

Experiment – 7: MongoDB

Name of Student	Spandan Deb
Class Roll No	13
D.O.P.	
D.O.S.	
Sign and Grade	

Aim: To study CRUD operations in MongoDB

Overview Of Tasks Performed:

This experiment focused on implementing CRUD operations in MongoDB for managing student data, including inserting, updating, and deleting records. Additionally, a RESTful API was developed using Node.js, Express, and Mongoose to handle student data operations such as retrieving, adding, updating, and deleting students. The tasks involved creating a database and collection, performing various MongoDB operations, and testing the API endpoints to ensure proper functionality.

GitHub Link- <https://github.com/spandandeb/WebXex7>

OUTPUT:

A) Created a Database ITDeptDB and collection students

Create Database ✕

Database Name

ITDeptDB

Collection Name

students

☐ 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

Inserted 1 and multiple student records

```
> use ITDeptDB
< switched to db ITDeptDB
> db.students.insertOne({
  name: "Spandan Deb",
  rollNo: 13,
  className: "IT-101"
})
< {
  acknowledged: true,
  insertedId: ObjectId('67eb5ad5420f754efddae655')
}
```

localhost:27017 > ITDepartmentDB > students

Documents 0 Aggregations Schema Indexes 1 Validation

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

Explain

Reset

ADD DATA

EXPORT DATA

UPDATE

DELETE

25

1 - 5 of 5

```
{
  "_id": ObjectId('67eb66897052e10356c1db63'),
  "name": "John Doe",
  "age": 19,
  "grade": "A",
  "__v": 0
}
```

```
{
  "_id": ObjectId('67eb66897052e10356c1db64'),
  "name": "Jane Smith",
  "age": 20,
  "grade": "A-",
  "__v": 0
}
```

```
{
  "_id": ObjectId('67eb66897052e10356c1db65'),
  "name": "Michael Johnson",
  "age": 18,
  "grade": "B+",
  "__v": 0
}
```

Display a student record for a particular class

```
> db.students.find({ className: "IT-101" })
< {
  _id: ObjectId('67eb5ad5420f754efddae655'),
  name: 'Spandan Deb',
  rollNo: 13,
  className: 'IT-101'
}
```

Display a particular student record for a given className and rollNo

```
> db.students.find({
  rollNo: 13,
  className: "IT-101"
})
< {
  _id: ObjectId('67eb5ad5420f754efddae655'),
  name: 'Spandan Deb',
  rollNo: 13,
  className: 'IT-101'
}
```

Update one student record

```
> db.students.updateOne(
  { name: "Jane Smith" },
  { $set: { rollNo: 110 } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Delete one student record

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

B) Restful API

```
// File: server.js
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(express.json());

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/ITDepartmentDB')
  .then(() => {
    console.log('MongoDB connected');
    // Seed the database with initial data
    initializeDatabase();
  })
  .catch(err => console.log('MongoDB connection error:', err));
```

Conclusion:

Implemented CRUD Operations in MongoDB and implemented a Restful API using Node.js, express and mongoose. We learned about create, read, update and delete student records both via MongoDB shell commands and API endpoints.