

<**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 | <03/30/2025> | <Roberto Hernandez> | <Brief description of changes in this revision> |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

Draw It or Lose It has teams guess a puzzle based on the displayed drawings. The teams compete in multiple timed rounds. There must only be one existing game instance at any given time. The game name and team names must be unique. We can use the singleton pattern to ensure only one instance of the game is used and an iterator pattern to ensure unique names.

## 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.>*

*Business requirements:*

*Develop a web-based version of Draw It or Lose It app. Streamline development.*

*Technical requirements:*

*The game must only allow for one instance of the game to exist at a given time. The game must ensure that game and team names are unique. The game must be able to run in a web-based environment and allow multiple player connections.*

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

1. Because the game must function in a web-based, distributed environment it must be built using web-based technologies and require compatibility with servers and different browsers.
2. Because the game needs to support users connecting at the same time from different locations the game needs to have efficient communication between the server and multiple players. The game must also handle the different connections to maintain consistency between each player.
3. Since the game must only allow for one instance and the game and team names must be unique, the use of the singleton and iterator pattern are necessary in development.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

The UML diagram shows an abstract Entity class. The Game, Team, and Player classes inherit from the Entity class. The Game class can contain 0 to many Team objects and the Team class can contain 0 to many Player objects. The UML diagram also contains the ProgramDriver class which is used to control the execution of the program. ProgramDriver uses SingletonTester to ensure that the singleton pattern functions correctly. GameService manages the creation of games, teams and players.

**"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** | macOS server is not typically used for production hosting. Best used for local or staging servers. | Great support for hosting with Apache, Nginx,Node.js. Scalable with platforms like AWS, GCP, and Azure. Linux is used often for web servers. | Windows supports web apps with IIS or Azure. Windows is commonly used but can be costly. | Mobile devices are not used to host servers. |
| **Client Side** | App needs to work well with Safari or Chrome. Macs have strong hardware, but the app still needs to run efficiently in the browser without using too many system resources, especially when played on older models or laptops running on battery power. | Linux users are often more technical, they expect fast performance and minimal bugs. Users typically use Firefox or Chrome, so the app should be tested on both. | Windows supports all modern browsers. Interface must be responsive and work on different screen sizes. | Mobile clients typically use Safari and Chrome. It should be responsive for small screens and touch controls. |
| **Development Tools** | Languages include JavaScript, or Swift. IDEs can include Xcode, VS code. | Languages can include JavaScript, Python, PHP, Java. For IDEs: VS code, Eclipse. | Languages can include: C#, JavaScript. IDEs: VS code. Great for .NET integrations. | Languages: JavaScript, Java, Swift. IDEs: Android Studio, Xcode, VS code. |

## 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 best for hosting Draw It or Lose It due to its reliability, performance, security and scalability. Linux servers are flexible and are compatible with many hosting services like AWS, Google Cloud, and Azure. Another advantage is that Linux is open source, which makes it more cost effective as there aren’t any expensive licensing fees like with other operating platforms.
2. **Operating Systems Architectures**: Linux uses modular open-source architecture that allows developers to have a lot of control over how the system is set up and managed. This flexibility is useful for hosting web-based applications. Linux can run using only the essential services, which helps keep the game fast and responsive.
3. **Storage Management**: Cloud-based storage is recommended for this platform. Some options are Google Cloud or AWS. Cloud storage makes it easy to store and access game and player information from anywhere. Cloud platforms also have built in security features which help keep user data safe.
4. **Memory Management**: Linux uses techniques like virtual memory and swap space. These can be helpful for Draw It or Lose It because the game should stay responsive while handling multiple players. Linux also allows developers to adjust memory to improve performance.
5. **Distributed Systems and Networks**: Draw It or Lose It could use REST APIs for real-time gameplay communication. Because the game components will be connected over the internet if one part of the system goes down, others can still work.
6. **Security**: The game should use HTTPS so data is encrypted. Login systems should use secure authentication methods, and there should be systems to check for suspicious activity.