

Draw It or Lose It.

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

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

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/19/2022 | Nevena Young | These are my recommendations for the proper creation and maintenance for the application. |

**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)

The Gaming Room wants to develop a web-based game that can run on multiple platforms. The game is called, “Draw It or Lose It”, and is only available on android. The way to play this game is by having multiple teams, consisting of several people, going four rounds at a minute each. When a picture is pulled from a library, one team guesses the answer until time runs out. If no answers, then each person gets to answer for only 15 seconds. It would be ideal for this app to work on multiple platforms to reach any device. It is unclear if the app would be web-based and multi-platform due to the requirements. Due to the structure of the game, we would need the images to render at a steady rate and completed in 30 seconds.

## [Design Constraints](#_2et92p0)

The Gaming Room wants this app to run on all devices. Even though we already have it on android, we need to work it into another mobile device… Along with machines like Windows, Linux, and Apple. To do this we will need to come up with a way to use existing code to run on other devices by inheriting other languages. Like fusing java and c++.

## 

## [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)

Entity creates a relationship between Game, Team, and Player class. This means they all get information from Entity. Each class will share common references like “name” and “id” ... Making Entity a superclass. Looking at their relationship, we see Team and Player is a “has a” type. While Game has Team and GameService has Games. When we look at this diagram, we see GameService has a reference of Games, Games has a reference of Team, and Team has a reference of Player.

**"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**  Each O.P. does offer a server-based deployment method.  Using a Linux based server with an open source for potential licensing is free of charge. | Flexible terminal commands to configure the server.  Characteristics: The most popular for web hosting.  Advantages: It can be upgraded and has multiple options for web hosting requirements.  Disadvantages: Less preferred for web hosting services. | Linux is more cost-friendly  Characteristics: Very Secured and most preferred. Has an open source that is secure and free!  Advantages: Security flaws are quickly caught. Plus, very cost effective.  Disadvantages: Difficult to find applications to support the web hosting required needs. Also, has less support. | More software available compared to other OS.  Characteristics: Dominate to the other platforms. This O.P is not free.  Advantages: High resource requirements. (Less loading time and high comfortability)  Disadvantages: Easy to catch viruses and overall poor tech support. Also, windows are expensive. | Specifications are better in other devices.  Characteristics: Very popular with high portability.  Advantages: Has a wider reach and cost sufficient.  Disadvantages: Poor security |
| **Client Side**  Apps on a desktop is good for several reasons. With intricate apps, a desktop version can provide the best experience… Although, there are more steps required to download and operate an app, rather than a quick download or in-browser use, like a mobile device. | Moderate time required.  Depending on the make and model, the cost is minimum.  On Mac we use IDE – XCode and a few different programming languages. | Maximum expertise of Linux is needed, as well as maximum time.  Depending on the make and model, the cost is minimum.  On Linux we us IDE – Eclipse, which is the popular choice. Also, C++ is the computer language used. | Minimum expertise and time required. Depending on the make and model, the cost is minimum.  On Windows we use IDE – Visual Studio. The default programming language used is Visual Basic… Even though it’s difficult to develop network apps with V.B., it's still manageable. | Provides clients with updates at any place. Slightly more difficult to implement than other devices.  Depending on the make and model, it can be costly. |
| **Development Tools**  We would use technology stacks with HTML, CSS, and JAVASCRIPT. Plus, we would use modern framework lick REACT or ANGULAR JS to build front ends. The impact web apps have on the developers is a positive one. It’ll be easier to update to the latest version instantly… Although there is a downside to creating web apps. The need for them to be super-responsive is a tedious thing, therefore, relying on data from the server for a web app can be extremely difficult. Simply because it’s dealing with the internet, something we don’t control.  This can be costly due to the intricate proceedings. | When running languages on macs, we can run the more popular option. While mixing in nice tools like notepad++. Though Macs can run all languages. | Linux can work with visual studio, eclipse, along with notepad++ for a nice and easy-to-use tool. Along with many more languages and tools. | Easier to use than Linux but can run the same as is. Apps with languages can be used, like visual studio and eclipse. Plus, notepad++ is a simple to use tool. | You can create countless apps using android and swift. Both languages and software can be run on all three machines |

Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

\*BE SURE TO SWITCH MY RECOMMENDATION TO LINUX FROM WINDOWS\*

1. **Operating Platform**:  I would recommend The Gaming Room starts on a Linux OP because Linux appears to be the best choice due to them being more secure than then Windows or Mac OS. Since security is a major priority for the client, Linux would put them at ease. Linux isolates administrator access when Windows give access by default. Lastly, Linux is free and while other OP’s require payment.
2. **Operating Systems Architectures**: Linux contains something called a monolithic kernel design… It runs all of the hardware and manages call related to processing. The OP runs the shell… The shell is a layer of command-line interfaces, where a user can tell the computer what to do. The user can also change the default shell at anytime with Linux.
3. **Storage Management**: The storage for this project is very simple. It requires approximately 2 GB of room for images, plus the code for the program. Since the storage must remain accessible, it’s required to RAM to support a massive amount of users. Using a Linux server would be ideal due to the storage capacity not needing a license.
4. **Memory Management**:  As explained before, Linux can run on any device with little use of RAM. RAM is what stores temporary items as they are being used within the app.

While creating this game, you will need to create a library with lots of pictures. The memory allocation allows for easy storage of pictures outside of the default picture folder… Allowing you to keep your whole project together in a more secure area on your computer. This will be possible and achievable with RAM and Docker. Docker enables the operating system to run for multiple sessions within the app. This combination is best for saving space and working efficiently… Which is available on Linux and windows only.

1. **Distributed Systems and Networks**:  Due to each operating system being different, I investigated ways to publish the game to run on all dives. I found Develop 4, which enables cross-platform game creation… It’s an IDE that can run on any device. Once the game is created you can simply export the game file into the web, iOS, Android, and many more options that allow cross-play. This will help with dependencies. To prevent other problems like outages or connectivity, the company will need to make sure their servers are strong enough to support large player volumes, along with backup power for outages.
2. **Security**: Protecting the user’s information is the upmost importance. If possible, a local server should be used, rather than a distributed system due to the chance of unauthorized access. Using Linux, the client will be able to add on more protection… Giving the option of Admin or User for example. Linux comes with multiple layers of protection, so to access Admin or User, would require a multi-step authentication to ensure no unauthorized user. With the use of Linux, the client can also grant or minimize what the user can or cannot do. With that said, I believe Linux is the best OP for the application ‘Draw It or Lose It’.