

<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 |
| --- | --- | --- | --- |
| 3.0 | 2/29/2024 | Kaila Lancaster | The game application has been updated to implement a singleton class design. Requirements and design constraints have been identified. Development requirements on the client-side and server-side sections have been updated, as well as the developmental tools. I have chosen what I believe to be the most beneficial operation system and my reasoning. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room would like to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It. Draw It or Lose It is currently only available on Android smartphones only. If the game is now intended to be web-based and available on multiple platforms, we should identify the popular computer platforms and what it takes to create a game available for each one.

## Requirements

* The popular computer operating systems consist of Windows, Linux, MacOs, and ChromeOS. We should consider making the game available on all of these operating systems to have the biggest audience. The game can also be considered for release on apple devises which operate on iOS.
* A budget will need to be determined.

## [Design Constraints](#_2et92p0)

1. Possible programming languages necessary to make the game available on the above computer operation systems:

* For Windows: C++
* For Linux: C++, Java, C#, or Python
* For MacOS and iOS: Swift
* For ChromeOS: JavaScript

1. The Game Room may want to consider using a game engine to aid in the development process by providing a software skeleton with tools for the game which can save much time and money. Engines to consider:

* Unreal Engine
* Unity 3D
* GoDot Engine

1. A timeline or schedule will need to be discussed. Does the Game Room have a deadline for the game to be completed?
2. It will need to be determined if the game will need to have special art design, and if additional personal will need to be used to create the game art, or if this can be achieved with the already existing game art department.

## [Domain Model](#_8h2ehzxfam4o)

* The Entity class is the parent class to the subclasses Game, Team, and Player. Game, Team, and Player inherit the attributes and methods from the Entity class to keep the program clean and concise and keeps the code from being redundant.
* The ProgramDriver class contains the main method for the game. This class will create instances of the game in the GameService class.
* A singleton design pattern has been implemented for the GameService class to allow only one instance of a game to exist in memory at a time. This is accomplished by there being unique identifiers for each instance of a game, team, or player.
* The SingletonTester class tests to make sure that the GameService class is correctly implementing the singleton design pattern.
* The Game class adds teams a list of teams.
* The Team class adds player names to a list of players.
* The Player class holds the player information.

