

# **Draw It or Lose It**

# **Software Design**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/14/22 | Sean Bruyere | Draft design for Draw it or Lose it |
| 2.0 | 10/01/22 | Sean Bruyere | Update Client/Server Requirements |
| 3.0 | 10/15/22 | Sean Bruyere | Final Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to bring a bit of the 80s to the modern age. The up-and-coming gaming company has developed an Android game app similar to the 1980s television gameshow *Win, Lose or Draw.* The game is called *Draw It or Lose It* and consists of teams going against each other to guess an image as it is rendered within the allotted time. Each round lasts 1 minute, each image is fully rendered by 30 seconds, and if the team does not guess the puzzle - the remaining teams have one guess each to solve it. To fully realize the potential of the product, The Gaming Room is looking to expand out of the Android based environment and move to a web-based platform. This would give the client access to a broader audience that would otherwise be missing out on the spirit of the game.

## [Design Constraints](#_2et92p0)

* Must be web-based game
* Browser based computer language such as JavaScript
* Full Stack development process – Front end design and back-end server
* Validate unique games and team names with identifiers
* Single instance of each game should exist in memory
* Each game must have one or more teams

A web-based video game for a low-level complexity game like this is a smart idea to expand onto other operating platforms. Each browser type from Safari to Edge will support a JavaScript based game, this would make *Draw it or Lose it* universally accessible by Linux, MacOs, or Windows operating systems. However, since many Android apps are Java based this would require a whole rework of the program using the JavaScript language for the front-end operation of the game while Java can still be used for back end processing. There also needs to be consideration as to what platform will host the game server while allowing for the best performance of the game. The design of the game itself has certain criteria which needs to be met in order to satisfy the client. First, there can only be a single instance of the game in memory to help minimize performance issues, this can be accomplished using the Singleton Pattern. Next, each game is required to have at least one team in order for the game Object to be created. Finally, there needs to be unique instances of each game and team. The Iterator Pattern will help encapsulate sensitive information and be useful to validate if a Team name or game already exists. This validation will help the performance of the game on the web platform and ensure and enjoyable experience for the user.

## [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)

Object Oriented Programming is a design pattern based around the creation of objects. This greatly helps reduce redundancies in a program, makes them far more modular and easier to update, and helps secure sensitive data. In the UML diagram below, we start with the Entity class. This class is considered the parent class to the Game, Team, and Player objects. Here, any redundant attributes and methods are stored and inherited by the children objects. An example of this is the encapsulated (private) attributes Id and Name. Each Game, Team, and Player object will have its own Id and Name associated with it and will be set through each objects’ constructor. All this information will be handled by the Entity class, away from any mischievous user. This increases security of the program from any unwanted changes that may affect performance. With this, we also have an example of polymorphism which means the ability to take on many forms. As stated, each child object has its own Id and Name associated with it, however, a single instance within the entity class handles the attribute for each. This would also include the methods within the Entity class. If getName() is called for a Player, the name of the player will be displayed, but if the same method was called in Team, the team name will be found. This helps reduce redundant code in a program which makes it far easier to debug or even upgrade in the future, saving time and cost for the developer and company. Looking further in the UML diagram there are four classes with multiple associations to one another; GameService, Game, Team, and Player. Each of these classes will take advantage of methods within another to help form the object. The Game class will use the Team class to form a list of teams and uses GameService to help create the new team object within the game. The Team class will do the same, but for a list of Players. Each object serves a specific purpose that makes it easier to address problems and modify future upgrades. This is what makes OOP such a valuable and trusted method for developing programs.

