EXPERIMENT NO. 7 - MongoDB

Name of Student	**Rohan Lalchandani**
Class Roll No	**25**
D.O.P.	**13/03/2025**
D.O.S.	**20/03/2025**
Sign and Grade	

AIM : To study CRUD operations in MongoDB

OVERVIEW OF TASKS PERFORMED :

The experiment involves creating a **student database** for the IT department with fields **Name, Roll No, and Class Name**. A single student record was inserted, followed by multiple student entries at once. Queries were performed to **filter students by class**, retrieve students with a **specific roll number**, update a student's roll number, and delete a specific student's entry.

Additionally, **RESTful APIs** were implemented using **Node.js, Express, and Mongoose** to manage student data. The server was connected to **MongoDB**, and endpoints were created to **retrieve all students, get details of a student by ID, add a new student, update student details, and delete a student by ID**. The student schema included attributes **name, age, and grade** for data storage.

GITHUB LINK –

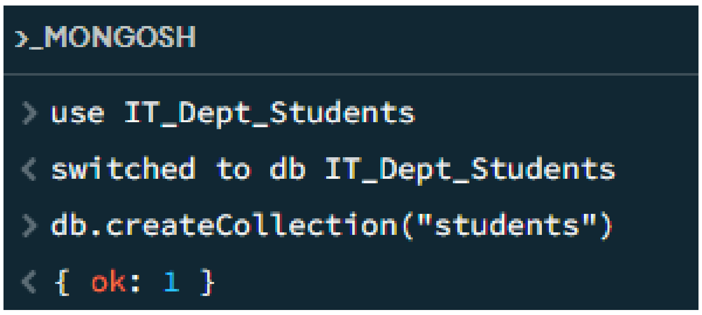
https://github.com/Rohan-Lalchandani08/WebX_Lab/tree/main/WebX%20Exp%207

OUTPUT

Step 1: Use or Create a Database and Create and Use a Collection (e.g., students)

use IT_Dept_Students

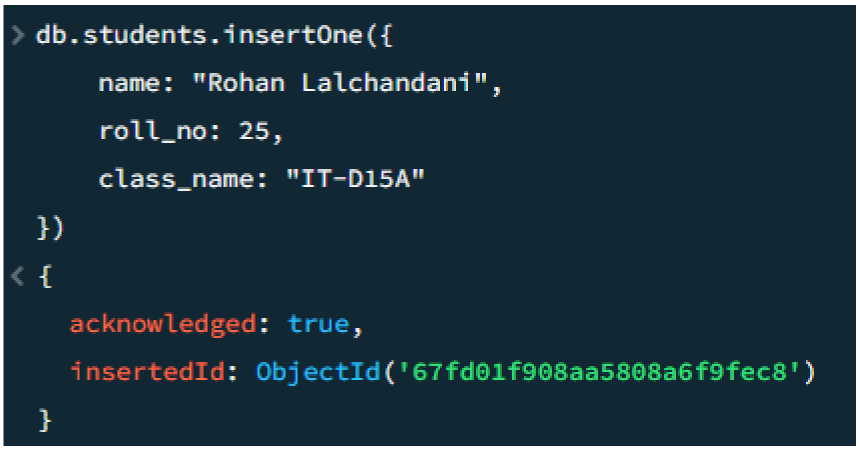
db.createCollection("students")



```
>_MONGOSH
> use IT_Dept_Students
< switched to db IT_Dept_Students
> db.createCollection("students")
< { ok: 1 }
```

