

Experiment – 7: MongoDB

Name of Student	Aryan Patankar
Class Roll No	D15A_33
D.O.P.	13/03/2025
D.O.S.	20/03/2025
Sign and Grade	

1) **Aim:** To study CRUD operations in MongoDB

2) **Problem Statement:**

A) Create a new database to storage student details of IT dept(Name, Roll no, class name) and perform the following on the database

- Insert one student details
- Insert at once multiple student details
- Display student for a particular class
- Display students of specific roll no in a class
- Change the roll no of a student
- Delete entries of particular student

B) Create a set of RESTful endpoints using Node.js, Express, and Mongoose for handling student data operations.

The endpoints should support:

- Retrieve a list of all students.
- Retrieve details of an individual student by ID.
- Add a new student to the database.
- Update details of an existing student by ID.
- 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.

3) Output:

A)

1) Insert one student details

```
> db.student.insertOne({
  name: "Alice Smith",
  rollNo: "IT001",
  className: "IT-A"
})
< {
  acknowledged: true,
  insertedId: ObjectId('67db9718a294c23ca6ea3f85')
}
```

The screenshot shows the MongoDB Compass web interface. The top navigation bar includes 'Welcome', 'student', 'admin', and two tabs for 'mongosh: localhost:27017'. The main breadcrumb is 'localhost:27017 > admin > student'. Below this, there are tabs for 'Documents' (0), 'Aggregations', 'Schema', 'Indexes' (1), and 'Validation'. The 'Documents' tab is active. A search bar contains the text 'Type a query: { field: 'value' } or [Generate query](#)'. To the right of the search bar are 'Explain' and 'Reset' buttons. Below the search bar, there are buttons for '+ ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. On the right side of this row, there is a dropdown set to '25' and text '1 - 1 of 1'. The main content area displays a single document in JSON format:

```
{
  "_id": ObjectId('67db9718a294c23ca6ea3f85'),
  "name": "Alice Smith",
  "rollNo": "IT001",
  "className": "IT-A"
}
```

2) Insert at once multiple student details

```
> db.student.insertMany([
  { name: "Bob Johnson", rollNo: "IT002", className: "IT-A" },
  { name: "Charlie Brown", rollNo: "IT003", className: "IT-B" }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67db972aa294c23ca6ea3f86'),
    '1': ObjectId('67db972aa294c23ca6ea3f87')
  }
}
```

localhost:27017 > admin > student Open MongoDB shell

Documents 0 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 3 of 3 ↺ ↻ ↷ ≡ {} | ⌂

```
{
  "_id": ObjectId("67db9718a294c23ca6ea3f85"),
  "name": "Alice Smith",
  "rollNo": "IT001",
  "className": "IT-A"
}
```

```
{
  "_id": ObjectId("67db972aa294c23ca6ea3f86"),
  "name": "Bob Johnson",
  "rollNo": "IT002",
  "className": "IT-A"
}
```

```
{
  "_id": ObjectId("67db972aa294c23ca6ea3f87"),
  "name": "Charlie Brown",
  "rollNo": "IT003",
  "className": "IT-B"
}
```

3) Display students by roll number in a particular class

```
> db.student.find({ className: "IT-A" })
< {
  _id: ObjectId('67db9718a294c23ca6ea3f85'),
  name: 'Alice Smith',
  rollNo: 'IT001',
  className: 'IT-A'
}
{
  _id: ObjectId('67db972aa294c23ca6ea3f86'),
  name: 'Bob Johnson',
  rollNo: 'IT002',
  className: 'IT-A'
}
```

- 4) Display students of specific roll no in a class

```
> db.student.find({ rollNo: "IT002", className: "IT-A" })
< {
  _id: ObjectId('67db972aa294c23ca6ea3f86'),
  name: 'Bob Johnson',
  rollNo: 'IT002',
  className: 'IT-A'
}
```

