TASK REPORT

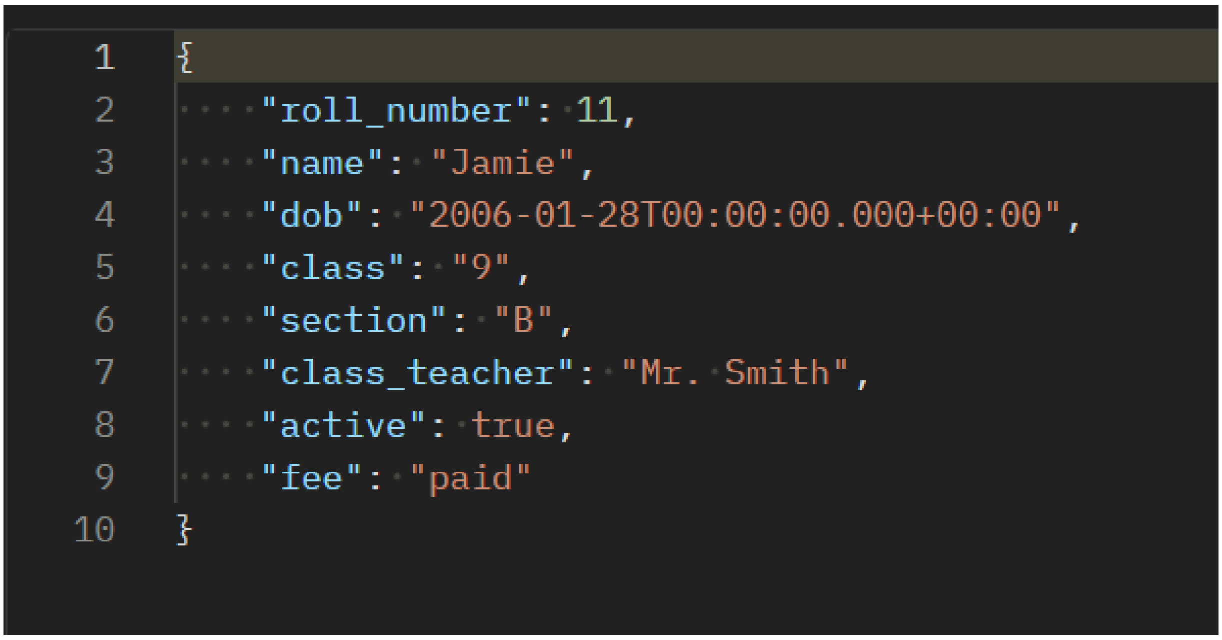
Problem statement:

Create a Python Flask API

1. **PUT Operation:** Create student information and generate a student ID for each new student.
2. **GET Operation:** Retrieve information for a single student based on the student ID.
3. **GET Operation:** Retrieve information for all students.
4. **PUT Operation:** Update any student information based on the student ID provided.
5. **DELETE Operation:** Delete student information.
6. **POST Operation:** User login with user ID and password verification (this service can have hard-coded values for user ID and password).

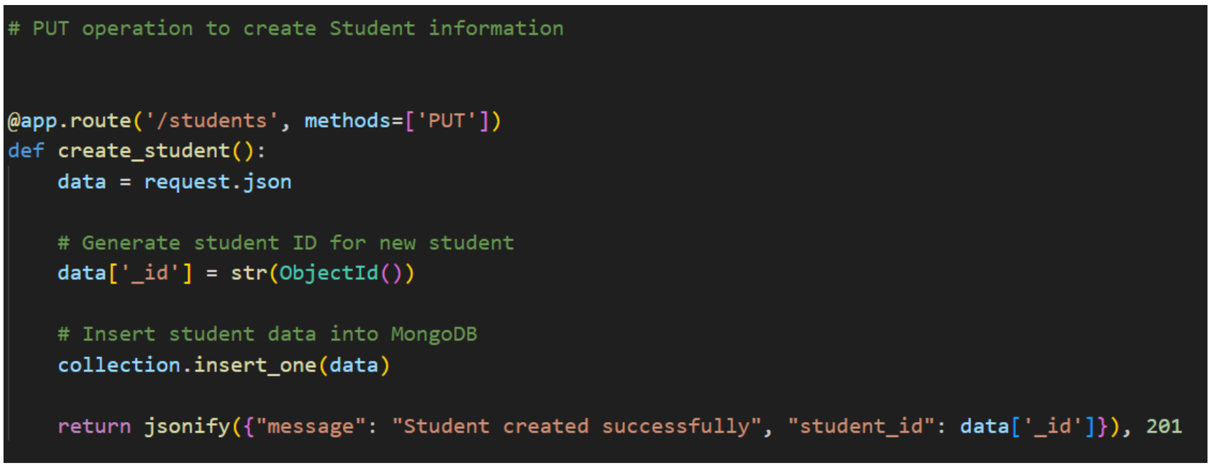
All student data must be stored in MongoDB. The student information should be in JSON format.

