

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/23/2023 | Alexander Cummings | added Executive Summary, Design Constraints, and Domain Model, Evaluation, 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](#_heading=h.35nkun2)

Our client, The Gaming Room, is asking to develop a web-based, interactive, multiplayer version of their current game, Draw It or Lose It. The Gaming Room is looking to advance their current into the ever-unfolding world of online gaming. Their game, Draw It or Lose It, already has all the elements in place for a game that connects different players, whether those players are in the same location or not. The Gaming Room is asking for the basics of online gaming, unique teams, with multiple players on each team and multiple teams per game. They are also asking for a single-player mode.

## [Design Constraints](#_heading=h.1ksv4uv)

* The application must exist in an online environment. This would require JavaScript or possibly Python in order to achieve compatibility amongst different platforms
* Single-Player Mode: consisting of one single user against a pre-programmed opponent of varying difficulties.
* Multi-Player Mode: consisting of games that include two or more (upper limit not specified) teams that consist of two or more players (again upper limit not specified).
* Each Game must also only consist of unique Teams and Players, this means each Game, Team and Player needs a unique Tag.
* This can be done by utilizing a singleton pattern when creating new Games.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The base class, Entity, is a superclass. The classes Game, Team and Player are all sub-classes to Entity, this way those classes will inherit all the attributes from Entity and do not have to be re-written in each individual class. The classes that are associated with each are: Game, Team, Player and GameService. This association is defined as ‘zero-to-many’, so it does not limit how much information can be passed between these. ProgramDriver is responsible for running this entire package. It first checks itself with the SingleTon Tester class, this is what ensures unique Games. ProgramDriver inherits Singleton Tester.

**"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](#_heading=h.z337ya)

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** | Advantages:  -lots of good options for web hosting solutions  -customizable  Weaknesses:  - less preferred than Windows | Advantages:  -by far the most secure  -sort security issues very early, so they don’t cause further issues  Weaknesses:  -less resources and therefore less web hosting solutions available | Advantages:  -large amount of resources available  -User confidence/knowledge of Windows is high  Weaknesses:  -the least secure platform, more prone to viruses | Advantages:  -very popular  -very cheap  -reach lots of people more easily  -easier to access anywhere  Weaknesses:  -limited performance/speed by user’s phone  -security is sub-par  -limited on amount/types of devices that are compatible as there are a plethora of different smart-phones |
| **Client Side** | Requires specialized software that needs a high level of expertise to develop properly.  Less common than Windows.  Long loading/dev time.  Software is expensive. | Requires an even higher level of expertise than Mac. Much less popular than the other options, so finding a properly trained staff might be difficult.  Software is also expensive, but loading times are on the shorter side. | The most common platform. Large amount of resources/support available. Short loading times. High amount of resources available leads to higher cost. | Very common.  Lowest cost and loading times of all options. Limited performance available due to limitations of user’s phone. |
| **Development Tools** | -PHP  -JavaScript | -PHP | -Java  -HTML  -CSS | Android Studio |

## 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 can recommend Linux as the operating platform. While it is not as easily operated and not as familiar to people as Windows or MacOS, it is more robust in other qualities. The price would be much less, it can provide a more unique environment than the other options, and there exists a virulent community to provide advice and resources.
2. **Operating Systems Architectures**: Linux would include both the operating system and system architecture. Linux can be developed to favor whatever attribute you desire, whether that be user connectivity or speed or graphics. The other main advantage of using Linux is that it is an open-source operating system. This means its underlying programming is open to view by anyone, this results in the ability to create a much more unique environment in which Draw It or Lose It can operate.
3. **Storage Management**: Linux provides a hierarchical structure to its file system. This is a very sturdy and reliable way of managing storage. It can support everything the other operating systems can, the customizable nature of Linux provides the advantage over other storage management systems.
4. **Memory Management**: Memory management is one of the areas in which Linux is head and shoulders above the competition. Linux utilizes a method of memory management that just loads the pages that are demanded by the executing process, this is a very efficient way of operating virtual memory. It is known as Demand Paging.
5. **Distributed Systems and Networks**: Linux supports all of the network protocols necessary for communication between various systems. The main advantage of using Linux in this area is its stability across multiple servers. Linux can distribute network traffic across multiple servers. By using this method, it ensures the application will continue functioning even in the event of a server failure.
6. **Security**: The reason for choosing Linux in terms of security is because of its built-in security features. On top of Linux’s password policies and encryption standards it offers several valuable tools that will help protect The Gaming Room and its users. One of these features is AppArmor, a Linux kernel security module that uses path-based access control to restrict actions installed programs can take. This is one of several security measures that Linux offers that its competitors do not.