

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/08/2023 | Tammy Hartline | Updated Summary, Requirements, Constraints, and other pertinent information needed for this project. |
| 1.1 | 03/28/2023 | Tammy Hartline | Updated recommendations and system architecture view. |
|  |  |  |  |

**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 client is seeking to make their current game, “Draw It or Lose It,” currently an Android based application only, into a multi-platform web-based game. They want the game to be based on the current Android application and are seeking help with environment setup and streamlining their development. The game will need to be developed using a cross-platform. Each game and team name needs to be unique, and the game should only allow one instance of the game to exist in memory at a time. Each team will have more than one player assigned per team, and the game will need the ability to have multiple or single teams playing per game.

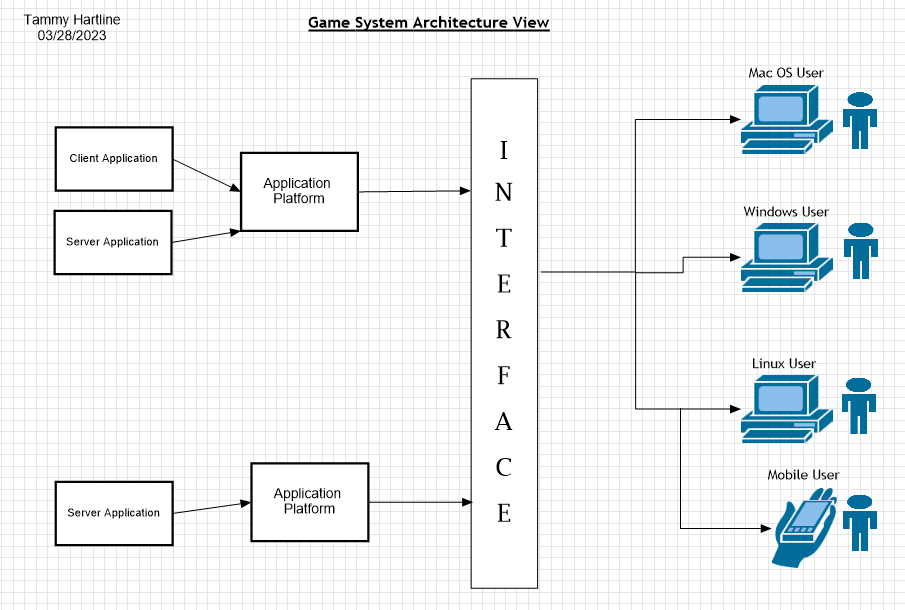
## Requirements

The client requires the game to have the ability to have one or more teams involved, and each team to be assigned multiple players. Only one instance of the game should exist at a time. They request this be accomplished by creating unique identifiers for each game instance, team, or player. The user needs the ability to check whether a name is in use when choosing a team name, therefore, each game and team name should be unique.

## [Design Constraints](#_2et92p0)

The main constraint for this project are developing the game on multiple platforms. Without team members familiar with cross-platform development, it will require multiple teams developing different products based on environment expertise. The other constraints are ensuring that the development meets all client requirements on each platform.

## [System Architecture View](#_ilbxbyevv6b6)



## [Domain Model](#_8h2ehzxfam4o)

The Entity Class is the parent (super) class of the Game, Team, and Player classes. This means that Game, Team, and Player class, as Entity’s child classes, will inherit Entity’s attributes, while each being assigned attributes of their own, that are separate to the parent class. The Game Service Class is used to ensure the client’s requirements are met, providing a single game instance at a time, unique team name (id), unique game name (id), and unique player name (id). Program Driver contains the main statement and uses the Singleton Tester class. The Game class contains a team list, Team class contains a Player list. The Player class does not contain a list, as it ensures that each player has a unique id, that can be assigned to a team. While a player can be on a team, and a team does have players, the player class does not contain or have a team or a game.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

