EXPERIMENT NO. 7 - MongoDB

Name of Student	**Riya Varyani**
Class Roll No	**D15A_61**
D.O.P.	
D.O.S.	
Sign and Grade	

AIM: To study CRUD operations in MongoDB

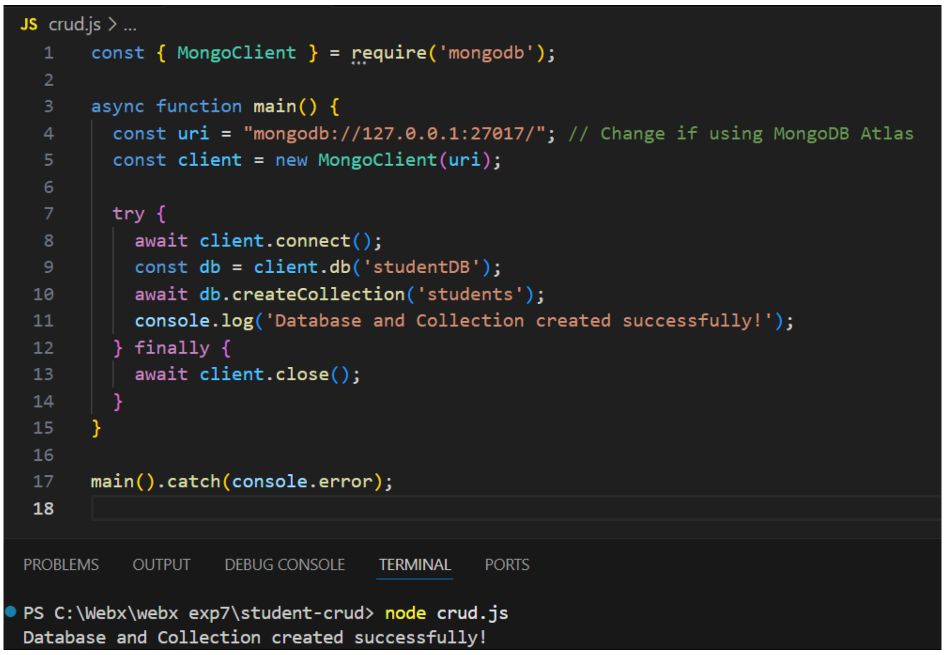
PROBLEM STATEMENT:

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.

OUTPUT:

