

Draw it or Lose it

# **CS 230 Project Software Design**

Version 3.0

## 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 | 5/24/2025 | Clayton Houle | Initial software design |
| 1.1 | 6/7/2025 | Clayton Houle | Second Design Draft |
| 1.3 | 6/20/2025 | Clayton Houle | Final Design Draft |

**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 software design problem lies in creating a game application that is web-based, in Java, and will be created using object-oriented principles. (OOP) This game will utilize multiple classes for across the board such as the following: GameService, Game, Team, and Player, as well as others that wil interact with each other. All of which will be connected via the Entity class, which will provide attributes ID and name for all entities. Following this design, the team can fulfill the requirements for the game, as well as the problem at hand.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* **Web based distribution environment:** This game application needs to be developed in a web-based distribution environment. This means that the application must be able to be accessible over the internet, support multiple concurrent players, and it must be able to handle any potential network latency for security issues.
* **Java Programming Language:** The application will be coded in Java. This limits the technology window to exclusively Java-based libraries, tools, and frameworks.

## [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 design consists of the following Classes: GameService, Game, Team, Player, Entity, ProgramDriver, and SingletonTester.

The Entity class will be the base class for all entities in the app. This will give all subclasses of Entity common attributes such as ID and name and gives other classes and inheritance of said attributes as well as behaviors of the Entity class. The Entity class will improve efficiency and avoid writing redundant code throughout the project.

The GameService class acts as the central class for managing the game. It contains lists of game names and IDs and methods to receive and add existing and new games respectively. This class follows a singleton design to ensure only a single instance of any given name or ID is used when being called.

The SingletonTester class <<uses>> the ProgramDriver to test behavior throughout the game when played.

The Game class represents the actual Game. It has a team attribute will be used to maintain a list of teams participating in games, as well as has methods to add teams and retrieve a string representation of a team.

The Player class simply has methods that are inherited by Team, which is inherited by Game to be called and used.

**"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** | OS X Server is available, however, it is very expensive to use if the owner wishes to own the hardware. | Linux is the most popular webhosting OS and is utilized by most big companies. Linux is an opensource OS, so license costs and fees, as well as maintenance, tend to be cheaper compared to other OS. | Windows is a GUI based server, meaning that most applications will run smoothly using the Windows OS.  When compared to Linus, license costs are quite high.  For web hosting, there may be limited options when compared to Linux. | Mobile devices are not quipped for multi-user serving, making it difficult for web-based hosting that has more than one user at a time.  The cost would be unknown, as the hosting tools would need to be designed and made by us.  They are also not scalable and are limited compared to other OS |
| **Client Side** | We would need a Mac computer running the latest version of Xcode.  To use the macOS SDK, we would need a team that specializes in Objective-C and SWIFT.  When compared to Windows, there is a smaller market opportunity, as Windows is the market leader in OS’. | Linux development uses the most common programming languages – Java, Python, Or C/C++, so no specialization is needed.  Multi-user support is available for Linux. | Windows is developed using C# or .NET, which are common languages.  Windows has decades worth of time as a multi-user platform.  Windows is the leading preferred OS on the market. | Mobile devices are not designed as a multi-user platform.  The Android SDK uses Java, and the iOS SDK uses SWIFT, so a specialized team would be needed for iOS. |
| **Development Tools** | macOS uses objective-C and SWIFT for programming languages.  Xcode is a common IDE for Mac development.  Xcode is $99 USD for each developer, per year. | Linux development can be used in most languages, primarily Java, C/C++, or Python  Python, C++, and Java IDE’s are usually free e.g PyCharm for Python development, Eclipse for Java, and C/C++, | Windows development uses .NET and C# languages.  Microsoft’s Visual Studio is a popular IDE that offers many plugins and iterations.  Visual Studio costs anywhere from $45 to $250 USD per user, per year. | Android SDK uses Java and can be used via Android Studio, which is free to download.  iOS is exclusively developed using Objective-C and SWFIT, which needs Xcode, which costs about $99 USD per user per year. |

## 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**: Windows is a recommended operating platform. Windows large userbase will give the most opportunity to most users, as well as developers on the team. Windows can also easily be integrated with Android systems, providing the team with more time to develop for stand-out platforms, such as macOS.
2. **Operating Systems Architectures**: I believe developing with hybrid architecture is important for Draw It or Lose It. Using hybrid architecture is safer when compared to a monolithic architecture system. This gives us more customization options as well as giving us more, better options to pull from when compared to monolithic architecture.
3. **Storage Management**: Since Microsoft works well with Windows, we should utilize Microsoft Azure for our primary storage system. There are many advantages to using it, such as its competitive pricing, consistent updates, and concurrent customer support, all of which prove beneficial for the ongoing support aspect of Draw It or Lose It.
4. **Memory Management**: Windows uses virtual memory to extend its available physical memory (RAM) by using a part of its hard drive as a swap file. This five Draw It or Lose It the ability to draw from more memory than is physically installed. To support this, Windows OS utilizes various memory allocation techniques to lower memory fragmentation and utilizing dynamic memory allocation.
5. **Distributed Systems and Networks**: To communicate across platforms, the most efficient options would be to utilize a cloud-based service. The central server will manage communications and data exchanges between all devices, regardless of differing platforms. This allows us to scale and add support with ease. Distributed software systems rely heavily on network connectivity. Meaning that if there are any issues with a network, than the server will suffer consequences. Latency is also a concern as if the network is not strong enough, than delays in the game could occur. For outages, they occur often and can only be resolved using a few strategies such as using redundancy or a failover system, which means to have copies of datasets and backup systems to lessen the damage an outage can have.
6. **Security**: to protect user-data across networks and platforms, a layered approach must be used. At the user level, multi-factor authentication with a strong password could be a simple solution. MFA gives a physical requirement to log into an account, as most MFA authenticators can only be accessed through a mobile device. Keeping these systems up-to-date is very important for security because present security systems are broken all the time, so in order to combat this, security needs to change and evolve constantly.