

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/13/23 | Saram Nadeem | Descriptions added to required portions. |
| 1.1 | 09/30/23 | Saram Nadeem | Added Development Requirements section. |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

Our client The Gaming Room wants to expand their total userbase by making their game available through the web browser. They want to have multiple teams, each with their own unique identifier and have multiple members per team as well as only have one instance of the game can be allowed to exist at once. A solution to this is to create a database or server where the game instances are stored. This would allow users to access one central website that stores all the games code and information.

## Requirements

There are a few requirements for this product. The first few being that the game must have the ability to have one or more teams involved, and that each team will have multiple players assigned to it. There also must be a unique team name, and the ability for users to check if their desired name is taken or not. Finally, the last requirement is that only one instance of the game can exist in the memory.

## [Design Constraints](#_2et92p0)

For the identifiable design constraints, the biggest one would be the server architecture required to host the website and store the information. We must be able to access a bank or memory, creating this is another design constraint. Teams can also not have the same name, so there must be in place to check this as well.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

Through the given domain model, we can see how these classes all relate to each other. Through the class entity, the three classes Game, Team, and Player will be inheriting some of the information. The three courses will inherit the use of id and name which can be seen within their own respective functions as well as the entity class. The GameService class takes in functions from all four of those existing class. It can be seen with id and name functions as well as ones specifying it exactly for either game, team, or player.

**"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](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | When looking at server-side use for Mac, it’s an overall great option because of its ability to consistently provide a solid performance. It can also be considered as a great choice for service hosting as even the low tiers of Mac products have great specs for running and hosting games. On the downside, Mac is not the most common software system that’s used by consumers. This could lead to a lower overall user base as people would need a Mac OS to strictly run the game. | Linux is another great choice when looking to host a web-based application because of its ability to provide a higher level of security and maintain a consistent connection between the users and the server. One downside to Linux would be that it’s also not a common OS. Windows and Mac OS are two of the most common products used by consumer, so using Linux would appeal to a smaller overall target and would be much more difficult to get set up as it can require more knowledge about the OS to get up and running. | Windows is known to have the largest user base amongst consumers and has the highest compatibility with different types of frameworks and software’s. It’s an overall great choice as it’s very easily adaptable and can host a wide variety of server types. One major weakness that can be accredited to the Windows OS would be its lack of security or more necessarily it’s innovations to battle security threats. | Mobile devices can be considered a great choice because it can accommodate to an extremely large percentage of the population. One large identifiable problem with hosting the server side on a mobile device would be the different OS that phones run on. The big two are IOS and android, but you would have to accommodate for those individually as well as other factors such as phone design, size, and certain specs. |
| **Client Side** | When using Mac, it provides a very easy to use and visually appealing OS to the users. It also comes with a great deal of the security and constant software updates which is another plus for the user. On the other hand, developing more than one client may become more difficult as all OS must interact with each other for different users to play the game together. | Clients who use Linux might find themselves having a much more difficult time using the game or service. Since Linux is adaptable, it would require different features which could cause potential issues when attempting to access the game. We would have to ensure that the Linux systems are compatible what what’s set up. | Windows is also extremely adaptable and holds a large portion of the users. This would allow many more people to be able to access the service and stay connected for longer. Again, security may be another issue if it’s not constantly updated or innovated to combat the latest threats. | For mobile users, you would have to ensure that all types of platforms and OS are supported. Since they’re designed for phones, many of these different ones require an in depth and advanced knowledge of how the OS’s cross communicate and function based off each phones specs. There would have to be compatibility testing between all different mobile OS’s and PC OS’s to ensure all users don’t have issues. |
| **Development Tools** | Mac is capable of using a wide variety of IDE’s and programming languages. VS Code, VS, Eclipse, IntelliJ, Java and Python are all great choices to name a few. It also allows for a great deal of maneuverability when developing for Linux and windows which is a great plus as a developer. | Linux can also handle quite a large amount of IDES and programming languages. The only downside is to get access to all of them or most of them, It would require a virtual machine connection to work in the different IDE’s. This itself is not a difficult thing to accomplish as Linux is very accommodating for it. Eclipse and Java are one Ide and programming language that would | Windows is extremely versatile when it comes to IDEs and programming languages. Through the use of VS code we can use almost any language available, and this allows for the use of terminal which makes cross collaboration very easy between developers working on different systems. | Development on a mobile device can be a little more difficult. Xcode which can be used for iOS and SwiftUi are good tools to use when looking to develop upon a mobile device. Apple and android also have development tools that can be used on the Mac or windows OS to develop apps with more ease. |

