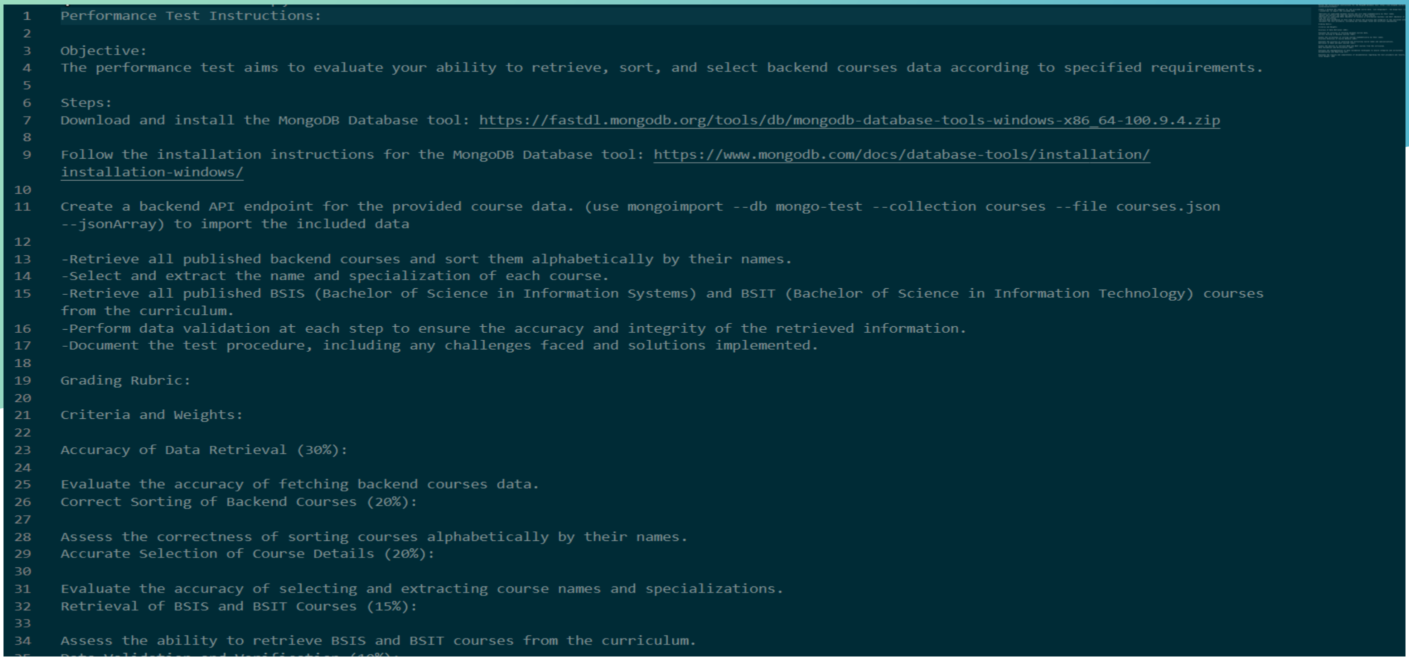
Kamil P. Saporna
BSI/T 3F

February 23, 2024

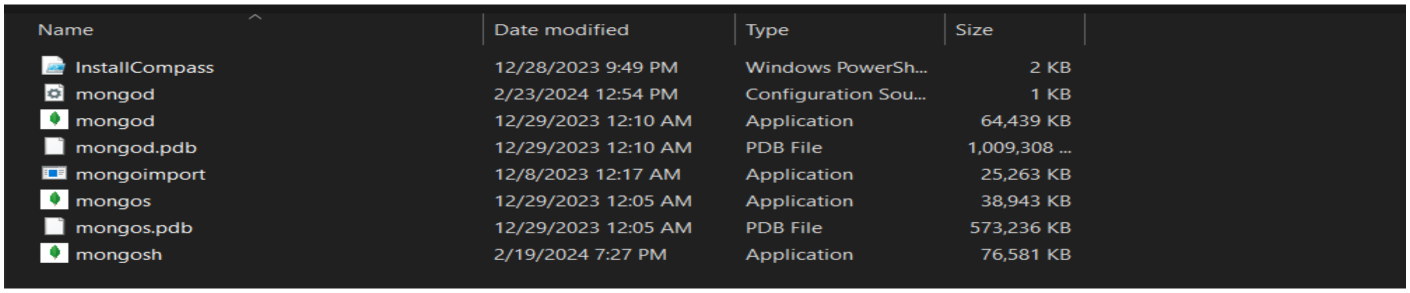
Individual Performance Test #1- Documentation

