

WEB-X EXPERIMENT-7

Name of the Student	Eesha Chavan
Class and Roll No	D15A-08
DOP	
DOS	
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/eeshachavan/webx-exp7-eesha>

OUTPUT

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

Create Database

Database Name

Collection Name

☐ Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ Additional preferences (e.g. Custom collation, Clustered collections)

Cancel Create Database

a) Insert one Student Detail

Insert Document

To collection students

VIEW { } ≡



```
1 {  
2   "name": "Alice",  
3   "roll_no": 101,  
4   "class_name": "TYIT"  
5 }  
6
```

Cancel Insert



b) Insert multiple Student Details

Insert Document

To collection students

VIEW  

```
1 [
2   { "name": "Bob", "roll_no": 102, "class_name": "TYIT" },
3   { "name": "Charlie", "roll_no": 103, "class_name": "SYIT" },
4   { "name": "Daisy", "roll_no": 104, "class_name": "TYIT" }
5 ]
6
```

Display Students for a Particular Class { "className": "TYIT" }




IT_dept.students

STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

Filter { "class_name": "TYIT" }   

QUERY RESULTS: 1-3 OF 3

`_id: ObjectId('67f3e5336d902bcabdddf95')`
`name: "Alice"`
`roll_no: 101`
`class_name: "TYIT"`

`_id: ObjectId('67f3e5526d902bcabdddf96')`
`name: "Bob"`
`roll_no: 102`
`class_name: "TYIT"`

`_id: ObjectId('67f3e5526d902bcabdddf98')`
`name: "Daisy"`
`roll_no: 104`
`class_name: "TYIT"`

d) Display Student of a Specific Roll Number in a Class
{ "className": "TYIT", "rollNo": 101 }

IT_dept.students
STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

Filter { "class_name": "TYIT", "roll_no": 101 } Reset Apply Options

QUERY RESULTS: 1-1 OF 1

	<pre>{ "_id": ObjectId('67f3e5336d982bcabdddfe95'), "name": "Alice", "roll_no": 101, "class_name": "TYIT" }</pre>	
--	---	--

e) Change the Roll Number of a Student Filter:
{ "name": "Alice" }
Update:

IT_dept.students
STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

Filter {} Reset Apply Options

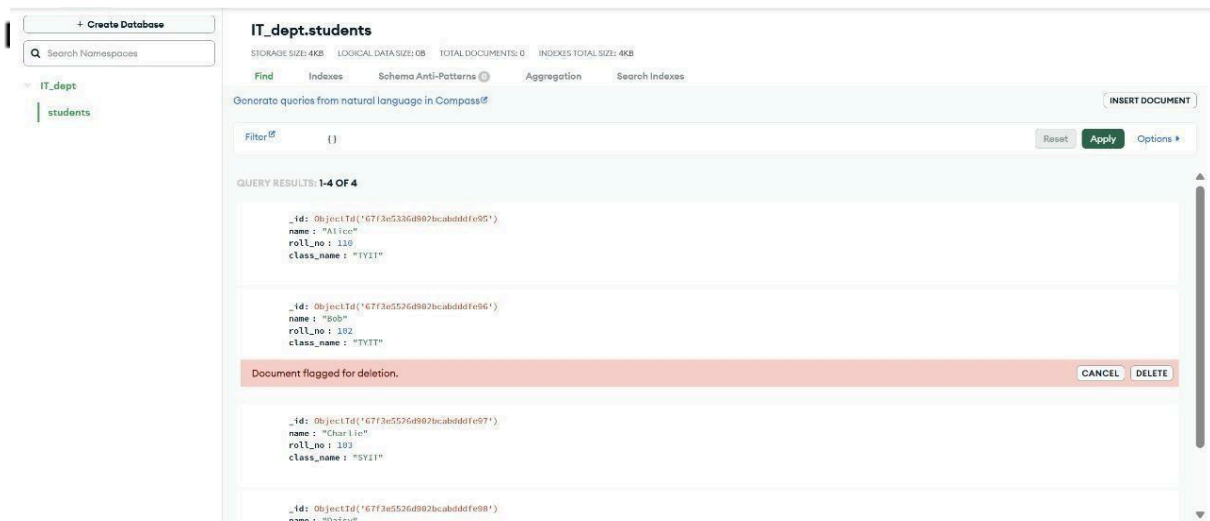
QUERY RESULTS: 1-4 OF 4

1	<pre>{ "_id": ObjectId('67f3e5336d982bcabdddfe95'), "name": "Alice", "roll_no": 101, "class_name": "TYIT" }</pre>	ObjectId String Int32 String
2		
3		
4		