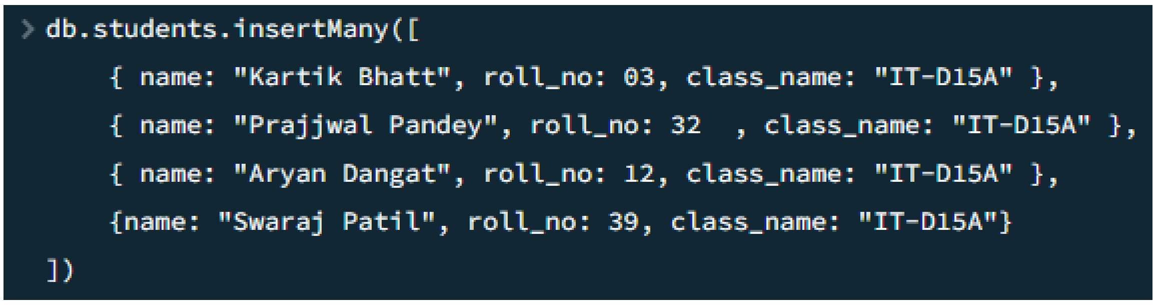
Step 2:

a) **Insert One Student Detail**



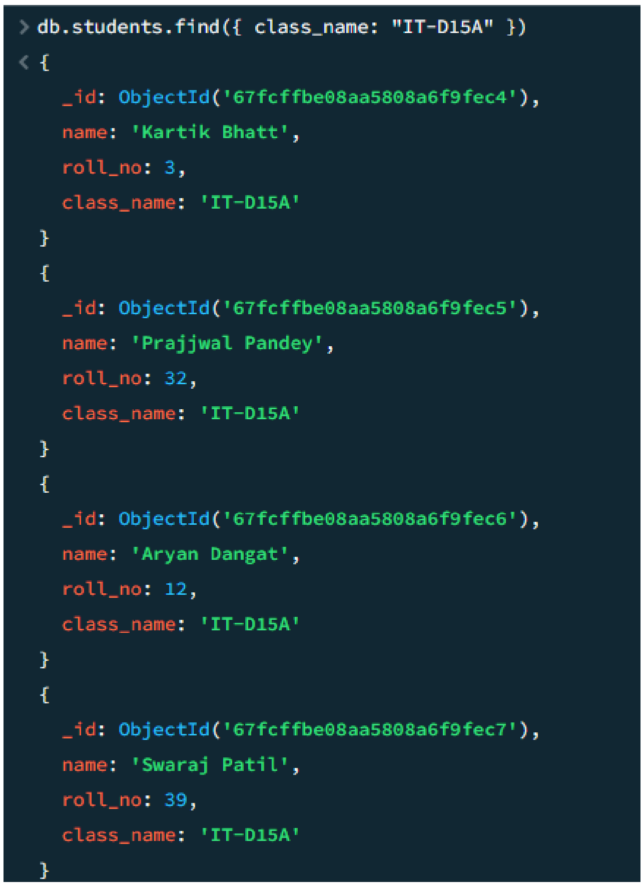
```
> db.students.insertOne({
      name: "Rohan Lalchandani",
      roll_no: 25,
      class_name: "IT-D15A"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('67fd01f908aa5808a6f9fec8')
  }
```

b) **Insert Multiple Student Details at Once**



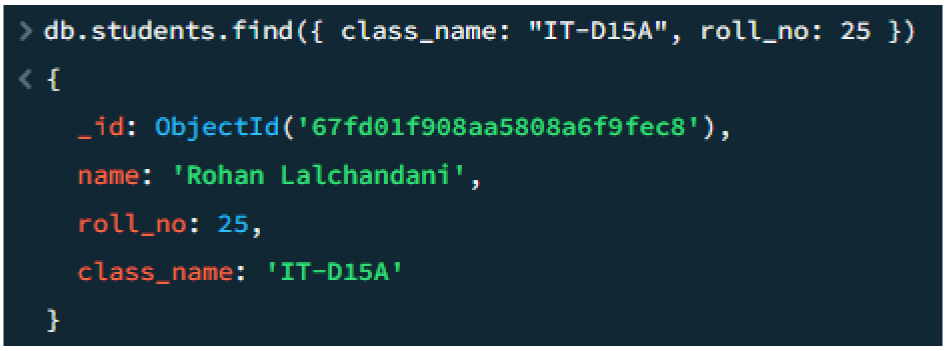
```
> db.students.insertMany([
      { name: "Kartik Bhatt", roll_no: 03, class_name: "IT-D15A" },
      { name: "Prajjwal Pandey", roll_no: 32  , class_name: "IT-D15A" },
      { name: "Aryan Dangat", roll_no: 12, class_name: "IT-D15A" },
      {name: "Swaraj Patil", roll_no: 39, class_name: "IT-D15A"}
  ])
```

