

# **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 | 01/22/2025 | Zachery Davis | First Draft |
| 1.1 | 02/05/2025 | Zachery Davis | Second Draft |
| 1.2 | 02/21/2025 | Zachery Davis | Third 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 Rom needs us to create a web-based version of their game Draw it or lose it that currently is a Android-base game. The game will allow multiple teams with multiple players per team each instance of the game.

## Requirements

*-Multiple teams*

*-multiple people per team*

*-unique game token per instance of game*

## [Design Constraints](#_2et92p0)

## The game currently only exists for Android platforms so we must take in consideration that OS need to be compatible with both. The program must be designed to allow the existence of both Android and web-based applications. The program must be able to allow multiple teams to exist. Each team much be a unique instance within that specific game.

1. Security
2. Multiple Languages
3. Handling Multiple Request
4. Player Validation
5. Resource allocation

**Security**

The client would like each player to be differentiable using a multifactored authentication method should be considered to handle login and authentication purposes

**Languages**

The current program needs to be adapted to multiple languages to integrate and operate on multiple types of devices

**Multiple Request**

The server must have the ability to handle multiple concurrent request from each of the users available platforms

**Player Validation**

Applications need to verify users and the privileges that they are assigned

**Resource allocation**

The server needs enough storage to hold the games current collection of photos and should be expandable to allow more photos to be added as the clients need.

## [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 contains the entry point which contains the SingletonTesterclass

The SingletonTesters uses a Singleton pattern to validate if an instance of the game exist already.

The GameService class contains a list of games and has the ability to create new games with the associated teams and player tokens.

The entity class is the parent class to the classes Game, Team, and player.

The game class extends Entity class, which has a list of teams for the game token and can create a new team and ensures that teams token is not already established.

The team class extends the Entity class and has a list of players and ability to create new players. It has a verifier that ensures each player is unique.

The player class extends the entity class and creates the player’s token.

**"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)

There is many targets for potential growth like Mac, Windows, Linux, and mobile platforms. Server-side choices are agnostic of client side. For example, choosing a Linux based server means this could still run on a mobile device if set up properly.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | macOS can be used for server deployments but this requires Apple hardware which can get expensive and has limited features when compared to other OS options. Since this is uncommon a team will have to be found or trained | Linux has a variety of options such as Ubuntu, Debian there are Enterprise Servers which require a subscription. This requires a high level of expertise but training or finding a team should be much quicker. | Windows also has a variety of choices and tools such as the Datacenter Edition windows Server 2025 which can support unlimited Virtual Operating System Environments and GSM which is a Game Service Manager but costs can run high. A specialized team will most likely not be needed. | Mobile apps will rely on backend services hosted on cloud platforms such as Firebase or AWS which will require licensing fees. Hosting a server from a mobile device is not advisable as the program will have to be built and the scalable nature of this will be called into question. Highly skilled teams will have to be found to achieve this |
| **Client Side** | You would need a Mac computer with Xcode and familiarity with SWIFT is less then the other languages. Mac is also a niche software that doesn’t capture much of the market. Less common then other languages but will be feasible. | Linux is capable of multiple language compatibility from jaca, C++, or Python these are the most common languages in use today.  This should have minimum impact on the team as these languages are widely used. | Windows uses C#, .net, or Javascript with Visual studio as the primary IDE. Windows has been a multi user platform for decades and holds the largest share of PC users in the world. This should have minimum impact on the team as these languages are widely used. | Android SDK is Java base. Mobile devices are not built to be multi-user.  A highly skilled team should be sought for this endeavor. |
| **Development Tools** | Xcode which supports Swift and Objective-C  Xcode is listed $99 dollars per user/year | Visual studio and most popular languages, free just technical knowledge is needed | Visual Studio and compatible with most popular programming languages. Visual studio ranges from $99-$499 a month depending on version acquired | For cross platform tools React Native and Flutter for specifically Android Android studio and for iOS XCode |

## 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**: The operating platform I recommend is Linux, as it’s highly customizable, secure and cost effective. These attributes will be crucial for expanding Draw it or Lose it.
2. **Operating Systems Architectures**: Lunix offers modular architecture allowing customization and optimization it supports the widest range of software and hardware and the nature of open source will keep costs down.
3. **Storage Management**: I would recommend using google Cloud storage as it’s scalable, and high performance.
4. **Memory Management**: Linux uses management techniques that include virtual memory and memory allocation/deallocation memory managers can be implemented to optimize memory usage
5. **Distributed Systems and Networks**: A client-server architecture should be used.
6. **Security**: Security measures should include encryption during transmission and storage, regular security audits and strict access controls. As well as Multi-factor authentication.