

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 3.0 (Updated Evaluation Table)

## 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 | 11/20/24 | Yusuf Bajwa | <Brief description of changes in this revision> |

**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 objective of this project is to develop a web-based game, loosely inspired from the 1980s television show game, Win, Lose, or Draw. “Draw it or Lose it” challenges players to guess puzzle photos. In order to ensure that the game runs smoothly, I propose that we use OOP and singleton patterns.

## Requirements

Business:  
1. The game will have the ability to involve one or more teams.

2. Each team will have multiple players.

Technical:  
1. Each game, team, and player must have a unique instance.

2. The game must support a web-based environment, while leveraging OOP.

## [Design Constraints](#_2et92p0)

The web based environment involves multiple users interacting with a server simultaneously.

It must be able to support multiple teams, with multiple players concurrently.

The performance must remain steady.

This means, the system must handle latency and large loads properly.

It must have the ability to be very scalable.

## [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 UML diagram represents the game system. GameService manages a single instance of games, teams, and players. It utilizes the Singleton pattern for central control. The Entity class is the base for Game, Team, and Player, this is a good way of utilizing OOP, specifically inheritance. Composition is used to model relationships, which helps to ensure organization and efficient management.

**"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** | Mac provides a stable environment for web hosting and supports server-based deployment through tools like Apache and Nginx. However, it is less common in enterprise environments due to a lack of scalability options compared to Linux. Licensing costs are high since macOS requires Apple hardware, which increases initial and maintenance expenses. | Linux is the most popular platform for web servers due to advanced stability, security, and flexibility. It supports server deployment with tools like Apache, Nginx, and Docker. It is open-source, so there are no licensing costs for the OS, making it cost-effective for scaling. Advanced expertise is needed, which is sure to be a downside, however, its popularity ensures lots of community support. | Windows supports server deployment via IIS (Internet Information Services) and third-party tools like Apache. Licensing costs are higher due to Microsoft’s unique software. The Windows Server is easy to use and well-supported but not nearly as efficient compared to Linux’s in handling large-scale traffic. Making it cheaper than Mac, but more expensive than Linux. You won’t need as much expertise as Linux but it will not be nearly as optimal. | Mobile devices are definitely not nearly as suitable for hosting web applications due to their limited processing power and storage. Instead, a better alternative for mobile devices would be cloud hosting services. |
| **Client Side** | Mac offers a user-friendly interface and supports modern web browsers like Safari, Chrome, and Firefox. Ensuring compatibility with Safari requires additional testing, increasing development time. But as usual, supporting Mac clients may also increase costs as developers would likely need access to macOS devices. | Linux provides high flexibility and supports all modern web browsers, but its user base for desktop environments is smaller. Testing may require ensuring compatibility with some more niche browsers. Expertise in Linux environments might be necessary for debugging, which would of course, be increasing development time. | Windows is widely used, so ensuring compatibility with major browsers like Edge, Chrome, and Firefox is critical. Development and testing are streamlined due to the platform's popularity. Windows-based tools simplify development for desktop clients. | Developing for mobile platforms requires responsive design and frameworks like React Native or Flutter to ensure compatibility with both iOS and Android. Testing on multiple devices and operating systems will increase costs and development time. Expertise in mobile development is probably the most important compared to any other platform. |
| **Development Tools** | Mac development often involves Xcode and Swift for iOS applications. Cross-platform tools like React Native, Flutter, or Unity can be used for broader compatibility. Licensing costs are minimal unless proprietary tools are chosen. The development team should be familiar with macOS. | Linux supports tools like VS Code, Sublime Text, and Eclipse. It is ideal for web development due to its open-source nature, ensuring no licensing costs. However, managing cross-platform compatibility might require additional frameworks or teams skilled in Linux environments. As usual in Linux, expertise would be a good idea. | Windows offers extensive development tools, including Visual Studio and .NET. It supports cross-platform development but might require additional configuration for compatibility. Licensing costs for tools like Visual Studio can add up. A team with expertise in Windows-based tools will be essential. | Development for mobile devices depends on platforms like Xcode for iOS and Android Studio for Android. Cross-platform frameworks such as Flutter and React Native reduce complexity but require skilled developers. Testing across various devices increases development costs and demands expertise in both iOS and Android environments. |

## 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 recommend Windows as the operating platform for "Draw It or Lose It." Windows is a widely supported operating system for developing web-based applications, which makes it a versatile choice for scaling across different environments. It provides extensive documentation, developer support, and a relatively gentle learning curve, especially compared to other platforms like Linux. All of these factors can contribute to making sure that the development and maintenance of the software will be efficient and manageable.
2. **Operating Systems Architectures**: Windows uses a client-server model that works with both 32-bit and 64-bit processors, this allows it to run on a wide range of hardware. Which means that it is accessible to a very large number of users. It also supports multitasking and multiprocessing, which will help to improve the application’s performance, and it could be the difference between running smoothly or not.
3. **Storage Management**: For storage management, Windows employs the NTFS (New Technology File System). NTFS offers powerful performance and reliability, supporting large volumes of data efficiently. It includes features like file compression, disk quotas, and built-in encryption, which are all crucial for handling the expected data load of "Draw It or Lose It" and ensuring data integrity.
4. **Memory Management**: Windows uses virtual memory to expand the capacity of physical RAM. By using paging and swap files, the system allocates memory to processes efficiently, making sure to deliver a smooth performance even on devices with limited resources. This is especially important for keeping applications responsive during busy times.
5. **Distributed Systems and Networks**: To enable communication between platforms, "Draw It or Lose It" can use Windows-based software and network protocols like RESTful APIs or WebSocket. These technologies allow real-time data exchange between servers and clients, no matter what the device it is being used for. The app can also use load balancers and redundancy to deal with connectivity problems and keep the service running during outages. However, dependencies in the system, such as server uptime and bandwidth, will need to be monitored and optimized for smooth operation.
6. **Security**: Windows offers a few built-in security features, like Windows Defender and BitLocker to protect user data. Windows Defender defends against malware, and it also protects against unauthorized access, while BitLocker encrypts sensitive data stored on the device. To protect data transmission, protocols like HTTPS and TLS should be used. Some additional steps can be taken, like two-factor authentication and regular updates, which will strengthen security and help protect user information from breaches.