c) **Display Students of a Particular Class:**



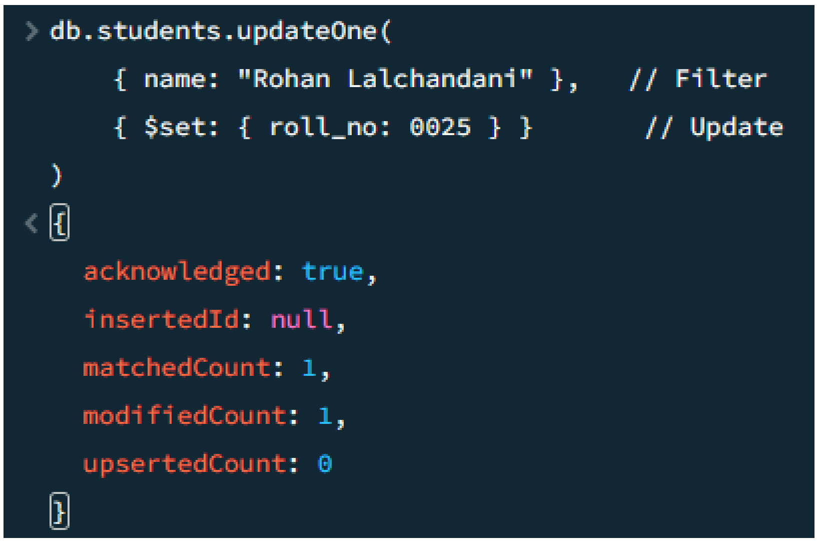
```
> db.students.find({ class_name: "IT-D15A" })
< {
    _id: ObjectId('67fcffbe08aa5808a6f9fec4'),
    name: 'Kartik Bhatt',
    roll_no: 3,
    class_name: 'IT-D15A'
  }
  {
    _id: ObjectId('67fcffbe08aa5808a6f9fec5'),
    name: 'Prajjwal Pandey',
    roll_no: 32,
    class_name: 'IT-D15A'
  }
  {
    _id: ObjectId('67fcffbe08aa5808a6f9fec6'),
    name: 'Aryan Dangat',
    roll_no: 12,
    class_name: 'IT-D15A'
  }
  {
    _id: ObjectId('67fcffbe08aa5808a6f9fec7'),
    name: 'Swaraj Patil',
    roll_no: 39,
    class_name: 'IT-D15A'
  }
```

d) Display Students of a Specific Roll No in a Class



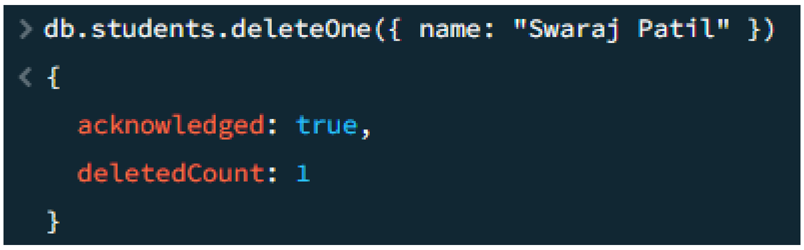
```
> db.students.find({ class_name: "IT-D15A", roll_no: 25 })
< {
    _id: ObjectId('67fd01f908aa5808a6f9fec8'),
    name: 'Rohan Lalchandani',
    roll_no: 25,
    class_name: 'IT-D15A'
  }
```

