

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/16/2023 | Justin Baldwin | Added Executive Summary, Design Constraints, and Domain Model sections to the Software Design Template. |
| 1.1 | 07/29/2023 | Justin Baldwin | Added Evaluation section |
| 1.2 | 08/13/2023 | Justin Baldwin | Added 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.

## [Executive Summary](#_heading=h.35nkun2)

The Gaming Room wants to create a web-based game that is available on multiple platforms based on their current game. The game is called Draw It or Lose It and is only available as an Android app currently. The Gaming Room must determine exactly which platforms and web browsers they want their game available on, only then is it possible to determine which languages the game must be written in.

Draw It or Lose It is loosely based on a 1980s television game called Win, Lose or Draw, where teams compete to guess what is being drawn. However, Draw It or Lose It will have an application that renders images from a large library of stock drawings. A game consists of four rounds of play lasting just one minute each. Every drawing is fully complete at the 30-second mark after a steady rate of rendering. If one team does not guess the puzzle before time expires, the other teams have an opportunity to offer one guess each to solve the puzzle in 15 seconds.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_heading=h.1ksv4uv)

The first design constraint for The Gaming Room is making the web-based version as similar as possible to the Android application of their game. They will want to make a smooth and coherent experience across all platforms for users that play on more than one device. The Gaming Room will want to create an experience that is recognizable for users regardless of the platform that they are using.

The second design constraint for The Gaming Room to consider when making a consistent experience across platforms is budget. The budget will always be a concern when you are attempting to make a coherent experience for your users. Making their Android application available as a web-based game will surely increase the overall expenses of the company.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The first object-oriented programming principle that is implemented in the UML diagram is the concept of inheritance. The Entity class is the super class or parent class to the Game, Team, and Player subclasses. This parent-child relationship helps to reduce code redundancy and improves the ease to add more subclasses in the future if necessary.

There is only one instance of the game existing in memory at a time because of the unique identifiers for each instance of a game, team, or player. There is a zero-to-many associative relationship between the GameService, Game, Team, and Player classes, which fulfills the technical requirements. The ProgramDriver class uses a SingletonTester class to ensure that only one instance of the GameService class can exist in memory at a time. The ProgramDriver hosts the main method for starting the entire application.

**"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](#_heading=h.z337ya)

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** | Mac offers a simple graphical user interface to improve server-side programming. Mac also offers server configuration under the system preferences menu. One of the biggest weaknesses when using Mac to host a web-based software application is the price. Generally, when compared to other options, Mac tends to be the most expensive to deploy and purchase licensing. | The biggest advantage to using Linux on the server-side is customization. There is a never-ending amount of options for server-side development. Another advantage to consider with Linux is how inexpensive it can be when compared to other operating systems on the licensing side of things. The biggest disadvantage to Linux is the high barrier to entry, it can be difficult for new programmers to do any server-side programming. | Windows has a wide range of available software for the server-side, similar to Linux. However, Windows can be a bit more expensive for licensing costs, comparable to that of Mac. A user-friendly GUI is also available to programmers on Windows. Windows tends to be one of the most common options for the server-side. | There is a lot of variety on the mobile side since there is no single, uniform operating system. It can be difficult to ensure a seamless server-side experience across the wide range of mobile devices. Potential licensing costs will vary across all mobile operating platforms. |
| **Client Side** | Mac tends to be the most expensive option on the client side for users. There is not too much experience needed from the programmer for client-side on Mac. The programmer must ensure that the client-side not only works on Safari, but other browsers as well, which most tend to be Chromium-based. Time needed on Mac will be quite comparable to that of Windows. | The costs of Linux on the client-side are generally lower when compared to Mac and Windows. However, the expertise needed might be a bit more for the programmer to build the client-side. Time will also be a substantial factor on Linux, most likely more than Mac and Windows. Proper testing will need to take place to ensure that the application is compatible with all web browser platforms. | The cost of Windows for the client-side will be pretty similar to Mac, but potentially a bit cheaper. Expertise will also be similar to Mac and quite a bit less proficiency needed when compared to Linux. Time needed on Windows will be about the same as Mac with some minor differences. Proper testing will also need to be done on Windows by the programmer to ensure complete compatibility. | Costs of mobile devices on the client-side are generally quite low. The expertise needed for mobile devices will be higher because there isn’t just one platform to build on. Time will also be higher for the programmer because of the multitude of platforms available. Client-side compatibility will depend on the device being used. |
| **Development Tools** | Regarding the client-side, HTML, CSS, and JavaScript will be needed to complete the front end. Swift will be the main language needed for server-side development on Mac. It is most likely that the development team will be using Xcode as the IDE. It’s unlikely that multiple development teams will be needed. There are no licensing costs involved. | Regarding the client-side, HTML, CSS, and JavaScript will be needed to complete the front end. Linux will be using JavaScript, Python, PHP, or Ruby to build out the back-end. It’s possible that multiple development teams will be needed if more than one of these languages is used. There are no licensing costs involved. | Regarding the client-side, HTML, CSS, and JavaScript will be needed to complete the front end. Windows will be written using C, C++, or C#. There most likely won’t be more than one development team as long as just one of these languages is chosen. There are no licensing costs involved. | Client-side will also be using HTML, CSS, and JavaScript for mobile devices. The back-end will be written in JavaScript, Python, PHP, or Ruby. It’s also likely to be written in Objective-C or Swift for iOS. Java or Kotlin will be used for Android devices. Multiple development teams will be needed with the wide range of languages being used. |