Instructions:



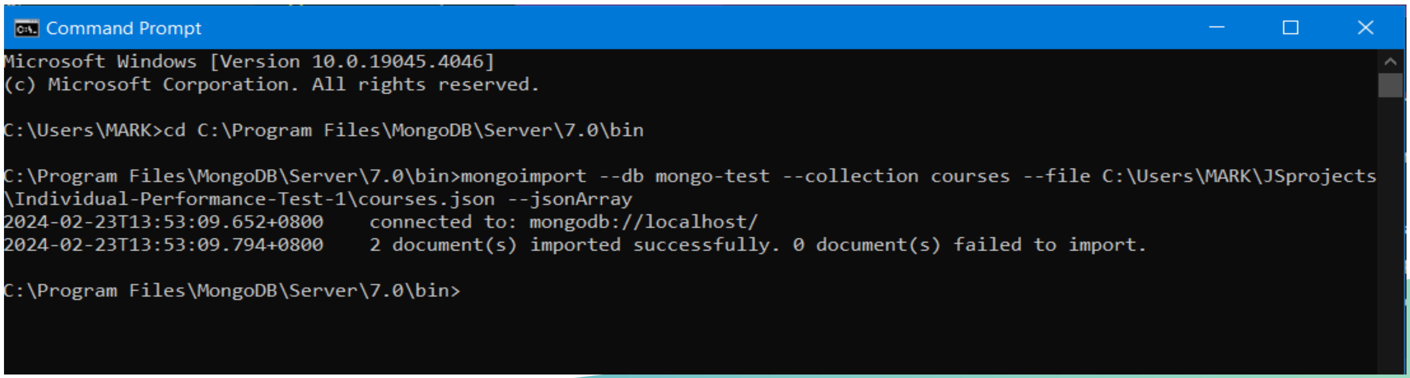
```
Performance Test Instructions:

Objective:
The performance test aims to evaluate your ability to retrieve, sort, and select backend courses data according to specified requirements.

Steps:
Download and install the MongoDB Database tool: https://fastdl.mongodb.org/tools/db/mongodb-database-tools-windows-x86_64-100.9.4.zip

Follow the installation instructions for the MongoDB Database tool: https://www.mongodb.com/docs/database-tools/installation/installation-windows/

Create a backend API endpoint for the provided course data. (use mongoimport --db mongo-test --collection courses --file courses.json --jsonArray) to import the included data

-Retrieve all published backend courses and sort them alphabetically by their names.
-Select and extract the name and specialization of each course.
-Retrieve all published BSIS (Bachelor of Science in Information Systems) and BSIT (Bachelor of Science in Information Technology) courses from the curriculum.
-Perform data validation at each step to ensure the accuracy and integrity of the retrieved information.
-Document the test procedure, including any challenges faced and solutions implemented.

Grading Rubric:

Criteria and Weights:

Accuracy of Data Retrieval (30%):

Evaluate the accuracy of fetching backend courses data.
Correct Sorting of Backend Courses (20%):

Assess the correctness of sorting courses alphabetically by their names.
Accurate Selection of Course Details (20%):

Evaluate the accuracy of selecting and extracting course names and specializations.
Retrieval of BSIS and BSIT Courses (15%):

Assess the ability to retrieve BSIS and BSIT courses from the curriculum.
```

MongoDb Community Edition & MongoDB Database Tool Installed:



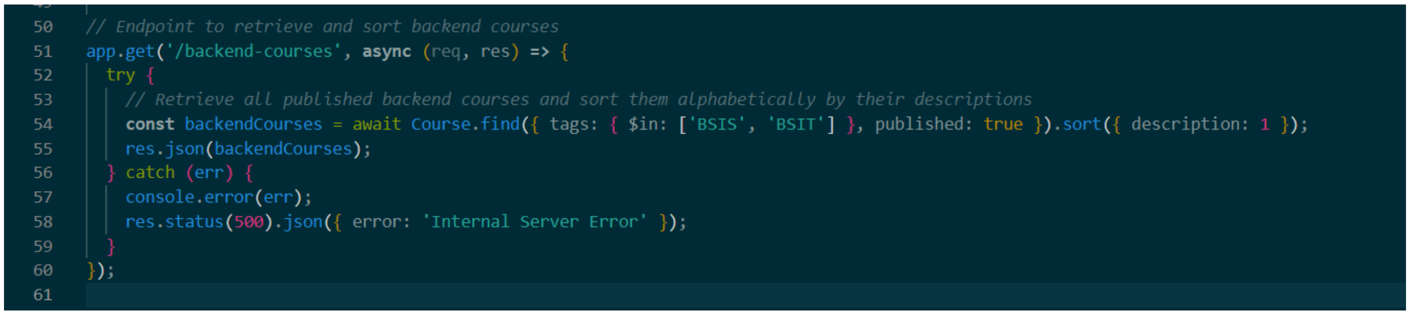
Name	Date modified	Type	Size
InstallCompass	12/28/2023 9:49 PM	Windows PowerSh...	2 KB
mongod	2/23/2024 12:54 PM	Configuration Sou...	1 KB
mongod	12/29/2023 12:10 AM	Application	64,439 KB
mongod.pdb	12/29/2023 12:10 AM	PDB File	1,009,308 ...
mongoimport	12/8/2023 12:17 AM	Application	25,263 KB
mongos	12/29/2023 12:05 AM	Application	38,943 KB
mongos.pdb	12/29/2023 12:05 AM	PDB File	573,236 KB
mongosh	2/19/2024 7:27 PM	Application	76,581 KB

