

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)4

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/15/22 | Ariel Cabrera | Completed Executive Summary, Design Constraints, Domain Model, Evaluations and Recommendations portions. |
| 1.1 | 03/29/2022 | Ariel Cabrera | Revised and expanded the Evaluations sections with more information. |
| 1.2 | 04/12/2022 | Ariel Cabrera | Revised and expanded upon the recommendations section. |

**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 is looking to expand upon their current game, Draw It or Lose It, to be web-based and hosted on multiple platforms, hosted currently on Android only. They are looking to expand to Windows, MacOS, Linux, and mobile devices. They want there to be one only game in memory at a time, with one or more teams in each game, each with unique names.

## [Design Constraints](#_2et92p0)

Web-based:

* Integration – making sure we can have the same experience across all platforms
* Functional requirements – making sure that we have all the required features

Apps:

* Devices sizes – since devices are in an array of sizes, need to ensure that it is still a functional app regardless of size.
* Updates – when we update the web, needing to also be able to time that with app updates.

## [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 program starts in the ProgramDriver folder, as it has the main function, allowing it to run. We then have the Entity class that has methods and attributes that will be inherited by the Game, Team, and Player classes. The Game, Team, and Player classes also have their own distinct methods and attributes. There is also a GameService class that has a relation to the Game Class. There is also a relation case between Game and Team, and Team and Player. All these relations have a zero-to-many option. By using inheritance between the Entity and the Game, Player and Team classes, these classes have less code and make it more readable. As well since they all inherit from the same class, it allows for easier editing if we need to add something to all the classes. The SingletonTester class uses the program driver and ensures out singleton pattern is working as it should be.

**"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 has a Profile Manager that is available on MacOS, Microsoft and Linux, and contains Xsan a storage consolidation service. MacOS servers tend to be one of the few that have all the tools for hosting for Mac products including their phone apps. | Linux already is built into many websites is open source where the company doesn’t pay to use it. Generally described as difficult to learn and no dedicated support, though there are many resources for assistance. Also, since Linux changes versions often, would need to look into one that will have longevity. | Already powers many of the backside of the computers including non-Microsoft products. Also compatible to MacOS. There is a higher cost with using a windows server but there is also more support. However, it is not the most common for hosting web apps. Is better when hosting windows specialty apps. | There is a lot of different server options with Mobile devices. Since there are so many this could be problematic with pursuing one to develop the app with. |
| **Client Side** | Computers with MacOS generally tend to be more expensive than a windows computer but also come with programs to help design and create app for apple products. | Since Linux is open sourced that also means it is free. It also comes with a host of different programming languages to cater to other OSs languages as well. Even though there are many different variants of Linux, many apps can be compatible from one version to another. | Since many of Windows users are on the same OS version, there is less need to be backwards compatible and this streamline the process. There tends to be a higher cost associated with running on windows Servers. | There are a lot of different phones and devices. While apple products tend to have the same OS, many Android devices have differing versions. However, there is already an android version of the game available. |
| **Development Tools** | Has the exclusivity to have XCode to design apps made for apple devices. Thus, a mac will be necessary to create an app for iPhones and iPads. It also can run Visual Studio. | Many different IDEs can be downloaded to and ran on Linux. The Software also comes with some languages ready to go, and there is a host of other apps to help with programming. | Windows has Visual studio as well as Eclipse to help develop apps and frameworks for apps and games. | There are many programs to develops apps for mobile devices. If we were to use C# we could use it on both types otherwise we may need two different programs for each type. |

## 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 would recommend Linux for this project.

**2. Operating Systems Architectures:** I would recommend Linus since it is open-source, keeping costs low, and we well as exponentially versatile and customizable. This means there is also the ability to run different languages for each of the different platforms.

**3. Storage Management:** Linus has a Logical Volume Manager which we can use to assign a certain amount of data for the game. Since we know that starting there are 200 images, we could create a partition for the game for that specific size.

**4. Memory Management:** Linux uses paging and swapping as the main component of its memory management. Swapping will allow us to swap out the images in the game and paging will make it so that there are designated locations for the images.

**5. Distributed Systems and Networks:** We would create sockets in the program to allow communication to Mac or Windows, of the program through the internet.

**6. Security:** Since Linux is open source, there can be many different versions. This leads it to be more difficult for malware and virus to spread since they have to adapt to each version. As well, there are already many different security features already in place. One of those being access rights. Since they are already restricted it helps keep the program more secure. Linux also has to have a login required for each individual.