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 users' progress. Our systems need to be fast and secure to enhance the user experience. We use Spring Boot with Java to develop our backend services.

Definitions

- **LiveOps Event:** A limited-time event where users can engage in various challenges: sometimes competing against each other, sometimes working individually to achieve personal goals, or teaming up to collaborate on a shared objective.
- **A/B Test:** To improve the user experience, we run A/B tests on our feature releases. In an A/B test, there are multiple groups. For instance, in our game, a simple A/B test could involve determining different event targets and rewards.

1. User Progress

1.1 Features

- User progress should be stored in a MySQL database.
- A new user starts with 2,000 coins at level 1.
- When a new user is created, they should be assigned to one of the A/B Test groups (Group A or B).
- After completing each level, the user advances to the next level and earns 100 coins.

1.2 Flow and Requests

- **CreateUserRequest:** This request creates a new user, returning a unique user ID, level, coins, and assigned A/B Test group.
- **UpdateLevelRequest:** This request is sent by the client after each level is completed. It updates the user's level, increases coin count and increases helium count (if the user has joined Pop The Balloon Event), and returns the updated progress data. (More details in the "Pop The Balloon Event" section)

2. Pop The Balloon Event

This is a time-limited cooperative event that allows users to collect helium by completing levels. Users can use the collected helium to inflate a shared balloon with a partner. The goal is to inflate the balloon until it pops, revealing the reward.

2.1 Features

- The event automatically starts at 08:00 and ends at 22:00 (UTC) each day. During this period, users can perform invite, accept, reject, or update balloon progress actions.
- Users must be at least level 50 and pay 2,500 coins to participate.
- A user can invite multiple users to form partnerships, but they can only be matched with one partner per event. If the user and their partner successfully pop the balloon before the event ends, they cannot participate again in that same event.
- Users automatically collect 10 units of helium from level wins once they join the event. However, they need a matched partner to consume the helium and inflate the balloon.
- Partners must consume a total of 1,000 (Group A) or 1,500 (Group B) helium units to pop a balloon. The target depends on the users' A/B Test group.
- After the balloon is popped, both users in the partnership can claim 1,000 (Group A) or 1,500 (Group B) coins as a reward. The reward amount depends on the A/B Test group, and both partners can claim the same amount regardless of their individual contributions.

2.2 Flow and Requests

- **GetSuggestionsAndInvitationsRequest:** Returns 10 randomly available players from the same A/B group.
- **GetInvitationsRequest:** Returns any pending invitations.
- **InvitePartnerRequest:** Sends an invitation to one of the suggested players to pair up for the event.
- **AcceptPlayerRequest:** Accepts an invitation, pairing the two players for the event so they can start inflating the balloon together until the event ends.
- **RejectPlayerRequest:** Rejects the invitation, and the rejected player cannot send any more invitations to this user during the current event.
- **UpdateBalloonProgressRequest:** This request allows the user to consume their helium to inflate the balloon. The amount of helium used directly determines how much the balloon inflates. Users cannot update balloon progress after the event ends.
- **GetBalloonsInfoRequest:** Returns all balloon progress, including partnership details, inflating scores, and unused helium counts.
- **ClaimRewardRequest:** Allows users to claim 1,000 (Group A) or 1,500 (Group B) coins if a balloon has been popped. Both partners should be able to claim the same reward.

3. Global Leaderboard

This game has millions of users worldwide. We want to give users the opportunity to compete globally based on their level.

3.1 Features

- Users should be able to see the top 100 players by their level.
- The leaderboard should reflect real-time data as much as possible.

3.2 Flow and Requests

- **GetLeaderboardRequest:** Returns the top 100 players' IDs and scores.

Notes

We will focus on:

- The design and structure of the code.
- The readability and comprehensibility of the code (clean code).
- Consideration of concurrency and performance issues under high load.
- Using external data sources along with MySQL is optional but can be considered if it helps improve performance.
- Writing unit tests.

Submission

Complete all tasks within the provided Docker project. Include the necessary MySQL table creation queries in the *mysql-db-dump.sql* file. If you are using any additional data sources besides MySQL, ensure to add its container image to Docker and include the necessary scripts or queries for it.

Create a private GitHub repository to share your work. Use the *readme.md* file to provide a brief explanation of how you organized your implementation and the design choices you made. Additionally, include a Postman collection for the API endpoints.

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