Document modified. CANCEL UPDATE

<pre>{ "_id": ObjectId('67f3e5526d982bcabdddfe96'), "name": "Bob", "roll_no": 102, "class_name": "TYIT" }</pre>
<pre>{ "_id": ObjectId('67f3e5526d982bcabdddfe97'), "name": "Charlie", "roll_no": 103, "class_name": "SYIT" }</pre>
<pre>{ "_id": ObjectId('67f3e5526d982bcabdddfe98'), "name": "Daisy" }</pre>

f) Delete Entries of a Particular Student Filter:
{ "name": "Bob" }



2. Restful api

```

JS server.js > useUnifiedTopology
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const cors = require('cors');
4  const studentRoutes = require('./routes/studentRoutes');
5
6  const app = express();
7  app.use(cors());
8  app.use(express.json());
9
10 mongoose.connect('mongodb+srv://riachaudhari68:centralorchid17@cluster0.lzxhhat.mongodb.net/?retryWrites=true&w=majority&appName=mongo-express', {
11   useNewUrlParser: true,
12   useUnifiedTopology: true
13 }).then(() => console.log('MongoDB connected'));
14
15 app.use('/api/students', studentRoutes);
16
17 app.listen(5000, () => {
18   console.log('Server running on http://localhost:5000');
19 });
20

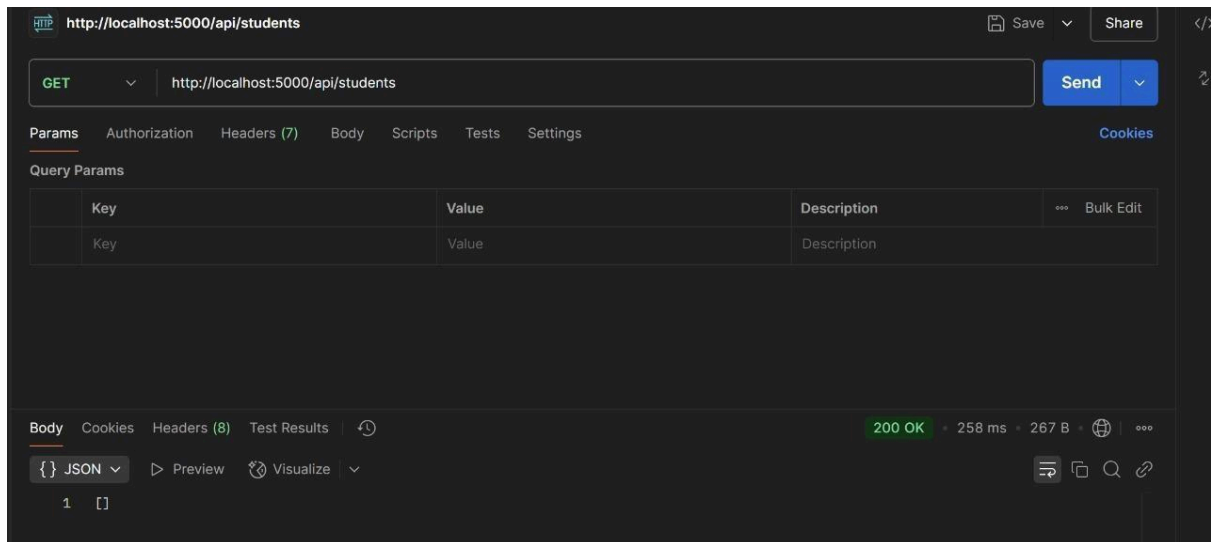
```

```

$ node server.js
(node:20972) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 3.6.0 will be removed in the next major version
(node:20972) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 3.6.0 will be removed in the next major version
Server running on http://localhost:5000
MongoDB connected

```

a) Retrieve details



HTTP `http://localhost:5000/api/students` Save Share

GET `http://localhost:5000/api/students` Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

