

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 | 9/17/2022 | Dylan Andrews | Filled in all blanks and described diagram. |

## [Executive Summary](#_sbfa50wo7nsh)

Gaming Room wants us to develop a web-based game based on their current game, Draw It or Lose It, which exists as an Android-only application. The Gaming Room wants to allow multiple clients (end users) and platforms (web browsers) to run the application.

The clients requirements are as follows:

* A game will 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 to see if 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](#_2et92p0)

We will need to be able to either compress or manipulate the size of the library of images so that we can host them on our server. This will solve our main issue of the fact we are holding hundreds or thousands of images with many different users on a server of unknown size, in situations like this, memory management is key.

We will need to add an administrative interface to protect our clients' and users' information. This will solve our issue of not having the android built-in authentication systems thus we will have to include an authentication system with encryption features.

We are going to need a development environment and test environment for different platforms as our application was only developed for android based platforms.

## [Domain Model](#_8h2ehzxfam4o)

Game, Team, and Player objects all inherit from Entity. This allows them to have unique id’s and names as per our clients' requirements. GameService, Game, Team, and Player all have an Aggregation relationship. This allows GameService to run multiple games with multiple teams with multiple players assigned to each team as per our client's requirements. The singleton pattern allows only one instance of the GameService to reside in memory as per our client's request.



## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The main advantage of Mac is its UI and community. With these two things it will be very easy to learn how to set up a application. The main downside is that you have far less options to actually host on Mac rather than Linux or Windows, therefore price may fluctuate far more do to lack of competition. As for security, I would say it is incredibly easy to be secure on and is not something to focus on with this OS.  Other downsides include:  Very difficult to develop VM’s on (I have not been able to do it).  Requires licensing to use. | While the learning curve for this OS is very high I can recommend this OS because of how many programs have been successfully deployed with this OS. Security on this website is entirely reliant on the development team, Linux is known for providing very little in most aspects and security is definitely one of them. The positive side of this however, you will not have to deal with licensing as it is open source.  Another outside is how developing VM’s work on this OS.  It is very easy to set up and test VM’s on. And is in my opinion the main draw of it. | Very easy to set up and has many resources to host a web-based software application (for example I have used Go-Daddy to host my software online in the past). Security on this platform is a happy medium, it is not especially secure or insecure and depends mainly on the development team.  Another outside is how developing VM’s work on this OS.  It is very easy to set up and test VM’s on.  The biggest downside in my opinion is that you have to pay to use this OS as it is a licensing based company.  Fun fact about this platform though is that it is the only one listed that is not UNIX based, I do not know if this is a good or bad thing, but a fun fact none the less. | Mobile Devices are specially set up to connect to servers from other OS so while it cannot host itself nearly as effectively as other options it is very good at connecting to other servers.  From what I also understand about mobile devices, it is completely impossible to access kernels and develop VM’s on it. On a positive note however, they are incredibly secure. However, android is open source, which may make up for it. |
| **Client Side** | While being technically the most restrictive in terms of browsers and services, the options available to clients are all of high quality and all users of it are, on average, more knowledgeable about the platform than the other 3 platforms. On top of this, the services provided are completely unique to this platform and some of them (like Adobe photoshop for example) are the best in their respective fields. | While there is nothing unique about Linux that would help the clients it is able to use all 3rd party services and browsers. This includes my favorite browser, Brave, so there are quality products available in it for clients.  HOWEVER,  This is easily the smallest of the 4 platforms in terms of users and I do not recommend spending to much time on this one as it has the lowest number of clients. | Windows has the best in terms of amount of different browsers and services available to it. While this should make it a lock for the best platform client side it sadly comes with Microsoft edge which cannot be deleted without breaking the OS and will do nothing but waste time and processing power. | The most unique thing is that the browsers available are limited by the phone that was bought (like an Apple phone has different browsers available than an android). This means that technically there is every type of browser available on all phones in general, but each client is incredibly limited.  The other unique aspect of this platform is that it is without a doubt the most popular as it is the most cheap and easy to use, therefore, most profit possible for our game. |
| **Development Tools** | Mac is very unique in where it comes with custom IDE’s and deployment for it and is why you would get the OS itself. It is capable of using every language and the development tools provided do not favor any one in particular.  The main upside in my opinion is that the development tools for mac are also excellent at developing for the IPHONE and is the best for developing for mobile devices. | Linux does not have much in the ways for IDE’s and special tools from Linux to use, however, the ones made by 3rd parties are great to use to deploy programs on other platforms. Other benefits include: once you learn how to use linux it is without a doubt the best enviroment to test games for other platforms due its ease of creating VM’s. | To begin, Windows technically has all the IDE’s that Linux has plus visual studios however, it is definitely best to develop using either C# or c++ on this platform as Windows offers no support for things like Java or Python and it is admittedly a pain to use either of these two languages on versus on mac or Linux. | Almost every deployment software or IDE has special tools for deploying on mobile devices. They are specified to the left. |