e) Change the Roll No of a Student



```
> db.students.updateOne(
      { name: "Rohan Lalchandani" },    // Filter
      { $set: { roll_no: 0025 } }        // Update
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

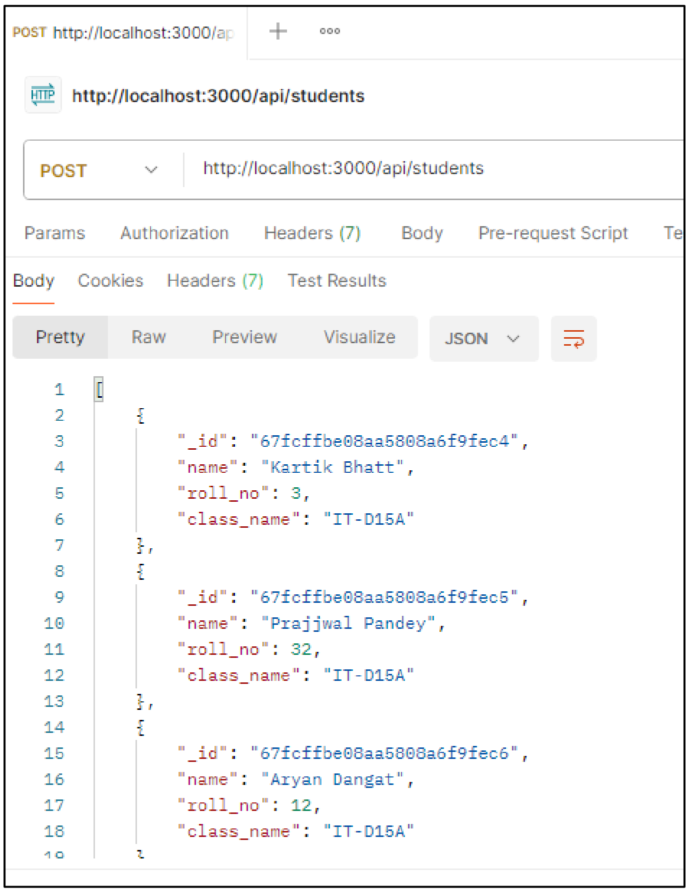
f) Delete Entry of a Particular Student



```
> db.students.deleteOne({ name: "Swaraj Patil" })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Restful API:

a. Retrieve a list of all students.



POST http://localhost:3000/ap

http://localhost:3000/api/students

POST http://localhost:3000/api/students

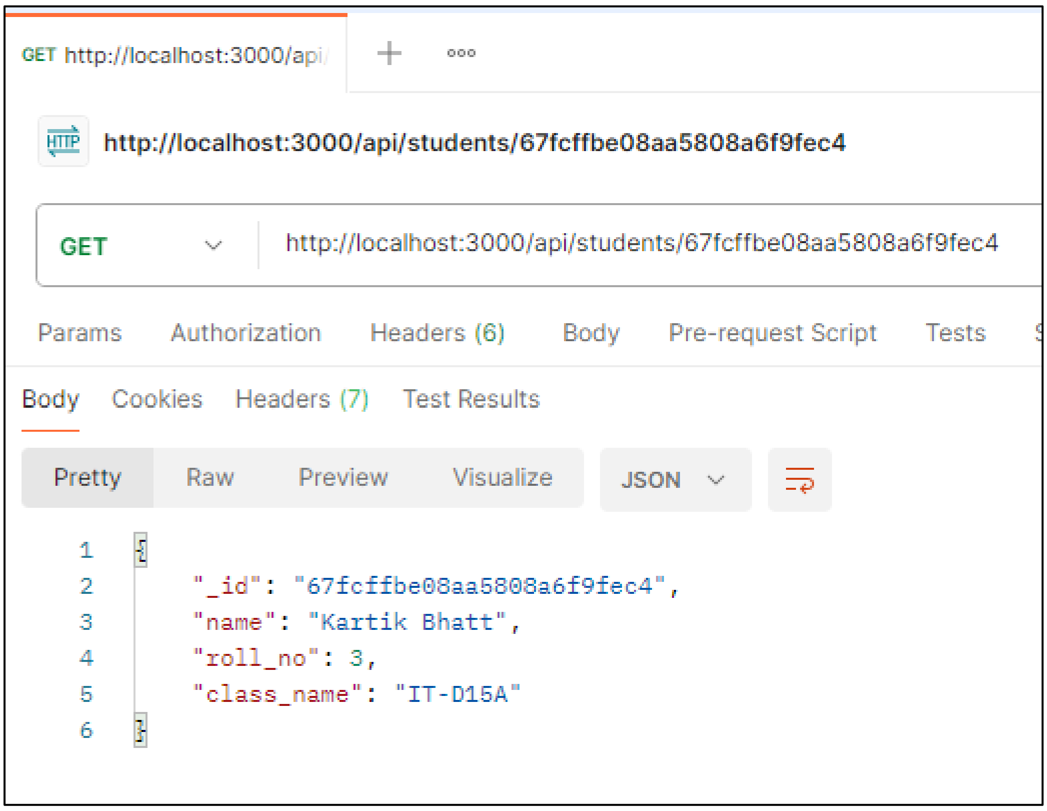
Params Authorization Headers (7) Body Pre-request Script Te

