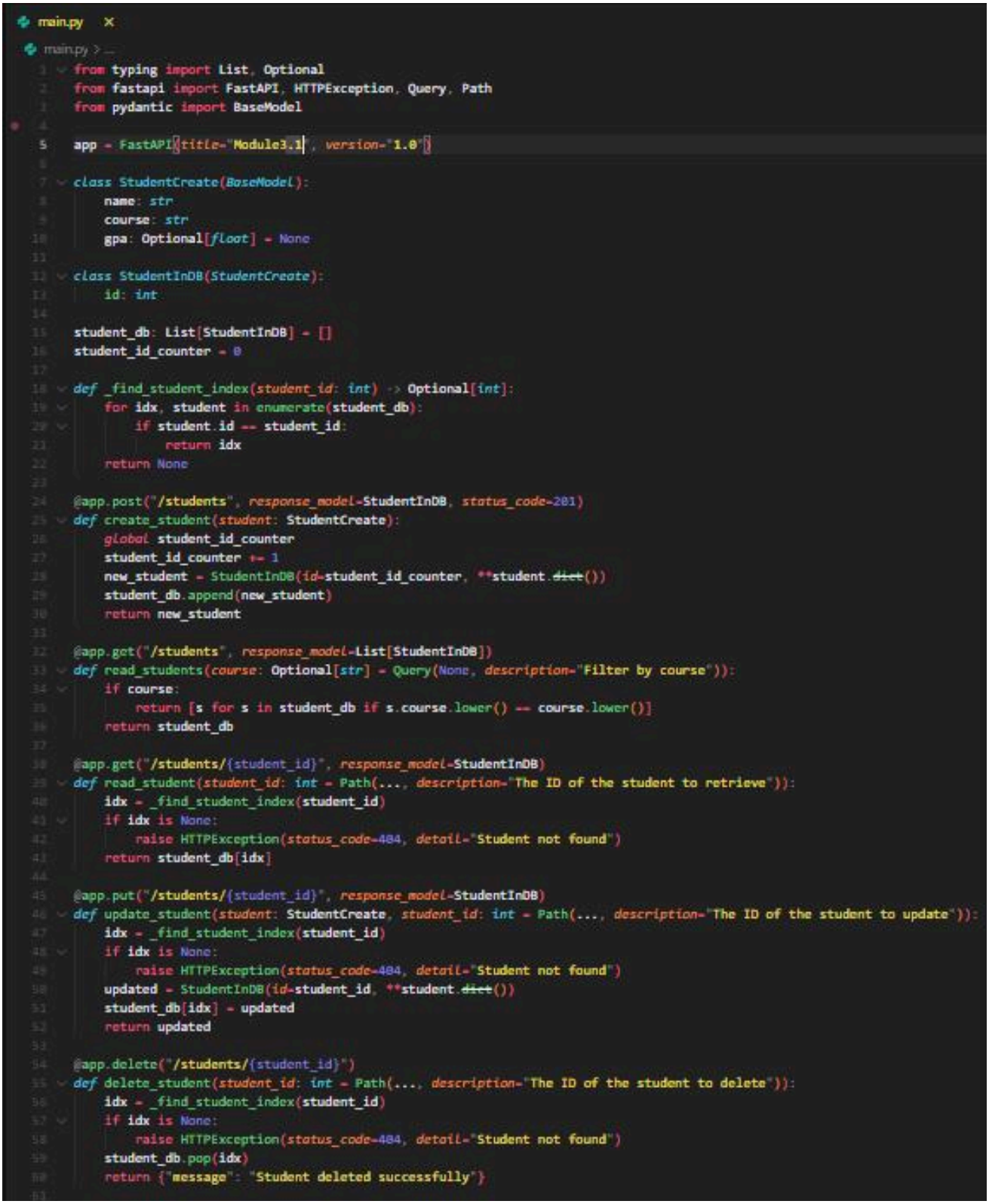
JAN LEANDER MATUTES
BSIT - 3A

Module 3.1 Activity: Building a Full CRUD API with Pydantic

main.py



```python
from typing import List, Optional
from fastapi import FastAPI, HTTPException, Query, Path
from pydantic import BaseModel

app = FastAPI(title="Module3.1", version="1.0")

class StudentCreate(BaseModel):
    name: str
    course: str
    gpa: Optional[float] = None

class StudentInDB(StudentCreate):
    id: int

student_db: List[StudentInDB] = []
student_id_counter = 0

def _find_student_index(student_id: int) -> Optional[int]:
    for idx, student in enumerate(student_db):
        if student.id == student_id:
            return idx
    return None

@app.post("/students", response_model=StudentInDB, status_code=201)
def create_student(student: StudentCreate):
    global student_id_counter
    student_id_counter += 1
    new_student = StudentInDB(id=student_id_counter, **student.dict())
    student_db.append(new_student)
    return new_student

@app.get("/students", response_model=List[StudentInDB])
def read_students(course: Optional[str] = Query(None, description="Filter by course")):
    if course:
        return [s for s in student_db if s.course.lower() == course.lower()]
    return student_db

@app.get("/students/{student_id}", response_model=StudentInDB)
def read_student(student_id: int = Path(..., description="The ID of the student to retrieve")):
    idx = _find_student_index(student_id)
    if idx is None:
        raise HTTPException(status_code=404, detail="Student not found")
    return student_db[idx]

@app.put("/students/{student_id}", response_model=StudentInDB)
def update_student(student: StudentCreate, student_id: int = Path(..., description="The ID of the student to update")):
    idx = _find_student_index(student_id)
    if idx is None:
        raise HTTPException(status_code=404, detail="Student not found")
    updated = StudentInDB(id=student_id, **student.dict())
    student_db[idx] = updated
    return updated

@app.delete("/students/{student_id}")
def delete_student(student_id: int = Path(..., description="The ID of the student to delete")):
    idx = _find_student_index(student_id)
    if idx is None:
        raise HTTPException(status_code=404, detail="Student not found")
    student_db.pop(idx)
    return {"message": "Student deleted successfully"}
```

