Project Documentation: UMS (User Management System)

Project Overview

The UMS project aims to create a backend system for fetching user data from the Random User Generator API, processing and storing it in Redis for queue management and caching, and finally storing the processed data in MongoDB. A cron job is implemented to handle this process every minute or every 3 1/2 hours based on the `MINUTE_MODE` environment variable.

Technology Stack

- Nest.js (with TypeScript)
- MongoDB
- Redis

Api endpoints

- Application doesn't require any API endpoint.

Other references

- Endpoint has an optional param results where we have to give user data count. this has been set up in the .env file.
- Added cron execution information by epoch time. More on Readme.md

Task Steps Completed

1. **Set Up Project:**
    - Created a Node.js project using TypeScript.
    - Installed necessary dependencies including Redis and MongoDB connectors.

2. **Fetch User Data:**
    - Implemented a module to fetch 10 user data from the Random User Generator API.
3. **Redis Queue:**
    - Used Redis as a queue to store the fetched user data.
    - Implemented a mechanism to push 10 user data into the Redis queue daily.
4. **Worker Process:**
    - Implemented a worker script to process data from the Redis queue.
    - Upon processing, stored the data in MongoDB.
5. **MongoDB Schema:**
    - Designed a MongoDB schema to store user data.
    - Created a separate collection for logs.
6. **Logging:**
    - Implemented logging of request time, URL, HTTP method, response JSON, and total response time for each request.
7. **Cron Job:**
    - Set up a cron job to run every minute or every 3 1/2 hours based on the `MINUTE_MODE` environment variable.
    - Used epoch time for scheduling the cron job.
8. **Testing:**
    - Implemented unit tests to ensure the functionality of each module.
    - Included edge cases and error handling in the tests.
9. **Documentation:**
    - Provided clear documentation on how to run and test the application.
    - Included details on the cron job configuration.
    - Added information about the `.env` file and the `MINUTE_MODE` variable.
10. **Submission:**
    - Submitted the project with a README file explaining the architecture, technologies used, and instructions for running the application.
    - Published the work in the personal GitHub repository and shared the URL.

Additional Notes

- TypeScript was used for type safety and better code organization.
- Best practices for error handling and logging were followed.
- The code is well-documented and easy to understand.
- The cron job logic is based on epoch time to run every minute or every 3 1/2 hours depending on the `MINUTE_MODE` environment variable.

To set up the project, please refer to the Readme.md file. Additionally, ensure to configure the `.env` file with the provided information, including the `MINUTE_MODE` variable.