

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 | 03/24/2024 | Emily Staats | Filled out all required sections |
| 1.1 | 4/7/2024 | Emily Staats | Added additional info and revised completed sections |
| 1.2 | 4/21/2024 | Emily Staats | Revised and added additional information |

**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 can serve multiple platforms based off their android only app, Draw It or Lose It. The staff at The Gaming Room does not know how to set up the program, nor how to make it accessible across multiple platforms. The game must support teams, and names must be unique to allow users to check whether a name is already being used when choosing a name. The solution in this case would be to implement the singleton design pattern, which will allow only one instance of the game to run at any time, in addition to being able to assure that there are no duplicate teams.

## 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)

Some design constraints involve the environment itself and how it operates. For example, given that the application currently operates as an Android app only, we must consider scalability and deployment. We must balance the number of users and servers to ensure the program runs smoothly and avoid the game crashing. Though multi-user platform is a great implementation within an application, it will be difficult to implement in the case we want so many unique identifiers. In addition to this, the run time may increase as the game grows, so the setup of the players and teams can’t overload the system, which can cause a potential crash.

## [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)

**Encapsulation**

Encapsulation within the program is important due to the protection of data, in addition to the use of that data, and the state of each instance during runtime. In this case, we are protecting the instance itself and treating it as an individual so we can efficiently run a program.

**Abstraction**

We can justify the abstraction of certain aspects of the program, for example the singleton tester to assure that the functionality of the program is correct, in addition to the usage of the encapsulated object within the method of using the singleton pattern. This abstraction is what allows the program to function efficiently, and fluidly.

**Inheritance**

The usage of inheritance is more of a baseline of logic that is foundational for understanding the actuality of presence within the game when the user connects, and then is put into a specific team given the allocation of the identifier from the singleton class.

**Polymorphism**

Lastly, we also use polymorphism. In one way, we can say that the different teams show a specific type of polymorphism. The differences between the objective teams based on the individuality of the identifiers is important to consider. We need to be able to distinguish that although we have these teams, they have different identifiers, and therefore they are this object, stored in this place, which we will assign these players too. The transformation of this object in runtime is important to the program as it is what allows the program to run efficiently.

**"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** | **+**Good use of CLI **+**Easy Administration  **-**Not great for large scale distribution  **-**Not open source | **+**Open Source **+**Easy to use  **+**Can customize specifically for development  **-**Can be difficult for non-Linux users  **-**Takes time to learn how to use it | **+**Most stable  **+**Easy server  development and  administration  **-**Not open source  **-**May be difficult  to setup  environments if  configuration not perfect | **+**Mobile Web Server  **+**Can develop for  both websites and  phones  **-**Not always connected to the  internet  **-**Not the best  for handling large  amounts of traffic |
| **Client Side** | **+**Great for Mac users  **+**Easy to develop on Mac  **-**Mac’s can be expensive  **-**not known for gaming  **-**Knowledge is necessary | **+**Linux is free  **+**Linux is easy for  people who know  how to use it  **+**Linux is used in  some form in  everything  **-**Least used OS | **+**Dominates the  gaming market  **+**Most used  Operating System  **+**Great Graphics  **-**User issues and  configuration  problems are  difficult to fix  **-**Not as secure as  Mac or Linux | **+**Different web  browser  **+**Don’t have the  amount of  configuration as a computer  **-**Dependent on  brand of device  **-**Different devices  react differently |
| **Development Tools** | **+**Great for Java,  website design,  and Python  **+**Swift and Xcode  are specifically for Mac  **-**C++ is difficult due to compiler issues  **-**May have to  translate languages  into Mac language | **+**Large variety of  development  tools  **+**Can pretty much  run anything | **+**Great for app  development  **+**Microsoft offers  a great number of  tools and tutorials  **-**Not specifically  made for usability  when developing  Visual Studio  Eclipse  C++, Java, Python,  Website  development | **+**Apple has new M1  chips  **+**Specific Library for  applications that  is easy and secure  **+**Offers in-house  tools for growing  your application  **+**Android has  Xamarin and Kotlin |

## 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:** Windows is one of the best development platforms. The number of tools in addition to differing virtual environments and tutorials for applications is unbeatable. Windows, because of the many tutorials offered, makes it possible for the developer of a web application to use any of the tools at their disposal whether they are familiar with them or not.
2. **Operating Systems Architectures:** Windows allows users to develop and set up the environment specifically the way they want it with an interactive GUI that is easy to use. In addition to having great technical support, there are many resources online to help solve any issues that a user may have. Again, we can also use the OS to split up computer resources easily for running dual operating systems for testing applications on a different OS.
3. **Storage Management:** There are many different options, some more popular options in the present day are Dropbox and OneDrive. You can also use coded databases, in addition to SQL.
4. **Memory Management:** Windows allows you to specify the amount of memory that you use within a designated virtual instance during runtime. Allocation of different types of memory on your computer is exponential when developing applications.
5. **Distributed Systems and Networks:** Cloud technology lets us run servers with company databases so users can call the application on whatever platform. The ability to organize the type of requests and respond with different instances of the game with multiple types of virtual instances is extremely useful.
6. **Security:** Security will always be an issue for whatever type of platform we use. Using Windows, we can create personalized databases, implement firewalls, and use encryption. The ability for the company to be able to adapt and respond to any type of cyber-attacks is very important, and Windows offers this in addition to tech support that can help as well.