. The response body contains the message: "Employee with ID 1 deleted successfully." Below this, there are sections for "Response headers" (Content-Length: 54, Content-Type: application/json, Date: Mon, 27 Oct 2025 05:01:48 GMT, Server: uvicorn) and "Responses" (Code: 200, Description: Successful Response, Media type: application/json). The bottom of the screen shows a Windows taskbar with various icons and system status.

Task 4 – Real-Time Application: Online Food Ordering API Scenario:

Design a simple API for an online food ordering system.

Requirements:

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

Code :

```

lab15_food_api.py
1 # lab15_food_api.py
2 from fastapi import FastAPI, HTTPException
3 from pydantic import BaseModel, Field
4 from typing import List, Dict, Optional
5
6 app = FastAPI(
7     title="Online Food Ordering API",
8     description="A RESTful API that simulates a basic online food ordering system",
9     version="1.0.0"
10 )
11
12 # ----- Data Models -----
13 class Dish(BaseModel):
14     """Represents a menu item (dish) available for order."""
15     id: int
16     name: str
17     price: float
18
19 class Order(BaseModel):
20     """Represents a customer's food order."""
21     id: int
22     items: List[int] = Field(..., description="List of dish IDs included in the order")
23     status: str = Field(default="Pending", description="Current status of the order")
24     total: float = 0.0
25
26 class OrderUpdate(BaseModel):
27     """Represents updates that can be applied to an existing order."""
28     items: Optional[List[int]] = None
29     status: Optional[str] = None
30
31 # ----- In-memory databases -----
32 menu_db: Dict[int, Dish] = {
33     1: Dish(id=1, name="Margherita Pizza", price=8.99),
34     2: Dish(id=2, name="Cheeseburger", price=7.49),
35     3: Dish(id=3, name="Pasta Alfredo", price=9.25),
36     4: Dish(id=4, name="Caesar Salad", price=5.75),
37 }

```

Commands :

uvicorn lab15_food_api:app --reload

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

POST :

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": 16.48
}'
```

Request URL

<http://127.0.0.1:8000/order>

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 2, "items": [1, 2], "status": "Pending", "total": 16.48 }</pre> <p>Response headers</p> <pre>content-length: 55 content-type: application/json date: Mon, 27 Oct 2025 05:04:35 GMT server: uvicorn</pre>

GET :

The screenshot shows a browser window with two tabs: "Student Records API" and "Online Food Ordering API - Sw". The "Online Food Ordering API - Sw" tab is active, displaying the URL "127.0.0.1:8000/docs#/default/update_order_order_id_put". Below the URL, there are sections for "Responses", "Curl", "Request URL", "Server response", and "Responses". The "Responses" section shows a table with one row for code 200, labeled "Successful Response". The "Server response" section shows a 200 status with a response body containing JSON data:

```
{ "id": 5, "items": [ { "id": 1, "name": "Pasta", "quantity": 1, "status": "Pending", "total": 8.99 } ]}
```

The response headers are:

```
content-length: 52
content-type: application/json
date: Mon, 27 Oct 2025 05:12:31 GMT
server: unicorn
```

The browser's taskbar at the bottom shows various icons, and the system tray indicates the date and time as 27-10-2025.

PUT :

The screenshot shows a browser window with two tabs: "Student Records API" and "Online Food Ordering API - Sw". The "Online Food Ordering API - Sw" tab is active, displaying the URL "127.0.0.1:8000/docs#/default/update_order_order_id_put". Below the URL, there are sections for "Responses", "Curl", "Request URL", "Server response", and "Responses". The "Responses" section shows a table with one row for code 200, labeled "Successful Response". The "Server response" section shows a 200 status with a response body containing JSON data:

```
{ "id": 5, "items": [ { "id": 1, "name": "Pasta", "quantity": 1, "status": "Completed", "total": 8.99 } ]}
```

The response headers are:

```
content-length: 54
content-type: application/json
date: Mon, 27 Oct 2025 05:13:46 GMT
server: unicorn
```

The browser's taskbar at the bottom shows various icons, and the system tray indicates the date and time as 27-10-2025.

DELETE :

The screenshot shows a web browser window with two tabs open: "Student Records API" and "Online Food Ordering API - Swift". The "Online Food Ordering API - Swift" tab is active, displaying a DELETE request for an order with ID 5. The request details are as follows:

- 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	Description	Links
200	Successful Response	No links
- Response body:**

```
{ "message": "Order 5 has been cancelled successfully." }
```
- Response headers:**

```
content-length: 54
content-type: application/json
date: Mon, 27 Oct 2025 09:14:18 GMT
server: uvicorn
```

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, realtime updates (via WebSockets), and database integration demonstrate how AI can enhance backend architecture planning.