Jennelyn L. Encarnacion

BSI/T 3E

DOCUMENTATION

This backend API provides endpoints to interact with course data stored in a MongoDB database. It allows users to retrieve information about courses, including backend courses and BSIS/BSIT courses.

To test the code thoroughly, follow these steps:

1. **Installation and set up**
    - Download and install the MongoDB Database tool
    - Follow the installation instructions for the MongoDB Database tool
    - Download the provided code.
    - Once MongoDB tool is installed, import the provided course data into MongoDB using the command **mongoimport --db mongo-test --collection courses --file courses.json --jsonArray.** This command imports the data from the **courses.json** file into a collection named courses in the **mongo-test** database.
    - Ensure you have Node.js installed on your system.
    - Ensure that MongoDB is running locally on port 27017.
    - Create a new directory for the project and navigate to it in the terminal.
    - Run **npm init -y** to create a package.json file and follow the prompts.
    - Install the required dependencies like **Express.js**, **mongoose**, and **fs** using the command **npm install.**
    - Start the server using the command **node app.js.** By default, the server runs on port 3220.

2. **API Endpoints**

- Once it is running in the port successfully, you may now retrieve all Published Backend courses, using Get.
- Type the url: http://localhost:3220/backend-courses to retrieves all published backend courses sorted alphabetically by their names, since my endpoint is **GET /backend-courses.**
- Using the url: http://localhost:3220/bsis-bsit-courses to retrieves all published BSIS (Bachelor of Science in Information Systems) and BSIT (Bachelor of Science in Information Technology) courses from the curriculum, since my endpoints is **GET/bsis-bsit-courses.**

3. **Data Validation**
    - Validation is conducted at every stage to guarantee the precision and reliability of the acquired data. Course data is stored in a MongoDB database, and Mongoose serves as an Object Data Modeling (ODM) tool for defining schemas and managing interactions with the database.

4. **Error Handling**
    - Error handling is implemented to catch and log any errors that occur during database operations or API requests. Proper HTTP status codes are returned along with error messages to indicate the nature of the problem.

5. **Challenges**

In completing this activity, I encountered several challenges, especially since I had limited knowledge about MongoDB and Node.js. Connecting to MongoDB alone took me hours to figure out, as I struggled to understand how it works. Node.js proved to be the most difficult to grasp for me. Since I couldn't fully comprehend it, I resorted to searching online and tried my best to understand it.

After finally connecting and coding in the app.js file, I encountered another issue when trying to retrieve data in Postman. The result was always an empty array "[ ]", indicating that the data in courses.js couldn't be read. To solve this, I first checked if it was connected to the MongoDB database, which it was. This meant that wasn't the problem. I spent hours trying to figure out what the issue was, even attempting to change my code, but the result was the same.

Frustrated, I decided to take a break and come back to it later. After some time, I sought help from a classmate who had already completed the task. It turned out that my problem was with the tags, I was using the wrong tags. Once I corrected this, the code ran successfully. These challenges really tested my patience, and I spent a long time trying to figure out how to display the result in Postman. Sometimes, it's really important not to be afraid to ask for help.