

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_gjdgxs)

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/17/2024 | Connor Holohan | Original documentation for Draw It or Lose It |
| 2.0 | 11/30/2024 | Connor Holohan | Updated Documentation for “Draw it or lose it”  Evaluation table. |
| 3.0 | 12/15/2024 | Connor Holohan | Updated Recommendations |

## [Executive Summary](#_35nkun2)

Client named "The Gaming Room" has requested development of a game named "Draw it or lose it". It is based on a classic tv game show called "Win, Lose, or Draw". Currently the application is only available on the Android platform. The client has outlined several software requirements that need to be met for the application, which will be reviewed in this document.

## Requirements

-Games will allow for single or multiple teams

-Teams will have multiple players

-Checking for unique ids within team or player names, so that no names can be used twice.

-Only one instance of a game can run at one time. Use unique Identifiers for each instance.

-images are rendered from a library of images

-picture rendering time to be completed at the 30 second mark

-If team A fails to guess the picture within the initial 30 second rendering time, other teams will receive 15 seconds to attempt to guess the picture.

## [Design Constraints](#_1ksv4uv)

-The Customer has requested that the game function on multiple web platforms.

-Only one instance of the game can run at one time, requiring unique identifiers

-The pictures must be capable of being rendered within the time limit

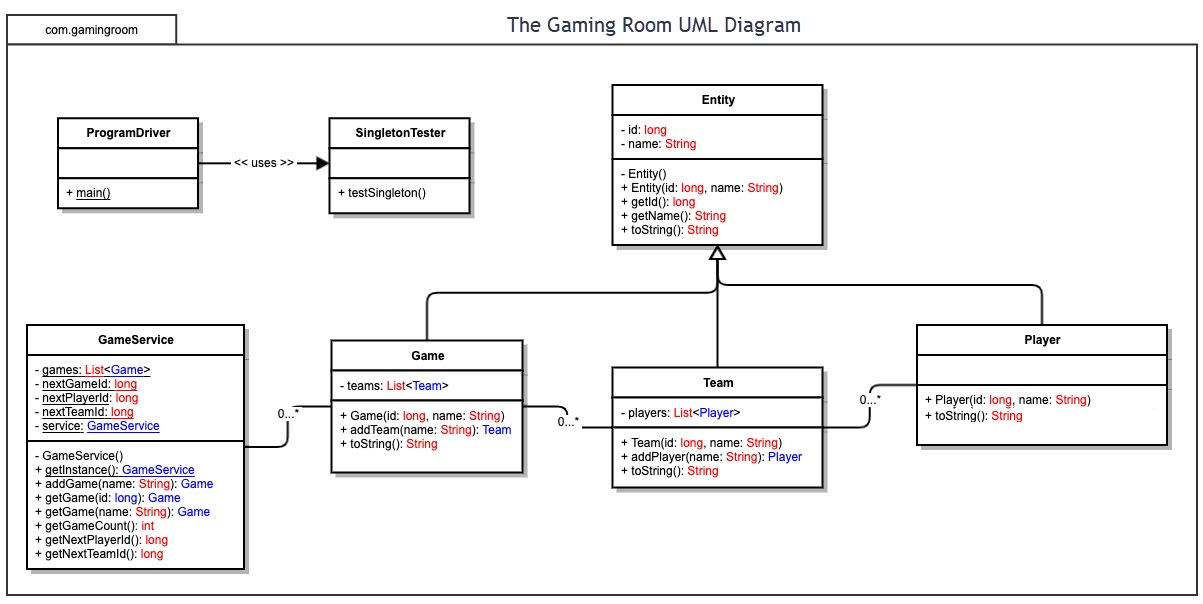
-The game must be compatible with multiple players.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

****

**"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 is the UML diagram for the current program. The Entity Class is a superclass with the subclasses Player, Team, and Game. The Subclasses Player, Team, and Game are extensions of Entity and inherit the attributes ID and Name. These are accessed via the superclass method. The game service has a composite relationship with Game, Team, and Player classes. The program driver is our main class and utilizes the singleton tester to test the program.

## [Evaluation](#_z337ya)

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** | High security, but used less commonly than linux or windows.  Expensive with limited hardware choices. | Largely Used, high functionality, scalability. Cost effective.  May require specialized skills to utilize. Most likely the best server side option. | Frequently used, has a robust ecosystem for high compatibility.  Licensing costs can be high. | Limited. Peer 2 peer functions and a few other capabilities. Mobile is generally a poor choice for hosting or maintaining servers.  High mobility of servers? High diversity in platforms will increase cost. |
| **Client Side** | Very large user base, customers will be familiar with the platform. High accessibility and user friendly UI.  Proprietary development tools. | Free use, but much more complicated for the average person to utilize compared to the other options. | Very low level of expertise to use.  Platform cost is greater than linux. | Costs will scale with the number of mobile platforms targeted. Platform specific languages will require expertise in multiple mobile coding languages. |
| **Development Tools** | Supports C++, Swift,Objective C.  Primary IDE is Xcode. | Most major IDES.  Eclipse, Visual studio. Robust language support.  Virtually anything | Visual studio primary IDE. Can run most major IDES. Robust language support. | Xcode for IOS  Kotlin, android studio for Android.  Cross platform tools exist. |

## 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 client has requested that the game be available in a web based platform. This will allow us to utilize any operating platform with internet access. It would be the recommendation of this review to utilize cloud based networking to minimize the maintenance costs of operating a server. The scalability of a cloud based system will also be superior to server based, minimizing the need of the game owner to purchase additional servers or space.
2. **Operating Systems Architectures**: Microsoft Windows has all of the required tools. Generally speaking all programmers who can effectively use linux to code, will understand how to use windows. The reverse may not be true, and utilizing such a common operating system will allow us more developer real estate. It will also be capable of utilizing CSS and HTML.
3. **Storage Management**: Microsoft Azure, for the capability to utilize cloud storage. However Microsoft does come with built-in storage management functions. Disk management, Disk cleanup, and storage sense can all be utilized to clean up and maintain storage space. Microsoft's suite of storage management tools creates a robust and user friendly storage and storage management system.
4. **Memory Management**: Microsoft utilizes a kernel mode memory manager. This allocates memory virtually and dynamically. If further efficiency is required, a large scale data processing tool could be utilized, such as apache spark. This however is unlikely to be required given the scale of the project.
5. **Distributed Systems and Networks**: This would only require a cloud network or centralized server. Since the game is web based we won't encounter many compatibility issues. A centralized server will handle synchronization, as long as the client is able to connect with a steady internet connection. The strong scalability of utilizing a cloud based network for scalability will provide a more permanent solution, as opposed to the stop gap of purchasing a new server, or more server space, as the player count increases.
6. **Security**: Strong password policies and data encryption will be required for maintaining secure connections. Access control and ID tracking should be implemented, and will have little impact on performance. If additional security is required 2 factor identification could be implemented, although considerations should be made as to whether or not this feature is mandatory.