Importing JSON file using mongoimport Command:

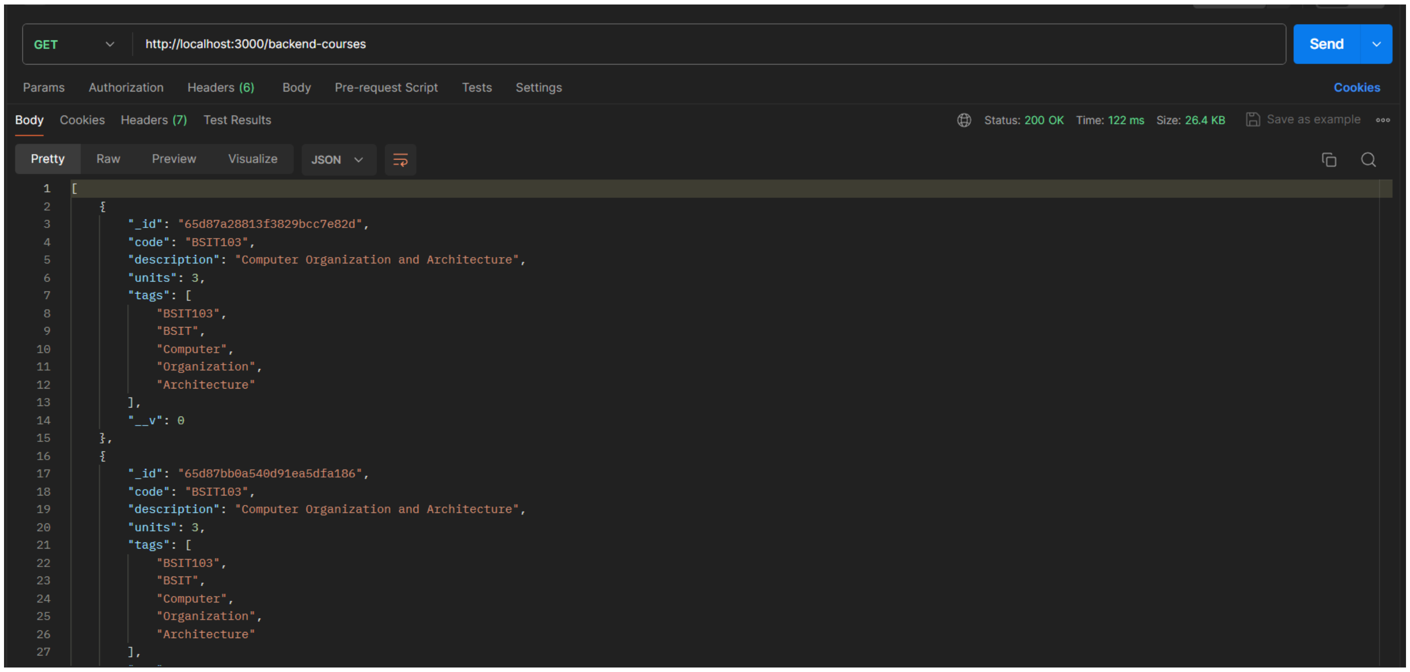


```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MARK>cd C:\Program Files\MongoDB\Server\7.0\bin

C:\Program Files\MongoDB\Server\7.0\bin>mongoimport --db mongo-test --collection courses --file C:\Users\MARK\JSprojects
\Individual-Performance-Test-1\courses.json --jsonArray
2024-02-23T13:53:09.652+0800    connected to: mongodb://localhost/
2024-02-23T13:53:09.794+0800    2 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\7.0\bin>
```