**"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** | Features:  Profile Manager, Open Directory, File Server, Caching Server, and Time Machine Server.  Advantages: interoperability with Apple products.  Weaknesses:  As of April 2022, Apple has discontinued the Server platform and transitioned features to macOS High Sierra and later.  Does not scale well when compared to other platforms.  Works best with Mac based applications. | Features:  Open source, built-in security controls. web interface for managing storage, network etc.  Advantages:  Stability, Security, Economical (Costs range from $225 to $1500 per year for Ubuntu and free for Debian), Multitasking capabilities, and Scalable across many platforms.  Weaknesses:  Requires more expertise to implement.  Many different flavors of Linux Server OS, each with their own cons that need to be taken into consideration. Updates often require community support. | Features:  Admin Center to set up a quick website and web interface to monitor and configure servers, Windows Defender Advanced Threat Protection  Advantages:  Best active support.  Integrates well with Microsoft Products which are often the most used.  More features than the other server possibilities.  Weaknesses:  High-Upfront costs (standard: $1069, Datacenter: $6155). | Features:  Mobile devices were not designed with the intent of becoming servers. However, open sourced software such as Java based i-jetty has given them the possibility to be run as such.  Advantages:  Little processing power is needed to create a web server, the low cost energy usage of a Mobile Device compared to server stacks could be an advantage.  Weaknesses:  Reliance on ISP connection if connected through WiFi.  Cellular service provider Data Plan considerations. Both of these should be cost considerations that add to costs of running the server.  Bandwidth issues could occur. |
| **Client Side** | **Cost:**  Higher upfront Costs due to more stringent Software requirements and less customizable Hardware capabilities.  **Time:**  Moderate time requirements as Apple products are relatively user friendly.  **Expertise:**  Moderate expertise is needed to setup Cross-Browser compatibility testing. Chrome and Mozilla can be run on Mac. However, more set up may be needed to test on Mobile devices outside of Mac. | **Cost:**  Open-Source product which makes it cost effective as many are free. However, there is little technical support.  **Time:**  High level of expertise needed to setup which will add to time constraints.  **Expertise:**  High Level of expertise as many devices are not set up to run or operate on Linux. More of a niche OS with lots of terminal -bash work. Cross-browser testing may be more difficult. | **Cost:**  Some upfront cost to run with the average Windows OS costing around $200 for Pro. Can find more affordable devices for software development than that of mac.  **Time:**  Least time to develop as Windows has widely available applications that share compatibilities already.  **Expertise:**  Minimum expertise needed for setup as Windows has many compatibilities. May take more time to work with MacOS Safari browser. Will work well with Chrome, Edge, and others. | **Cost:**  Higher costs to develop with specific mobile apps in mind such as Android or iOS.  **Time:**  It would take longer to develop and setup the product through two separate applications in Android or iOS  **Expertise:**  More work and experience would be needed to develop cross-platform capabilities through the mobile devices. |
| **Development Tools** | The more common and preferred programming language for deploying on Mac is Swift. Past deployment languages for Mac include Objective-C. However, many common languages such as C++, C# and Java can be used on Mac.  The Common IDE is XCode which is built into the MacOS, it supports Swift, C, C++, Objective-C, Java, and Python. Code editors such as  Visual Studio Code can be downloaded and used for C++, C#, HTML, CSS, or Javascript development.  Framework tools such as Xamarin and React can be used to help reduce costs and create cross-platform capabilities.  Should only need a single development team for the app for required deployment on Mac.  For Enterprise use of Visual Studio Code, the cost is $250 per user per month. Full Xcode capabilities is $100 while it is generally available for free for Mac users. Xamarin framework, owned by Microsoft is free for general use, $1000 per year for professional and $1900 for Enterprise use. | The open-source nature of Linux provides support for many different programming languages.  Code editors such as Sublime text which supports many different programming languages including:  C++, C#, Java, JavaScript, Objective-C, Python, and SQL to name a few. Other, more specific IDEs are available as well, such as IntelliJ for Java, or even Visual Studio Code or Psycharm for python.  Many of the IDEs are free to use for the single developer. Depending on the language, multiple development teams may not be necessary. Code Editors such as Sublime text may not offer many of the capabilities IDEs on other platforms offer if building with C/C++ language. | Many Windows applications are developed using C# with a .NET framework.  Due to the Windows OS popularity many different IDEs work on the platform. One of the most popular for this environment is Microsoft’s own Visual Studio. This IDE supports more than 36 programming languages and includes many free-to-use Microsoft APIs (Sogbesan).  This OS provides the development team a plethora of tools to build the product. The popularity of Windows based products provides lower startup costs and setup when compared to Mac or Linux. Framework tools such as Xamarin and React can also be used in this capacity.  The cost of Visual Studio is $45 per month per user or $250 per month per user. Similar costs for framework usage as for Mac. | Many of the most popular mobile devices are based on Open Source Android SDK which is largely based on the Java programming language. Google provides the main IDE tool for Android app development called Android Studio. Google has transitioned the language preference to the Java based Kotlin language.  The other most popular devices are based on the apple iOS which is often developed using Apple’s Xcode IDE. The languages for this device include Swift and Objective-C.  To develop applications for these devices, multiple development teams would be needed to create an Android app and an iOS app. This provides less cohesive code as each are developed separately which also adds to the cost of the project.  The android studio license is free to use while the Xcode IDE requires a Mac computer and a $100 cost for full privileges. |