Module3.1 1.0 OAS 3.1

/openapi.json

default

POST /students Create Student

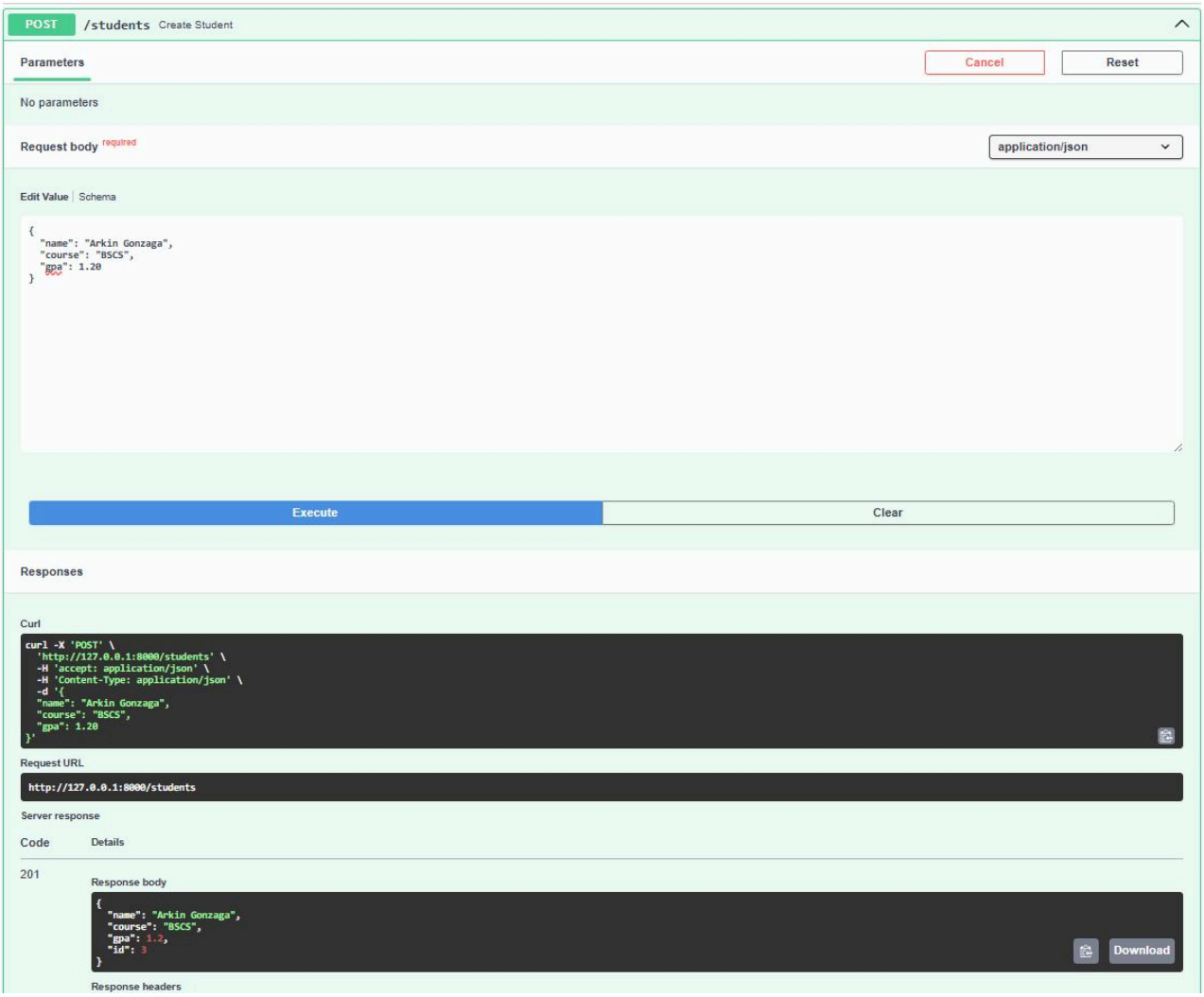
GET /students Read Students

