

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 | 1/24/2021 | Jason Marcil | Added information to the Executive Summary, Design Constraints, and Domain Model Sections. |
| 1.0 | 2/7/2021 | Jason Marcil | Filled out the evaluation table. |
| 1.0 | 2/17/2021 | Jason Marcil | Updated the evaluation table. |
| 1.0 | 2/18/2021 | Jason Marcil | Made recommendations. |

**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 wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. The game is similar to the 1980s game show, *Win, Lose or Draw.* Each game will have one or more teams playing at a time with multiple players assigned to each team. Game and team names must be unique and will be checked to see if names are already in use. Finally, only one instance of a game can exist in memory at any given time, which can be done by assigning unique identifiers to each instance of a game, team name, or player.

## [Design Constraints](#_2et92p0)

* Need enough storage on the server for large library of stock drawings.
* Need to develop an administrative application to manage games, teams, and players.
* Need a mechanism for players to log in with a username (and possibly a password) to allow the application to differentiate players from each other.
* Need to ensure network connectivity is stable for each player in a game.

## [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)

The Entity class is the super class from which the Game, Team, and Player classes inherit. GameService, Game, Team, and Entity display encapsulation due to the attributes being private and are only accessible through the public methods via getters and setters. In the Entity class, the constructor is polymorphic because there are two constructors with the same name (Entity() and Entity(id: long, name: String)). The GameService class references the Game class, and the Game class references the Team class, and the Team class has a reference to the Player class. These references between the classes is done via aggregation. “0…\*” indicates that there can be zero to many instances of each class. The execution of our program takes place in the ProgramDriver class and it uses the SingletonTester class to run one game at a time with multiple teams and multiple players. Teams and players are in the form of a list which the program parses through to check the uniqueness of team and player names.

****

## [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** | Advantages:  Great option if network is made up of primarily Mac clients. Ease of use due to user friendly interface. User commands are flexible for server configuration. Overall, it’s easier to create features on MacOS than to do the same on Windows or Linux.  Disadvantages:  Price tends to be higher. Runs on a small set of hardware compared to Linux and Windows. Software that is not as widely used is less likely to work on MacOS unless using virtual machine. | Advantages:  Powerful, stable, reliable, and easy to use. Has a command shell for server configuration and accessibility. Price tends to be more reasonable. Linux servers rarely need to be restarted and can run for years without error.  Disadvantages:  Platform can be difficult to navigate for new users. | Web hosting that locates user file on a web server that uses windows OS.  Advantages:  Friendly user interface. Plenty of choices for software that can be limiting on Mac and Linux. Supports hypertext mark-up language.  Disadvantages:  Windows server side tends to be more expensive. | Advantages:  Cheaper option. Better if server is in one place.  Disadvantages:  Specifications vary from user to user due to differences in every model device. Need to find a provider that will host the app. Requires specific levels of expertise. Can be a lengthy process. |
| **Client Side** | Advantages:  Known for ease of use. Can use bootcamp to run Windows which is great for building and testing across the two platforms. Virtualization can be used to run Linux.  Disadvantages:  Cost can be higher and requires you to buy Apple products. Working across platforms can be more cumbersome. | Advantages:  Usually considered to be one of the best operating systems around by software programmers.  Most software is open source. Virtual machine can be used to run MacOS. Very easy to change according to needs. Fast and secure.  Disadvantages:  Learning curve is the greatest for new Linux users. Lack of hardware drivers. A variety of applications that work on MacOS and Windows will not work on Linux. | Advantages:  Typically requires the least amount of time and expertise. Cost is like Mac but usually slightly less expensive. Great for developing native windows apps. PC’s can run Linux in several ways including virtual machine, Cygwin, Ubuntu.  Disadvantages:  Not as easy to run Mac versions of apps, so using bootcamp on a Mac may be the preferred method for developing apps for both platforms more efficiently. It is easier to switch back and forth between Windows and MacOS this way. | Designing specifically for mobile devices is less ideal due to differing screen sizes ranging from phones to tablets. Even the largest screens are still smaller than most laptops. Because of this, mobile UI’s have been specifically designed for the smaller form factors, so it will be cumbersome to scale for other screen sizes. Mobile screen sizes also force us to get more creative with how to fit all features onto the screen. |
| **Development Tools** | Objective-C is the primarily used language for MacOS programming. Apple recommends Xcode for MacOS app development, but there are plenty of other IDE’s that can be used as well. | Linux supports most programming languages. Languages include Python, Java, JavaScript, C, C++, HTML, PHP, Perl, Ruby. Issues can always come up, but overall, it should well no matter which language is used. Popular IDE’s for Linux include Eclipse and Sublime. | C++ is the most popular language for developing Windows apps, and Windows uses C++ to build most of its applications. If you are using Visual Studio or VSC, you will have a better experience with Windows. | Swift would be the best language to use for iOS. Xcode would make the most sense since iOS is a Mac product.  Java is the best for Android as it is the official language for Android App Development. Popular IDE’s include Eclipse, Visual Studio, Netbeans, IntelliJ. |

