

Draw it or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_gjdgxs)

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_35nkun2)

[**Design Constraints 3**](#_tyjcwt)

[**System Architecture View 3**](#_2s8eyo1)

[**Domain Model 3**](#_17dp8vu)

[**Evaluation 4**](#_3rdcrjn)

[**Recommendations 5**](#_lnxbz9)

## [Document Revision History](#_1ksv4uv)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | Sep/17/2023 | Nate Dukes | Initial Revision |
| 2.0 | Oct/14/2023 | Nate Dukes | Final Revision |

**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](#_44sinio)

The Gaming Room is intending to develop a multiplatform social competition game by the name of Draw it or Lose It. The Gaming Room’s current setup is only available on Android, but with the help of Creative Technology Solutions, we will enable cross platform compatibility for the multiplayer component of the game.

## [Design Constraints](#_2jxsxqh)

*\*Multi-platform*

*\*Multi-player*

*\*Unique identifiers*

## [System Architecture View](#_z337ya)

## [Domain Model](#_3j2qqm3)

GameService references Game, which references Team, which references Players. There is an Entity class, which Team, Game, and Player all inherit from. The entity class is the core of the program here, everything deriving function from it or ultimately being referenced to it through one class or another.

**"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](#_1y810tw)

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** | Typically serverside applications for Mac devices only really support other Mac applications. However, compared to linux, they are easier and compared to windows, cheaper to run. | Very focused and typically open source applications available already which could help out a great deal. | Super well researched, but slightly more expensive compared to Mac for hosting server side. | Unlikely to be effective for server hosting, but for small scale games like these, it could very well work for locally hosted application. |
| **Client Side** | More expensive and requires a bit more expertise compared to other platforms. | Significant user expertise required for operation. More expensive. | Even more expensive than linux somehow, but is very accessible compared to the other platforms. | Challenging implementation, though all other factors considered, fairly even. |
| **Development Tools** | Same as windows. | Java script is available on all of these platforms, and as the game is already developed with it, it’s probably the only one needed. Anything that mac and windows can use, Linux has alternatives, including Eclipse. | Java script is available on all of these platforms, and as the game is already developed with it, it’s probably the only one needed. Very accessible programming, could be used to develop for other platforms like mobile. Using an IDE like Eclipse. | Java script is available on all of these platforms, and as the game is already developed with it, it’s probably the only one needed. Programming on a phone doesn’t seem easy though. |

## 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 would be to use Windows- the large scale accessibility and knowledge already present for most people is incredibly valuable for development. This will allow the development team to expand to other platforms more easily as well with a strong baseline on windows.
2. **Operating Systems Architectures**: Windows has just about every feature you can think of developed for it already- choosing any of the others limits availability of these systems in some way no matter what you end up doing.
3. **Storage Management**: For an application or game of this scope, storage should not be a major issue, though in the worst of cases, there are numerous built in storage management systems for Windows OS. The drawings made by users can take up a fair amount of space.
4. **Memory Management**: Most modern windows operating systems will have far more than enough memory to deal with games like Draw it Or Lose It, given the scope of the application.
5. **Distributed Systems and Networks**: Developing either servers hosted independently from the players to send information across networks and between different operating systems is the most straightforward route; however, the only real stipulation for cross platform compatibility is real time rendering of images and drawings, something which is as simple as sending over a few packets of data between the devices.
6. **Security**: Limiting the accessibility to data is a good first step- after all, there isn’t much information that needs to be shared between devices. Hiding the user’s information between machines is well documented on all platforms and can be done quite easily.