

EXPERIMENT NO. 7 - MongoDB

Name of Student	Anuprita Mhapankar
Class Roll No	D15A_28
D.O.P.	13/02/2025
D.O.S.	20/03/2025
Sign and Grade	

AIM: To study CRUD operations in MongoDB

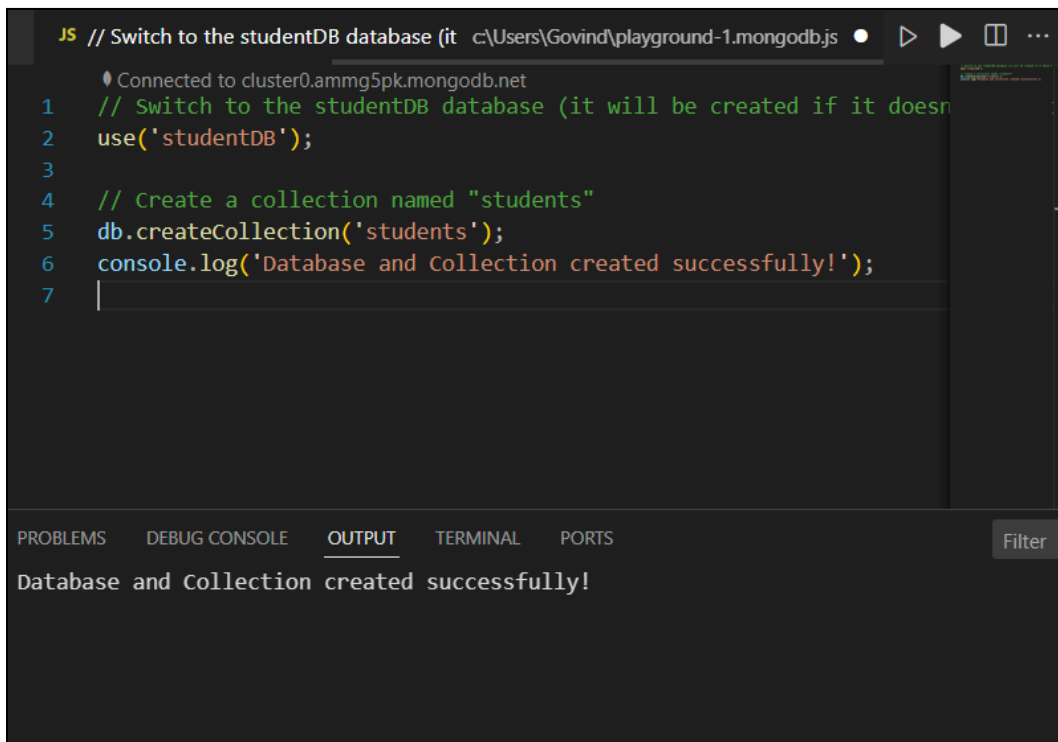
PROBLEM STATEMENT:

1. Create a new database to storage student details of IT dept(Name, Roll no, class name) and perform the following on the database
 - a. Insert one student details
 - b. Insert at once multiple student details
 - c. Display student for a particular class
 - d. Display students of specific roll no in a class
 - e. Change the roll no of a student
 - f. Delete entries of particular student
2. Create a set of RESTful endpoints using Node.js, Express, and Mongoose for handling student data operations. The endpoints should support:
 - a. Retrieve a list of all students.
 - b. Retrieve details of an individual student by ID.
 - c. Add a new student to the database.
 - d. Update details of an existing student by ID.
 - e. Delete a student from the database by ID.Connect the server to MongoDB using Mongoose, and store student data with attributes: name, age, and grade.

