

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 | 09/11/2023 | Brock O’Malley | Added Executive Summary, Requirements, 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)

Our client The Gaming Room is wanting to create a game called Draw It or Lose It and needs help on setting up the enviroment for the game. Each game needs to be able to have multiple teams with multiple players where each team name and player name is unique. Only one instance of the game can exist at a time. The code for this needs to be completed using Java.

## Requirements

* 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.
* Program be written using Java.

## [Design Constraints](#_2et92p0)

The design constraints of making the game web-based is that there are many different kinds of browsers people use meaning that issues could arise when being used in each browser for different reasons. On On the application side having each game, team, and player be unique can be a challenge depending on how implemented. Each instance of a game will need to output the names of each team and player and store then in a database that is globally accesible to all other instances of the game.

## [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)

In the UML class diagram the Enitity class is inherited by the Game, Team, and Player class. The ProgramDriver class acts as the primary class used to run the program. The SingletonTester is a class where the method in it tests if there is a singleton instance of the Game running. The GameService class calls various method from the Game, Team, and Player 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** | Very similar to Linux on the server side so development should be relativly similar. | Easiest out of the group for development. This is due to most servers running a Linux based OS. Cheapest to develop and run as most servers are Linux based. | Most difficult for web development as most servers run Linux based OS and Windows server is the least similar to Linux of the two other platforms. Most exspensive to develop and run as most servers are Linux based | Android is built on a Linux based OS so development for this should be easy to extend from Linux. |
| **Client Side** | More limited by the operating system in what can be done. This in turn could add time for development. More secure though for the end users. | A little more difficult to develop than windows but not as time consuming as Mac development. | Windows has the advantage in this as it holds a majority share of the market. Security on a Windows machine is not as good as something one would receive by default on a Mac though. | Developing on Mobile Devices means you need to account for the difference in input. This in turn will require a different version of the program to be written with these constraints in mind. This in turn will add cost to development but could with worth it as mobile games are a big market. A level of speacialization may be needed as well as development for ios based game would require a development team with knowledge in Swift. |
| **Development Tools** | Swift is the most relevant programming language for developing ios web-based applications. Xcode is one of the top rated IDEs for Swift based development. | Linux like windows supports a wide range of coding langauges with Java being one of them. For IDEs most of the windows IDEs will work with my recommendation being IntelliJ. | Wide range of prorgramming languages with Java being compatible. For IDEs either Eclispe or IntelliJ will work VScode is another popular one. | Langauges for this depend on what operating system you want to write for. Apples ios Swift is the go to langauge with Xcode being the recommended IDE. For Android Java will work and Android Studio for the IDE. |

## 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**: As Draw It or Lose It will start as a web based platform I would recommend it to be developed on Mac.
2. **Operating Systems Architectures**: In Mac Operating system all parts of the system work together to share processing time and memory. In turn this means that if one application is unresponsive it comprises the responsivness of the whole Operating System.
3. **Storage Management**: Recommended storage system for the game would be utilizing a cloud based system like AWS. This would allow the games storage needs to scale dynamically with whatever the current demand for the game would be. This in turn could be a cost savings measure.
4. **Memory Management**: Mac OS allocates a total of 4 gigabytes to each process being run on the computer. Any of that data that isn’t currently being used is then written onto the disk. When memory on the computer is reaching its max capacity it then takes unused data and writes it onto the disk.
5. **Distributed Systems and Networks**: As mentioned above utilizing something like Amazon’s AWS would be best. It is supported across many different platforms and would allow them to be able to communicate with each other.
6. **Security**: There are many different ways to keep the customer’s data secure. One of these ways would be to utilize a company called Cloudflare which is a website security company. It would help prevent agaisnt data theft or even Ddos attacks.