## 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**: After working on this project, I would highly recommend using the Windows operating system. The two reasons I have for this operating system are the following: Windows is used on both PCs and consoles making it the best for a video game; The second reason is ease of use, Windows has many libraries to both, host websites (think of things like Microsoft Azure, a service which hosts servers easily and relatively cheaply) and many libraries that have been used to create video games. For example, when I made my first game, I only was able to find useful libraries on Windows and thus I developed my game on it.
2. **Operating Systems Architectures**: For the architectural details of Windows operating systems, I will start with how hosting works on this operating system. Basically, you create your program on an IDE of your choice (preferably something like visual studios), and then you can host it on a service like Azure (which provides libraries for hosting web applications) or a 3rd party service like GoDaddy (which hosts on a Windows platform). As for the libraries to create games, it is well-known by developers that Windows provides many libraries that make programming a lot easier, this is shown by the two most popular game engines (Unreal and Unity) using windows libraries to run.
3. **Storage Management**: For a storage management system, you can use a service like visual studios which provides a built-in SQL management system. This is an excellent way to organize and manage all user data for your program. While this service is available on all OSs on Windows, it is simply easier to use (in my opinion, I know my uncle prefers MAC and he manages SQL databases for a living, but I think windows is still better due to it being attached to the IDE that runs the program).
4. **Memory Management**: Memory management in Windows is very dependent on the IDE you use. For example, since this program was originally made on Eclipse, we would have to make our own management system in our code, unless we use a 3rd party service that handles it (to mention GoDaddy again I know they have their own memory management system that they provide to their customers). If you are not using that service however, you will need your own custom server set up to handle all memory issues.
5. The other possible way to handle this however is cheaper but I do not recommend it. You could instead host nothing yourself and make your customers download all data themselves and have to deal with this memory management issue themselves. This would be done by making the customer download all pictures, and data that the program uses and have your program call from their device rather than a server.
6. **Distributed Systems and Networks**: This is a two part answer so let me break this up into two paragraphs.
   1. Server programming stage: The server should check the users operating system and call the correct game version (which I will get into in part two) and allow the user to stream the game version from the server that corresponds to their operating system.
   2. Game programming stage: In game development, after completing the first base version of the game (hopefully designed for Windows for reasons I described in the first question) the code should be copied and refactored to be compatible with all platforms that it will be deployed on. Upload all versions on the server to be called.
7. **Security**: Basic security fundamentals do not change from platform to platform, the most that the different platforms will affect security is that the developers will have to call different libraries and use different file paths to extract data from securely. HOWEVER, we still need to define what we need to do for a program like this.
   1. Encryption: All user data must be encrypted where it is both stored, and user data sent to the server must also be encrypted to avoid any possible data leaks. I highly recommend using SHA-256 as it is very secure and avoids collisions.
   2. Manage dependencies: All dependencies should be analyzed by the developers and there should be no security vulnerabilities found in them if they are going to be deployed on the server. On top this, we should use as little 3rd party dependencies as possible in order to decrease risk of our dependencies turning into security flaws (since we do not secure 3rd party dependencies, we cannot immediately find security problems with them therefore they are incredibly risky and should be avoided).