

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 | Jeremy Stotts | Initial prototype 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)

The Gaming Room currently has available an Android app called Draw It or Lose It. They would like to develop a web-based game that serves multiple platforms based off the current game. The game consists of multiple games which host multiple teams and each team has multiple players. For the game to function properly each game and team name needs to be unique. Draw It or Lose It contains a large library of stock drawings. The staff at The Gaming Room do not know how to set up the environment.

## [Design Constraints](#_2et92p0)

Android, IOS, and the web all have different software development kits.

The API should be tailored to work with 3 different platforms.

The API needs to be able to allow 1 or more teams from any of the platforms.

Game and team names must be unique.

The ability to alert the team captain that a team name already exists and allow them to choose another one.

Use unique ID’s for each instance of a game, team, and player to limit instances of the game to one.

## [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 the most popular of these and is free as far as licensing goes. | Windows has server and they are very secure and easy to set up and use. But the licensing is expensive. | Although mobile devices can be used as servers. The are not quite equipped to excel at it. They lack the power for high end. It could be used for development though. |
| **Client Side** | Mac has good and easy to use SDK’s but the thing is you must have a Mac Book to develop for Mac. This will add up cost and requires someone that has developed swift. | The cost for this would be highest in development time, you also must have someone that is used to using python. | The expertise is probably the highest requirement for Windows. Would highly recommend using the .NET framework for security and capability. | For mobile devices you want to find developers that have experience developing apps. User interaction and how things are displayed need to be taken care off differently than on the web. |
| **Development Tools** | A Mac Book that has iCode on it. All coding will be done using swift. | Python comes already installed on most Linux distributions. You could use IntelliJ’s Ultimate IDE to code for this. | Visual Studio Code is the standard and best way to code Windows applications. You could use relatively any language but C++ or c# is what most Windows programs are wrote in. | There are 3 options as far as mobile goes. Android’s you will need someone that specializes in Android Studio to develop the app. For iPhones you need someone that has a Mac book that can develop using swift in iCode. Or you can have someone develop the app using unity. Which is C++ and then can convert to either an android app or and iPhone app. But you will still need a mac to convert it to an iPhone app. |

## 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 highly recommend using Linux Ubuntu Server to host Draw It or Lose It on a Kubernetes cloud setup.
2. **Operating Systems Architectures**: The Linux kernel is stable and secure. The Kubernetes clusters allow easy separation of system and even hardware needs.
3. **Storage Management**: Can either use a HDD storage or a SSD storage but I recommend the later. The SSD will allow for faster access to assets, having to load the pictures to users’ devices this will be helpful with user experience. To optimize storage, I recommend setting up a Kubernetes node for file storage and a NoSQL node for game data and user management. In the NoSQL link the URL to the location of the picture.
4. **Memory Management**: To assist with costs I recommend setting up a watcher for the load on the system. That way when usage is low you can lower the required memory and during peak times you can add all that you need so that you will only have to pay for what is necessary to give the best experience.
5. **Distributed Systems and Networks**: Being as your system will be in the cloud if there is required miniatous on the servers your game will not have to stop. You can just move the node to another server or if the system crashes another server will start automatically. Going this route will allow you to host everything you need to run the game except for on the client end. This will allow all operating systems to have a client made that can access that information. Separation of functions that Kubernetes provides easier management and organization of your system.
6. **Security**: I recommend using a role-based security system. It will server your needs best and allow separation from admin, game, team, player, and user. With this you can make it so that a user cannot access information that they should not.