

Game Host

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template 1**](#_Toc110923357)

[**Table of Contents 2**](#_Toc110923358)

[**Document Revision History 2**](#_Toc110923359)

[**Executive Summary 3**](#_Toc110923360)

[**Design Constraints 3**](#_Toc110923361)

[**Domain Model 3**](#_Toc110923363)

[**Evaluation 5**](#_Toc110923364)

[**Recommendations 8**](#_Toc110923365)

[**Citations 9**](#_Toc110923366)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | 8/8/2022 | Nicholas Kreuziger | Completed Recommendations Section.  Added IDE Developer License Costs in Evaluation Section with Hyperlinks to cost pages.  Added Citations Section.  Sunset System Architecture Section due to non-use. |
| 1.1 | 7/29/2022 | Nicholas Kreuziger | Completed Evaluation section to fully outline the pros and cons of various operating systems in relation to web-based server hosting. |
| 1.0 | 7/14/2022 | Nicholas Kreuziger | Outlining the Design requirements for Game Host for The Gaming Room Client. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game serving multiple platforms for a game they have. It is presently only available in an Android App. The desire is to scale this game into a web-based environment. An application that enables one or more teams with multiple players to coexist is required. These games, their teams and those team’s usernames must be unique with functionality to users to disallowed from duplicating or editing any of these values.

We are proposing developing the web-based application in the JavaScript Language. With the script we can compose the storage of games, teams, and their usernames to avoid duplications. Furthermore, we can host the script with ease on a web server, allowing universal access for the target audiences. There are many providers of web server hosting at affordable prices to choose from.

## [Design Constraints](#_2et92p0)

Structurally this game will require the ability to be hosted in a web server. This limits the programming languages that can be utilized. Not every programming language is as easy to host on a web-based application. This will also require paying for a domain to host the web-based application upon.

Design constraints for the web-based distributed environment involve communication between the game and its users. This requires the game to be able to store user-configured parameters of Names and Team configurations, in addition to the various games composed of these User and Team configurations. The application’s design requires them to formulate teams and game sessions without any overlap.

No duplicates can be allowed to exist in addition to multiple games being allowed to run at once. This will require building in checks that disallow any duplicate creations. Users will have to be able to test the availability of game, team, and usernames before their creation.

## [Domain Model](#_8h2ehzxfam4o)

**"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.**

This UML diagram describes the classes required to make The Gaming Room fully operational. This class adheres to a Singleton Design Pattern. A Singleton Design Pattern utilizes only one existence of a Class that serves as a global access point for all objects, this prevents duplication of objects in the program.

The Class **ProgramDriver** initializes the code.

The Class **Singleton Tester** is a class used to test the Singleton functionality. This is useful to verify if the design is adhering to the Singleton Design Pattern before moving forward with development.

The Classes **GameService**, **Game**, **Team**, **Player,** and **Entity** are all interrelated.   
  
The Class **GameService** serves a few different roles. It hosts the list of Games, allows for making new games, and retrieving games. It has a 0 to many relationship with games, as there can only be one GameService but many instances of a Game.

The Class **Entity** serves as a template for the **Game**, **Team,** and **Player** classes. It has the characteristics id, name, and the ability to access those attributes. The Game, Team and Player Classes inherit this class to help streamline the code.

The Class **Game** hosts the **Teams** in the Game, the Game ID and Name and allows for adding Teams to

the game. Inherits id and name traits with its associated accessors from Class **Entity**. It has

a 0 to many relationship (0…\*) with Team, since there can be many Teams associated with one Game.

The Class **Team** hosts the Team information. The Name of the team, Team ID, list of players and ability to add players to a Team reside here. Inherits id and name traits with its associated accessors from Class **Entity**.