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

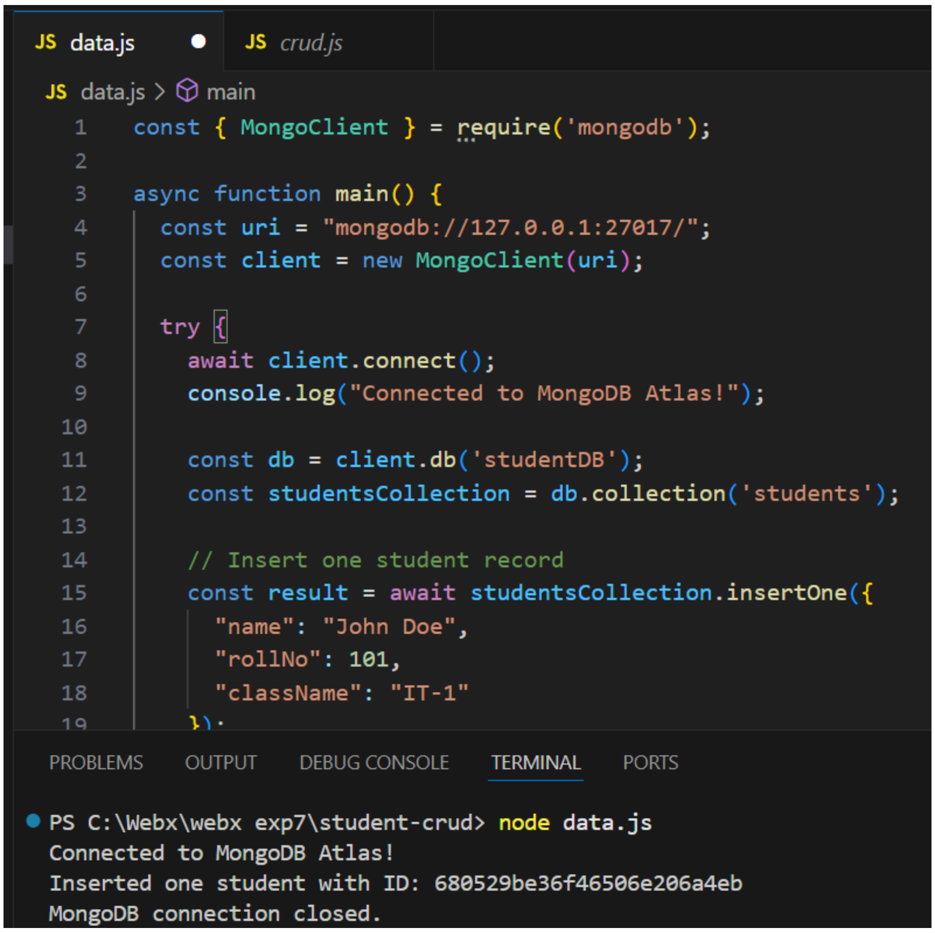


crud.js > ...

```js
const { MongoClient } = require('mongodb');

async function main() {
  const uri = "mongodb://127.0.0.1:27017/"; // Change if using MongoDB Atlas
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('studentDB');
    await db.createCollection('students');
    console.log('Database and Collection created successfully!');
  } finally {
    await client.close();
  }
}

main().catch(console.error);
```

```
PS C:\Webx\webx exp7\student-crud> node crud.js
Database and Collection created successfully!
```

a) **Insert one Student Detail**



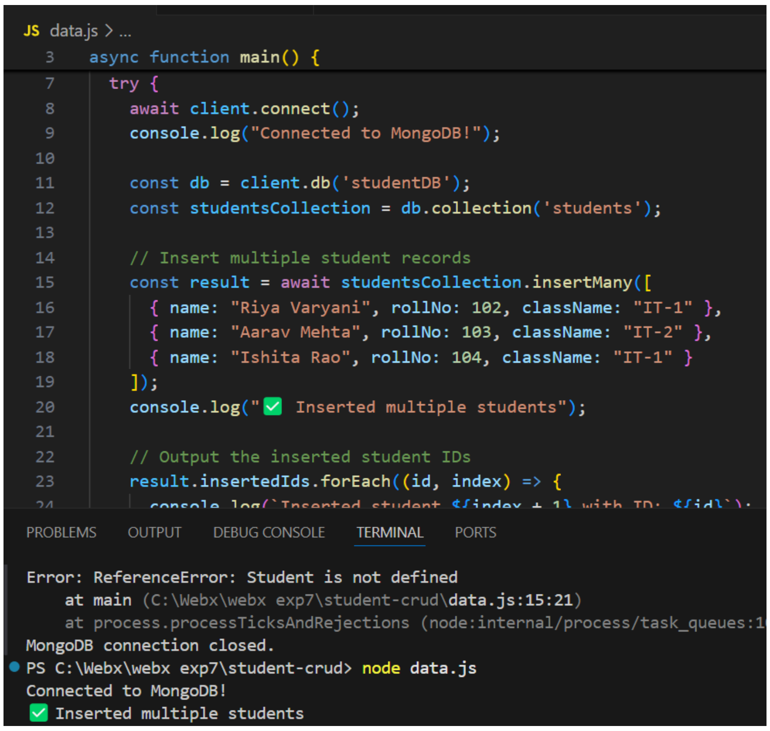
data.js   crud.js

data.js > main

```js
const { MongoClient } = require('mongodb');

async function main() {
  const uri = "mongodb://127.0.0.1:27017/";
  const client = new MongoClient(uri);

  try {
    await client.connect();
    console.log("Connected to MongoDB Atlas!");

    const db = client.db('studentDB');
    const studentsCollection = db.collection('students');

    // Insert one student record
    const result = await studentsCollection.insertOne({
      "name": "John Doe",
      "rollNo": 101,
      "className": "IT-1"
    });
```

```
PS C:\Webx\webx exp7\student-crud> node data.js
Connected to MongoDB Atlas!
Inserted one student with ID: 680529be36f46506e206a4eb
MongoDB connection closed.
```

