

<Draw it of 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 | <07/19/24> | <Sean Born> | For prototype software design. |
| 1.1 | <07/30/2024> | <Sean Born> | Added information to the Evaluation Section |
| 1.2 | <8/18/24> | <Sean Born> | Added in the Recommendations |

**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 wanting to develop a web-based game that can serve multiple platforms that is based off of an app that Gaming room has on Android called Draw it or Lose it. The game has multiple games which can have multiple teams each with many players. For proper functionality each game and team name needs to be unique.

## Requirements

*Teams and Game names must be unique. Contains a large stock of drawings.*

## [Design Constraints](#_2et92p0)

Different operating systems have different ways software can be developed.

Needs to work on 3 different platforms

Allows 1 or more teams for any platform

Game and team names are unique

Tells captain that name is already existing and that a new one needs to be chosen.

Each game is provided with a unique ID, team, and player to make sure there is only one instance to a single game.

## [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 main code id located in the ProgramDriver class. It uses SingletonTester to see if there is an instance of the game in GameService. The Entity class is parent to the game, team, and player classes. These classes will inherit the attributes within the entity class. When looking at the code the team can have a player but player cannot have a team. The game service has a game. GameService can only have one instance of the team, player, and game playing at any single time. The team names and game names must be unique.

**"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 is a server but is expensive and a Mac book must be used for development. | Does well in a web-based scenario for hosting games. Licensing is free. | Secure server and is very easy to use but licensing is very expensive | Does not do the greatest at being a server. They do not typically have the necessary power needed. |
| **Client Side** | SDKs are easy to use but development cannot happen on any other device than a mac book. Must be able to use swift. | Cost for Linus it highest when looking at development time. Must be able to use Python. | By far needs the highest amount of experience. | Needs developers that are well versed on developing apps for mobile devices because mobile devices display is very different from web based programs. |
| **Development Tools** | iCode  Coding is completed using swift. | Python  IntelliJ’s Ultimate IDE | Visual Studio  Any language can be used with the exception of C++ | Android Studio for Android devices  iCode for Apple devices(must have a mac book)  Unity (can be converted to either) Apple will still require a Mac for conversion. |

## 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 Linux as the operating system that will have the server to host the Draw it or Lose it Game because its flexibility and security.
2. **Operating Systems Architectures**: The architecture include the Kernal, hardware, and shell. The Kernal is very stable and is the bridge between applications and the hardware. The hardware include the CPU and any other physical devices. Lastly the shell is the actual interface the user experiences.
3. **Storage Management**: Since the game is a web based program I would recommend using a cloud based storage for the game. The google cloud would allow for storage of what Is needed to run the game. If storage needs to be expanded it would cost little to no money to expand the storage using a cloud.
4. **Memory Management**: Since the game will be coded using java as the programming language we can utilize the automatic memory management java has called the garbage collector to manage the memory storage of the game.
5. **Distributed Systems and Networks**: Since the game is web based we can use a web browser that is available on all operating systems. I would recommend using the browser google chrome for these purposes.
6. **Security**: Using google cloud services provides the infrastructure for security. It is also important to complete regular updates and patches to any issues as well had performing some sort of quality control on the program regularly.