03/28/2023

Updated information for each recommendation based on further learning. I also added the system architecture view to include the model I made. While I am not certain that I completed or created it correctly, I was able to do some research into what a system architecture view consists of and thought I would give it a try. As far as I have been able to gather thus far, the majority of this game could be coded in C++, as it appears to be the most popular language for most game developers. I was able to find information regarding servers, and cost-effective options, which include renting servers or using a community-based peer to peer network system server. There are multiple games that already implement the rented servers, or P2P, such as Minecraft and RUST, making it a considerable recommendation. I am not familiar enough with them, however, to fully recommend this as option, given I am not certain of the security risks they impose.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | A huge advantage MacOS has, is its ability to run MacOS, Windows, and Linux apps side-by-side. Another advantage that is often overlooked with MacOS, is its consistency. MacOS and its system has remained consistent, making it easy to operate for those used to the OS. The cons to MacOS are mainly its cost and limitations to hardware options when compared with Windows or Linux. | Main advantages for Linux is its wide range of distribution choices, and it being open sourced with several free or cheap options. Linux is well known for its versatility in servers and embedded solutions due to ease of customization. Linux also has superior security protocols, as compared to Windows or MacOS. The disadvantages are lack of choices for pre-built machines, and file format compatibility issues. | The main advantage for windows is for corporate users, Windows based authentication integrates with Active Directory based corporate servers, right out of the box, no add-ons or additional expense needed. The disadvantages of Windows is lack of mobility development, and its security protocols are lacking which can lead to malware, spyware, and ransomware. | Using a mobile device to host web application can be beneficial if the users are limited and the application is not a large or complicated one. There are several low-cost web server applications available to android users which also works as an advantage for using mobile devices to host web applications. The disadvantages are that in most scenarios, the mobile based hosting options are cloud based, which can leave companies more open and accessible to hackers. Mobile devices are also typically as a whole, more vulnerable than PC’s. |
| **Client Side** | The pros for MacOS are its ease of use once skills and understanding are developed. The cons are its availability outside of Apple OS. It is not accessible unless on an Apple device, which can limit developers, whom may have the knowledge and training to develop MacOS programs, but do not operate and develop from a Mac device. | Pros are affordability, and the control over your developments offered by Linux. Because Linux is a free and open-sourced OS, it is usually simpler to maintain upkeep and thus takes less time for maintenance. There are also cons to each of the pros mentioned. With Linux being open-sourced, and user controlled, security could be an issue that is faced alone, when compared to Window’s or MacOS and their technical support services offered for their products. | Pros with Windows is availability, and at several price ranges that can be customized based on project usage and need. They offer technical support and advanced security options when compared with Linux. Cons, are that you need someone with Windows OS expertise and cost can be added for specific features you may want to include. | While there are many applications and tools available for mobile devices, they usually lack the accessibility and full features that are offered on a PC. Pros are that they are easily available, at many price points. The cons, are that mobile devices have MANY different OS and usually each device requires or is geared toward a specific OS, making it inoperable with other systems. |
| **Development Tools** | MacOS uses the programming language, Swift. The main tool available and readily usable for MacOS and IOS developers is Xcode and Xcode Cloud. Xcode cloud is a service that was developed specifically for Apple developers, to allow teams a faster and simpler way to build, test and deploy apps more efficiently. | Linux has an enormous amount of development tools to choose from. Because there are so many, I am going to go over one option. Docker, which is used to offer consistent dev environment, build cross-platform applications and easy deployment. Docker Hub is also used to help skip the dev environment setup and allows users to dive right into development. | Windows was written primarily in C, with some parts in assembly language. Perhaps the most known and used IDE for Windows is Visual Studio. It is used for an IDE, as well as code editor and in some instances, a source/version control. There are many tools offered by Visual Studios, although I must admit, I was slow to see their benefits, I now primarily use it for development, both for school and work. | For most mobile applications, the go-to language is Java. Its object-oriented features make it an optimum choice among many developers creating mobile applications. Although, Python and C++, or other variants of C are also utilized for mobile development projects, particularly gaming. While there are countless IDE’s that support mobile app development, the most popular are VSCode, IntelliJ IDEA, and Eclipse. Personally, I am not a fan of Eclipse, as I find it a bit to cluttered, and usually use Visual Studio with Xamarin, allowing me to develop cross-platform native applications. |

## 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**: Windows is the recommended operating system for this project. Windows is the optimal choice due to its ability to integrate with their current Android application of “Draw It or Lose It.” With Windows large userbase, making up around 90% of OS for users worldwide, will offer a wider range of developers, skillsets, and tools that are available to get the project completed for a cross-platform application at the best price point.
2. **Operating Systems Architectures**: For the system architecture, I suggest using a hybrid architecture. This will pull in all the best features from the monolithic, layered, and micro-kernel architectures. One operating system made with a hybrid architecture is Microsoft Windows NT kernel. The advantage of choosing hybrid over one of the other singular architectures is the allowance this provides to their services respectively. Because of its layered approach, it is also easy to manage. The number of layers is lesser as well. The hybrid architecture will provide improved security and protection versus using just monolithic, layered, or micro-kernel.
3. **Storage Management**: Server-based storage will allow central access to files, adding functionality, and failover clustering. It also provides redundancy, automated backups, and faster, optimized performance. However, one should have a cloud storage available and use often for backups, managing data loss, and the ability to access previous versions.
4. **Memory Management**: Windows offers a variety of storage and memory management options. Such as Azure Storage. The OS includes virtual and physical address space for memory allocation. There is also the consideration of OneDrive, Visual Studio or even Azure Cloud services that can be used to store and manage versions. Using industry standard best practices when developing the program will help with runtime and memory allocation and management, especially when considering storage container types, such as those that do not allow memory expansion, version those that do.
5. **Distributed Systems and Networks**: Using a cross-platform development environment will help limit the need for multiple expertise and make for a smoother application development process. One of these options is “Develop 4.” Another option is pre-packaged libraries that allow developers to develop a program, implement the library, which then allows the program to be accessible on a variety of platforms and operating environments. To address connectivity or outage issues, I suggest ensuring the servers are built with the capacity that meets the client’s needs based on forecasting game usage/users once the application is launched to the new environments.
6. **Security**: With ways to steal user data being a constant and ongoing threat, I would recommend special focus and attention to security protocols. One service that offers protection for PC, Mac, Linux, Android and IOS devices is Aura. While this will come at a cost, using standard security that comes with an OS is not recommended, as an additional layer of protection is needed to keep systems safe and operating optimally. Aura also offers 24/7 U.S. based customer support, which is a huge pro for this service. Another consideration is ensuring company employees understand the importance of password security protocols and implementing an authentication process in tandem with authorization. I also recommend standardizing authorization protocols, that include limiting access to information on an *as needed* basis only, versus allowing the same level of access to everyone or allowing everyone access to everything.