## 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**: My recommendation for an operating platform that will allow The Gaming Room to expand Draw It or Lose it to other computing environments would be to use Windows. A Mac OS server will cater mostly to mac users. Linux may also work, but Windows is more ubiquitous and if the goal is to expand the player base, Windows is a more widely used platform especially for gamers.
2. **Operating Systems Architectures**: Windows is my recommendation because it can run applications and programs in both “user” mode and “protected kernel mode.” Using Windows Server will provide applications to use a GUI that is available to all windows-based applications. Windows also offers PowerShell for direct maintenance of the machine and allows for the use of a kernel mode to access a wide range of system resources, including memory and file management which allow for the user to manipulate the memory to spec. Also, multi-processor scheduling will assign percentages of the processing power to tasks from a ready state to a running state. This allows for optimal performance of the computer.
3. **Storage Management**: Since this is a web-based application, most of the data for the application should be stored on the server side. Most client systems will only store temporary files, pulling them as needed from the servers and accessing them locally. This will limit the server traffic by limiting the number of requests for pieces of data that can be stored locally. Windows server offers Storage Sense, which allows for user to easily manage the locations of files and applications, as well as remove temporary files that may take up unnecessary space.
4. **Memory Management**: On the server side, memory should be limited to handling application data for clients, such as the transferal of image files, synchronization information, etc. Memory on the server side should be limited to ensure that the application runs quickly. Memory on the client side should also be limited to focus on functionality such as user interface and rendering. Implementation of how memory works for each operating system will need to be designed differently since they each handle memory differently. Windows 10 Server uses RAM, physical and virtual spaces. The system will move pages of virtual address space to the hard drive to free RAM for additional use. This will help to limit the amount of memory that is used.
5. **Distributed Systems and Networks**: Distributed systems and networks not only offer ways for the application to grow as more clients use the application, it also offers protection and redundancy measures. Cloud servers would be the recommended solution for this application because ownership of physical servers would entail maintenance costs and general upkeep. If we have a power outage or physical servers fail for any reason, getting the application back online will be a bigger hassle and downtime could be significant. Cloud architecture removes these concerns because we will not be responsible for troubleshooting problems. If a cloud server goes down, the company that runs those servers surely will have backup plans that will minimize downtime.
6. **Security**: Security will be important not only for user safety, but also for building trust in the application. Username/password security and possibly multi-factor security should be considered to protect user information. An SSL certificate must be implemented on the application as well so that user information is secure and encrypted. This should also help the application rank better in search engines as well, which could contribute to expanding the user base of the application. On the server side, the principle of limited privilege should be implemented for anyone that will need direct server access to ensure that sensitive server-side systems are not breached. When talking about Windows specifically, there are built-in measures that protect against virus, worms, and malware in the form of Windows Defender. This included service receives definition updates on a daily basis to ensure that that we have the most up to date protection. There is also account and browser protection included in Windows Defender as well.