

<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 | <05/19/22> | Christine Hershey | -Build a web-based game to be available on multiple platforms.  -Game should have unique names, multiple players and more than one team. |

**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 gaming room wants to create a web-based game that is based on their current game, Draw it or Lose it. The current game is only available on android devices, so the new game should perform on all platforms. The client asked that the game can have more than one team involved. The teams should have unique names so that team names can be searched, so there are no duplicates. They ask that there is one instance of the game with unique Identifiers for each instance of the game.

## [Design Constraints](#_2et92p0)

* The game should run on multiple different platforms.
* Unique team names and numerous players on each team have unique identifiers.
* One instance of the game should exist at the same time.

These are what the client has requested and are for the code only. After that, the application development will also have to be developed. In addition, the gaming room has asked that Draw it or Lose it be playable on multiple devices, for example, Windows, Linux, and Apple devices.

## [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 contains the Gaming Room classes. It also shows object-oriented principles.

The classes Game, Team, and Player contain attributes also listed in Entity. This is the principle of inheritance. That assists other classes also in inheriting features because the other classes listed have a relationship with Entity. For example, if I look at the class GameService, it references Game, Game references Team, and Team references Player.

**"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** | Can be easy to change or configure the server with the terminal commands. This can also be inexpensive to impliment, but on the other hand is not as popular as say Linux or Windows.  This runs on a small set of hardware, so it could limit hardware options. | Known for their stability and reliability.  Linux is open-source which would provide a lot of resources.  Does not have as many users that are proficient with Linux, so it would be slightly harder to find someone to work the server. | Windows is most likely the most used platform. Windows offer windows servers, but after doing some research I notice they can be a bit costly to set up. Since windows is the most used operating system, finding users to operate the server would be pretty easy. | Would have an easier time reaching users and have better compatibility.  Mobile devices are very popular  Since mobile devices do not have the power that a PC has, running a full server on it may not be the best option. An advantage to mobile is they are they most cost effective. |
| **Client Side** | Mac is costlier and the user is forced to buy a Mac that is built by Apple.  Someone who has experience on Mac would not need as much time, but someone with less experience would need more time. | This would not cost as much as Mac or Windows. It is open-source. Linux is not commonly used, so a user who is proficient would be needed. Even at that it can be more difficult so the user would need the time to do the work. | Cost would be a lot like a Mac. Windows is commonly used so not as much time would be needed to do the work. Someone who is not as proficient in Windows would need more time. | This would not really cost that much compared to other platforms. As for experience, it should not be that much of a challenge, because mobile devices are easier to work with. Time would be needed for an unexperienced user to learn their way around |
| **Development Tools** | Swift would be the most commonly used application. With swift you can mix in tools like notepad, and Macs can run all languages. | Linux has a command prompt. It works with Visual Studio, Eclipse and notepad. | This would be easier to use than Linux, it can also work with Visual Studio, Eclipse and notepad. | Mac and IOS apps typically are written in Swift. Countless apps can be used for android, Java is the official language for Android. |

## 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**: After my research, I recommend that the Gaming Room goes with the Windows platform. It's the most used operating platform, so it would not be difficult to find the right people with the right knowledge. In addition, there are plenty of IDEs that can be used with windows, and the total cost would be lower.
2. **Operating Systems Architectures**: Windows OS is just easier to use. Applications can utilize windows with a GUI/windows setup. In addition, there are many programming options for users, including command prompt.
3. **Storage Management**: Windows has a storage sense feature built right into the platform. This allows you to manage your files on your hard drive. This will also let you see how much storage your files take up and can delete unnecessary files.

1. **Memory Management**: Windows already has memory management built into it. But we would need to build a database with an abundance of images for the game.
2. **Distributed Systems and Networks**: I believe a client-server distributing system should be used. Since we will have each client application that will depend on a single server application, each client application would be developed to that specific client's strengths. We would also need to have a strong network since the success of this game will rely on multiple clients connecting to a single server to play all together.
3. **Security**: Windows Defender is built into Windows like a security feature for all windows systems. Windows offers Direct Access for work sites. This feature uses authentication and encrypted ESP when the users connect to the network.