

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 | 05/21/2021 | Afahri Kerr | Initial Review |

**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 develop a web-based game that serves multiple platforms based on their current game “Draw It or Lose It”. It is currently available on the Android app only.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The web-based game should have the same user experience as the android version. We want it to be playable on multiple platforms so we should make it as least technically demanding as possible. The server size could limit the amount of players that can use the site at the same time.

## [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)

There are 7 different classes. The Game, Team, and Player classes are inherited from the Entity class. Those three classes also have a 0 – many relationship with each other. The players make up a team and a game can have different teams in it. The game service class can have more than one game. The program driver uses a singleton design so there should only be one instance.

**"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** | Some Advantages of the Mac OS are that it is upgradable and has a simple GUI. There are various options for different web hosting requirements. Flexible terminal commands to configure the server, access, or make changes. Mac has one of the best anti-malware programs to protect against spyware, adware, and worms. Apple makes using a Mac as a server very easy. Some disadvantages are that the cost is very high due to only having to buy Mac products with higher licensing costs and web hosting services. | Similar to Mac Flexible terminal commands Very secured. Linux offers a lot of what Mac does but is very cost-effective. Linux is also less prone to cyber threats. Due to the open-source format that makes Linux fully customizable and generally the preferred choice for web hosting services. Some disadvantages are that Linux is not user-friendly and is far less compatible than windows for gaming. It is harder to find applications and drivers that support web hosting needs. There are far fewer Linux pre-built machines. | Some advantages are that the OS is straightforward to use there is an extensive library of software that offers extensive support. Some disadvantages are susceptibility to getting viruses due to the constant updates and poor technical support. Windows can also be expensive to start up due to the upfront cost for licenses but still less than the price of apple. | Some advantages are that mobile phones have a large user base, are user-friendly, and are less prone to viruses such as Malware. Some disadvantages are that the hardware is not upgradable, and if a virus happens, there is no support. It would be better if the server was immobile and could be tracked in a single place. |
| **Client Side** | Moderate expertise and time are required. Integration between other Apple devices is straightforward. The updates are significantly less and automatic. It is great at multitasking, able to run macOS, Windows, and Linux apps side by side. The cost of setup is expensive, and you could be charged monthly for some software. Often there are years between new hardware updates. | Maximum expertise and time are required. Due to the lack of applications available. Linux is the hardest to set up. However, it could be the most secure behind apple. Linux's open-source format also makes it the most cost-effective. However, because it is free and the lack of debugging, there are more bugs and no tech support. | Minimum expertise and time required. There is a vast selection of window-based PCs in all price ranges. Many applications share compatibility that helps streamline the process and makes it the best for gaming. Furthermore, the cost is less than or the same as mac. There are forced updates that can take time though. Inconsistent functionality and quality among off-brand makers and Malware, spyware, and ransomware could be an issue | Very cost-effective and user-friendly. It takes less time to load a page and provides flexibility. However, designing and maintaining applications for phones or other mobile devices could be very costly, depending on the device and the frequency of the updates. Both android and apple use their own languages, so development can take longer and require more expertise. |
| **Development Tools** | Mac can use all the normal programming languages. It is better suited to do much more from the terminal compared to windows. IDE's can be for Java, Python, PHP, and ruby. Development tools within Mac OS can be pycharm, eclipse visual studios notepad++, and online tools. | Linux can work with all the Similar IDE used in the Mac OS, but all the software is unsupported. | Windows is easier to use than Linux because of its GUI. It is possible to run both Windows and Linux at the same time though. There are many tools and resources designed for windows. | You could use apps such as swift that to create the software. Since the programming language is Java, it wouldn’t be too hard to port over to other platforms |

## 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 recommend using Windows. The game is already on Android and Windows has many tools to help developers design the program.
2. **Operating Systems Architectures**: Windows uses a Layered design consisting of a user mode and kernel mode.
3. **Storage Management**: Storage can be handled through a cloud storage solution to keep track of all the players and their wins and losses.
4. **Memory Management**: Windows has a physical address and a virtual address space. Threads cannot access memory that belongs to another process.
5. **Distributed Systems and Networks**: We would have to use a cross-platform development tool to cover multiple platforms. We also should have a dedicated gaming server to handle the traffic loads and keep costs low.
6. **Security**: Windows has many security features built in and other security programs can be purchased. On the server side, it is the developers' job to ensure that all the account information is kept secure. Users should create an account and password to log into the game server.