

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 08/15/2021 | Alexis Alexander | 1. Recommendations Completed |

**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)

The Gaming Room needs to create a web-based gaming app for their game Draw It or Lose It. The gaming app needs to allow the involvement of multiple teams with multiple players assigned to the teams, require unique game, player, and team names, and only allow one game to be run at a time. Our proposal is to create a program based off the company’s existing android-based game that consists of classes that are associated with each other and inherit from a base class to allow multiple teams and players, but also only allow one game at a time.

## [Design Constraints](#_2et92p0)

The two major design constraints are the need for multiple players within multiple teams and that this will need to be a web-based gaming app. Because the gaming app will need to be web-based, we will need design the game to implement logic allowing the game to handle different browsers. We will need to use a cross platform web framework to handle the browser differences. We must implement logic that enforces unique team, game, and player names. It will be helpful if our code is reusable in this aspect as we will need to use it to check for three identifiers.

## [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)

We have a base class, Entity, that creates a relationship between the Game, Team, and Player classes. Game, Team, and Player are all associated with another but also all inherit from the class Entity. Because they inherit from the Entity class there is a bit of polymorphism going on with the toString method, and some encapsulation with the setter and getter within the Entity class. We then have the GameService class that has a multiplicity association with the Game class. This means that the GameService class can have zero or more objects that participate within the relationship. This is the same for the association between the Game and Team class and the Team and Player class. GameService has some encapsulation as its getters are included within its class, but its setters are in the associated Game class. Lastly, we have the ProgramDriver and SingletonTester classes that are not associated with any other classes. The ProgramDriver class uses the SingletonTester class to test the entre program.

****

## [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 must work together.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Con: MacOS 11 does not operate as a real server as many of the features of the original MacOS server before High Sierra  Con: A server-based deployment method is virtually nonexistent as the MacOS server does not truly operate as a server | Pro: Server built upon open-sourced OS. This makes the server low cost as a license is not required.  Pro: Unlike the Windows and MacOS servers, Linux has many servers. There is a variety of different servers instead of updated versions of the same server.  Pro: Linux servers give the staff full root access to the servers meaning that they control everything.  Con: Time must be allotted to learn to manage and configure server. Otherwise, you will have a server that is highly susceptible to malware and failure. | Pro: Allows for developers to use Linux operations like Open SSH, Tar, and Curl.  Pro: Hybrid server system with Azure. This allows for Virtual Machines that run on physical servers within the Data Center.  Pro: Windows Defense Exploit Guard to help prevent malware attacks.  Con: Closed source with a Shared Source Initiative. This makes it expensive to license. (Around $1500 for Windows 2019) | Pro: Web server apps available via the Apple and Android app store. This makes the server low cost as many of the apps are a few bucks.  Con: You cannot run a web server over wireless connectivity. A local area network would be necessary.  Con: CPU would be required for low power use. |
| **Client Side** | Pro: Built-in browser (Safari) that enables JavaScript by default  Pro: Safari is a freeware web browser so it will be free to use for the end user  Con: Safari is no longer available across platforms so wouldn’t be of use for a Windows OS | Con: Linux does not have any built-in browsers on its OS. A Linux distro needs to de downloaded to create a Linux OS and distros sometimes come with different browsers.  Pro: Most Linux OS users use Mozilla Firefox as a browser so this would more than likely be the browser of choice for this situation  Pro: Mozilla Firefox is an open-source web browser that is also free. Open source means that we will have access to source code. Mozilla Firefox is also a cross-platform browser  Pro: With a Linux OS, almost all browsers are available for download.  Con: Moderate learning curve for staff and users as a browser will need to be downloaded if the distro does not come with one. The download process for a browser in a Linux OS can sometimes be difficult. | Pro: Built-in browser (Microsoft edge) that is a cross platform web browser.  Pro: Microsoft Edge is a chromium browser which gives access to extensions that one would usually get with a chrome browser. This provides additional browser features that may be needed that aren’t built into the browser.  Pro: Minimum learning curve for staff and users as Microsoft Edge is a newer version if internet explorer. | Pro: Almost every major browser is available and can run on a mobile phone.  Con: User Interface would need to adapt to the mobile device to provide more of an app feel  Con: With a mobile web browser, there is always the possibility of input lag, portrait vs landscape issues, and navigation gesture malfunctions. |
| **Development Tools** | Con: The MacOS server app is needed along with an old Mac Computer to create a “server”. But the “server” does not act like a true server at all. | Pro: There are many distros available to create Linux servers (Ubuntu, RHEL, CentOS, SUSE, etc.)  Pro: There aren’t any developmental restrictions  Pro: Distros like RHEL and Ubuntu offer different subscriptions that provide different support levels dependent on the additional support needed that the staff cannot provide.  Con: The additional need for extra support regarding the server will include additional costs that may not have been included in the budget. | Con: Windows Server package must be purchased to have a windows server. This can cost up to $1500.This cost includes Windows and SQL server licenses.  Pro: Server options offered. (Cloud, Hybrid, and On-Premises)  Pro: Microsoft Azure to help manage and configure Cloud and Hybrid servers. | Pro: No registration required on the server side or client side for any development tools.  Con: Additional UI and UX design needed to help with the adaptation of the game to a mobile device (Unity, Corona SDK, Cry Engine, etc.) |

## 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:** I would recommend The Gaming Room runs the gaming app on the Windows OS. This operating system requires minimal training for staff and users on the client side. The windows operating system also has many built in apps and specs that assist with building a server and running a web-based application.
2. **Operating Systems Architectures:** Windows architectures consist of kernel mode and user mode. The client and end users will have to navigate in user mode while the server side will require some work within kernel mode. Windows comes in a 32-bit and 64-bit version of Windows 10. The 32-bit version comes with around 4GB of RAM while the 64-bit comes with 128GB of RAM.
3. **Storage Management:** We would like for our web-based application to load and run smoothly and quickly so I think the best storage management would be an SSD. SSD’s are faster and smaller than your standard HDD drives that are used when a large amount of storage is needed. In this case, we need about 1.6GB of space for our images alone so an SSD will give us more than enough space with most allowing for 120GB of space.
4. **Memory Management:**  To manage memory, the best plan of action is to have only the needed images loaded into main memory at one time. We know that, depending on the version (32-bit or 62-bit), we get a good amount of main memory or RAM. Going with the 64-bit would provide a large amount of unused RAM however, I would still only load images that are needed at the time to avoid the possibility of main memory overload.
5. **Distributed Systems and Networks:**  To handle simple issues like outages or connectivity, the company will need to make sure their servers are secure and strong enough to support large player volumes along with backup power for power outages. We know that the Eclipse IDE can be used for cross-platform web-based application creation but there is also NW or NodeWebkit. It’s an IDE that can be run on Windows, MAC OS, and Linux. There are other platforms that do this, but I chose this one as those are our three main platforms. When the game is created you export the game file into the web and then the web-based app is available through almost all web-based operating systems when using the specified operating systems and browsers.
6. **Security:** Windows has a built-in security protection software. The security system scans for malware, viruses, and threats in real-time, with automatic updates to the system to keep the user info and system itself safe. Windows also comes with defense exploit guard when you purchase a server to provide additional protection. We can also build security into our code by securing our REST API’s.