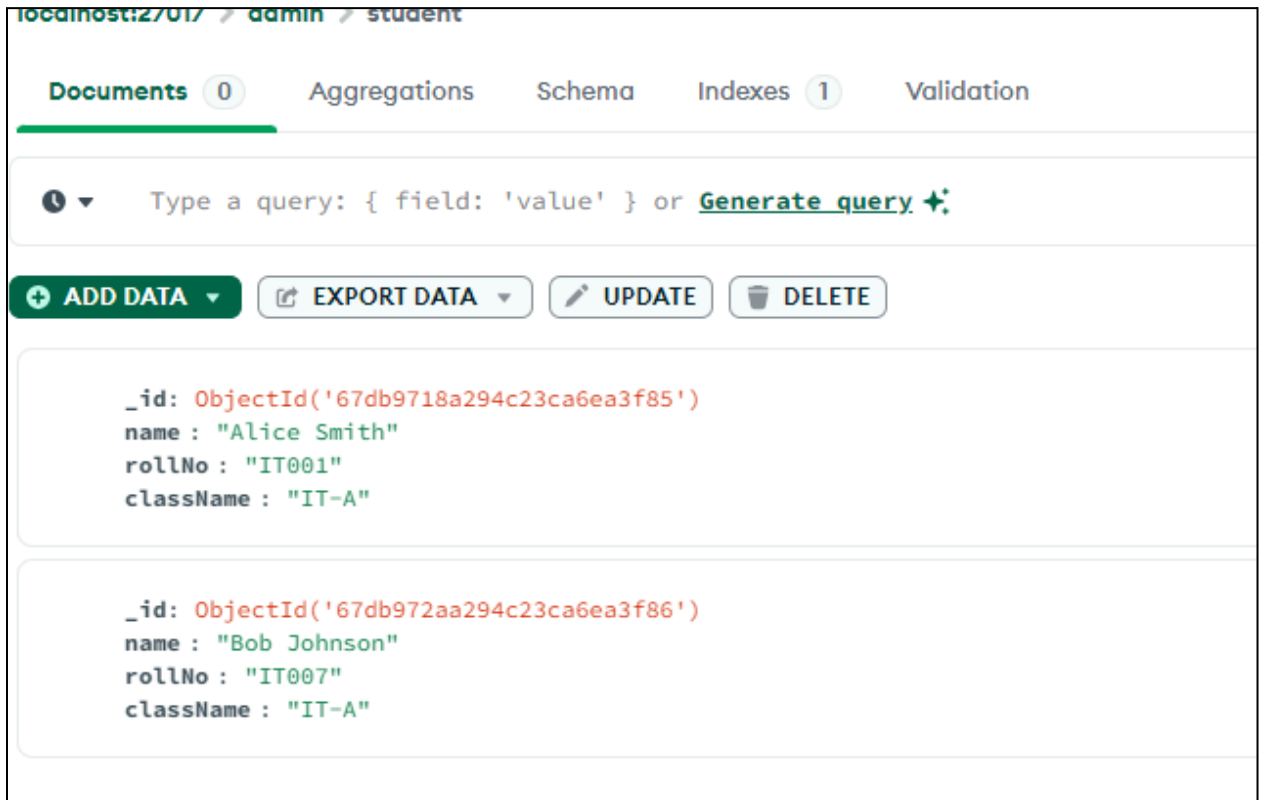
- 5) Change the roll no of a student

```
> db.student.updateOne(
  { name: "Bob Johnson" },
  { $set: { rollNo: "IT007" } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

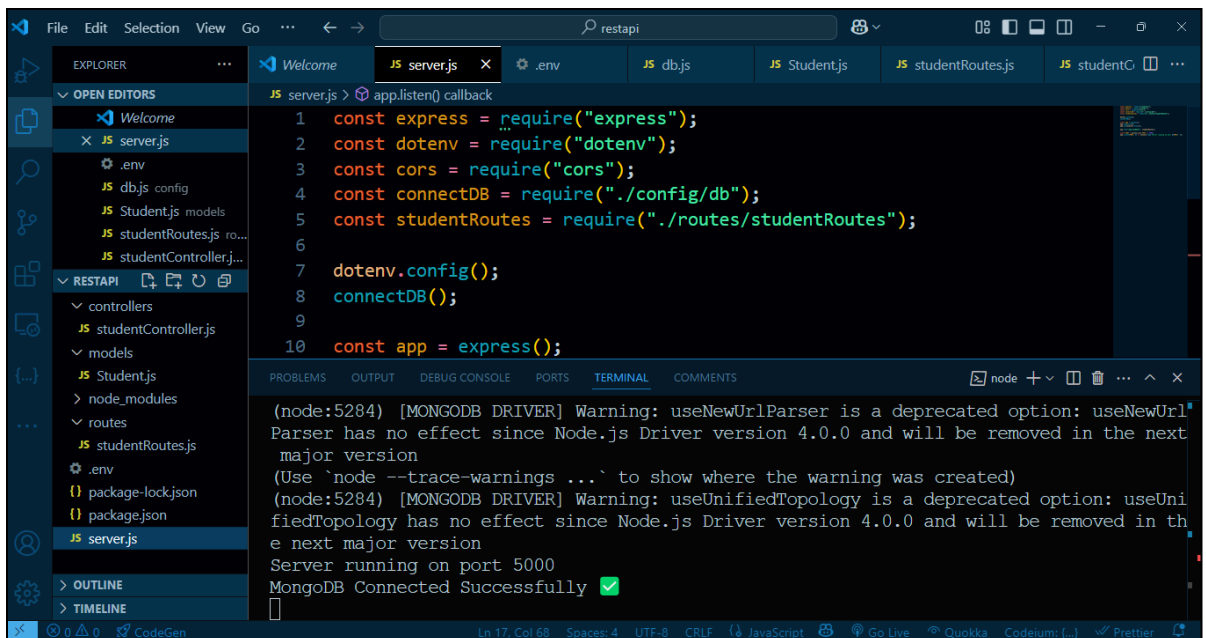
```
_id: ObjectId('67db972aa294c23ca6ea3f86')
name : "Bob Johnson"
rollNo : "IT007"
className : "IT-A"
```

- 6) Delete entries of particular student

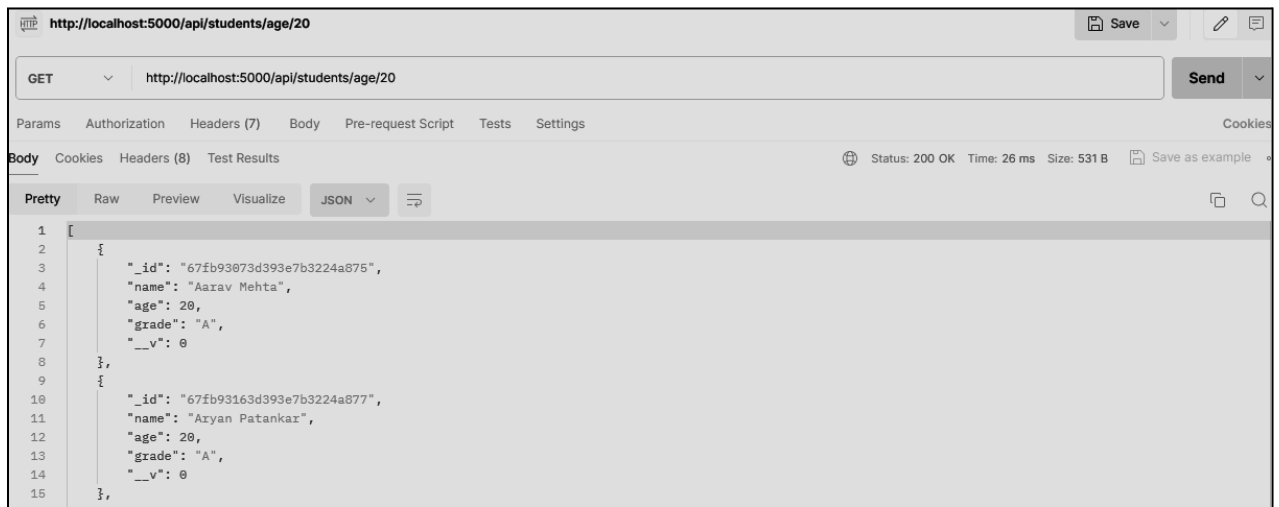
```
> db.student.deleteOne({ rollNo: "IT003" })
< {
  acknowledged: true,
  deletedCount: 1
}
```



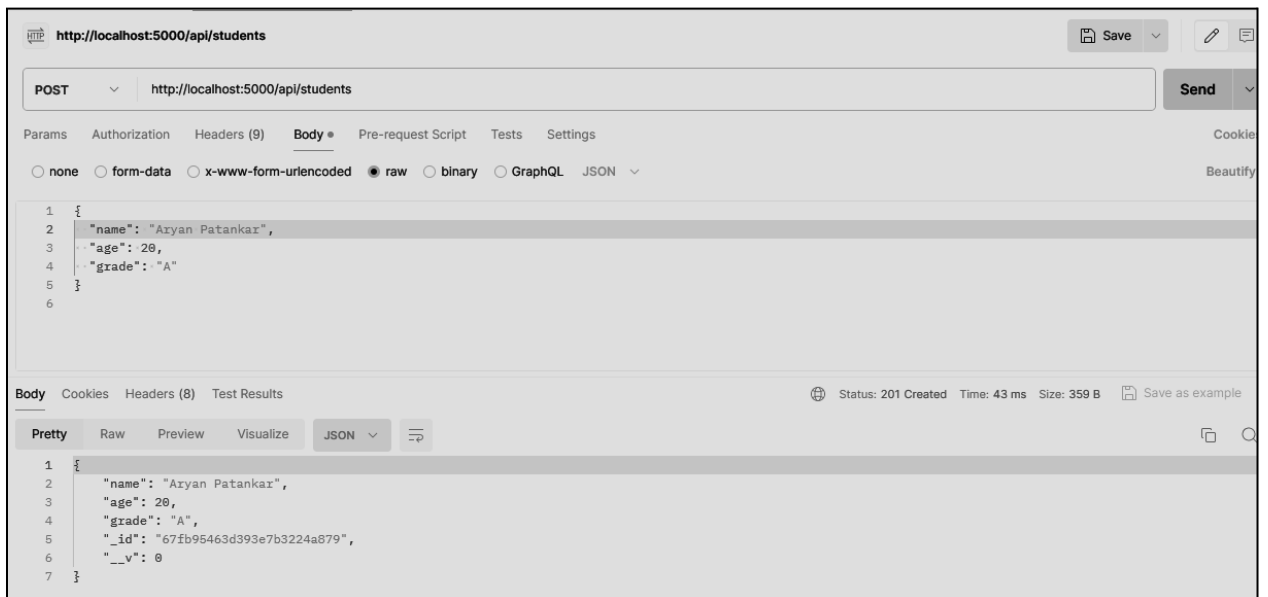
B) Restful API's