## 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 server operating platform I suggest for this project is the Linux OS. The open-source software provides incredible security, reliability, and scalability at an affordable cost. The overall management of the system can be a great benefit as “you can modify a system or program configuration file and effect the changes without necessarily rebooting the server, which is not the case with Windows” (Kili). This allows regular updates without disrupting the performance of the game. Another benefit is the added security of the Linux based system, as the platform highly restricts outside access to important systems. The one difficulty presented is the expertise required to set up the server. However, the overall cost and reliability of the system over the lifetime of the server makes it the obvious choice.

1. **Operating Systems Architectures**:

The basic Linux architecture is made up of 5 layers. At the center of the system is the hardware, which contains the main components of the computer such as the CPU or RAM. In order to communicate with these peripherals, the kernel is needed. The kernel is the core piece of the OS which communicates directly with the hardware and provides ‘low level services to the user of the system’. The key parts of the kernel include the system call interface, process management, memory management, the virtual file system, the network stack, and the device drivers for the hardware (Jones 2007). Next, is the system library which includes a set of predefined programs or functions which help implement functionality in the system. Then, there is the system utilities that are responsible for specialized and/or individual tasks as directed by the Kernel and System Libraries. Finally, is the shell which provides a way for the user to provide input to be executed by the kernel layer.

Some of the benefits or features of the Linux OS is that it is portable, providing the ability to work on many different hardware devices. It also is a multiuser and multiprogramming system that allows multiple users and applications to access system resources or run at the same time. Finally, it provides a hierarchical file system that provides a way to manage files in an easy to navigate tree like structure.

1. **Storage Management**:

A useful storage management technique to implement for this game is indexed allocation with direct access. A key for storing all files on the server will be minimizing any fragmentation on the device. Indexed allocation helps minimize waste by providing each file with its own index block. Each index block address is stored in a directory which allows direct access and results in faster load times. In the Linux file system, a multi-tiered index allocation is used which stores files on blocks near their index to reduce the time to find the files. This reduces the time needed to shift the data from storage to memory (Silberschatz, 2009).

1. **Memory Management**:

The Linux OS utilizes both virtual memory as well as physical memory managed by the Linux kernel through memory mapping. The kernel provides each process with a virtual memory space that is larger than the physical memory space available. Only the virtual memory that is in line to be used will be allocated to the physical memory for processing. How the virtual memory is allocated to the physical memory is called memory mapping. Here, the kernel maintains a page table for each process and ‘maps’ the relationship. Page swapping is used to help manage the physical memory as processes are completed and new virtual memory is needed for the next process. To help recognize faults and perform the swaps more efficiently, demand paging can be implemented. Here, ‘a page is brought into the physical memory only when it is needed or demanded by the CPU’ (Studytonight.com). An effect of these page swaps to virtual memory is called thrashing. This greatly reduces the performance of the system as the virtual memory is strained from overuse, leading to page faults, and slowing of the system. To help minimize thrashing, Linux implements a Least Recently Used algorithm to utilize swapping more efficiently. This isolates the older pages on the memory for swapping and keeps thrashing low and keeps overall performance at acceptable levels.

