

Draw It Or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/01/24 | Jason Harmon | Prototype Design for application |
| 2.0 | 06/10/24 | Jason Harmon | Updated evaluation |
| 3.0 | 06/22/24 | Jason Harmon | Final Iteration |

**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 currently has a Program called Draw It or Lose it available on the WEB but the want it to be able to be played on a mobile device in the form of an application.

## Requirements

Application needs to be identical to the Website.

Needs to stay within budget

Available for IOS and Android

## [Design Constraints](#_2et92p0)

There can’t be to players with the same name

Has to stay within budget

Needs to be able to allow a player to have multiple teams

## [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 is what pulls all the other classes together and runs the program. It is used directly with SingletonTester to see if there is already a running gameService. Game, Team and Player are child classes of the Entity Class which means that they pull information from ENTITY. A team can have a player but not the other way around along with a game can have a team. Each game can only have one unique team at a time.

**"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** | Macs have expensive licensing and you must require a MAC in order to develop applications. | Licensing is free and is most commonly used for servers. | Very Expensive licensing but is very user friendly and easy to set up | Mobile devices were not built to run servers but the can be used for low end development |
| **Client Side** | Must have a MAC in order to develop for this OS which makes it more costly. Will also require someone that develops in swift | Primarily python. Will take a lot of development time which will be high in cost | Highest expertise requirement. .NET for security purposes a long with framework and capability. | Want to find someone that is used to building applications. The way that the program displays is differently then the web. |
| **Development Tools** | Mac device that has ICODE on it as all coding will be done using swift | Python is pre-installed on linux using Intellij’s Ultimate IDE | Visual studio is the most common and efficient way to code windows apps as you can use any language. It does prioritize C++ or C# as that’s what it is programmed in | Androids need Android studio  Iphones need a MAC device  Develop the app in unity and convert the code later on |

## 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**: After further research and deliberation, I believe that windows would be the best operating system to use for this game. Windows is one of the most widely used OS and there are a lot of people that are specialized in coding structure. It is also fairly cheap to maintain
2. **Operating Systems Architectures**: With the way the OS is set up the application can utilize the user interface to help keep the processor usage down.
3. **Storage Management**: Built in the operating system of windows there is a storage management tool that helps organize and get ride of data that isn’t being used anymore. This helps keep the program running smoothly and efficiently.
4. **Memory Management**: There is also a memory management option in the Windows OS that will help manage the images to help the application easily access the files.
5. **Distributed Systems and Networks**: Being that the system would be in the cloud the game would not have to stop. You would be able to move the storage to different servers. This lets you host everything that’s need to run the game besides the client side.
6. **Security**: I recommend using a system of roles to separate the admins from the game, team, player and user. This prevents a user from being able to access things they shouldn’t be able to. There is also a built in security application in windows that encrypts its data that is being transmitted.