

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc1877819677)

[Table of Contents 1](#_Toc11675208)

[Requirements 3](#_Toc162945821)

[Design Constraints 3](#_Toc1573609196)

[System Architecture View 3](#_Toc588932384)

[Recommendations 7](#_Toc220974998)

## 

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/18/2024 | Roger Fisher | Initial planning for the application |
| 2.0 | 10/5/2024 | Roger Fisher | Filling out additional details and editing table in Evaluation section |
| 3.0 | 10/17/2024 | Roger Fisher | Filling out and adding additional details in the Recommendations section |

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.

## 

The Gaming Room is looking to transition their existing Android game, Draw It or Lose It, into a web-based platform. They would like it to be accessible from several operating systems such as Linux, MacOS and Windows. Their objective is to make the game more accessible to a broader audience while maintaining its original gameplay, where teams compete to guess images over four rounds, each lasting a minute.

## Requirements

* The Gaming Room requires a web-based version of their current game that is available on Android
* The game must be able to run across several different operating systems such as Windows, Linux and MacOS
* The game should allow for one or more teams per game, with each team having several players
* Team and game names must be unique to prevent duplications
* Only one instance of the game should be running in the memory at any given time

## [Design Constraints](#_2et92p0)

* Cross-platform compatibility: The game must work across Windows, MacOS, and Linux. This requires a framework that can support multiple operating systems
* User interface Consistency: The existing Android interface should be replicated or updated to match the web-based platform (preferably replicated)
* Unique identifiers: The system must generate unique IDs for games, teams, and players to avoid duplicates
* Memory management: The system must ensure that only one instance of the game is in the memory at a time

## [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.

## [￼](#_8h2ehzxfam4o)

The UML diagram shows how the GameService class manages the Game, Team, and Player classes. GameService contains references to the Game class, and each Game has a reference to Team, while each Team references Player. This “has-a” relationship signifies that each of these entities contains an instance or list of the related class.

The Entity Class acts as the base class, connecting the Game, Team, and Player classes by providing shared attributes such as id and name. This inheritance ensures that each of these classes can access the common properties while maintaining their unique functionalities.

Additionally, we can see how the ProgramDriver uses the SingletinTester to test the singleton pattern implemented in GameService, ensuring that only one instance of GameService exists in the 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.**

## [￼](#_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 | Mac servers are no longer offered by Apple, which limits the options for hosting large applications such as Draw It or Lose It. While MacOS is still a secure and user-friendly system for smaller applications or web apps, it is typically not suited for server environments, especially large scale. Mac hardware and software can be prohibitive for some projects as well as expensive. | Linux is popular for hosting websites and programs because it is free to use and change. Linux has powerful performance and good security, making it a top option for servers, especially when handling many users. Unlike paid systems, Linux lets people change both the machine and the software. This is why big companies use it a lot but setting it up can be hard since it needs more knowledge compared to Windows or Mac. Linux is free to use, so it tends to be one of the cheapest server options even when trying to scale up. | Windows is a widely supported platform for hosting web applications. It has excellent resource management and offers a wide variety of tools for server management, though it can be vulnerable to malware and viruses without proper security. Windows is easy to use and familiar to many, but the licensing fees can make it more expensive than other platforms since it requires you to buy the operating system, security, and management tools to run it. | Mobile devices are often used to access apps from anywhere and require servers that can handle many connections. Android and iOS dominate the markets with their own specifications and hosting requirements which may make things easier. The cost is usually related to cloud services for hosting mobile applications. |
| **Client Side** | Developing on Mac costs more because Apple’s machines are pricey and unique. But Apple gives tools like Xcode that really help with building apps. Developers need to know how to use MacOS, which can be harder since few people use it compared to Windows or Linux. Despite that, Mac provides a smooth and steady experience, which is particularly important if users interact with the app.  **Time and expertise:** The developer must be acquainted with Apple products, though it may take more time to shift to Apple’s own languages like Swift. | Using Linux often costs less since there are no fees for using it. Developers have the freedom to change their setup as needed, but Linux can seem hard for those new to it. Even so, it works with many apps, providing a strong space for building websites or programs that work across different systems.  **Time and expertise:** Development for Linux is more complex than other platforms as the operating system have more freedom, but developers must be familiar with Linux development tools, often taking more time for development | Developing Windows is straightforward, with minimal cost and time compared to other platforms. There are several IDEs available, and Windows offers support for web browsers and mobile applications. Windows is a popular platform for developers to work with.  **Time and expertise:** The development in Windows is easier and takes less time than in other operating systems, such as Linux. However, tool licenses for Windows can be costly. | Developing apps for mobile platforms like iOS and Android can be slightly more complex because of platform specific languages like Swift (Apple) and Kotlin (Android), though frameworks like React Native offer cross platform support. Mobile apps need to run across different devices, adding an extra layer of difficulty to development.  **Time and expertise:** If the need for cross-platform functionality is involved, mobile development can take longer compared to development on PC platforms. Some measures of development difficulty are averted with knowing and using frameworks unique to mobile developments such as React Native. |
| **Development Tools** | Key tools for working on a Mac include Xcode, Swift, and Objective-C, all made to fit Apple’s system. For extra software, many people use Homebrew to add different programs. Mac uses a UNIX base, which makes it flexible and open to many other tools. Common languages like Java, C++, and Python work well on it. But if developers want to create something for devices from different companies, they might find it harder to use languages and tools that do not come from Apple.  Cost: Xcode and most of the development tools for Mac are free, but Apple’s hardware offsets the benefits by being costly. | Linux works well with several coding languages like Python, Java, and C++. IDEs like Eclipse or IntelliJ are often used for coding. Linux’s free libraries and frameworks allow developers to easily find things they need to build websites or apps. Many developers prefer it because it gives them power and control of their environments.  Cost: Most development tools for Linux are open source and free, making this a very cost friendly choice. | Popular tools for Windows include Visual Studio, C#, .NET, and Java. Developers also commonly use Eclipse and IntelliJ for cross-platform development.  Cost: Development cost for Windows can be flexible as there are several popular tools for developers to utilize. There are some tools such as Visual Studio that have licensing fees while there are others, like Eclipse and IntelliJ, that offer free versions of IDEs. | Android development uses Kotlin or Java, while iOS development mostly uses Swift. Tools like Android Studio and Xcode are standard for these platforms. There are a few options for cross-platforms, such as Flutter or React Native.  Cost: Android Studio and Xcode are free to use, however there are licensing fees for Google play and Apple’s app store. |