Sample student data:

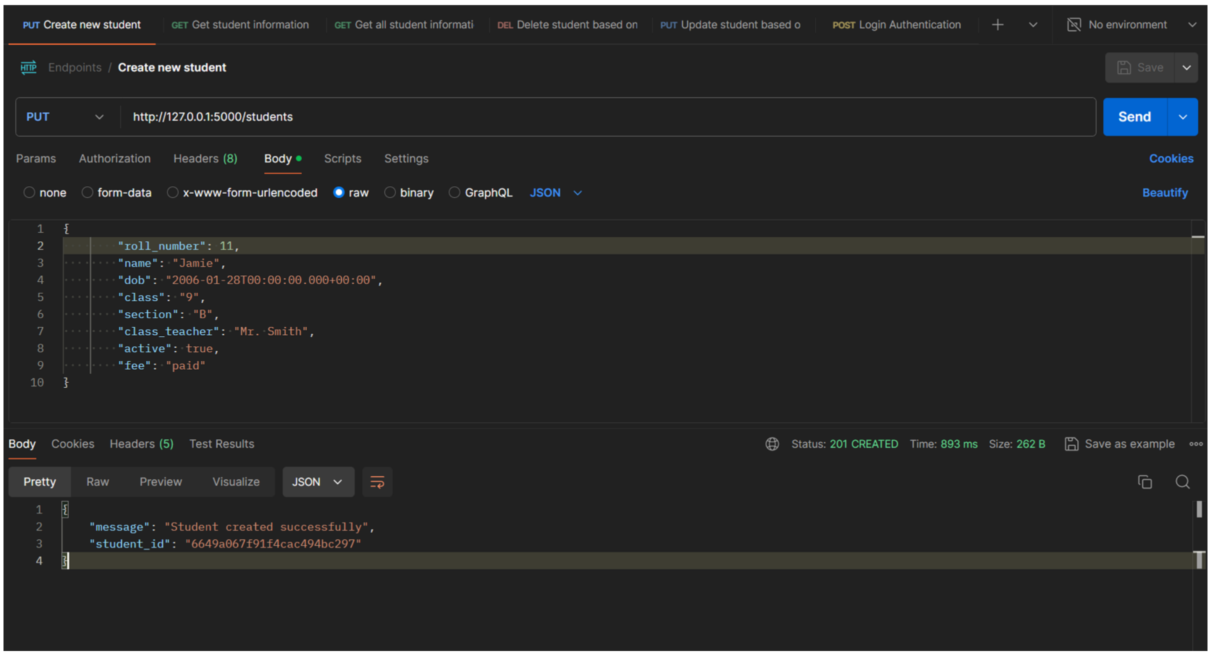


```json
{
    "roll_number": 11,
    "name": "Jamie",
    "dob": "2006-01-28T00:00:00.000+00:00",
    "class": "9",
    "section": "B",
    "class_teacher": "Mr. Smith",
    "active": true,
    "fee": "paid"
}
```

Creating student information and generate a student ID for each new student.



```python
# PUT operation to create Student information


@app.route('/students', methods=['PUT'])
def create_student():
    data = request.json

    # Generate student ID for new student
    data['_id'] = str(ObjectId())

    # Insert student data into MongoDB
    collection.insert_one(data)

    return jsonify({"message": "Student created successfully", "student_id": data['_id']}), 201
```