The Class **Player** hosts the player id and player name attributes of players. The Class **Player** Inherits id and name traits with its associated accessors and mutators from Class Entity.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Standalone app Mac OS Server was discontinued in 2021, with all its features now being offered as a packaged feature of MacOS.  Mac OS has a market share of 14.64% (Liu, 2021) as of July 2022.  Due to their smaller market share and hardware-based security approach it is extremely difficult to fall victim to a cybersecurity threat.  Hardware is more expensive due to the smaller market share and exclusivity. | Linux has a time-tested and well-developed Community. Resources to aid development and server-support are widely available.  Additionally, due to the OS Source Code being openly available, bugs and exploits are reported by skilled developers regularly. The community is ever evolving to improve the OS.  Linux has a market share of 2.42% (Liu, 2021) of the OS market. | Windows has excellent Third-Party application support. Each Windows Server Distribution comes with 10 years of support.  Windows has a market share of 76.33% (Liu, 2021). The largest number of possible consumers. Making the application Server communicate with Windows users is a top priority.  Windows has a larger market share and a more open Operating System setup. In addition to its larger market share that makes it the largest target of malware and cybersecurity attacks. Great care must be taken with manually configuring the windows server. | Mobile devices can be used to run a web server, although it is not recommended. Applications such as Jetty, Tomcat, Glassfish and Resin can be deployed by mobile devices and function as Mobile Web Servers.  Hardware limitations are the main factor, as a mobile device has limited RAM, Storage, and processing power.  Advantages, Strengths, and Weaknesses aren’t readily available. Given that it’s not common or best practice to host a web server from a Mobile Device. |
| **Client Side** | Licensing for Mac OS is $129.99 initially and $19.99 for upgrading to a newer version from a previous version.  Mac OS is not as popular for hosting Servers. Due to this the cost of hosting them, time-investment and expertise will likely come at a premium. Hardware Cost would be factored in if a Host could not be found.  On a positive note, if you are a regular Mac user you will find Mac Server easy to operate and utilize.   Software availability is more restricted with a Mac OS. If employees are not proficient in Mac systems, there will be a cost to educating employees in Mac Native Software. | Linux is open source. As such there is no cost associated with licensing.  Typical Cloud providers such as Google and Amazon prefer Linux to Windows. Due to its popularity as a Server OS of choice finding a host would be no problem. In the event the cost of hardware was prohibitive.  If the choice to configure a Linux Server natively is made, be sure to find an experienced developer. There is a host of customization and complex tasks associated with Linux Server upkeep. | Windows, due to its popularity and market share, has many hosting options available at a reasonable cost.  Windows Server Standard Edition is $999 dollars, and the Data Center Edition is $6,155 dollars. Standard Edition will be ideal for the hosting initially and Data Center would come after substantial increases in server load and moving into the cloud.  If the choice to configure a Windows Server natively is made it is prudent to find an experienced developer. There is a host of customization and complexity associated with Windows Server Upkeep. | It would be cost-prohibitive to pursue this option as the Server would be able to store and handle very minimal amounts of data and traffic.  An investment in the most powerful and storage-scalable mobile device would have to be made. Additionally, the amount of developer tools would be small, a developer would need to be hired to custom code most of the application. |
| **Development Tools** | Mac OS natively runs Objective-C and Swift programming languages.  [Visual Studio](https://www.microsoft.com/en-us/d/visual-studio-professional-2022/dg7gmgf0d3sj?activetab=pivot:overviewtab) ($499), [Atom](https://atom.io/) (Free), [Eclipse](https://www.eclipse.org/downloads/) (Free) and [XCode](https://developer.apple.com/xcode/) ($99/Year for Developer’s License) are popular IDE choices in a MacOS environment. | Linux development can take the form of C/C++, Java, or Python.  [Gnome Builder](https://apps.gnome.org/app/org.gnome.Builder/) (Free), [Visual Studio](https://www.microsoft.com/en-us/d/visual-studio-professional-2022/dg7gmgf0d3sj?activetab=pivot:overviewtab) ($499), [Atom](https://atom.io/) (Free), [Eclipse](https://www.eclipse.org/downloads/) (Free) and [CodeLite](https://downloads.codelite.org/) (Free) are popular IDE choices for a Linux environment. | Windows predominantly uses C# and .Net programming languages.  [Visual Studio](https://www.microsoft.com/en-us/d/visual-studio-professional-2022/dg7gmgf0d3sj?activetab=pivot:overviewtab) ($499), [Eclipse](https://www.eclipse.org/downloads/) (Free), and [NetBeans](https://netbeans.apache.org/download/index.html) (Free) are popular IDE choices in a Windows environment. | Android Development was Java-based and is now predominantly Kotlin-based. Kotlin is a version of the Java Language.  [Visual Studio](https://www.microsoft.com/en-us/d/visual-studio-professional-2022/dg7gmgf0d3sj?activetab=pivot:overviewtab) ($499), [Android Studio](https://developer.android.com/studio) ($25 one-time to publish to play store), [IntelliJ IDEA](https://www.jetbrains.com/idea/download/?fromIDE=#section=windows) ($49/User/Month) and [DroidScript](https://droidscript.org/) (Free) are all touted as excellent IDE options for Android App Development.  iOS development takes place in Swift or Objective-C. [XCode](https://developer.apple.com/xcode/) ($99/Year for Developer’s License), [AppCode](https://www.jetbrains.com/objc/) ($89/Year), [Atom](https://atom.io/) (Free) and [Sublime Text 4](https://www.sublimetext.com/download) (Free) are popular IDE choices for iOS app development. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

The Operating Platform of choice would be a Linux Server. The cost for Linux Server is free and the platform is less prone to cyber security threats (Pedamkar, n.d.). Linux had a total market share of 33.9% of the Server OS Market for its RedHat distribution alone in 2018, this is not including other flavors of the Linux Server OS (Server Operating System Market Share | T4, n.d.).

