Major Flaws and Suggested Improvements

The Texas Hold'em Poker Simulator suffers from a number of critical limitations at the moment that make it not production-ready for a multi-user environment.

In fact, the core application lacks such a basic functionality as a registration and authentication of its player. As a result, it's impossible for a user to create personalized identities or some kind of a profile. Therefore, all users remain anonymous; it is not possible to track their playing statistics, let alone provide them with consistent game-play across different playing sessions. Further, this game will be unable to ensure the protection of any personal information about users without the abilities mentioned above and cannot prevent unwanted persons from getting access to the game; that risks safety in use in a real-world scenario. This may lead to sensitive information exposure or alteration by attackers and tampering with gameplay or compromise user data.

This absence of persistence is also complemented by an absence of persistent game state storage. All game progress and historical data are lost if a session ends or the application is restarted. Hence, without persistence, one cannot establish coherent game history or track player progress over time, which would be required for a fully functional and engaging multiplayer experience. Another problem is that the application works on a single, global game session shared among all users, so this results in a messy environment where every player's action influences all others at once. The lack of isolation between sessions leads to many players actually playing on the same game instance, which can lead to confusion and a bad user experience. In a real-time deployment, users should have independent private game sessions.

This design flaw is compounded by the lack of synchronization mechanisms, leading to potential conflicts when multiple users attempt actions at the same time. If actions are done simultaneously, there may be race conditions where different players’ inputs clash, leading to inconsistent or wrongly determined game outcomes. For example, if two players try to place a bet simultaneously, one player’s action might override the other’s, creating an unpredictable and possibly unfair situation.

The absence of proper instance management and multi-session support severely impacts the application's scalability, as it struggles to handle increasing user loads due to its shared-state architecture. As more users join, this shared-state architecture faces load problems. The number of users determines the scope of possible performance bottlenecks, which might cause the game to hang, disconnect, or even crash the system. Additionally, the existing design does not support multiple poker games within a single session, as is usual in every multiplayer gaming application.

This application must be refined to become a fully prod-ready system. Some of the improvements include user authentication support, database persistence of states during games, support of independent game sessions, synchronization controls for concurrent operations, and scalability through balanced loading and instance management. Such changes would make multiplayer poker more reliable, scalable, and enjoyable. The requirement of supporting multiple independent game sessions cannot be neglected. All the game sessions must be independent in nature so that the isolation of individual players does not allow interference among users. This will provide a cleaner environment with proper organization for better management and experience for users. In addition, controls should be introduced to synchronize the results of concurrent actions to avoid race conditions and to ensure proper ordering of actions.