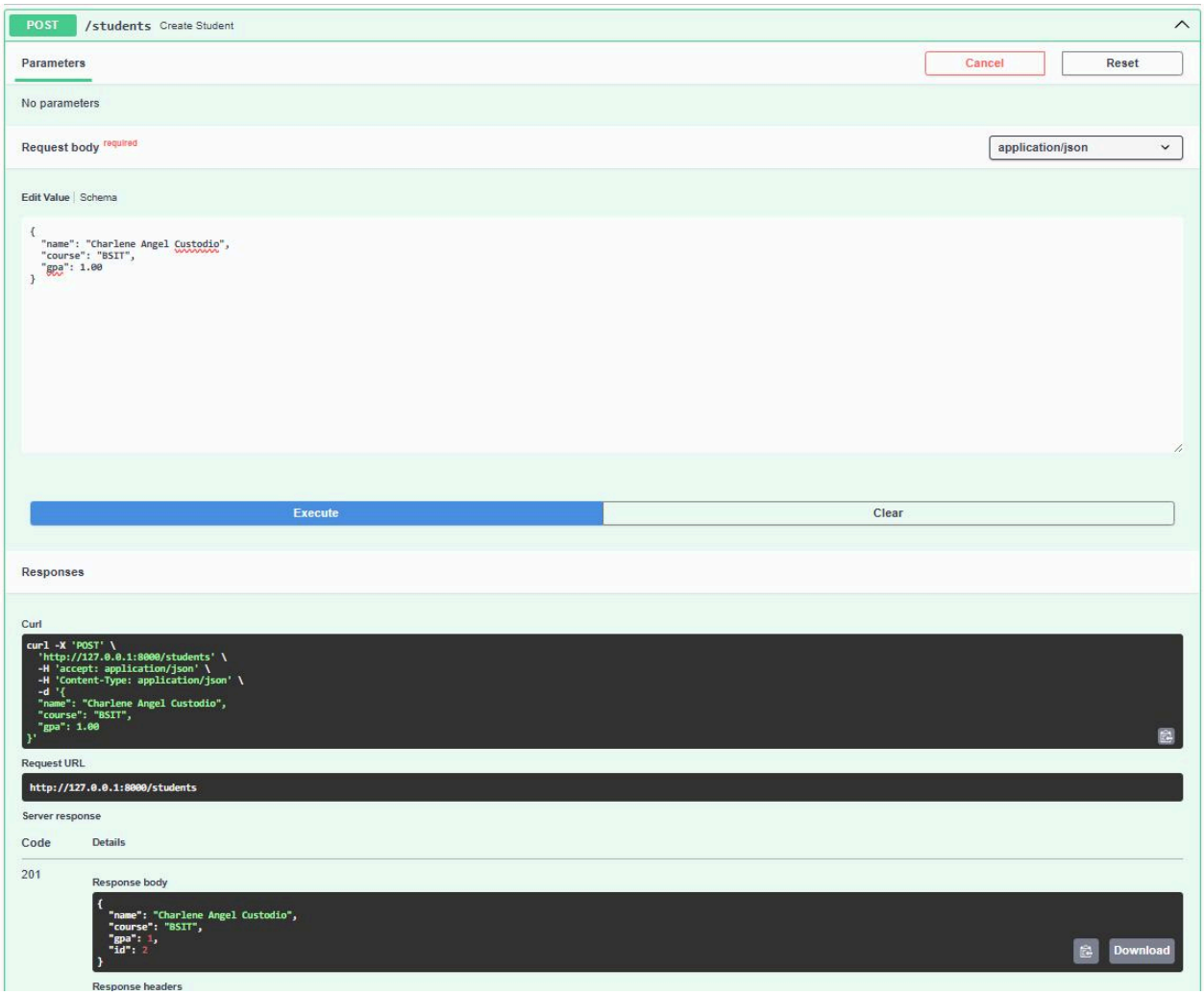
GET /students/{student_id} Read Student

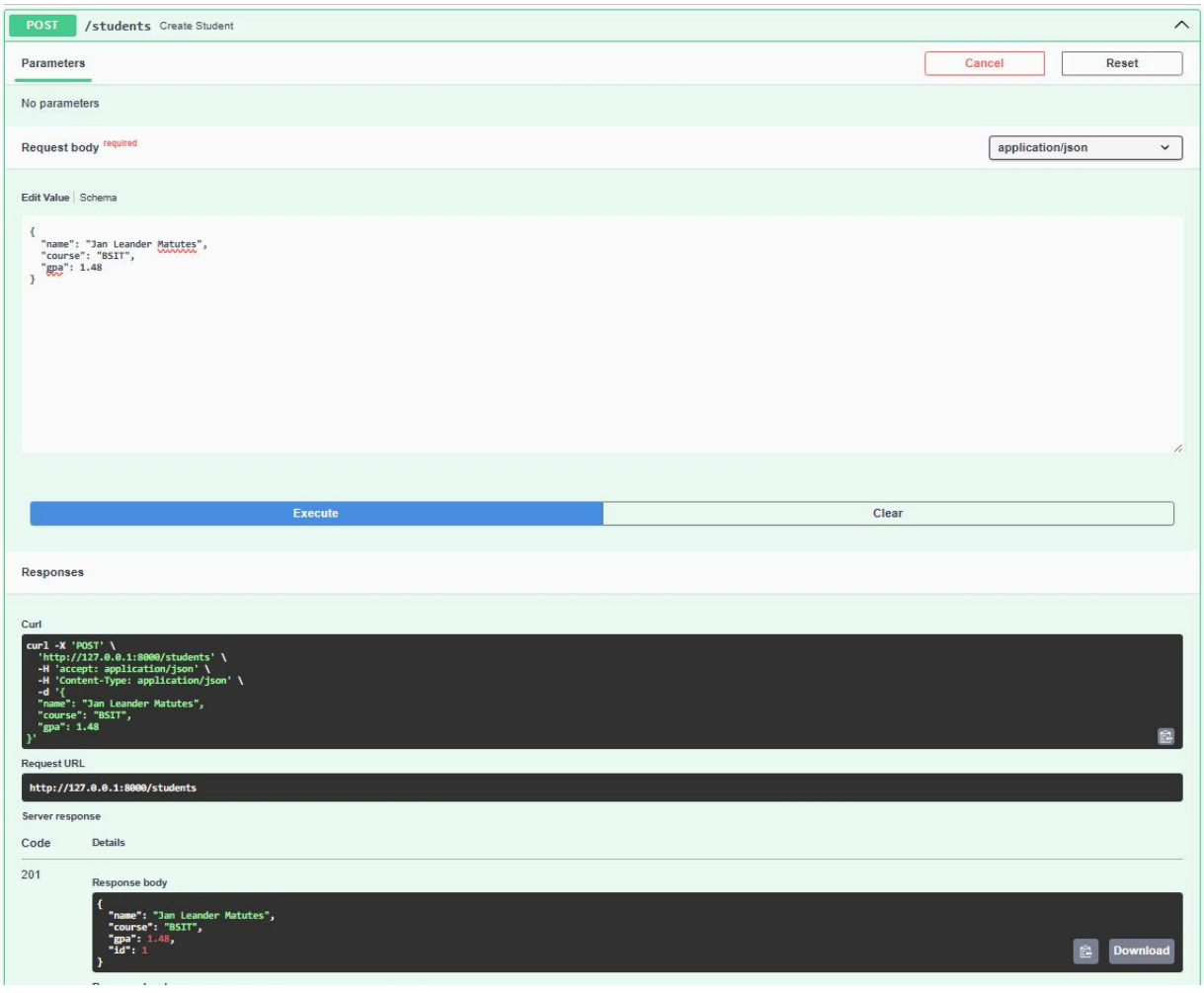
PUT /students/{student_id} Update Student

