

# **CS 230 Project One - Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.3rdcrjn)

[**Domain Model 3**](#_heading=h.2s8eyo1)

**Evaluation 4**

**Recommendations 8**

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/16/23 | Josh Hall | First submission |
| 2.0 | 07/30/23 | Josh Hall | Added project 2 tables covering evaluation of multiple platforms. |
| 3.0 | 8/14/23 | Josh Hall | Added Recommendations section |

## [Executive Summary](#_heading=h.35nkun2)

The Gaming Room is looking to implement an online version of their gaming app. Due to the multiplayer and distributed gameplay, they are looking for assistance in developing their new platform to ensure players, teams, and games remain stable during gameplay. We will be able to assist with the development of this type of platform, and will be able to utilize object oriented strategies using Java to illustrate the platform's utility.

## Requirements

- One or more teams can play a game.

- Each team will have multiple players.

- Game and team names must each be unique.

- Only one instance of the game can exist in memory.

## [Design Constraints](#_heading=h.1ksv4uv)

Due to the game taking place across space on the internet, keeping the data in sync will be the main constraint. Additionally, there is a constraint in keeping the naming unique for players, teams, and games for games across the internet.

## [Domain Model](#_heading=h.2jxsxqh)

The Java class structure will rely on keeping the game instance a singleton through the GameService class, this will address the design constraint of keeping game, team, and player data in sync across browsers. An additional strategy to keep the data in line will be to iterate over all class instances to ensure only unique data can be added to the game instance. Finally, the team, game, and player classes will also inherit their fields and field accessor methods from a parent Entity class in order to keep the code more manageable.

"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.

## [Evaluation](#_heading=h.26in1rg)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | **Advantages**: macOS is generally considered a secure operating system, which can be beneficial for hosting web applications. And if the web application needs to integrate with other Apple products or services, hosting it on macOS can simplify the process. Additionally, Xcode, the integrated development environment for macOS and iOS, provides tools for iOS app development and could be beneficial if we wanted to target iOS specifically at a later date.  **Disadvantages**: Hosting a web application on a Mac platform limits you to Apple's hardware. Because of this, Mac will not often be the most cost effective or scalable solution. | **Advantages**: Linux is known for its stability and reliability, making it an excellent choice for hosting web applications. It can run for extended periods without requiring reboots. Additionally, Linux is considered more secure than other operating systems due to its strong access controls, permissions system, and a small number of known vulnerabilities.  **Disadvantages**: For or organizations unfamiliar with Linux, there can be a steep learning curve. The command-line interface and system administration may pose challenges. Although Linux supports a wide range of hardware, some specialized hardware might have limited driver support, which could lead to issues. | **Advantages**: Windows seamlessly integrates with other Microsoft technologies like .NET, and SQL Server. Windows is known for its strong support of ASP.NET, a powerful web development framework. Additionally, Microsoft's cloud platform, Azure, fully supports Windows-based hosting, which is highly scalable and reliable.  **Disadvantages**: Licensing costs for Windows Server can be higher compared to open-source alternatives. Windows Server generally requires more resources Linux-based solutions, which could impact hosting costs. And importantly, Windows has historically been a more frequent target of malware and cyber-attacks due to its popularity. | **Android advantages**: Android offers more flexibility for developers, and better access to device hardware, which is advantageous for certain types of web applications that require access to features like NFC, Bluetooth, or the file system.  **Android disadvantages**: The Android ecosystem is highly fragmented due to various devices running different versions of the OS and customized user interfaces.  **IOS advantages**:The strict App Store guidelines provide a secure and controlled environment. Since Apple controls both hardware and software aspects of iOS devices, developers can more easily create a consistent user experience. **IOS disadvantages**: iOS offers limited customization options. Users cannot extensively alter default apps or system behavior. behaviors.ely. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| Client side | Tailoring the user experience for different client types, such as desktops, laptops, and tablets, can be demanding in terms of design and development. Different Mac models and OS versions might have varying capabilities and requirements. This may require additional testing efforts to ensure smooth functioning across various setups. If you plan to distribute your software through the Apple App Store, there are specific guidelines and requirements you must adhere to. Understanding and complying with these guidelines can influence the development process and timeline. | Supporting multiple types of clients on Linux may require additional costs for testing and ensuring compatibility with different distributions and versions of the operating system. Developers must have a strong understanding of the Linux ecosystem, including different distributions, package management systems, and library dependencies. Expertise is needed to address challenges related to compatibility, such as handling different libraries, system calls, or kernel features between client types. | Ensure that your software works seamlessly across various Windows versions and any future releases. This may require additional development and testing efforts. Take into account the hardware capabilities of different devices. Ensure that your software doesn't demand excessive resources or incompatible features that might limit its use on certain client types. Address potential security vulnerabilities specific to different Windows versions and devices. Keeping up with security updates and best practices may require continuous efforts and expertise. | Each mobile platform has its app store and submission process, which must be followed to make the app available to users. This process can be time-consuming, especially if you need to manage different release cycles for each platform. Consider using cross-platform development tools and frameworks. These tools allow developers to write code once and deploy it to multiple platforms, potentially saving time and effort. However, it's essential to carefully assess the suitability of such tools for your specific project's requirements, as they may have limitations or performance trade-offs. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| Development tools | **Web App**: We should utilize JavaScript, HTML, and CSS for the web version of the app, since this will streamline the development across all platforms. Additionally, we could use React as a JavaScript front-end framework for the user interface, and Express.js for the backend. **Tools**: Git for version control, and NPM for Node and React packages. MySQL for a persistent database. Visual Studio Code IDE is a lightweight, cross-platform code editor that supports a wide range of languages and is highly extensible.  **Mac specific**: We could utilize Apple’s Xcode platform if we wanted to target an iOS specific release in the future. | **Web App**: We should utilize JavaScript, HTML, and CSS for the web version of the app, since this will streamline the development across all platforms. Additionally, we could use React as a JavaScript front-end framework for the user interface, and Express.js for the backend. **Tools**: Git for version control, and NPM for Node and React packages. MySQL for a persistent database. Visual Studio Code IDE is a lightweight, cross-platform code editor that supports a wide range of languages and is highly extensible.  **Linux specific**: The command-line interface used to interact with the Linux system and execute various commands. | **Web App**: In addition to the previous Web app strategy using JS, HTML, and CSS, Windows offers the C# language and ASP.NET framework, which work well together to produce web apps. A Javascript and HTML based front-end may still be necessary for the online interface. **Microsoft tools**: Visual Studio, which is Microsoft's flagship IDE, offering comprehensive support for C# and ASP.NET. Microsoft SQL Server: A relational database management system developed by Microsoft. **Tools**: Git for version control, and NPM for Node and React packages, and NuGet for ASP.NET packages. | **Web App**: In addition to the previous Web app strategy using JS, HTML, and CSS, a cross-platform framework should be considered for the application. **Cross-platform code:** React Native is a JavaScript framework that allows you to build mobile apps from existing React code base.  Flutter: A UI toolkit from Google that enables the development of natively compiled applications for mobile, web, and desktop from a single codebase.  Xamarin: A Microsoft-owned framework that allows you to build cross-platform mobile apps using C# and .NET. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: I recommend a Linux operating system running an XFS filesystem, as this will give the server innate security and very fast read times for images. Additionally, the server logic should be served over Node.js for communication with the client, as the ecosystem for Javascript is vast and will allow for maximum momentum and scalability as the system expands.

