8-2 Journal: Portfolio Submission

Richard Backscheider

CS-230-T5459 Operating Platforms 23EW5

6/25/2023

Southern New Hampshire University

1. Prompt: Specifically, you must address the following rubric criteria in your journal submission:
2. **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 architecture is an older and well-established pattern (Broadhead, 2022). A simple version of a layered pattern has both a front-end client and a back-end client (Broadhead, 2022). The client-server approach focuses on resources, embracing the idea of a central server which clients connect to for sharing data (Broadhead, 2022). To scale the application, the client-server architecture pattern uses a central point (the server) (Broadhead, 2022). Client-server architecture patterns have two main components, client and server (Butani, 2023). Both the client and server are located within the same system, communicating over a network but on separate hardware (Butani, 2023). The client, regarded as the server requester, initiates certain interactions with the server generating the services needed (Butani, 2023). Both client and server “are linked by request and reply connectors” (Butani, 2023). A major advantage of the client-server architecture pattern is its central computing of data (i.e. all files are stored in a central location for the network and data as well as the network peripherals are centrally controlled) (Butani, 2023). A main principle behind the client-server constraints is the separation of concerns (Visual Paradigm, n.d). “Separating the user interface concerns from the data storage concerns improves the portability of the user interfaces across multiple platforms; improving the scalability by streamlining the server components (Visual Paradigm, n.d). Within the web-based game application, the client makes the request to the server and the server parses the request. A benefit of this is that each team will be able to create a unique profile to that team. The unique identifier will enhance the company’s reputation amongst their patrons and ensure a secure environment for players, instead of using actual personal information.

1. **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**.

REST stands for Representational State Transfer (Visual Paradigm, n.d.). REST defines six architectural constraints (i.e. client-server architecture, statelessness, cacheability, layered system, code on demand, uniform interface) (Visual Paradigm, n.d.). REST is “an architectural style for building web services that interact via an HTTP protocol” (Editor, 2021). REST API’s principles were created in 2000 by computer scientist Roy Fielding, gaining “popularity as a scalable and flexible alternative to older methods of machine-to-machine communication” (Editor, 2021). The key elements of the REST API paradigm include client software that runs on a users’ computer or smartphone and initiates communication; a server offering an API as a way of accessing its data or features, and any piece of resource that the server provides to the client (Editor, 2021). “To get access to a resource, the client sends an HTTP request, in return the server creates an HTTP response with encoded data on the resource” (Editor, 2021). Both the HTTP request and HTTP response messages are “self-descriptive” containing information on how to interpret and process them (Editor, 2021). REST requests include four major parts, the HTTP method, an endpoint, headers, and body (Editor, 2021). The HTTP method describes what needs to be done with the resource. REST applications use HTTP requests to POST (create a resource), GET (make queries or retrieve a resource), PUT (create or update a resource), and DELETE (delete data) (Farcic, 2014). The endpoint contains a Uniform Resource Identifier (URI) that indicates where and how to locate the resource on the internet (Editor, 2021). Headers are used to store information pertinent to the client and server (Editor, 2021). Headers are used to provide authentication data, such as an API key, name or IP address of the computer where the server is installed, or information regarding the response format (Editor, 2021). The body conveys additional information to the server, such as a piece of data that needs to be added or replaced (Editor, 2021). REST initiates best practices or constraints that describe how requests and responses are processed by the server (Editor, 2021). By operating within the given restraints, “the system gains desirable properties”, which include client-server autonomy, uniformed interface, layered architecture, caching, stateless interactions, and Code on Demand (CoD) (Editor, 2021). JavaScript Object Notation (JSON) “is a lightweight data-interchange format” based on a 1999 subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition (Farcic, 2014 & JSON, n.d.). JSON is easy for machines to parse and generate, as well as for humans to read and write (Farcic, 2014). “JSON is a text format that is completely language independent but uses conventions familiar to programmers of the C-family” (i.e. C, C++, C#, Java, JavaScript, Perl, Python, and others), these properties make JSON a perfect “data-interchange language” (Farcic, 2014). Built on two structures, JSON is a collection of name/value pairs and ordered list of values (Farcic, 2014). Name/value pairs are thought of as an object, record, struct, dictionary, hash table, keyed list, or associative array (Farcic, 2014). “An ordered list of values is realized as an array, vector, list, or sequence” (Farcic, 2014). An object is an unordered set of name/value pairs, beginning with a left brace and ending with a right brace, each name is followed by a colon, and the name/value pairs are separated by a comma (Farcic, 2014). An array is an ordered collection of values that begins with a left bracket and ends with a right bracket, the values are separated with a comma and can include a string in double quotes, a number, true, false or null, or an object or array—the structures can be nested (Farcic, 2014). Within the web-based game application, the client makes a request to the server and the server in return creates an HTTP response with encoded data providing communication to the client side using REST API.

