

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | 10/09/2022 | Nick Basile | Recommendations Added |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to develop Win Lose or Draw as a web-based game. They do not currently have the skills in building this in a web-based environment, so they have instead asked for assistance in the development. The game must meet the design constraints presented in the request and the technical requirements will be adjusted as needed for the software.

## [Design Constraints](#_2et92p0)

* Needs at least one or more teams to participate
* Each team contains multiple players
* Only one game instance can exist at any given time
* Unique Game and Team names
* Multi-platform accessible

## [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 Team, Player, and Game classes inherit from the Entity class which makes Entity the superclass. The GameService class references the Game class which inherits from the Entity class, and the class references to eachother is done through aggregation. The ProgramDriver and SingletonTester classes are used to execute the application and test instances of it.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS is not a common use for server applications. As MacOS typically is better for client usage, it is not recommended. | Linux is currently one of the most popular worldwide for hosting server software. Because of its extremely low overhead, and non-existent cost, it is an excellent choice. | Windows offers server capabilities as well, but the downside is that Windows server applications are typically expensive to implement and maintain. | A mobile device is ineligible for consideration as a server. A server needs to be stationary, and maintain a constant current with minimal interruptions. |
| **Client Side** | The cost of the MacOs would be similar to Windows, as the both Operating Systems require purchasing. Time and expertise would go hand in hand directly proportional to how knowledgeable the client is with MacOS | Maximum expertise and time, with minimum cost. Linux is open source and free, however it is the least popular OS and requires a good knowledge of how to use properly. | The cost would be similar to MacOS. Very few skills would be required client side to use Windows, as it is very intuitive and widely used. Windows is the most used OS in the world, and nearly every person is capable of using it. | Cost could vary, depending on the implementation from a typical mouse and keyboard to a touch screen environment. Minimum expertise and time required for users as all mobile devices are extremely similar. |
| **Development Tools** | Javascript and Swift as the two most popular languages and an IDE to run them would be required. | C++, Python are the two most popular languages used in Linux, and an IDE to run them would be required. | Python, JavaScript, and the C family of languages are the most popular. IDE’s to run them, such as Visual Studio would also be required. | Swift, for Apple phones, and Java for Android based devices. An IDE for either can be found relatively easily such as Eclipse or Xcode. |

## Recommendations

1. **Operating Platform**: I would recommend Windows as the operating system, as it is the most commonly used Operating System in the world. The cost would be similar to developing it for MacOS while retaining a significantly larger userbase.
2. **Operating Systems Architectures**: Windows provides multiple easy to use graphical interfaces that can be used by the Gaming Room to interface with. The can both save on system resources by accessing already existing elements and can also integrate with other Windows applications easier if needed.
3. **Storage Management**: Windows comes with Storage Sense, and Disk management built into the OS. The Gaming Room would use the onboard Disk management to deal with storage management issues.
4. **Memory Management**: Windows also has a build in memory management system. Very little overhead would be required by The Gaming Room for this, as it would handle all the memory allocation for pictures and data storage.
5. **Distributed Systems and Networks**: The best possible scenario is to have a client and server-side application. The server-side, would run and offer access of the data to the Clients while being versatile enough to handle multiple operating systems. The ideal Server would be Linux for this task, as it’s minimal overhead and free cost would be perfect for hosting. The network connected to the Server would need to be strong enough to handle multiple clients accessing it at once, as well as good stability and backup of data in case of data failures.
6. **Security**: Windows comes with build in security software such as Windows Defender. In addition, the data that is sent back and forth between the Client and Server can and should be encrypted, so that user data even if accessed is not at risk of being exposed for malicious reasons.