

Jenel Ricafrente

BSI/T-3E

## Individual Performance Test #1

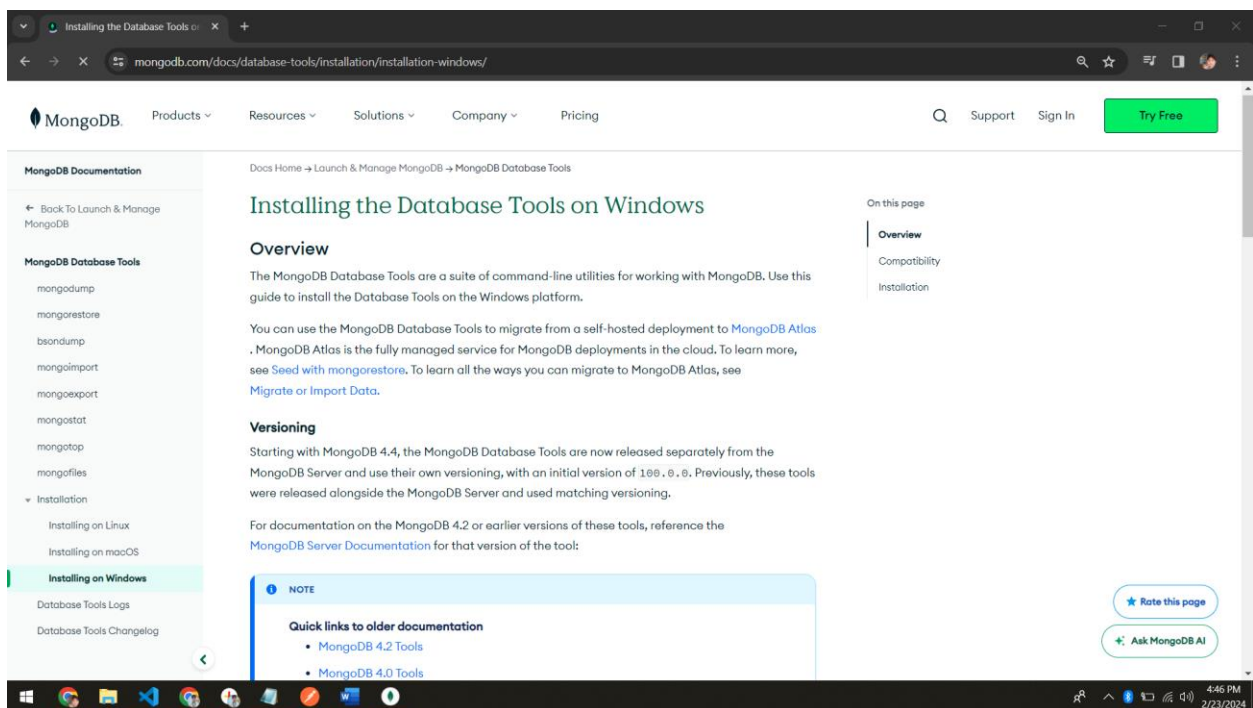
### Backend Course Data API Test Documentation

#### Test Procedure

##### Step 1: Setting Up the Environment

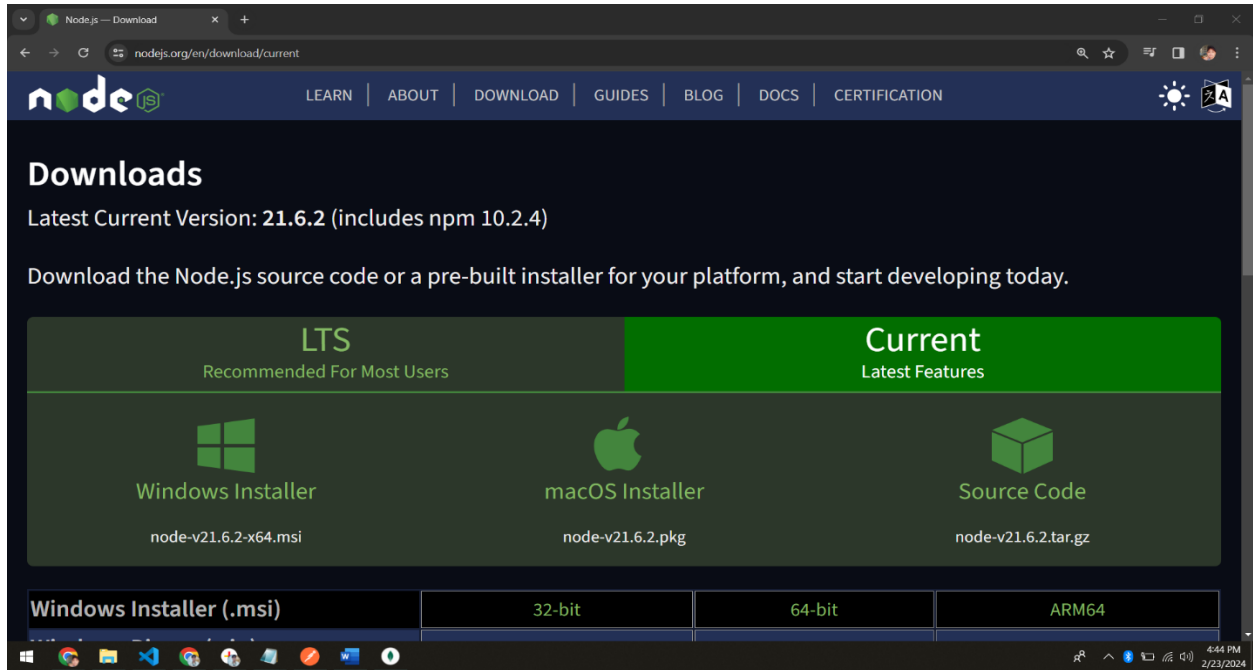
###### 1. Install MongoDB Database Tool:

- Downloaded and installed the MongoDB Database Tool from [this link]([https://fastdl.mongodb.org/tools/db/mongodb-database-tools-windows-x86\\_64-100.9.4.zip](https://fastdl.mongodb.org/tools/db/mongodb-database-tools-windows-x86_64-100.9.4.zip)).



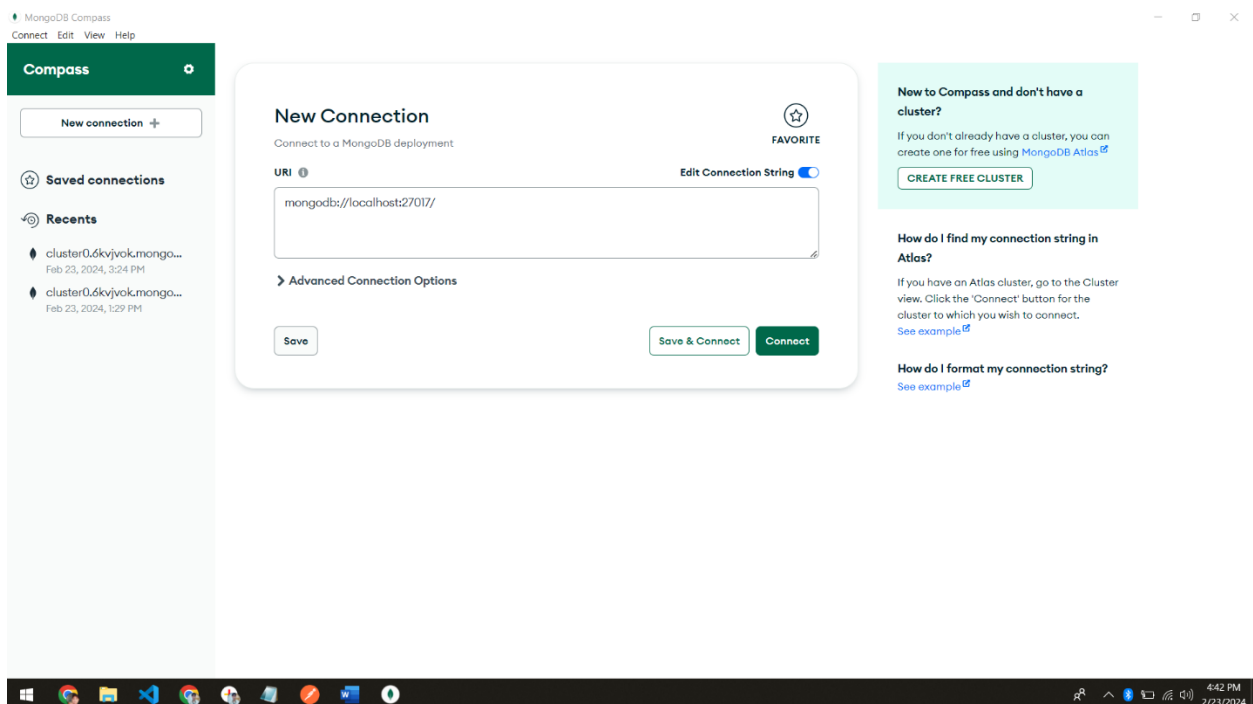
## 2. Installed Node.js:

- Ensured Node.js is installed on the local machine by downloading it from [https://nodejs.org/](https://nodejs.org/).



