

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## 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 | 03/19/21 | Krista Mosser | Initial design specifications |
| 2.0 | 04/03/21 | Krista Mosser | Edited Executive Summary, Design Constraints, and Domain Model  Added Evaluation and Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room would like to develop a web-based game based on their current game, Draw It or Lose It. The newly developed game will serve multiple platforms, while the current game is only available as an Android app. One instance of a game may exist at any given time. A game consists of four rounds of play lasting one minute each. Each game will have one or more teams. Game names and team names must be unique.

## [Design Constraints](#_2et92p0)

The Gaming Room will need to provide a website to host the game and its libraries.

App must be able to run in various browsers.

Needs authentication of unique names for games, teams, and players.

Security for players logging in to system.

## [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 is the parent class for Game, Team, and Player. Each Entity has an ID and a Name and methods for creating entities, getting the ID, getting the name, and sending the information to a string.

GameService is a singleton class that ensures that there is only one instance of the Game in memory. It also verifies that the game name and ID are unique by holding the next ID to be used to create a new game. It also holds the next IDs to be used when creating Games, Players, and Teams. This ensures that Games, Players, and Teams are unique. These IDs are also private to the GameService class, which means that no other class can manipulate them. They can only access them through their getters.

The Player class is a child of the Entity class and inherits from that class. It consists of a method to create a Player based on a unique ID which is provided by the GameService class, and a string of the user's choice. It also uses polymorphism to override the sending of information to a string.

The Team class is a child of the Entity class and inherits from that class. It also contains a list of Players. The methods include the creation of a team based on a unique ID which is provided by the GameService class, as well as a string of the user's choice. It also uses polymorphism to override the information sent to a string. The class also contains the method to add Players to a list after checking that player name is unique.

Game class is a child of Entity and inherits from that class. The Game class also includes a list of teams. It includes a method for declaring a Game using the GameService singleton, a method to add teams to a game, and an override of the writing of information to a string. The class also adds teams to a list after checking that team name is unique.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Regularly releases updates to enhance security.  Can run Windows and Linux.  Integrates well with mobile versions. | Strong firewall.  Virtually virus and malware-free.  Makes more efficient use of resources like CPU and memory. | Malware, spyware, and ransomware more prevalent than MAC and Linux | Keeps service stable.  Would need to make game compatible across devices |
| **Client Side** | Software is expensive and can only be used with a Macintosh PC.  Closed source code. | Costs no money.  Open source code.  Issues with compatibility with widely-used software | Less expensive than Mac, but still expensive and resource-intensive.  Closed source code.  Most popular and widely used system. | More difficult to implement.  Android app won't work on iOS and vice versa, so double the work to implement.  Changes easier to implement without down time. |
| **Development Tools** | Python  C++  Java  Objective-C  Swift  PERL  JavaScript  Visual Studio  Atom  Sublime Text  Komodo Edit  Brackets  Coda  BBEdit  Netbeans | Java  Python  C++  Shell  JavaScript  Visual Studio  Atom  Brackets  CudaText  VS Codium  BBEdit  Netbeans | Python  JavaScript  Java  C++  C#  Assembly language  Visual Studio  IntelliJ IDEA  Aptana Studio 3  PyCharm  PhpStorm  WebStorm  Netbeans  Eclipse  RubyMine  Komodo | JavaScript  Kotlin  C++  C#  Python  Swift (iPad and iPhone)  Android Studio  Xcode  Visual Studio  IntelliJ  Xamarin |

## 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**: It is my recommendation that The Gaming Room’s application for Draw It or Lose It should run on Window OS. Windows is the most common OS in use, and the one that the most developers have expertise in using. It is more expensive than Linux, but less expensive than Mac OS. Since Mac OS can only be used by a Macintosh PC, and Linux has issues with compatibility with widely used software, Windows is the best choice for overall ability to market the app and also to maintain it once delivered.
2. **Operating Systems Architectures**: Windows OS is a layered design consisting of two components, user mode and kernel mode. In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Kernel mode is generally most trusted functions of the operating system. User mode programs and subsystems in user mode are limited in terms of to what system resources they have access. In user mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory. Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable. The Hardware Abstraction Layer (HAL) is a layer between the physical hardware of the computer and the rest of the operating system. It was designed to hide differences in hardware and provide a consistent platform on which the kernel is run. The HAL includes hardware-specific code that controls I/O interfaces, interrupt controller and multiple processors.
3. **Storage Management**: The Windows OS uses Disk Management to perform advance storage tasks. With this, a user may partition a hard drive, allocate more storage to a drive, shrink a drive, or even combine drives. This allows the user to customize how the storage is set up in order to ensure that the application always has enough storage for its libraries.
4. **Memory Management**: The Draw It or Lose It app will contain a large amount of image files and also databases of users, games, and teams. Windows OS has both virtual memory and physical memory, and on a 64-bit architecture, the memory is theoretically 16 exabytes (or 1 million terabytes), though for practical reasons, today it only supports 16 TB, split equally between user and system space. This provides the needed space for the app's files.
5. **Distributed Systems and Networks**: Because the users of Draw It or Lose It will be working in teams, each most likely using their own computer or device, distributed systems will need to be employed. In distributed systems, a network of computers works together to solve pending tasks. Despite being physically separated, these autonomous computers work together closely in a process where the work is divvied up.  The participating computers will use the internet to communicate with the client server, as well as with each other in order to start and play games. Also, if one computer (player) gets disconnected for whatever reason, the rest of the team can continue to play without interruption. Distributed systems are commonly used in online gaming.
6. **Security**: Windows provides its own security with Windows Defender, which is regularly updated by Microsoft. However, additional security software can be implemented via third-party software, such as Norton, McAfee, Bitdefender, and Kaspersky. User information on the client's platform will be protected by only being available to certain level users, and by the requirement of user names and passwords to access the server. Windows also uses Address Space Layout Randomization (ASLR) that makes all memory locations no longer fixed and predictable, but chosen randomly. Consequently, the stack, the heap and the loaded modules will have different memory addresses each time a process is started, making it less vulnerable to attack, as it is harder for attackers to find usable memory locations, such as the beginning of a certain function or the effective location of malicious code or data.