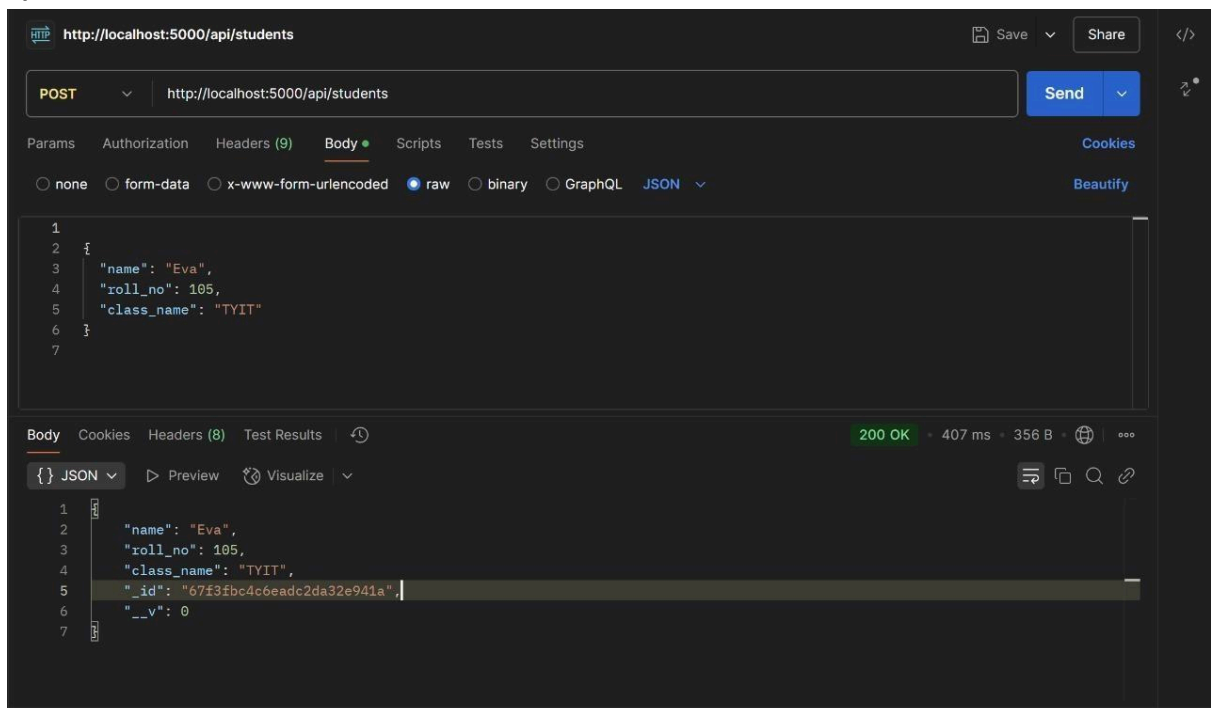
Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results 200 OK • 258 ms • 267 B • Visualize

```
{}
```

1

b) Add a new student to the database



HTTP `http://localhost:5000/api/students` Save Share

POST `http://localhost:5000/api/students` Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