PUT http://127.0.0.1:5000/students

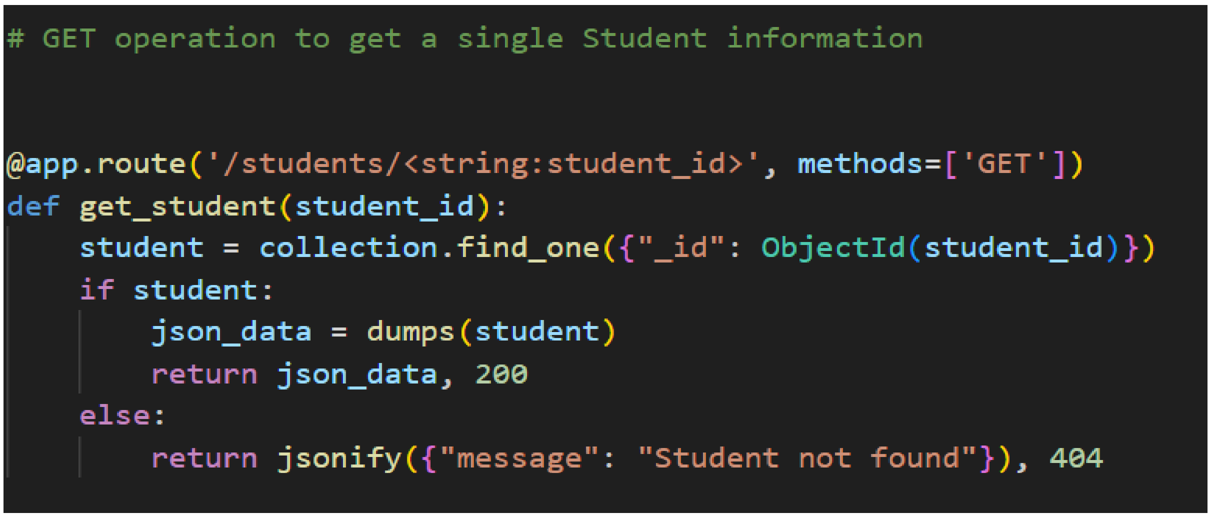
Body: none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```json
{
        "roll_number": 11,
        "name": "Jamie",
        "dob": "2006-01-28T00:00:00.000+00:00",
        "class": "9",
        "section": "B",
        "class_teacher": "Mr. Smith",
        "active": true,
        "fee": "paid"
}
```

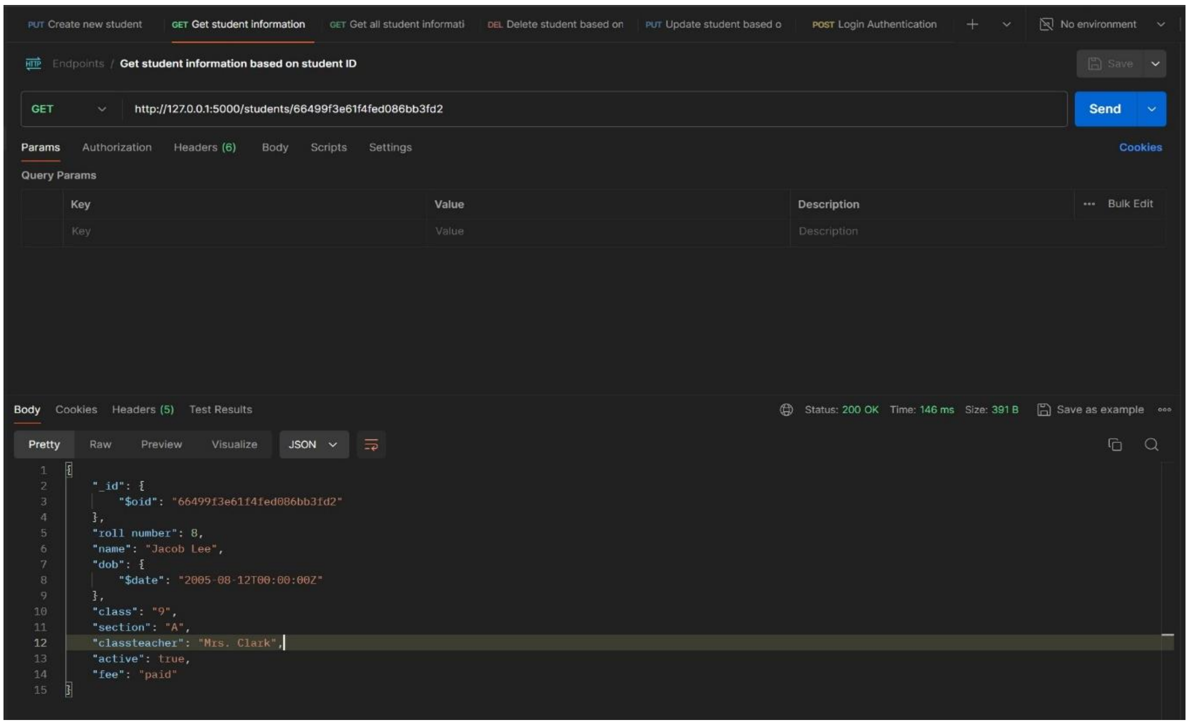
Body   Status: 201 CREATED   Time: 893 ms   Size: 262 B

```json
{
    "message": "Student created successfully",
    "student_id": "6649a067f91f4cac494bc297"
}
```

Retrieving information for a single student based on the student ID.