## 3. MongoDB Server:

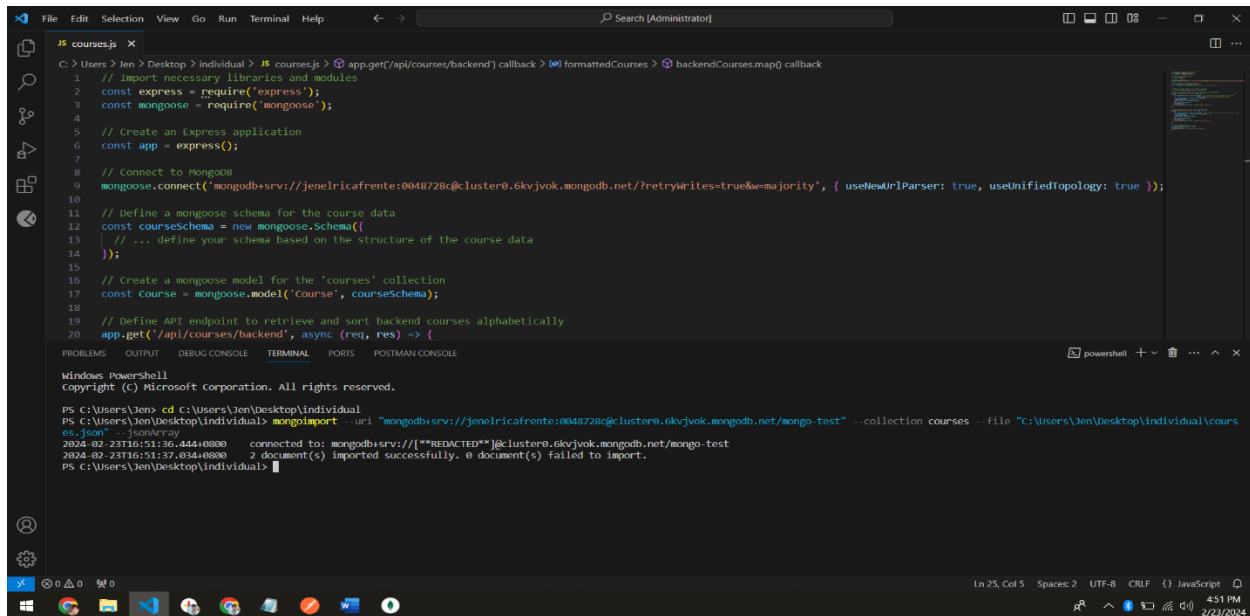
- Confirmed that the MongoDB server is installed and running locally.



## Step 2: Import Course Data

- Used the 'mongoimport' command to import the provided course data into MongoDB:

mongoimport --db mongo-test --collection courses --file courses.json --jsonArray



The screenshot shows a Visual Studio Code editor with a file named `courses.js` and a PowerShell terminal window. The `courses.js` file contains the following code:

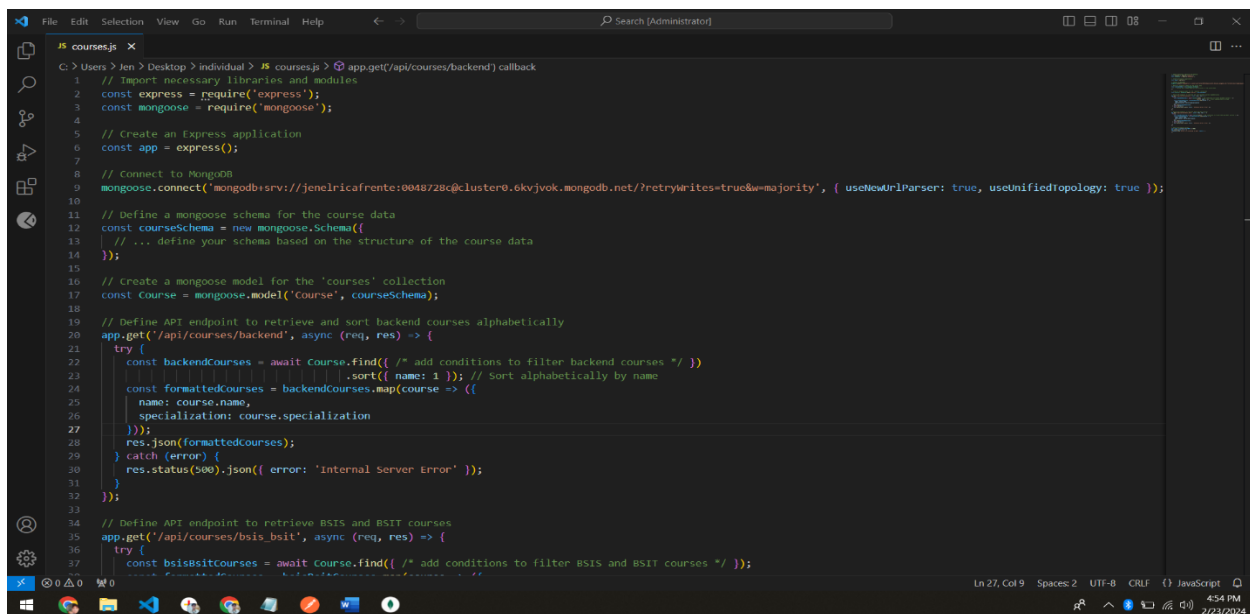
```
1 // Import necessary libraries and modules
2 const express = require('express');
3 const mongoose = require('mongoose');
4
5 // Create an Express application
6 const app = express();
7
8 // Connect to MongoDB
9 mongoose.connect('mongodb://jnelricafrante:0048728c@cluster0.6kvjvk.mongodb.net/?retryWrites=true&majority', { useNewUrlParser: true, useUnifiedTopology: true });
10
11 // Define a mongoose schema for the course data
12 const courseSchema = new mongoose.Schema({
13   // ... define your schema based on the structure of the course data
14 });
15
16 // Create a mongoose model for the 'courses' collection
17 const Course = mongoose.model('Course', courseSchema);
18
19 // Define API endpoint to retrieve and sort backend courses alphabetically
20 app.get('/api/courses/backend', async (req, res) => {
```

