John Kenneth Fetizanan 02/23/2024

BSIT 3E

Performance Test 1

The objective of this performance test is to evaluate the ability to retrieve, sort, and select backend courses data according to specified requirements.

Test procedure:

1. Install the MongoDB database tool
2. Import the courses.json into MongoDB
3. Creating API endpoint for retrieving backend courses alphabetically

This is the endpoint for retrieving backend courses alphabetically.

http://localhost:4000/api/backend-courses/

And here is the function I use for backend courses.

And here is output of all the published backend courses.

4. Select and extract the name and specialization of each course

```
// Function to extract name and specialization of each course
const extractCourseDetails = () => {
  const courseDetails = [];
  coursesData.forEach(year => {
    Object.values(year).forEach(courseList => {
      courseList.forEach(course => {
        // Extract name and specialization and add to courseDetails array
        const { description, tags } = course;
        const name = tags[0];
        const specialization = tags[1];
        courseDetails.push({ name, specialization });
      });
    });
  });

  return courseDetails;
};
const extractedDetails = extractCourseDetails();
console.log(extractedDetails);
```

Here is the function I use to Select and extract the name and specialization of each course

```
// Define the endpoint name and specialization
app.get('/api/course-details', (req, res) => {
  try {
    // Call the function to extract course details
    const extractedDetails = extractCourseDetails();
    res.json(extractedDetails);
  } catch (error) {
    console.error('Error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

Here is the endpoint I use:

http://localhost:4000/api/course-details

Pretty Raw Preview Visualize JSON

```
[
    {
        "name": "BSIS101",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS102",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS103",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS201",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS202",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS203",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS301",
        "specialization": "BSIS"
    },
    {
        "name": "BSIS302",
```

Here is the extracted the name and specialization of each course.

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

This is the endpoint to retrieve all the published BSIS and BSIT courses.

http://localhost:4000/api/courses

This is the retrieved BSIS and BSIT courses

This is the endpoint to retrieve all the published BSIS courses.

http://localhost:4000/api/courses/bsis

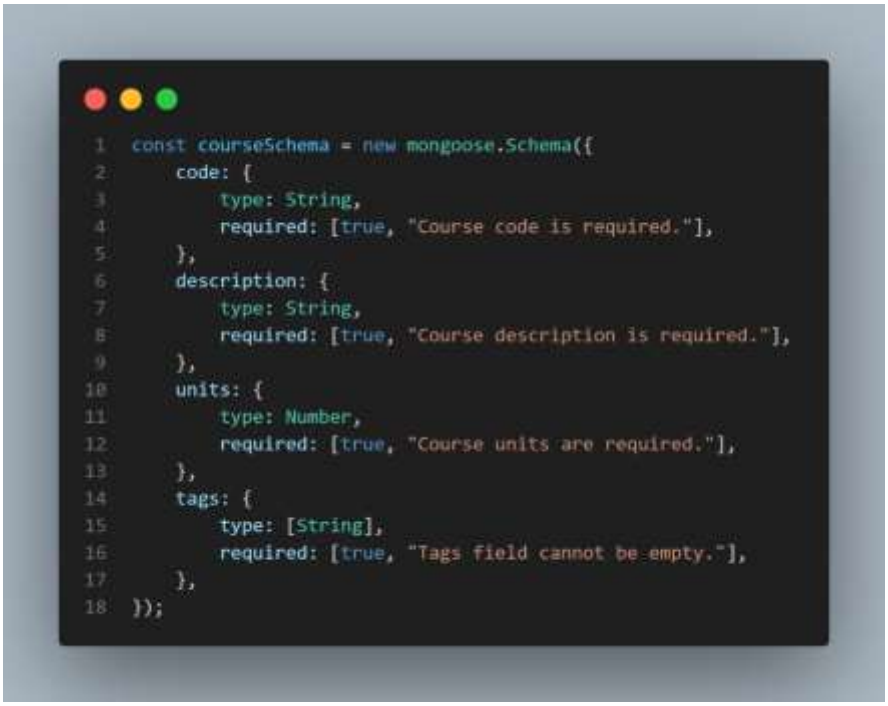
This is the retrieved BSIS courses.

This is the endpoint to retrieve all the published BSIS courses.

http://localhost:4000/api/courses/bsit

This is the retrieved BSIT courses.

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



```
const courseSchema = new mongoose.Schema({
    code: {
        type: String,
        required: [true, "Course code is required."],
    },
    description: {
        type: String,
        required: [true, "Course description is required."],
    },
    units: {
        type: Number,
        required: [true, "Course units are required."],
    },
    tags: {
        type: [String],
        required: [true, "Tags field cannot be empty."],
    },
});
```

This is the validation I made to helps maintaining the integrity and consistency of the data by enforcing rules such as required fields, data types, and custom validation logic.

During the development process, I encountered several challenges. First of all, I had trouble grasping MongoDB's concepts and effectively using the database because of my inadequate knowledge of it. I set aside time for online resources and documentation-based self-learning in order to go past this. Second, the requirement to comprehend MongoDB's sorting methods and incorporate sorting logic in Express.js API endpoints made it difficult to create capability to get data alphabetically. Even yet, getting over them improved my ability to solve problems and broadened my knowledge of backend programming techniques.