b) **Insert at once multiple Student Details**

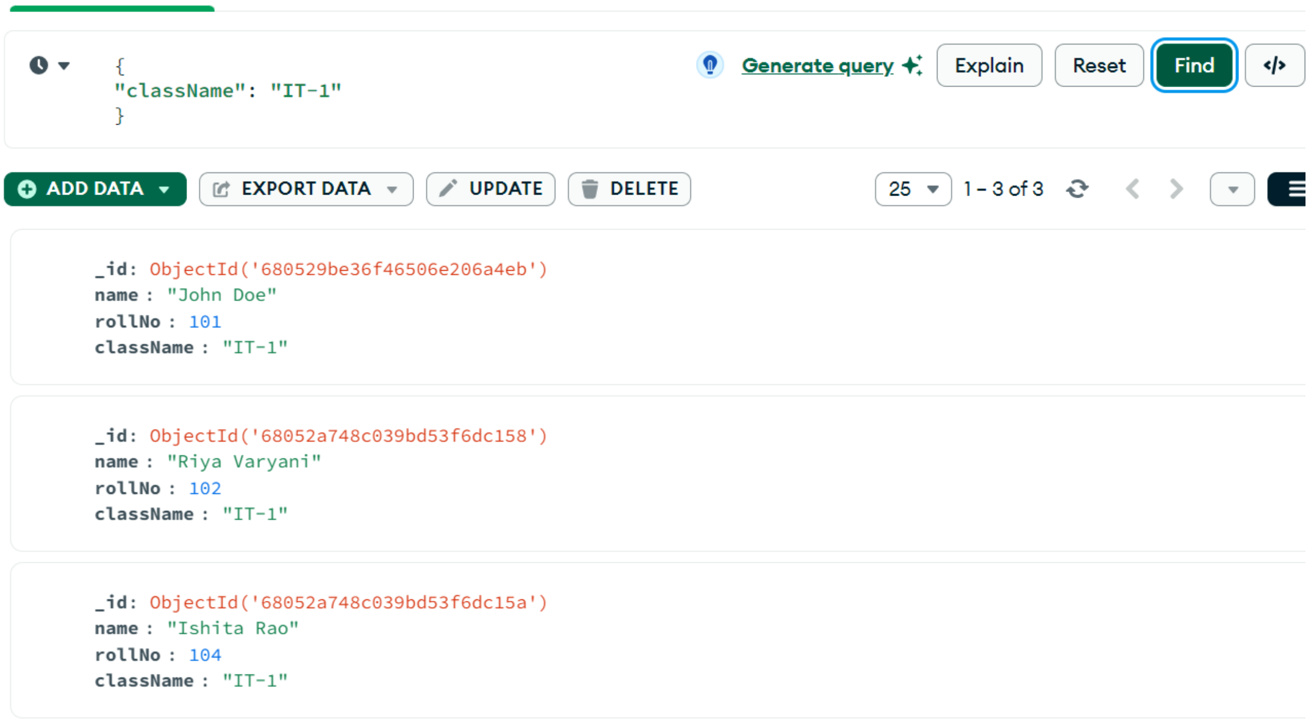


```
async function main() {
  try {
    await client.connect();
    console.log("Connected to MongoDB!");

    const db = client.db('studentDB');
    const studentsCollection = db.collection('students');

    // Insert multiple student records
    const result = await studentsCollection.insertMany([
      { name: "Riya Varyani", rollNo: 102, className: "IT-1" },
      { name: "Aarav Mehta", rollNo: 103, className: "IT-2" },
      { name: "Ishita Rao", rollNo: 104, className: "IT-1" }
    ]);
    console.log("✅ Inserted multiple students");

    // Output the inserted student IDs
    result.insertedIds.forEach((id, index) => {
      console.log(`Inserted student ${index + 1} with ID: ${id}`);
```

```
Error: ReferenceError: Student is not defined
    at main (C:\Webx\webx exp7\student-crud\data.js:15:21)
    at process.processTicksAndRejections (node:internal/process/task_queues:1
MongoDB connection closed.
PS C:\Webx\webx exp7\student-crud> node data.js
Connected to MongoDB!
✅ Inserted multiple students
```

c) Display Students for a Particular Class

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

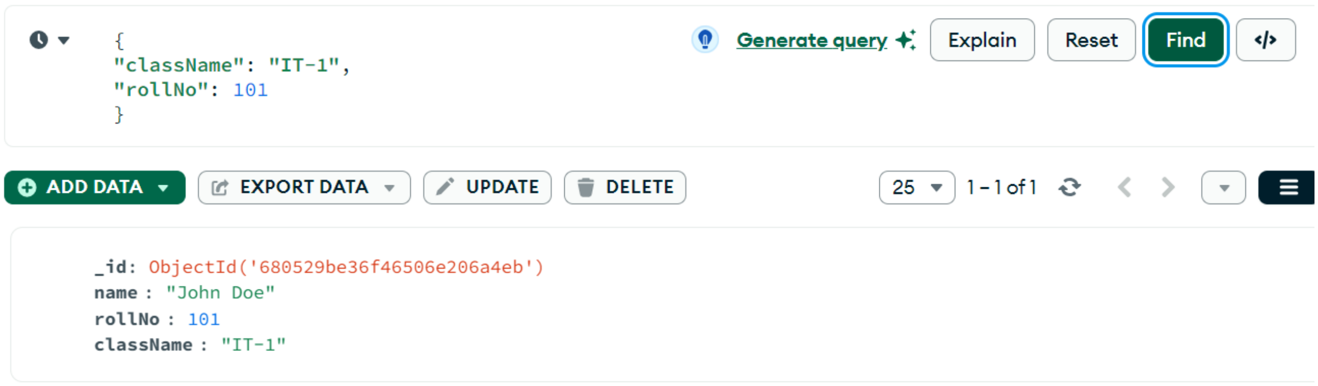


```
{
"className": "IT-1"
}
```

```
_id: ObjectId('680529be36f46506e206a4eb')
name : "John Doe"
rollNo : 101
className : "IT-1"

_id: ObjectId('68052a748c039bd53f6dc158')
name : "Riya Varyani"
rollNo : 102
className : "IT-1"

_id: ObjectId('68052a748c039bd53f6dc15a')
name : "Ishita Rao"
rollNo : 104
className : "IT-1"
```

