**The Gaming Room: Software Design Summary**

**Client Overview and Software Requirements**

The Gaming Room was a fictional client created for immersion in the project. They had an existing game developed for Android and aimed to expand its availability across multiple operating systems to reach a broader audience.

**Strengths in Documentation Development**

One of the strongest aspects of my documentation was the clear correlation between evaluation and recommendations. Not only did I present the pros and cons to help the client make an informed decision, but I also included insights on how the recommended approach could be utilized more effectively.

**Value of the Design Document in Development**

I’m a top-down thinker, so I thrive on timelines, roadmaps, and abstraction. These tools ensure a smooth workflow by providing structure and clarity. Understanding the flow of an application makes it easier to develop reusable components and minimizes the need for rewriting or fixing code later. Having a clear plan upfront eliminates guesswork and makes the coding process much more efficient.

**Potential Revisions and Improvements**

If I could revise one part of the documentation, it would be the evaluation section. Since the document was pre-designed as part of a curriculum, I didn't have control over its structure, but the evaluation felt underdeveloped. The sections on server-side considerations and development tools lacked strong options. It was unclear whether we were expected to choose native monolithic applications, yet the exercise seemed to suggest a single-server, multi-browser architecture. This left me questioning the intended learning objectives.

**Understanding and Implementing User Needs**

You can’t solve a problem without first understanding it—that’s the foundation of problem-solving. I carefully analyzed the client’s requirements and designed a solution that would allow for the most efficient execution of their vision. Aligning software design with user needs is crucial because a well-designed system is only as good as its ability to meet its intended purpose.

**Approach to Software Design and Future Strategies**

As a top-down thinker, I naturally follow the Software Development Life Cycle (SDLC). I appreciate both the creative and logical aspects of software development, which means I take a structured approach:

1. **Requirements Analysis** – Understanding the problem and defining the project scope.
2. **Design** – Creating an effective and efficient blueprint.
3. **Implementation** – Bringing the plan to life through development.
4. **Testing & Refinement** – Identifying and fixing weak spots.
5. **Deployment** – Releasing the software for real-world use.

In future projects, I’ll continue using this method, as it ensures a balance between structure and flexibility, allowing for creative problem-solving while maintaining efficiency.