DELETE /students/{student_id} Delete Student

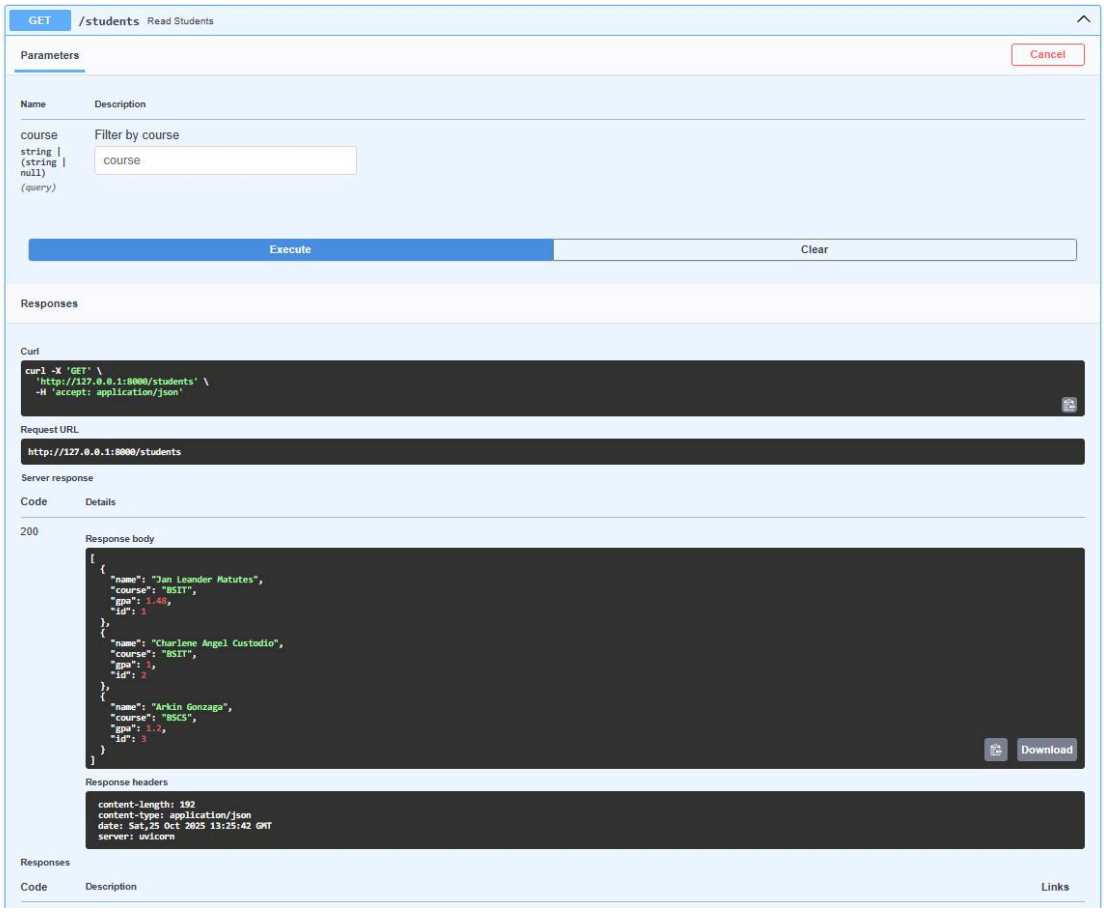
1. POST: Create 2-3 different students (e.g., some in "BSIT" and some in "BSCS").



