

<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 | <01/17/23> | <Tim Hansen> | <Layout of the Draw It Or Lose it software> |

**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)

<Draw It Or Lose needs to be available on as many platforms as possible, without compromising security. The game will need to have multiple teams as they will compete against one another. Pictures will then pop up and the teams will need to guess what they are.>

## Requirements

*<-Game needs to be appealing to players and be as fast as possible*

*-Correctly implement players and teams*

*-Have proper validation and security measures*

*-Authorization needs to be well done and be a focus. We should begin developing authorization checks early in the software development lifecycle.*

*.* >

## [Design Constraints](#_2et92p0)

< Game needs to be available on all platforms

-Have more than 1 team with multiple people

-Unique names need to be available as well as the correct validation.

-Only allow one instance in memory at any time. Need to create unique identifiers for each instance.

-Have a time limit

>

## [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 superclass entity allows the classes player, team, and game to inherit information from it. Those three classes also share an id, such as game id, player id, and team id. In this diagram, there is also aggregation as well as inheritance. Game service is interacting with the game class, the game class is interacting with the team class, and the team class interacting with the player class. Finally, main() is called in the program driver as it is the driver 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](#_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** | <-Apple products are very well built and high speed  -Widely used  -Built in security  -Can be expensive  > | <  -Very good security system  -Cheap  -Not as supported and popular  -Difficult to learn  > | <-Low expertise  -Fast loading time  -Poor support system  -UI could be better  > | <-Very Popular  -The most portable out of the four  -Security isn’t that great  > |
| **Client Side** | <  -Decent cost  -Longer lifecycle then windows  -Very good security  -Compatible with apple iPhones  -Cloud based solutions, backup is much easier.  -Not easily compatible  > | <  -Very cost effective  -Steeper learning curve  -Linux can be difficult to learn.  > | <  -Low expertise level  -Medium price level  > | <  -Much more flexible  -Client’s will need the proper hardware on phones to run the game.  -Renders lower technology obsolete.  .> |
| **Development Tools** | <  -Mac has a wide variety of editors, the most popular being visual studio code.  This editor would allow us to develop our app with ease  > | <  -Also has a lot of useful editors  Such as visual studio code, which is extremely powerful and popular.  > | <  -Can run visual studio code, and Microsoft visual studio. These both are extremely powerful and popular.  > | <  Mobile development is trickier then on the other three.  -Platform specific.  > |

## 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**: <Based on all four, I would recommend Mac as the operating platform to the client. I believe it is the best option out of the four.>
2. **Operating Systems Architectures**: <Mac has an extremely fast and well-designed OS. Macs also contain top of the line processing components, as well as requiring less support than other systems.>
3. **Storage Management**: <Clean My mac X is a top tier Mac storage analyzer, that is also the most powerful one available. This allows you to optimize your storage and see which files you may not need. This will allow us to have a more organized and optimized development of our app. Macs also come with at least 256 GB of storage, which is plenty for storing our files.>
4. **Memory Management**: <Draw It Or Lose It will require quite a bit of memory that will need to be stored in the correct places. Mac allows you to easily create folder names as well as files within them so you can see where each bit of data is saved and what it is interacting with. This will allow our app to grab files it needs from storage, display them, and then finally send it back when they are not needed. This will ensure Draw It Or Lose is can function properly without lag.>
5. **Distributed Systems and Networks**: <Apple allows you to connect to a server so that you can connect and distribute your game files on another operating system. This is a great feature that will allow us to be flexible on other systems. >
6. **Security**: <Mac OS comes with arguably one of the best security systems. Mac anti-virus scans files and prevents the user from opening a file/folder if it is deemed malicious. This would allow us to work in a secure environment without stressing about data leaks and breaches.>