

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 | 02/17===/21 | Tom Czubat | Information was added to the design document. |

**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 a web-based-game that can run on multiple platforms based on its popular Android game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show from the 1980s. The platforms game is already developed on Android but the company wants expand this game using various software patterns in a distributed environment. They want to expand this game to Windows, Mac, Linux and mobile (iOS) operating systems. This game will have one or more teams involved and each team will consist of multiple players. The game and team names have to be unique to allow users to check to see if a name is in use when choosing a team name. Also, only one instance of the game can exist in memory at any given time so we must create unique identifiers for each instance of a game, team or player.

## [Design Constraints](#_2et92p0)

* Game needs to run on multiple platforms(Windows, Mac, Linux, mobile)
* Multiple teams consisting of multiple team members
* Checks for unique team/game names must be present
* Only one instance of the game must be allowed at any one time
* Needs enough storage on the server for the large library of stock drawings
* Need mechanism for managing security/login/identification to allow the application to differentiate each player from all other players
* The web application will require player logins or an alternative method for players to identify themselves to the game.

These constraints are what the game requires. In terms of development, The Gaming Room wants this game to be run on multiple platforms, so we would need more developers to build the code for each of the required platforms. We also need to make sure we can distinguish unique players, teams, and games so a login system can help with this.

## [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)

To describe the UML diagram, I first looked at the top where the Entity class creates a relationship with between the Player, Team, and Game classes. These three classes inherit methods and attributes from the Entity class. These classes will inherit the **id** and **name** attributes and the **getId**, **getName**, and **toString** methods. This means that Entity is a superclass of the Player, Team and Game classes. When we look at these three classes we see that there is a “has a” relationship which is called aggregation. This is because Team “has a” Player and Game “has a” Team. We see that the GameService class has a 0 to many relationship with the Game class meaning there can be zero games or more than one game. We also see that the ProgramDriver class uses the SingletonTester class.

****

## [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** | Mac has the option of Mac OS X server available for use. According to Apple’s website, Mac OS X Server is only $19.99, so it would be inexpensive to implement. It is important to note that Mac is not as popular as the options for Windows or Linux for performing these tasks. | Linux is a good option for the server side because it has many distributions that have server capabilities. Linux Server would be low-cost and open source so it has many resources to help you. Many people are not familiar with using Linux but it has the largest percentage of servers in the world. | Windows offers Windows Server. Windows Server could have a higher cost to implement but it is a solid option. Windows is likely the most used operating system, Many users/developers are comfortable with using Windows. | Given that mobile devices do not necessarily have the power that computers do, hosting a fully-fledged server on one may not be the best option compared to computers. Running servers on mobile devices is the most advantageous in terms of cost, as there is little to none to get one started. |
| **Client Side** | Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Mac would need less time and someone who does not have as much experience with Mac would need more time. | Cost would be low with Linux, because it is open source. Maximum time and experience would be necessary, as Linux is not commonly used and you would need people who are experienced with Linux and allow them time to work, as Linux can be difficult even for someone with experience. | Cost would be similar to Mac since these operating systems are not open source. Time would depend on expertise, as someone who has experience with Windows would need less time and someone who does not have as much experience with Windows would need more time. More people are familiar with Windows than Linux. | Cost would not be too much of an issue with Mobile devices. Experience may not be too much of an issue, as mobile devices can be easier to work with. More time would be needed, as there are multiple operating systems and multiple mobile devices that would need to be worked on. |
| **Development Tools** | Swift and Objective-C are the languages for writing applications for Mac. There are multiple IDEs that can be used for Swift and Objective-C, such as Atom and Xcode. The team working on the Mac version of the game would also be working on the iPhone/iOS version of the game, since these versions would be similar. Development tools usually wouldn’t cost anything, as they are usually downloadable directly from their websites. | Eclipse and Atom are commonly used IDEs on Linux. Eclipse is primarily used for Java, although it can support other languages like C++. Atom can also be used for developing in multiple languages. The team working on the Linux version of the game would also be able to work on the Windows and Android versions of the game. Development tools usually wouldn’t cost anything, as they are usually downloadable directly from their websites | Eclipse and Visual Studio are popular IDEs for Windows. Visual Studio can be used for developing in HTML, C# and JavaScript among others when using plugins. Eclipse is primarily used for Java, although it can support other languages like C++. The team working on the Windows version of the game would also be able to work on the Linux and Android versions of the game. Development tools usually wouldn’t cost anything, as they are usually downloadable directly from their websites | For iPhones, the development tools are like those for Mac, and iOS apps are typically written in Swift and Objective-C, though iOS and macOS are different in terms of appearance and functionality. Android apps are written using Java, Kotlin and C++. Android apps are usually written in Android Studio. Developers for iPhone would probably come from a team of Mac, and Android developers would come from the team developing Windows and Linux. Development tools usually wouldn’t cost anything, as they are usually downloadable directly from their websites |

## 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 to start the project with Windows Operating System after considering the level of expertise required and the fact that it’s relatively low cost. The Windows operating system also has many great IDEs to choose from which will make the development process faster and easier. Another benefit is that most people have used and are comfortable with using Windows.
2. **Operating Systems Architectures**: The Windows architecture allows for applications to utilize the platform’s kernel processes without directly affecting those processes. The applications can utilize the power of Windows to have a Graphical User Interface (GUI) set up, access to memory and other important processes that make the application without inadvertently affecting the processes that make the operating platform work.
3. **Storage Management**: Windows offers you Disk Management and Storage Sense built into the operating platform itself which is very useful. Windows also has a Disk Cleanup tool that can be used. Disk Management is a Windows system utility that is mainly used for advanced storage tasks and Disk Cleanup and Storage Sense are used to help maintain the storage on the system by deleting unnecessary files that are taking up storage space.
4. **Memory Management**: Windows has Memory Management built in as a system utility. We would need to build the database for all the game’s image files so that they can be easily accessed by the game. The memory allocation allows for easy storage of pictures outside of the default picture folder. This would allow you to keep your whole project together in a more secure area on your computer.
5. **Distributed Systems and Networks**: Even though I am recommending starting with a Windows-based development first, I understand that eventually we have to cover all of the platform regardless of their operating system or devices. There exists a cross-platform game development tool that could be a good solution for this project. Through my research, I have found that a commonly used and cost-effective cross-platform game engines is called “Unity” which can support Windows, Linux, Android, and iOS. Selecting the right server for the job is also important. Since many users will use the game at the same time, it’s important to pick a good network and server which can handle the high volume at all times.
6. **Security**: Windows has its own software that handles security. Although this built-in program (Windows Defender) may perform well, there is other software available if you are willing to purchase those programs to enhance the security level of the entire system since we will be dealing with sensitive information with the user’s data. McAfee and Norton Antivirus programs can be helpful programs which will protect us from malicious contact with malware or viruses which may further threat our sensitive information of our users and our company as well. We would need to make sure that we encrypt all of the data that is being sent back and forth.