

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc2101765942)

[Table of Contents 1](#_Toc1489430186)

[Document Revision History 2](#_Toc839887045)

[Executive Summary 2](#_Toc822213724)

[Requirements 3](#_Toc1759978111)

[Design Constraints 3](#_Toc2112940063)

[System Architecture View 3](#_Toc108276467)

[Domain Model 3](#_Toc56384679)

[Evaluation 4](#_Toc1401331230)

[Recommendations 5](#_Toc2023014382)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/09/2023 | Brandon Randall |  |

**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)

The gaming Room is hoping to move their game, Draw It or Lose It, from an android based app to a web-based app. Currently the client has no experience with web-based environments.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

* Translating the game from an android app to a web-based program will involve different programming languages and understanding the differences between a web-based program and an android app. Currently, the company, The Gaming Room, does not have any experience with this transition so outside help or internal training will be necessary to build a functioning web app.
* A web-based app will need to be tested using multiple devices to ensure that it runs properly. With the application being planned to be used on both mobile and desktop devices some of the popular devices should be tested to ensure they run correctly across the board.
* Different UI/UX designs would need to be developed to make the application accessible for both mobile and desktop users. Most desktops use a primarily horizontal screen that is far larger than a phone, and phones tend to be used vertically on smaller screens. There is also the difference in power between the systems which could lead to a mobile UI needing to be toned down to run efficiently.

## [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)

Within the program we have established Entity as a super class for the Game, Team, and Player classes, this allows these classes to inherit the variables and methods of Entity, leading to less repetitive code in these classes. Within the GameService class an instance of the service has been used rather than a typical constructor, this allows only one instance of a service to be available at any time. Within the GameService, Game, and Team classes we have methods to add games, teams, and players, which allows for multiple unique instances of these to be created. To ensure these instances are unique the methods iterate over the Lists incorporated into each class to ensure that the object does not already exist before allocating memory to a new object.

**"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** | MacOS is developed by Apple and runs best with iOS and macOS devices. It is easy to use and has a lot of support, but only runs on Apple hardware, which can be expensive. This likely wouldn’t be a great option unless the focus was primarily on MacOS/iOS devices. | Linux is open source and free, while also being very secure. Linux runs on a variety of different hardware types and allows for a lot of advanced integrations, but this also requires advanced technical knowledge. There are also many different distributions of Linux that have their own pros and cons, such as lack of ongoing support. | Windows is a well-rounded OS with wide application support and many features. Windows also runs on many different hardware types. The main drawback of Windows is its popularity, which makes it prone to security threats. Another drawback is the need for user-based licensing, which can be expensive. Apple users may also have issues connecting to Windows-based servers. | While mobile device servers do exist, they would not be a good option for dependent and long-term use due to the limitations of mobile devices. |
| **Client Side** | Macs are easy to use and have good support. They are limited by a few factors, such as limited software availability and high price points. Mac development also must take place using MacOS, so developers either need a Mac or a virtual machine running MacOS. | Linux is free but has a high learning curve for use. Most consumers do not use Linux as a primary OS. Linux is a very versatile OS with many advanced options available to utilize, if the user has the skills. | Windows is the most popular OS for users and is easy to use. It supports a multitude of software. As with the server side, Windows’ popularity makes it prone to security issues, so developers should have a good understanding of these security risks. | Mobile devices come with different OSs, the most popular are iOS for apple devices and Android for most others. It is possible to develop directly on the device, but with limited hardware it makes more sense to use a PC running one of the other OSs; iOS specifically needing to be developed using MacOS. Development for these devices usually requires a virtual machine or connected devices to verify that the application runs correctly on multiple devices. |
| **Development Tools** | Apple uses their own proprietary programming language, Swift, for apps on iOS and MacOS. Their website, Swift.org, recommends a few different IDE’s such as Atom and VS Code. | Linux itself is programmed using mostly C, but applications can be made using C, C++, Perl, and Python. These languages have many IDEs available for multiple platforms. | Windows apps can be created using a multitude of languages such as C, C#, Python, and Java. There are many IDEs that can be used for these languages. | Depending on the devices OS you could use Swift for Apple applications and Java or Kotlin for Android applications. for Java and Kotlin development IDEs such as Eclipse and IntelliJ, as well as Androids own IDE, Android Studio. |

## 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**: I would recommend Linux as the operating platform for this application. The server cost is free, it supports many different languages, and can communicate well with both Mac and Windows systems. Linux also provides far more security over Windows and is not as limited in capabilities as Mac. The support for multiple languages allows The Gaming Room to potentially develop their code on their existing operating systems and then launch them using Linux, which is a lot more cost effective compared to something like Mac, which uses a proprietary language.
2. **Operating Systems Architectures**: Linux is designed similarly to a typical OSs, but one thing that can be helpful on the server side is the inclusion of “service” as a user type to run several types of services, such as a server. Linux’s systems also run more efficiently and perform better than Windows or Mac systems for a plethora of reasons, such as being open source, which allows users to modify the source code of the OS to meet their own specific requirements. Linux also updates frequently since the community can fix bugs quickly, and, unlike Mac or Windows, Linux does not require a restart for software and system updates. This is extremely important for a server since a restart would lead to downtime for your site.
3. **Storage Management**: I would recommend a physical storage solution, such as a network attached storage unit, for this program. While cloud storage can be useful, it can also become expensive in the long term. With this being a relatively small application, it would be more cost-effective to spend a larger one-time payment on a physical NAS than to make subscription payments for cloud storage. Both forms of storage work similarly, so if cloud storage looks like a better option in the future, it would be easy to transition for additional reliability.
4. **Memory Management**: Memory management can be done efficiently using a client-server setup and a process called paging. The program will be stored in the server and utilize the user’s device’s memory. Paging allows the device’s memory to believe it has more space than it does by utilizing the storage on the server.
5. **Distributed Systems and Networks**: With the program being a web-based application it is easy to allow users on different browsers to access the game from different devices. The program utilizes a client-server structure, with the server storing the site information and the client requesting it. The client's device is used to perform actions using the server's data, keeping the data used for the application secure as well as reducing the memory and storage used by the client by only returning what is requested. There are many variables in this system that can affect performance, such as server size on the server side and connection quality on the client side, but overall, this system allows for low down times and high efficiency.
6. **Security** On the client side of the application, user input validation is important to ensure users cannot manipulate the application using malicious inputs. Password protection is also another important security measure to ensure users data is secure to their own account. Server-side security involves user permission management to ensure that only specified users can change the server-side components. The program itself should incorporate error handling to ensure that attackers cannot abuse the code to get unexpected results. Linux itself is a very secure OS mostly due to its open-source nature. Linux users can collaborate or work on their own to fix errors and push updates to the system, which can be a lot more effective than Windows or Macs security update system.