1. **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:
   1. How would you add more users to the database?
   2. What other features might you include in the game app?
   3. What if The Gaming Room asked you to host the application on a fourth and fifth client? For example, on Xbox and PS4.

The creation of a new user is initiated on the client-side. User name, passwords, and or multi-factor authentication would be established on a client-side, but stored permanently on the server-side. In this way user login and access is always available from any client that has web access and proper credentials. To assign more users to the database the client could consider using a server with a build-in user-interface, for the purpose of making it useable for all employees and not just for those who already know how to code. I would include a multi-factor authentication (MFA) as a means for logging in to the server. The MFA is an essential security requirement, enabling a secure pathway to login. The server would need to keep a record of which users are logging into the system, when and where (i.e. name, location, time date, and IP address). If the Gaming Room wanted to host the application on a fourth and fifth client, the client does not necessarily have to change the server operating system that they’re running, but instead adapt new hardware or upgrade the hosting capability to adjust to the operating system they have currently running.

1. Prompt: For this course, **submit the completed software design document** that you created for your client, The Gaming Room, into the GitHub repository for your portfolio. This completed design document demonstrates your ability to be effective in working through the design of a software application and to communicate it effectively to a client and your development team members.

Once you have submitted your software design document, **update the README file** in your repository and include your answers to each of the questions below. You could include the questions and write a few sentences in response to each one, or you could write a paragraph or two weaving together all of your answers.

1. Briefly summarize The Gaming Room client and their software requirements. Who was the client? What type of software did they want you to design?

Creative Technology Solutions (CTS), in which I work as a Technology Consultant, has recently taken on a new client, The Gaming Room (SNHU, n.d.). The client would like CTS to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It; currently only available in an Android app (SNHU, n.d.). Draw It or Lose It, mimics the 1980’s television game show *Win, Lose or Draw*, with teams taking turns to draw images on an easel to help their teammates guess the puzzle (a phrase, title, or thing) (SNHU, n.d.). The application for Draw It or Lose It will provide images from a large library of stock drawings as clues (SNHU, n.d.). Each game will consist of four rounds of play, lasting one minute each (SNHU, n.d.). Drawings will be provided at a steady pace and will become fully complete at the 30-second mark (SNHU, n.d.). If a team is unable to guess the puzzle before the time expires, the remaining teams will be given a chance to solve the puzzle with a 15-second time limit (SNHU, n.d.). The client hopes to facilitate and expand the development of the web-based version of the gamming app., using various software patterns in a distributed environment, has asked CTS to streamline the development (SNHU, n.d.). As the Technology Consultant I will begin developing the game application, working to address The Game Room’s software requirements (SNHU, n.d.). The hardware requirements will be addressed once the technical manager and client have reviewed the progress made as a result of the software application decisions (SNHU, n.d.). The Gaming Room has requested that their game “Draw It or Lose It” will be capable of having one or more teams involved at a time, and that each team would consist of multiple players (SNHU, n.d.). “Draw It or Lose It” will also require a function in place to track whether a unique name is in-use or not (SNHU, n.d.). Finally, The Gaming Room has also requested that only one instance of the game can exist in memory at a time (SNHU, n.d.).

My next task as CTS’s Technology Consultant is to evaluate the three traditional operation platforms (Linux, Mac, and Windows), as well as mobile platforms, for how the game application software can be deployed and run, and what would be required to do so (SNHU, n.d.). As the Technology Consultant, I will conduct an evaluation of the characteristics, advantages, and weaknesses of the suggested operation platforms (Linux, Mac, and Windows), as well as the mobile platforms (SNHU, n.d.). The client has requested the server-style configuration for hosting the website and allowing it to scale up to thousands of players (SNHU, n.d.). The client would like to move beyond just their Android-only app and offer supporting players on iOS and Android mobile platforms, as well as traditional desktop-based operating systems (OS) (SNHU, n.d.). A major requirement for the application is that it is delivered as a modern, responsive HTML interface running inside the web browser for desktop clients (Linux, Mac, and Windows) and mobile platforms (SNHU, n.d.). Each platform will have the capability of communicating with the back-end web application running on the server (SNHU, n.d.).