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
[
    {
        "_id": "67fcffbe08aa5808a6f9fec4",
        "name": "Kartik Bhatt",
        "roll_no": 3,
        "class_name": "IT-D15A"
    },
    {
        "_id": "67fcffbe08aa5808a6f9fec5",
        "name": "Prajjwal Pandey",
        "roll_no": 32,
        "class_name": "IT-D15A"
    },
    {
        "_id": "67fcffbe08aa5808a6f9fec6",
        "name": "Aryan Dangat",
        "roll_no": 12,
        "class_name": "IT-D15A"
```

b. Retrieve details of an individual student by ID.



GET http://localhost:3000/api/

http://localhost:3000/api/students/67fcffbe08aa5808a6f9fec4

GET http://localhost:3000/api/students/67fcffbe08aa5808a6f9fec4

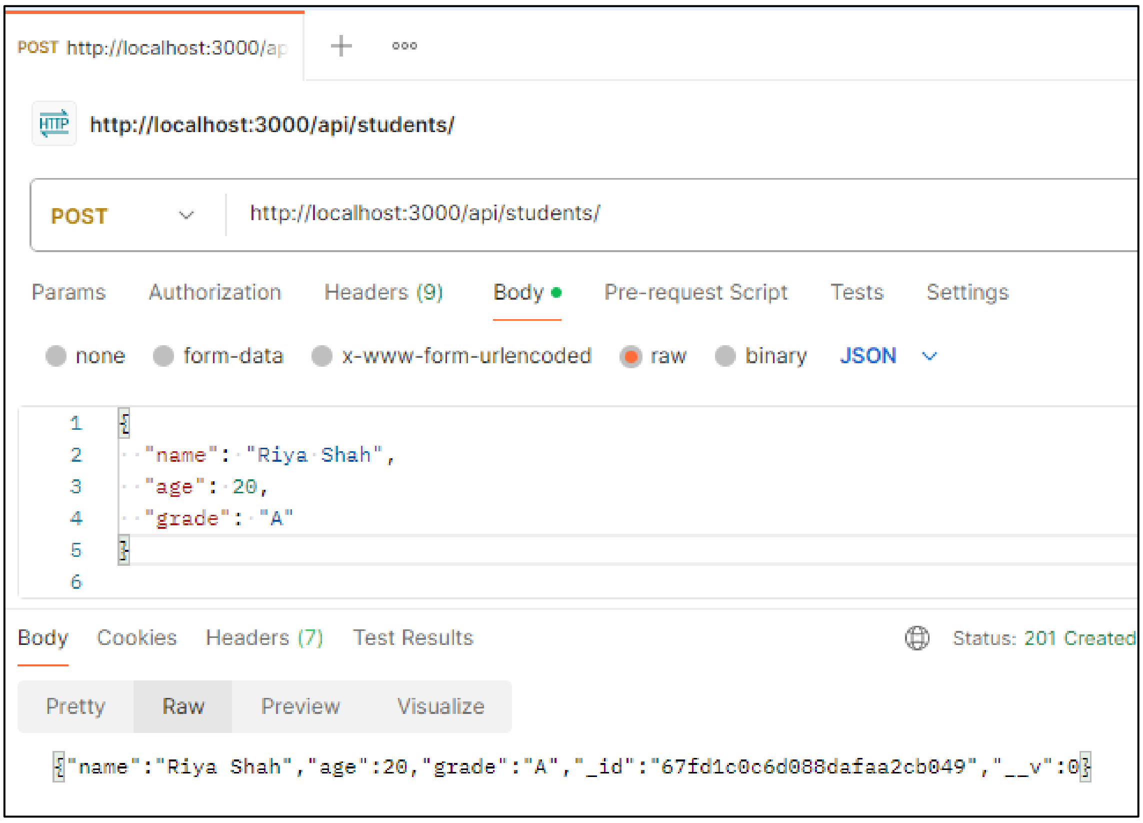
Params Authorization Headers (6) Body Pre-request Script Tests

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "_id": "67fcffbe08aa5808a6f9fec4",
    "name": "Kartik Bhatt",
    "roll_no": 3,
    "class_name": "IT-D15A"
}
```

