* Client-Server Pattern: Discuss how the client-server pattern can be used to satisfy software requirements and efficiently solve a problem. Specifically, the web-based game application must be able to be run on multiple operating platforms.

The client-server model is efficient in solving problems that utilize a distributed model. Multiple clients can connect to the same servers in order to communicate among servers and other clients. A uniform approach is used for client-server interaction. All information required by the server must always be included in communication thanks to the concepts of the uniform interface and statelessness. This information’s resources must be self-descriptive so that media types can be handled properly. The concept of separation of concerns allows simpler client-server interactions by allowing the clients and servers to work independently when not sending data back and forth. The isolation of the two layers in the client-server model allow clients and servers to have this independence. So long as the information sent is formatted as the server expects, the intricacies of the client side program and its state need not influence the server side. In this model, caching is a common technique used to increase efficiency by storing information that is likely to be used later, decreasing the time it takes to recall said information. Information sent should inform the server whether or not sent information is cacheable.

* Server Side: You have developed the application from the server side. Discuss how the server side provides communication to the client side with REST API style.

The REST API style enforces a stateless and uniform model to the information that the server is to receive. This means that whenever information is given to the server, it must include all information required to allow the server to do what is expected, and given the stateless approach, the server may interact with many different forms of clients. This means that the server side programs do not need to be redesigned for a variety of clients, greatly reducing development time and making it much easier to add new kinds of clients, though this is often not the case with client-side programs.

* Client Side: You wrote an application for multiple clients where the multiple environments can interact with the server. Discuss what is required of the developers so that the application on all three clients is able to be used on the website. Consider what next steps would entail to develop for the client side of the game application. For instance:
  + How would you add more users to the database?

Clients can send a request for to the server to add a new user. This process can then be taken care of in the back-end, as, once the new user’s information has been provided, the server can handle the request and take care of the proper database work with a tool such as SQL.

* + What other features might you include in the game app?

Utilizing this client server model, an interesting feature to add would be cross-play

between different types of clients. This would work by relaying information to clients based on information other clients provide to the server. This would allow any variety of clients to communicate among each other, allowing the game to be played with users utilizing an array of different client types.

* + What if The Gaming Room asked you to host the application on a fourth and fifth client? For example, on Xbox and PS4.

A new client program would be developed for the new host. Conforming to the REST API style would allow easy communication to the already developed server, however, the new client program will have to conform to the client it is being built for. This may require enlisting specific developers who have experience in the new clients to be added. This means that a variety of front-end developers may be required for the project, but this should have little effect on the back-end given the separation of concerns as outlined in the client-server model.