Experiment – 7: MongoDB

Name of Student	Mahvish Siddiqui
Class Roll No	D15A 56
D.O.P.	03/04/25
D.O.S.	
Sign and Grade	

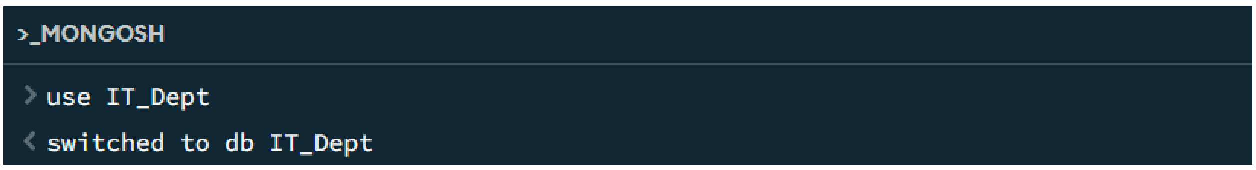
Aim: To study CRUD operations in MongoDB

Problem Statement:

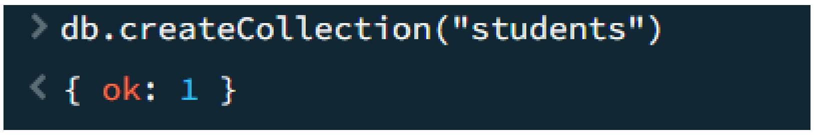
A) Create a new database to storage student details of IT dept(Name, Roll no, class name) and perform the following on the database
   a) Insert one student details
   b) Insert at once multiple student details
   c) Display student for a particular class
   d) Display students of specific roll no in a class
   e) Change the roll no of a student
   f) Delete entries of particular student

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.

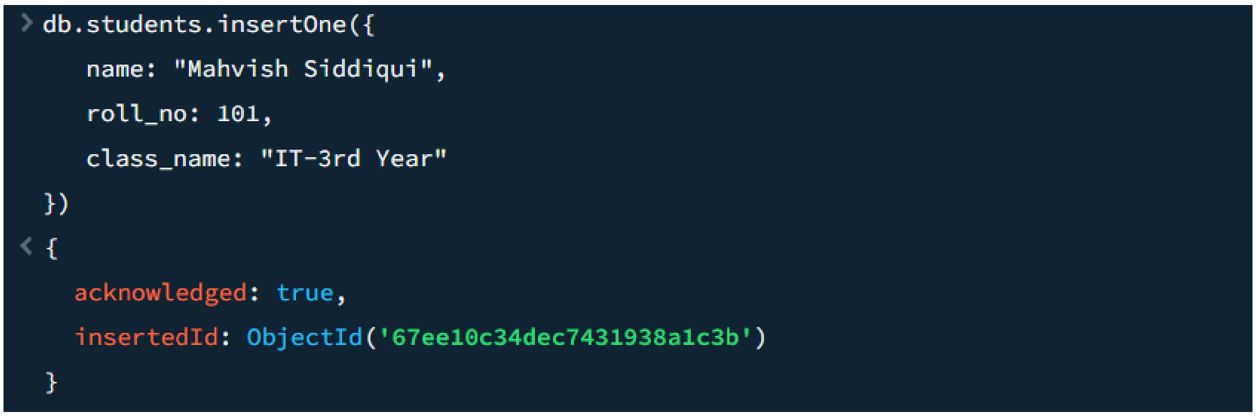
1) **Output:**
   A) MongoDB database IT department



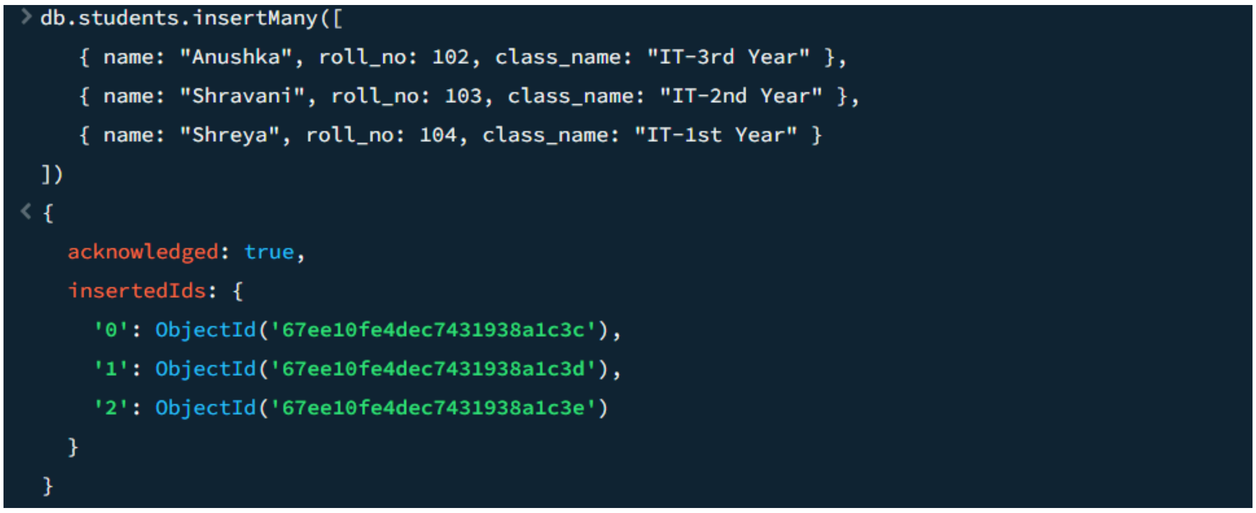
```
>_MONGOSH
> use IT_Dept
< switched to db IT_Dept
```



