Courtney McDonald

CS 230 Operating Systems

Module 8 Journal

**Client Summary and Software Requirements**

For this course, my client was The Gaming Room, which currently offers *Draw It or Lose It* as an Android-only game. They wanted to expand the game to multiple platforms—including desktops (Linux, Mac, Windows) and mobile devices—while keeping the gameplay fast and reliable. The game involves multiple teams and players, enforces strict timing rules, and renders high-resolution images quickly. The main goal was to create a system that is scalable, secure, and able to run multiple simultaneous game instances without errors or conflicts.

**What I Did Well**

I feel I did a good job of turning the client’s requirements into technical solutions. For example, in the domain model, I used the singleton pattern for the game service to make sure only one game instance is active at a time. This matched the client’s need to avoid conflicting game states. I also carefully considered **cross-platform compatibility**, thinking about how memory and storage management would differ between server-side Linux and mobile clients.

Another strength was in the Recommendations section, where I explained why I chose certain technologies, like Redis for real-time game state management and PostgreSQL for relational data. I also included security measures such as TLS/HTTPS encryption and role-based authentication, which were critical to protecting users and the system.

**Helpful Parts of the Process**

Working through the design document helped me see how planning before coding makes development smoother. Mapping out the system architecture and constraints, like unique identifiers and memory limits, allowed me to anticipate challenges, such as rendering multiple images quickly or handling multiple users at the same time. It also made me think realistically about platform differences. For example, why mobile devices aren’t suitable as servers for a multi-user game.

**Part I Would Revise**

If I could revise one part of my work, it would be the Evaluation table comparing operating platforms. While I explained the pros and cons, adding numbers or estimates, like memory requirements for 200 high-resolution images or expected server loads would make it more concrete. I also think adding visual comparisons like charts or graphs would help the client quickly understand trade-offs.

**User Needs and Implementation**

One of the most important parts of this project was thinking about what the users actually need. I considered timing, responsiveness, and gameplay fairness. For example, I added memory management strategies to ensure images render smoothly in the allotted time. Security was also considered, with encryption and access control measures. This reminded me how important it is to **align design decisions with real user needs**, not just technical capabilities.

**Approach to Designing Software**

I approached the design using object-oriented principles and modular architecture. I outlined entities and their relationships, then layered in memory, storage, and distributed system considerations. Thinking through **different deployment scenarios** for Linux, Mac, Windows, and mobile devices was especially helpful. In the future, I would include performance testing and user feedback cycles earlier in the process and make sure collaboration between development teams is planned from the start.

**Overall Reflection**

Completing this design document helped me connect technical knowledge, client requirements, and user experience in a meaningful way. I now better understand the planning and decisions that go into building scalable, secure software. It also taught me how to document my decisions clearly, which will help with maintaining and improving software in the future.