

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

Andrew Park

## 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 | <04/10/23> | <Andrew Park> | Append executive summary, design constraints and describe UML diagram |

**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)

Our client, The Gaming Room, is seeking to develop a web-based version of their game “Draw It or Lose It”. Currently it is only available on Android. The web-based version of the game will need to be scaled for both mobile and desktop platforms. Since it will be web-based we can consolidate on a single web-based language rather than scripting for each platform that can connect to the Internet.

## [Design Constraints](#_2et92p0)

1. One web-based application
2. One developer team
3. Verified scaling on mobile and desktop platforms

## [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)

From the UML class diagram below, the Game, Team, and Player class inherits from the Entity parent class. These child classes inherit the id and name attribute while also inheriting a few methods. This modularizes the code and let’s programmers employ Abstraction which prevents programmers from repeating lines of code. Furthermore, all the child classes of Entity are associated with one other. All these associations are “0 to many” associations where each class can associate with 0 or many of these other classes. Finally, the GameService class is only associated with the Game class which means that the GameService class can only be called once the Game class is instantiated.

**"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** | According to Apple’s support site, macOS Server has been discontinued since 4/21/2023 but users can access server features using macOS Monterey (workstation OS). It's not recommended to use macOS Monterey for a web server. | Linux server is already a very popular platform to host websites as many Linux distributions are open-source, resource efficient, and robust. Linux web servers typically don’t have a Desktop GUI so many operations must be done through scripting and command lines which requires intimate knowledge of Linux directories and commands. | Windows IIS is another popular web server platform that does come with a GUI interface which makes deployment much easier. The GUI makes the server not as resource efficient as Linux but allows for more users to be able to quickly spin up a website. Windows IIS requires a Windows Server license. These license can cost up to $1069. | Mobile device operating platforms do not have the processing, memory, or electrical power to host a web server. Although phones have become arguably stronger than the servers of the late 20th century, the amount of internet traffic has also significantly increased which means that mobile OS platforms will not be able to sufficiently perform server tasks. |
| **Client Side** | Given that the application will be available by web, there’s nothing particular about the coding except for Safari (native macOS web browser). The macOS licensing is usually included with the hardware purchase for Apple products. Typically macOS is used by all kinds of users ranging from very technical to non-technical given it’s easy usage. | Given that there are many workstation distributions for Linux, programming the web client application side would involve popular browsers like Chrome, Firefox, Opera, and more. Most workstation distributions are free/open source and can be spun up relatively quickly. Linux workstation installs are typically quick and behave similarly to a Windows, or macOS, installation process. The directories for Linux are different which would take a Linux expert. Typically Linux is used by expert users, or by those who just like Linux the best. | Windows OS costs can be minimal if users upgrade during Microsoft’s offer periods. For example, when Windows 11 came out, those with licensed Windows 10 machines could upgrade at no charge. However, purchasing a new Windows license may cost up to $150 depending on the version desired (Home, Pro, Enterprise). Windows installations are quick and programming for Edge (native Windows web browser) should be relatively easy since it is Chromium based. Windows typically attracts businesses and has a wide range of experts. | Mobile devices like iOS or Samsung Android are available with the purchase of hardware and aren’t for individual sale. The standard Android platform is open source and available to the public. Since most mobile devices can use web browsers, programming for phones would require resizing of content but the backend processing should be the same. However, mobile devices will need apps which will take longer for programming. iOS may take experts within Apple’s domain but Android should have many experts to speak with and collaborate since it is open source. |
| **Development Tools** | Object oriented programming languages like Java and Swift can be used to interact between the web browser and operating system. A popular macOS IDE, ironically, is Visual Studio. Visual Studio is free for students and some organizations purchase this for it’s developers. Multiple teams will be needed. | Similar to macOS, Linux operating systems can interact with web browsers using object orient programming languages including C++ and Java. A popular Linux IDE is Sublime Text which can be used for a multitude of programming languages. Sublime Text is free. Multiple teams may not be needed as most of these programming languages are common. | Windows typically relies on C++ and Visual Studio to interact with web browsers. Java can be installed but is not considered native. A popular Windows IDE is CodeBlocks which is primarily for C++. CodeBlocks is free. Multiple teams may not be needed as most of these programming languages are common. | Like macOS, iOS uses Swift for it’s operating system interactions with web browsers and apps. Android is a modified version of the Linux kernel which is written in the C programming language. There are not many popular IDEs for mobile devices since the screen is so small and there are so many lines of code. However, there are IDEs like Android Studio. Android Studio is free. Multiple teams will be needed to adjust for Swift and the backend. |

