

Draw it or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

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

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)4

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/18/2022 | Matthew Tietz | Prototype software, completed a full report |
| 1.0 | 6/4/2022 | Matthew Tietz | Continue work on Server, Client, Development chart. Also corrected some error within the other section in the report. |
| 1.0 | 6/16/2022 | Matthew Tietz | Finished 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 currently looking into having a web-based game that serves many platforms. The game room currently only have one application that is on android only as the company which to expand their fan base. This game makes up of many players which could be in many different teams, to work properly each team has have a unique name. The Gaming Room has a huge library of stock drawings for the game Draw it or lose it, However the staff do not know how to set up the environment.  
  
An API designs that can interact with all 3 platforms

Team names must be unique as well as the Game names to minimize system confusion

Allow the system to tell the team a name is already taken so no same team names exist

Use a different ID for all games, team, and players to limit copy of the same information.

## [Design Constraints](#_2et92p0)

Design constraints would be more focused on the server part of the game “Draw It or Lose It” Was created to be an android application only. The environment the application uses would have to be changed to help support a web-based game that can support all 3 OS while still connection to the Mobile application for cross play. While expanding the player base server balances should be taken into consideration to help with stand to new load on the servers, if not then it can cause playtime lag or even crashes for the clients. Systems should be put in place to help limit the load that would be placed on the servers. an example, If a devices would try to create a game and connection was lost, The request should be discarded after a short period of time to help limit the resources usage.

## [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)

ProgramDriver contains the main method which drives the program together. The programDriver Is Directed association with the singletonTester class to test to ensure there is not another instance of the GameService. Entity is the parent which is inherited by Game, Team, and Player class. GameService can have a Game, but a game can’t have a GameService. A game can have a Team, but a team can’t have a game. A team can have a Player, but a Player can’t have a Game.

**"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 can be used as a server however, running a mac server can be very expensive.  Not a good option for large scale distribution.  Depending on the hardware that would be using to host the server can create clutter due to some of the Mac systems designs. | Linux can be used as a server and can cost lest to run. Linux is well equipped for running a server. Linux is not very user friendly which could be difficult.  Linux Distro can be designed to fit the needs of the company.  Linux is open sources which can be very beneficial on the server side. | Window servers are very secure and stable. It also very user friendly and very common for most people can use windows. Licenses are very expensive.  Window server set up can be complicated due to the amount of configuration that would need to be done to work properly | Technically can run a server on a mobile device however it is not recommended. Mobile device server can be unsecured.  Since this device is wireless it can cause connection disruption.  Not the best device to handle large amount of data at one time. |
| **Client Side** | Mac has a good software development tool but must either own a computer with macOS or a vm with macOS on it. Macs can be very expensive on the client side and mac is not optimized for gaming purposes. | Linux software development would be the most time consuming. The developer would have to have experiences. Linux is free for the client side and comes in many other Linux distribution to choose from | Windows can be very adaptable for development but would require the most expertise to create a secure application. Windows is known in the gaming world and is the most supported OS. | Mobile devices development would require a lot if time and money as the web page design would have to be different to fit into the display of a mobile device  Require.  Require little configures versus other OS systems |
| **Development Tools** | Mac book has Xcode which is an IDE that is designed for mac which uses Swift.  C++ is hard to use because of a compiler issue with this OS. | Linux comes with python as most application is created with python. Using Linux could be difficult. Linux has a wide variety of development tools | Visual Studio is what windows come with visual Studio and best used with C++ and C#. If system has enough storage space other languages could be installed like Python. Using 3rd party tools would give the user access to most programming language | Developing on a mobile device is possible but could be extremely difficult and would require someone with the know how to do it.  Library for applications has been designed to be very secure. |

**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 server should be hosted on a Linux distribution as it would be the most cost-effective way. Linux has been shown to expand easily as more comes to the network. Linux would require more technical knowledge however the benefits it will bring will make it worth it.
2. **Operating Systems Architectures**: Linux is a very stable and secure system. Linux allows the user to have full control over the system and what can be used. This could help limit unnecessary resource usage.
3. **Storage Management**: Storage all depends on how much data the application would need to have and how much data it will collect. Data could be stored on either an HDD or an SSD depending on The Game Rooms desire. HDD is the most cost effective due to the cheaper price for more storage vs an SDD however, SDD have a lot faster load time which could give them access to assets faster. Linux also offers a software called “XigmaNAS” which can help proved an easy-to-use data management tool for server usage.
4. **Memory Management:**  A method to use would be memory mirroring which is a process of having physical memory is split into two parts. The server will use the first part of the memory and use the 2nd part only if the 1st part is unavailable or full to help minimize disruption. This can help limit overuse.
5. **Distributed Systems and Networks**: Having a 2nd server would be the best rout of action as it could serve as a backup if the primary go out or can help balance the load on the system. Doing this will allow you to run the game on server side which would allow all operating systems to have access to the information they need. Another method would be using a cloud service which would help balance the distribution the game Draw it or Lose It to all the clients which also provides easy access to the information they would need to use the software. The use of Kubernetes Kubectl can help prove and easier management of the system.
6. **Security**: For the server it would be important to great separate roles to limit the access of the team and players which is called role-based security. Different roles for Admin, team, player, and users would need to be created to ensure that the user can gain access to information they are not supposed to. Another thing is to ensure the is clean and follow best practices to minimize exploits that could be performed. Firewalls implemented on the network can help prevent unwanted access to parts of data on the network that shouldn’t be accessed.