GITHUB LINK: https://github.com/Anuprita2022-26/WebX_Exp7

OUTPUT:

1. Create a database to store student details of IT Department

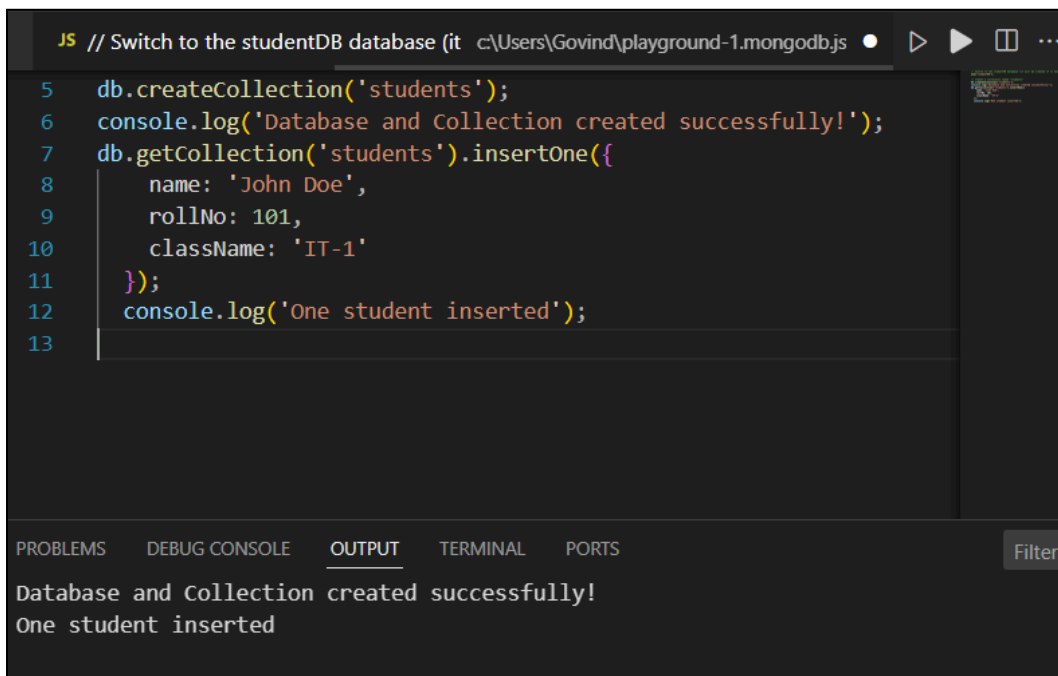


The screenshot shows a VS Code editor with a JavaScript file named `mongodb.js`. The code connects to a MongoDB cluster and creates a database named `studentDB` and a collection named `students`. The output window at the bottom shows the message: "Database and Collection created successfully!".

```
JS // Switch to the studentDB database (it c:\Users\Govind\playground-1.mongodb.js
Connected to cluster0.ammg5pk.mongodb.net
1 // Switch to the studentDB database (it will be created if it doesn't exist)
2 use('studentDB');
3
4 // Create a collection named "students"
5 db.createCollection('students');
6 console.log('Database and Collection created successfully!');
7
```

Database and Collection created successfully!

a) Insert one Student Detail



The screenshot shows the same VS Code editor with the `mongodb.js` file. The code now includes an `insertOne` operation to add a student record. The output window shows two messages: "Database and Collection created successfully!" and "One student inserted".

```
JS // Switch to the studentDB database (it c:\Users\Govind\playground-1.mongodb.js
db.createCollection('students');
console.log('Database and Collection created successfully!');
7 db.getCollection('students').insertOne({
8   name: 'John Doe',
9   rollNo: 101,
10  className: 'IT-1'
11 });
12 console.log('One student inserted');
13
```