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

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



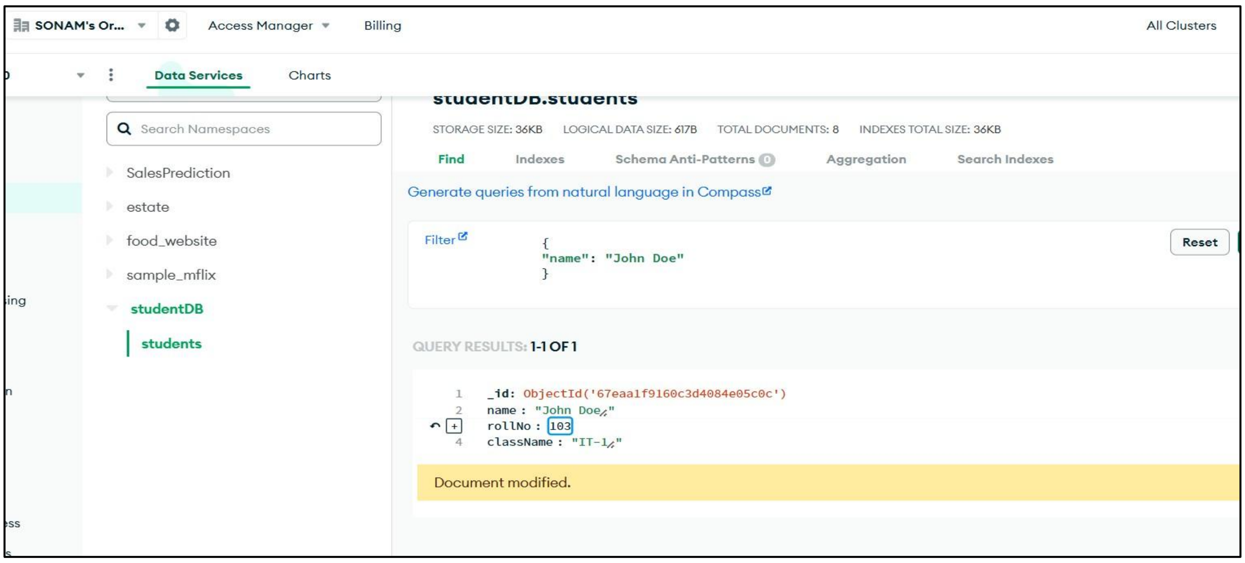
```
{
"className": "IT-1",
"rollNo": 101
}
```

```
_id: ObjectId('680529be36f46506e206a4eb')
name : "John Doe"
rollNo : 101
className : "IT-1"
```

e) Change the Roll Number of a Student Filter: {

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

Update:



studentDB.students

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 617B TOTAL DOCUMENTS: 8 INDEXES TOTAL SIZE: 36KB