```python
# GET operation to get a single Student information


@app.route('/students/<string:student_id>', methods=['GET'])
def get_student(student_id):
    student = collection.find_one({"_id": ObjectId(student_id)})
    if student:
        json_data = dumps(student)
        return json_data, 200
    else:
        return jsonify({"message": "Student not found"}), 404
```

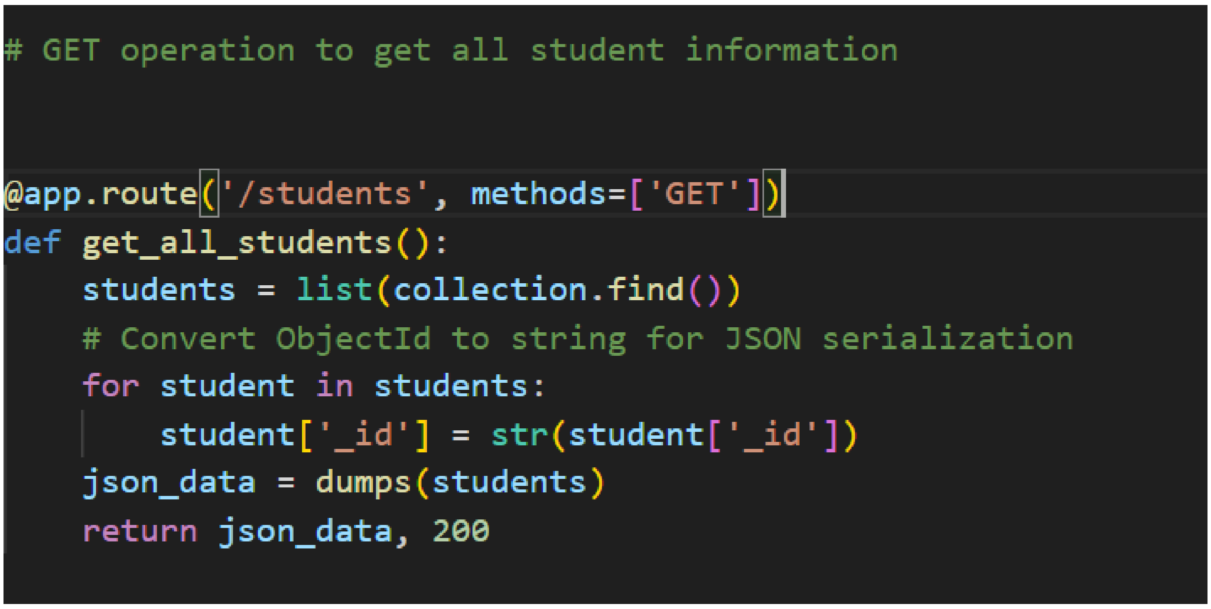


GET http://127.0.0.1:5000/students/66499f3e61f4fed086bb3fd2

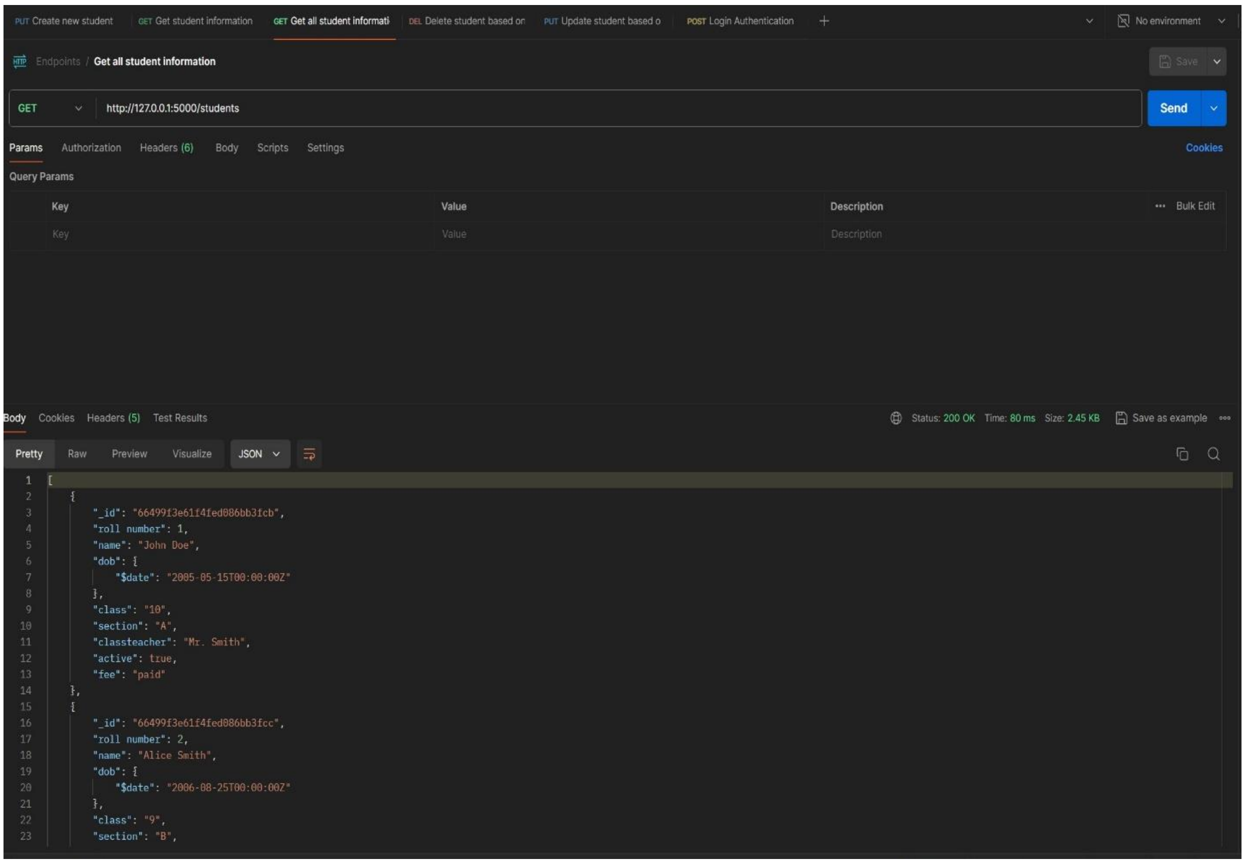
Status: 200 OK Time: 146 ms Size: 391 B

```
{
    "_id": {
        "$oid": "66499f3e61f4fed086bb3fd2"
    },
    "roll number": 8,
    "name": "Jacob Lee",
    "dob": {
        "$date": "2005-08-12T00:00:00Z"
    },
    "class": "9",
    "section": "A",
    "classteacher": "Mrs. Clark",
    "active": true,
    "fee": "paid"
}
```

Retrieving information for all students.