1. **Distributed Systems and Networks**:

The best distributed system to use for this scenario is the client-server model. Here, the game user of sends a request to the server through HTTP protocols and the necessary files or processes are returned in response. Using REST methodology with a stateless system allows the server to authenticate and take numerous requests from multiple different users independently. This provides scalability for the *Draw it or Lose it* game. The use of REST APIs and HTTP protocols also allow portability amongst the many different browsers or platforms a user may use.

A useful method to ensure connectivity and to prevent outages is to implement load balancing to spread client requests evenly across the servers. This has the added benefit of adding security and can allow fast response times. The redundancy of these systems also provides protection from connection issues and outages. If a server instance is unavailable the load balancer algorithm can find a good server to implement the process an prevent downtime (Citrix).

1. **Security**:

Authentication and authorization with the implementation of least privileges are keys to providing the necessary security in the system. To implement authentication a simple login credential with a username and password can be used. In order to increase user security, a password using letters, numbers, and a specific set of characters should be suggested. The use of biometrics or one-time passwords may be too complex and unnecessary for this simple game. To ensure secure storage of the password, the sensitive data should be stored in a hashed state that prevents plaintext view that is accessible from a malicious source (Silberschatz, 2009). For authorization, the player should only have access for the necessary features needed to run the game and should not have the ability to modify the game in any way.

The open-source nature of Linux operating systems provides constant overview of any vulnerabilities by the many users of the platform. This can provide increase levels of securities not always available to other platforms. The Linux platform restricts root access with strict user privileges, only allowing just enough permissions for the common user to accomplish common tasks. It also provides kernel defense systems including “firewalls that use packet filters in the kernel, the UEFI Secure Boot verification mechanism, the Linus Kernal Lockdown configuration, and the SELinus or AppArmor Mandatory Access Control security enhancement systems” (Day, 2021). These systems help prevent malicious attacks from ever penetrating the low-level operations of the computer system and causing a meltdown. All these components help prevent access to any of the player information and ensures a secure system.

Reference

Arslan, Ali. “The 10 Best Ides and Code Editors for Linux.” *MUO*, 20 July 2022, <https://www.makeuseof.com/best-ide-code-editors-for-linux/>.

Day, Brittany. “How Secure Is Linux?” *Linux Security*, https://linuxsecurity.com/features/how-secure-is-linux.

“Demand Paging in Operating Systems.” *Studytonight.com*, https://www.studytonight.com/operating-system/os-demand-paging.

Jones, M. “Anatomy of the Linux Kernel.” *IBM Developer*, 6 June 2007, https://developer.ibm.com/articles/l-linux-kernel/.

Kili, Aaron, and Abhijith K S. “Aaron Kili.” *6 Reasons Why Linux Is Better than Windows For Servers*, 22 Nov. 2017, https://www.tecmint.com/why-linux-is-better-than-windows-for-servers/.

Manchanda, Amit. “The Ultimate Guide to Cross Platform App Development Frameworks in 2022.” *Insights - Web and Mobile Development Services and Solutions*, 15 Sept. 2022, <https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/>.

Silberschatz, Abraham, Peter Baer Galvin, & Greg Gagne. (2009). *Operating System Concepts, Eighth Edition*. John Wiley & Sons, Inc.https://learning.oreilly.com/library/view/operating-system-

concepts/9780470128725/.

Sogbesan, Tayo. “The 5 Best Ides for Programming on Windows 10.” *MUO*, 2 Aug. 2022, https://www.makeuseof.com/windows-10-best-ide/.

“Visual Studio IDE Pricing 2022 - Trustradius.” *Visual Studio IDE Pricing Overview*, 2022, <https://www.trustradius.com/products/visual-studio-ide/pricing>.

“What Is Load Balancing? What Load Balancing Solutions Are There? - Citrix.” *Citrix.com*, https://www.citrix.com/solutions/app-delivery-and-security/load-balancing/what-is-load-balancing.html.