1) Retrieve details of an individual student by ID



2) Add a new student to the database.



3)Update details of an existing student by ID.

The screenshot shows a REST client interface with a PUT request to `http://localhost:5000/api/students/67fb93163d393e7b3224a877`. The request body is a JSON object: `{ "name": "Aryan Patankar", "age": 20, "grade": "A" }`. The response status is 200 OK, and the response body is a JSON object: `{ "_id": "67fb93163d393e7b3224a877", "name": "Aryan Patankar", "age": 20, "grade": "A", "__v": 0 }`.

```
PUT http://localhost:5000/api/students/67fb93163d393e7b3224a877

{
  "name": "Aryan Patankar",
  "age": 20,
  "grade": "A"
}
```

Status: 200 OK Time: 74 ms Size: 354 B

```
{
  "_id": "67fb93163d393e7b3224a877",
  "name": "Aryan Patankar",
  "age": 20,
  "grade": "A",
  "__v": 0
}
```

4)Delete a student from the database by ID

The screenshot shows a REST client interface with a DELETE request to `http://localhost:5000/api/students/67fb95463d393e7b3224a879`. The response status is 200 OK, and the response body is a JSON object: `{ "message": "Student deleted successfully" }`.

```
DELETE http://localhost:5000/api/students/67fb95463d393e7b3224a879
```

Status: 200 OK Time: 29 ms

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