Senior Backend Engineer at Roya Negar

This assignment aims to assess your skills in designing and implementing a robust backend system for tracking user video watch time. The goal is to collect events from a separate service that records the time a user spends watching a video. Your task is to build a backend service that processes these events and provides insights into the total movie watch time, total user watch time, and the latest moment watched by each user.

Requirements:

1. **Event Source:**
    - A separate service will generate events containing the username, movie ID, and the timestamp indicating the time the user spent watching the movie.
    - Assume that the events will be sent in real-time, and your backend should be capable of handling a high volume of events.
2. **Data Storage:**
    - Implement a data storage solution to persistently store the events. Choose an appropriate database technology for this task (e.g., relational, NoSQL).
    - Store the necessary information such as user ID, movie ID, and timestamp.
3. **Watch Time Calculation:**
    - Design a mechanism to calculate the total watch time for a specific movie and user. This involves aggregating the timestamps of events for each user and movie combination.
    - Calculate and update the total watch time for each user and movie whenever a new event is received.
4. **Latest Moment Watched:**
    - Keep track of the latest moment watched by each user for a specific movie.
    - Update this information whenever a new event is received, ensuring that it reflects the most recent timestamp.
5. **API Endpoints:**
    - Implement API endpoints to retrieve the total watch time for a specific movie, total watch time for a user across all movies, and the latest moment watched for a user in a particular movie.
6. **Scalability and Performance:**
    - Consider scalability and performance optimizations, ensuring that the backend can handle a growing number of users and events without compromising response times.
7. **Documentation:**
    - Provide clear documentation on how to set up and run your backend service.
    - Include API documentation for the endpoints you implement.

How to run the event generator:

The event generator send the events to your localhost on port 8000, run the following command to start the generator:

docker compose up

Payload Example:

```
{
  "user":"username",
  "slug": "show_slug",
  "at":4500
}
```

Submission Guidelines:

- Please submit your code along with any necessary instructions for running and testing the backend.
- Include documentation that explains your design decisions, trade-offs, and any potential improvements.
-

Acceptance Criteria:

1. **Event Handling:** The backend should successfully receive and process events from the external service.
2. **Data Storage:** Events should be persistently stored in the chosen database.
3. **Watch Time Calculation:**
    - Total watch time for a specific movie should be accurately calculated by aggregating the timestamps of events related to that movie.
    - Total watch time for a user across all movies should be accurately calculated by aggregating the timestamps of events for that user.
    - Ensure proper handling of concurrent events to prevent data inconsistencies.
4. **Latest Moment Watched:**
    - The system should accurately track the latest moment watched by each user for a specific movie.
    - Latest moment watched should be updated correctly when new events are received.
5. **API Endpoints:**
    - Implement API endpoints for:
        - Retrieving the total watch time for a specific movie.

        - Retrieving the total watch time for a user across all movies.
        - Retrieving the latest moment watched by a user for a specific movie.
6. **Scalability and Performance:**
    - The backend should be capable of handling a large number of concurrent events and users without significant performance degradation.
    - Implement performance optimizations to ensure quick response times for API endpoints.
7. **Error Handling:**
    - Implement appropriate error handling mechanisms for cases such as invalid events, database errors, or API request failures.
    - Provide meaningful error messages to assist with debugging.
8. **Readability and Maintainability:** Ensure that the code is well-organized, follows best practices, and is easy to understand.

Note: Your submission will be evaluated based on how well it meets these acceptance criteria. Feel free to provide any additional information or insights that you believe will enhance the evaluation of your work.

Feel free to contact me on rahimisajad@outlook.com should you have any questions or problems along the way.

Good luck!