**822"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** | High quality display and graphics. Mac computers are more expensive, making them less commonly owned.  Macs are not as easily upgradable compared to other devices which might deter gamers on this system. By hosting an app on MacOS, you can take advantage of the built-in security features that will provide a safer environment for an app and the app users. Mac computers come with a set of developer tools, for example, Xcode, which has a development environment great for creating and testing apps, which will make it easier for the app developers to build and maintain the app. | Linux allows one to build a more dynamic gaming ecosystem which would be better for various games, due to its open-source nature. Linux has increased compatibility with different Windows games. Linux has limited driver support. Some gaming platforms like DR< system or game launchers may not have support on Linux. Linux is an open-source operating system, which means it is free to distribute and use. This can reduce the cost of hosting an app compared to other operating systems. Linux is highly scalable, letting businesses optimize their budget as their app grows. Linux has a low memory footprint, great resource management abilities, and great multitasking abilities, making it good for hosting apps that require a high performance. Linux has a smaller market share compared to Windows and macOS. Linux doesn’t have a standardized gaming infrastructure like Windows or MacOS, which can make it more difficult to finely tune your game for the best possible performance. | Most games work well on Windows. Steam and Origin are very popular sites for buying and downloading games, and these works well on Windows. Depending on the requirements of the app, Windows 11 can be more resource-intensive than its predecessors which can lead to decreased performance for users on older versions of Windows. Microsoft provides regular updates and support for Windows, ensuring that your app remains compatible and secure over time. Windows is the most used operating system overall, which an app on Windows operating systems will give you the ability to reach a larger customer base. | The game is already available for Android devices, so it will be good to expand to the iPhone since the design of the game may not need to change much as it is already made for phones. Mobile devices are portable and always accessible, allowing users to access your app anytime and anywhere. This convenience can lead to increased engagement and usage. App stores also handle payment processing, security, and updates, simplifying the app distribution process. Mobile devices may have limited access to certain hardware features when compared to Linux, Windows, and MacOS computers. This can restrict the functionality of the app or require workarounds to achieve certain desirable features. Mobile devices are more susceptible to security threats compared to MacOS, Linux, and Windows computers. Mobile devices often times have limited storage when compared to the other operating system. This can be a bad if the app needs to use a large amount of data or if it needs to store anything locally. Mobile devices have smaller screens compared to computers, which can severely limit the amount of content that can be on screen at once. |
| **Client Side** | There are costs to be determined to make a game available in the MacOS app store. Thoroughly testing the application on the commonly used Safari web browser on MacOS will be important to ensure compatibility. Staff will need to ensure that the app’s UI scales properly on various MacOS devices. MacOS devices come in many different sizes, so the game will need to be able to scale to the right size for the device being used. Staff tasked with maintaining the app, identifying bugs, and releasing app updates will be required, and a cost will need to be determined. | Linux users have access to several different web browsers to choose from, and each browser may render web content differently. Testing of the app on commonly used browsers like Google Chrome, Firefox, Opera GX, etc., will be important to ensure that the app runs well and displays correctly on all these different browsers. The app should be compatible with the user’s version of the Linux operating system to ensure compatibility since different versions may have different system requirements. The app should be compatible with Linux-based operating systems like Fedora, CentOS, and Ubuntu. | Depending on the specific requirements of the app, hosting it on Windows computers may require purchasing licenses for the operating system and other software components. Windows has been historically more susceptible to malware and security threats than other operating systems. Windows users have access to several different web browsers to choose from, and each browser may render web content differently. Testing of the app on commonly used browsers like Google Chrome, Firefox, Opera GX, etc., will be important to ensure that the app runs well and displays correctly on all these different browsers. | There are costs to be determined to make a game available in the Apple app store. The game will need to have good responsiveness to touch interactions on the touch screen mobile devices it will be used on. |
| **Development Tools** | Swift computer programmers are needed, along with the IDE software necessary to create the game program. Staff tasked with maintaining the app, identifying bugs, and releasing app updates will be required, and a cost will need to be determined. The purchase of frameworks would be beneficial in aiding the development process. Getting a cloud hosting service to make the app available on the web would be a great way to lower costs, provide flexibility to make fast updates and changes, and allow greater scalability for the app on computers. Testers needed to ensure app functions correctly and displays properly on each platform. | HTML, C++, Java, C#, or Python computer programmers are needed, along with the IDE software necessary to create the game program. Staff tasked with maintaining the app, identifying bugs, and releasing app updates will be required, and a cost will need to be determined. The purchase of frameworks would be beneficial in aiding the development process. Getting a cloud hosting service to make the app available on the web would be a great way to lower costs, provide flexibility to make fast updates and changes, and allow greater scalability for the app on computers. Testers needed to ensure app functions correctly and displays properly on each platform. | HTML or C++ computer programmers are needed, along with the IDE software necessary to create the game program. Staff tasked with maintaining the app, identifying bugs, and releasing app updates will be required, and a cost will need to be determined. The purchase of frameworks would be beneficial in aiding the development process. Getting a cloud hosting service to make the app available on the web would be a great way to lower costs, provide flexibility to make fast updates and changes, and allow greater scalability for the app on computers. Testers needed to ensure app functions correctly and displays properly on each platform. | Swift computer programmers are needed, along with the IDE software necessary to create the game program. Staff tasked with maintaining the app, identifying bugs, and releasing app updates will be required, and a cost will need to be determined. The purchase of frameworks would be beneficial in aiding the development process. Getting a cloud hosting service to make the app available on the web would be a great way to lower costs, provide flexibility to make fast updates and changes, and allow greater scalability for the app on computers. Testers needed to ensure app functions correctly and displays properly on each platform. |

## Recommendations

1. **Operating Platform**: I believe that Windows would be the best operating server platform for The Gaming room to choose for their game Draw It or Lose It. Windows has the largest customer base and is also known for its ease of use and customization options. It also allows for the user to download additional security software programs for added protection. With the goal of The Gaming Room wanting to reach a larger audience in mind and Windows also being known for the best environment for computer gaming, this would be the way to go.
2. **Operating Systems Architectures**:

* Windows kernel is a hybrid, meaning that it combines the features of monolithic kernel and microkernel architectures to get the performance and stability of both kernel types. The hybrid type kernel is easy to manage due to its layered approach, and it provides better security.
* Windows’ hybrid kernel can manage hardware and resources much more efficiently, leading to improved stability and system performance which is important to many PC gamers.
* Along with Windows being the most widely used operating platform, it is also known for its ease of use, and its long-term support from Microsoft. Microsoft provides long-term support for each version of Windows, meaning that Windows users can always get necessary system updates and security updates for many years after that version of Windows’ release.
* Windows has a lot of compatibility with other hardware/software devices. This makes it possible for many different devices to use this operating system.
* With the popularity of Windows over other operating systems, it is uncommon to find a program that is not compatible with most versions of Windows.
* For users of Windows 11, this version has a Game Mode setting, which is set to on by default, that can optimize the computer for gaming by adjusting how your games use your gaming graphics card.

1. **Storage Management**:

* Considering that Draw It or Lose It would like to have 200 high-definition image files for players to pick from, storage management will be important. There are file compression techniques that we could use, for instance zipping images files. Doing so would allow the decompressed files to be accessed when needed while reducing their size and making better use of storage. Another good approach would be dynamic asset loading. The game app can be programmed to load the image files on demand from the storage, rather than loading all the image files at the same time when a game is started. Use of cloud storage would be effective here, since you can offload the storage management and ensure scalability and reliability of the games data, reducing the storage requirements for user devices.

1. **Memory Management**:

* Since Windows has a hybrid kernel and receives benefits that come with a monolithic kernel, Windows can implement good memory management by associating all resources with the kernel space.
* Paging is a procedure that involves locating code in programs that are not currently being used and storing them on a hard drive to make better use of memory. Cache memory is important to make use of as well, since it can be most quickly accessed, and the cache can handle large amounts of data. Data stored in the cache can be used while instances of the game are in session, but can be discarded once the game is exited, meaning that memory is not being used up when it does not need to be. Caching frequently used photos in memory for when they are needed can reduce load times and optimize memory management.
* I believe the client should consider using a cloud server due to the many benefits it could bring to the application. For one, scalability. A cloud server would have no limits on computing power and it’s easy to upgrade memory space to support many users of the application. This would be good for the type of game that Draw It or Lose It is, one that aims to support many instances of the game at once with several teams and players. Cloud servers function a lot like traditional servers. They offer storage and processing power just the same. Since cloud servers are remotely accessed though, they are more secure and stable. They also provide reliable connections to authorized users. Cloud servers are great for reducing costs for companies as well. They reduce hardware expenses and the work of maintaining the cloud servers is the responsibility of cloud service provider. There would be monthly costs for this type of service, but the data can be accessed from anywhere and it can be expanded with the growth in a player base, which is one of The Game Room’s goals for making the application available on several other platforms.

1. **Distributed Systems and Networks**:

* A distributed app is a type of program that runs on one or multiple computers at the same time, communicating through a network. These apps can be stored on servers or cloud computing platforms.
* Distributed apps are broken up into two models, one for the server software, and one for the client software. The client-side software can access the app data from either a server or a cloud server, leaving The Gaming Room the option of what type of server they would like the best. The server-side software will process the app data from the server or cloud server. If a distributed app component fails, it can “failover” to another component in order to continue running.
* Components in a distributed system divide up the work and coordinate efforts to complete tasks efficiently.
* Distributed systems are great for gaming since they are meant to handle changes in workload well, such as a sudden increase in users, which is one thing that The Gaming Room will want to expect by releasing a game on a new platform.

1. **Security**:

* Windows has various security features. Anti-malware, anti-virus, and firewall that protect devices from attacks. It is possible for additional security systems to be downloaded onto your Windows computer as well, such as Norton, McAfee, and Aura.
* If Draw It or Lose It were to be a distributed application, it would be resistant to cyber-attacks, as there is no singular point of failure.
* Windows offers Defenders Smart Screen that protects its users from potentially nonsecure websites and will block them.
* Windows has what’s called Exploit Guard that protects your device from boot level malware. This system identifies attackers who are trying to attach drivers without a digital certificate, and it will not load these drivers, or load Windows. This allows only authorized drivers, apps, and files on your computer.
* UAC, or the Windows User Account Control, is a Windows feature that asks for the admin authorization of the user to run when you try and install something on your computer that is being run without these admin privileges. This makes it to where any system running in the background of your computer cannot run without permission.