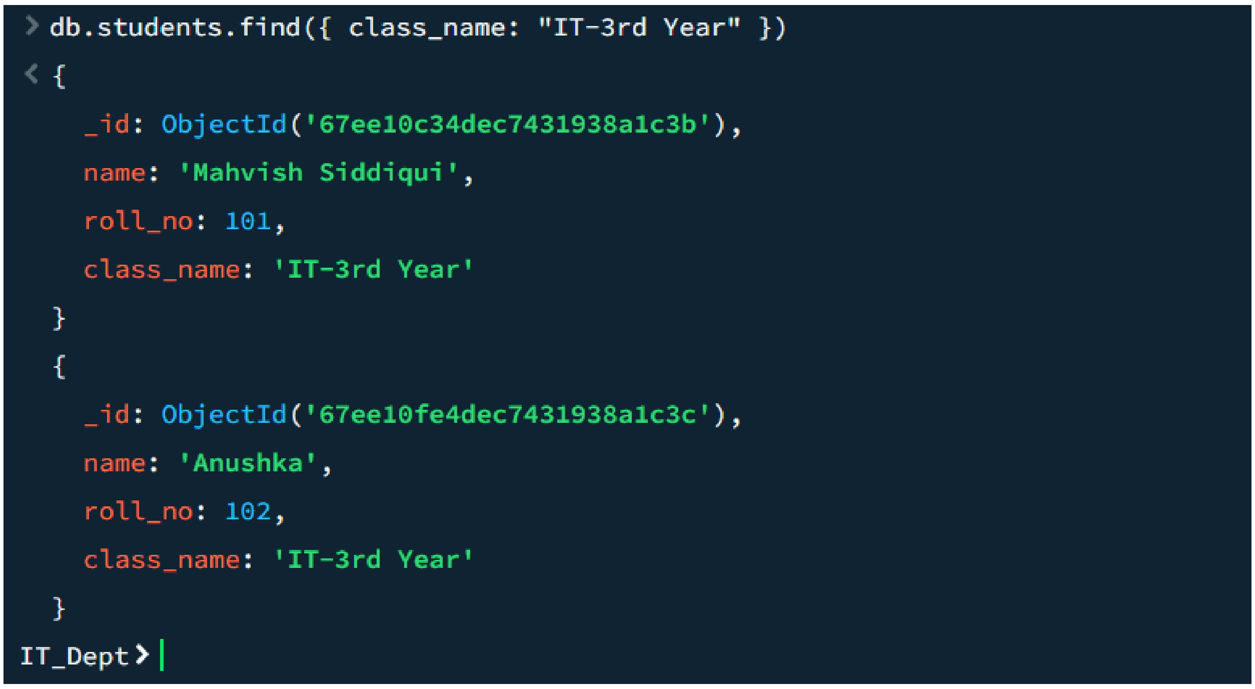
```
> db.createCollection("students")
< { ok: 1 }
```



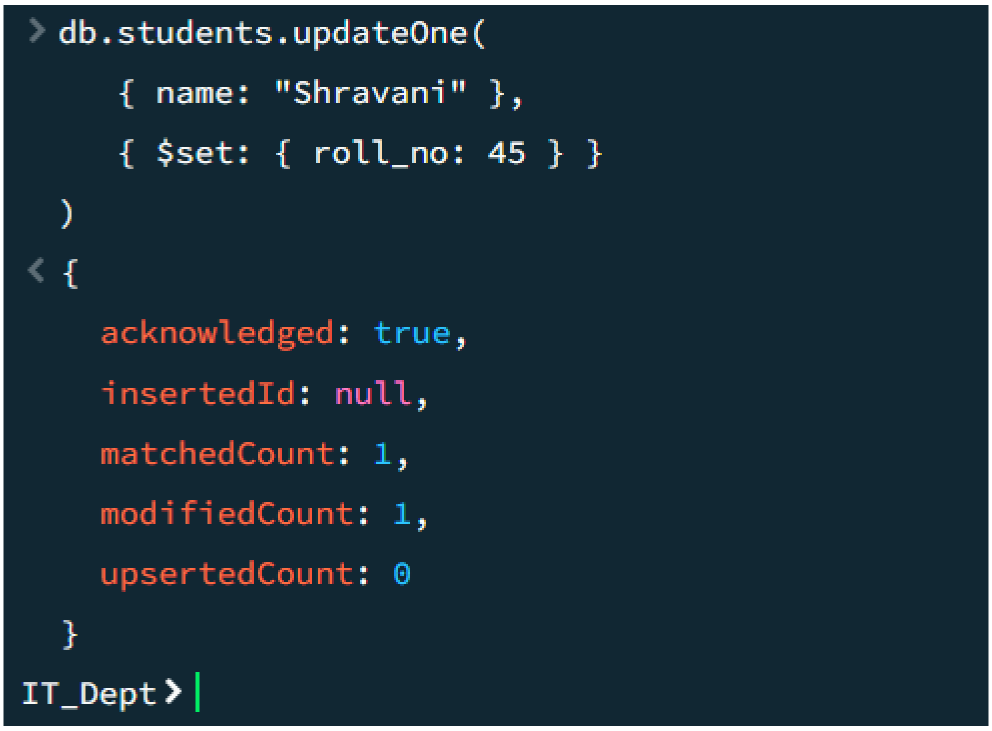
```
> db.students.insertOne({
    name: "Mahvish Siddiqui",
    roll_no: 101,
    class_name: "IT-3rd Year"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('67ee10c34dec7431938a1c3b')
  }
```



