

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 | 07/16/2023 | Angelo Mangalindan | Initial Draft |
| 2.0 | 07/30/2023 | Angelo Mangalindan | Evaluation |
| 3.0 | 8/11/2023 | Angelo Mangalindan | Recommendation |

**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)

CTS has taken another client, The Gaming Room. They want us to develop a web-based version of their android game Draw It or Lose It. The game should allow multiple teams and members with unique team names. Only one instance of the game can exist in memory at any given time.

To maintain a single game instance, a singleton design pattern is used. Similarly, the iterator pattern is employed to effectively manage teams and team members and avoid conflicts.

## Requirements

The Gaming Room asked us to develop a web-based game based on their android game. The game needs to allow multiple teams and team members with only one instance of the game exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The design constraint for this project is to ensure that the web-based version of "Draw It or Lose It" maintains the same gameplay mechanics, features, and overall experience as the original android-based game. In our development approach will use Java as the programming language for this project.

## [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 design of this model is singleton design pattern it means that only one game should exist in one memory location. The ProgramDriver class holds the main class that uses the SingletonTest to check if the game exists in only one memory location.

The Entity class is the parent class of Game, Team, and Player class. These classes are all inherited from Entity class meaning that they also have all the variables and functions of the Entity class. Those classes are also related to each other and can have zero to many instances of it. In the Team class there is a List<Players> this is a list of instances of Players that can be zero to many. This concept is same to Game that has a List<Team>.

The GameService class is responsible for making sure there's only one game running at a time. It does this by preventing the creation of multiple instances through a private constructor. The only way to create a GameService object is by using the getInstance() function, which checks if an instance already exists and creates a new one only if it's not already in memory.

**"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 servers run in Apple macOS it offers user-friendly interface. It has a built-in software tool making it easier to use for server hosting. The weakness is it has limited hardware options. | Linux is a highly stable and reliable operating system. It is known for its strong security, making it a good server hosting to protect data and transactions. However, it can be difficult to learn the user interface of Linux. | Windows offers a user-friendly interface and robust hardware support, making it accessible to a diverse user base and ensuring compatibility with various devices. The disadvantage is it has a higher licensing cost. | The advantages of hosting a web browser in mobile devices is they are portable, allowing users to access it anywhere and everywhere. The disadvantages are it has limited resources and compatibility challenges. |
| **Client Side** | The client needs to be familiar with macOS development tools to ensure optimal performance. They also need to ensure the compatibility of different types of macOS version. | Linux is an open-source operating system making it cost-efficient. Developers should be well-versed in Linux development, and depending on the clients, they may also need expertise in web technologies, mobile app development, or other specialized areas. | Windows offers different APIs, such as desktop applications (Win32 API), Universal Windows Platform (UWP) for modern Windows apps, and .NET for web applications. Ensuring compatibility across these APIs and platforms is essential to host a server. | The clients need expertise in different mobile platforms (Android, IOS, etc) to ensure optimal performance and user experience on each platform. The cost and time will vary depending on the difficulty of the web hosting server. |
| **Development Tools** | The development tools for Mac are Xcode for IDE and Swift for programming language. | The programming language development tools for Linux are python, java etc. The IDE are PyCharm, Visual Studio etc. | The programming languages for Windows are C#, C++, JavaScript etc. The IDE are PyCharm, Visual Studio etc. | The development tools in mobile devices depend on what operating system is used. For IOS there are Xcode and Swift for android there are java and eclipse. |

## 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**: The best operating platform to use for “Draw it or Lose it” game is Windows. It has a friendly user interface, versatility, and widespread adoption. Having a friendly user interface will make it easier for the clients and users to use the application. Most gamers use windows compared to other operating platforms because of its versatility.
2. **Operating Systems Architectures**: Kernel is the best choice because it can manipulate the hardware CPU, main memory, and Random-Access Memory (RAM), enabling efficient allocation and utilization of these critical components. This efficiency translates to optimized performance, particularly vital for resource-intensive applications like games. It serves as a middleman between software and hardware, it offers a standardized interface, streamlining communication and allowing software to interact with diverse hardware components.
3. **Storage Management**: Asset Bundling is the best storage management for game applications because it stores files and assets into a package to reduce redundancy. By doing so, asset bundling enhances loading times and overall performance, ensuring that only the necessary resources are fetched for a given level. The packages will load based on the stages of the application. For example, the game is in level one the package that will be loaded is the package that is programmed to output in level one.
4. **Memory Management**: The recommended memory management for “Draw it or Lose it” is memory caching. It uses Random-Access Memory (RAM) to allocate data that is frequently used as a temporary storage to reduce loading time. RAM and operating systems can work together and not put a lot of pressure on the main memory. This makes the flow easier and shorter compared to the operating system and main memory working together.
5. **Distributed Systems and Networks**: To enable communication across different platforms for 'Draw It or Lose It,' a distributed software architecture is used. This involves breaking down the application into modules that run on different devices, connected through a network. Dependence on network connectivity is essential, ensuring real-time data exchange. To manage potential disruptions like network outages, redundancy strategies and offline capabilities are valuable. Overall, a well-structured distribution system and network enable seamless cross-platform communication for the game.
6. **Security**: To protect user information they must have an account that includes a personal email and password. This will prevent other users accessing their account. To make the security stronger there should be an authentication method via email or phone that will send a code for the user to input when accessing their account. There should also be a requirement in the password like having at least one capital letter, number, and special character. This will make the password stronger and harder to guess.