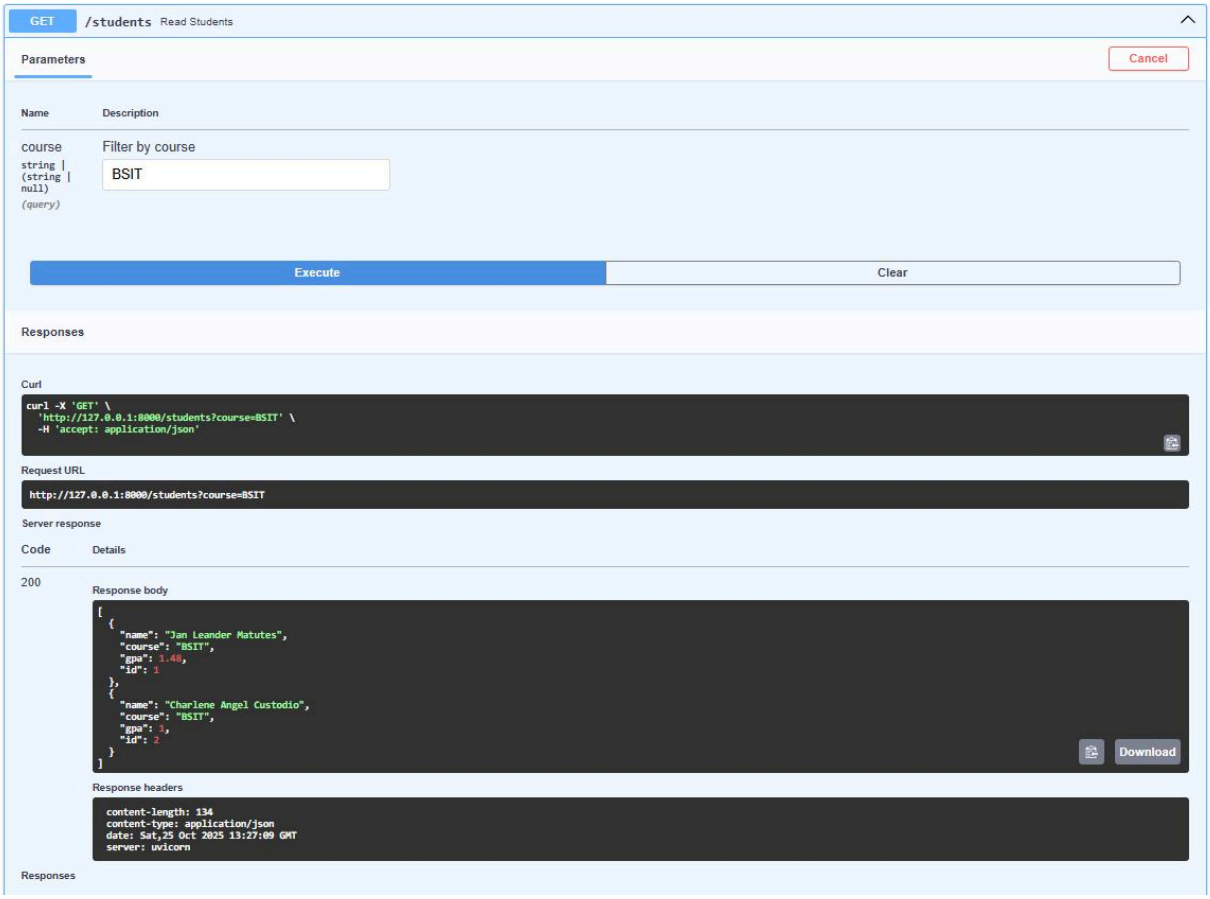

POST /students Create Student
Parameters
Cancel
Reset
No parameters
Request body required
application/json
Edit Value | Schema
{
"name": "Charlene Angel Custodio",
"course": "BSIT",
"gpa": 1.00
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/students' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"name": "Charlene Angel Custodio",
"course": "BSIT",
"gpa": 1.00
}'
Request URL
http://127.0.0.1:8000/students
Server response
Code
Details
201
Response body
{
"name": "Charlene Angel Custodio",
"course": "BSIT",
"gpa": 1,
"id": 2
}
Download
Response headers


POST /students Create Student
Parameters
Cancel
Reset
No parameters
Request body required
application/json
Edit Value | Schema
{
"name": "Jan Leander Matutes",
"course": "BSIT",
"gpa": 1.48
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/students' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"name": "Jan Leander Matutes",
"course": "BSIT",
"gpa": 1.48
}'
Request URL
http://127.0.0.1:8000/students
Server response
Code
Details
201
Response body
{
"name": "Jan Leander Matutes",
"course": "BSIT",
"gpa": 1.48,
"id": 1
}
Download

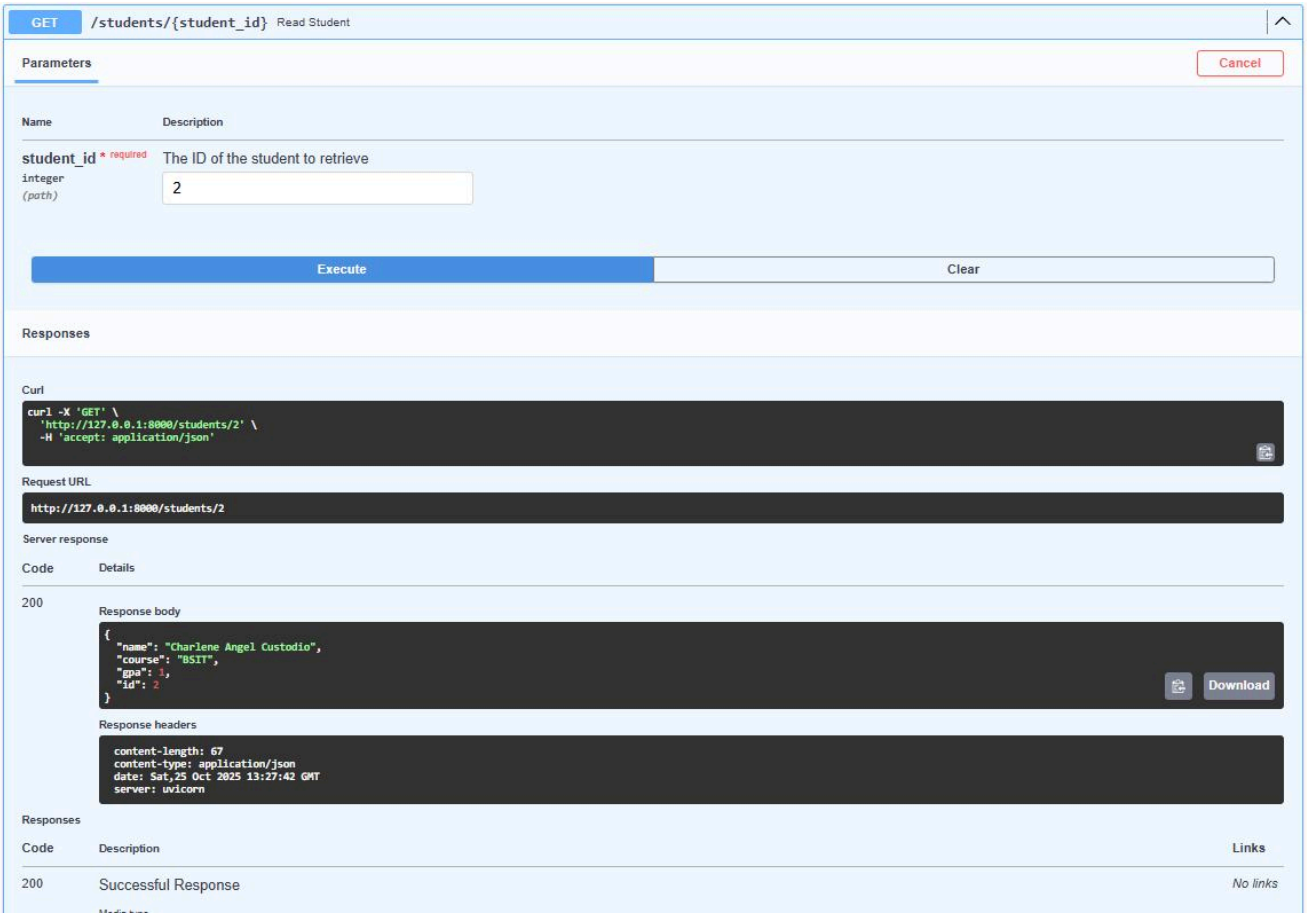
2. GET (All): Call the /students endpoint to see all students you created.