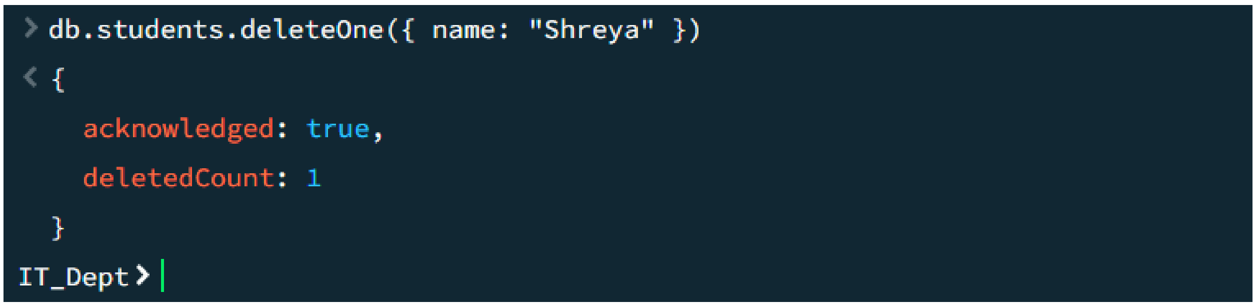
```
> db.students.insertMany([
    { name: "Anushka", roll_no: 102, class_name: "IT-3rd Year" },
    { name: "Shravani", roll_no: 103, class_name: "IT-2nd Year" },
    { name: "Shreya", roll_no: 104, class_name: "IT-1st Year" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67ee10fe4dec7431938a1c3c'),
      '1': ObjectId('67ee10fe4dec7431938a1c3d'),
      '2': ObjectId('67ee10fe4dec7431938a1c3e')
    }
  }
```



```
> db.students.find({ class_name: "IT-3rd Year" })
< {
    _id: ObjectId('67ee10c34dec7431938a1c3b'),
    name: 'Mahvish Siddiqui',
    roll_no: 101,
    class_name: 'IT-3rd Year'
  }
  {
    _id: ObjectId('67ee10fe4dec7431938a1c3c'),
    name: 'Anushka',
    roll_no: 102,
    class_name: 'IT-3rd Year'
  }
IT_Dept>
```

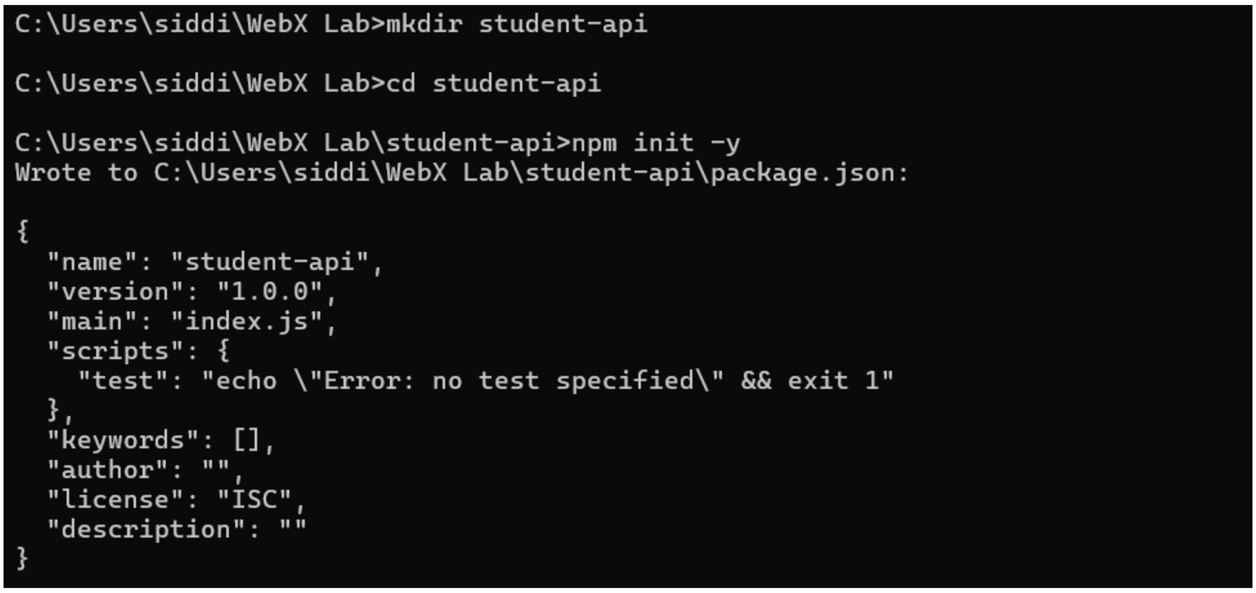


```
> db.students.updateOne(
      { name: "Shravani" },
      { $set: { roll_no: 45 } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
IT_Dept>
```

