

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 | 09/18/23 | Matthew Blake | Wrote Summary, Design Constraints, and Domain Model. |
| 2.0 | 10/7/2023 | Matthew Blake | Filled out Evaluation table. |
| 3.0 | 10/19/2023 | Matthew Blake | Filled out recommendations. |

**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 has requested that we assist in developing their game, Draw It of Lose It. The first software design problem that we plan on addressing will be the need to have only one instance of the game team or player in memory at a time. I plan on solving this problem by using a singleton pattern of development in order to instantiate and ensure that the requirement is met. The singleton pattern of development is just a fancy way of programming classes and variables so that only one instance of that variable or class can exist in memory at any given moment.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

One design constraint that I anticipate being a problem is allowing for an extremely large number of people to be using the website all at the same time. This means that the instances for the game, teams, and players need to essentially be unlimited, while also being unique to that instance of the game, team, and player. This means that the code must be designed in a very specific way, which may also be tedious.

## [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)

This UML diagram has seven classes; ProgramDriver, SingletonTester, Entity, Game, Player, Team, and GameService. The Game, Team, and Player classes all have an “is a” relationship with the Entity superclass. A class having an “is a” relationship with another class means that it inherits from the other class, usually the superclass. The Team and Player classes have a “has a” relationship with each other, which is also called an aggregation. This means that, in this diagram, the Team class has references to multiple instances of the Player class. And this means the same for how the Game class relates to Team, and GameService relates to Game. You can literally translate this to mean that GameService has multiple Games, Game has multiple Teams, and Teams have multiple Players. And all of these classes relate back to the Entity superclass by inheriting their attributes from it in different ways. ProgramDriver and the SingletonTester classes are where the rest of the actual code will get executed, which is why they are separate in the diagram. The main Object Oriented Programming practice used so far is inheritance. GameService, Game, Team, and Player, all inherit multiple attributes and operations from the Entity superclass.

**"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** | **Characteristics:**  Adaptive and intuitive terminal commands that can be used to access, configure, or make changes to the server. Incredibly popular for web-hosting.  **Advantages:** Easily upgradable, and has multiple options available to customize per web hosting requirements.  **Disadvantages:**  Relatively inflexible when developing for other platforms. | **Characteristics:**  Arguably more adaptive than Mac, but less intuitive command console. Overall higher learning curve to use, but the most efficient once you learn it.  **Advantages:**  Linux is the most secure of the four platforms which makes it the most popular for web hosting.  **Disadvantages:**  Fairly difficult to actually design apps for, and the user base is very limited. | **Characteristics:**  Constantly in a battle for dominance versus Mac. Adaptable and user friendly. Easy to develop for, and on.  **Advantages:**  Most adaptable. Most amount of resources for developing any type of application. Easiest to actually write code for itself, and other operating systems.  **Disadvantages:**  Least secure, and most vulnerable to exploitation. | **Characteristics:**  Largest user base of the four platforms, but too mobile to actually use for primary web hosting services.  **Advantages:**  Highly portable.  **Disadvantages:**  Less secure. Wider variety of hardware constraints. Dependent on cellular towers or Wifi. |
| **Client Side** | Although the cost is comparable to Windows (in terms of development), the expertise required is more than Windows but less than Linux. The time required to develop is also comparable to Windows. | The cost is basically non-existent as Linux is open-source. The time requirement is much more than it is for Windows or Mac because of the initial learning curve, which also means that you need a lot of expertise in Linux to succeed. | Minimum expertise, similar cost and time investment to Windows. Overall the easiest to use from the client side. | Client side applications are the most flexible on mobile, and can be updated at any time, from anywhere. |
| **Development Tools** | Although Swift is the most popular language for Macs, it is compatible with HTML, CSS, Javascript, Java, Python, PHP, and Ruby. | Linux can use all the usual IDE’s including Eclipse and Visual Studio. It also supports all the same languages that Windows and Mac can also use. | Windows can make use of any IDE you want/need, and supports development in any language you want or need. | You can use almost any language to create apps in Android, but iPhones require you to use Apple’s proprietary IDE to compile code because it won’t compile properly without it. |

## 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**: For this specific game, I recommend using Windows Server for the server operating platform, and JavaScript for the language. Using Windows Server and JavaScript will allow for the most flexibility when attempting to make the game cross-platform.
2. **Operating Systems Architectures**: I recommend using Windows for the Operating System to build the game on. It is the most flexible and allows you to program the game in any language, if needed, and almost any IDE. It also offers a large knowledge base that can be utilized for collaboration or any issues that you may encounter.
3. **Storage Management**: I feel that the best storage management system for Windows Server should be a Content Delivery Network. Although this type of storage system largely depends on how close or far away from the physical location of the server you are, because of the size of Draw It or Lose It, I don’t think that you will have much latency loss unless you are in a different part of the world when playing.
4. **Memory Management**: The primary benefit of using Windows Server for the server operating platform is the built in tools used to monitor how much memory the servers are using at any given moment, and if there are any problems, there is an extensive knowledge base to pull from for troubleshooting solutions. Windows Server also uses various memory optimization, balancing, and resource management techniques that make using this software extremely efficient.
5. **Distributed Systems and Networks**: With this type of system, you will be able to manage the servers using Windows Server, and the main data will be stored on regular servers with server blades. The clients will request the data they need directly from the server, and the server will use various caching techniques to ensure the lowest latency possible when data is requested.
6. **Security**: Windows Server is very frequently updated to ensure that the data on your server is very secure. It also comes with User Access Control which allows certain users with certain permissions to access certain data. It also has Role-Based access control which allows you to give users certain roles in the server settings that only allows users to access the data of the role that they are assigned. Windows Server also shares the same antivirus and anti-malware that the actual Windows OS has access to as well. So it is very secure.