Major Flaws and Suggested Improvements

The Texas Hold'em Poker Simulator currently faces several critical limitations that prevent it from being production-ready for a multi-user environment.

At its core, the application lacks basic player registration and authentication capabilities. This means users cannot create unique identities or maintain personalized profiles. This anonymity makes it impossible to track individual player statistics or provide consistent gameplay experiences across sessions. Moreover, without authentication, it’s impossible to protect user data or prevent unauthorized access to the game, which poses a security risk in a real-world scenario. Without secure handling of GET and POST requests, sensitive data can be exposed or altered, allowing attackers to tamper with gameplay and compromise user data.

Adding to these concerns is the lack of any game state persistence. Once a session ends or the application restarts, all game progress and historical data vanish without a trace. Another problem the application has is that it operates on a single, global game session shared by all users, creating a chaotic environment where every player's actions affect everyone else simultaneously. This lack of session isolation results in many players playing what is virtually the same game instance, with confusion and poor user experience. This setup is also not feasible for real-time deployment, as users should have independent private games.

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.

To transform this application into a production-ready system, several improvements would be necessary. These include, but are not limited to, implementing robust user authentication, adding database persistence for game states, supporting multiple independent game sessions, introducing synchronization controls for concurrent actions, and enhancing scalability through proper load balancing and instance management. These modifications would create a more reliable, scalable, and enjoyable multiplayer poker experience. Supporting multiple independent game sessions is also crucial. All game sessions must be independent, where the isolation of individual players prevents interference among users. This would provide a cleaner environment with appropriate organization for better user management and experience. Besides that, synchronization controls for the results of concurrent actions should be implemented to avoid race conditions and guarantee appropriate ordering of actions.