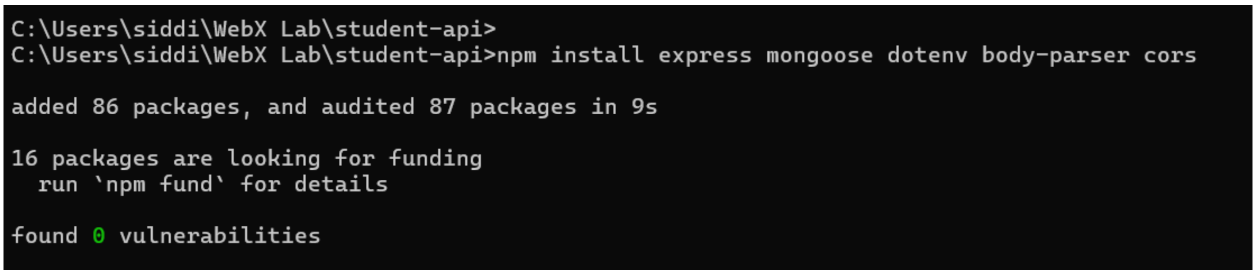


```
> db.students.deleteOne({ name: "Shreya" })
< {
    acknowledged: true,
    deletedCount: 1
  }
IT_Dept>
```

B) REST API



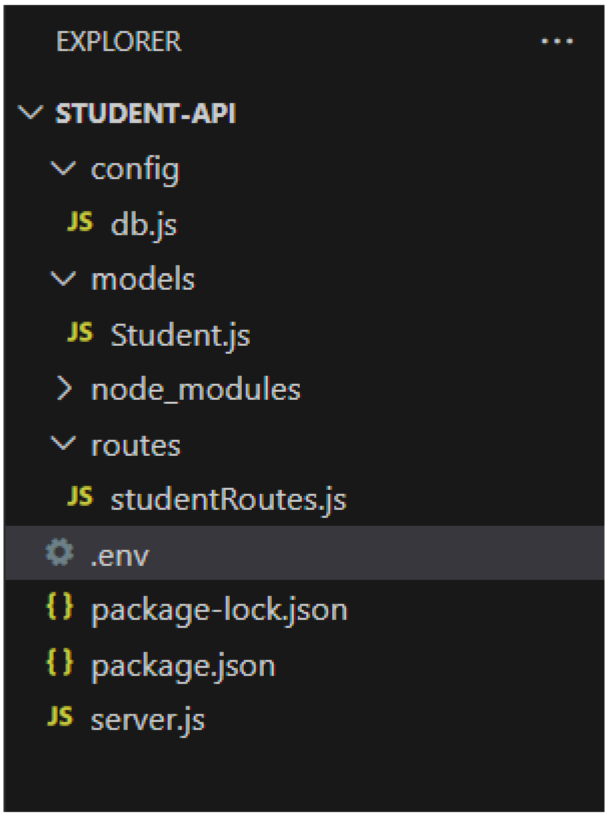
```
C:\Users\siddi\WebX Lab>mkdir student-api

C:\Users\siddi\WebX Lab>cd student-api

C:\Users\siddi\WebX Lab\student-api>npm init -y
Wrote to C:\Users\siddi\WebX Lab\student-api\package.json:

{
  "name": "student-api",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```



```
C:\Users\siddi\WebX Lab\student-api>
C:\Users\siddi\WebX Lab\student-api>npm install express mongoose dotenv body-parser cors

added 86 packages, and audited 87 packages in 9s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

In VS CODE

Folder Structure:



```
EXPLORER
STUDENT-API
  config
    db.js
  models
    Student.js
  node_modules
  routes
    studentRoutes.js
  .env
  package-lock.json
  package.json
  server.js
```

Server.js

require("dotenv").config();

const express = require("express");

const connectDB = require("./config/db");

const studentRoutes = require("./routes/studentRoutes");

const bodyParser = require("body-parser");

const cors = require("cors");

const app = express();

