Journal Submission: Module Four Assignment

In this journal entry, I would like to focus on some key ideas from module four: how we could use the client-server pattern in satisfying software needs and solve problems for a web-based game that had to work on different devices. I will also discuss how we develop both sides of the application: the server side and the client side.

Client-Server Pattern

This is an approach where applications consist of two parts: a client and a server. Such an approach suits perfectly when there is a need for developing web applications that are cross-compatible with many operating systems, such as Windows, Mac, iOS, and Android. Thus, the client-server pattern breaks the application into two fundamental components: the server and the client.

Server: This part does most of the significant work, like storing data and processing requests.

Client: This part does what the player sees and with which they interact, like the game interface. Thus, keeping these two parts separate allows updating or changes in one part without affecting the other part of the app.

Server Side

With the server side, we utilize an API known as a REST API, which is the abbreviation for Representational State Transfer Application Programming Interface. A REST API allows the server to interact with the client using standard methods of the web: GET, POST, PUT, and DELETE. It lets the server manage the requests from both sides of different clients smoothly and securely. An online game of ours would see the server responsible for user login, in-game progress, and leader boards REST API endpoints. The server manages requests from various clients and guarantees that the interfaces are uniform and secure, irrespective of whatever kind of device the users are currently using.

Client Side

Implementing a client in essence means that part of the application with which final users directly interact by the help of an application that runs on different devices. Developers must make their client application compatible with other platforms, such as web browsers, mobile phones, and sometimes even gaming consoles like Xbox and PS4. Tools that come in handy for developers are cross-platform development tools like React Native or Unity in the development of such applications. These tools help ensure a similar look and feel across various devices.

Client Development Next Steps

Add more number of users into the database: To increase the count of users we need to implement a scalable system for management of user accounts, In which we need to implement API endpoints for signup, login and management of profiles of users. This would help us handle many more users with efficiency through some potent database system like Postgres or MongoDB.

Add features: We continue to add more features to the game, such as chat, friend lists, in-game purchases, among others. Each new feature is going to come with its API endpoints and client interface changes.

Support for more clients (Xbox and PS4): To support the game on the Xbox and PS4, the developers need to follow the guidelines set by these respective platforms. This includes the usage of their development kits, optimization of performance for consoles, and ensuring that the game maintains all the requirements of the console, including game controllers and integration with user accounts.

Conclusion

The client-server is a fantastic pattern to implement a web-based game compatible with different devices. With the help of REST APIs, it is possible to make effective interaction between the server and its clients to allow independence of updates and scalability. On the client side, there are many aspects of compatibility between platforms that developers must take into consideration, using proper tools and practices. As the game scales with more users, features, and device support, it is important that they are well orchestrated and executed to ensure everything runs well.