Dream Games - Case Study

Backend Engineering

Description

We are developing a mobile game enjoyed by tens of millions of players globally every day. As backend engineers, we provide a Rest API to maintain the users' progress. Our systems need to be fast and secure to enhance user experience. We use Spring Boot with Java to write our backend services.

1. User Progress

1.1 Features

- A new user begins with 5,000 coins from level 1.
- Every new user is randomly assigned to one of these five countries - Turkey, the United States, the United Kingdom, France, and Germany - which cannot be changed later.
- User progress (level, coins, country) should be kept in MySQL Database, which is our main persistent storage.
- After completing each level, the user advances to the next level and receives 25 coins.

1.2 Flow and Requests

- **CreateUserRequest:** This request creates a new user, returning a unique user ID, level, coins, and country.
- **UpdateLevelRequest:** This request is sent by the client after each level completion. It updates the user's level and coins. Returns updated progress data.

2. World Cup Tournament

This time-limited competition allows users to compete against each other, represent their countries, and earn rewards.

2.1 Features

- The tournament runs daily from 00:00 to 20:00 (UTC). A new one starts automatically the next day.
- Users must be at least level 20 and pay 1,000 coins to participate.
- Each tournament group should include 5 users, one from each country.

- The competition for a group begins when five users from different countries are matched. Each level passed increases user score by 1 after the group begins.
- Users compete within their groups until the tournament ends.
- Users can claim rewards based on their rankings in their group after the tournament ends.
- Users cannot enter a new tournament if they haven't claimed their last tournament's rewards.

2.2 Leaderboard

2.2.1 Group Leaderboard

- Every tournament group has its leaderboard.
- It is sorted by the highest to lowest scores, displaying user ID, username, country, and tournament score.
- Provided leaderboard data should be real-time.
- The first-place user wins 10,000 coins, while the second-place user wins 5,000 coins.

2.2.2 Country Leaderboard

- This leaderboard shows the total scores contributed by each user competing for their respective country in a tournament.
- A country's tournament score should be the sum of the individual tournament scores of the users competing for that country in the tournament.
- The scores are sorted from highest to lowest.
- The leaderboard data includes the country name and the total tournament score.
- Provided leaderboard data should be real-time.

2.3 Flow and Requests

- **EnterTournamentRequest:** This request allows a user to join the current tournament and returns the current group leaderboard.
- **ClaimRewardRequest:** This request allows users to claim tournament rewards and returns updated progress data.
- **GetGroupRankRequest:** This request retrieves the player's rank for any tournament.
- **GetGroupLeaderboardRequest:** This request fetches the leaderboard data of a tournament group.
- **GetCountryLeaderboardRequest:** This request retrieves the leaderboard data of the countries for a tournament.

Notes

We will pay attention to the following:

- The design and code structure.
- Readability and comprehensibility of code (Clean code).
- Consideration of concurrency and performance issues at high load.
- Welcome to use any external data sources with MySQL if it helps to have better performance.
- Unit tests.

Submission

Perform all your tasks within the provided Docker project. Include the necessary MySQL table creation queries in the *mysql-db-dump.sql* file. If you are utilizing any data source along with MySQL, ensure to add its container image to Docker and include the necessary initial scripts or queries for it.

You can share your work with backend.engineering.study@dreamgames.com by creating a private GitHub repository. Use the *readme.md* file to briefly explain how you organized your implementation and the design choices you made to solve problems. Additionally, include a Postman collection for API endpoints in your repository.

Please feel free to contact us if you have any further questions.