

Draw-It or Lose-It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 01/20/23 | Emilio Hernandez | I am going the explain the key points necessary to convert the gameplay of Draw It or Lose It to a web-based game. |

**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 first design challenge that I will have to face is the desire to only unique team names and to set up a team with a limited number of players to stay equal. I can overcome this with creating a library of team names that when added will check to see if that team name already exists or does to not make all team names unique. Because the game can only have one memory at any given time means that it will have to reset each play session to gather a new random set of stock images and ask for new team names.

## Requirements

The technical requirements of the project are that the game needs to be like the app version but capable of running on a webpage. As my client I will provide the necessary tools needed to complete the project and will provide progress of the project.

## [Design Constraints](#_2et92p0)

The major design constraint is that only one instance of the game can exist in memory at any given time. This limits a player from playing with more then one team in more then one game. The other limitations is the amount of stock photos that will be in the library and because there are a limited amount there are bound to be repeated photos.

## [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 UML diagram below shows the relationship of the multiple classes to each other. For the player it starts in the Player class this will get player ID and name from everyone. Next that information if transferred to the Team class which will create a list of players in a team and will create a unique team name and ID and add the player to a team. Next in the Game class the game ID and name are created and the teams are added. Above all these categories is the Entity class this is here to gather and store player IDs and team names to ensure they are not currently in use. The Entity class uses inheritance from the Game, Team, and Player classes with means the classes beneath it share programs from the Entity class. Finally, is the GameService class is the actual process of the game being played and scores being kept throughout the 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)

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** | The main language required for Max OS X is objective C. The Server would cost $499 for a limited 10-clients and $999 for unlimited clients. | Linux is a very popular choice because it tends to be a lot cheaper than Mac and Windows. The servers are most commonly used by cloud providers. | After looking into a Windows server, it would cost around $749.99 for the data center. The main supported language is C++. Windows would be more easily accessible because of its abundance. | The hardware for a mobile device is way more limited then other operating systems. They will require the use of a built-in house server so the cost would be different based on our personal web server. |
| **Client Side** | For clients when the program is using a Mac server the clients are required to use a Mac computer to run the program. The macOS SKD requires Objective C and SWIFT which a limited known code.  **For the Mac OS the most common web browser is Safari which is the preinstalled software. Safari is one of the most secure web browsers. Chrome is a great web browser that is installable on all operating systems.** | This is easier and uses the most common known programming languages such as Java and C++. This requires Linux platforms which as the least common use for clients.  **The most common used web browser on Linux is Firefox. This is a good secure browser but not used by most people. Chrome is the second-best option for Linux.** | The required languages for Windows are C# and .NET which are fairly common and used languages. This is the most used operating system across all clients making web-based game the most easily accessible.  **For Windows Microsoft Edge is the main Web Browser installed on all PCs and laptops. This is not as secure as most web browsers.** | Mobile devices require different languages based on the device that is being used. Android devices uses Java, which is very common, but iOS uses SWIFT which is not as popular. The biggest issue is that mobile devices are not meant for multi-users.  **For iOS devices safari is the main supported browser for their phones. For Androids Chrome is the most common browser used but it can change depending on the user.** |
| **Development Tools** | Mac requires Objective-C and SWIFT for development. XCode is used as the most common IDE which is $99 per year per developer. | Linux requires C/C++, Java, or Python for development. The IDEs’ used for Linux are for the most part are free such as PyCharm or Eclipse. | Windows uses C# and .NET for development. Microsoft Visual Studio is the most commonly used IDE for Windows. This IDE can very in price and get to as high as $250 per user per year. | Android devices use Java for development. Android Studio is the IDE and is free for all users. iOS uses Objective-C and SWIFT for development. This also requires Xcode which is $99 per year per user. |

## 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 using Windows as the operating platform because it is the most used operating system and can be writing in C++. This would make the operating system more secure and easier to develop because it is the most widely known and understood language.
2. **Operating Systems Architectures**: Because this is the most used operating system it comes with a great technical support team to solve almost all problems. It is has an easy to use and learn user interface that can be used by all members of the team to achieve the goal of the game. The interface can also be used to easily run multi development to test to software across other operating systems.
3. **Storage Management**: To manage storage we should use an inhouse protected data base to protect our code and date. This will need to be able to store all images along with the program to run the game. This data should not be accessible to the public.
4. **Memory Management**: Windows allows a set amount of memory to be used for a certain amount of time which would be the game length. This will need to allocate different parts of the storage to send and receive information from and to the users of the game. This will be used on the user’s side but getting information from the inhouse storage system.
5. **Distributed Systems and Networks**: The use of similar web browsers allows the users to be able to play on different operating systems. The game will send all players in a single session the same pictures at the same time. When users enter their inputs it will be taken by the memory and checked in the storage but this information will not be shared to the other users. The program can also run on cloud-based servers that will allow users access across all other operating systems.
6. **Security**: The security of other is the highest priority when creating anything that may ask for user information. When creating this we can use firewalls along with our encrypted storage management to protect the information we get from users. When users play the game extra information not necessary will be asked for but things like IP address are used when using the web and playing games, but our system will need to be aware of this and constantly work to prevent hackers from accessing this information.