You will find a community readily available to assist you with any growing pains that accompany adoption and maintenance of a new platform.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

Linux is an open-source software operating system built around the Linux kernel. Iterations of Linux are called distributions, usable for both desktop and Servers. The three bodies of code in the Linux System are the Kernel, System libraries and System utilities. The Kernel maintains all important abstractions such as virtual memory and processes. The System libraries define the set of functions that applications can talk to the Kernel with. System utilities are programs that perform specialized tasks ranging from temporary initialization of system tasks to permanent background processing such as delegating network connections. All processes are separated between Kernel and non-Kernel code. Kernel code has access to all system resources, whereas non-Kernel Code does not (Silberschatz et al., 2008).

Linux provides many lucrative features in its architecture. Linux is stable, it can run for years without fail. Linux can multitask multiple processes without degradation of stability. Linux also does not need rebooting for any system changes. Furthermore, Linux is a small-footprint OS, meaning it can function admirably on just about any processor or machine architecture (Five Reasons Linux Beats Windows for Servers, n.d.).

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

The storage required for this game will be 1600 megabytes of image data in addition to the stored information about game sessions, teams, and users. There are many admirably priced cloud storage services that could be utilized to cut the cost of storing all the game and user data on-site. [Box](https://www.box.com/pricing/) ($25.00/month/user) and [DropBox](https://www.dropbox.com/plans) ($24/month/user) are two excellent cloud-based storage options that would allow for routing the user and game data to a cloud-based destination. This would significantly cut the costs of buying the storage for local servers. With our utilization of Cloud Storage, we will free up our Storage availability for virtualization to process active user game sessions instead of writing/retrieving data from local storage.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

The memory requirements of the local server will largely depend upon the number of users and the system requirements of each game session. Linux manages memory usage with Virtual Memory. Linux will decode instructions and read or write from the system memory, translating the virtual address encodings of physical address storage. After reading the addresses it will use the stored information (whether in RAM or Physical Storage) to execute the task. Performance will vary depending upon whether the address is in RAM or Cloud Storage. With our utilization of Cloud Storage, we will free up our Memory to process active user game sessions instead of writing/retrieving data from our local storage. Our recommended form of Memory Management will increase the number of users we can serve at any given time.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

Linux is a stable server OS. As a result, it can be relied upon to maintain it’s network throughput without any significant down-time. Since we will be hosting a web server based upon Linux, any platform can access this web server to utilize the game and our server resources. This solution will be cross-platform and greatly expand availability of Draw It or Lose It to new users. Network Connectivity will be largely dependent on our Internet Service Provider on-site, in addition to our Cloud Providers for storage.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

Linux is a secure OS. From the beginning Linux was designed to be a multiuser OS. There is always a Root User (Administrator) built into every deployment. This makes the OS run in a modular fashion that inhibits the ability of non-administrative users. In a Linux OS non-admin users do not have access to core system settings. This makes violations and security gaps very easy to recognize and solve (Pedamkar, n.d.).

In the Linux World, when an attack does occur, the active community of developers and users will find and fix the bug (Five Reasons Linux Beats Windows for Servers, n.d.). Linux is seldom the target of cyber attacks or hacking. This is largely in part due to its dedicated developer community in addition to its multiuser OS features described in the paragraph above.

## Citations

*Five Reasons Linux Beats Windows for Servers*. (n.d.). PCWorld. https://www.pcworld.com/article/502784/why\_linux\_beats\_windows\_for\_servers.html

*Server Operating System Market Share | T4*. (n.d.). Www.t4.Ai. Retrieved August 9, 2022, from [https://www.t4.ai/industry/server-operating-system-market-share#:~:text=While%20Red%20Hat%20and%20Microsoft%20are%20the%20most](https://www.t4.ai/industry/server-operating-system-market-share%23:~:text=While%20Red%20Hat%20and%20Microsoft%20are%20the%20most)

Liu, S. (2021, September 10). *Desktop OS market share 2013-2018 | Statista*. Statista; Statista. Retrieved August 9, 2022, from <https://www.statista.com/statistics/218089/global-market-share-of-windows-7/>

Pedamkar, P. (n.d.). *Linux vs Windows Server* [Review of *Linux vs Windows Server*]. Educba. Retrieved August 9, 2022, from <https://www.educba.com/linux-vs-windows-server/>

Silberschatz, A., Baer Galvin, P., & Gagne, G. (2008). *Operating System Concepts* (8th ed.) [Review of *Operating System Concepts*]. John Wiley & Sons, Inc. <https://learning.oreilly.com/library/view/-/9780470128725/>

‌

‌‌