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

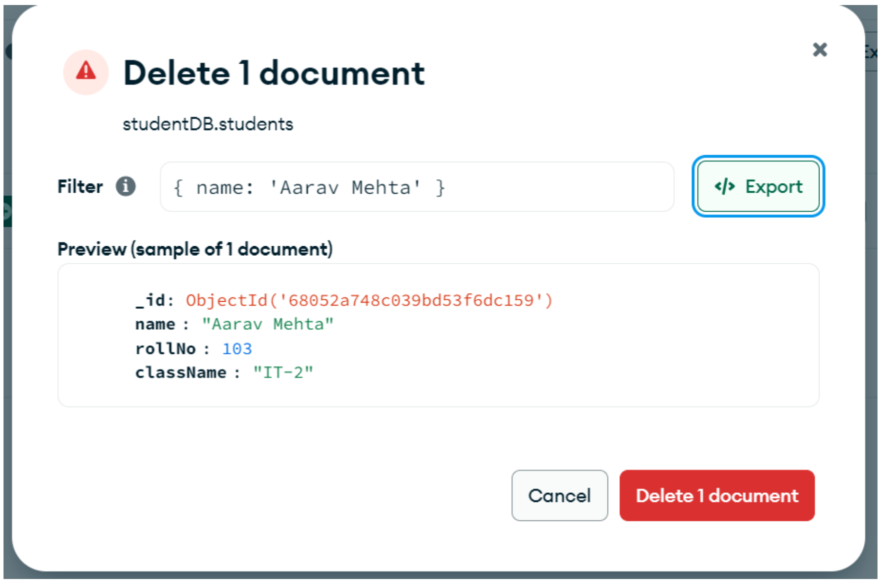
QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67eaa1f9160c3d4084e05c0c')
name : "John Doe"
rollNo : 103
className : "IT-1"
```

Document modified.

f) Delete Entries of a Particular Student Filter:

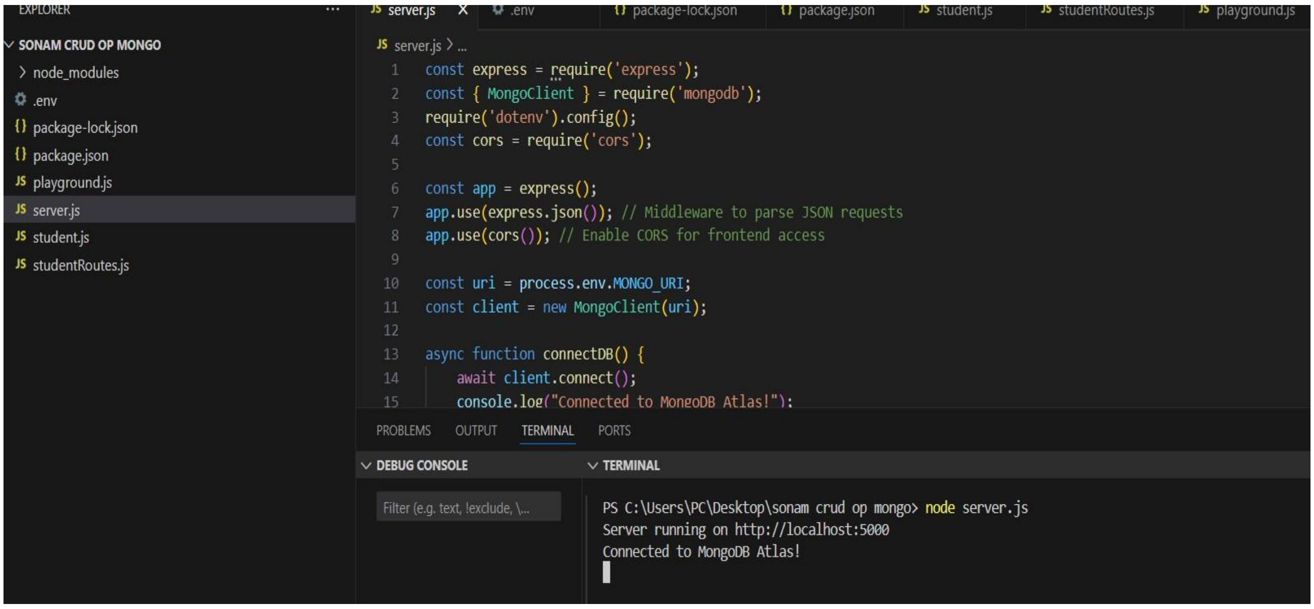
```
{
        "name": "Aarav Mehta "
}
```