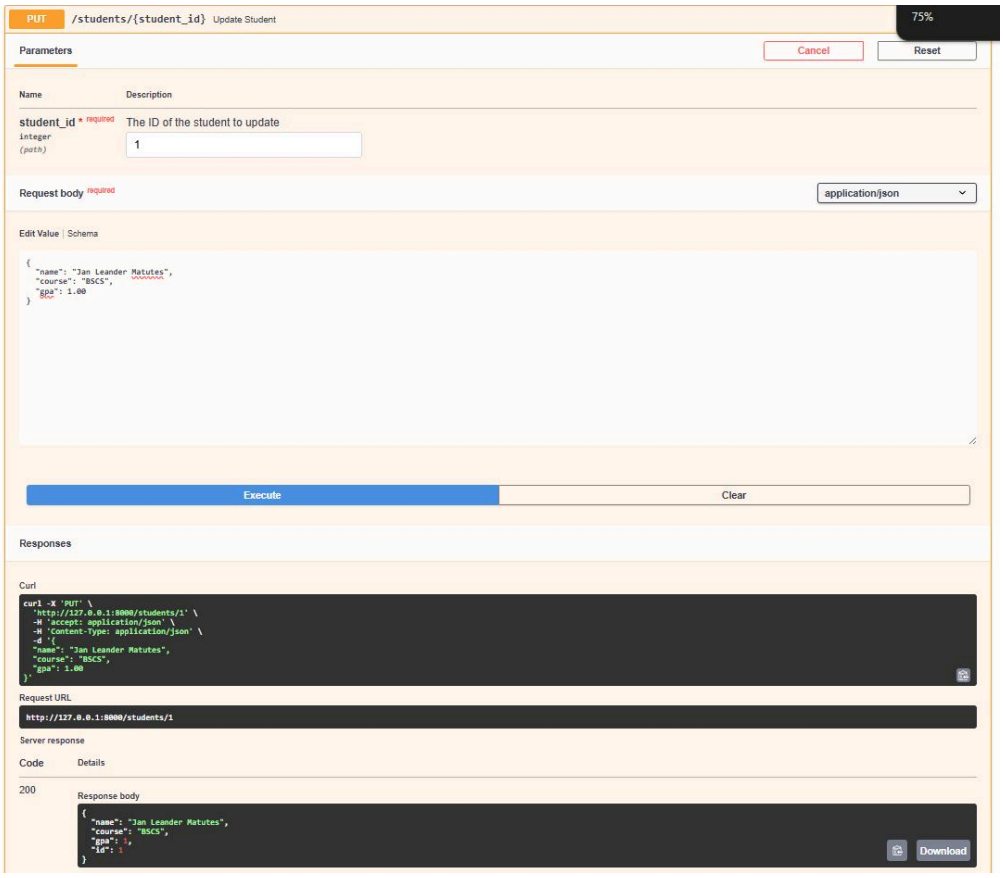
3. GET (Query): Call the /students?course=BSIT endpoint to test the filtering.



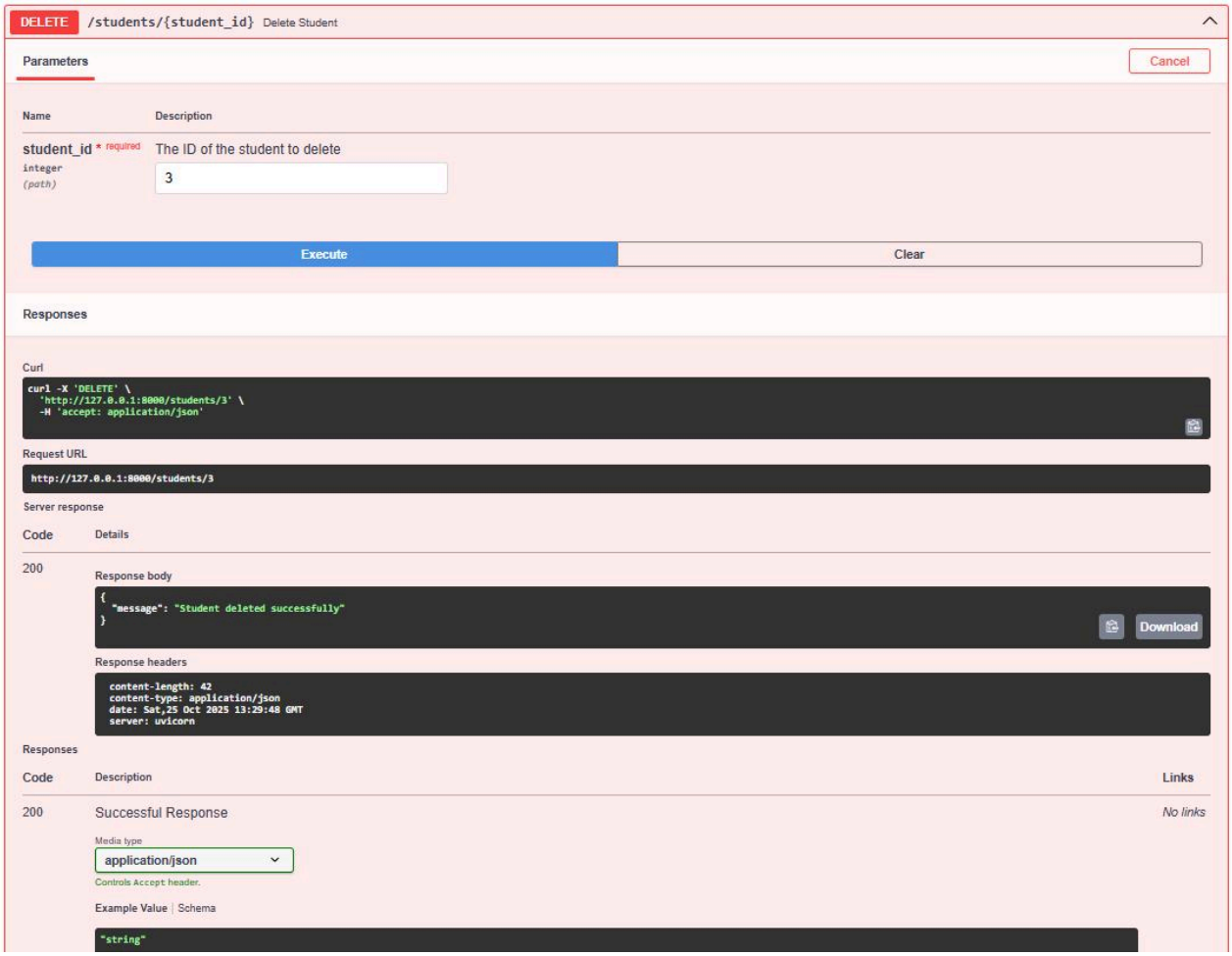
4. GET (One): Get one of the students by their ID (e.g., /students/1).