Database and Collection created successfully!
One student inserted

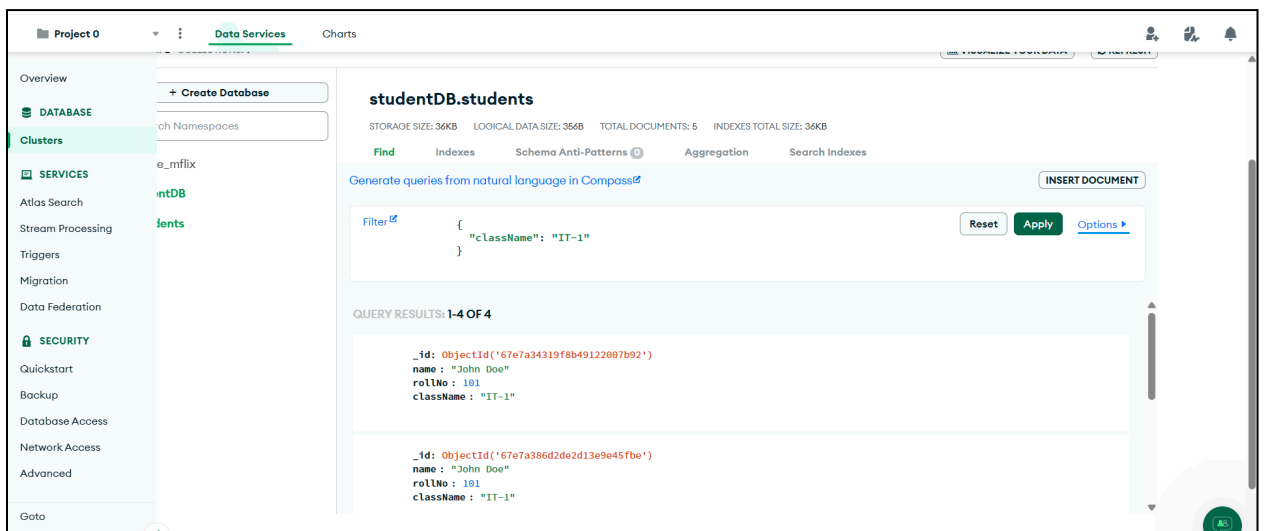
b) Insert at once multiple Student Details

```
JS // Switch to the studentDB database (it c:\Users\Govind\playground-1.mongodbs
12 console.log('One student inserted');
13 db.getCollection('students').insertMany([
14   { name: 'Alice', rollNo: 102, className: 'IT-1' },
15   { name: 'Bob', rollNo: 103, className: 'IT-2' },
16   { name: 'Charlie', rollNo: 104, className: 'IT-1' }
17 ]);
18 console.log('Multiple students inserted');
19
```

Database and Collection created successfully!
One student inserted
Multiple students inserted

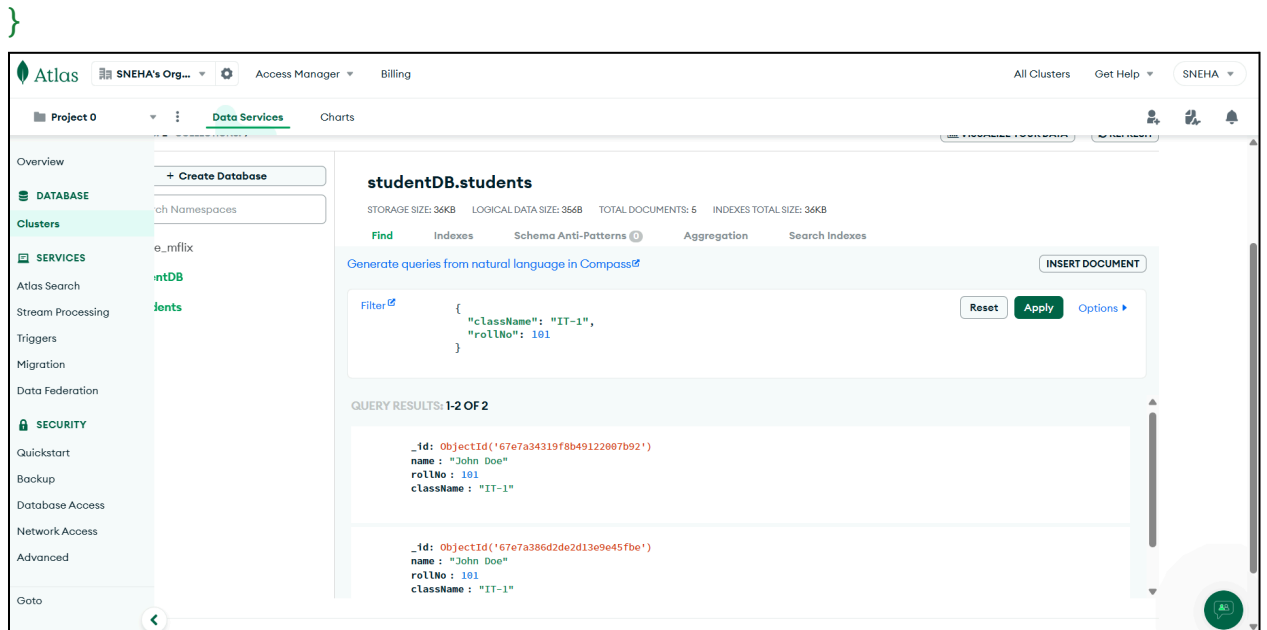
c) Display Students for a Particular Class

```
{
  "className": "IT-1"
}
```



d) Display Student of a Specific Roll Number in a Class

```
{
  "className": "IT-1",
  "rollNo": 101
}
```

