

<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 | 05/24/25 | William-Wootton | In this version of the game Draw it or Lose it we allow for unique people to join the game and group them by unique teams. |

**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 company has tasked the team with creating a web-based game based off their app *Draw It or Lose It.* The Gaming Room has tasked the team with setting up the environment for the main game. The team will create a singleton system that allows for each player, team and game to have unique names.

## Requirements

*The game must have the ability to have any number of teams involved inside each game. Each team will be allowed to contain any number of players. Each game, team, and player names must be unique across all teams and all games. Only one single instance of each game will be available in memory, following the singleton pattern that is to be adhered to by the team and player classes respectively.*

## [Design Constraints](#_2et92p0)

Within a web-based system the game allows for different network delays that can be felt, and client-side servers may be different depending on where someone is playing the game. If a player on the west coast wishes to play with a friend on the east coast and they are playing on servers based in CA then the east coast player may see delays in inputs compared to the player on the west coast.

## [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 entity class is an inaccessible superclass for the Game, Team, and Player classes allowing them to access necessary information without crowding their own code. Game Service allows for multiple Game classes to be used at any time, Game allows for 0 to many teams to be within each individual Game, and Team allows for 0 to many players to be inside each Team at any time.

**"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** | Mac uses a UNIX based operating system which is said to be great for web development. However, it can only run on apple hardware causing it to have licensing restrictions | Linux is the most popular OS for web-based servers because it is highly customizable, it's open-source and free it's easy to automate and supported by major cloud providers. However, it is harder to learn for beginners and can require more setup. | .NET support automatically, it has a familiar interface and has PowerShell scripting. However, it has high licensing costs, requires more resources and can be less flexible than the other options. | The main disadvantage to mobile devices is that they are not designed to handle long term hosting. They also have limited processing power, storage and memory, and have no support for server frameworks. However they do work for local networking and it is possible to run lightweight servers on one. |
| **Client Side** | The cost to develop client side servers with mac is high since you need to use apple hardware for testing / development. It takes a decent amount of time because safaris browser engine called WebKit may have different behavior compared to other browsers. | Since Linux is free and open-source there is very low to 0 cost. However, it may take some time since there must be compatibility over the OSs on top of Linux kernel. And programmers need to understand Command-line tools, and environment variations. | Cost is moderate since you need to get things like Windows licenses and software like Visual Studio that can have costs of their own. It has some of the most well-known browsers like Chrome and Firefox. Programmers need to understand thing .NET support. | Mobile devices require a high cost since you need to test on multiple devices and Apple’s developer program to use with IOS testing costs $100 per year. Each mobile device has a different screen size so all must be tested, as well has the fact that there are many different input methods like touch and swipe to be evaluated in code. Programmers must know mobile web issues, and understand how to handle inputs from a touch screen. |
| **Development Tools** | Mac like the other OSs support HTML frontend languages, as well as Python, and Java for backend. However it needs to be tested for any WebKit issues. | Linux also uses HTML frontend development and supports Python and Java for backend. Linux is the most common for web-development backend development and is often preferred for web development | HTML is supported for front end and Python, .Net, and C# are used in backend. Coding in windows is good for Window-s apps and has good Chrome and edge support. | HTML for front end and can use Javascript and C# for cross platform. |

## 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**: Linux is the most popular OS for web-development for a reason and has many important pluses for web development. Its low cost, ease of use and accessibility make it the best choice for Draw It or Lose It.
2. **Operating Systems Architectures**: Linux offers a multi-tier style of architecture which has many advantages over single or two-tier architecture. These include but are not limited to, modularity allowing you to improve and change code in different tiers that can improve and be improved upon without having to rewrite tons and tons of code. When making changes to one tier you are less likely to mess with other tiers as well. And finally, it provides greater security for the program.
3. **Storage Management**: Linux again is the most common OS for things like database servers and uses one of the best database management systems called PostgreSQL which just like Linux is open-source and free.
4. **Memory Management**: Linux utilizes a few tools that help with memory management, a few of which are the following. Paging which breaks up the memory into “pages” and swaps between active and inactive pages to disk storage to free up RAM. Memory Caching and shared memory which stores frequented data into memory, and allows multiple processes to access memory in the same region.
5. **Distributed Systems and Networks**: Linux again is considered to be the best OS for distributed systems and networks because of your ability to control the system and the control you have over process management.
6. **Security:** Linux has an encryption system called dm-crypt that helps encrypt up to entire disks, is also uses software for encrypting files and backups. It also has encryption for in transit data and uses software such as OpenSSL, and LibreSSL.