## Recommendations

Analyze the characteristics of techniques specific to various systems architectures and recommend to The Gaming Room. Specifically, address the following:

1. Operating Platform: For expanding Draw It or Lose it to other systems, I would recommend Linux as it is a solid choice. Linux is flexible, highly secure and an open-source OS which makes it to be cheaper. Linux can also run on a lot of different systems, so it should help the Gaming Room to reach more players on different platforms. In contrast, Linux’s capability for easy modification is still one of the significant factors for the Gaming Room to achieve their goal of attracting more players.
2. **Operating Systems Architectures**: Linux has three main parts: the kernel, shell, and hardware. The kernel manages important tasks like handling memory, CPU, and devices, which makes it ideal for the Gaming Room’s expansion. The shell is what users interact with to run commands or applications, it also allows the Gaming Room to modify settings. Linux works well for web apps that need to run smoothly across different devices.
3. Storage Management: For Draw It or Lose it, I suggest using Google Cloud storage services to manage data and game’s assets. Cloud storage provides easy scalability as the player base grows, without needing to worry about local storage limitations. Google Cloud also integrates well with Linux based systems. This recommendation also simplifies data backup and recovery, making sure that the game is always available for players.
4. **Memory Management**: Memory management is well-executed in the Linux system, which is desirable for Draw It or Lose It, as it is a multiplayer game. Through such factors as dynamic memory allocation and virtual memory, Linux guarantees that system resources will be availed in ways that favor gameplay. Second, if the game involves Java, the garbage collection of Linux, automatically frees all used memory, which makes the game remain active even if the number of users increases.
5. Distributed Systems and Networks: To allow Draw It or Lose It to work across different platforms, we can use a distributed system. This means that the game will be hosted on servers that players from various devices can connect to. Using cloud-based services helps keep things consistent, making sure that even if one part of the system has an issue, the others will keep running. Players can connect from anywhere if they have an internet connection. It is also important to plan for potential connectivity issues or outages, so having backup servers in place can help mitigate disruptions and ensure an enjoyable experience for all users. Cloud services also help with scaling, so as the number of players grows, the system can handle the extra load without the system crashing under the extra load.
6. **Security**: Linux users security convention is very appropriate when it comes to the Gaming Room since user security starts with permissions, firewalls, and encryption. It is also important that all aspects of the user data are protected, especially by ensuring that all the user’s accounts use multi-factor authentication and all the data in transit is encrypted. Security updates and scanning for vulnerabilities will offer ways in which to correct a threat and maintain the security of the user information.