c. Add a new student to the database.


POST http://localhost:3000/ap

http://localhost:3000/api/students/

POST http://localhost:3000/api/students/

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

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

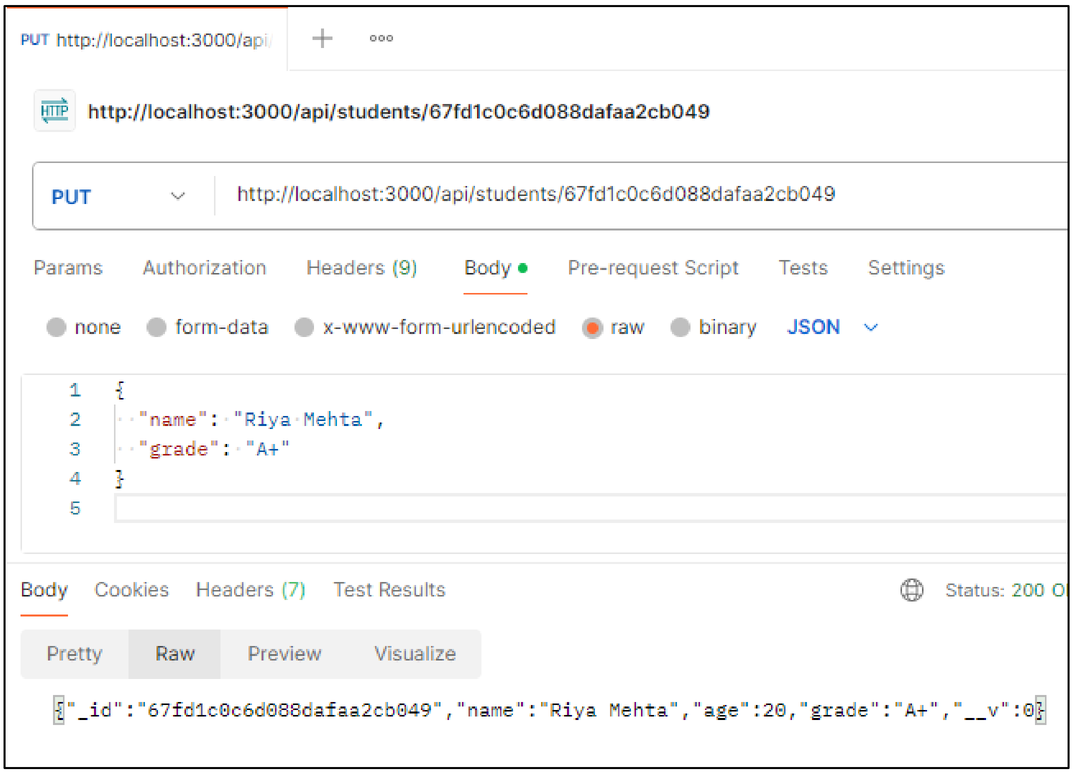
```
{
  "name": "Riya Shah",
  "age": 20,
  "grade": "A"
}

```

Body Cookies Headers (7) Test Results Status: 201 Created

Pretty Raw Preview Visualize

```
{"name":"Riya Shah","age":20,"grade":"A","_id":"67fd1c0c6d088dafaa2cb049","__v":0}
```

d. Update details of an existing student by ID.


