Todd Gavin
10 February 2023
DSCI551: Foundations of Data Management

Final Project Proposal

Proposal (5 points, due on 2/10, Friday)

- Project: Firebase Emulator using Flask, WebSockets, and MongoDB
- Team: Solo Project
- Implementation:

1. Set up the development environment:
    a. Install the required packages and tools, such as Flask, WebSockets, and MongoDB.
    b. Create a new project directory and configure the development environment to match your needs.
    c. Set up a virtual environment to isolate the dependencies for your project.
2. Design the database:
    a. Decide on the structure of your database and how it will store JSON data.
    b. For example, you can use MongoDB to store the data as documents.
    c. Design the schema for your database, including the fields and data types for each document.
3. Implement the RESTful API:
    a. Create the routes and endpoints for your RESTful API, including functions for PUT, GET, POST, PATCH, DELETE, and filtering functions like orderBy, limitToFirst/Last, equalTo, startAt/endAt.
    b. Develop the functions for each of these operations, for example:
        i. PUT: A function to update an existing document in the database.
        ii. GET: A function to retrieve a document from the database.
        iii. POST: A function to create a new document in the database.
        iv. PATCH: A function to partially update a document in the database.
        v. DELETE: A function to delete a document from the database.
        vi. Filtering functions: Functions to filter documents based on the query parameters provided in the URL, such as orderBy, limitToFirst/Last, equalTo, startAt/endAt.
4. Connect to the database:
    a. Integrate your Flask application with MongoDB.
    b. Write the code to connect to the database, perform CRUD operations, and retrieve data.
    c. Use the PyMongo library to interact with MongoDB from your Flask application.
5. Implement indexing:
    a. Create indexes on the relevant fields in the database to support the orderBy function.

    b. For example, create an index on the 'name' field for orderBy=”name”.
    c. This will ensure that querying the database for documents ordered by the 'name' field is efficient.
6. Implement the command-line interface:
    a. Create a command-line interface using curl commands.
    b. Develop functions to parse the curl commands and translate them into the appropriate RESTful API calls.
    c. For example, the curl command "curl -X GET 'http://localhost:5000/users.json?orderBy="name"&limitToFirst=5'" could be translated into a GET request to the '/users.json' endpoint with query parameters for 'orderBy' and 'limitToFirst'.
7. Test the system:
    a. Test the system thoroughly to ensure that it meets the requirements.
    b. Test the RESTful API, the command-line interface, and the database to make sure that everything works as expected.
    c. Use a testing framework, such as unit test or pytest, to automate the testing process.
8. Deploy the system:
    a. Deploy the system on a suitable platform, such as a cloud-based virtual machine or a dedicated server.
    b. Make sure that the system is secure, scalable, and performant.
    c. Consider using a cloud provider, such as AWS or Google Cloud, to deploy the system.
9. Monitor and maintain the system:
    a. Monitor the system to make sure that it is running smoothly.

- Timeline:
    - Project Proposal (5 points, due on 2/10, Friday)
    - Midterm progress report (5 points, due on 3/27, Monday)
    - Final report with Code (10 points, due on 4/28, Friday)
    - Demo (5 points, in class in the week of 4/24)
    - Video (5 points, about 20 minutes, due on 4/28, Friday)