```
# GET operation to get all student information


@app.route('/students', methods=['GET'])
def get_all_students():
    students = list(collection.find())
    # Convert ObjectId to string for JSON serialization
    for student in students:
        student['_id'] = str(student['_id'])
    json_data = dumps(students)
    return json_data, 200
```

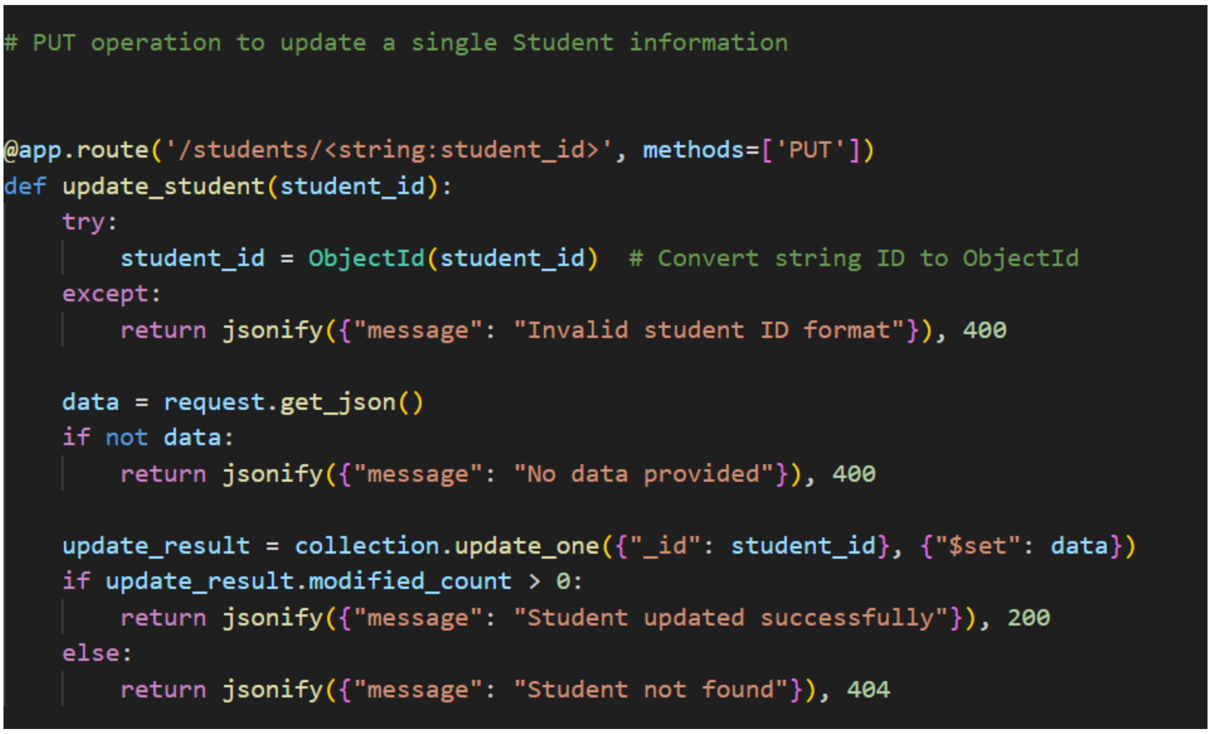


GET http://127.0.0.1:5000/students

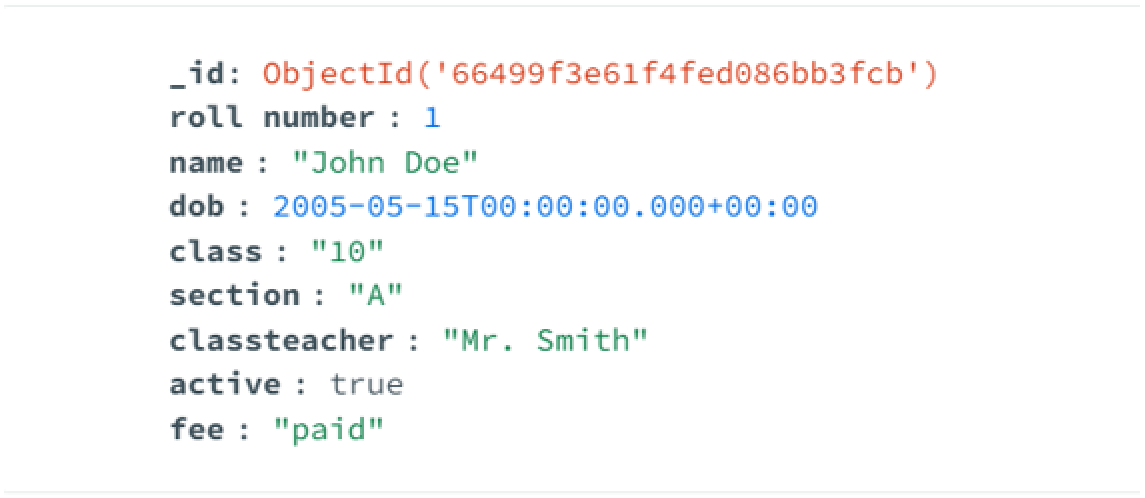
Status: 200 OK Time: 80 ms Size: 2.45 KB

