

# 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 | 09/24/24 | Samit Datta | Updated the executive summary, requirements and design constraints |
| 1.1 | 10/04/24 | Samit Datta | Updated the server side, client and development side |
| 1.2 | 10/14/24 | Samit Datta | Updated the recommendations |

**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 introduced their customers with a new game called Draw It or Lose it. The application follows very similar to the 1980s television game; Win, Lose or Draw. Using render images from a large library of stocks as clues, the players must guess the puzzle (a phrase, title or thing) within 30 seconds before given to the other remaining teams a chance to guess the puzzle within 15 seconds. The Gaming Room approached consultant in hopes of developing for the software to present to their audiences.

## Requirements

* Client requests that the application supports multiple teams and players
* Client requests that we create a system that gives unique identifications for games, team and players
* Client requests that each round is time-based with dynamic image rendering
* Client requests that the mechanism for guessing and answers are met with strict time limits

## [Design Constraints](#_2et92p0)

A technical constraint would be that the application would have handle a scenario where there is a massive number of teams and players without performance, connections and data management issues

A technical constraint would be that the application would be used in multiple OS systems, so functionality must work between different OS system interactions without issues

A technical constraint would be that due to the potential massive customer base using the application at once, we must integrate secured user data and gameplay sessions to prevent unauthorized access and improve customer interactions.

## [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)

As you can see in the UML Diagram, we are introduced with Encapsulation as three subclasses (GameService, Game, Team) and a super class (Entity) has private attributes and public methods. This is followed by Inheritance, where Game, Team, and Player (subclasses) inherits the instance variables, methods and protected members from Entity (super class). Polymorphism plays a role in the diagram as we can see that toString(), which comes from Entity and inherited by Game, Team, and Player, could be overridden in each class to provide specific representations. Abstraction is shown, particularly in the GameService class where we see methods such as addGame(), getGame(), and addPlayer() as it involves the complexity of managing games, teams and players but users would only know how to interact with the options but not know how these options work in a technical setting. The Gaming Room UML Diagram has represents all four object-oriented programming.

**"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 must 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** | Supports graphical interface, powerful terminals commands, and the servers are often used by developers  Mac costs more than other server options, both in hardware and software. macOS isn’t commonly used for server development as it lacks specialized took that larger server environment often uses. The lack of popularity also creates less community support and few resources for troubleshooting resources | Nearly every part of the Linux server is configurable, making it flexible. It is open source, free and known for being fast, stable and lightweight.  Linux is not user-friendly, especially for beginners. Not every software or hardware is supported in Linux as it may have in other systems. It is also not as intuitive as other systems, especially for users who are more used to graphical interfaces. | User friendly and server is very manageable. It also has a powerful command-line tool called PowerShell that handles server tasks.  Due to the paid licensing to run a server, the costs can add up, especially for larger setups or companies. Windows are more prone to experience crashes and require more hardware resources than others. | Ideally for remote server management but can act as a small-scale server in specific situations.  Mobile devices aren’t designed to be full-fledged servers yet. The lack of power and capability to handle heavy server workloads eliminate them as a candidate to serious server usages. Mobile devices are also more vulnerable to security threats and the networks aren’t fast or reliable to handle heavy data or complex tasks. |
| **Client Side** | Great for creativity and it being user friendly but are expensive with software and gaming options | Highly customizable and free but requires an expertise in Linux as it harder to learn with software compatibility issues | Very outgoing with amazing software support, especially for gaming and business but are prone to security and bloating the system resources. | iOS is very secure and smooth but unfortunately every expensive and restrictive while Android is customizable and affordable but very inconsistent in performance and security |
| **Development Tools** | macOS uses Swift, Objective-C, C/C++, Python and JavaScript, with IDEs and Tools such as Xcode, Cocoa, Swift UI and Homebrew | Linux uses C/C++, Python, Java, Bash, JavaScript (Node.js), with IDES and Tools such as VS Code, PyCharm, Eclipse and Vim/Emacs. | Windows uses C#, C/C ++, VB.NET, JavaScript (Node.js) and Python, with IDES and Tools such as VS Code, PowerShell, and .NET Framework/.Net Core | For iOS system, Swift and Objective-C are the languages being used, with IDES and Tools such as Xcode, Swift UI, UIKit, CocoaPods, and TestFlight  For Android system, Kotlin, Java and C++ are the languages being used, with IDES and Tools such as Android Studio, Gradle, Android SDK and Firebase. |

## 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**: Upon a good amount of research, I would recommend Window, as it is one of the most popular operating system that is well-known for supporting a large range of development frameworks and tools and being compatible with different hardware configurations. Since Windows is popular, it means it gets regular updates, has lots of community support, and plenty of resources available. This makes it a good choice for *Draw It or Lose It* to reach a big audience since so many people use Windows. Plus, Windows is great at running both new updates and older apps, so it’s compatible with a wide range of software.
2. **Operating Systems Architectures**: Windows have Window NT Architectures which supports advance features like preemptive multitasking, security and hardware abstraction. Windows also separates processes into User Mode and Kernel Mode. User mode limit access to critical system resources, which helps with security and stability while Kernel mode has full access to things such as hardware or system functions. Windows also supports both 32-bit and 64-bit versions, but 64-bit is usually recommended because it has more processing power and can handle larger amounts of memory. For *Draw It or Lose It*, using 64-bit would be a good idea since it can handle better graphics, smoother performance, and makes it easier to add multiplayer or real-time features if needed.
3. **Storage Management**: The New Technology File System (NTFS) is the best file system for Windows because it’s secure, stable, and can handle large files. NTFS is great for safely storing game data, player profiles, and game history since it has strong controls for those who can access files, built-in encryption, and good disk space management. It’s also reliable and helps keep data safe, even if there’s a crash or power outage.  
   Using cloud storage, like Microsoft Azure, which works well with Windows, can also be part of a good storage system. While NTFS takes care of local storage, Azure provides secure, flexible cloud storage that makes it easy to sync data across different platforms and devices.
4. **Memory Management**: Window uses a hybrid memory management model, combining paging and segmentation to efficiently manage memory allocation. Windows would use virtual memory to augment RAM when necessary for an application such as Draw It or Lose It, allocating memory resources for user interactions, game assets, and real-time updates. With caching mechanisms, it helps ensure that the game remains responsive even as the demand increases (such as when more players join the game)
5. **Distributed Systems and Networks**: Windows can use tools like Windows Communication Foundation (WCF) or RESTful APIs to help different platforms (like Windows, Android, and iOS) communicate with each other. With these tools, all versions of the *Draw It or Lose It* game on different devices can connect by sharing data and processing power through a cloud system, such as Microsoft Azure or AWS. This setup lets the game send messages between players, keep game progress in sync, and manage multiplayer sessions in real-time.
6. **Security**: Despite talking about Window constant updates that can interrupt workflow and very time consuming, it’s helpful as it makes sure Window is updated with the improvised security features. Windows also has several built-in security tools, like BitLocker, Windows Defender, and built-in encryption, which can help keep users safe in *Draw It or Lose It*. Using encryption, secure network protocols like HTTPS, and Windows' access controls, the game can protect user data when sending information between platforms. To make data transfer even safer, things like Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols can be used. Microsoft Azure’s Active Directory (AD) could help manage user identities, and adding multi-factor authentication (MFA) would make accounts even more secure. Regular system updates and anti-malware software are also important to keep out viruses and prevent unauthorized access to the game.