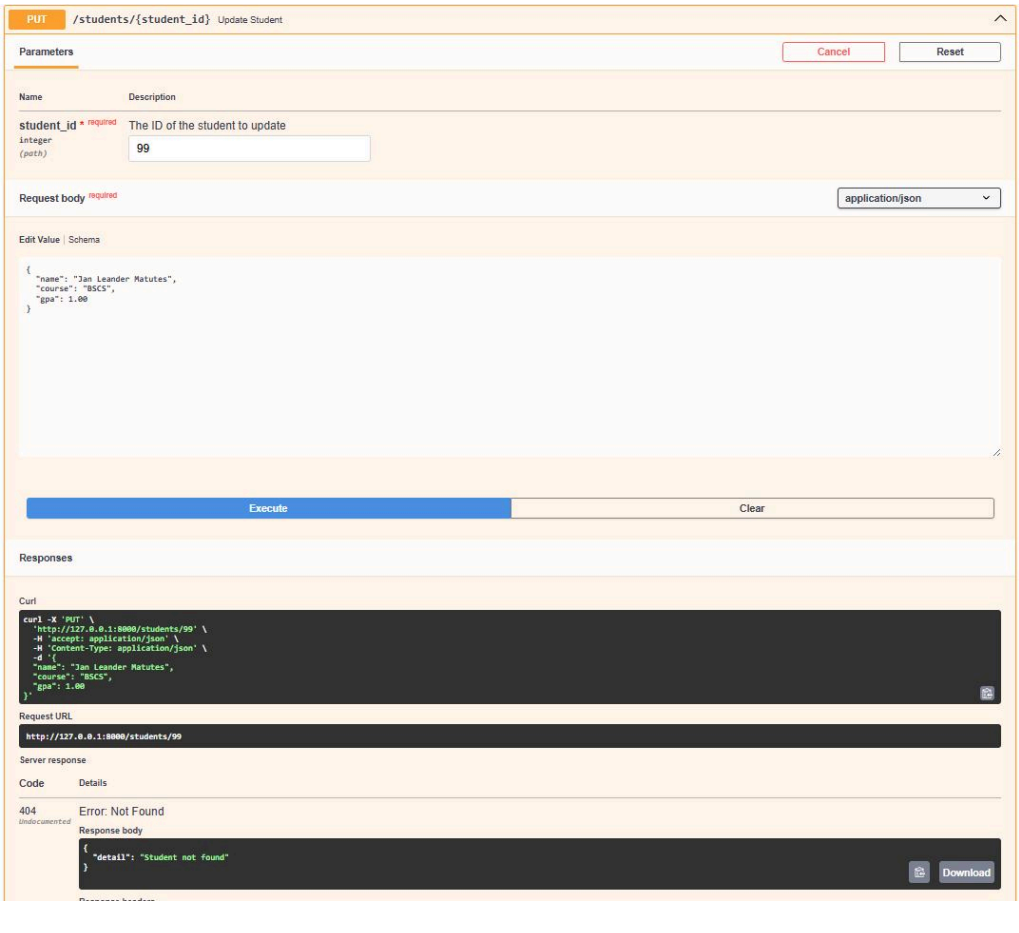
5. PUT: Update one of the students (e.g., change the gpa or course for /students/1).