```
[
    {
        "_id": "66499f3e61f4fed086bb3fcb",
        "roll number": 1,
        "name": "John Doe",
        "dob": {
            "$date": "2005-05-15T00:00:00Z"
        },
        "class": "10",
        "section": "A",
        "classteacher": "Mr. Smith",
        "active": true,
        "fee": "paid"
    },
    {
        "_id": "66499f3e61f4fed086bb3fcc",
        "roll number": 2,
        "name": "Alice Smith",
        "dob": {
            "$date": "2006-08-25T00:00:00Z"
        },
        "class": "9",
        "section": "B",
```

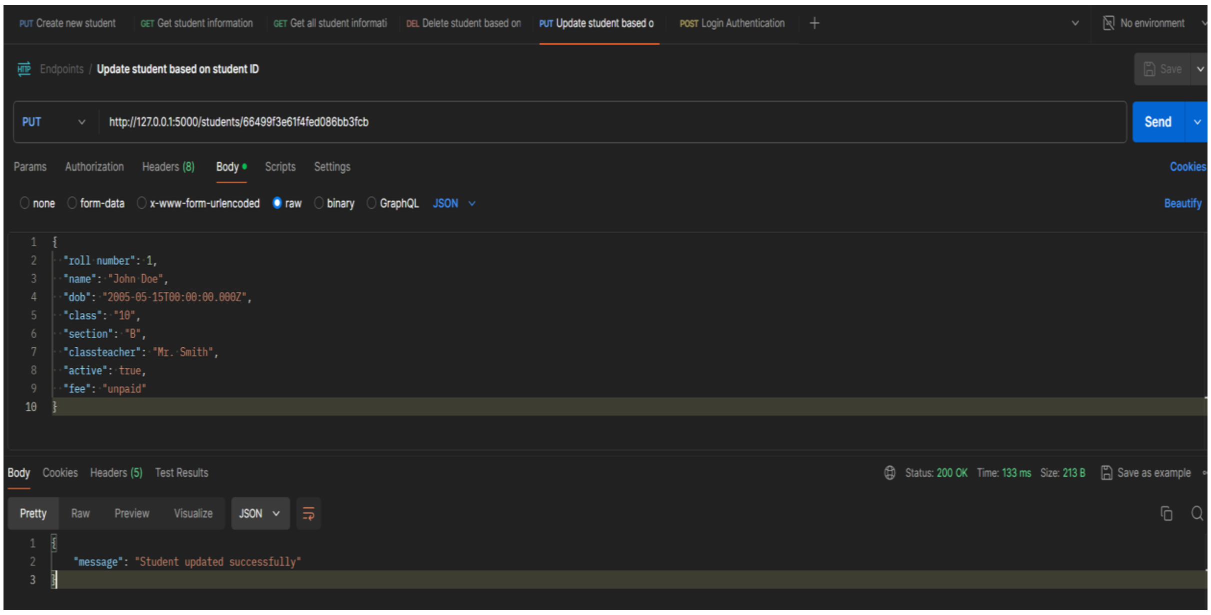
Updating any student information based on the student ID provided.



```python
# PUT operation to update a single Student information


@app.route('/students/<string:student_id>', methods=['PUT'])
def update_student(student_id):
    try:
        student_id = ObjectId(student_id)  # Convert string ID to ObjectId
    except:
        return jsonify({"message": "Invalid student ID format"}), 400

    data = request.get_json()
    if not data:
        return jsonify({"message": "No data provided"}), 400

    update_result = collection.update_one({"_id": student_id}, {"$set": data})
    if update_result.modified_count > 0:
        return jsonify({"message": "Student updated successfully"}), 200
    else:
        return jsonify({"message": "Student not found"}), 404
```



```
_id: ObjectId('66499f3e61f4fed086bb3fcb')
roll number : 1
name : "John Doe"
dob : 2005-05-15T00:00:00.000+00:00
class : "10"
section : "A"
classteacher : "Mr. Smith"
active : true
fee : "paid"
```



Endpoints / Update student based on student ID

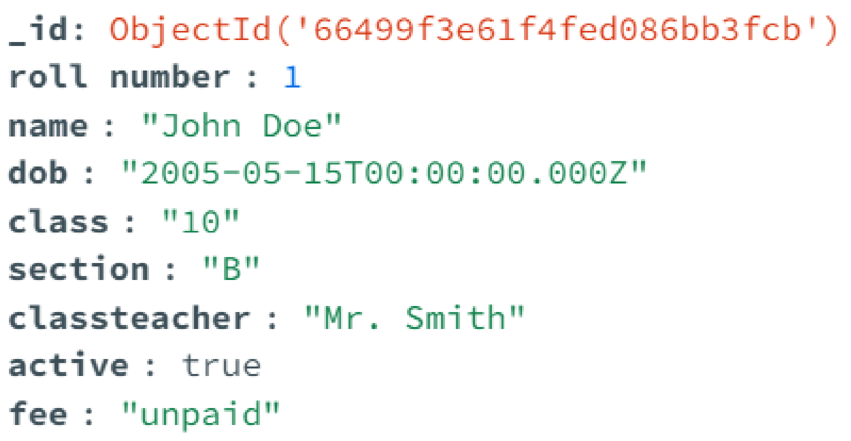
PUT http://127.0.0.1:5000/students/66499f3e61f4fed086bb3fcb

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
  "roll number": 1,
  "name": "John Doe",
  "dob": "2005-05-15T00:00:00.000Z",
  "class": "10",
  "section": "B",
  "classteacher": "Mr. Smith",
  "active": true,
  "fee": "unpaid"
}
```