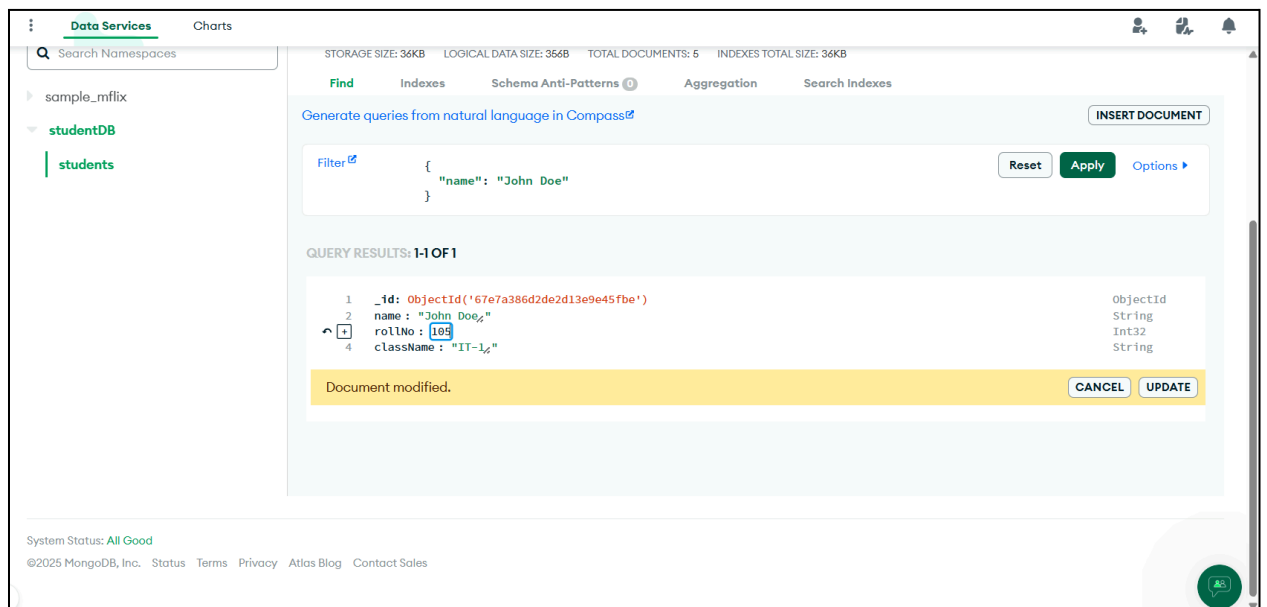


e) Change the Roll Number of a Student

Filter:

```
{
  "name": "John Doe"
}
```

Update:

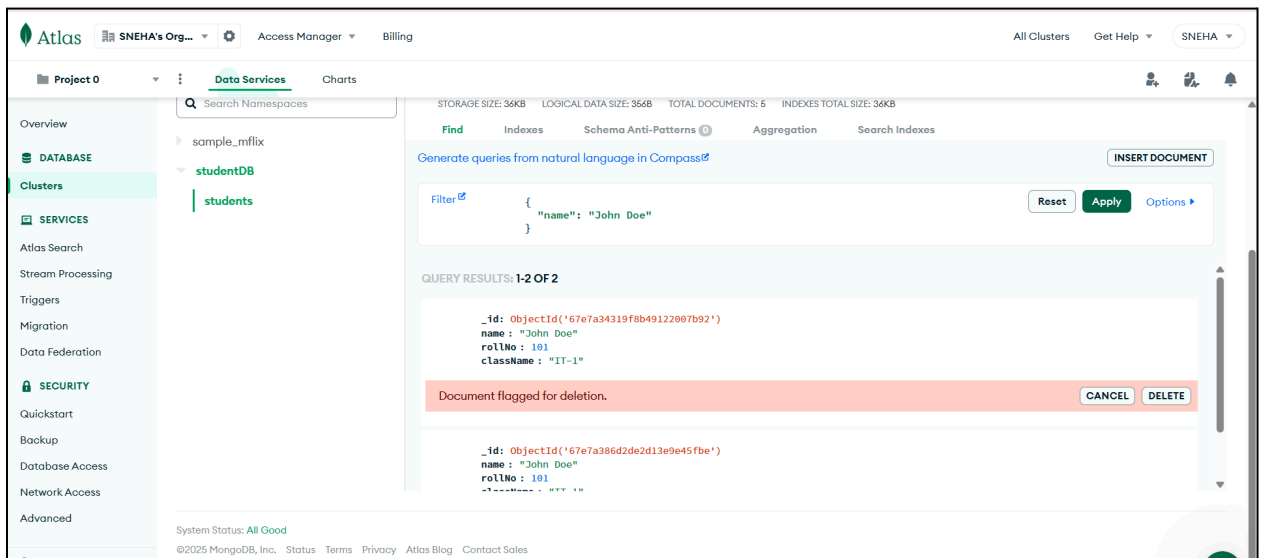


f) Delete Entries of a Particular Student

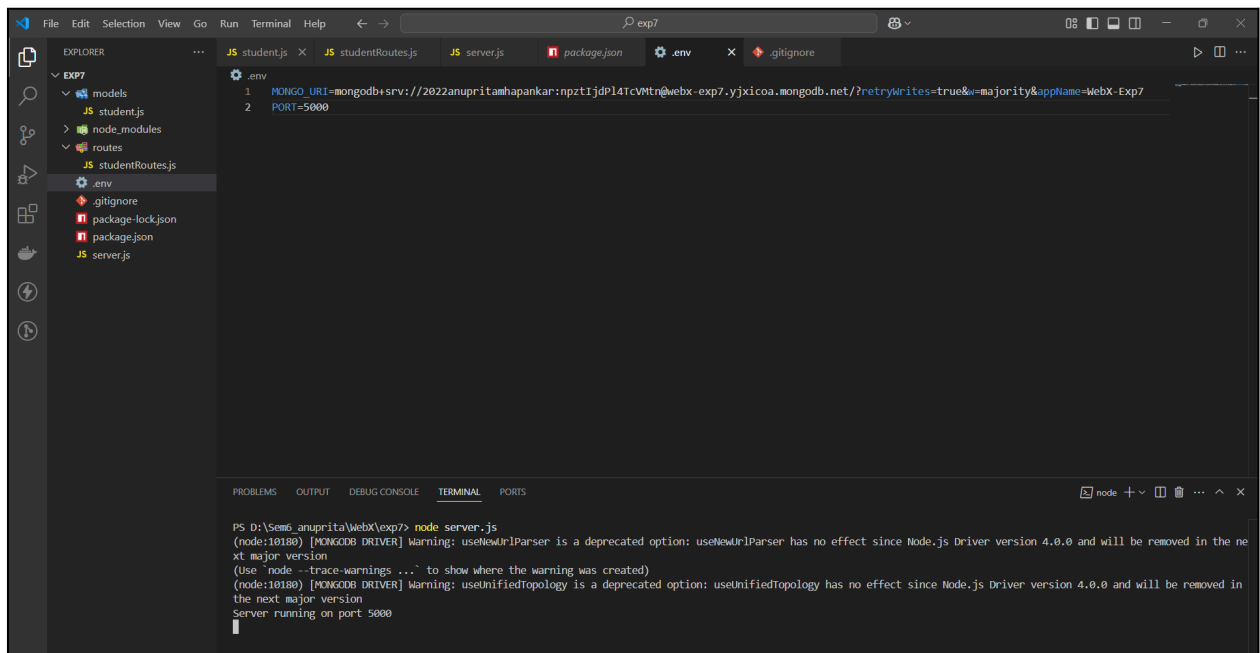
Filter:

```
{
  "name": "John Doe"
}
```

Delete:



2. Restful api



a) Retrieve details of an individual student by ID.

