JavaScript implementation of MongoDB data models:

**1. Project Setup**

* **Install MongoDB:** Follow the instructions on the official website.
* **Install Node.js and npm:** Download and install Node.js from the official website. This includes npm (Node Package Manager).
* **Create a Project Directory:** Make a new directory for your project.
* **Initialize npm:** In your project directory, run npm init -y to create a package.json file.
* **Install MongoDB Driver:** Install the MongoDB driver for Node.js: npm install mongodb

**2. Define Data Models (using JavaScript objects)**

JavaScript

const studentSchema = {

\_id: { type: String, required: true }, // Using String for simplicity

name: { type: String, required: true },

age: { type: Number, min: 0 },

grades: { type: Array, of: Number },

courses: { type: Array, of: String }

};

const courseSchema = {

\_id: { type: String, required: true },

name: { type: String, required: true },

description: { type: String },

instructor: { type: String }

};

**3. Connect to MongoDB**

JavaScript

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

const uri = "mongodb://localhost:27017/"; // Replace with your connection string

const client = new MongoClient(uri);

async function run() {

try {

await client.connect();

console.log('Connected to MongoDB');

// ... your database operations here ...

} finally {

// Ensures that the client will close when you finish/error

await client.close();

}

}

run().catch(console.dir);



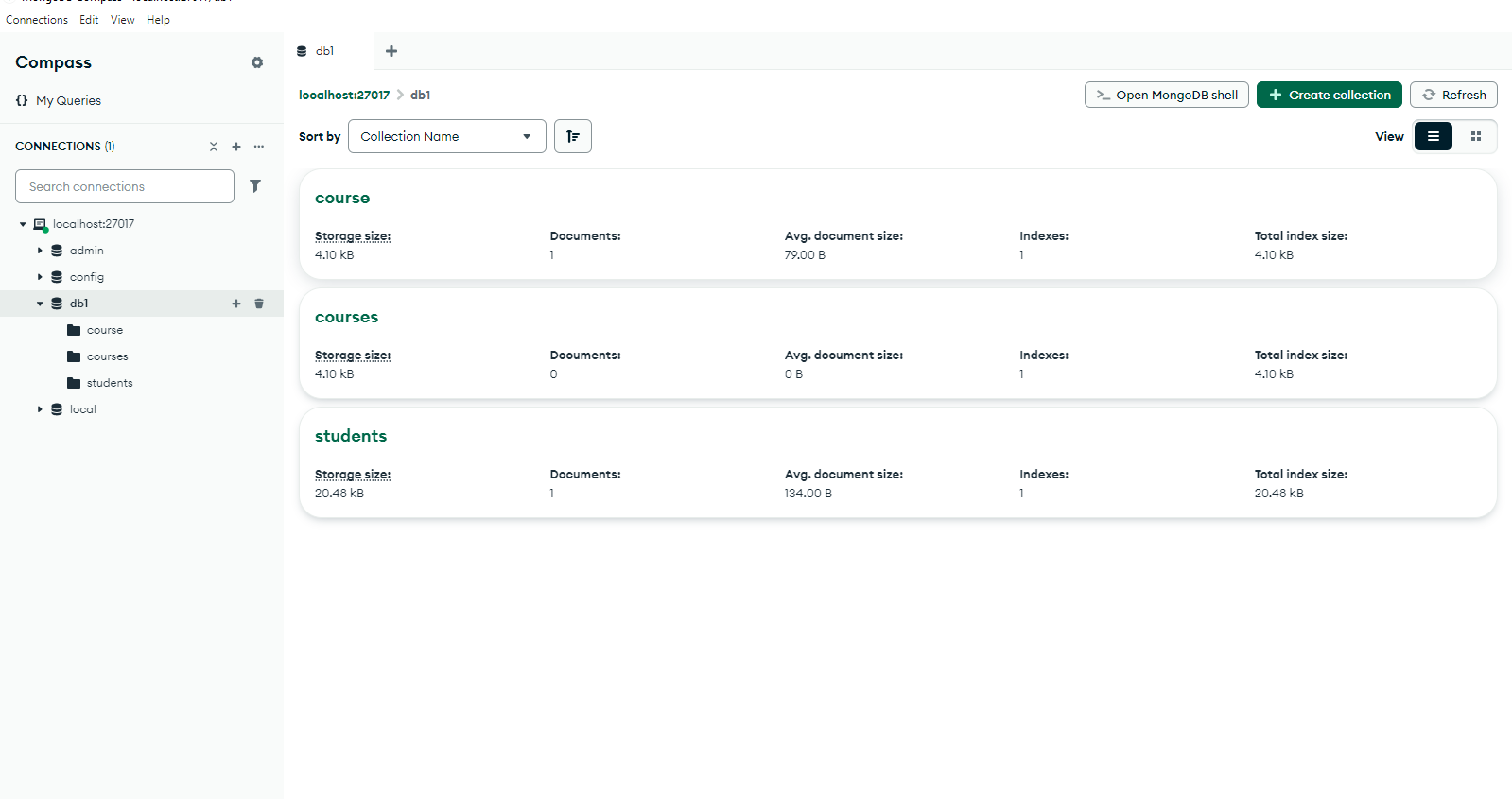
**4. Create Collections**

JavaScript

const db = client.db('your\_database\_name');

const studentsCollection = db.collection('students');

const coursesCollection = db.collection('courses');



**5. Insert Data**

JavaScript

const newStudent = {

\_id: '1',