```
{ name: 'Aarav Mehta' }
```

```
_id: ObjectId('68052a748c039bd53f6dc159')
name : "Aarav Mehta"
rollNo : 103
className : "IT-2"
```

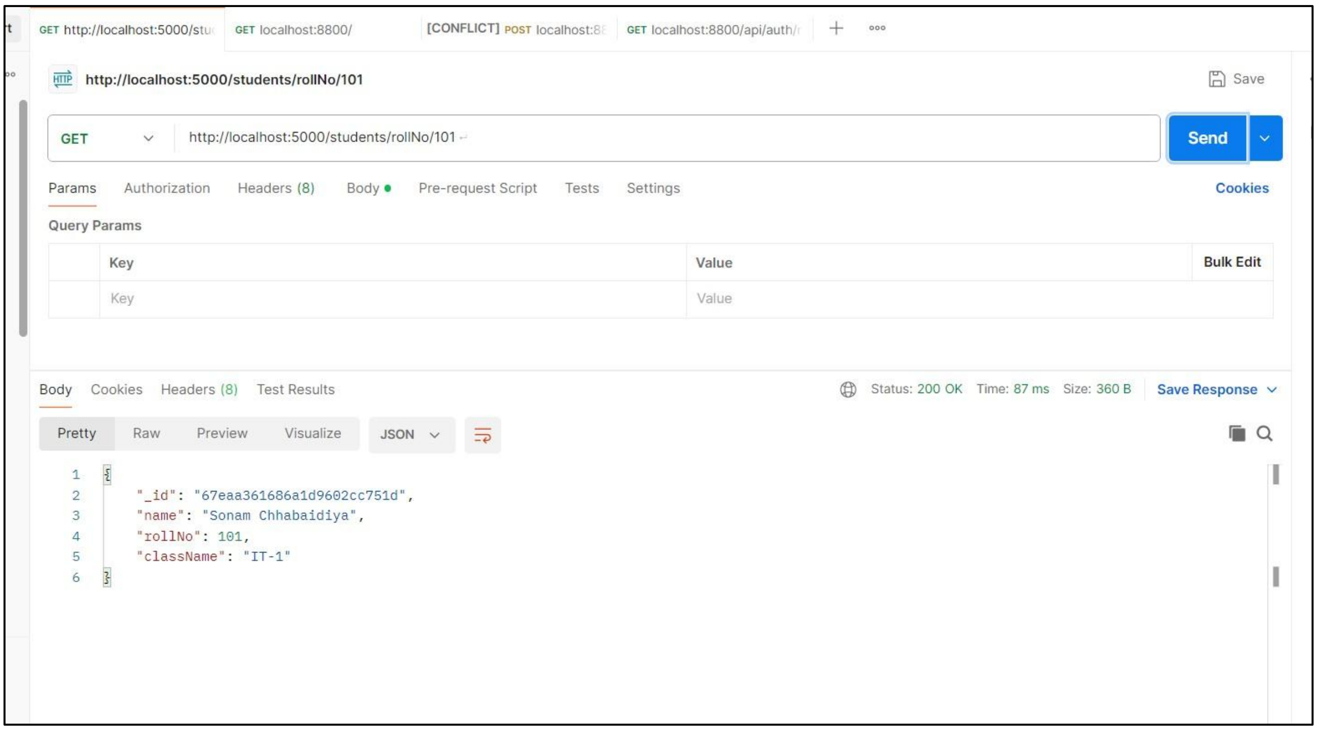
2. Restful api



```
const express = require('express');
const { MongoClient } = require('mongodb');
require('dotenv').config();
const cors = require('cors');

const app = express();
app.use(express.json()); // Middleware to parse JSON requests
app.use(cors()); // Enable CORS for frontend access

const uri = process.env.MONGO_URI;
const client = new MongoClient(uri);

async function connectDB() {
    await client.connect();
    console.log("Connected to MongoDB Atlas!");
```

