Aim: To connect MongoDB with a Node.js application and perform basic CRUD (Create, Read, Update, Delete) operations.

Requirements:

- **Hardware:**
  - A computer with internet access
  - Minimum 4 GB RAM
- **Software:**
  - Node.js (latest stable version)
  - MongoDB (local server or MongoDB Atlas)
  - MongoDB Compass (optional GUI tool)
  - Postman (for API testing)
  - VS Code or any code editor
  - npm (Node Package Manager)
- **Node Modules Required:**
  - `express`
  - `mongoose`
  - `body-parser`
  - `nodemon` (optional for auto-reload)

Theory:

MongoDB is a NoSQL database that stores data in JSON-like documents. It is flexible, scalable, and works seamlessly with Node.js.

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It provides a schema-based solution to model data.

CRUD Operations:

- **Create** – Insert data into the database.
- **Read** – Retrieve data from the database.
- **Update** – Modify existing data.
- **Delete** – Remove data from the database.

Express.js is a minimal and flexible Node.js web application framework that provides APIs to create server-side applications easily.

💻 Code:

1. Project initialization:

```
mkdir mongo-crud
cd mongo-crud
npm init -y
npm install express mongoose body-parser
```

2. Folder Structure:

```
mongo-crud/
├── index.js
├── models/
│   └── student.js
```

3. index.js (Main Server File):

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const Student = require('./models/student');

const app = express();
app.use(bodyParser.json());

// Connect to MongoDB
mongoose.connect('mongodb://127.0.0.1:27017/webx0db', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
})
.then(() => console.log("Connected to MongoDB"))
.catch(err => console.error("MongoDB connection error:", err));

// Create (POST)
app.post('/students', async (req, res) => {
    try {
        const student = new Student(req.body);
        await student.save();
        res.status(201).send(student);
    } catch (error) {
        res.status(400).send(error);
    }
});

// Read All (GET)
app.get('/students', async (req, res) => {
    const students = await Student.find();
    res.send(students);
});

// Read One (GET)
app.get('/students/:id', async (req, res) => {
    try {
        const student = await Student.findById(req.params.id);
        if (!student) return res.status(404).send();
        res.send(student);
    } catch {
        res.status(500).send();
    }
});

// Update (PUT)
app.put('/students/:id', async (req, res) => {
    try {
        const student = await Student.findByIdAndUpdate(req.params.id,
req.body, { new: true });
```

```
        if (!student) return res.status(404).send();
        res.send(student);
    } catch {
        res.status(400).send();
    }
});

// Delete (DELETE)
app.delete('/students/:id', async (req, res) => {
    try {
        const student = await Student.findByIdAndDelete(req.params.id);
        if (!student) return res.status(404).send();
        res.send({ message: "Deleted Successfully" });
    } catch {
        res.status(500).send();
    }
});

