

**Draw It or Lose It is**

# **CS 230 Project Software Design Template**

Version 5.18.21

## 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 |
| --- | --- | --- | --- |
| 5.18.21 | <06/03/2021> | Ethan Daugherty | * Program Ensured that only one instance of a player or team name can exist at a time. * Added ability to create and add new player and new teams |

**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 Client, The Gaming Room, has reached out to Creative Technology Solutions to develop a web-based game that serves multiple platforms based on their game, Draw It or Lose It. The client The Gaming Room has request that the game be developed with the following requirements in mind. The application needs to have the ability to have one or more teams with multiple players assigned to them. Additionally, each Game and team names must be unique. Finally, only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The Gaming Room has established that the web-based application needs to have multiple teams with multiple players. Additionally, each team and each player will need to have a unique identifier without being the same. Finally, as the application is web-based and native to android it will need to be available to run on Windows, Linux, and Apple devices. The program will either need to be re-written to run on multiple system types or include code to allow it to be run by different operating systems which may require a lot of time, resources, and budget.

## [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 Class diagram shows the relationship between Game, Team, and player share with Entity. Game, Team, and player all inherit with an Is-a relationship using the object-orientated principal of inheritance. Team and player share a Has-a relationship that allows an instance of one class to be used in another. Additionally, Gameservice uses instances of Games, and games uses instances of Team and team players. Gameservice allows for multiple games with unique ids, each one with multiple teams and each team multiple players. Each Game, Team, and player uses a unique id, preventing duplicates and allowing multiple players per team and multiple teams per game.

****

## [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 OSX provides more flexible commands in the terminal to configure the server and what can access it, though is limited to hardware options only provided by Apple and can be quite expensive. | The most reliable and stable of the options. Additionally, is very cost effective as it is available on a variety of hardware options. | Available to run on a variety of different hardware options with a variety of different release options additionally supports html files. | This would be difficult to both code and store information on long term. Other options would be better suited. |
| **Client Side** | Expensive and end user is required to buy product developed by apple. Software is less versatile unless using a virtual machine to load into other OS. | Cost effective though as most people are not native to linux would take considerable time to learn and use. Though available on a variety of machine. | Most available and open with software. Most users have some background with windows and ease of use is minimal. Also available on a variety of machines. | Limited to form factor and battery life. Though provide ready updates to clients and users. |
| **Development Tools** | Objective-C is the primary programming language on Mac OS. | Supports most programing languages: C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, Ruby, and Vala, not inclusive. | Again supports most programming languages, C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, Ruby, and Vala, not inclusive. | Swiftic is typically used to develop on the iOS platform and Java is typically used on Android 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**: As The application is already available on Android devices, I would suggest making it available on Windows and putting the application on the game on the windows app store or on the Steam marketplace. Additionally, as the application is available on Android, I would assume it was written in Java. Java is well suited to work with and write on a Windows computer.
2. **Operating Systems Architectures**: I would suggest a PC with an X86 AMD processer that the application could both be developed and run on.
3. **Storage Management**: for running the game as SSD have become much more affordable, I would suggest just a standard sata SSD. For development of the application as multiple copies and backup of the application should be stored I would suggest a NVME SSD.
4. **Memory Management**: With windows being the recommended OS to both develop and run the application on and 64-bit Windows systems the current norm available to windows PC’s allows for 8 TB of address space allowing all threads of a process to access its own virtual address space. However, windows do not allow threads to access memory that belongs to another process to protect one process from corrupting another.
5. **Distributed Systems and Networks**: Multiple player games based on a network or online games typically have a database shared among the users. Keeping only one instance of a game running at a type helps cut down on duplicate matches and lowers server use. Additionally, I would keep the game programmed primarily in Java to keep it usable and on other OS.
6. **Security**: To start, encapsulation helps to protect protects cases with information within the program. Additionally, according to the General Data Protection Regulation programs need to minimize the data collected when processing and using personal information, for example during storage the information needs to be anonymous or replaced with pseudonyms when ever possible.