

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 01/20/23 | Derek Kwasniewski | Updated cover page, summary, Design Constraints, and Domain Model |
| 2.0 | 02/03/23 | Derek Kwasniewski | Completed the evaluation section |
| 3.0 | 02/15/23 | Derek Kwasniewski | Completed recommendation section |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is requesting that Creative Technology Solution (CTS) develop a web-based game based on their current game, Draw It or Lose It, which is only available in the Android app, so that it can be on multiple platforms.

## [Design Constraints](#_2et92p0)

Current Design Constraints Are:

* Must be a web-based game based off of their other game, Draw It or Lose It which is only available on Android.
* Must be available on multiple platforms.
* Must render images from a library of stock drawings.
* Must render images steadily and completed at the 30-second mark.
* Must have 4 rounds.
* Must switch input to other teams for 15 seconds each if the main team guessing doesn’t guess within 30 seconds.
* Must have one or more teams.
* Must be able to assign multiple players to a team.
* Must make sure there are no duplicate team names.
* Only one instance of the game in memory at a time.

The best way to overcome the constraints is to use classes for the game, the teams, and the players. In the game class we will keep a list of all the teams so that when a new one is to be created, we can check the list to make sure the name isn’t already taken.

## [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 ProgramDriver class will be the class that *drives* the program. It will use a SingletonTester class to test to make sure that there is only one instance of the GameService class created. The Game, Team, and Player classes are all child classes of the parent class, Entity. From the Entity class the child classes will inherit variables id (long), and name (String), as well as methods Entity(), Entity(id, name), which are constructors, and a method to return the id, one to return the name, and another to replace the default method to display details of an instance (toString()). As specified by the connecting lines, there can be zero to many Players, zero to many teams, and zero to many Games, which will all be managed by the single GameService instance. The GameService class instance will hold data such as a list of the current games, the next ID to use for the next game ID, the next player ID, and the next team ID.

**"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** | Mac servers will cost more but are able to run Apple products when used in a business setting. However, to use the mac server in a larger business setting, this will require much more knowledge than that of a large Linux or Windows server due to its complexity. | Linux is a free-of-charge, open-sourced option that is rarely hacked into and has very little security errors. However, Linux is more complex to operate and update with little support for older versions of the server. | Windows servers -are a beginner-friendly option with the use of a gui. Updates are made available quickly and the server has the option of automatic updates. This also has its benefit of guaranteed support and allows the integration of windows apps. | Hosting a web-based app on a mobile device can be quite difficult due to its ranging performance depending on the mobile device’s specifications and its limited support with other apps, if any. |
| **Client Side** | Using Mac for the client side will usually end up costing more due to the higher cost of Apple products. Due to one of Mac’s main languages being SWIFT, this may cost more time as SWIFT is not used as much as other languages as well as Mac not being a large part of the operating system market. | Linux is a free, open-sourced operating system that utilizes almost all programming languages. Linux has a higher learning curve but due to its use in development, it has a higher volume of support available. | Windows has a moderate cost but is relatively lower than that of Mac depending on the needs of the user. Web-based applications are mainly developed in .NET languages such as C# which is a relatively common language. Moreover, Windows is a dominant OS with a little over 75% of the market ensuring there is plenty of support if need be. | Mobile devices that utilize Android mainly use Java which is a very common language whereas iOS based mobile devices utilize SWIFT, same as Mac devices. The cost of both is relatively the same in this field for mobile devices as Apple products tend to price a little higher than most, however newer mobile devices are around the same, if not more expensive then that of Apple’s. |
| **Development Tools** | Developing software for Mac clients will utilize Objective-C and SWIFT and will use an IDE such as XCode. XCode is normally a free IDE however when used for publishing apps to the Apple app store, it will require a $99 yearly fee. | Developing software for Linux based clients will utilize C/C++, Python, Java, and many others. There are many free IDE options available for Linux such as Visual Studio Code, PyCharm for Python development, and many others which are free. | Developing software for Windows based client will utilize .NET languages such as C# where one will need to use IDEs such as Microsoft’s Visual Studio Code that has a free plan for the community, and plans that run from $45, to $250 depending on the features needed for development. | Developing software for Android based clients will see the use of Android SDK as an IDE utilizing Java as the primary language. Android SDK is a free to use software with plenty of support and help online. When developing software for iOS clients, you will see the use of XCode, same as Mac with the same cost of $99 a year. |

## Recommendations

1. **Operating Platform**: The recommended operating system for The Gaming Room would be a Linux based server since the front-end of the program will be based on that of the client’s platform. Linux will ensure that the cost of running the server will be cheaper than that of the others and has much more support for security risks to prevent unwanted access.
2. **Operating Systems Architectures**: Linux would be the backend operating system for the application’s setup. Linux has a couple main components with the main one being Kernel, which is the core of Linux which virtualizes hardware resources to provide each resource with its resources making it feel as though those processes are the main process being run. Next is the system library which just has the main functions used to implement the OS. Shell is the next component that is the “interface” between the complex functions of the Kernel, and the commands that the user enters. Next is the hardware layer which is all the main components of the device such as the GPU, CPU, RAM, HDD, etc. and lastly is the system utility which allows the user to utilize its functionality.
3. **Storage Management**: Linux does its own storage management during its runtime so the main choice that needs to be made is either The Gaming Room would like to use HDD, or SSD. Either option will be ok as there will be no need for large data transfer during the game’s operation.
4. **Memory Management**: Linux’s subsystem manages the memory by using techniques of demand paging and virtual memory where the subsystem maps files in the address space of the processes happening. This is a more complex system only having active elements in the memory which will help save more memory.
5. **Distributed Systems and Networks**: Draw It Or Lose It will use RESTful APIs to allow the server to communicate which each of the clients regardless of their platform.
6. **Security**: When user data is stored to the server, extra information can be saved with the user’s data so that when the user logs back onto the game/server, the user must complete extra verification that they are that user and are allowed to retrieve that information. Such methods that could be implemented is 2 factor authentication such as a normal email/password login with a one-time password sent to the users linked email and/or phone via text or call.