PUT http://localhost:3000/api

http://localhost:3000/api/students/67fd1c0c6d088dafaa2cb049

PUT http://localhost:3000/api/students/67fd1c0c6d088dafaa2cb049

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

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

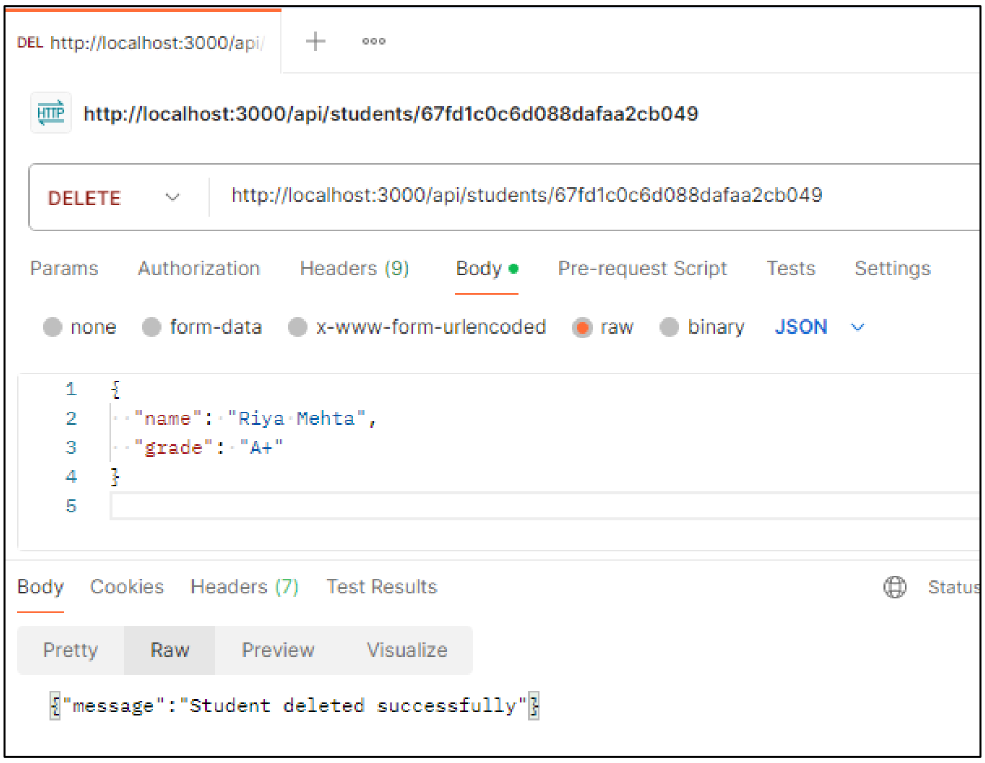
```
{
  "name": "Riya Mehta",
  "grade": "A+"
}

```

Body Cookies Headers (7) Test Results Status: 200 O

Pretty Raw Preview Visualize

```
{"_id":"67fd1c0c6d088dafaa2cb049","name":"Riya Mehta","age":20,"grade":"A+","__v":0}
```

e. Delete a student from the database by ID.



DELETE http://localhost:3000/api/students/67fd1c0c6d088dafaa2cb049

```
{
  "name": "Riya Mehta",
  "grade": "A+"
}

```

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

CONCLUSION

In this experiment, we successfully performed CRUD operations in **MongoDB** and implemented a **RESTful API** using **Node.js, Express, and Mongoose**. We learned how to create, read, update, and delete student records both via **MongoDB shell commands** and **API endpoints**.