

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)4

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/23/2022 | Jacob Silberstein | Identified solutions and filled software design packet to completion. |

**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)

Game Room is working to develop a new web-based game that serves multiple platforms and draws heavily from their original game *Draw it or Lose It*, which is currently only available on the android market. *Draw it or Lose it* is a game loosely based off of the popular 80’s television game show *Win, Lose, or Draw*, where players work in teams to decipher what is being drawn. Similar to the game, this app will use drawings as its main point of interaction, but with the caveat that players will instead be observing images as they get rendered by a large library of stock drawings. The game is four rounds of one minute each. Drawings are rendered steadily and consistently until the 30 second mark, win which the images are completely rendered. Currently the staff involved are unaware of how to set up the environment. The company wants the ability for multiple teams to be involved, as well as each team to have multiple players assigned to it. The game will also require an index of in-use team names so no two teams overlap in name. Additionally, only one instance of in game memory can exist at any given time which is possible through the use of unique identifiers for each instance of a game, team, or player.

Here at CTS, we recommend several progressive solutions to best fit your needs. We would recommend for this style of project, utilizing a system that dynamically stores user data based on the teams created, utilizing unique storing principles for the player names, team names, and rooms in which the players will ultimately play in. Aside from this, we would also create a series of layers that will start the game, end the game, or pause the game, and after a period of time or conclusion of the game, data from previous game is cleared and opened for new dynamic storage.

## [Design Constraints](#_2et92p0)

Major constraints associated with this project include the knowledge level of the Gaming Room team, as well as an unidentified platform other than web-based. Another major constraint will be creating a code that allows for dynamic gameplay that is not the same every time, but also allows for customizability. Additionally, due to the utilization of cross-platform play, several different Operating systems will need to be utilized and regarded for the final product to be functional and editable, development side and client side. The final constraint that we have currently identified is the use of a single instance game state, in which the code will need to have two ultimate outputs of which the user can end the game themselves, or the end of the game is reached, and a new game is started.

## [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 illustrates several very important factors of the program that draw on a base inheritance to the Entity class. The entity class inherits all items from the Player, Team, Game, and GameService Class’s with its inheritance from the Game class drawing additionally from GameService detailing most of the singleton instances of the game itself. The team and Player classes respectively alter and store details regarding to the player and their team. Additionally, for faster run time, the game encapsulates the main details of the game so that the processes associated with gameplay are different than that of what are associated with creating and storing customizable details.

**"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 OS is fantastic for hosting items that are solely meant for Mac OS but is not great for cross platform hosting. Due to the specificity of Apple’s Mac OS’s operating systems, the programs hosted by mac and used by mac run very smoothly, but suffer when being used in a cross-platform methodology. | Linux OS is traditionally incredibly flexible and reliable. Due to its variability with Windows and Mac based OS’s the Linux option tends to be very secure and reliable, while having a relatively easy translation process from system to system. | Windows OS is very good for hosting between non-apple-based OS’s due to its similarity in style and programming. Window is fairly secure, being able to host with additional securities from third party software’s, seamlessly. | Mobile devices are not ideal for hosting web-based programs due their general lack of dedicated space for such tasks. Due the generally smaller size and utilization purpose, Mobile devices tend to be able to handle much smaller tasks and non-complex computational items while maintaining base levels of security. |
| **Client Side** | In terms of the cost, time, and expertise required for working with mac OS, a specialist will likely be required that uses java or C, which is useful for developing dynamic games. Mac OS also has a very streamlined UI which is useful or accessibility and time constraints, but slightly more difficult for identifying pathways and major program directions. | Ultimately the client side of development for Linux is similar in accessibility, time, specialization, and cost, with the main difference being that Linux client isn’t identical to windows client leading to disproportionate levels of access between the two. | In terms of windows client side, the expertise is a bit more open as windows supports several major languages such as HTML, CSS, and Java, and tends to be comparable in time commitment to the other systems. | Client side on mobile devices, much like the server hosting is a bit more strained than ideal. With the client-side time commitment, due to the limitations associated with mobile devices, this process should be slightly faster than others, with a similar cost and level of expertise. The main consideration here would likely be the portability from client to client given the numerous types of mobile devices. |
| **Development Tools** | While Swift is Mac OS’s language for development, there are plenty of IDE software’s for developing in almost any language you could need. Mac OS supports programming in most major IDE’s such as visual studio, eclipse, and even most web-based design services. | Similar to Mac, the Linux systems can operate most IDE’s and can operate similarly to windows and Mac. Aside from that, the common languages seen on Linux are C++, JavaScript, Java, Shell, and Python. | Similar to the other two competitors, Windows can utilize most IDE’s and plug-ins regarding languages associated with most systems. As windows can run most any IDE, similarly to Linux, it is also possible to develop in most any programming language associated. | The most relevant programming tools for mobile app development are visual studio and eclipse due their wide array of plug ins, applications, and variabilities with languages. Most commonly, when developing an app, the languages utilized will be one or a mix of several popular languages such as Java, JavaScript, Python, PHP, Kotlin, Swift, HTML5, C#, or Ruby, just to name a few. |

## 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**: Due to its variability in terms of languages, as well as its consistency and security, we would professionally recommend an approach from Linux. We believe this is going to be the most versatile platform to expand the service you are aiming to provide.
2. **Operating Systems Architectures**: Linux is a popular version of the UNIX OS that boasts free or low costs for personal use, however, also being known as a fast and efficient system as a whole. The Linux system also incorporates a GUI that encompasses several of the competitors’ platforms such as can X windows, TCP/IP, and emacs editor. The systems main components consist of several layers including the kernel, Hardware layer, System library, Shell, and Systems utility.
3. **Storage Management**: In terms of storage management, we would recommend using a Session Storage based system, due to the nature of the program. Since it is a web-based program, regardless of platform, the Session Storage will dynamically adjust for the current play session, and refresh on close.
4. **Memory Management**: Linux memory management is a very complex system of several layers of configurable settings, but for the sake of the web-based application, will focus on the virtual memory primer. This system within Linux accounts for converting physical memory from different sources into smaller sized details, permitting only required information.
5. **Distributed Systems and Networks**: Linux has a long and complex history with distributed programming. The current architecture for this is that Linux utilizes a DIPC or distributed inter-process communication that utilizes separate nodes on individual computers and requires a TCP/IP connection. Due to the lack of an assumption for a synchronized clock, this makes these features usable on a Wide Area Network.
6. **Security**: Linux has a notoriously good system for security, boasting one of the most secure systems for development, due to its programs using high levels of encryption and permission acceptance. If at any point security is a concern, cloud-based storage is also an affordable option, but not entirely necessary given the access that Linux gives its users to protecting themselves. The main thing that Linux has going for it is the significantly lower number of exploits to target the system.