The PowerShell terminal window shows the execution of the `mongoimport` command:

```
PS C:\Users\Jen\Desktop\individual> mongoimport --uri "mongodb://jnelricafrante:0048728c@cluster0.6kvjvk.mongodb.net/mongo-test" --collection courses --file "C:\Users\Jen\Desktop\individual\courses.json" --jsonArray
2024-02-23T16:51:26.444+0800    connected to: mongodb://jnelricafrante:0048728c@cluster0.6kvjvk.mongodb.net/mongo-test
2024-02-23T16:51:37.034+0800    > document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\Jen\Desktop\individual>
```

## Step 3: Backend API Implementation

- Created a Node.js application using Express to implement backend API endpoints.



The screenshot shows the `courses.js` file with the implementation of the `/api/courses/backend` API endpoint. The code is as follows:

```
21   try {
22     const backendCourses = await Course.find({ /* add conditions to filter backend courses */ })
23       .sort({ name: 1 }); // Sort alphabetically by name
24     const formattedCourses = backendCourses.map(course => ({
25       name: course.name,
26       specialization: course.specialization
27     }));
28     res.json(formattedCourses);
29   } catch (error) {
30     res.status(500).json({ error: 'Internal Server Error' });
31   }
32 });
33
34 // Define API endpoint to retrieve BSIS and BSIT courses
35 app.get('/api/courses/bsis_bsit', async (req, res) => {
36   try {
37     const bsisBsittCourses = await Course.find({ /* add conditions to filter BSIS and BSIT courses */ });
38     res.json(bsisBsittCourses);
39   } catch (error) {
40     res.status(500).json({ error: 'Internal Server Error' });
41   }
42 });
```

#### Step 4: API Endpoint Testing

##### 1. Retrieve and Sort Backend Courses:

- Sent a GET request to `/api/courses/backend`.
- Verified that the response contains an alphabetically sorted list of published backend courses.

##### 2. Retrieve BSIS and BSIT Courses:

- Sent a GET request to `/api/courses/bsis\_bsit`.
- Verified that the response contains a list of published BSIS and BSIT courses.

#### Step 5: Data Validation

- Ensured data validation at each step:
  - Validated the structure of the course data during import.
  - Implemented Mongoose schema validation to ensure correctness.

#### Step 6: Documentation

- Documented the entire test procedure in this document.

#### Challenges Faced and Solutions Implemented

##### 1. Challenge: MongoDB Connection Issues

- Encountered issues connecting to the MongoDB server.

##### Solution:

- Verified the MongoDB server status and ensured the correct connection string in the application code.

##### 2. Challenge: Incorrect Data Format in Imported JSON

- The provided JSON file format was not directly compatible with the Mongoose schema.

Solution:

- Modified the Mongoose schema to match the structure of the imported JSON data.

### 3. Challenge: Data Not Sorting Correctly

- Initial implementation did not sort backend courses alphabetically.

Solution:

- Modified the MongoDB query to include sorting by the course name.

### 4. Challenge: Error Handling

- Implemented proper error handling to return meaningful responses in case of failures.