

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 3.0 | 10/12/2023 | Reese Thurman | Version 3 |

**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’ newest client, The Gaming Room, is requesting the development of a web-based application that emulates the 1980s television gameshow, *Win, Lose, or Draw*. The application will be based on CTS’ current game, *Draw It or Lose It*, which is only available as an Android app. The new application, however, will be designed for compatibility on multiple platforms, with each instance of the game having the ability to host one or more unique teams. This feature will be accomplished by requiring users to check whether a team name is currently in use when selecting their team’s name. Finally, each game will be unique, which will be achieved by creating unique identifiers for each game instance.

## Requirements

The client’s software requirements include the following: (1) Each game can host one ore more teams; (2) each team can have multiple players; (3) games and team names must be unique, with only one game instance existing in memory at any given time; and (4) cross-computability with multiple web-based platforms.

## [Design Constraints](#_2et92p0)

Given the software requirements, there are several design constraints that should be considered. These design constraints include cross-platform compatibility, resource management, bandwidth limitations, network latency, security, and regulatory compliance.

The client requires the application to be available on multiple platforms, which implies that the web-based game will need to be compatible with various web browsers and devices, such as desktops, mobile phones, and tablets. This will require game testing on multiple browsers to ensure consistent functionality and cross-browser compatibility. Additionally, the application must incorporate user interface adaptability to ensure the game is functional on different device platforms and screen sizes.

Regarding resource management, web browsers may have limitations on memory usage, which will require the application to optimize memory management, given that poor resource management can affect game performance. Additionally, since network latency and bandwidth limitation can affect the performance of web-based games, the application should be designed to compensate for network delays and offer quality settings to accommodate users with varying degrees of internet bandwidth.

Web-based games can be susceptible to security threats, such as cheating or data breaches. Accordingly, the application should implement security measures, such as data encryption to protect user data. Finally, given that the application may be used by minors, it is important to ensure that the software adheres to all legal requirements related to age and content restrictions.

## [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 UML class diagram depicts seven rectangles representing classes, with four of the classes organized into an distinct hierarchy. The class at the top of the hierarchy is Entity, with the Game, Team, and Player classes organized below Entity. This hierarchy depicts the inheritance relationship between the classes using directional arrows, in which Entity is a parent class of Game, Team, and Player. Additionally, the diagram of the Entity, Game, Team, and Player classes includes descriptions of the variables, constructors, and methods contained within the classes.

The UML diagram also depicts the GameService class, which has an association with the Game class. In turn, the Game class is associated with Team, and the Team class is associated with Player. The relationship between these classes is further depicted to indicate that an instances of the class on one end of the relationship can be associated with zero or more instances of the class on the other end of the relationship. Finally, the diagram includes a depiction of the ProgramDriver class, which includes the main method. The ProgramDriver class uses the SingletonTester class, which contains the testSingleton method.

**"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** | The game can be hosted on a MacOS server. MacOS is relatively user friendly for developers in comparison to Linux. MacOS also integrates well with other related platforms (iPhones and iPads). However, the cost of Apple hardware and software is expensive, which may limit scalability of the game depending on the client’s budget. | The game can also be hosted on a Linux server. Linux is an open-source OS that supports programming languages such as Python and PHP. Accordingly, Linux is the most cost-effective platform because the OS is generally free. However, Linux is more difficult to learn in comparison to MacOS or Windows. | The game can be easily hosted on a Windows server. Windows is a widely used, well supported OS that can run Microsoft technologies. There are many hardware options available at reasonable costs. However, the licensing costs for the OS and other Microsoft software products are relatively high. | Web-based applications can be accessed on mobile devices, including iPhones and Androids. However, mobile devices (iOS and Android) are not generally used as web servers. Therefore, hosting the game on a mobile device server is not recommended. |
| **Client Side** | Overall development considerations for Mac include extensive testing, expertise in web development, and ongoing maintenance to resolve cross-platform compatibility issues. The game should be tested on MacOS browsers (Safari) to resolve any browser specific issues. UX design and performance will likely vary between platforms, but the code should be optimized for MacOS. Additionally, using cross-platform frameworks can reduce cost and the time required to support multiple platforms. | Overall development considerations for Linux include extensive testing, expertise in web development, and ongoing maintenance to resolve cross-platform compatibility issues. The game should be tested on Linux compatible browsers (Firefox) to resolve any browser specific issues. UX design and performance will likely vary between platforms, but the code should be optimized for Linux. Additionally, using cross-platform frameworks can reduce cost and the time required to support multiple platforms. | Overall development considerations for Windows include extensive testing, expertise in web development, and ongoing maintenance to resolve cross-platform compatibility issues. The game should be tested on Windows browsers (Edge) to resolve any browser specific issues. UX design and performance will likely vary between platforms, but the code should be optimized for Windows. Additionally, using cross-platform frameworks can reduce cost and the time required to support multiple platforms. | Overall development considerations for mobile devices include extensive testing, expertise in mobile development, and ongoing maintenance to resolve cross-platform compatibility issues. The game should be tested on mobile browsers (e.g., safari) to resolve any browser specific issues. Mobile devices also have less resources compared to desktop platforms, so optimization for mobile devices is required, along with touch screen support. The software must also adhere to App Store and Google Play Store requirements. Additionally, using cross-platform frameworks can reduce cost and time required to support multiple platforms. |
| **Development Tools** | A combination of programming languages and IDEs is necessary for a multi-platform game. For MacOS, these programming and markup languages include Ruby, Javascript, HTML, and CSS. Useful IDEs include Visual Studio and GitHub for managing code repositories. Accordingly, development teams will require expertise in the various programming languages, with separate teams for front end and back end development, as well as separate teams for mobile app development. Finally, there will be additional costs for IDEs and web hosting services. | A combination of programming languages and IDEs is necessary for a multi-platform game. For Linux, these programming and markup languages include Javascript, Python, PHP, HTML, and CSS. Useful IDEs include Visual Studio and PyCharm, and GitHub for managing code repositories. Accordingly, development teams will require expertise in the various programming languages, with separate teams for front end and back end development, as well as separate teams for mobile app development. Finally, there will be additional costs for IDEs and web hosting services. | A combination of programming languages and IDEs is necessary for a multi-platform game. For Windows, these programming and markup languages include Javascript, Python, C#, PHP, HTML, and CSS. Useful IDEs include Visual Studio and PyCharm, and GitHub for managing code repositories. Accordingly, development teams will require expertise in the various programming languages, which separate teams for front end and back end development, as well as separate teams for mobile app development. Finally, there will be additional costs for IDEs and web hosting services. | A combination of programming languages and IDEs is necessary for a multi-platform game. For mobile, these programming languages include Java and Swift. Useful IDEs include Xcode and Android Studio, and GitHub for managing code repositories. Accordingly, development teams will require expertise in the various programming languages, with separate teams for front end and back end development, as well as separate teams for mobile app development. Finally, there will be additional costs for IDEs and web hosting services. |