The screenshot displays a REST client interface with the following details:

- URL:** `http://localhost:3000/students/age/21`
- Method:** `GET`
- Status:** `200 OK` (14 ms, 327 B)
- Body:** JSON response showing student details:

```
1 {
2   "_id": "67e828cec4578f3b4cc5b1b1",
3   "name": "Anupriya Mhapankar",
4   "age": 21,
5   "grade": "A+",
6   "__v": 0
7 }
```

b) Add a new student to the database

The screenshot displays the Postman interface for a REST client. The top bar shows 'Workspaces' and 'More' menus, along with search, user, settings, and notification icons. The main workspace is titled 'http://localhost:3000/students'. The request method is set to 'POST' and the URL is 'http://localhost:3000/students'. The 'Body' tab is selected, showing a JSON payload:

```
{
  "name": "Anuprita Mhapankar",
  "age": 20,
  "grade": "A+"
}
```

. The response status is '201 Created' with a response time of '19 ms' and a size of '332 B'. The response body is shown in the 'Body' tab, displaying the JSON response:

```
{
  "name": "Anuprita Mhapankar",
  "age": 20,
  "grade": "A+",
  "_id": "67e82877c4578f3b4cc5b1ad",
  "__v": 0
}
```

. The bottom status bar includes icons for Postbot, Runner, Vault, and other utilities.

Workspaces ▾ More ▾

⏏ 🔍 👤 ⚙️ 🔔

Upgrade ▾

⏏

🔒 < http://localhost:3000/students > + ▾ No environment ▾

🗑️ http://localhost:3000/students Save ▾ Share </>

POST ▾ http://localhost:3000/students Send ▾

Params Auth Headers (9) Body ● Scripts Settings Cookies Beautify

raw ▾ JSON ▾

```
1 {
2   "name": "Anuprita Mhapankar",
3   "age": 20,
4   "grade": "A+"
5 }
6
```