## 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**: Given the software requirements, the Linux OS is recommended as the solution. Specifically, a Linux OS like CentOS or Ubuntu with an Apache web server component deployed on a cloud computing platform developed with a distributed systems framework. This will allow The Gaming Room to expand their game by scaling up easily at low cost. Although Linux is more difficult to manage from an administrative standpoint, it’s efficiency and efficacy compared to Windows Server OS is considerable. Furthermore, it can provide the same web interface for users across the globe. In a nutshell, Linux is the optimal choice for web-facing applications like “Draw It or Lose It.”
2. **Operating Systems Architectures**: Linux’s kernel OS architecture, is a modular monolithic type; meaning that although there is a core component of the OS, additional modules can be added as loadable kernel modules where drivers and file systems can easily be added at runtime without recompiling the entire kernel which would cause system runtime interruption. In simplified terms, with the kernel OS architecture, we can add features with minimal downtime from a plethora of operating platforms including but not limited to Windows NFS, Andriod OS, and Apple iOS.
3. **Storage Management**: The cloud computing platform will be key in storage management for scaling. With AWS elastic block storage, The Gaming Room will be able to add additional storage space for their game instantaneously with ease. Furthermore, the elastic block storage will be composed of an SSD based storage solution using charge-flash trap technology instead of magnetic disk writing. This recommendation comes from the insane performance increase from SSD technologies. In order to access the storage data, which includes player information, game functions, some images, and website code, the direct access (or relative access) method will be implemented. Due to the random nature of the game, sequential access is not favored.
4. **Memory Management**: Linux has a few techniques to manage memory effectively and efficiently, including memory allocating, paging, swapping, sharing, and protecting. The benefit of virtual memory is that the Linux OS can use paging to divide memory into pages and map each of them to a physical page frame in RAM which allows for the full utilization of the physical RAM chips and keep process lifecycles flowing continuously. Swapping is another technique in which the RAM borrows from the storage resources in order to queue and run processes which is another benefit of virtual memory. Compounding the benefits of virtual memory, the use of jpeg files and lossy compression will increase website performance and allow the Linux OS to run other processes simultaneously by minimizing the amount of memory needed for a single web page or game application instance.
5. **Distributed Systems and Networks**: Using a distributed systems framework on top of a cloud computing platform will allow for improved runtime speed, efficiency of hardware, global presence, and redundancy in case of a catastrophic failure. Using distributed systems techniques like load balancing will allow the application to run smoothly across the internet at scale. With the load balancer near the front end of the interface, it will connect to the resource servers at a different site (with different hardware) providing each resource server (web server, database server, etc) with an equal number of requests thus preventing an overload and maintaining resources efficiently. Cloud computing allows companies to replicate their data across availability zones allowing for faster access to resources by placing them near the edge of their users. Furthermore, by distributing the resources, we can use other techniques like Content Delivery Networks to provide scalable photos with lossy and lossless compression rates so that users can download high resolution images if they want and still get fast runtimes when they are viewing the game webpage.
6. **Security**: First and foremost, any communication between user’s client machines and the front end facing components of the architecture will all be encrypted using TLS effectively making all public facing traffic HTTPS based. This will allow for the safe communication over a public medium such as the internet. This will not only keep The Gaming Room safe but also it’s users safe. To add an additional layer of security, all user credentials will be hashed in a database. Although this could cause some user support to be difficult, it will be imperative to maintain credential security across the internet and devices. If The Gaming Room admins can’t even see their information, it will make it that much harder for attackers to gain access. Furthermore, 2FA authentication will be offered and rewarded to users through the use of free bonus material if they sign up. By incentivizing increased security, we can make “Draw It or Lose It” a pleasant gaming experience for all.