Skyhawk Security: BE Home Assignment

Dear Candidate,

Thank you for your time and effort in completing this exercise.

As part of our interview process, we would like you to design and implement a **scalable backend system** for logging NBA player statistics (e.g., points, rebounds, assists, etc.). You can use **Go (Golang) or Java** to implement this, without relying on ORM. The system should be easily deployable using **Docker Compose** or **Minikube**.

Please follow the guidelines and requirements below and document your approach, decisions, and design choices in the solution.

Pre-requisites:

1. **Language**: Code must be written entirely in **Go (Golang)** or **Java** (without using ORM framework)
2. **Deployment**: The project must be containerized and easily deployable using **Docker Compose** or **Minikube**.

Functional Requirements:

1. Logging NBA Player Statistics

You need to design an endpoint/system to log NBA player statistics. The system should handle multiple player stats per game, ensuring that they are validated correctly.

- Each player is associated with a basketball team.
- The statistics recorded per game are **Points, Rebounds, Assists, Steals, Blocks, Fouls** (integer, max value: 6), **Turnovers, Minutes Played** (float, between 0 and 48.0)
- **Input/Output**:
    - The system will consume this data from an external source (non-human system), meaning the input will be machine-readable.
    - **Live Data Fetching:** The system must support fetching up-to-date statistics even while a game is still ongoing. For instance, if someone requests LeBron

James's stats for the current season, the response should include both completed games and the current, in-progress game data up to that point.

2. Calculating Aggregate Statistics

Once data is logged, the system should calculate aggregate statistics and serve them via another API.

- Season Average per player.

- Season Average per team (average stats for all players in a team).

The system should provide these stats in a human-readable format The stats should always reflect the most up-to-date data once the player statistics are written.

Mandatory Considerations:

- **Scalability, High Availability and Fault Tolerant: The system should support high thruput and handle tens or hundreds of concurrent requests.**
- **Real-time Data Availability: Once data is written for a player's game, it should be immediately available for fetching aggregate stats.**
- **Solid Architecture.**
- **The system should be maintainable and support frequent updates and changes across all levels.**

Submission:

Please provide the following in your submission

- Code: A GitHub repository with your implementation.
- Document your solution: Explain your thought process, challenges encountered, and trade-offs made during implementation.
- **Architecture Diagram:** Provide a high-level system architecture diagram illustrating all key components (e.g., API, database, services). Describe the data flow for both writing (updating) and reading (retrieving) player statistics within the system.
- **Deployment Diagram:** Include a cloud deployment diagram (AWS, GCP, or Azure) showing how your system would be deployed in a production environment. Specify any required services (e.g., load balancer, database, container orchestration) and explain their role in ensuring scalability, availability, and performance.
-
    - Explain any decisions you made regarding the system's architecture, database choices, scalability, and maintainability.
    - Describe your design and implementation considerations. Why did you choose X over Y.
    - Provide examples/explanations on how to run the project with its entire functionality.

- Describe how you would implement and deploy this over AWS.

Once your solution is ready, share the GitHub repository link with recruiting24@Skyhawk.security.
Looking forward to your solution!

Good Luck!!
Skyhawk Engineering BE Team