Body ▾ ⌛

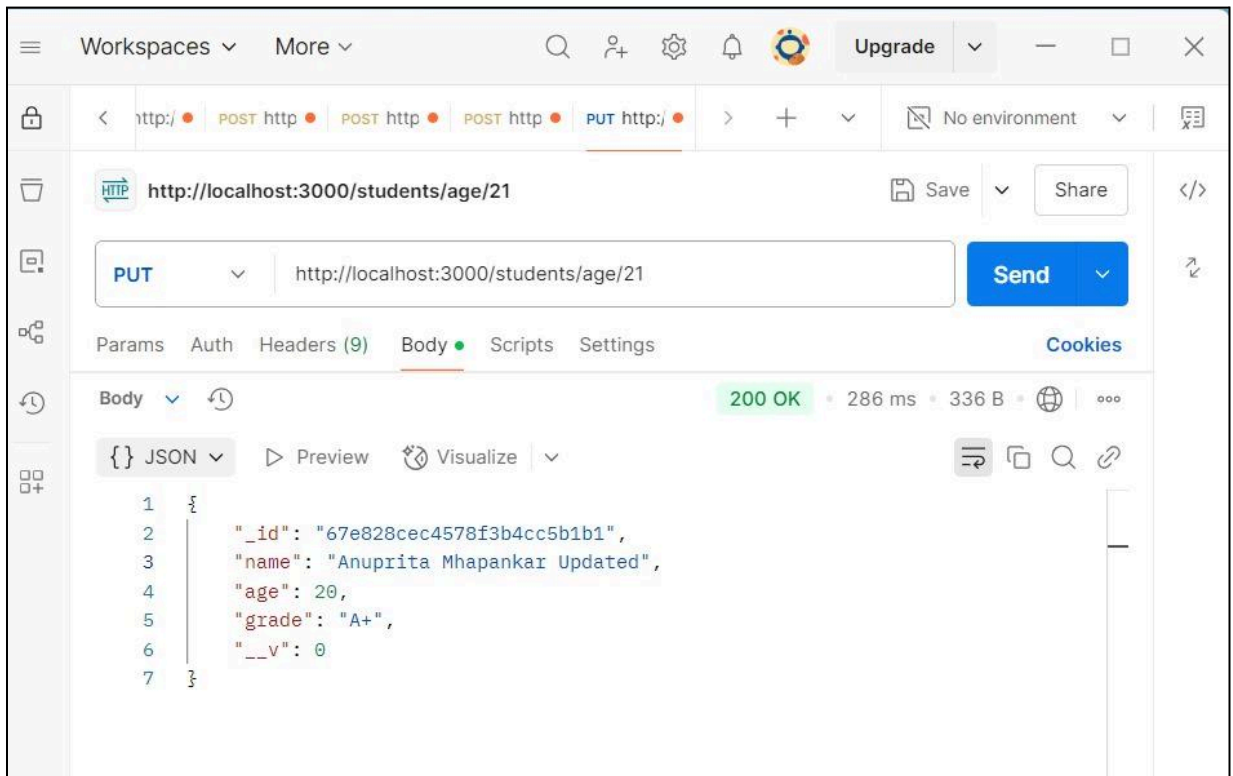
201 Created • 19 ms • 332 B • 🌐 ⋮

{ } JSON ▾ ▶ Preview ⌛ Visualize ▾

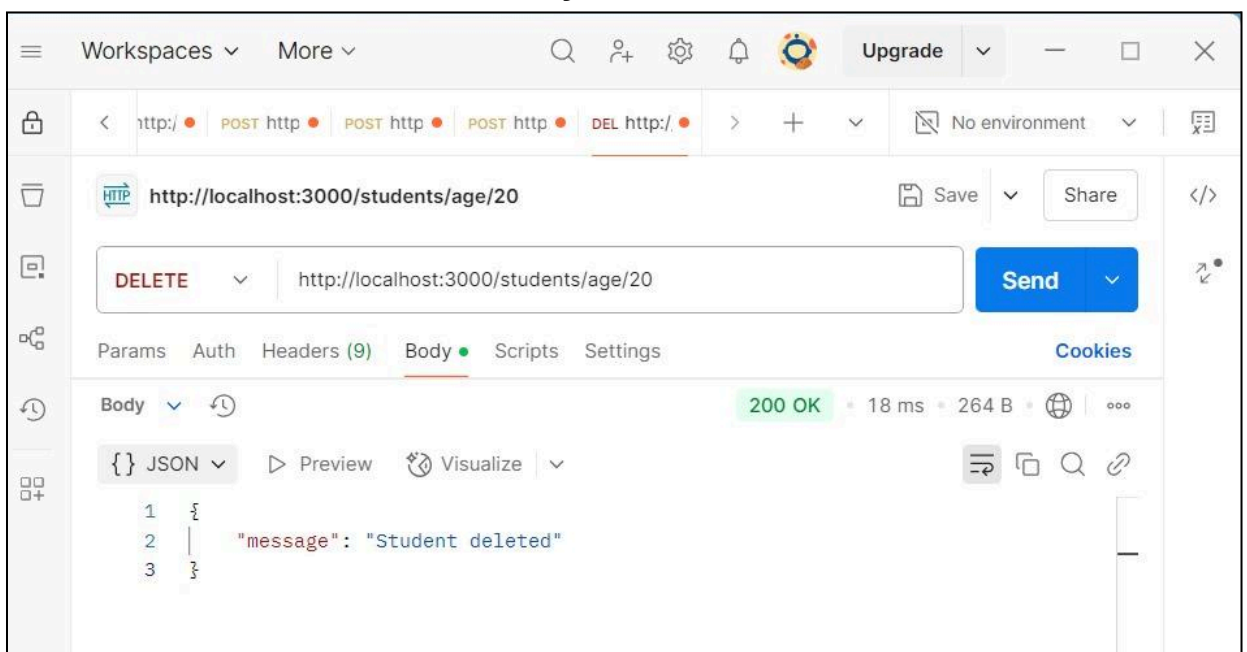
```
1 {
2   "name": "Anuprita Mhapankar",
3   "age": 20,
4   "grade": "A+",
5   "_id": "67e82877c4578f3b4cc5b1ad",
6   "__v": 0
7 }
```

📄 🔍 📄 Console Postbot ▶ Runner 🔔 🌐 Vault 🗑️ ⏏ ?

c) Update details of an existing student by ID.



d) Delete a student from the database by ID.



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**.