

Draw it or lose it

# **CS 230 Project Software Design Template**

Version 1.1

## 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 |
| --- | --- | --- | --- |
| 1.1 | 12/08/2023 | Alex Mehr | Expand draw it or lose it to other platforms |

**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 is currently available on Android system as draw it or lose it. Client would like to expand its development to web-based game to serve all platforms. The game contains multiple games and multiple teams and players. The functionality of the game depends on the names created for teams and requires a unique name for each team. Client’s personnel are not trained on how the system and environment should be set up.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* IOS, Android, and web apps require a specific software development kit.
* The API should be generated to work across all platforms.
* Each instance of a game, team, and player should have a specific name to limit instances of the game to one.
* Names for game and team should be unique names.
* Notification alert to be send to the team leader that the name picked already exists and ask them to create a different 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 SingletonTester to test and make sure that there is an existing instance of GameService.

Entity class is the parent class to Game, Team, and Player classes. Game, Team, and Player all inherit all required attributes from Entity class. 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.

**"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** | Mac computers could be used as a server, but the licensing is expensive, and you must have Mac computers 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. They are not quite equipped to excel at it. They lack the power for high end. It could be used for development though. |
| **Client Side** | There are great SDK’s ready for Mac, but it requires a Mac computer or laptop to develop Mac applications. This will increase cost and requires a developer who had expertise in developing swift applications. | The cost for this will be higher than other platforms and also possibly 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 XCode on it. All coding will be done using swift. | Python is very easy to use in Linux systems. And there are many IDEs such as IntelliJ IDE to code. | Visual Studio Code is most easy access and popular code editor Windows applications. Majority of languages could be compiled and run on Visual Studio. | 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 XCode 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**: My recommendation will be to use Linux as it is open source, and it provides more flexibility. Ubuntu and CentOS are two of the most popular Linux operating systems.
2. **Operating Systems Architectures**: Linux is compatible with many of the architecture systems and is very stable, although Mac is very stable and user friendly, but Linux works greatly with AWS and Kubernetes. Kubernetes allows system partitioning through the clusters.
3. **Storage Management**: My recommendation for storage would be either an SSD storage or cloud storage. Cloud storage provides higher storage availability and easier access to multiple devices. Also, Kubernetes provides storage for files and creating nodes for different files will help with storing them properly.
4. **Memory Management**: To be cost effective, I would recommend load watchers to help with adjusting the memory usage when lot of memory space is not needed to save space and when you need more space you could purchase more space. This way you don’t have to prepay for so much storage space when you don’t need them.
5. **Distributed Systems and Networks**: Having the system on the cloud will help with latency and provides less downtime if a part of the system requires update, then the entire system does not go down.
6. **Security**: As far as the security, which is an important part of system, I recommend having a role-based security level. This way, each role has access to an specific part of the system and they could not access the part of the system that are not related to them.