

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 03/26/24 | Carter W | Document release. |
| 1.1 | 4/11/14 | Carter W | Evaluation added.  Recommendations added. |

**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 come out with a web-based game called Draw It or Lose It. As of now, it is only available to play on Android, but they’re looking to expand to Apple, Windows, and more. The game is kind of like “Win, Lose, or Draw”; It’s played in four rounds, while teams take turns guessing what pictures are being sketched.

## Requirements

* This soon-to-be web-based game must be available on platforms outside of Android, including: IOS, Windows, Linux, etc.
* Must have ability to have one or more teams that can play.
* Each team should have assigned players.
* Game should tell players when they have chosen a username that is already in play.
* Must have games that are unique, in a sense that the same game cannot exist in the same memory profile.

## [Design Constraints](#_2et92p0)

* The game is going to be web-based, meaning it can be played on most of any platform that has web access. This includes creating apps for cellular devices and other CPUs.
* We must develop interfaces for other platforms.
* There must be a database or server that records all teams and usernames/players across all platforms simultaneously. Each platform should be able to cross-over.
* The games’ existences must be understood from a cross-platform sense. This means that the same game cannot exist on different servers on different 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)

In short, this UML depicts what the design of the game looks like in flow. To explain, this is the application of the game and how the classes fit into each of their specific roles. The *ProgramDriver* class will use the *SingletonTester* class to run a test of the program (This is why an arrow is pointing to this class; It derives from it). While the *GameService* class holds most of the code to function the game. The classes connected to this (such as the *Game* or *Team* class) ensures that each player/team/game is unique (this is also why there’s a “0…\*” that links them). All three of these connecting classes derive from the *Entity* class that is shown below.

**"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** | Apple has application servers, established in Unix OS. They offer apps to purchase in their stores and can provide good business with user interface. It’s easy to operate and navigate, and offers helpful coding tools, but can get expensive quickly. Their servers require licensing and equipment is costly. But it is a good method to start building games and applications from. Of course, roaming is open to anyone, it does get pricey and can be a choice to those who are loyal to the brand. | Linux is a cheaper method than Apple In terms of server expansion and security. Their servers are cheap, but their interface is not as inviting. It's not too favorited apart from the other choices, because it sometimes cannot support certain applications people are fond of. It does have good security and proper hosting abilities; This route would be an easier platform to open a server on, I believe. | Windows is a very iconic platform. It has a friendly user interface and is great for navigation and lacks large service costs. This system is widely liked and very popular and is supported with strong hardware. Security is something of concern compared to Linux, because of its popularity. Something that would make this an easier option is its availability in supporting many different web-based applications. | Mobile devices aren’t the best for server applications due to memory storage and sizing. They’re extremely easy to navigate and provide a well-rounded interface experience. It would be another cheap option. The data sizing would be an issue, but this could be solved by third-party vendors that are able to provide database needs. If this isn’t provided, it would be difficult to implement. |
| **Client Side** | Apple has a very user-friendly GUI. It’s easy for people to comprehend and get used to as it's customizability offers a lot of help to those who are new to web surfing and CPU interaction. But game development is not ideal for this application due to its cost (for this specific application currently). It would require high server licensing and maintenance fees for the audience size. | This would be a pretty cheap option. Providing our client with a low number for application release would be of high chance. One thing to consider would be time in creating different client options. It requires specialty treatment due to its complicated end-user effect. The Linux shell makes for slightly more difficult navigation but can be favoring to get used to for some. | Windows will also require licensing, but it’s widely used. The number of people that operate on this platform is significant. But it is not open-sourced, and will get costly, much like our Apple selection. I think this would be a good place for development, but I think it is too pricey for this project’s application. | This would depend on the device that is desired. Because they have Android, IOS would be the next big mobile device to release on. The cost of this would be a bit more, just due to the restrictions in data storage. Its development would be tricky because of the different devices. It’s very easy to interact with once it’s complete, but this would take more time and concentration than the rest. |
| **Development Tools** | Swift (or converting C++ [or another language] from another platform), Eclipse | Java or PHP (there are some more options). Eclipse | C, C++, Java, Python, etc. (many options). Eclipse, PyCharm, etc. MicrosoftVS | Java, Swift (there are other options). Eclipse, Database (third-party, i.e. Oracle) |

## 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 say that Linux is the most suitable option. It is an open platform and friendly in coding applications, and IDEs (If we were looking to expand further with extended cost, I would say Windows).
2. **Operating Systems Architectures**: Linux works on a central operating system for its activity, called *Kernel*. It has surrounding system libraries to build around its centrality. The *Shell* executes the external functionality of the system operations when browsing and operating. While its applications are open to the operator from an interface standpoint.

(2023, September 25). *Architecture of Linux Operating System*. <https://www.geeksforgeeks.org/architecture-of-linux-operating-system/>

1. **Storage Management**: This game should run on pCloud, as this provides a great live-speed interface, and is very common in Linux systems. Because we’re using cloud-based storage, it’s easy to manipulate and flexible in ways which we can expand when needed.
2. **Memory Management**: PHP may be a good option as this is well-liked on this operating system, but Java is a larger programming software, and (in my experience) easier to use. Java should be the application used for memory management.
3. **Distributed Systems and Networks**: Google can be accessed by most of any computing device. Because certain systems have their own search engines – Windows has Edge, Apple has Safari, etc. – it isn’t ideal to go with a Linux-specified search engine. This way, all other platforms can access the cross-functionality platforms when they’re ready for release.
4. **Security**: Nikto is an ideal security application for Linux. Application data and system management would be well protected in Linux with its onion-layered system architecture. But Nitko will ensure that our security is solid in scanning for antiviruses, attempting hackers, and other wall-breakers when building our application and maintaining its defined properties.