

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## 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 |
| --- | --- | --- | --- |
| 2.0 | 10/01/23 | Shayna Mitchell | Changes were made to this project. Please review the evaluation section of this document. |

**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)

This project is called The Gaming Room. This is a entertaining creation that will be based on the web. The idea is brought about by a game called Draw it or Lose it which can only be found on android devices. The entertaining aspect is that multiple people will be playing this game together and must be available on multiple platforms. The players will need to guess a picture before the time is up.

## Requirements

This game will require multiple platforms – which means it should be able to run on multiple operating systems. This allows more people to be able to play it whether that be online, windows os, android, or mac os. We will need to the skill and the knowledge to create this project on multiple operating systems.

## [Design Constraints](#_2et92p0)

This game will need to be able to support multiple players on multiple operating systems. Someone playing on a google pixel should properly be able to play on the same team as someone playing on an iphone. This raises the need for proper storage and ram for each of these players. What will the speed requirements be while playing? They will not have to run into the game crashing in the middle of it. Part of this process will be the ability to check which usernames have already been taken. This database aspect will be important to create a fun environment for all players.

## [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)

We use the UML diagram below to help us visualize the structure of our project all in one example. Our classes are all broken up and organized in a way that benefits the final project. Here we have one class that rules all of the other classes: Game, Team, and Player. These three classes inherit the capabilities of the Entity class. These are all listed inside of Entity. These classes reference eachother when you look at GameService in addition to Game, Team, and Player. The whole driver will be using the Singleton method to organize this project.

**"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** | Mac is easy to use and a nice user interface. | This operating system is not the easiest to use. | This option has a command prompt that will help the user at all times. | These can’t be as powerful as the others, but are an option. |
| **Client Side** | This option is not cheap. | Linux is open source and requires some work to feel comfortable using it. | This option is easier to learn and begin using. | Users tend to have a lot of broad options here for creativity. |
| **Development Tools** | Visual studios is popular here. HTML and Javascript. | HTML and CSS. Ruby and Python. | Eclipse and Pycharm. | C++ is typically used here and Python. |

In looking into our project further, each operating platform offers a server based deployment method where the website will be hosted. If the decision is made to use a linux server, there will be no cost to worry about. If the decision is made to use Windows or Mac OS a cost will be present. If windows os is used, it will require visual studio to be used which can support a few different languages. If linux os is to be used, eclipse is usually chosen which typically uses the language C. If mac os is to be used, xcode is typically used which uses the language swift. Developing for a desktop app is great, but we want our users to be able to play the game on multiple types of platforms. This game can also be available in user browsers and updated in the browser as well. Our users who are playing on an android phone will be downloading an app from the store. To begin this process for Iphones we will use xcode and use the language swift. To create this for android, we will use android studio which typically is written in java. Since our users will be using all kinds of platforms, testing will be really important, as well as, frequent updates. When we look into UX/UI we will want to keep this in mind. Simple may be better in this case and create a look that will work on many different platforms. Knowing what our users are playing on will help in the functionality of the game for them. The tools we will be looking at are css, html, and javascript. These will help us with our project. Since our project will encompass wide variety of cabalities and knowledge, the team will be vast or highly experienced. We will need a lot of developers who know how to work on different platforms and design for many different devices. The licensing costs are like the costs stated previously – you aren’t going pay anything with Linux, but you might have to otherwise.

## 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**: A web based platform would be great for this game, as well as, being available to be installed on windows OS. This gives users a wide range of options to play the game. This is also very lost cost for the users and reaches a very wide audience. The user will be able to update the game in their browser to ensure security updates or they have the option of downloading them to their computer.
2. **Operating Systems Architectures**: This IDE allows the user to organize their files, run software, and play games depending on what size ram they choose. If we are going to code it in Java we will want to use the Eclipse IDE for this project.
3. **Storage Management**: The windows operating system allows the user to only use as much space as needed. The windows os interface is easy to see where space could be allocated at any given time. Since these players will also be playing on their chosen browsers, we will be using the cloud for our memory and storage management. We can buy more cloud space when it is needed in the future.
4. **Memory Management**: For this project, we will be using caching quite frequently to help with memory. Since these players will also be playing on their chosen browsers, we will be using the cloud for our memory and storage management. We can buy more cloud space when it is needed in the future.
5. **Distributed Systems and Networks**: We would use a database that all users would be able to access so that they aren’t storing too much on their computers unnecessarily, but in the cloud. We will be using a load balancing system to ensure that our game plays smoothly for all users.
6. **Security**: We would not store user information anywhere but a trusted source space in the cloud. It would make sure that we are controlling the data and ensuring no data leaks. We also will not be saving any personal information that isn’t strictly necessary. WE will be performing very frequent security updates to all browsers and encourage all windows os users to download frequesnt updates. Encryption will be key in this project.