David Vega

CS-230 Operating Platforms

Week 4 – Journal Entry

September 26, 2022

**Client-Server Pattern**

The client-server pattern can be used to satisfy software requirements for the gaming application because the hardware and software components are independent of each other. The ability to scale clients should not influence the functionality of the game or the data it retrieves. Most devices provide some sort of web browser for users to interact with web pages. These front-end applications take in user input and make requests to external application servers that then pass the request onto another external server that houses the data. The database resource then sends the data back through the application server and back to the client as a response. The separation and centralized architecture are ideal when multiple clients are involved and that run on different operating systems. The only configuration to the front-end would be at the client level to ensure the application is compatible with multiple browsers.

**Server Side**

The server-side communicates with the client side by issuing responses in a standardized text-based format. This format is often in the form of JSON or XML which is a lightweight format easy to be displayed by web applications. These text-based formats are the result of http requests configured on the server. There are various tools that a developer has at his/her disposal to implement this type of architecture. One of the tools being used for the game application is Dropwizard. Dropwizard is a Java framework that is used to build RESTful backends. The main purpose of this framework is to provide the Java project with a variety of libraries in one embedded Java development project. This tool can also be used to implement basic authentication.

In security, the principal object is used to represent the user for which the credentials are provided in the header of the HTTP request. The Authenticator class is used to validate the user credentials (i.e. – username and password) while the Authorizer class validates the roles (e.g.- “Admin,” etc.) and determines if the user is permitted to perform certain functions. The @Auth annotation is used as a parameter to trigger the authentication filter.

**Client Side**

The next steps in developing the game app are to consider scalability. Adding more clients should be simple enough if the right architecture were put in place. I think one of the features I might include in the game app is input validation. Since security is a critical component of this application and the application is authenticating clients, it would be a good idea to implement a set of restrictions at the start of user account creation. I would implement a set of rules for both username as well as password to ensure that user credentials are unique and secure.

If the Gaming Room asked to host the application on an Xbox or PS4, I think this would be straight forward to implement if both systems can send and receive Http requests and responses.