

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 10/02/2021 | Carlos Oviedo | Added information for initial design project |
| 1.1 | 10/03/2021 | Carlos Oviedo | Added information for the evaluation section |
| 1.2 | 10/23/2021 | Carlos Oviedo | Added recommendations on the recommendation section |

**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 new client The Gaming Room, wants to develop a web-based game that can be available on multiple platforms. The current game they want to base on is *Draw It or Lose It.* This game only works on Android currently and the company wants the game to be able to run on other Operating Systems such as MacOS, Linux, Windows, etc. The game consists of teams competing to guess what images are being drawn. There are time limits for guessing as well for showing the drawing mark time. The game consists of four rounds of play lasting one minute each.

## [Design Constraints](#_2et92p0)

Here let’s look at the design constrains for the game application:

* **Cross-platform:** The game is currently an android app and to make it available to other platform we will translate the app to a webapp that can be accessible from other operating systems. To avoid having to choose a compatible language across platform we can use a REST API to communicate using HTTP
* **User Interface:** The game has been already deployed as an android application. We would need to develop a similar if not identical interface design for users. Another option is to upgrade the design and deploy it like a brand-new game.
* **Multiple teams and player per game:** the game must have the ability to support more than one team and each team will have multiple players. To do this, we would need to use a client-server architecture and make sure that the server can handle multiple players at once.
* **Unique iDs:** games and team names must be unique and only one game can exist in memory at any given time. To accomplish this, we would need to incorporate unique identifies and ids to games, teams and player. Also, we would need to find the best approach to manage memory allocation considering that the application will be available for multiple platform we also would need to identify player platform ids.
* **Images and Copyrights:** The gaming room wants to use images when playing the Draw, It or Lose It game. We will adopt the images already used in the android platform and make sure they are compatible for all the other platforms. We also need to get licenses for other images added or copyright permissions during design.

## [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 UML diagram also known as Unified Modeling Language will be utilized to provide a visual design of the game system. Let’s look at how the program will be developed. On the UML diagram we can see the entity class creates a relationship between the game, team, and player classes. The arrow indicates that all these classes will inherit from a attribute from a super class. We can see the classes, variables and methods that will be used during development. The programDriver Class located in the top left corner is pointing to the singletonTester. This shows us that he programDriver will use the SingletonTester to test the code. This is developed so we can test the restriction of having one instance where the game can exist in memory. The Gameservice class will hold all the complex method that make up the skeleton of the game and it functionality. One of the requirements of the program is to unique game, teams and player that’s why the classes for each have been design in this diagram. The links or lines connecting to each class tell us that they all are associated. The numbers between the lines tells us the number of associations within each class. For example, the GameService can have zero or more games associates with Game class and so on with the other classes. This diagram will be use during the development process to create a final product and some adjustments can be made along the way.

