

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 | <05/21/23> | Kylle Waldie | Created the Entity class and added addPlayer() and addTeam() functions. |
| 1.1 | <06/04/23> | Kylle Waldie | Added the Evaluation section of the software template. |
| 1.2 | <06/18/23> | Kylle Waldie | Added the final section to the template. |

## [Executive Summary](#_sbfa50wo7nsh)

The game for the company The Gaming Room is looking to create a game called Draw It or Lose It that renders images from a large library of stock photos and the team must guess what it is. The game can have multiple teams playing and have multiple players on each team. The game name and team names must be unique to allow players to check whether a name is in use when checking names. There can be only one instance of the game running at a time. Using a singleton design model will solve the two main problems that being there can only be one game running at a time and each team must have a unique name.

## [Design Constraints](#_2et92p0)

The main design constraint that the program will use is the singleton model so that only one instance of the game will be running at the same time. As well as only having teams that have-to-have different names. Processing power shouldn’t be a problem since the requirements to run the game shouldn’t be that high. Other design constraints are having only one instance of the game running at a time. Each team must have a different name from any other team. Each image is rendered in 30 seconds and if the team doesn’t guess the correct answer before time runs out the other team has 15 seconds to guess the correct answer. The client must also be web-based.

## [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)

Looking at the UML I can see that the classes Game, Team, and Player all inherited from the Entity class. The GameService class does not inherit from the Entity class like the other classes, but it is a 0 to many relationship with everything else. I can also see that the class ProgramDriver holds the main function to run the entire program. It also uses the SingletonTester to test if the program is running a single instance of the GameService class since this program requires it to be of a singleton design.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | <Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.> The advantages for hosting on mac is that it is secure and the hardware is top of the line. The downside of it is that a little more expensive than other platforms for hosting. There is a company called Mac Highway that starts out at about $30 a year. | Linux is about the same as Mac since they both use the same servers. The servers that they use are the Apache servers but the upside to linux is that it is free to host on. | Windows hosting is a little worse than Mac and Linux but if you are looking to use applications that use ASP, .NET, or MSSQL you are pretty much forced into hosting on windows. There are also a lot of windows-based host with a range of prices depending on the company and plans. | I cant find a lot on mobile but it does seem pretty easy to host on a mobile device but a lot of the time it is done in the cloud and not exactly on the device. Android seems to be the best option for hosting since there are a plethora of apps that help you host. |
| **Client Side** | Mac has a wide range of supported web browsers. They have a quick development time and deployment time. The con of Mac is that you need Apple products that are running MacOS | Linux has a ton of open-source software, so it is pretty easy to develop for. Linux also has a quick development and deployment | Windows has a wide range of supported browsers with easy cross platform testing other than MacOS. Windows also has a quick deployment and development time. | You need to have a pretty good understanding of Android app development to really work with mobile. It also has a long development time |
| **Development Tools** | Depending on what language you are using to host with. Mac has Eclipse IDE for Java and a dedicated language for developing applications for it in Swift. | Again Linux has a ton going for it since it is open source so there is a lot that you can use. Linux also has Eclipse for developing in Java and VSCode for other languages | Windows has most applications to help develop with the only ones that it doesn’t have are the MacOS specific applications for development so you would not be able to develop using Swift. | Most Applications will be developed on Android so you would use Android studio as your IDE or IntelliJ for developing in Kotlin. |

## 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**: I would recommend Windows for hosting since it is common, and most people are familiar with a windows system.
2. **Operating Systems Architectures**: Windows has a pretty simplistic GUI that is pretty easy to navigate. It also has a wide range of IDE’s that you can choose from that allows for a lot of customization.
3. **Storage Management**: Windows storage has two options to choose from either on-site storage using SSDs and hard drives there is also cloud storage while more expensive can also be expanded easier in case the client has future plans to increase the size of the game.
4. **Memory Management**: Virtual memory would be preferred since it hands larger programs easier. It also has memory protection.
5. **Distributed Systems and Networks**: The main issues that you are going to have with windows is lagging and having overloaded servers. Once the game is created it shouldn’t be hard to implement cross platform into it by using outside applications that mange it.
6. **Security**: Windows has a built in antivirus program in Windows Defender protecting the client side information. But it is also important to do routine security checks.