CP 323: Lab Activity: Connecting Node.js to MongoDB Using the Native MongoDB Driver

📚 Objective:

By the end of this lab, students will:

- Understand how to connect Node.js to MongoDB using the native MongoDB driver.
- Insert and retrieve documents from a MongoDB collection.
- Connect to a secured MongoDB instance using authentication.

🧰 Prerequisites:

- Node.js and npm installed
- MongoDB installed and running locally
- Basic knowledge of JavaScript and MongoDB
- A code editor (e.g., VS Code)

🧭 Part 1: Connecting Node.js to MongoDB Without Authentication

🔧 Step 1: Project Setup

Create a project folder:

```
mkdir node-mongo-lab
cd node-mongo-lab
```

1.

Initialize the project:

```
npm init -y
```

2.

Install the MongoDB driver:

```
npm install mongodb
```

3.

💻 Step 2: Write Connection Code

1. Create a new file named `app.js`.
2. Copy and paste the following code:

```
const { MongoClient } = require('mongodb');

const url = 'mongodb://localhost:27017';
const dbName = 'myDatabase';

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

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

    const db = client.db(dbName);
    const users = db.collection('users');

    await users.insertOne({ name: 'Alice', age: 30 });

    const allUsers = await users.find().toArray();
    console.log('Users:', allUsers);
  } finally {
    await client.close();
```

```
  }
}

main().catch(console.error);
```

▶️ Step 3: Run the Code

```
node app.js
```

✅ Expected Output:

- A message confirming the MongoDB connection.
- A list of users including "Alice".

🔐 Part 2: Connecting to MongoDB With Authentication

🔐 Step 4: Enable MongoDB Authentication (Optional for instructor setup)

> Skip if your MongoDB instance already uses authentication.

Create a MongoDB admin user:

```
use admin
db.createUser({
  user: "myUser",
  pwd: "mySecurePassword",
  roles: [{ role: "readWrite", db: "myDatabase" }]
})
```

1.

Restart MongoDB with authentication enabled:

```
mongod --auth --dbpath /path/to/your/db
```

2.

🧪 Step 5: Connect Using Authentication

1. Create a new file `secureApp.js`.
2. Copy and paste the following code:

```
const { MongoClient } = require('mongodb');

const uri =
'mongodb://myUser:mySecurePassword@localhost:27017/myDatabase?authSour
ce=admin';

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

  try {
    await client.connect();
    console.log('Connected successfully with authentication');

    const db = client.db('myDatabase');
    const users = db.collection('users');

    const docs = await users.find().toArray();
    console.log(docs);
  } catch (err) {
    console.error('Connection failed:', err);
  } finally {
    await client.close();
  }
}

main();
```

▶️ Step 6: Run the Authenticated Connection

```
node secureApp.js
```

✅ Expected Output:

- Confirmation of a successful connection with authentication
- A list of user documents

🧠 Lab Questions for Reflection

1. What happens if you try to connect with wrong credentials?
2. How would you modify the code to update a user's age?
3. What security concerns should you consider when storing credentials?

✅ Lab Completion Checklist

- Installed `mongodb` Node.js driver
- Connected to MongoDB without authentication
- Inserted and queried user data
- Created a MongoDB user for authentication
- Connected with authentication
- Answered reflection questions