

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 | 09/19/2024 | Michael Puckett | First Draft |

**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 has asked for a web-based program to be adapted from their Android program called Draw It or Lose it.

## Requirements

The software requirements are as follow:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The Gaming Room already uses Android for their currently game. They want it to be available on a web-based version. Java is the language for Android and Java is also the selected language for the project, which should help with the implementation.

## [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 UML model is below with it’s description below it.

"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.

This program uses polymorphism and inheritance with the Entity class as well as overloaded constructors. Encapsulation is used with adding teams and players. The program begins with a creation of a GameService instance. After a GameService is created, and only one may be created, a Game may be created. Each Game can have a Team added. Each Team can have a Player added. There is no limit to the amount of Games, Teams, or Players.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | There are servers available for Macs but the price can be expensive | Linux is open source so the cost is going to be cheaper. | Windows offer a familiar GUI and many applications run great on windows, but the price for a server can be high. | Mobile devices are limited by hardware since they are smaller and more packaged. |
| **Client Side** | Mac does not run on Java, it runs on objective-C or Swift, which is not as well know as something like Java, which could cost more to program | Programming for linux would be easy as those languages it uses are more well know (Java, C, Python) | Windows is a highly preferred OS and is used by a majority of computer users. The languages (C# and .Net) are common. | Code developed for Android uses Java which would work well for this program. |
| **Development Tools** | Xcode is common for Macs and is relatively cheap. | There are many IDEs on linux that are free or very cheap. | Microsoft’s Visual Studio would be the main choice developing on Windows. | Like Mac, Xcode is common for iOS and is relatively cheap. |

## 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 an excellent choice for The Gaming Room. There is good security as well as numerous options. Licensing would be cheaper for costs. Linux is also a very common server platform so it has a lot of tools to offer.
2. **Operating Systems Architectures**: As this program won’t require extremely fast speeds since everything is loaded into RAM before the game starts, doing a cloud based database server would be good and the client’s system could manage the front end (displays).
3. **Storage Management**: A cloud based storage for the server side would be recommended. Even though there will be charges for storage used, speed with low latency is not a necessity.
4. **Memory Management**: There will not be a need for a lot of memory for this program. At most, the client side will only need to store a few images, which won’t take much RAM for this program.
5. **Distributed Systems and Networks**: If this is a cloud-based service, then the benefit of that is that a cloud service that can shift services among different deployment would do well as it can help to avoid large outages during the game.
6. **Security**: Users will only be permitted certain rights within the game. No user will be an admin, and only some users (such as a team-captain) will be permitted to make changes to their team.