connectDB();

```
app.use(cors());
app.use(bodyParser.json());

app.use("/api/students", studentRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

<u>Student.js</u>

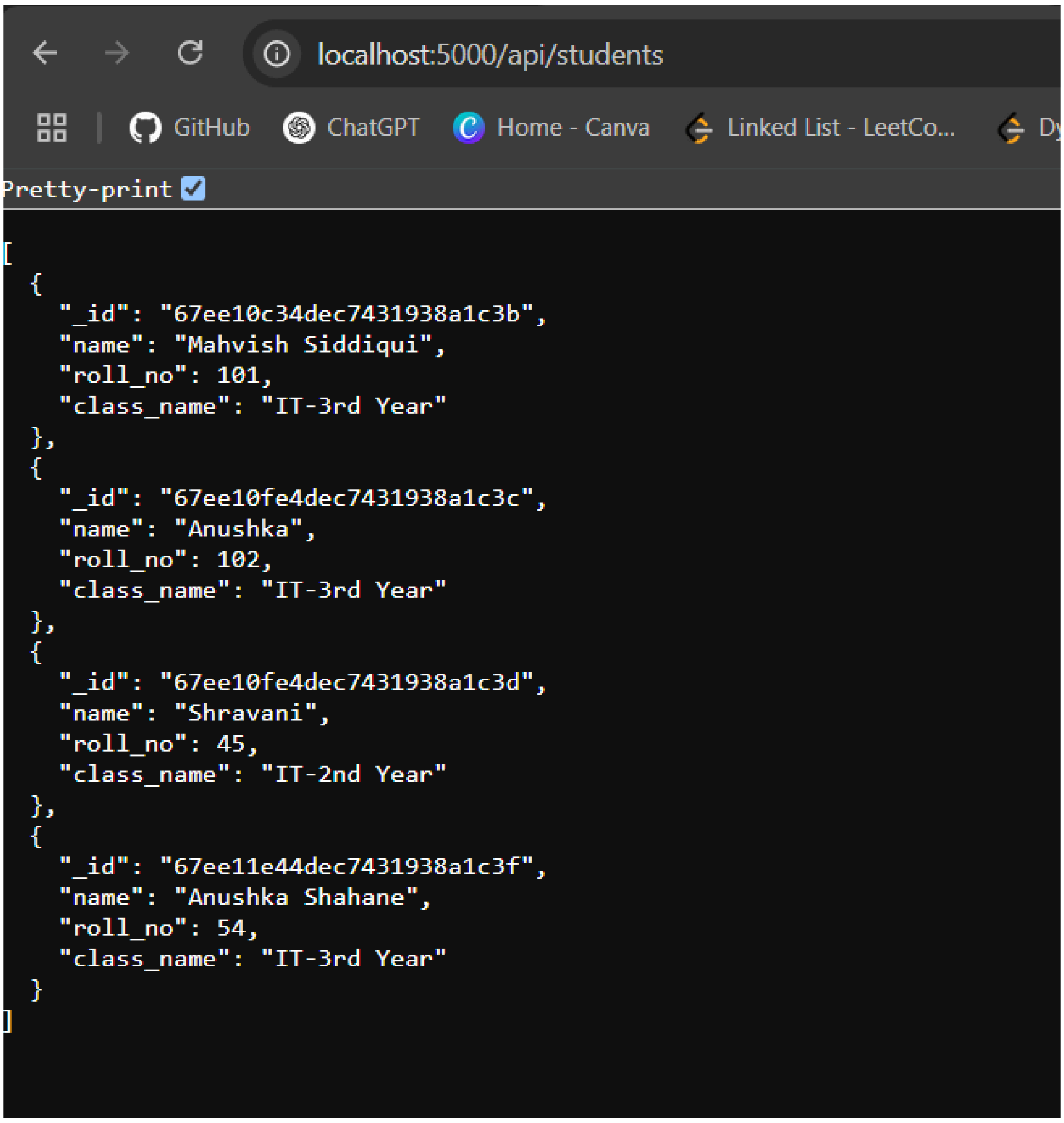
```
const mongoose = require("mongoose");
const studentSchema = new mongoose.Schema({
   name: { type: String, required: true },
   roll_no: { type: Number, required: true, unique: true },
   class_name: { type: String, required: true }
});

module.exports = mongoose.model("Student", studentSchema);
```

<u>studentRoutes.js</u>

```
const express = require("express");
const Student = require("../models/Student");
const router = express.Router();

router.get("/", async (req, res) => {
   const students = await Student.find();
   res.json(students);
});

router.get("/:id", async (req, res) => {
   const student = await Student.findById(req.params.id);
   res.json(student);
});
```

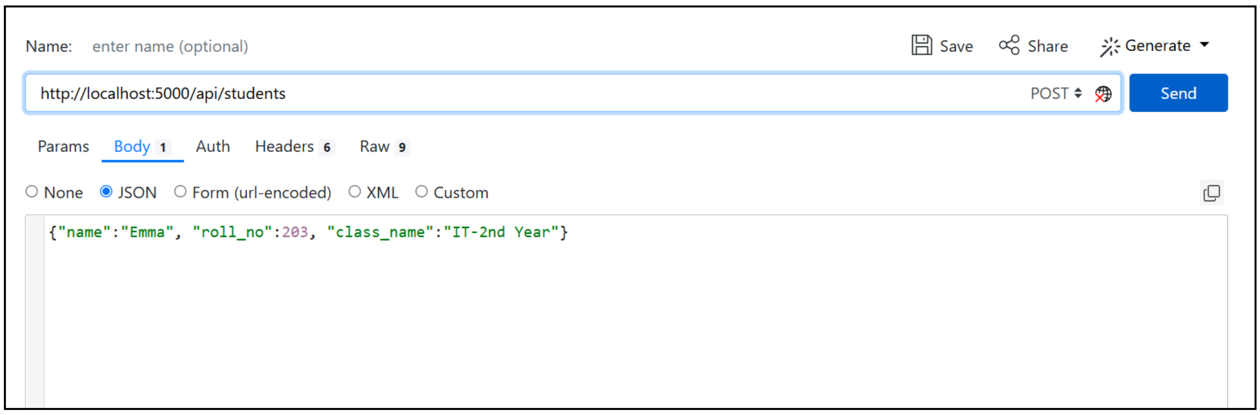
```
router.post("/", async (req, res) => {
   const newStudent = new Student(req.body);
   await newStudent.save();
   res.status(201).json(newStudent);
});