The final task will be to analyze the characteristics and techniques specific to various system architectures, and make recommendations to The Gaming Room, addressing the following: Operating Platform, Operating Systems Architectures, Storage Memory, Distributed Systems and Networks, and Security (SNHU, n.d.). Once completed, I will present my evaluation to the client, informing them of my findings and aiding in their decision on the benefits of each platform and their utility in a distributed environment (SNHU, n.d.).

1. What did you do particularly well in developing this documentation?

The thing that I feel I did particularly well in developing the documentation was covering the cloud-based server portion of the project. I think that I covered the evaluation section the best. In the evaluation I included information highlighting both sides of the server systems, along with their software. I made the client aware of both the benefits and negatives of each.

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

I personally can’t say the design document was too helpful. The code took multiple attempts on various computers in my house running different versions of Eclipse to get working properly. I consulted with the class discussions but didn’t receive any replies and had to contact my instructor for further information. If I had to choose one helpful part of the design document, I guess it would have to be the step-by-step guide. Although the guide ended up leading to errors depending on the version of Eclipse, the guides provided thorough information and could be potentially helpful if matched with a concurrent Eclipse application.

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

One part of my work on these documents that I would like to revise would be the details of different operating systems. I would have liked to provide a much deeper analysis of each server operating system. I believe I may have contributed too much information towards Ubuntu server, and not enough detail towards Mac OS, IOS/Android, and Windows server. In retrospective, I would like to extend both cloud-based/physical server software explanations.

1. 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?

The user’s needs were implemented depending on the budget and scalability requirements. For example, if the client was planning a slow growth approach, I would recommend a physical server system. If the client was planning a fast growth approach, I would recommend a cloud-based server system. I also considered the requirements of storage, memory, and overall computational performance. If the user needed high-performance, their best bet would be to rent a cloud-based system, whilst, if the user needed a secure storage server, I would have recommended a physical server with large storage volumes. The user’s needs are important due to the need to be able to make clear judgment for prospective growth. If the user didn’t see them growing rapidly, the need for the highest specified server may not be very necessary.

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

The approach I took to designing the software was to read each given scenario, so that I could better understand the client’s wants and needs. As the Technology Consultant I needed to be aware of my given responsibilities and requirements. I then used these responsibilities and requirements to make an informed decision of the best route--cloud-based and/or physical server/software.

Works Cited

Broadhead, R. (2022, January 2). *Client-server pattern - software architecture*. Building Better Developers Develpreneur. https://develpreneur.com/client-server-pattern-software-architecture/

Butani, A. (2023, March 9). *5 essential patterns of software architecture*. Enable Architect. https://www.redhat.com/architect/5-essential-patterns-software-architecture#:~:text=Client%2Dserver%20pattern,a%20network%20on%20separate%20hardware.

Editor. (2021, March 4). *REST API: Key Concepts, best practices, and Benefits*. AltexSoft. https://www.altexsoft.com/blog/rest-api-design/

Farcic, V. (2014, August 12). *REST API with JSON*. Technology Conversations. https://technologyconversations.com/2014/08/12/rest-api-with-json/

JSON. (n.d.). *Introducing json*. [ECMA-404 The JSON Data Interchange Standard](http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf). https://www.json.org/json-en.html

Southern New Hampshire University (SNHU). (n.d.). *Project One Guidelines and Rubric Module Three*. Project One Guidelines and Rubric. CS-230-T5459 Operating Platforms 23EW5. https://learn.snhu.edu/d2l/le/content/1309582/viewContent/24267069/View

Southern New Hampshire University (SNHU). (n.d.). *Project Two Guidelines and Rubric Module Five*. Project Two Guidelines and Rubric. CS-230-T5459 Operating Platforms 23EW5. https://learn.snhu.edu/d2l/le/content/1309582/viewContent/24267070/View

Southern New Hampshire University (SNHU). (n.d.). *Project Three Guidelines and Rubric Module Seven*. Project Three Guidelines and Rubric The Gaming Room. CS-230-T5459 Operating Platforms 23EW5. https://learn.snhu.edu/d2l/le/content/1309582/viewContent/24267049/View

Visual Paradigm. (n.d.). *What is REST API?*. What is rest api? https://www.visual-paradigm.com/guide/development/what-is-rest-api/