

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

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

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/26/25 | Colin Timko | Added information about what the client wants out of the game. Also described four different platforms to develop on, compared and contrasted them, and gave a recommendation of what is the most prefered |

**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](#_35nkun2)

We want to create a game that works on many different platforms based on the game Draw it or Lose it which is currently only on Android. The game involves having players on teams guessing what the drawing is as it is rendered from stock images as clues.

## Requirements

* **Allow the game to be played on multiple platforms**
* **Give the teams many different pictures to allow for longer playtime**
* **Make sure team names are unique and cannot be copied**

## [Design Constraints](#_1ksv4uv)

There are a couple major constraints that come with this being developed on the web. One is that the developers want this game to be able to be accessed on many different platforms which can come with challenges to make it compatible. Another constraint is that the names of the teams have to be unique so we need to make sure we run checks to see if the team names are taken. We also need to make sure that the server is responsive as to give each team the images at the same time to prevent any advantages due to the image not loading properly or late.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

**The entity class is the base class that provides attributes like names and ids. The classes that inherit from the entity class are game, player, and team. We can see the relationship further using a “has a” mindset. For example the game has a team and the team has a player. The OOP principles used here are inheritance which we have already seen as the three classes inherit attributes from the base class.**

**"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](#_z337ya)

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** | One of the major disadvantages of using Mac is the limited amount of server side software. While the advantages are ist security on the Unix foundation it is also more flexible for changes. | Advantages for Linux is that it is open source so the cost is very low and it is preferred for web hosting. While the disadvantages are it can be complex to use and it can be hard to find services for web hosting that support Linux. | The advantages of windows is that it is the most popular choice for best hosting so there is a large amount of software available. One of the disadvantages is that it is more susceptible to viruses | Depending on the size of the app it can be very easy and cheap to launch. However since a phone is mobile it is more susceptible to bandwidth and network issues along with security risks |
| **Client Side** | There is a very limited number of languages able to be used due to being limited to Apple's tools. Cost can be somewhere in the middle of the other platforms. | Linux is one of the harder platforms to use especially compared to others like Windows and Mac. However being open source keeps the cost relatively lower than other platforms | You would need to know more common languages like C# using visual studio. | The cost of mobile can vary depending on how many platforms you want to create for. You would also need to know things like Swift for IOS or even cross platform languages like Flutter |
| **Development Tools** | Languages that are used are Swift, Objective-C and C++ on the IDE Xcode | Many IDEs used for Linux are Eclipse, KDevelop, and IntelliJ IDEA which include languages like C++, Java, Python, and many others | The most common IDE for windows is Visual Studio in which you can use languages like C#, C++, and CSS | For IOS you would use things like Swift and for Android it is Java and Koltin for both you can use React Native and Flutter |

## 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 believe the best platform to use the most common to many people which is Windows. The cost may not be the lowest but most people are familiar with Windows as it is used in most people's everyday life.
2. **Operating Systems Architectures**: Windows uses a layer design with the kernel, applications, and user mode. The hardware abstraction layer allows the kernel to interact with the hardware without knowing exactly what it is. User mode is what the user can access like programs.
3. **Storage Management**: Windows uses a disk based system which allows the user to manage and view their different harddrives and save certain applications and programs wherever they need to. Since windows is so common, finding and managing the storage and going through file explorer has become easier for more people. This is important because for this game we need to make sure we have a sufficient amount of pictures so that the game does not get old after a couple of rounds. We can also use Microsoft Azure to take a majority of the data needed to be stored.
4. **Memory Management**: Windows implements Kernel mode drivers which separates memory in to both virtual and physical. This makes sure that the memory is optimized especially since the game needs to access many photos for the game to work.
5. **Distributed Systems and Networks**: Windows has Azure which is a cloud based service and can take the brunt of the network load and distribute it to many different platforms. It will also help when there are many different games and many different players going on at the same time.
6. **Security**: Windows has built in security that scans the computer and its files for any type of discrepancies like Malware and viruses. We should also make sure that our game and its information is encrypted within the application itself.