1. **Operating Systems Architectures**: The Linux kernel is responsible for managing hardware resources, including memory, CPU, and input/output operations. It provides essential functions like process scheduling, memory management, and managing device drivers. The Linux file system is responsible for managing how data is stored, organized, and accessed on storage devices like hard drives and SSDs. I'll go into more details of the XFS file system below. As for Node.js, it is a runtime that utilizes JavaScript on the server-side. It's known for its asynchronous and event-driven architecture, making it highly efficient for handling concurrent connections. It can easily serve web applications with tools like Express. Node.js is a flexible environment for managing the requirements of a web application.
2. **Storage Management**: For our storage solution, I recommend we store files on our XFS Linux system. XFS is designed for scalability and efficient handling of large files and storage volumes. It offers features like delayed allocation and online defragmentation. XFS is optimized for large-scale storage and can handle vast amounts of data efficiently. It's well-suited for scenarios where high throughput and large file support are crucial. It also has several features, such as journaling and metadata logging, that make it resilient against crashes and fast at recovering.
3. **Memory Management**: Linux employs various levels of caching to improve memory access speed. This includes caching frequently used files in RAM (disk cache) to reduce I/O operations and improve overall system performance. The kernel employs a page cache to store frequently accessed data from files, such as images, CSS files, and JavaScript files. The file system cache is closely related to the page cache. It involves caching the metadata of files, which can speed up operations involving file system access. Javascript and Node utilize garbage collection to clear unused objects on the client side as well as the server side, keeping memory available for execution. Node.js also offers buffer management, which allows binary data to be manipulated with raw memory outside of the Javascript object model, which can greatly enhance performance and reliability.
4. **Distributed Systems and Networks**: I recommend deploying the Node.js application within Kubernetes clusters on a cloud computing platform, such as Google Cloud Platform. There are several dependent pieces to such a structure. The Kubernetes clusters consist of multiple nodes (virtual machines) managed by the Kubernetes control plane. The control plane includes the API server and various controllers. Kubernetes uses the Service abstraction to manage networking. Services provide stable IP addresses and DNS names for pods, making it easier for microservices to communicate with each other. Kubernetes is equipped to control connectivity and handle outages. It offers DNS-based service discovery, allowing components to locate each other using service names. This helps handle dynamic scaling and the addition or removal of pods without disrupting connectivity. Additionally, Kubernetes services can distribute traffic across pods using load balancing. This ensures even distribution of requests and helps prevent overloading specific pods. Kubernetes can automatically scale your application based on resource usage. This helps handle increased traffic and prevents overloading
5. **Security**: There are several strategies for protecting user data using GCP and Kubernetes. We should utilize encryption in transit (TLS/SSL) and at rest to protect user data. Kubernetes and GCP provide tools for managing SSL certificates and encrypting data stored in various services. We will want to implement Role-Based Access Control (RBAC) to restrict access to resources. Similarly, we should have network policies to control communication between pods and services, reducing the risk of unauthorized access. Additionally, we'll want to schedule regular scan container images for vulnerabilities. GCP provides tools like Container Analysis for this purpose.

Since Kubernetes will have containers with a Linux distribution using XFS, we can utilize included firewalls like iptables or nftables to define rules for incoming and outgoing network traffic to protect against unauthorized connections.

Built-in security of Node.js will also help secure our information. Node.js's non-blocking I/O model minimizes the risk of thread-related vulnerabilities like race conditions, deadlocks, and buffer overflows, which can be common in multi-threaded environments. Another in-built security feature will be npm (Node package manager) audit that helps identify and fix vulnerabilities in the third-party packages used by Node.js applications.