```
PS C:\Users\PC\Desktop\sonam crud op mongo> node server.js
Server running on http://localhost:5000
Connected to MongoDB Atlas!
```

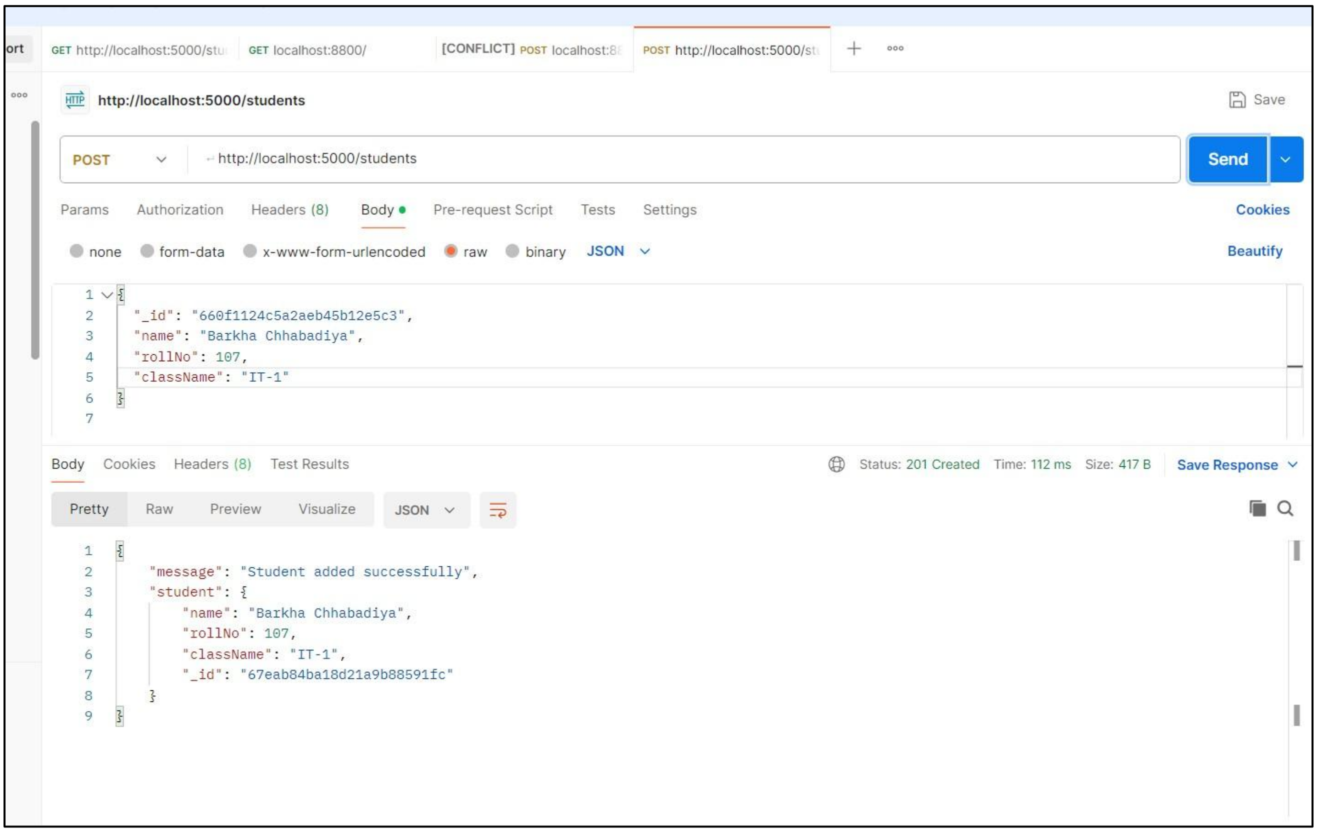
A) Retrieve details of an individual student by Roll No.


GET http://localhost:5000/students/rollNo/101

Status: 200 OK Time: 87 ms Size: 360 B

```
{
    "_id": "67eaa361686a1d9602cc751d",
    "name": "Sonam Chhabaidiya",
    "rollNo": 101,
    "className": "IT-1"
}
```

b) Add a new student to the database

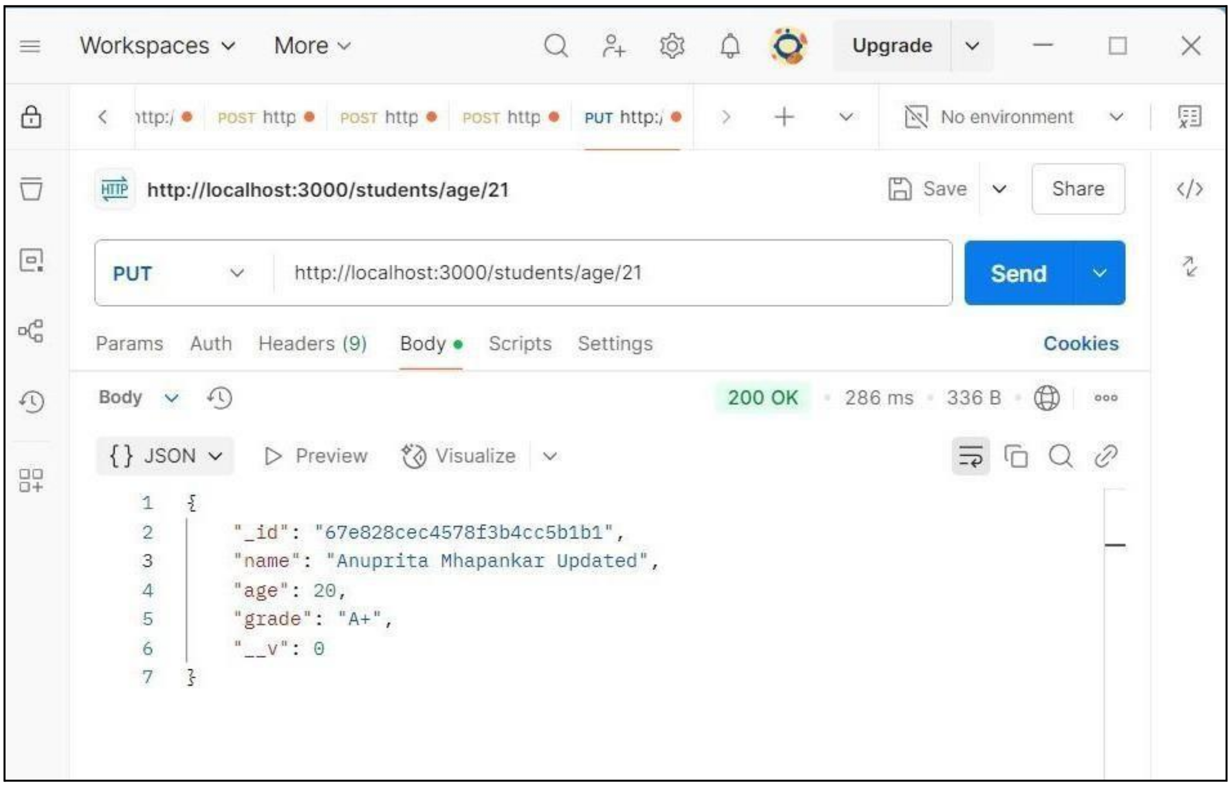

POST http://localhost:5000/students