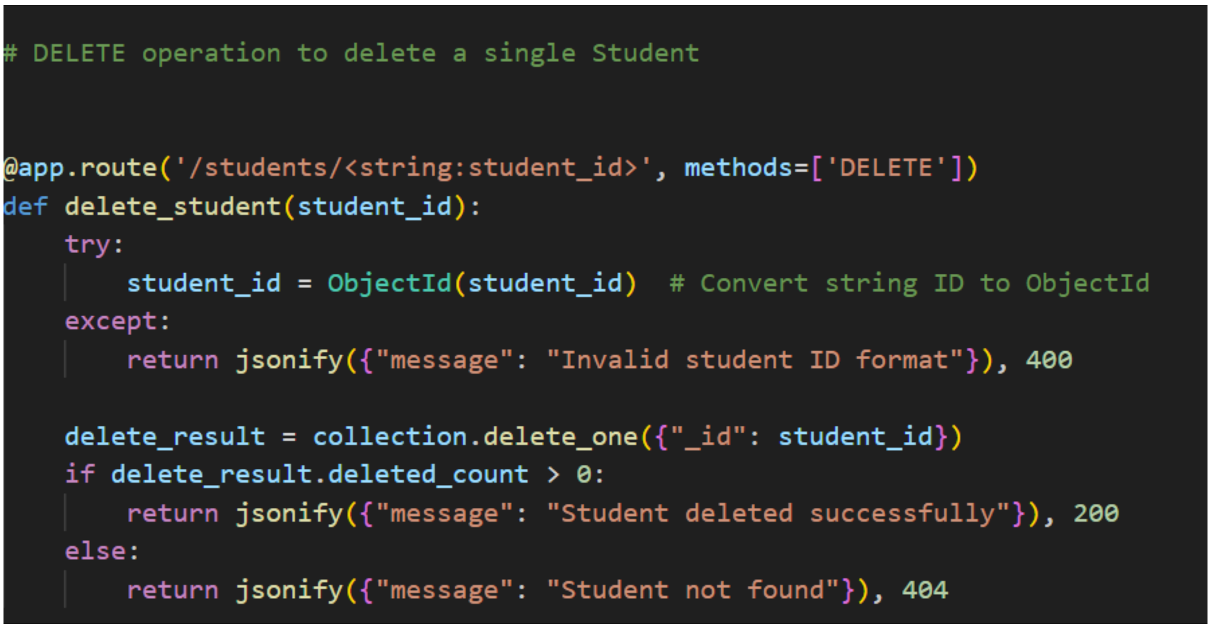
Body Cookies Headers (5) Test Results — Status: 200 OK Time: 133 ms Size: 213 B

```
{
    "message": "Student updated successfully"
}
```

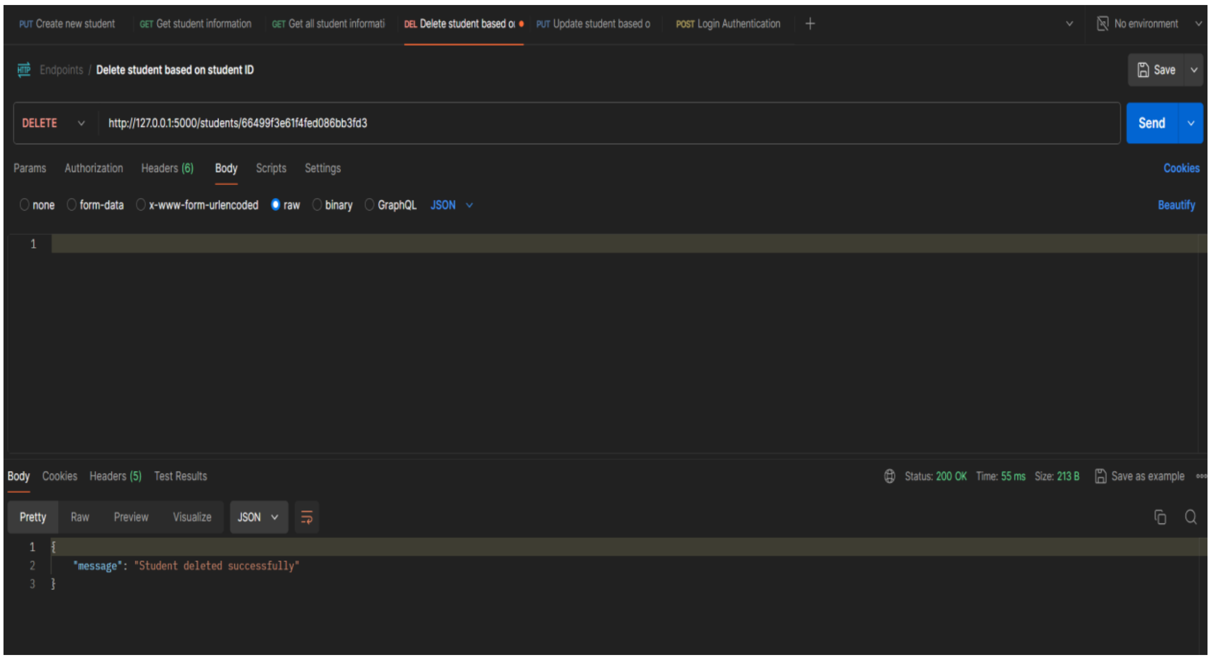


```
_id: ObjectId('66499f3e61f4fed086bb3fcb')
roll number : 1
name : "John Doe"
dob : "2005-05-15T00:00:00.000Z"
class : "10"
section : "B"
classteacher : "Mr. Smith"
active : true
fee : "unpaid"
```

Deleting student information for a single student based on the student ID.



```python
# DELETE operation to delete a single Student

@app.route('/students/<string:student_id>', methods=['DELETE'])
def delete_student(student_id):
    try:
        student_id = ObjectId(student_id)  # Convert string ID to ObjectId
    except:
        return jsonify({"message": "Invalid student ID format"}), 400

    delete_result = collection.delete_one({"_id": student_id})
    if delete_result.deleted_count > 0:
        return jsonify({"message": "Student deleted successfully"}), 200
    else:
        return jsonify({"message": "Student not found"}), 404
```