## 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 recommended operating platform for Draw It or Lose it is Windows Server. Windows Server can run a wide range of applications, including web-based games, that can be built using various programming languages. Windows Server also offers access to powerful Microsoft technologies, including IDEs (Visual Studio), libraries, APIs, and frameworks. Most importantly, Microsoft’s .NET Framework offers important tools for developing cross-platform compatible web applications, such as ASP.NET. These technologies will enable The Gaming Room to expand Draw It or Lose it to other computing environments.
2. **Operating Systems Architectures**: Windows Server is an operating system designed to host web applications, manage networks, and complete server-side tasks. Windows Server is comprised of multiple components; these key systems include user mode, kernel mode, executive services, security, and the hardware abstraction layer. User mode primarily runs applications, whereas kernel mode enables access to system resources and hardware. The executive services system is responsible for managing I/O operations, memory, objects, and processes. Next, the security system handles user authentication and permissions to control user access. Lastly, the hardware abstraction layer acts as the intermediary between the computer hardware and the operating system, which enables the operating system to communicate and interact with the hardware.
3. **Storage Management**: The storage estimate is an important factor of storage management. Here, the storage estimate is based on the total size of the game data, including user data, images, and audio files. The storage estimate also accounts for future growth, given that the game will expand its player base across multiple platforms. Considering the foregoing, the recommended storage management approach is a combination of database optimization, user data management, and cloud technology. The developers will analyze and improve game-related database queries and identify critical user data in order to remove unnecessary game data from storage. With regard to cloud technology, Microsoft Azure cloud storage will be used to maximize cost effective, long-term storage. Azure provides both secure and scalable storage for game data. These approaches work together to ensure efficient storage management.
4. **Memory Management**: Windows Server uses several important memory management techniques that would be useful for Draw It or Lose It. These techniques include tools for monitoring memory usage and diagnosing memory issues, virtual memory management, and resource prioritization. Virtual memory will allow the game to access secondary memory as if it was main memory, thereby increasing the amount of memory available to the game. Windows Server also allows resource prioritization, in which memory allocation priorities can be set for the game. Additionally, techniques such as object pooling, caching, image compression, and garbage collection can improve the overall performance of the game by optimizing memory management. Object pooling is a design pattern that allows objects to be reused from a pool rather than created and destroyed, thus improving program performance. Caching allows images to be stored on the client side and image compression reduces the size of image files without sacrificing image quality, both of which optimize memory management. Lastly, garbage collection is a process that clears memory of objects no longer in use, thereby conserving memory resources. Although garbage collection is largely automated in Java, knowledge of garbage collection is important for identifying and resolving memory leaks in software.
5. **Distributed Systems and Networks**: Draw It or Lose It can accomplish cross-platform communication using a client-server architecture, APIs, standard HTTP protocols, data serialization (JSON), and cross-platform game engines. Under the client-server architecture, the Windows Server will act as the central point for game communication for clients from each platform. HTTP protocols, data serialization, and APIs will be used to standardized communication over the network. Finally, a cross-platform game engine, such as Unity, will be used to develop Draw It or Lose It on each platform simultaneously. Network challenges can be addressed through latency compensation, bandwidth optimization, and network monitoring. Network latency and bandwidth limitation can affect the performance of web-based games. Accordingly, the application will compensate for network delays using techniques such as buffering and offer quality settings to accommodate users with varying degrees of internet bandwidth. Additionally, network failover, redundancy, and network monitoring will be implemented to detect, troubleshoot, and quickly resolve network issues gracefully with minimal interruption to the game.
6. **Security**: Web-based games can be susceptible to security threats, such as cheating or data breaches. Accordingly, the application should implement security measures across all platforms, including secure APIs in addition to general data encryption, user authorization, and authentication. Furthermore, data minimalization will be enforced, in which only the minimally necessary private information from game users is kept in secure storage. Robust security measures should also include security assessments and audits to ensure protective measures adhere to industry standards for all platforms. Finally, activity monitoring and logging should be implemented to detect and quickly resolve anomalies that could comprise user data. Identified vulnerabilities should be resolved using the latest software patch available for the affected platform.