Retrieve all published backend courses and sort them alphabetically by their names:



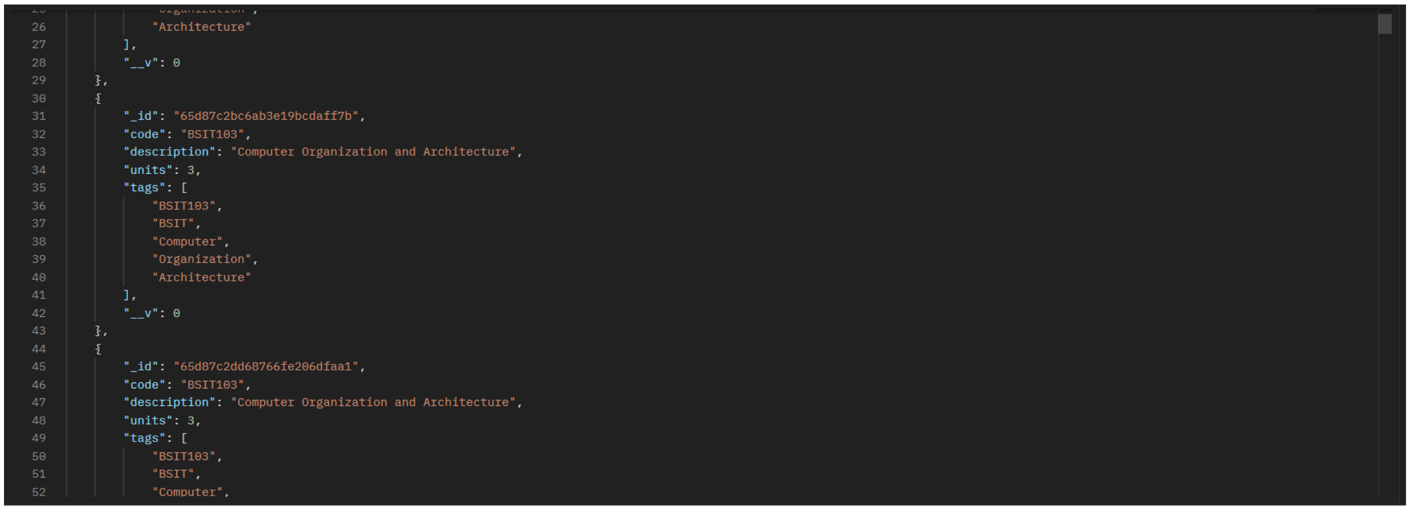
```
// Endpoint to retrieve and sort backend courses
app.get('/backend-courses', async (req, res) => {
  try {
    // Retrieve all published backend courses and sort them alphabetically by their descriptions
    const backendCourses = await Course.find({ tags: { $in: ['BSIS', 'BSIT'] }, published: true }).sort({ description: 1 });
    res.json(backendCourses);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```



GET http://localhost:3000/backend-courses

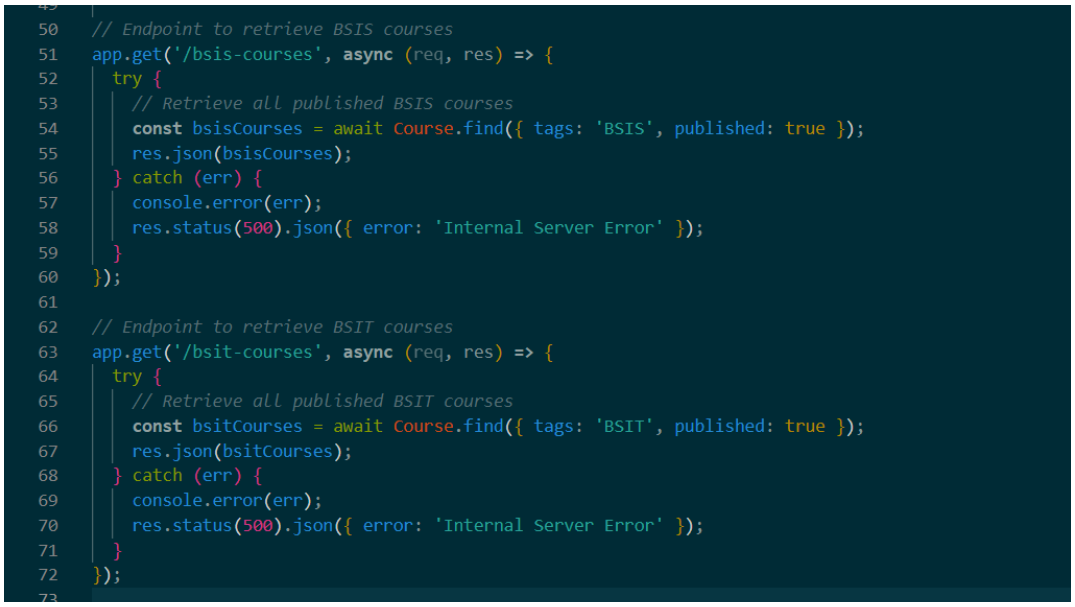
Status: 200 OK Time: 122 ms Size: 26.4 KB