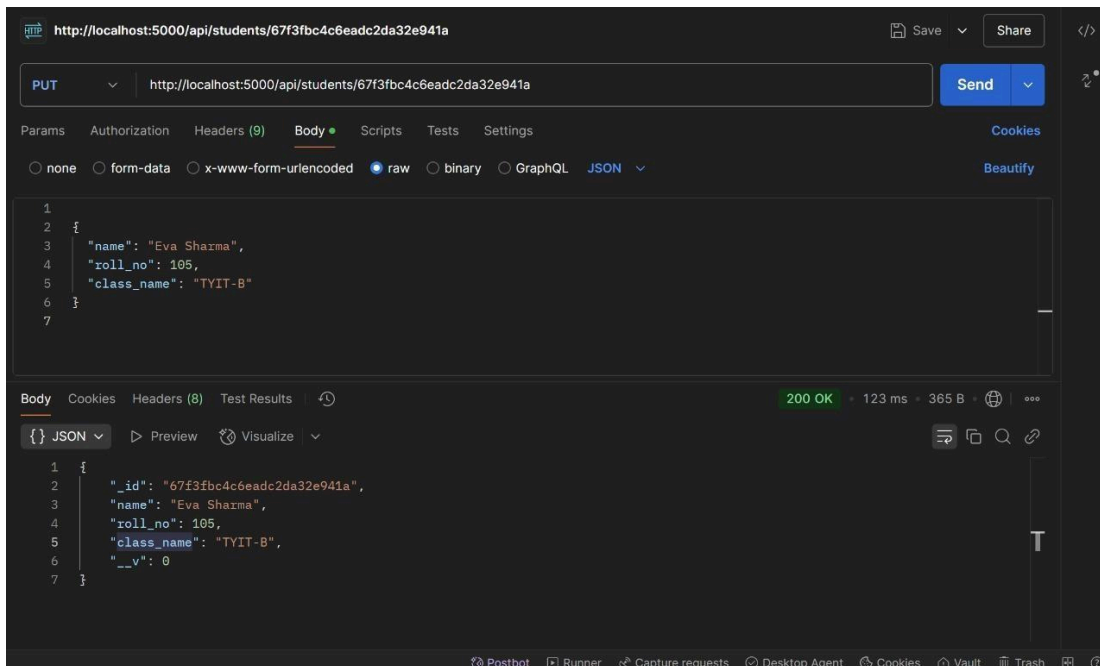
☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "name": "Eva",
3   "roll_no": 105,
4   "class_name": "TYIT"
5 }
```

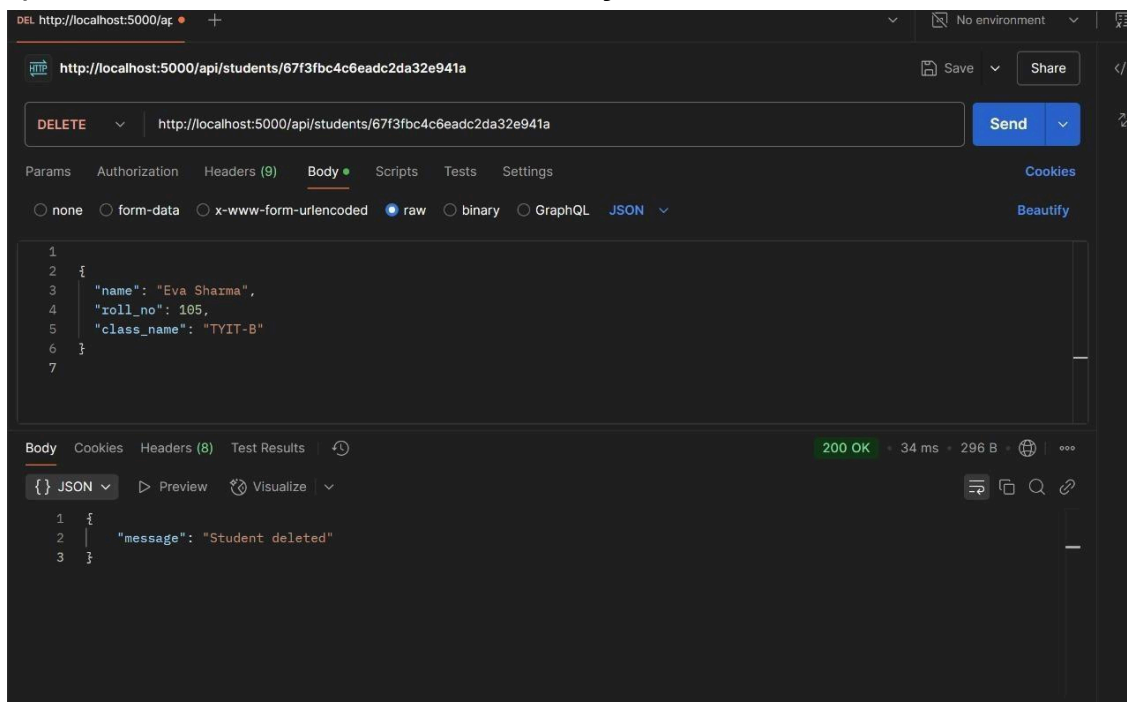
Body Cookies Headers (8) Test Results 200 OK • 407 ms • 356 B • Visualize

```
{
  "name": "Eva",
  "roll_no": 105,
  "class_name": "TYIT",
  "_id": "67f3fbc4c6eadc2da32e941a",
  "__v": 0
}
```

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.