

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 16Jul2022 | Frangerwy brito feliz | Executive Summary and Design. |

**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's employees would like us to create a web-based version of their game Draw It or Lose It. We are to create a game that can have several teams, various players per team, only allow unique team names to allow players to check whether a name is already in use, and only allow one instance of the game to exist in memory at a time.

## [Design Constraints](#_2et92p0)

## -To be completely effective in any web-based environment, the game must be built in a web-based language.

## -Team and player objects must be implemented to support multiple teams with many players per team.

## -Use a singleton technique in the game and team creation to ensure the uniqueness of the game and 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)

Entity creates a connection between the Game, Team, and Player classes. This indicates they all get information from Entity. We can demonstrate this using inheritance in UML. As a result, each class will have common references such as "name" and "id." Making Entity a 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** | Although Mac is more expensive than Linux, it boasts superior performance and stability. user-friendly graphical user interface. Scaling is not ideal, and doing so necessitates thorough understanding of the system. | more challenging to use and requiring a terminal. Linux works with the majority of web hosting software and hardware, but it is incompatible with Windows applications, which might be problematic if the business has to use a Windows application on the server. | Issues in the past, including system crashes, glitches, and blue screens.  higher price. user-friendly graphical user interface.  Fast and practical FTP tools.  greater choice in hardware and software, but more frequent updates and downtime. | To write and run servers from mobile Android smartphones, there are numerous programs. low price.  Easy to complete. The best way to do this is through cloud hosting, although you can use a device. |
| **Client Side** | simpler, more uniform, and more streamlined across all Mac devices. A single ecosystem enables seamless fusion of iOS and Mac devices. | open source and free. Without experience, difficult to operate. convenient to install on any OS system, works. compatible with all web browsers | Workflow is more complex and not as easily synchronized with other platforms and devices.  simple, intuitive graphical user interface. | smaller-sized screen.  Portability is very user-friendly.  It is challenging to implement numerous items in a single window. limited offline capabilities |
| **Development Tools** | We can use languages to quickly run the most popular options on Macs. which we can include in excellent tools, like notepad + +. being aware that Macs can run all available languages. | both open source and free. operation is challenging without prior experience. simple to install Suitable for all operating systems. using any web browser. | Workflow is more complex and not as easily synchronized with other platforms and devices.  simple, intuitive graphical user interface. more options for hardware, software, and application customization.  even more games. | smaller-sized screen. Portability is very user-friendly.  It is challenging to implement numerous items in a single window. limited offline capabilities.  Personalization. |

## 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 Gaming Room's "Draw It or Lose It" application's expansion and optimum results, Windows operating system is suggested. Windows server OS enhances computer reliability.
2. **Operating Systems Architectures**: Windows services are used by Windows-based applications and allow them to create a Graphical User Interface (GUI) while using system resources. These apps also include graphics and multimedia, communications, and web services.
3. **Storage Management**: The Windows server enables the routine movement of data items to solid-state storage to free up storage for user demands.
4. **Memory Management**: You will need to establish a database or library with a large number of images when developing this game. The RAM allocation makes it possible to save photographs outside of the normal picture folder.
5. **Distributed Systems and Networks**: The use of networking support in distributed systems is an excellent method of implementing and utilizing software for those systems. The distributed system and network functionality allow for simple communication between numerous individual workstations and varied CPUs.
6. **Security**: Windows includes built-in security protection software. Nevertheless, using other resources to secure user data and information is recommended.