

Draw It or Lose It!

# **CS 230 Project Software Design Template**

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 |
| --- | --- | --- | --- |
| 3.0 | 10/18/20 | Patrick Hensley | 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)

Creative Technology Solutions would like to develop their game “Draw It or Lose It” into a web-based game. It is currently only in the form of an Android app.

## [Design Constraints](#_2et92p0)

* Making the game look and feel exactly like the app, but in a larger formatted web-based game will be a constraint. The user needs to know they are playing the same game if they are going from one to the other.
* Making sure the game can be played by one team or more than one team.
* Making sure each team has more than one player on it.
* Each team name being unique.
* Only one instance of the game in memory at a time.

## [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)

Entity is the domain with Game, Team and Player being the clients. This means, the Entity class will have information shared to the three client classes from the one domain. Within the clients, the Player class is not referencing anything but the domain. The Team class references Player, Game references Team and GameService references Game.

****

## [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** | Moderately ease to configure and access the server. | It is a more programmer-based operating system, but won’t have as many tools or resources as Windows of Mac. | Most widely popular so there are more tools and resources available. In the same way, also is more preyed upon because it’s so common. | Its is a more simplistic design due to the size of the devices. It can have a more common domain but is pretty specific to only these devices. |
| **Client Side** | It will take a little more expertise that Windows because it’s a little stricter with what is allowed. Cost will be comparable to Windows. | This will take more expertise and time. It will cost the least. | Least amount of expertise needed. The cost will be comparable to mac. It will be a bit quicker than mac. | This requires a bit more expertise but is more versatile. Updating and making changes on the fly is a lot easier. It is more cost effective because you can start programming without a strict plan and make changes as they arise. |
| **Development Tools** | Mac is typically programmed in objective C which is the Mac OS native language. C, C++, and C# are also used. Mac OS libraries (frameworks) has an objective C interface, so, it’s the most compatible for use.  The other languages are compatible as well, HTML, Python, Java, etc.  There are a wide variety of IDEs that will work with Mac, including, but not limited to, Eclipse and Visual Studios. | Linux is typically a C/C++ environment, but is also not limited to that. The Linux libraries are usually standard C libraries.  Python and Java are also popular within Linux. Eclipse and Geany are popular IDEs for Linux. | The Windows language kernel is mostly written in C. C++, C# and Javascript are all popular languages in Windows as well. There are so many choices when it comes to IDEs. Visual Studio and Eclipse are popular choices. | Java is popular when writing for Android devices. Android Studio and Eclipse are IDEs that can be used to create Android programs.  Swift, or objective C is popular for the Apple mobile devices. The Apple libraries are written in Swift. |

## 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 use a Windows C/C++ for the server side of the program because it can be written and executed easily with the 3 non-mobile OS platforms.
2. **Operating Systems Architectures**: Having one main server platform and having the client’s reference to it is a clean and easy way to minimize programming time and also allow for easier updating and bug fixed. Instead of having to change each platform individually, most changes can be made to the one server and cascaded to all platforms.
3. **Storage Management**: The Gaming Room should store its data in a sorted and organized manner as to not take up excess space, while making it quicker and easier for the game or advertisers to search and find the data relevant to what they are doing.
4. **Memory Management**: Using sorting and search algorithms to best make use of the memory will be key for a game using larger picture files. Being able to use algorithms to quick get the data will make the game feel better to the user.
5. **Distributed Systems and Networks**: Having the main server to be able to have the client classes call back to makes it easier to write one main portion of code to refer to. This leaves all platforms out if there is a server outage. However, it will separate the rest of the platforms if there is a client outage. If the Mac client goes down, but the main domain serve is still active, the others will still continue running as they are not dependent on the Mac client to run.
6. **Security**: Having log in credentials will not only allow you to keep track of the users, it will also allow for the users to be assigned a permission level. The general population will be allowed to access the game and its features, without allowing them deeper into the program and won’t allow them to make changes. Certain programmers and testers should have elevated to privileges to allow for testing and programming while playing the game.