**4-4 Journal: Software Application Requirements**

Cristiano Miranda

Southern New Hampshire University

CS-230: Operating Platforms

Professor Sarkar

28 September 2025

**4-4 Journal: Software Application Requirements**

            The Gaming Room's new web application, Draw It or Lose It, utilizes the client-server pattern to allow access to the web application for multiple concurrent users. Rather than requiring participating users to connect directly to one another, which could result in security concerns, all users instead connect to a singular computer system. This computer system, known as the server, handles all incoming requests by providing the necessary data to match the requests made by users. For instance, if a user sends a signal to the server requesting a new clue to display on their screen, the server receives and processes the request by responding with a new clue or image, which is then displayed on the user's screen. Through this, the most critical aspects of the web-application, such as the current data for the game and the data generated by the users, are handled by the server allowing for a singular machine to solely manage the web-application's primary data, which avoids the potential for game data to become out-of-sync and allows for easier security since the only machine receiving data from other users and sending data to users is the server. Additionally, this pattern enables separation within the web application, allowing basic processes such as handling user interaction with the web application and displaying data returned by the server to be processed by the individual user's machine. Rather than bombarding the server with all processing for the web application, which would significantly limit the number of users that could use the web application at once, the individual clients help alleviate the amount of work done by the server by handling these basic tasks, which allows the server to focus on delivering data to multiple users. Additionally, this also allows the web application to be compatible with numerous operating systems, as the client's system is only responsible for handling basic web interactions and sending data, which are processes that virtually all web browsers can handle. By designing the web application to be compatible with almost any web browser and by splitting the web application's logic between the client and the server, the web application can be easily scaled to accommodate more users, as only the server itself needs to be upgraded, not the individual users' machines. In the end, the client-server pattern enables The Gaming Room to host a web application that can easily scale to accommodate more users playing concurrently. By utilizing the server to connect multiple clients and leveraging its system to handle basic processes, the web application is virtually compatible with any operating system and web browser, thereby increasing the potential number of users who can play their new game.

            To allow the server to focus mainly on handling more intensive processing, communicating with specific applications like the database, and for connecting all users, each user's systems handle all basic user interaction, but in instances where the user's interaction needs to update game information, such as a user making a guess or switching clues, the information needs to be sent to the server or the server needs to send information to the user. To accomplish this, The Gaming Room's web application utilizes a REST API style to allow users to access specific information or processes from the server, such as retrieving user information or requesting assignment to a specific team. Using this style allows developers to control the information that users attempt to access, and it enables the setting of strict controls on which particular users are allowed to access specific information. Thus, even in instances where unknown users attempt to access server information via a specific link associated with the REST API, the server will deny their request and prevent the communication of potentially sensitive information. Other benefits of this style include its scalability, as multiple URIs can be created and defined on the server to provide access to other processes and information. Plus, it can also allow predefining what data needs to be sent with the URI by including annotations during development to define what data needs to be included with the request, or by defining what user role should be allowed to access the URI to avoid running into issues in instances where processes need specific data to run or certain permissions are required. Overall, the use of REST APIs enables users to communicate with the server through different URIs, allowing them to access specific processes and information. The most important aspect of this style is that it allows developers to provide annotations for the URI that define what users have permission to make requests and what data type needs to be included with the request.

            The primary goal of developing this web application was to ensure compatibility with most operating systems, thereby allowing more users to access it. To accomplish this, all operating systems must be able to communicate with the server via REST APIs. Again, these REST APIs handle the various communication types between the server and the user, such as requesting information from the server or sending the server information generated by the user, so the focus for all three major operating systems, Windows, Linux, and macOS, is to develop their applications to send and receive information using these REST APIs. Regardless of the methods used to accomplish this, as long as the application can send and receive requests from the server, it will be able to interact with the web application and allow users of different operating systems to participate in games with one another. For instance, if multiple users sign-up for the first time to use the web-application, they would send a request to the server that includes their user information, and the server would in turn process this request by first verifying that all the necessary information is included with the request and that their informing was securely sent, before finally processing the data and storing the new user information in the database. Thus, users never directly interact with the various tools and processes used to enable the web application to function; instead, the server handles their requests, making it only necessary for the user to be able to communicate with the server. Still, to help ease development, it would be ideal to create a front-end for the web application, allowing all users to access it through a web browser. Although each operating system has distinct characteristics, one thing they all share is that they include a web browser, and these web browsers handle displaying and interacting with web applications in a nearly identical manner. Thus, by focusing on developing a front-end for the web application that can allow users to send and receive server requests via the URIs the server defines, users of different operating systems can utilize the same front-end web application to access and play the game. Additionally, this also helps when adding new features to the game, as it ensures that any new feature added during development will be functional on all operating systems, since they all use the same application to access the web application. For example, if developers added a sound to play while the timer ticks down, rather than having to spend time determining how to play audio through each different operating system's application, developers would have to discover and implement audio playback through their front-end of the web application, which saves time and allows all features to be shared regardless of operating system. However, there are situations in which developers need to consider adapting the front-end of the web application for specific devices. In the event that a user accesses the front-end of the web application via their mobile device or game console, specific changes should occur, as the user input is drastically different from that of a regular computer system. Controlling the cursor with a game controller or interacting with buttons on a smaller screen using touch inputs can lead to a poor user experience if not accounted for. Thus, using specific tools to determine whether a user is on a mobile device or a game console would enable the front-end to be displayed or interacted with in a way that accounts for these differences, ultimately ensuring that each user experience is as positive as possible.