

Draw it or Lost It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | <1/30/22> | Kayvon Ghadiri | Implemented simultaneous team play abilities, Administered data to allow specific username titles throughout the player base that will insure on instance of a game |
| 2.0 | <2/6/22> | Kayvon Ghadiri | Edited Evaluation Table and added more content |
| 3.0 | <2/20/22> | Kayvon Ghadiri | Added Recommendations section |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lost It is a software game that is based around 4 rounds in which a guessing game will occur based on the sketch that is shown. Based on the client’s request, greater abilities within the game will be developed and released. Among these requests, the client wants multi team gameplay that allows for unique player id’s. The client also mentioned only allowing one instance of the game.

As we look at the customers’ requests, we need to produce identifiers that will allow us to differentiate instances, games, teams, and players in a more unique way.

## [Design Constraints](#_2et92p0)

* One or more teams are required to participate
* There can be many people on each team
* In order to allow users to determine if a name is being used or not the game name and team names must be unique
* Only one game instance can occur at one given time
* The game must be runnable on several platforms

Listed above are the specifications that the customers have asked us to follow when we write our code for their application. The game needs to also be open to application growth. Gaming room wants this to be able to run on multiple platforms and multiple computers. We must be able to allow Linux users to be able to interact with Android, Windows, iOS, etc. In order for this to be possible we need to rewrite the code in Apples language, Swift, so it can be a part of this application. The code will need to be robust yet simple in order to include multiple platforms.

## [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 class diagram shown below is based on, ‘The Gaming Room’. This diagram illustrates an object oriented approach. The approach shows three different types of classes, “Game”, “Team”, and “Player”. Through these classes, principles of inheritance are shown as they attribute to the “Entity” class listed above them. The Entity class principle helps the other related classes inherit the abilities that the other classes listed have a relationship with. The Game, Team, and Player classes have “is a” relation to the Entity class while Team and Player have a “has a” relation. Another takeaway of this diagram is the encapsulation that Entity will protect the data applied to the program by limiting the access to public methods within the program. This will help protect the player’s private information.

**"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** | In order to configure, gain access, or make changes to the server you can use a plethora of terminal commands.  This is used consistently to web host  It can be upgraded and has many options for web hosting attributes.  Some web hosting services are less common | Similar to Mac, it has a variety of terminal commands for configuring the server, gaining access, and making changes to it. Linux is more cost effective in comparison to Mac.  Secured is the best option of Linux  Linux has the best web hosting services as bugs are dealt with quickly.  Finding software support for web hosting is not easy | Windows has the most software available as many use this as the default.  Windows has a huge advantage when it comes to finding channels. Uses close proximity as well.  Windows has fast load times, is very comfortable for people to use, and has many resource demands.  Windows on its own is not the best against virus’ and can lack when it comes to tech support | Mobile devices prefer to have standalone stationary servers. Mobile devices can have a large variety in the hardware included in their device.  Mobile devices are popular, easily accessible, and portable.  Mobile devices are cheaper and can be more compatible.  Not the best protection and there is a wide range of mobile devices. This means we must decide on which devices can support our game. |
| **Client Side** | Moderate skill and time are needed. The price is most similar to Windows. | High skill and time are needed. Linux is the cheapest. | Lowest skill and time are needed. Price is similar to that of Mac’s. | Most complex skill and time. Clients and developers can view updates anywhere. |
| **Development Tools** | The most common language for Mac is Swift. Other popular universal languages like JavaScript, CSS, Python, HTML can run as well. Eclipse, Visual Studio are the more popular libraries with more available as well. | Easy to use platform with a large variety of languages like Python, CSS, JavaScript, HTML, and more. There are also a plethora of Development tools like nodejs, Visual Studio, Github, Repl.it, command prompt. | Easiest to use platform similar to Linux with a large variety of languages like Java, Python, C++, C#, CSS, JavaScript, HTML, and more. Development tools are also plentiful with Visual Studio, Eclipse, Repl.it, command prompt, and more. | Android is the main language for Android phones and Swift is the main language for iPhones making these two the most common languages. Can also run many languages like HTML, CSS, JavaScript as well. Many popular libraries are also runnable on mobile devices. |

## 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**: The suggested operating system to develop Draw It or Lose It would be Windows. Windows is always the go to choice for developing games as it has the most support with the best structure and is readily available to most users. Windows is relatively inexpensive, provides ease of use, and access to most software packages. The other options were intriguing, but ultimately, Windows was the most well rounded option.
2. **Operating Systems Architectures**: Windows does not have a free operating system, however many laptops come with one included. Although the price, it still has a simple GUI design with many software packages at our disposal. Command Prompt’s power shell utilization is great for configuring our server and Windows also allows for a vast variety of program languages. Resources can be managed through an account or server.
3. **Storage Management**: I would recommend a hybrid model of storing the game itself on the cloud and more secure information such as user account details on local hardware. The reason for this would be that cloud storage would allow an easier experience when updating and maintaining the game. Also, as the client requested the game be available on multiple platforms, cloud storage would be accessible to all of these platforms. Cloud storages biggest downside would be security, and that’s where local hardware storage comes in to secure the safety of our user’s personal information.
4. **Memory Management**: The game has a large number of pictures that need to be stored along with various other items. Memory allocations will offer a simple way to store pictures into separate files other than the default file. By doing this, we can keep the majority of the project in the same file. Although the game has many pictures opening, the end users would still not require that much RAM to run this game.
5. **Distributed Systems and Networks**: As there are many operating systems that all have their own way of doing things, we must find a way to allow for cross platform gaming. Something like Develop 4 can work as it is a proven to be used in the creation of cross platform games. The servers will need to be able to support large player volumes, especially during launch as many gaming companies fail to predict the correct amount of players and have their servers fail at launch. Things like queues can be implemented if necessary to allow the server to not get a large spike in players and instead get a steady stream.
6. **Security**: Windows has an extensive firewall that also only allows the admin user the ability to change or use applications that will change settings of the computer. Windows scans and determines app’s that are on this list. Windows also has an included anti-virus software that although not often used, can be very efficient. Windows allows the use of many 3rd party virus protectors as well if users want to double up on protection.