

# 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 | 01/22/23 | Victor Gomez | Revised Executive summary, design constraints, domain model, evaluation, and recommendations |

**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 wants to make a game that is web-based, and that can run on multiple platforms. The game is already made as an android app. The game is named “Draw It or Lose It”. The game consists of four one-minute rounds and will render images from a library of stock drawings as clues. The drawings are rendered linearly and complete at the 30 second mark. If the guessing team does not guess the puzzle before the one-minute time expires then the other teams will be given one chance each to solve the puzzle with a fifteen second time limit.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* One design constraint is that they wish to make the app available on all different type of platforms. This could be a constrain because it could be costly, and take a lot of time.
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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 shows that Entity is the Parent to Game, Team, and Player this is a is-a relationship. They inherit from entity. GameService, Game, Team, and Player have association, and multiplicity with each other from zero or many.

**"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** | Great for mac clients. Flexible terminal commands | Open-source OS. Flexible terminal commands. Also, very cost effective. Linux is used more commonly for hosting services. | There are a lot of software for windows. Widely used everywhere. Virus prone. | Highly portable.  Cost effective.  Wider reach to everyone. |
| **Client Side** | Expertise would be moderate to high level.  Cost would be similar to windows.  Time would be similar to windows set up. | Expertise would be high level.  Cost would be the lowest of them all.  Time to set up would be long since it would be more custom. | Expertise would be low level.  Cost would be close to moderate cost.  Time would be low for the setup, easy to work with. | Expertise would be moderate to high level.  Cost would be moderate.  Time would be high because of difficulty setting up. |
| **Development Tools** | Some of the most common languages Swift, Java, Python, PHP, Ruby, HTML, CSS, and JavaScript  Mac IDE: Xcode | Some of the most common languages Swift, C, C++, Java, Python, PHP, Ruby, HTML, CSS, and JavaScript  Linux IDE: visual studio, eclipse | Some of the most common languages Swift, C, C++, Java, Python, PHP, Ruby, HTML, CSS, and JavaScript  Windows IDE: visual studio, eclipse | Some of the most common languages Swift, Java, Python, PHP, Ruby, HTML, CSS, and JavaScript  IDE Andriod and swift |

## 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 that The Gaming Room starts on windows. I would recommend this because cost is moderate but the expertise is low and time for set up is low. Also, more software catered to windows to work with.
2. **Operating Systems Architectures**: Windows uses a hybrid kernal. It is a mix between a microkernal and a monolithic kernel.
3. **Storage Management**: I would recommend a cloud-based storage for its back up capabilities and scalability. Also, it is somewhat cost effective.
4. **Memory Management**: There should be a database for the pictures used as clues for solving the puzzle.
5. **Distributed Systems and Networks**: The Gaming Room needs backup generators for the server so that there is no down time in the game because of power outages. Also, they need to be hard wired as there connections for the internet so there are no disruptions of service. The client needs strong enough hardware in order to keep up with the growing customers they may encounter. Some cross platform software that may be of use here is Codename One, Kivy, Qt, Flutter, NativeScript, Xamarin, Phonegap, Ionic, and React Native. These softwares may be capable of making a program that will allow the client to make just one iteration of the game and run on multiple platforms.
6. **Security**: The operating system comes with a built-in defender for security. Seeing that windows is very virus prone the recommendation would be to consider extra protections for sensitive data.