

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 | 08/23/20 | Shawn Frye | Initial software design |

**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)

Company "The Gaming Room" has an application for mobile called Draw it or Lose it and they'd like to create a web-based version. There can be multiple games played that most multiple teams. Each team is made up of multiple players. Each game and team name needs to be unique from the others and the Gaming Room needs help developing the environment.

## [Design Constraints](#_2et92p0)

Android, IOS, and the web use different languages to develop applications and can be trick to maneuver.

The game needs to work with 3 different platforms.

Scalable back-end to accommodate large amount of teams and players.

Game and team names must be unique.

Unique ID’s for each instance of a game, team, and player to limit instances of the game to one and reduce errors.

## [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 ProgramDriver Class contains the main method. ProgramDriver uses Directed Association with SingletonTester to test of there is already an instance of GameService.

Entity class is the parent class to Game, Team, and Player classes. Game, Team, and Player all inherit Entity’s required attributes. A Player cannot have a Team, but a Team can have a Player. A Team cannot have a Game, but a Game can have a Team. A Game cannot have a GameService but a GameService can have a game. Game Service must only have one instance of each game running at any time. Each Game can only have one unique Team at any time. Each Team can only have one of Each individual Player at one time.

****

## [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** | Macs can be used as a server, although the licensing is expensive, and you must have Mac books to develop. | Linux is well equipped for a web-based hosting situation. It is also free. Linux servers are extremely common because they are extremely versatile and free. | Windows has server systems and the GUI interface is easy to navigate. Prices vary but usually is really affordable. | Not well suited for server side as they are underpowered with limited connectivity. |
| **Client Side** | Mac/iOS has great SDK’s but everything must be developed from a Mac or PC with iOS which isn’t as easily configured. If the application’s front end is developed in Javascript, it will be easily accessed in web browser on Mac. | Linux is extremely versatile for development but not as easy to use as Windows. Best bet would develop the app’s front end with Javascript which is easily accessible from web browsers within Linux. | Windows is the easiest to learn and the most widely used. The GUI in Windows is unmatched compared to other systems. Application should be developed in Javascript, allowing easy accessibility | For mobile devices developing on them is quite difficult. Unless they have a large screen with keyboard connectivity, developing can be challenging. Accessing a web app is great for mobile devices and most are built with this in mind. |
| **Development Tools** | A Mac Book that has an IDE, preferably Atom on it so the coding and developing would be done using Swift. | Python comes already installed on most Linux distributions. You could use Pycharm IDE to code and develop. | Visual Studio Code is the most common way to code Windows applications. You could use Java or C++. | For mobile devices the most common would be Android OS and iOS for iPhones. Android Studio can be used with Java or Dart, while Xcode with Swift for iOS. |

## 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 recommend using a Linux server to host Draw It or Lose It. Linux systems are extremely common and for good reasons. Linux servers are versatile and the base software is open source and free.
2. **Operating Systems Architectures**: The Linux kernel is stable, secure, scalable and most widely used for hosting applications. The kernel is knows as the control module of the operating system and controls the order in which processes are executed in the processor.
3. **Storage Management**: Storage for the application would be best if it’s a solid-state drive (SSD). SSD’s have faster load times which will increase performance.
4. **Memory Management**: Memory is cheap for server systems and it is the best to have as much as possible for multiple games from multiple users. The Java virtual machine provides a runtime environment which creates a memory in the heap space and also has a garbage collector. The garbage collector destroys objects that are no longer needed which creates room for new objects.
5. **Distributed Systems and Networks**: Being as your system will be a cloud based system, if there are to much traffic from one server, the game can freeze or stop. Also, if the server goes down, all games stop. Separating the system into multiple nodes and servers increases the stability of the system and can increase performance.
6. **Security**: I recommend using a role-based security system. It will allow separation from administrator, game, team, player, and user. With this you can make it so that the privileges of a user cannot restrict access to information they shouldn’t be allowed to access.