

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 | 09-13-2021 | Jeremiah Schoenherr | Creation |

**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 is bringing their game “Draw It or Lose It” across platforms. The game is loosely based on a 1980s TV show called “Win, Lose or Draw” where teams compete to guess what is being drawn like the childhood game of charades. Except in this game, stock images are rendered as clues and the teams compete to guess the puzzle before time runs out. The Gaming Room staff doesn’t have the ability to set up an environment for the web-based version of the gaming application and has asked us to continue development.

## [Design Constraints](#_2et92p0)

1. Requires one or more teams to play.
2. Each team has multiple players.
3. Every game and team name must be unique to allow users to check whether a name is in-use or available to use.
4. Only one instance of the game can exist at any time.

## [System Architecture View](#_ilbxbyevv6b6)

N/A

## [Domain Model](#_8h2ehzxfam4o)

As shown below the Game, Team, and Player classes are all related to Entity the superclass. This means that the Game, Team, and Player classes all inherit from the Entity superclass. Carefully notice that Game has a relationship with Team, and Team has a relationship to Player. This is called aggregation when one class calls a reference to an instance in another class. GameService references to Games, then Games has a reference to Team, and Team references Player.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Open Source, meaning no licensing cost. However, tools are few are scarce. Support is available but not as readily available as Linux or Windows. Rare to find hosting. | Linux is an open source, meaning there is no licensing cost and most tools are free. There might be more of a learning curve, but the lack of cost can outweigh the extra time spent. | Proprietary, meaning there’s a paid licensing cost. Windows servers are more secure than Linux, and there is more support and tools readily available. | Widely portable, and highly popular. Mobile application run on the users device making them fast and responsive. However it is required to be downloaded and stored on said user’s device. Not suitable for hosting server-side. |
| **Client Side** | Mac as a mostly open sourced OS has its advantages, however it can be difficult to learn and there could be some licensing costs. | Linux comes with a big learning curve. It can be difficult to use, but the lack of licensing cost could greatly outweigh some extra man-hours learning the system. | Being a proprietary OS, there are a lot of options and support systems that can make using Windows easy. However, it comes at a pretty hefty cost that could make development go faster and easier. | Can be hosted on any OS, making it easy to extend a web-based application to mobile platforms. |
| **Development Tools** | XCode is the most used IDE for Mac applications. Swift is the most common language used. Being mostly open sourced there is little to no licensing fees. | Eclipse is the most popular IDE, using C language. Linux would be a good choice in development due to the open sourcing and free tools that can readily create web-applications. | On Windows, Visual Studio is the main IDE used. Using Visual Basic it would be very hard to develop web-based applications, but it is possible. There are licensing fees. | Android uses Android Studio and Java or Kotlin as language. IOS uses the same tools and language as Mac – XCode and Swift. Not suitable as a development environment. |

## 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**: My recommendation for the operating platform is Linux. There is a learning curve involved in the process, however being open-sourced would mean no licensing costs. There are plenty of tools available to complete Draw It or Lose It, and most of them are free. Since this project will go across platforms in the shape of a web browser game, using the cheapest option as far as the platform goes will save money for hosting and other variable costs.
2. **Operating Systems Architectures**: I would recommend x86, solely just because it is one of the most powerful architectures available. Intel and AMD both include x86 and can be 64bit if needed. More than 90% of systems today use x86, there is really no cost difference here to weigh-out another option that is less-efficient.
3. **Storage Management**: Linux can support multiple drives; I would recommend Solid State Drives over the conventional Hard Disk Drive and will most likely use sata unless Nonvolatile Memory Express options are available; they can be quite pricey though. Linux has a built-in partition tool that allows either a standard partition or an LVM partition. This can be helpful when it comes to designating storage for certain aspects of the game or system.
4. **Memory Management**: Draw It or Lose It will require a large library or database with a decent selection of stock photographs. Allocation will allow better management of this when it comes to storing the photos in a separate folder, this makes it easier to pull and use photographs in an IDE or in the game environment itself.
5. **Distributed Systems and Networks**: I found React Native to be a solid tool in achieving a cross-platform game. Using either JavaScript or C language and cross-coding to create Draw It or Lose It will enable React Native to interpret original code and covert it to the necessary native elements in less time. The company will need to budget for a massive server to withstand supporting a large consumer base and the heavy networking and power outages.
6. **Security**: Linux’s built-in security isn’t bad. It offers firewalls that use packet filters in the kernel, it offers UEFI secure booting firmware verification mechanism, and Linux kernel lockdown config option enhance the security options that are already available. This system is better than a stock Windows security system, but there is always room for improvement. I would recommend Linux’s stock security options as they are free and provide decent coverage.