

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 | <10/17/2021> | Jacob McPherson | Overall completion |

**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)

Draw It or Lose It is trying to develop a web-based game that can run on all operating systems currently in use. Currently the only system supported is the android IOS. A decent solution for this is to make a java window program that can be independently run on any of the desired operating systems.

## [Design Constraints](#_2et92p0)

The design requires that it runs on each operating system. Only one instance of the game can exist at any given time. The game and team names must not repeat if in use. These alone mean that the application will need to have instance checking and iteration for name checking. The application must also allow for multiple teams and players on each team. A list will need to be made to hold player and team names.

## [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 diagram we can see that the Game, Team, and Player classes inherit from the Entity class. Each of these classes uses the class to its right to create a list in its’ own class including the GameService class. The GameService class handles keeping track of IDs and game listing. Finally the ProgramDriver class runs the main program and uses the connected SingletonTester class.

**"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 offers quite a few server hosting methods that are backed with incredible security. That being said the cost of hosting a server is rather expensive even for the most basic server. | Linux has similar hosting servers are mac, but lacks the internal security Mac has. This can be remedied with personalized security using any of the open-source programs available to Linux. | Windows offers a range of server hosting methods, but have been known to have constant downtimes for software updates, which will intern require updates to the application. This could prove costly not only for hosting the server, but for maintenance. | Hosting the server can prove incredibly taxing on the system. It also would be a lot more secure if the hosting system was stationary. Better to choose one of the other options |
| **Client Side** | Mac requires a good bit of expertise in their systems understanding, along with a good bit of time to ensure no issues with other systems. This can prove costly with expensive exclusive software. | Linux offers the lowest cost, but requires the most expertise and time due to all the internal development required on the operating system. | Developing on Windows would cost less with its flexibility, but could still require some expertise in web programming to incorporate it into a browser. | Developing through mobile devices could prove cost and time expensive. Mobile platforms Mobile platforms provide for rapid availability of updates and resources. |
| **Development Tools** | Supports basic languages like python, C++, java, HTML with a selection of useful tools. FlexiHub, homebrew, visual studio, PyCharm. Several free programs are available. | Supports Python, C++, java, JavaScript, visual studio, Sublime Text, and many more. Many programs are free to use, but the development time will be daunting. It would be worth having a development team working on each operating system to make sure it works. | Has a large collection of development tools and IDEs that can be used relatively easily. Some programs require licensing, while there are still plenty that are open-source or available. | Mobile devices do still have the capability to develop using java, python, HTML, and several more. The system also could prove costly for maintenance of any developed web Hoster. |

## 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 suggest the windows operating system for developmental use.
2. **Operating Systems Architectures**: Windows is designed to be rather user friendly with an emphasis on flexibility. Windows allows many different varieties of programs to operate within its’ system. Execution of programs and control over certain properties of these programs is made easily available for the user so that they can run the desired program and many others with little work.
3. **Storage Management**: If the program is being made to be in constant hosting so users can pop in and out at any point through the day then I would recommend hard storage on a desktop or server hard drive that can keep the program running at all times. This would allow the easiest distribution since users would only need a small connection program to reach the main server. If the program is made to be small and quickly hosted for small host sessions then keeping the program in cloud storage and allowing systems to download the program for local use would be a better option. Both choices could have a backup thumb drive that can be kept safe in case of corruption is the application. This backup would hold a core build of the program that can be used for recovery.
4. **Memory Management**: Windows uses a combination of virtual space and a memory pool to allocate memory to necessary programs as the need arises. The memory pool keeps a limiter on the amount of memory available for programs to pull while still keeping a good amount available for various programs to pull from if the need arises.
5. **Distributed Systems and Networks**: If the different operating platforms each had the application already pulled up or installed then ideally all that would be needed is a stable internet connection. The program would need to be recognized on each systems firewall as a safe application so that information could be sent over the network to each of the connecting users. The application itself should not ask for any information outside of the game data and user id for the game. If the program is hosted by server then the client should only have outages if the server goes through maintenance or if their own system has issues. If the users are hosting the server than the hosting system going out would cause all connected systems to be booted from the connection.
6. **Security**: Every operating system has some manner of security implemented to keep user information safe. Mac OS has some astounding security that even the military uses for their systems. Windows has many different security system and anti-malware programs available for users to use. One of the easiest to use is Windows Defender which is already built into every windows system. This program scans your system routinely to make sure no processes are changing or doing anything that isn’t dictated by the normal windows programming.