

Draw It or Lose It

# **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 | 07/16/2023 | Brandon Luellen | Completed Executive Summary, Design Constraints, and Domain Model sections. |

**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 team at The Gaming Room has requested our assistance in creating an online version of their game, Draw It or Lose It. Our task involves devising a game with the follow specifications: incorporating multiple teams and accommodating multiple players within each team, implementing a system that verifies the uniqueness of team names to facilitate players in checking name availability, and ensuring that only a single instance of the game can be active in memory at any given moment.

## Requirements

The requirements for the game are:

* The game must be web-based.
* The game must allow multiple teams and multiple people on those teams.
* Only ONE unique instance can exist in memory at a time.
* Team names CANNOT be duplicated.

## [Design Constraints](#_2et92p0)

* The game mut be written in a language that can be supported in any web-based environment.
* Must utilize specific design patterns to only allow single unique instances into memory.
* Teams should be able to accommodate the increasing and decreasing number of players as needed.
* Multiple teams are required and should be checked prior to initiating games.

## [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)

The classes Game, Team, and Player all derive from the Entity superclass, enabling them to utilize the attributes and operations defined in Entity without repetition. The GameService, Game, Team, and Player classes are interconnected and can depend on each other as needed, or none at all. The ProgramDriver class steers the package and inherits/utilizes the SingletonTester class during its execution.

**"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** | While Mac OS can serve as a platform for server-side applications, its adoption is not as widespread as Linux due to its comparatively higher expense and smaller market share. Certain server-side applications, particularly those created with Apple's Swift programming language, might find macOS to be a more suitable option. Nonetheless, it is essential to acknowledge that macOS lacks the extensive server-side ecosystem present in Linux, making it less favorable for all types of server-side applications. | Due to its stability, reliability, and security, Linux has become a favored option for server-side applications. Specific applications and web servers like Apache and Nginx are tailored to operate seamlessly on Linux. Moreover, Linux boasts a remarkable level of customization and flexibility, rendering it ideal for an extensive array of server-side applications. | Windows is frequently employed for server-side applications, especially those that need to be compatible with Microsoft's .NET Framework. Nonetheless, in comparison to Linux, Windows is generally considered to be less stable and reliable, potentially making it more susceptible to security threats. Additionally, the ecosystem of server-side applications and web servers available for Windows is smaller than that of Linux. | Server-side applications are generally not run on mobile devices due to their lack of essential hardware or software capabilities required for supporting such applications. |
| **Client Side** | Mac OS possesses extensive compatibility with a diverse array of client-side applications and boasts a vast ecosystem of native software offerings. Numerous well-known client-side applications, including Microsoft Office and Adobe Creative Suite, are readily accessible for macOS. Nonetheless, certain specialized or proprietary software might not be fully compatible with the operating system. | Linux supports a diverse array of client-side applications encompassing web browsers, email clients, and productivity software. Linux users can access well-known client-side applications like Chrome, Firefox, and LibreOffice. Nevertheless, certain specialized or proprietary client-side software might not be fully compatible with the Linux operating system. | Windows supports a vast array of client-side applications, boasting the most extensive library available, encompassing a wide selection of popular software on its platform. Moreover, the operating system benefits from a vast ecosystem of specialized and proprietary software. Nonetheless, it's worth noting that certain applications might not be fully optimized for Windows, necessitating potential adjustments or additional configuration to ensure smooth functionality. | Mobile devices are predominantly employed for running client-side applications, such as web browsing, email, social media, and entertainment purposes. The majority of these popular client-side applications are accessible on both iOS and Android platforms. Nevertheless, certain specialized or proprietary software might not be accessible for mobile devices, especially for older or more niche applications. |
| **Development Tools** | Mac OS is widely favored for software development, especially in the realm of mobile app development. Several renowned development tools and frameworks like Xcode and Swift are specifically designed for macOS. Nevertheless, when it comes to certain types of development, such as web development, some developers may find Linux or Windows more suitable options. | Due to its open-source characteristics and the abundance of free and open-source development tools, Linux is highly suitable for software development. Notably, it enjoys particular popularity in the realm of web development, boasting numerous widely-used frameworks and libraries like Ruby on Rails and Node.js, which are readily available for the platform. | The Windows operating system boasts an extensive range of development tools and enjoys significant popularity in the realm of enterprise software development. Numerous widely-used programming languages like C# and Visual Basic have strong affiliations with the Windows platform. Furthermore, Windows serves as an excellent choice for crafting desktop applications, particularly those tailored for the Windows environment. | While mobile devices do offer some development tools for app development, they are not the primary choice for software development. Despite the availability of popular frameworks like React Native and Flutter, which cater to iOS and Android app creation, desktop or laptop computers are generally better suited for software development compared to mobile devices. |

## 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**: After evaluating the necessary skill level and the budget-friendly aspect, I suggest initiating the project using the Windows OS within The Gaming Room. Windows OS offers the advantage of numerous helpful IDEs, streamlining and expediting the overall process.
2. **Operating Systems Architectures**: Windows offers essential services utilized by all applications designed for the Windows operating system. These services empower applications to present a Graphical User Interface (GUI), granting them access to system resources and beyond. These applications encompass functionalities related to Graphics and Multimedia, messaging, as well as web services. These versatile services are accessible through both user accounts and dedicated servers.
3. **Storage Management**: Windows provides the opportunity to expand the system's storage capacity, particularly when there's space available for incorporating additional disks into a storage pool. This, in turn, can enhance performance and safeguard data integrity through the utilization of Storage Spaces. Moreover, it offers the potential to optimize existing storage by eliminating unnecessary clutter and harnessing the benefits of cloud storage solutions.
4. **Memory Management**: As we initiate the project, a substantial number of files will need to be generated and utilized. Although creating these essential files from the ground up is an option, there is a significant opportunity for enhanced efficiency and speed by leveraging game engines. These software development environments, tailored for game developers, come equipped with a variety of pre-built libraries and effects. Utilizing these resources can greatly expedite our progress compared to manual coding. Maintaining a personal library database on our computer will aid in organizing these files effectively.
5. **Distributed Systems and Networks**: While I suggest initially focusing on window-based development, I acknowledge the need to eventually encompass all platforms, regardless of their operating systems or devices. A potential solution for this project lies in cross-platform game development tools. Among these, 'Unity' stands out as a widely adopted and budget-friendly game engine capable of supporting various platforms like Windows, Linux, Android, iOS, and more. Selecting an appropriate server is equally crucial. Given the potential for numerous concurrent users, it's imperative to opt for a robust network and server infrastructure capable of consistently handling high volumes of traffic.
6. **Security**: Similar to all other operating systems, Windows features its own security software. While this integrated program might function effectively, there are alternative options you can consider purchasing to elevate the overall security of your system, especially when handling sensitive information like user data. McAfee and Norton Antivirus are prime examples of such beneficial programs. These software solutions serve as safeguards, shielding us from harmful interactions with malware or viruses that could potentially endanger both user data and our company's vital information.