

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 | 03/21/21 | Brad Quackenbush | Suggestions and requirements implemented for making *Draw It or Lose It* a successful gaming 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 build a web-based version of a gaming app called *Draw It or Lose it* with specific requirements regarding the game’s functionality and memory allocation. There can only be one instance of a game in the memory at any given time, but the game can have one or more teams per game as well with multiple players on each team. We have a solution that entails the program being able through sort through lists of memory within the gaming app to determine if the memory requirements are fulfilled regarding only one version of a game is in the memory and having the ability to add teams and players as needed.

## [Design Constraints](#_2et92p0)

1. Security features regarding players having their own unique username/gaming ID with a password to securely login to their account.

* Having a secure login is beneficial for the player using this gaming application for their own sense of privacy knowing their information is secure. Having a secure login can also attract more players knowing they can have a unique username visible to their opponents while playing the game.

1. Enough storage capacity to handle memory involving the game’s features and images.

* Making sure there is enough memory to handle all the game’s features and images within the game to ensure the game runs smoothly and does not crash for the users during gameplay.

1. Ensuring there are servers that are spread out to lower the latency for players in different regions.

* Players who play with higher latency will most likely not want to play the game as frequently due to having a poorer gaming experience with lag and unsmooth gameplay.

1. Adding on to the third constraint by requiring servers that will need to have an administrative system for managing and verifying player credentials/login information.

* This is also an addition to the first constraint regarding security features with the need for a system that verifies player credentials, so each player’s information is safe and there are no issues when they login to play the game using their unique username/gaming ID.

## [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)

This UML class diagram shows the relationship between each of the classes used to develop this gaming app. We have inheritance being used within this UML with the Game, Team and Player classes all inherited from the Entity class. With the Game class being inherited from the Entity class, we also have the GameService class which has an association with the Game class. The GameService class also uses encapsulation hiding the game, team, and player ids from the rest of the program with those being private attributes. The GameService class also uses accessors to read the app’s memory making sure only one instance of the game exists. We also have the ProgramDriver class which uses the SingletonTester class making sure we do not have multiple instances of the attributes that are only allowed to be in the app’s memory once. Lastly in our UML diagram, we explicitly state how the GameService, Game, Team and Player classes can either have no lists within their respective class or can have multiple.

****

## [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** | Advantages include reliable servers, reputable company and large cliental.  Disadvantages include Mac being exclusive making it more difficult for cross platform capabilities. | Advantages include wide variety of languages supported and thus making it easier to use. Disadvantages include it being not so popular and thus making it harder to expand the game’s audience. | Advantages include traditional programming making it an easier platform to work with and is also popular like Mac. Disadvantages include not as easy to work with compared to Linux. | Advantage that is also a disadvantage is having specific requirements that are different than the other three options if this were the sole platform for development. Would require specific set of developers for this gaming app. |
| **Client Side** | Mac is more expensive to develop than using Linux or Windows and clients are required to use Apple products to use this gaming app. | Open source so costs are cut tremendously compared to using Mac or Windows. Development has potential to be easier with having more opportunities to develop the gaming app using a wide variety of languages and ability to use more tools. | More expensive than Linux, but cheaper to develop than Mac. Similar to Linux, there is more opportunities for developers to have more tools and resources compared to Mac. | Very different than Mac, Windows, and Linux. Would need more developers considering most do not develop computer apps and apps for mobile devices. Has potential to be more costly adding mobile devices in the mix. |
| **Development Tools** | Objective C and Swift. Tools used include Xcode CodeRunner. | Java, JavaScript, C, C++, Python (Linux has more of a variety regarding multiples languages to choose from). Tools that can be used include Eclipse, PyCharm, and Atom IDE. | C++ and tools that can be used include Eclipse and Visual Studio. | Kotlin and Java (for Android devices), Objective C and Swift (for iOS devices). Tools used for these languages include Eclipse, Android Studio, Vim (for Kotlin and Java). Atom, and CodeRunner (for iOS devices). |

## 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 believe Linux is the more appropriate choice for an operating platform that will allow Draw It or Lose It to expand to other computing environments.
2. **Operating Systems Architectures:** Linux offers better security for our user’s protection regarding their personal data and is more cost effective than Windows with Linux being an open-source platform and not requiring licensing fees. Linux is also proven to be somewhat faster than Windows even with older hardware as well.
3. **Storage Management**: Utilizing a Cloud storage management system would be most appropriate. Clients will download the game application to whichever device they would rather run the application on, and then have the ability to retrieve their user information as well with specific gaming contents from the Cloud.

Another storage management system that would be beneficial for reducing Cloud memory usage would be to store all the game’s contents (and possibly images) on the disk space for whichever platform the client choses to run the application on. I believe this has its benefits if the gaming images were to stay the same throughout the life of the game, but if images were to be updated, changed, or even added to the game, then I believe using Cloud storage would be most beneficial to save the client from having to update the game periodically if images were to change as well with freeing up memory on their device.

1. **Memory Management**: Linux will implement the use of “paging” for its Memory Management. Thus, the system will be able to read and write data from a secondary source (the cloud or disk space) to be later used with the RAM’s memory. After the application’s contents are downloaded to whichever device is being used, the Memory Management will allocate memory as needed regarding the application’s contents and images. The RAM will also ensure optimal use of the memory allowing the application to run efficiently.
2. **Distributed Systems and Networks:** We will implement the use of the Three-Tier Client-Server pattern while using Linux allowing the client to have direct access to the server via the game application. This server pattern would be most beneficial for allowing multiple clients on different platforms to communicate with the Draw It or Lose It server. The client will communicate with the application which entails all the information involved with the Draw It or Lose It game. The client would also have communication with the server through HTTP which will allow the clients using different platforms to all connect to the same server. Lastly, the application being used by the client will also communicate with the server which covers all the backend information regarding player details and memory within the game application.
3. **Security**: With using Linux and the operating platform being well known for its security features, we will also implement a Two-Step Authenticator for each client that registers for the game application by sending the client a text message to verify their account. This additional security feature will help the client feel more at ease with their personal information. This additional step would also make it more difficult for bot/fake accounts being created.