

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 |
| --- | --- | --- | --- |
| 1.0 | 03/17/2024 | Jon Marvin | Included necessary information for the Executive Summary, Requirements, Domain Constraints, and Domain Model. |
| 1.1 | 04/03/2024 | Jon Marvin | Updated the Evaluation table |

**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)

Draw It or Lose It is an Android app that the Gaming Room wants to port to a Web-based platform capable of performing on any device. The game should be able to support multiple teams with unique names, multiple players on each team, and the ability to check for unique team names during creation. The Draw It or Lose It stock images must also be considered when creating the application.

## Requirements

The Gaming Room teams need additional software, and potentially hardware, to port the game across different operating systems.

## [Design Constraints](#_2et92p0)

Each OS (Android, web, iOS) require separate SDKs. The application needs to incorporate the use of multiple platforms at one time in one game. Teams should be able to join from any platform, regardless of if another platform is in use in the game. Game names and team names need to be unique, with the possibility for teams to be informed if a team or game name is currently in use.

## [Domain Model](#_8h2ehzxfam4o)

ProgramDriver and SingletonTester work together to ensure GameService is unique to each game. Entity is the parent class with Game, Team, and Player under it. The required attributes from Entity are inherited by Game, Team, and Player, and each must be unique to the others. Games can have Teams and Teams can have Players, but each Team and Player must be unique to each unique Game. GameService ensures that each game is 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** | Macs have the potential to be servers. Unfortunately, due to high prices, the cost would add up quickly. | With free licensing and high quality hosting, Linux is a widely used choice for servers. | Though licensing is expensive, Windows has secure and easy to create server options. | Mobile devices are not known for their server capabilities. Other options would be better suited. |
| **Client Side** | Mac development requires the use of Mac computers. That being said, the Mac software development kits are among the top of their respective class. | The Linux option would require the most time. It would also be a good idea to find a Python expert for this project. | .NET framework would be the most secure option here. A specialist may be required to assist with this. | App development is the top priority for Mobile. UI and UX would need to be specially created for any application created for Mobile. |
| **Development Tools** | Mac coding can be done using Swift, so long as the computer is equipped with iCode. | IntelliJ IDEA is the top performing IDE for Java and Kotlin and has a free community edition. Linux also is usually equipped with Python. | Windows Visual Studio and Visual Studio Code are the top choices for Windows. Packages for most languages can be installed through the program, and C++ is the most used language for Windows development. | Android Studio, Unity, Swift and iCode are the most prevalent options for Mobile development, depending on the type of phone. |

## 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**: Linux would be the best choice for Draw It or Lost It. Linus servers are open-source and customizable. They also allow for higher security at a much lower price. With fewer required resources, Linux has lower licensing fees.
2. **Operating Systems Architectures**: Kubernetes can be used to containerize applications. The Kubernetes cluster is more flexible than a virtual machine tasked with the same load.
3. **Storage Management**: Solid State Drives are lightweight and fast, but if they fail, they cannot be recovered easily. I recommend SSD main storage with HDD backups to ensure any issues can be resolved without pouring double the time into the applications.
4. **Memory Management**: Keeping memory in check is important for a game like this. Monitoring and reducing the memory at less busy times could assist in cutting costs and keeping the game running smoothly.
5. **Distributed Systems and Networks**: Keeping the game in the cloud helps with any kind of crashing. The node can be moved from one server to another (manually or automatically, depending on the server setup) and keep the game running during any kind of maintenance.
6. **Security**: To keep user information private from game and team information, role-based security would be the best bet. This security will keep game and team data separate, allowing the users freedom without gaining access to sensitive data.