router.put("/:id", async (req, res) => {
     const updatedStudent = await Student.findByIdAndUpdate(req.params.id, req.body, {
new: true });
   res.json(updatedStudent);
});

router.delete("/:id", async (req, res) => {
   await Student.findByIdAndDelete(req.params.id);
   res.json({ message: "Student deleted" });
});

module.exports = router;
```

1. Get all students



```
[
  {
    "_id": "67ee10c34dec7431938a1c3b",
    "name": "Mahvish Siddiqui",
    "roll_no": 101,
    "class_name": "IT-3rd Year"
  },
  {
    "_id": "67ee10fe4dec7431938a1c3c",
    "name": "Anushka",
    "roll_no": 102,
    "class_name": "IT-3rd Year"
  },
  {
    "_id": "67ee10fe4dec7431938a1c3d",
    "name": "Shravani",
    "roll_no": 45,
    "class_name": "IT-2nd Year"
  },
  {
    "_id": "67ee11e44dec7431938a1c3f",
    "name": "Anushka Shahane",
    "roll_no": 54,
    "class_name": "IT-3rd Year"
  }
]
```

2. Add a student

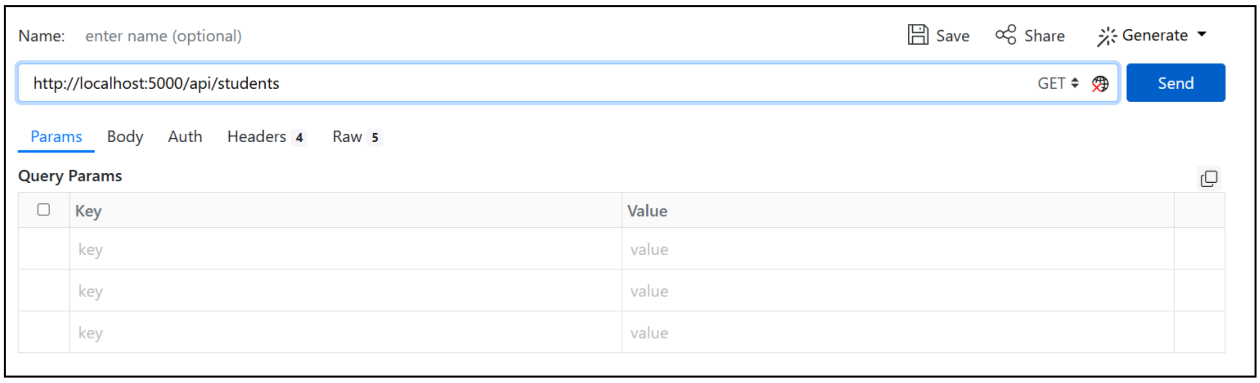


http://localhost:5000/api/students POST

Body: [✓ JSON]

```
{"name":"Emma", "roll_no":203, "class_name":"IT-2nd Year"}
```

GET request

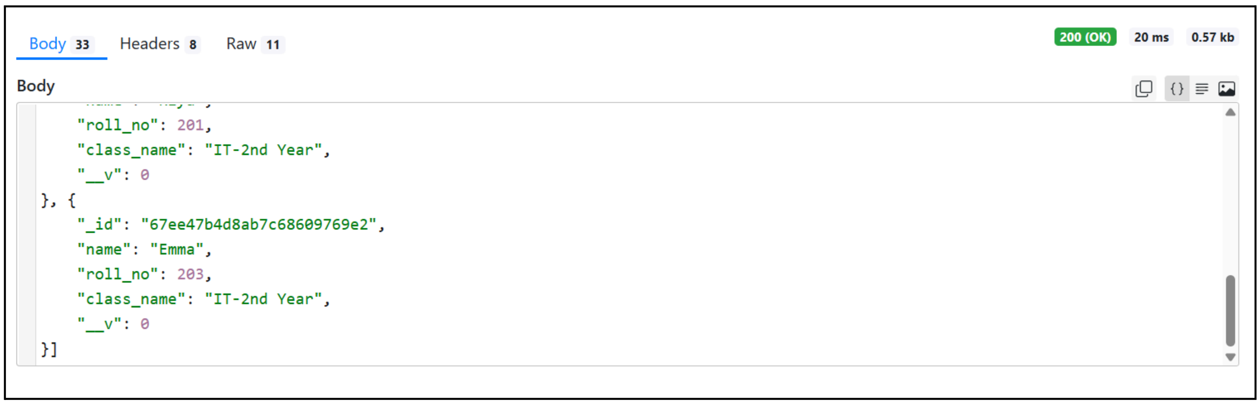

Name: enter name (optional) Save Share Generate

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

Params Body Auth Headers 4 Raw 5

Query Params

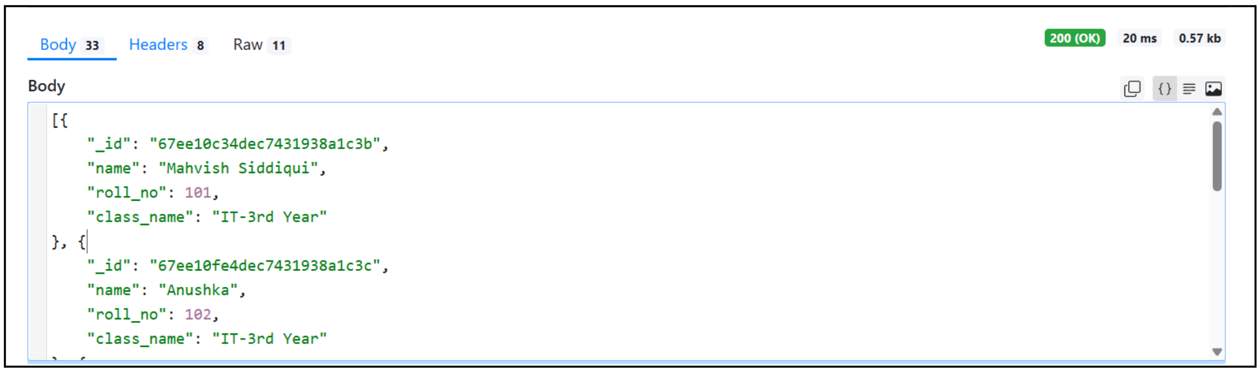
	Key	Value
	key	value
	key	value
	key	value


Body 33 Headers 8 Raw 11 200 (OK) 20 ms 0.57 kb

Body

