

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 | 06/19/2022 | Shawn Pierce | 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 point of this game is to have multiple teams go against each other in a round based game with four rounds. The object of the game is to be the first of the teams to guess a picture that is rendered on the screen. These drawings are rendered steadily and are complete after thirty seconds. After which the other teams can guess if not guessed correctly by the first team. They will have 15 seconds to do so. The game will also need to have one or multiple teams with multiple players on them. The team names will have to be unique and also allow users to check if the team name is in use already. A unique instance will need to be made using identifiers for the game team or player. Selecting a platform should not be much of an issue because the requirements are quite attainable. Even if this game was to be ran on all platforms it would be possible, however some platforms may be more ideal than others. If there is a specific request for a platform that would be ideal to know up front, otherwise I would have to write the programming for multiple different platforms, which will add to the amount of time needed to complete this vastly. It would also be good to know how long I have to complete this project.

## [Design Constraints](#_2et92p0)

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.

Other than the list of requirements you gave us above, the largest concern I have is the rendering of images on each person’s screen. This would be significantly more difficult depending on a few factors: Are the players all over the world? Do they all use the same platform?

I will also need to make sure I have identifiers for the game team or player so that each game is unique to the players in it. There will need to be input for players to add team names and their own names. Only one team can go at a time so there will need to be a restriction on every other team for the first 30 seconds. Then each team gets a single guess to input. The very last constraint I can think of is that the entire image gallery will need to be saved onto the application and accessible by every 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 Entity is the main program that all of the game is connected to in some capacity. All of the aspects such as Game, team, player and gameservice could exist without the entity, however in this UML it is identified by the entity. The programdriver cannot exist without the singletontester, however and is also a use of the singleton tester. The entire entity our extensions of each other as they are all connected and can involve zero to many of each extension. The gameservice is not a direct extension of the entity, but it is created anytime a game is created to actually run the game using the game, team and 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)

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** | In my experience Mac has great server side capabilities. It is one of the most used systems with many apps and also cloud based services. | Has multi-user capability making it ideal for our program. Linux also has great security making it great for the server side of things. | Windows owns many landscapes on the internet and has many services. It offers many servers that can be accessed across the world and has cloud based services as well. | Mobile devices have a large server side with huge online games and even run console games on mobile devices now. |
| **Client Side** | The client side of Mac is great, but not ideal. This is highly accessible and user friendly, but does not have the most functionality and can be slightly confusing at times. It is a powerhouse, but again not the best. | Linux is highly accessible and is open source so it can be accessed by all. Linux also supports different languages keyboards since it is used worldwide. | Windows is the most versatile and usable in my experience. Every application has a use and every use has an application. Many services are free or no charge and is used and appreciated by many. | This would most likely be the easiest of the four platforms to program for and would also be the most accessed by people across the world. |
| **Development Tools** | Java is the best programming language to use with Mac. Which is great because the code is already written in Java. It can use many programming languages however. | Linux is flexible making it usable for desktop applications and server applications as well. They offer many software applications of their own as well. | There are too many tools to list and many people who use Windows that can create more and more tools. Windows is constantly progressing and is one of the best operating systems out there. | There are endless tools to use on and for mobile devices and can also run multiple different platforms depending on the mobile device. |

## 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**: Mobile device
2. **Operating Systems Architectures**: This choice will make it so much better for the servers and make it much more accessible because nearly every person has a mobile device in their hand. The hardware also does not need to be that powerful in order to do the functions we need to carry out.
3. **Storage Management**: We will not need a large storage because we only need to store stock drawings and the names of the teams.
4. **Memory Management**: The memory system will need to be quite good for the reason that we want to render the images slowly. The system has to perform this task appropriately because it is a vital part to the game.
5. **Distributed Systems and Networks**: Using mobile devices are the best way to hit this benchmark because they operate off of multiple different platforms, but communicate to the same servers. As long as our servers are fast enough and strong enough to communicate quickly then this would be of no issue to successfully meet this requirement.
6. **Security**: Security admittedly is not the best on mobile devices, but we can still use very powerful anti-virus protection on them. Scans would be needed to be done more often for them.