The Gaming Room Software Design Document Reflection

Briefly summarize The Gaming Room client and their software requirements.

The Gaming Room is a client that wanted to take their game *Draw It or Lose It*, originally for Android, and turn it into a web-based app that works across multiple platforms. They were looking for a design that was scalable, easy to maintain, and flexible enough to support future growth.

What did you do particularly well in developing this documentation?

I think I did a good job explaining technical decisions in a clear way. I focused on writing recommendations that not only made sense for development team but also addressed what the client needed. I also made sure to cover important things like scalability, storage, and security to create a complete plan.

What about the process of working through a design document did you find helpful when developing the code?

The design document helped me plan before coding. It made me think of how everything connects, how data is managed, and where potential issues could come up. That planning made it easier to visualize the code and how to organize it logically. It helped create a clear picture of what the code would need to look like.

If you could choose one part of your work on these documents to revise, what would you pick? How would you improve it?

If I had the chance, I’d probably add more detail about the distributed systems part. I think I could’ve explained the network setup and specific tools (like load balancers or monitoring solutions) in more depth. I also think I could’ve learned more to help me explain better. That would have made the plan stronger for handling outages or spikes in traffic.

How did you interpret the user’s needs and implement them into your software design? Why is it so important to consider the user’s needs when designing?

I kept the user’s needs in mind by focusing on making the app scalable, reliable, and cross-platform. That’s why I recommend cloud hosting, flexible storage, and a secure setup. Considering user needs is key because the software is for them—it must solve their problems or it won’t be useful, no matter how technically solid it is.

How did you approach designing software? What techniques or strategies would you use in the future to analyze and design a similar software application?

I approached this by breaking the problem down into layers and looking at what tools or architecture would be best to support each later. I also made sure to think about scalability and maintenance from the start, not as an afterthought. In the future, I would probably use a similar layered approach, and also add more visuals like sequence diagrams to help communicate the design more clearly.