EXPERIMENT 7

AIM:

Connect a Node.js application to a MongoDB database. Implement CRUD operations to interact with the database using the MongoDB Node.js driver.

THEORY:

MongoDB is a popular NoSQL database that stores data in the form of JSON-like documents. Unlike relational databases, it does not require a fixed schema, making it ideal for applications with dynamic or unstructured data.

Node.js is a runtime environment that allows JavaScript to run on the server side. To interact with MongoDB using Node.js, we use the official MongoDB Node.js driver. This driver enables Node.js applications to connect to a MongoDB database and perform various database operations such as insert, read, update, and delete.

The MongoDB Node.js driver provides a MongoClient class that is used to establish a connection to the database. Once connected, we can access collections and perform CRUD operations. Each of these operations corresponds to a method provided by the driver, such as insertOne, find, updateOne, and deleteOne.

This approach is useful in building backend services or APIs where data storage and retrieval are essential. Using the MongoDB driver directly allows more control over queries and performance optimizations compared to higher-level abstractions.

INPUT:

Filename: Index.js

```
const { MongoClient, ObjectId } = require("mongodb");

// MongoDB connection URI
const uri = "mongodb://127.0.0.1:27017";

// Database and Collection
const dbName = "studentDB";
const collectionName = "students";
```

```
// Sample data (3 students)
const sampleStudents = [
  { name: "Alice", age: 20, department: "Computer Engineering" },
  { name: "Bob", age: 21, department: "Information Technology" },
  { name: "Charlie", age: 22, department: "Electronics" }
];

async function main() {
  const client = new MongoClient(uri);

  try {
    await client.connect();
    console.log(" Connected to MongoDB");

    const db = client.db(dbName);
    const students = db.collection(collectionName);

    // 1. CREATE - Insert multiple students
    const insertResult = await students.insertMany(sampleStudents);
    console.log(" Inserted IDs:", insertResult.insertedIds);

    // 2. READ - Fetch all students
    const allStudents = await students.find().toArray();
    console.log(" All Students:", allStudents);

    // 3. UPDATE - Update one student's department
    const updatedResult = await students.updateOne(
      { name: "Bob" },
      { $set: { department: "AI & Data Science" } }
    );
    console.log(" Updated Count:", updatedResult.modifiedCount);

    // 4. DELETE - Delete one student by name
    const deletedResult = await students.deleteOne({ name: "Charlie" });
    console.log(" Deleted Count:", deletedResult.deletedCount);

  } catch (err) {
    console.error(" Error:", err);
  } finally {
    await client.close();
```

```
    console.log(" Connection closed");
  }
}
```

main();

OUTPUT:


CHAT
TERMINAL
Code
Connected to MongoDB
Inserted IDs: {
'0': new ObjectId('67f8618b85816e68ce023a6a'),
'1': new ObjectId('67f8618b85816e68ce023a6b'),
'2': new ObjectId('67f8618b85816e68ce023a6c')
}
All Students: [
{
_id: new ObjectId('67f860b4204ecc05f145ab3f'),
name: 'Alice',
age: 20,
department: 'Computer Engineering'
},
{
_id: new ObjectId('67f860b4204ecc05f145ab40'),
name: 'Bob',
age: 21,
department: 'AI & Data Science'
},
{
_id: new ObjectId('67f8618b85816e68ce023a6a'),
name: 'Alice',
age: 20,
department: 'Computer Engineering'
},
{
_id: new ObjectId('67f8618b85816e68ce023a6b'),
name: 'Bob',
age: 21,
department: 'Information Technology'
},
{
_id: new ObjectId('67f8618b85816e68ce023a6c'),
name: 'Charlie',
age: 22,
department: 'Electronics'
}
]
Updated Count: 0
Deleted Count: 1
Connection closed

CONCLUSION:

In this experiment, we successfully connected a Node.js application to a MongoDB database using the official MongoDB Node.js driver. We implemented CRUD operations including insertion, retrieval, updating, and deletion of documents in a MongoDB collection. This experiment demonstrated the fundamental steps required to interact with a NoSQL database using Node.js, laying the groundwork for building more complex applications or RESTful APIs.