

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 | <9/29/2024> | <Phillip Kimbrel> | Added executive summary  Added Design Constraints  Edited Domain model |

**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 Game rooms need a web orientated game application Draw it or Lose it. This application needs to have the ability to have one or more teams. With that the teams must have the ability to have more than one player. Game and team names must be unique to make sure that the name is not already in use. Finally, only one instance of the game can exist in memory at any time.

## 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)

Some of the design constraints for this project is that it must work on web browser thus making it harder to find the best base to be stable choices to work with are limited to Java HTML5 and Unity Web

## [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 Entity is giving Game Team and Player bits of itself through inheritance as that is how it gets it ability to have ids and names with this GamesService also Polymorphs the stuff from Game Team and Player to form one big class of its own where the game will run off from.

**"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 Servers are only available to those that have a Mac thus making this the most expensive | Linux is the most used due to it being open source, this would be the less expensive as most cloud servers use Linux more than windows | Windows servers are easier for people because it is almost the same as normal windows but with server software and higher end hardware like bigger hard drives | Doing servers on mobile devices is difficult as they are the weakest of the 4 with RAM CPU and Storage not on par with the other servers |
| **Client Side** | MacOS is an easier to use then most but has the least number of users due to the expensiveness of the line of products | Linux is the most affordable as it is free. But besides it being free it also has a massive number of distros each different in their own right but still Linux | Windows is the most common OS used by most people as it is affordable and comes standard in most PCs but being most common means it is the most vulnerable | mobile is not made for multi users but are simple and easy to use as they are something that you will always have on you. |
| **Development Tools** | Macs would be using Xcode and coding with language like SWIFT and Objective-C  The cost for Xcode is $99 | Linux you can develop with almost any language as Linux  And you can use Microsoft visual studio to code for Linux for free | Windows is mostly coded in C and C# and like Linux you develop the code with Microsoft Visual Studio to program stuff for windows for free | Mobile devices depend on what Mobile OS you are using for android you are using java and can develop through Android studio while for IOS you use the same development style as Macs as they are apart of the Shared ecosystem |

## 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**: For the Operating Platform for a server for this web-based game I would recommend Linux as it offers great security and zero licensing cost. Along with that Linux servers can be connected to no matter what OS the Clients are using.
2. **Operating Systems Architectures**: Linux is an open-source OS where it uses extended file system and UnionFS. Also, since it is open source it is constantly getting worked on and improved
3. **Storage Management**: Linux manages it storage the same way a windows computer would with a hard drive just with names like ext and sda. These are the same as windows C: drive and they can use the same system file formats windows does like FAT32 and NTFS.
4. **Memory Management**: Linux manages it memory via virtual and resident memory. Virtual memory is nonexistent memory but it allows programs to reserve memory for the program that has requested it and then resident memory is just the actual memory that you are actually have and are using.
5. **Distributed Systems and Networks**: For this we can use docker and Kubernetes to make and deploy a large number of microservices this would help us expand who all can play and if one goes down it can easily be rerouted to another node of the system
6. **Security**: The server would have a firewall to protect unwanted access from hostile actors. Also, we would have role-based access control to keep normal players from any admin accounts and potentially stealing any information or code and then to cap it all off we would need to encrypt the data going between the servers and clients via symmetric encryption making safe while also not effecting the speed of the games