app.listen(3000, () => {
    console.log("Server is running on port 3000");
});
```

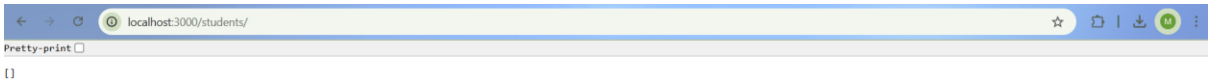
4. models/student.js:

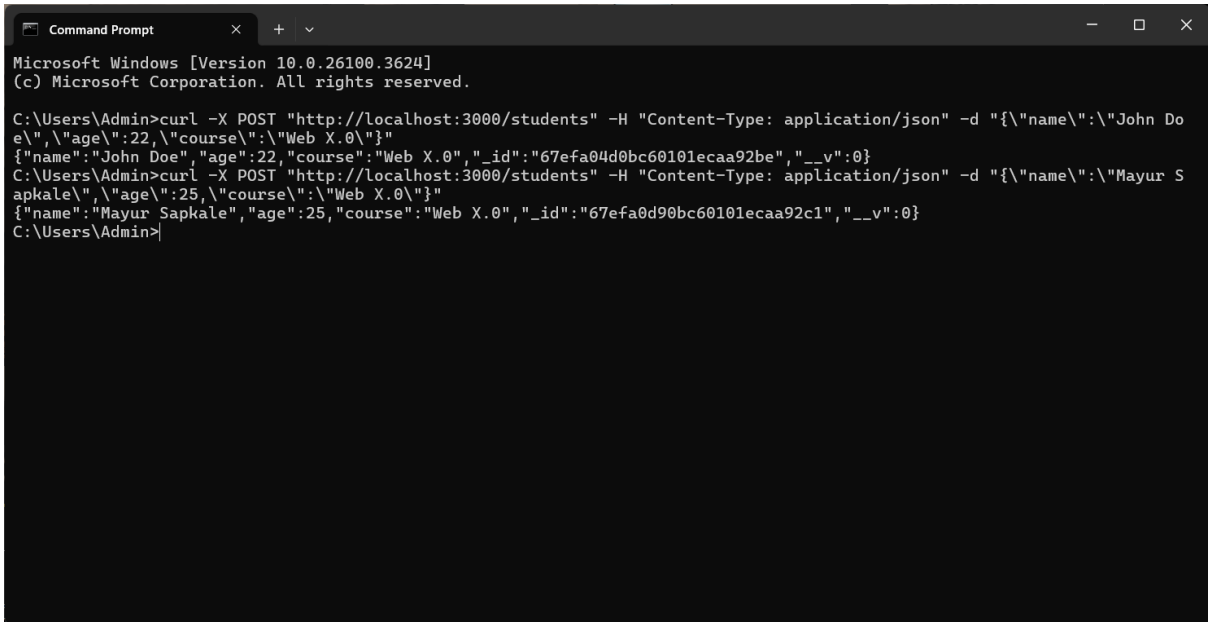
```
const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({
    name: { type: String, required: true },
    age: { type: Number, required: true },
    course: { type: String, required: true }
});

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

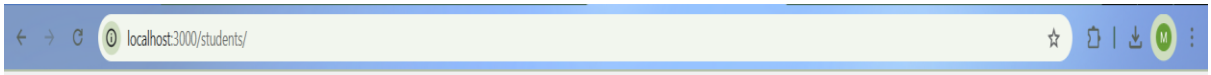
Conclusion:

In this practical, we successfully connected MongoDB to a Node.js application using the Mongoose library. We created a simple REST API to perform CRUD operations on a `Student` model. This practical helps understand how to build dynamic web applications with a backend database.

OUTPUT:

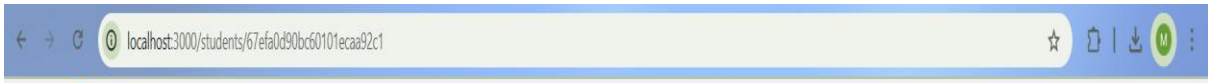

localhost:3000/students/
Pretty-print
[]


Command Prompt
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin>curl -X POST "http://localhost:3000/students" -H "Content-Type: application/json" -d "{\"name\":\"John Doe\",\"age\":22,\"course\":\"Web X.0\"}"
{"name":"John Doe","age":22,"course":"Web X.0","_id":"67efa04d0bc60101ecaa92be","__v":0}
C:\Users\Admin>curl -X POST "http://localhost:3000/students" -H "Content-Type: application/json" -d "{\"name\":\"Mayur Sapkale\",\"age\":25,\"course\":\"Web X.0\"}"
{"name":"Mayur Sapkale","age":25,"course":"Web X.0","_id":"67efa0d90bc60101ecaa92c1","__v":0}
C:\Users\Admin>



Pretty-print


```
[
  {
    "_id": "67efa04d0bc60101ecaa92be",
    "name": "John Doe",
    "age": 22,
    "course": "Web X.0",
    "__v": 0
  },
  {
    "_id": "67efa0d90bc60101ecaa92c1",
    "name": "Mayur Sapkale",
    "age": 25,
    "course": "Web X.0",
    "__v": 0
  }
]
```

Pretty-print


```
{
  "_id": "67efa0d90bc60101ecaa92c1",
  "name": "Mayur Sapkale",
  "age": 25,
  "course": "Web X.0",
  "__v": 0
}
```