

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 |
| --- | --- | --- | --- |
| 1.0 | <03/27/22> | William Leaver | Initial |

**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)

For this Software the client has requested that there be multiple teams with multiple players. To do this, we will have two separate classes one for the player and one for the team. The team class will interact with the player class. The game and team names need to be unique. To complete this requirement, we will implement a check to only add the name if it is not already existing. Finally, only one instance of the game can exist in memory at any given time, and this will be checked by using a singleton class.

## [Design Constraints](#_2et92p0)

The constraints that are present are as follows

* Each game will have more than one participating team
* Each team will comprise multiple players
* Each team and game will have unique names
* The game will only have one instance in memory

## [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)

In this model the ProgramDriver uses the singletonTester class to check if there is an active instance. Then the GameService Class interacts with the Game Class which interacts with the Team and Entity Class. The Team class interacts with Player Class and the Entity class. Finally, the Player class interacts with the Entity Class only.

**"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 is a closed platform which means that you will need to pay more to host. Most would say that Mac is the best option for Webhosting because apple provides a fantastic and secure environment for hosting | Linux is an open platform which means that hosting can be cheaper. The downside is that the security is laxer. | Window is a closed platform which means that you will need to pay more to host. If you are not using ASP, .net, or MSSQL then you can have issues with things being choppy. | While you can run servers from mobile devices it would not be recommended. |
| **Client Side** | Mac on the client side tends to be more costly for the hardware that you are getting. This can make the operating system less accessible to many people. The OS is very user friendly overall. | While linux is an open platform and can be easily access by many, it can have a larger learning curve. It is less user friendly overall and has a lot less support. | Windows is one of the most accessible paid Operating systems out there. There is a free version or a paid one if you would like to get more features. It is quite user-friendly and has a lot of support for many different applications. | Creating an application for mobile devices can make for a user friendly environment. The costs to publish to a popular app store can be costly. With the popularity of mobile devices many webapps are being created that are supported by mobile devices. |
| **Development Tools** | SwiftUI and swift is a great language for developing on and for MAC | On linux based systems you can use the C language to develop a web app | On windows I would recommend C++ and visual studio | Developing for IOS is easily done with swift & for android a C language would be best |

## 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 believe that the Windows platform would be the best for The Gaming Room to expand Draw it or Lose it. It has the most compatibility with multiple computing environments.
2. **Operating Systems Architectures**: The chosen architecture would have to be x64 as that is the most widely used architecture in the modern age of computing.
3. **Storage Management**: For storage management either MYSQL or Microsoft Access would be the best choices. For any photos that would need to be saved a JPEG format would be best to save on resources down the line.
4. **Memory Management**: Each process on 64-bit Windows has a virtual address space of 8 terabytes. All threads of a process can access its virtual address space. However, threads cannot access memory that belongs to another process, which protects a process from being corrupted by another process. For Draw it or lose it the program may benefit from using a media library in order to lower resource requirements.
5. **Distributed Systems and Networks**: For this we can use an app such as Xamarin in order to continue compatibility with the visual studio and C# language.
6. **Security**: The Windows operating system and its many features are heavy on security while there are a lot of users and it can make security difficult, Microsoft is constantly updating the Windows security in order to continue allowing people to stay secure when they are interacting with web apps developed for and on windows.