```
    "roll_no": 201,
    "class_name": "IT-2nd Year",
    "__v": 0
}, {
    "_id": "67ee47b4d8ab7c68609769e2",
    "name": "Emma",
    "roll_no": 203,
    "class_name": "IT-2nd Year",
    "__v": 0
}]
```

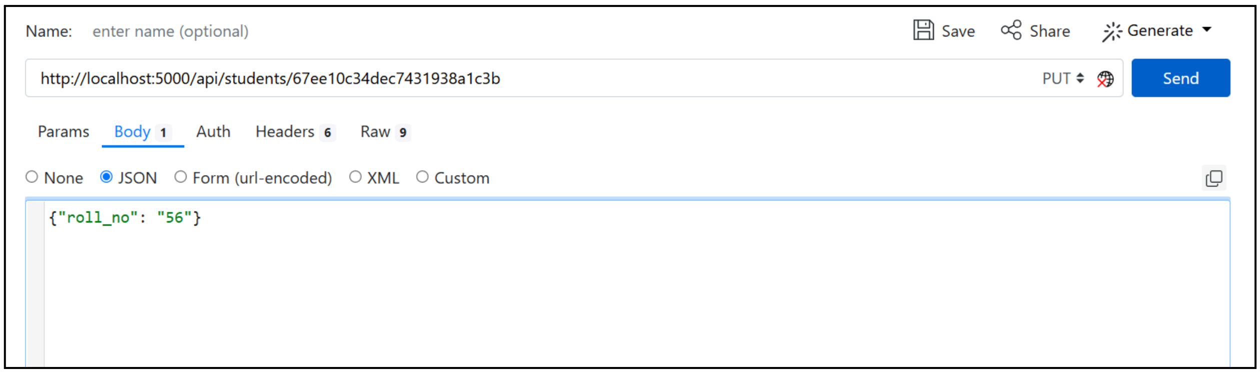
Initially:


Body 33 Headers 8 Raw 11 200 (OK) 20 ms 0.57 kb

Body