name: 'John Doe',

age: 20,

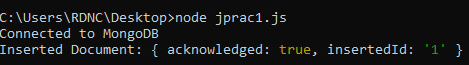
grades: [90, 85, 92],

courses: ['Math', 'Science', 'History']

};

const result = await studentsCollection.insertOne(newStudent);

console.log('Inserted Document:', result);



**6. Read Data**

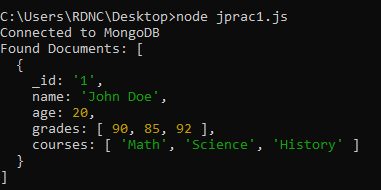
JavaScript

const query = { age: { $gte: 20 } };

const cursor = studentsCollection.find(query);

const results = await cursor.toArray();

console.log('Found Documents:', results);



**7. Update Data**

JavaScript

const filter = { name: 'John Doe' };

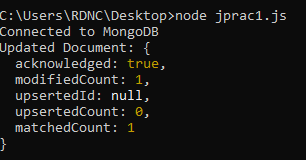
const updateDoc = {

$set: { age: 21 }

};

const result = await studentsCollection.updateOne(filter, updateDoc);

console.log('Updated Document:', result);



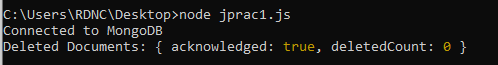
**8. Delete Data**

JavaScript

const query = { age: { $lt: 18 } };

const result = await studentsCollection.deleteMany(query);

console.log('Deleted Documents:', result);



**9. Close Connection**

* The connection is closed automatically in the finally block.

**Tools and Notes for Students**

* **MongoDB Compass:** Use the GUI for visualizing data and performing basic operations.
* **Node.js REPL:** Use the interactive console to experiment with code snippets.
* **Focus on:**
  + Data modeling principles.
  + CRUD operations.
  + Basic query operators (e.g., $gt, $lt, $in, $regex).
  + Data validation and sanitation.

**Key Considerations**

* **Error Handling:** Implement proper error handling to catch potential issues (e.g., connection errors, invalid data).
* **Asynchronous Operations:** Use async/await or promises to handle asynchronous operations effectively.
* **Security:** Always use appropriate security measures (e.g., authentication, authorization) to protect your MongoDB data.