## Recommendations

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

1. **Operating Platform**: The operating platform which is most recommended for the Gaming Room would be a Linux based system. Since the Gaming Room is looking to expand to other platforms and grow their overall user base, the Linux’s flexibility and cost effectiveness is an amazing choice for this company. The scalability that comes with these Linux systems will be extremely beneficial as the Game gets more and more users which will increase their need for resources across the board.
2. **Operating Systems Architectures**: In Linux systems like Ubuntu Server, all vital operations happen in one efficient core system. This design allows multiple users and tasks, crucial for servers. Linux's reliable Ext4 file system ensures data consistency and quick data handling, essential for managing game data and user interactions. Linux's open-source nature promotes flexibility, making it adaptable to different hardware and software requirements. Its stable performance and robust security features create a reliable foundation, ensuring smooth operations and safeguarding sensitive user data in gaming environments.
3. **Storage Management**: There are a few great storage management options which the Gaming room can choose from. If they are doing it internally and want to control it themselves, they can pick either an SSD or an HDD. HDD stands for hard disk drives, and SSD stands for solid state drives. The major differences between these two are that SSD’s offer a faster and more reliable performance but at a higher overall cost, and HDDs are cheaper altogether but are generally slower and less reliable. SSD’s can be a great option as data will need to be accessed quickly when users are attempting to play the game.

Another great choice would be using a tool such as MySQL which is a database management system. Implementing this to collect and hold the data would be extremely beneficial as it could allow the company to retain user information which they might require for them to play or login to their accounts. It’s also simple and easy to set up and allows for a great deal of scalability so when they have more users across multiple platforms, it’ll be much easier to manage it all.

1. **Memory Management**: Using the Linux operating platform is extremely useful for overall memory management. Linux’s dynamic memory allocation allows for the optimization of RAM usage by swapping between the disk storage and the ram to ensure that quality is sustained for all users throughout the game. They also have a system which is called page-caching, this makes sure to not load the pages which are not in use which helps with overall memory usage.
2. **Distributed Systems and Networks**: To address the distributed systems and networks, the use of a RESTful API can be one of the best choices. They can allow for a quick and easy communication system between the front and back end which is vital in ensuring that the program Is functioning properly in all aspects. To assist in handling connectivity issues and network outages, load balancing mechanisms must be implemented so that available resources can be utilized optimally. With these load balancing mechanisms, the network traffic can be distributed which can help stop interrupted services and mitigate total connectivity issues.
3. **Security**: There are many ways to protect user information between various platforms. To name a few, there is user authentication, data encryption, firewalls and network security, and regular operating system updates. User authentication is extremely simple yet efficient in terms of protecting user data and accounts. Only the authorized user can get access to the account using a password, username, and even an account code which can be sent to the users email or phone number. This can ensure that unauthorized users are not getting access into these accounts. Data encryption can allow for the safety of data being sent from various places to another. A great example of this would be when the user is sending data to our service, or the service is attempting to retrieve information for a part of the service. This ensures that the data cannot just be intercepted and read by unauthorized individuals. Firewalls and network security is another great tool that can be used to prevent individuals from getting into the system intentionally. This creates a layer of safety which can stop data stealing and protect our users from attacks by people looking to take advantage of them. This is where the important of regular software updates comes into play. As hackers and threats become stronger and more advanced, it’s important that the system is continually updated to adapt and improve their own security in the hopes of preventing unauthorized users from accessing the information and potentially harming the user base.

Sources

“What Is Image Compression? | Cloudflare.” *What Is Image Compression?*, Cloud Flare, www.cloudflare.com/learning/performance/glossary/what-is-image-compression/. Accessed 7 Oct. 2023.

“Fundamentals of Garbage Collection - .NET.” .*.NET | Microsoft Learn*, Microsoft, 28 Feb. 2023, learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals.

Work. “Cloud vs Local Storage: Which Is More Secure?” *LinkedIn*, Lunik, 27 Feb. 2023, www.linkedin.com/pulse/cloud-vs-local-storage-which-more-secure-lunik-explorers.

Marijan, Bosko. “Server Operating System: Server Os Types & How to Choose.” *Knowledge Base by phoenixNAP*, 17 Aug. 2023, phoenixnap.com/kb/server-operating-system.

“How to Choose between SSD, SSHD, and HDD Storage for Better Laptop Performance: Seagate Us.” *Seagate.Com*, www.seagate.com/blog/how-to-choose-between-hdd-storage-for-your-laptop-master-dm/. Accessed 15 Oct. 2023.

“MySQL 8.0 Reference Manual :: 1.2.1 What Is Mysql?” *MySQL*, ORACLE, dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Accessed 14 Oct. 2023.