```
[{
    "_id": "67ee10c34dec7431938a1c3b",
    "name": "Mahvish Siddiqui",
    "roll_no": 101,
    "class_name": "IT-3rd Year"
}, {
    "_id": "67ee10fe4dec7431938a1c3c",
    "name": "Anushka",
    "roll_no": 102,
    "class_name": "IT-3rd Year"
```

PUT request

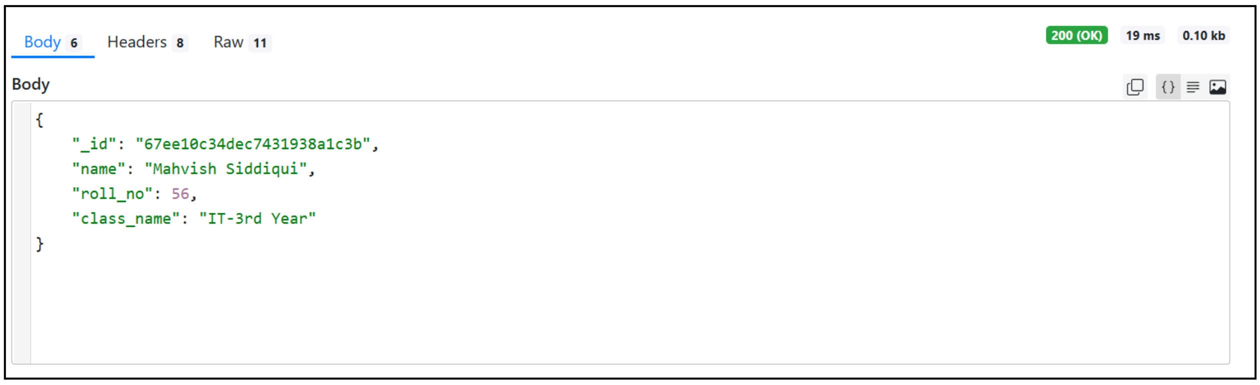

Name: enter name (optional) Save Share Generate

http://localhost:5000/api/students/67ee10c34dec7431938a1c3b PUT Send

Params Body 1 Auth Headers 6 Raw 9

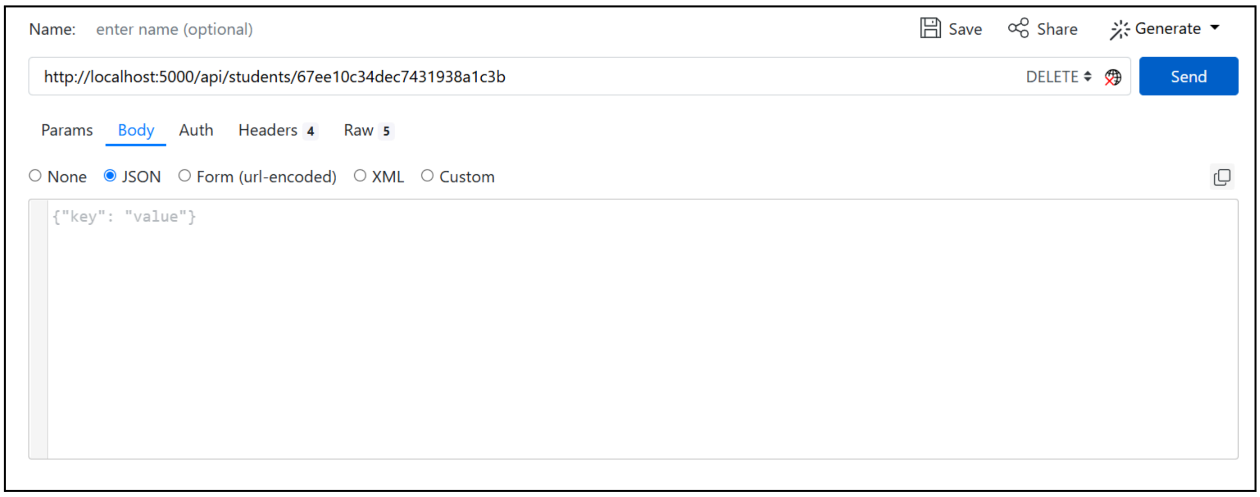
None [✓ JSON] Form (url-encoded) XML Custom

```
{"roll_no": "56"}
```



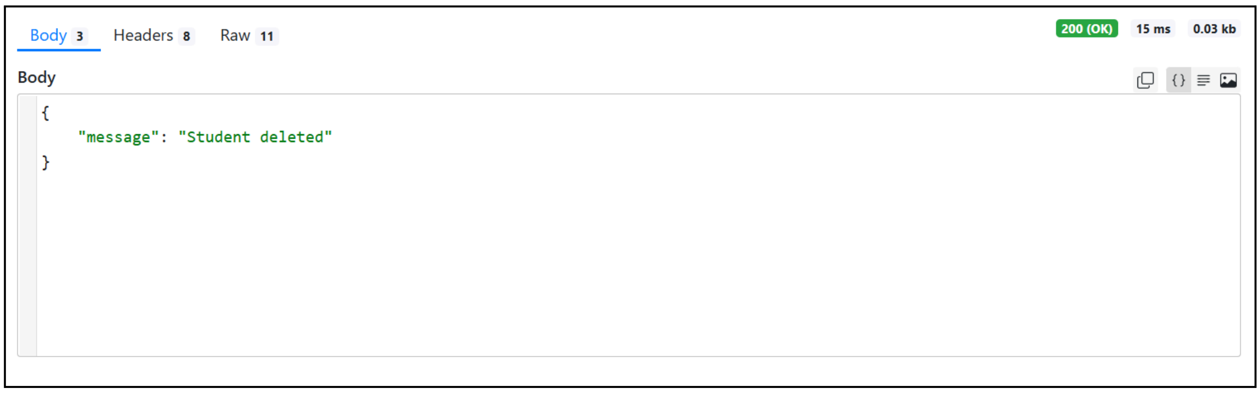
```
{
    "_id": "67ee10c34dec7431938a1c3b",
    "name": "Mahvish Siddiqui",
    "roll_no": 56,
    "class_name": "IT-3rd Year"
}
```

DELETE Request



```
http://localhost:5000/api/students/67ee10c34dec7431938a1c3b    DELETE
```

```
{"key": "value"}
```



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

Conclusion

This assignment demonstrated how to perform CRUD operations using MongoDB to manage student details, including inserting, querying, updating, and deleting records. Additionally, we built a RESTful API using Node.js, Express, and Mongoose to interact with the database. The API supported retrieving, adding, updating, and deleting student data. Overall, this task provided hands-on experience in backend development and working with a NoSQL database in a real-world context.