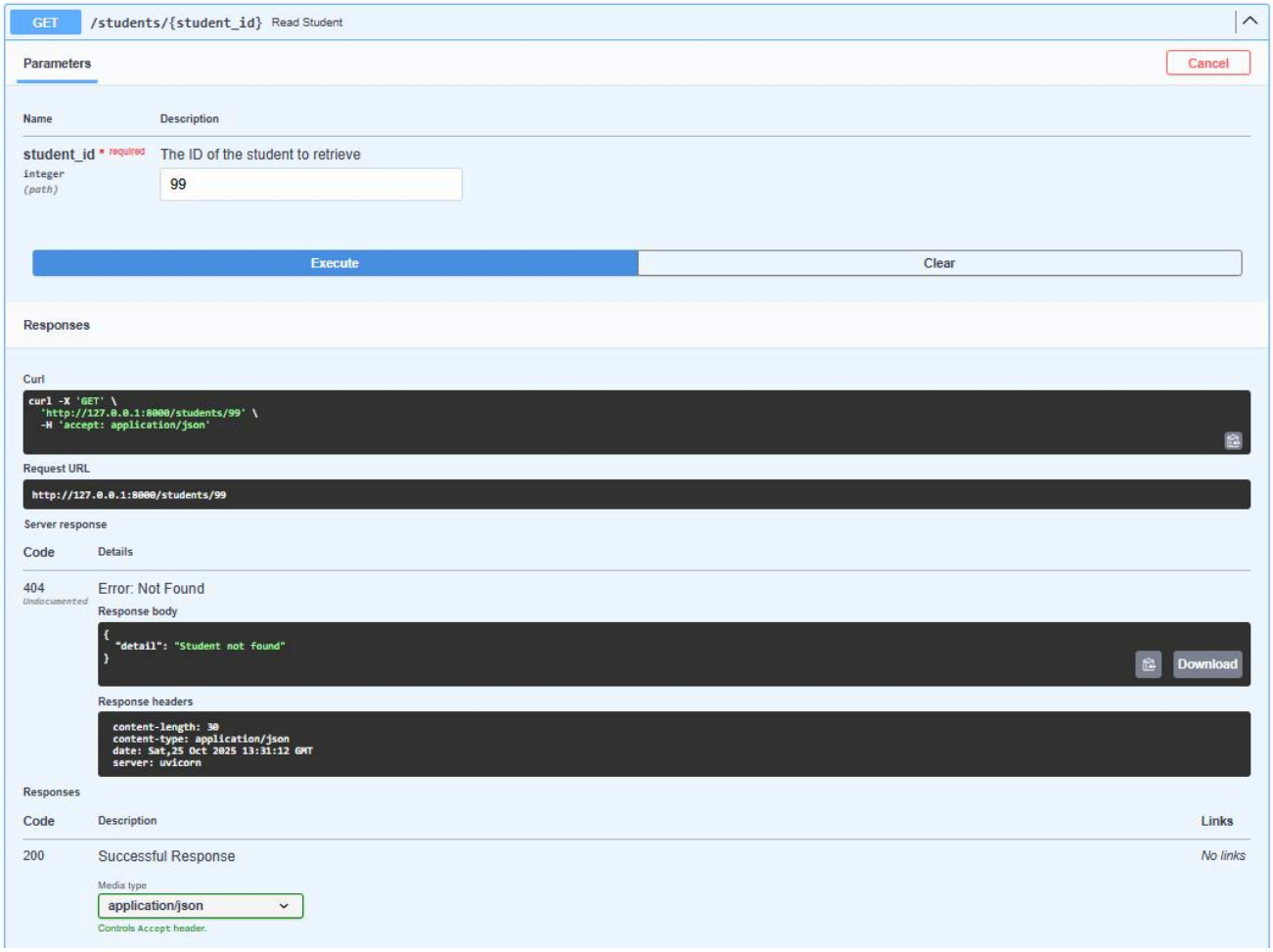


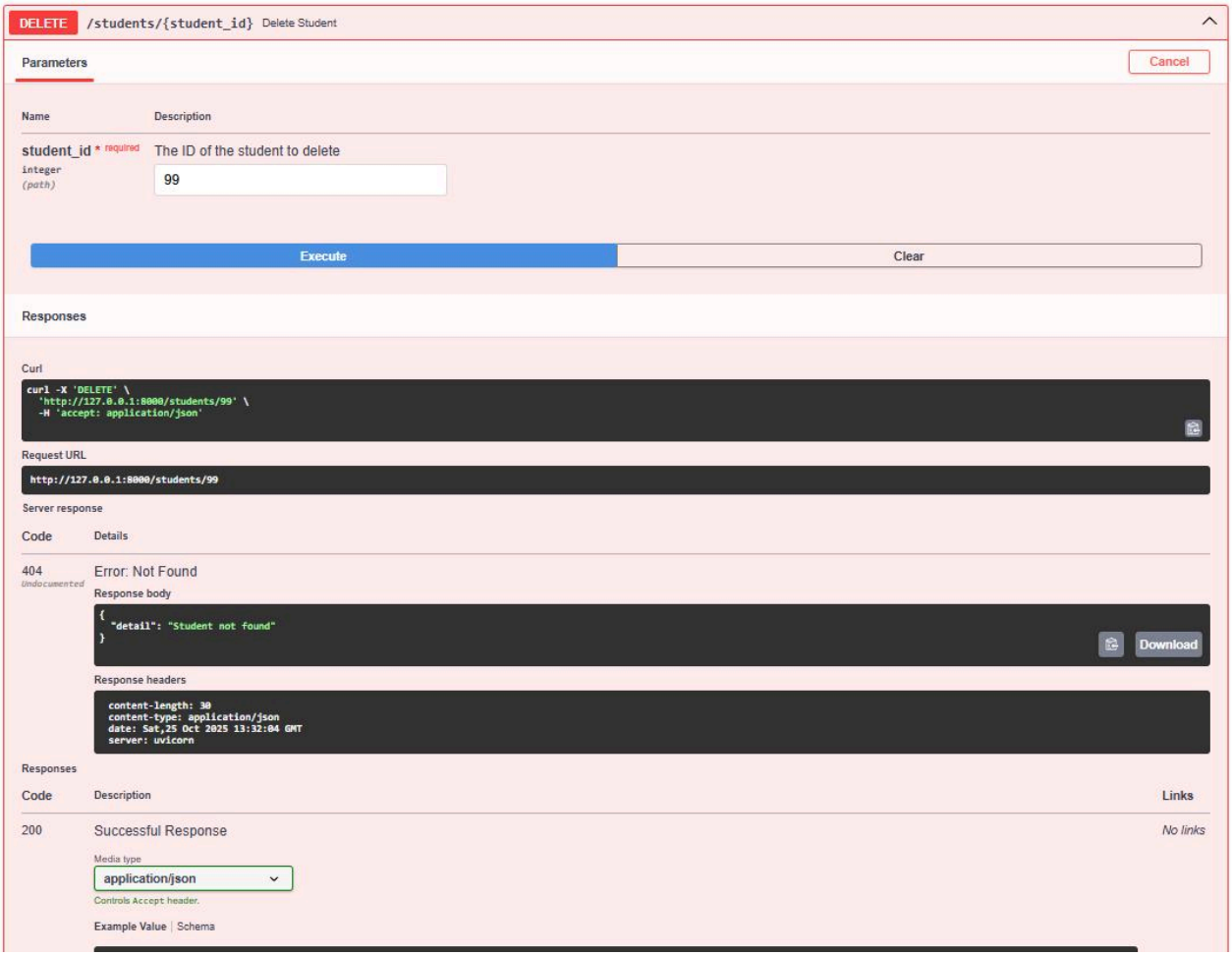
6. DELETE: Delete one of the students (e.g., /students/2).



7. GET (Invalid): Try to GET, PUT, or DELETE a student that doesn't exist (e.g., /students/99) to confirm you get a 404 "Not Found" error.




GET
/students/{student_id} Read Student
Parameters
Cancel
Name
Description
student_id * required
integer
(path)
The ID of the student to retrieve
99
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/students/99' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/students/99
Server response
Code
Details
404
Undocumented
Error: Not Found
Response body
{
"detail": "Student not found"
}
Download
Response headers
content-length: 30
content-type: application/json
date: Sat,25 Oct 2025 13:31:12 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.


DELETE
/students/{student_id} Delete Student
Parameters
Cancel
Name
Description
student_id * required
integer
(path)
The ID of the student to delete
99
Execute
Clear
Responses
Curl
curl -X 'DELETE' \
'http://127.0.0.1:8000/students/99' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/students/99
Server response
Code
Details
404
Undocumented
Error: Not Found
Response body
{
"detail": "Student not found"
}
Download
Response headers
content-length: 30
content-type: application/json
date: Sat,25 Oct 2025 13:32:04 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value | Schema