**"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** | Apple the company that created MacOS does offer server-based environments. Having a MacOS server can offer major advantages for mac clients on a network. These advantages are the fully support of Mac applications as well as easy general administration with a great graphic interface. The weaknesses of having a Mac server are first, it can become expensive to maintain, second it is not great for big companies or enterprises that rely more on third-party programs and customizations. | Linux offers many advantages for web hosting. Some big companies like google even used Linux severs. The most popular advantage of having a Linux server is that is free and open source. This means the deployment and use of tools would be cheap or free. Another advantage is its security, because Linux is very customizable organizations can customize their security as they see fit. Finally, Linux web hosting supports python, PHP, Perl and Ruby. The cons of having a Linux server are that it has a long learning curve if there is no experience. Also, some favorite applications might not be supported, and there can be migration issues from windows to Linux. | Windows is a well-rounded operating system. It is a proprietary software which means we would have to pay for licensing. Although windows licensing can be expensive there are good advantages for having a windows web host server. One of these advantages is that it supports lots of applications and third-party software, another advantage is the easy patch updates and hardware updates. Last for developers familiar with the platform scripts languages like ASP.NET and databases like MySQL are fully supported. | Although mobile devices for web server is not well known and probably impractical, it is possible to implement. Oracle is one of the companies that offer mobile server-side implementation. Oracle Database Mobile Server can manage applications, users, devices, and data on large deployments of mobile or remote devices.  The advantages of Oracle database structure is iOS dev tools support, android dev tools support, and synchronization to NoSQL oracle |
| **Client Side** | The cost for the client side can be cheap considering that the software will be web-based. We only need to consider the support of the safari web browser which is the default browser for macOS. We also need to consider if we want to create a web app that would be available on the apple store. There must be a developer license cost as well as to use the appropriate language to deploy the webapp. The development can be done with swift or object-c | Linux is open source which means that we won’t spend money on the client side. Also, Linux supports all major web browsers on the internet. Google chrome, Firefox, safari, etc. there can be a learning curve when developing a webapp since there are many Linux OS and not much documentation. It can be hard to support games as well. | The cost for a web-based or webapp deployment in windows is like the macOS. As mentioned before windows supports a lot of third-party applications as well as games. We would just need to consider the fees associated to deploying the webapp on windows store. Also note windows has a xbox store which is for games. For development windows support C#, .Net and lots of IDEs. However, if we only focus on deploying the game to the web, windows support all popular browsers | There is primary three mobile operating systems, windows, android, and iOS. The cost can be very high because we must consider licenses and fees associated to each OS. We can use the browsers that come with the OS, however a webapp would be the way to go. For developing the application, we need to consider the programing language. iOS supports swift/kotlin and android Java or Kotlin. Kotlin is a new language but can be used to reduce time development for cross platforms. |
| **Development Tools** | MacOS has the Xcode IDE. This is their primary IDE and supports various languages such as swift, C++, JavaScript, C, object-c. MacOS also supports other popular IDEs and text editors like visualStudio and Sublime. For developing the game to be web based we can use React, javascript, html, css, however. For a web app that would be available in the app store we need to consider going native which will require using a mac computer and swift programming language. | Linux as mentioned previously is open source and support all the popular IDEs. Linux support most of the programming languages out there such as C, C#, C++, Pearl, etc. Linux is written in C mostly so we might need to consider using C if we want a hybrid web application. However, Because the game will be web based, we would not need to worry about that and can use any development tool at little cost. | Windows relevant programing languages are C#, C++. Windows have an excellent IDE which is Visual Studio. Visual Studio similar to Xcode but better provides support to lots of programming languages like C++, python, JavaScript, and more. Window also have .NET support and PowerShell, very good tool for scripting. One disadvantage only supports native android development. | For mobile devices we only need to focus on iOS and android. These two are the primary mobile OS. If we are going the native route, we need to use swift for iOS as well as a Mac computer to develop the app using Xcode. For android, android studio is available and supported languages are Java and Kotlin. Because the game will be web-based we do not need to worry about this. Both platforms support React Native and web browsers are available. |

## 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**: based on our evaluations and we recommend the Linux operating system. Linux is open source reducing costs, also it supports various development tools, and it is flexible and secure.
2. **Operating Systems Architectures**: most operating systems have similar architectures. Operating systems act as an interface between a user of a computer and the computer resources. (cs2.edu). The architectures are hardware, kernel, and shell. The hardware architecture consists of the memory, CPU, I/O, an any other physical devices. The Kernel is a bridge between applications and the actual data processing done at the hardware level. Finally, the Shell is a piece of software that provides an interface for users to an operating system (c2.edu).
3. **Storage Management**: the game Draw it or Lose it is will be a web-based application. We recommend a Cloud storage unit. With cloud we can only utilize the storage space needed to keep the game running. As the game increase in popularity, we can upgrade the storage without having to spend money on the hardware. Based on our evaluation Google Cloud seems that best fit for this project.
4. **Memory Management**: Linux is a versatile and easy to customize operating system. For this game we will use Java as the main language for the backend development. Java does memory management automatically. Java uses an automatic memory management system called a *garbage collector*. Thus, we are not required to implement memory management logic in our application. (javapoint.com).
5. **Distributed Systems and Networks**: Based on our evaluations the google chrome browser is available in all existing platforms. The game is web-based, thus as long as the browser is available to any OS, People around the world would be able to play we won’t need to consider the user OS.
6. **Security**: By using Google cloud services. We don’t need to worry about data center security. Google will provide the hardware and infrastructure. When it comes to software, we can use google apps and just make sure the game has automated patches and updates from time to time. We also do QA testing before deployment.