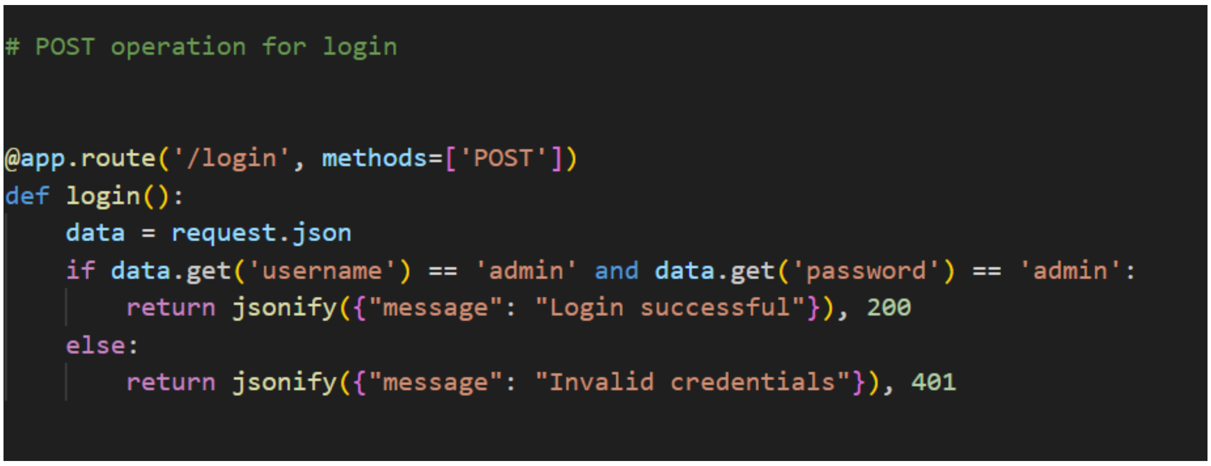


```
DELETE  http://127.0.0.1:5000/students/66499f3e61f4fed086bb3fd3
```

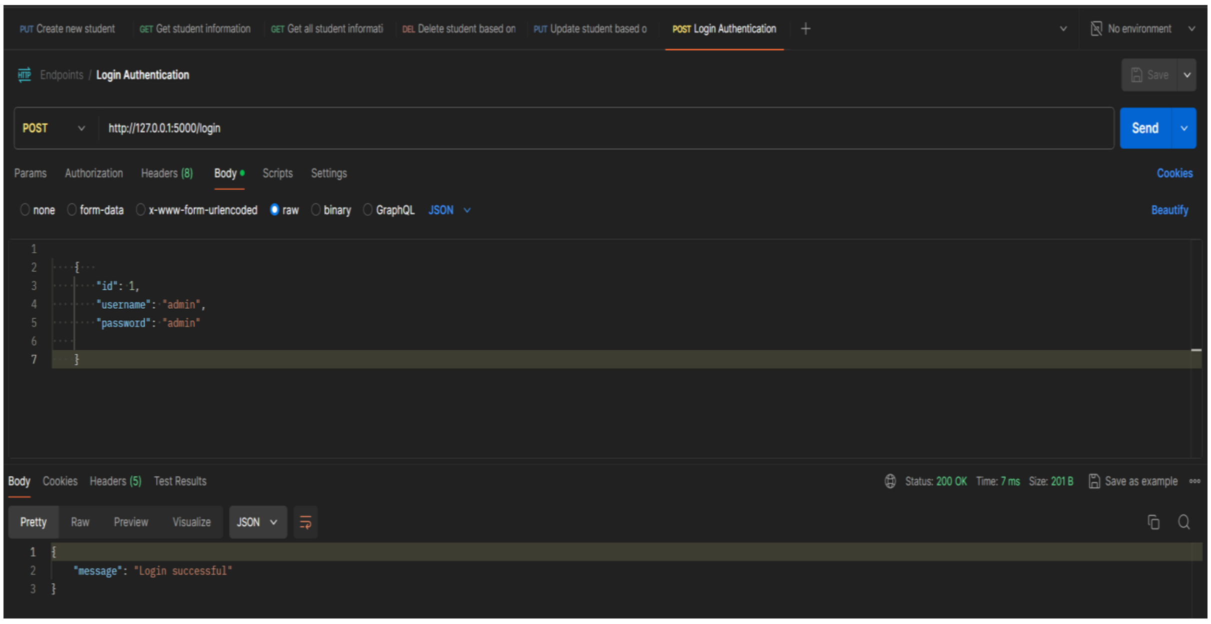
```
Status: 200 OK  Time: 55 ms  Size: 213 B
```

```json
{
    "message": "Student deleted successfully"
}
```

User login with user ID and password verification.



```python
# POST operation for login

@app.route('/login', methods=['POST'])
def login():
    data = request.json
    if data.get('username') == 'admin' and data.get('password') == 'admin':
        return jsonify({"message": "Login successful"}), 200
    else:
        return jsonify({"message": "Invalid credentials"}), 401
```



POST http://127.0.0.1:5000/login

Body: none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```

    {
        "id": 1,
        "username": "admin",
        "password": "admin"

    }
```

Status: 200 OK Time: 7 ms Size: 201 B

```
{
    "message": "Login successful"
}
```

API Documentation :

https://documenter.getpostman.com/view/29049633/2sA3QmCuPE