Backend queue system test

Task:

Develop a backend service that allows multiple users to submit jobs, tracks their progress, and assigns them to job processors for completion.

Requirements:

1. Job Submission: The backend service should accept job submissions from users. Each job submission should include the user ID, job ID, and job duration.
2. Job Processing: The backend service should assign jobs to job processors based on availability. A job processor can only handle one job at a time, and jobs should be processed in the order they were submitted. The backend service should keep track of the time spent on each job.
3. Job Progress Tracking: The backend service should periodically update the front-end client identified as the correct "user" of the current tasks in the queue and the time spent on any task currently processing.
4. Job Completion Notification: When a job is completed, the backend service should send a notification to the client identified as the correct "user."
5. Queue System: The queue system should be able to handle large volumes of job submissions from multiple users and ensure that jobs are processed efficiently and fairly OR a REST API to query the progress.
6. User Prioritization: The backend service should be able to prioritize certain user's tasks over others based on a set of rules, but not completely override them.

Assumptions:

1. Each job submission includes the user ID, job ID, and job duration.
2. The job processor can only handle one job at a time.
3. Jobs should be processed in the order they were submitted, unless certain user prioritization rules say otherwise.

Goals:

- Demonstrate the correct handling of the job submissions and tracking.
- Demonstrate the correct functionality of the queue system.
- Demonstrate the possibility of increasing the number of job processors to handle larger volumes of job submissions.
- Demonstrate the possibility of prioritizing certain users over others based on a set of rules.

Stretch goals:

- Use a database to store information about jobs and users.
- Implement a user authentication system to ensure that only authorized users can submit and view their jobs.
- Have the backend service scale to handle an increasing number of job processors as the volume of job submissions grows.

Code Quality, Modularity, Readability, and Documentation:

1. Code should be well-organized and modular, with clear separation of concerns.
2. The code should be easy to read and understand, with appropriate comments and documentation.
3. Any external libraries or frameworks used should be well-documented and clearly explained.
4. Code quality, modularity, readability, documentation, and performance should be prioritized over completeness.