## 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 operating platform that I believe will allow The Gaming Room to expand Draw It or Lose It to other computing environments is the Linux operating system. The Linux server is a lightweight server built on top of the open-source and widely used Linux operating system.
2. **Operating Systems Architectures**: A Linux server is a server running a version of the extremely popular Linux open source operating system. Linux servers have gained a lot of popularity due to their flexibility, stability, and security. By using a Linux server, The Gaming Room will benefit from the open source community that has been established for quite some time. The Linux operating system also provides administrative services that are necessary to operating a client-server architecture. The linux architecture is highly modular and configurable, allowing for increased levels of customization and flexibility for developers. The architecture is a layered structure consisting of the kernel, system libraries, system calls, shell, applications, graphical user interface, and device drivers.
3. **Storage Management**: One storage management system provided by the Linux operating system is standard disk partitioning. The biggest benefit of disk partitioning is that each partition can be managed separately. Disk partitioning is the process of creating one or more logical regions on secondary storage that can be managed separately. Disk partitioning assists with system storage administration and proper storage utilization to increase efficiency.

Linus also offers the XFS or Ext4 file systems, which are highly scalable, necessary for The Gaming Room’s multi-platform needs. XFS also uses b-tree indexing for efficient disk space management and storage scalability. XFS is known for its great performance in handling large files, which can assist with the high-quality images needed for the Draw It or Lose It game. On the other hand, Ext4 does a great job of handling many small files, which could be beneficial for the game depending on the size of every image. Read and write performance is also increased through the Ext4 file system because storage blocks are allocated more efficiently before being written to the disk.

1. **Memory Management**: Linux has the option to utilize swap space as a memory management technique. Most operating systems will now swap pages instead of swapping entire processes. Swap-space management allows virtual memory to use disk space as an extension of main memory. However, a downside to swap space is that it can decrease system performance because disk access is much slower than memory access.

Linux memory management also implements demand paging and virtual memory. The philosophy of demand paging is to ensure that no pages are loaded into main memory until they are required. All pages of the frames remain in secondary storage until they are needed. When a page is requested for the first time in main memory, that page will be located in secondary memory. Demand paging ensures that a process begins execution with no pages present in physical memory. Demand paging is often considered a lazy loading technique, a method to delay initialization of an object until it is needed. Demand paging can improve the performance of Draw It or Lose It because images will not be loaded until they are needed, ensuring that the rest of the game displays quickly for the user.

1. **Distributed Systems and Networks**: A distributed system is one where components located on different systems communicate with each other through passed messages. A distributed system is physically separated but connected through a centralized computer network using specialized distributed system software. To the end-user, a distributed system will usually appear as one interface where the system can prevent failures and maximize total available resources.

One common method for distributing network traffic is load balancing. Load balancing allows you to equally distribute network traffic across a shared pool of resources to support a single application. It is common for applications to have many resource servers with data replicated across them. An application’s availability, scalability, security, and performance may be improved through load balancing. For example, a load balancer can automatically detect server issues and redirect client traffic to a functioning server which increases fault tolerance. Application performance is improved because the load is evenly distributed between servers.

1. **Security**: Load balancing usually comes with built-in security features that add another layer of security to an application. Load balancing assists against distributed denial of service attacks in which servers are flooded with concurrent requests to cause failure. A load balancer can monitor server and client traffic, as well as block malicious content. Load balancers will automatically redirect attack traffic to multiple backend servers to minimize impact of distributed denial of service attacks. Network traffic is also routed through several network firewalls for added security.

Cryptography is one of the most important security tools to any operating platform. Cryptography is used to restrict the potential senders and/or receivers of a message. For a distributed application, it’s necessary to verify the destination address of a packet to determine who sent or received that packet. Cryptography ensures that private user data is transferred with confidentiality so that unauthorized users don’t have access. Many applications utilize hash functions and digital functions to ensure user protection.

User authentication is a major security concern for operating systems and Linux is no exception. User authentication is a crucial part of security for a distributed system. The most common approach for authentication in Linux is the use of passwords. The user-supplied password must match the password that is stored in the system’s database.

References

Amazon Web Services. (n.d.). *What is Load Balancing? Load Balancing Explained - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/what-is/load-balancing/

Hat, R. (2022, December 16). *What is a Linux server?* Www.redhat.com. https://www.redhat.com/en/topics/linux/linux-server

javatpoint. (n.d.). *OS Demand Paging - javatpoint*. Www.javatpoint.com. https://www.javatpoint.com/os-demand-paging

Kiarie, J. (2022, July 4). *Ext4 vs XFS – Which Filesystem Should You Use*. LinuxOPsys. https://linuxopsys.com/topics/ext4-vs-xfs-which-one-to-choose

Mitra, N. (2023, April 18). *Linux Architecture*. Scaler Topics. https://www.scaler.com/topics/linux-architecture/

Silberschatz, A., Galvin, P., & Gagne, G. (2009). *Title Page - Operating System Concepts, 8th Edition [Book]*. Www.oreilly.com. https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/silb\_9780470128725\_oeb\_tp\_r1.html