```
{
  "_id": "660f1124c5a2aeb45b12e5c3",
  "name": "Barkha Chhabadiya",
  "rollNo": 107,
  "className": "IT-1"
}
```

Status: 201 Created Time: 112 ms Size: 417 B

```
{
    "message": "Student added successfully",
    "student": {
        "name": "Barkha Chhabadiya",
        "rollNo": 107,
        "className": "IT-1",
        "_id": "67eab84ba18d21a9b88591fc"
    }
}
```

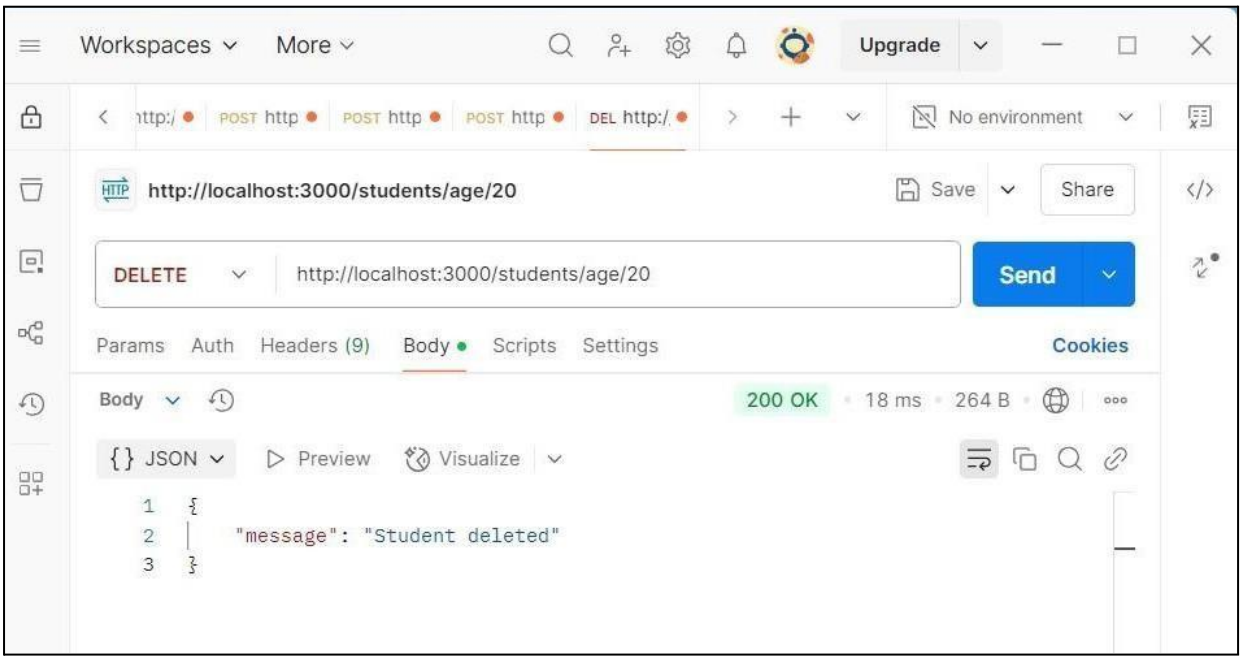
c) Update details of an existing student by ID


PUT http://localhost:3000/students/age/21

200 OK · 286 ms · 336 B

```
{
    "_id": "67e828cec4578f3b4cc5b1b1",
    "name": "Anuprita Mhapankar Updated",
    "age": 20,
    "grade": "A+",
    "__v": 0
}
```

d) Delete a student from the database by ID.


DELETE http://localhost:3000/students/age/20

200 OK · 18 ms · 264 B

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

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**.