

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 | <03/22/2024> | Shawna Freire | Wrote a summary to introduce the software problem.  Identified the design constraints for the application. |

**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 wishes to make a game that they have available on Android into a web-based game. The problem is that they do not know where to start when it comes to making a web-based game. They want the web-based game to reflect what the Android game can do.

## Requirements

* The game must be able to have one or more teams
* Each team will have multiple players
* Game and Team names must be unique
  + This is to check if a name is already in use when choosing a team name
* There can only be one instance of a game at any given time.
  + This means each game must have unique names for the game, team, or player

## [Design Constraints](#_2et92p0)

One thing to consider is that the original game was made to run only on Androids. This means that The Gaming Room will have to learn or use an entirely new OS for this project. This could offer some restraints based on what they already have experience with.

The game will have to use Singleton so that we can make sure there are no repeating 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)

The Program Driver is what will run the whole thing. It calls upon Singleton Tester to make sure that Game Service has only one instance. It can only have one instance to ensure that all of the game and team names are unique.

The classes Game, Team, and Player all inherit from Entity. This means that the methods used in Entity can be used from Game, Team, and Player. They all must inherit things from Entity because they all need the information that Entity has. This keeps us from having to get repeat information from the players.

Game Service is the class that holds the list of games and can have only one instance.

Game Service is associated with Game. Game Service will have zero or more instances of Game.

Game is associated with Team. Game will have zero or more instances of Team.

Team is associated with Player. Team will have zero or more instances of Player.

These four classes need to be this way to ensure that everything is connected correctly. If it was done another way the teams may not be with the proper game 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** | UNIX-based system. Mac has a lot of creative tools built into it. They have less graphic memory compared to other systems. | Linux is an open-source System. It also has a lot of customization options for users. It is also able to handle high workloads. There is a steep learning curve when it comes to using the interface. | It is supported by almost everything. It is known to be great for programming. If there is a problem with the OS, it has poor technical support. | Mobile devices are meant to be on the go. It can be more difficult to get a web page to look right. There is also the problem that if it were to be opened on a computer, the user could tell it was meant to be viewed on a phone. |
| **Client Side** | Mac tend to be more expensive. There are also fewer people who use Macs. Mac has a simpler interface design. This would make it easier for the client to use, but there might be fewer examples of things online when it comes to Mac. | Linux as an OS is free to use. Unfortunately, there is a steep learning curve, and there isn’t going to be a lot of readily available software and hardware. There is also a problem with standardization within the OS. | You have to pay for the Windows OS. A lot of applications that can be used on Windows also need to be purchased. Most people have Windows, which means that the game would be available to a lot of people. | The team at The Gaming Room has made a phone application before. So they are familiar with the layout and know that a page would need to look correct on a phone. However, there are a lot of different mobile phone OSes. |
| **Development Tools** | Apple has its own programming language known as Swift. Swift has many built-in functions to help with coding. Python can also be used on MacOS. | Linux uses C and C++. JavaScript and Python are other popular options used. Swift, the language MacOS uses, also works on Linux. However, there will be fewer tools on Linux. | Programming languages used for Windows are mostly C, C ++, and C#. Windows can also run Python programs.  Many tools are available on Windows, though it might not be free. | Many languages work on mobile devices. C, Python, JavaScript, and even Swift. Although most use C, C++, or C#. |

## 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 believe Windows would be the best to use. This is because most people have it, and it has a lot of software that is compatible with it. It can also run almost all programming languages. This means that if the client is more familiar with a language, Windows will most likely be able to run it. This will help with getting the product done faster, as they will not be forced to learn a new language. There are also many tools that the programmers will be able to use. There will also be many available resources for them to find out how something works if they are unsure because so many people use Windows.
2. **Operating Systems Architectures**: The Windows architecture is not very detailed to users. There are two sides to it: the user-mode and the kernel-mode. The process involves an executable program, a private virtual address space, system resources accessible by threads, unique identifiers, a thread of execution, and a security context. The threads can hold different information. This information is the unique identifier, the private storage area, two stacks with instructions (one for the user and one for the kernel), and something that represents the processor's state.
3. **Storage Management**: The Windows Storage Management provider is recommended for Windows platforms. It is designed for use by developers and is included in Windows OS. It also has an API, which can be used with programming languages like C and C++.
4. **Memory Management**: Windows comes with a virtual address space. If it is a 32-bit, it has 4 gigabytes of memory; if it is a 64-bit, it has 8 terabytes. These virtual address spaces cannot be accessed by other processes unless they are shared. This means that the data for Draw It or Lose It could only be accessed by itself or by things connected to it.
5. **Distributed Systems and Networks**: Windows has a Remote Procedure Call (RPC). It works with C and C++. This distribution style does not have dedicated clients and servers but instead uses a request-reply format. One part will make a request, the client, and then there will be a response, the server. In this distribution system, both sides can get information from each other. This makes communication between various platforms easier. RPC also comes with security packages, making it easier for developers to keep the information safe.
6. **Security**: There can be security within the program itself, such as making things password-protected, encrypting data, and firewalls, but Windows also has a firewall. This firewall can help enforce end-to-end network communications. On the hardware side, they also have encrypted hard drives. These hard drives will self-encrypt, which allows for greater security.