```
[
    {
        "_id": "65d87a28813f3829bcc7e82d",
        "code": "BSIT103",
        "description": "Computer Organization and Architecture",
        "units": 3,
        "tags": [
            "BSIT103",
            "BSIT",
            "Computer",
            "Organization",
            "Architecture"
        ],
        "__v": 0
    },
    {
        "_id": "65d87bb0a540d91ea5dfa186",
        "code": "BSIT103",
        "description": "Computer Organization and Architecture",
        "units": 3,
        "tags": [
            "BSIT103",
            "BSIT",
            "Computer",
            "Organization",
            "Architecture"
        ],
```



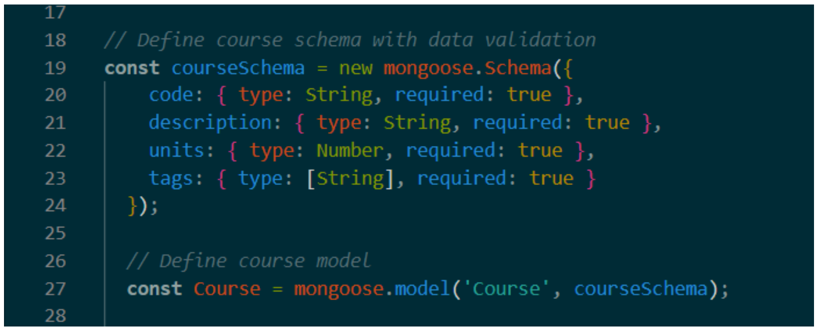
```
            "Architecture"
        ],
        "__v": 0
    },
    {
        "_id": "65d87c2bc6ab3e19bcdaff7b",
        "code": "BSIT103",
        "description": "Computer Organization and Architecture",
        "units": 3,
        "tags": [
            "BSIT103",
            "BSIT",
            "Computer",
            "Organization",
            "Architecture"
        ],
        "__v": 0
    },
    {
        "_id": "65d87c2dd68766fe206dfaa1",
        "code": "BSIT103",
        "description": "Computer Organization and Architecture",
        "units": 3,
        "tags": [
            "BSIT103",
            "BSIT",
            "Computer",
```

Retrieve all published BSIS (Bachelor of Science in Information Systems) and BSIT (Bachelor of Science in Information Technology) courses from the curriculum:



```
// Endpoint to retrieve BSIS courses
app.get('/bsis-courses', async (req, res) => {
  try {
    // Retrieve all published BSIS courses
    const bsisCourses = await Course.find({ tags: 'BSIS', published: true });
    res.json(bsisCourses);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

// Endpoint to retrieve BSIT courses
app.get('/bsit-courses', async (req, res) => {
  try {
    // Retrieve all published BSIT courses
    const bsitCourses = await Course.find({ tags: 'BSIT', published: true });
    res.json(bsitCourses);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```

Perform data validation at each step to ensure the accuracy and integrity of the retrieved information:



```
 // Define course schema with data validation
 const courseSchema = new mongoose.Schema({
     code: { type: String, required: true },
     description: { type: String, required: true },
     units: { type: Number, required: true },
     tags: { type: [String], required: true }
  });

  // Define course model
  const Course = mongoose.model('Course', courseSchema);
```

Challenges Faced and Solutions Implemented:

1. Installing and configuring MongoDB could have been a challenge, especially if you were unfamiliar with the process. We need follow the installation instructions carefully, and refer to MongoDB documentation or online tutorials for assistance if needed.

2. Issues connecting to the MongoDB database from your Node.js application may have arisen due to incorrect connection strings or network/firewall issues. We have to double-check the connection string and ensure that MongoDB is running and accessible from your application. Troubleshoot network/firewall settings if necessary.

3. Errors in writing MongoDB queries could have led to incorrect or empty results. We must review the query logic, ensure that the field names and values are correct, and test queries directly in the MongoDB shell to verify their correctness.