

Draw it or lose it

# **CS 230 Project Software Design Template**

Version 1.03

## 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)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <09/24/2022> | <Russell Gomez-Martinez> | < Initial Software design > |
| 1.01 | <09/25/2022> | <Russell Gomez-Martinez> | < Software Design Revision to “Recommendations > |
| 1.02 | <10/09/2022> | <Russell Gomez-Martinez> | < Evaluated the most effective method of implementing a server for Draw it or Lose it > |
| 1.03 | <10/22/2022> | <Russell Gomez-Martinez> | < Added Recommendation section for Software Design > |

**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 requested our assistance in the creation of a web-based application called Draw it or Lose it. An Android version already exist however they are looking to expand the game to more platforms and are looking to find the most effective way to branch out with their software. An additional responsibility aside from streamlining development to a specific OS is to implement a game instance in which there are one or more teams and 4 players in each team with time constraints on achieving points throughout a 4-round session.

## [Design Constraints](#_2et92p0)

Each OS has different constraints of their own, choosing the proper development kit is vital

Only one instance of a game at a time

Timing intervals of 15 and 30 seconds

Unique Identifiers for Teams and Games to maintain only one instance

Rendering Stock Imaging

## [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 Entity class is the cornerstone of this program as it is the parent class to Game, Team and Player

The ProgramDriver executes the main method while the SingletonTester will determine if the Singleton Method has enabled only one instance of a Game at a time. Game Service is associated with Game, Game with Team, Team with Player. With out a previous association a class can not exist (eg. Team can not exist without Game.) In order to have a functioning program all classes must be utilized to avoid method errors, the same logic were to go with that there are no Teams without players.

**"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** | Compared to the vastness of other servers and the desire to not exclusively work in MacOS, a server in this OS is not an option due to its own fallacies. | Linux is universal but niche, hosting a server would be effectively free. Security is fantastic and there are multiple languages that can be used. No official support however. | Windows is the most popularly used Operating system and as such has some of the best support if we run into an issue. | Mobile Devices today have plenty of power packed into them, not enough to act as a server however. Not useful for what the company wants to accomplish. |
| **Client Side** | Cost would be a major issue with Mac as Mac devices would have to be used unless we Bootstrap every computer in the office with Mac which is just inefficient. | The time to develop in Linux would take the longest, however it would most likely prove to be the most profitable of all options due it being open-source. | Windows has plenty of support for development and has multiple tools at it’s disposal however expertise in Window’s specific web development would be needed. | The good thing about this is that the Android App already exist, all that would need to be done is transfer this data to Apple OS to access most mobile platforms. Mobile Developers would have to be sought out to do this. |
| **Development Tools** | Mac has Ruby and Swift but ultimately can’t transfer to Linux/Windows without virtual Machines, this is also Vice Versa. | Linux has plenty of great open source tools and IDE’s to include ATOM and VSC. Linux also supports multiple coding languages and can transfer across platforms. | Visual Studio, SQL server, C# and .NET are all good tools that Windows can utilize to accomplish its task, VSC and plenty of other IDEs. | Xamarin can be a quick way to get an application developed and used cross platform, for this you’d need someone who can use .Net efficiently. |

## 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**: I believe Windows OS would be the optimal Operating Platform due to its popularity and its plethora of tools available both Server Side and Client Side which can made compatible to most platforms.
2. **Operating Systems Architectures**: Windows comes in a 32-bit and 64-bit architecture and developing in a 32-bit architecture would enable the app to run on the greatest number of devices possible. Windows also has a Kernel Mode and User Mode, where User mode allows for restricted intuitive use while Kernel mode allows unrestricted access to hardware, enabling detailed implementation of hardware specific processes.
3. **Storage Management**: SSDs allow for quicker allocation of memory, getting a large amount of RAM in the system will also help with sorting caches. Using an SQL and a Node.js can help with portability of the software.
4. **Memory Management**: A variable partition with a first fit basis would be ideal for managing memory on a first come first serve basis, the game’s memory consumption shouldn’t be high enough to cause issues. Paged memory also prevents fragmentation and allows for virtual memory processes should the use want to run more processes other than the game with efficiency
5. **Distributed Systems and Networks**: A distributed network/peer to peer connection enables multiple points of communication preventing the loss of any single participant in the network from affecting the communication of an application. Through this system there can be multiple application executables with launchers specific to the OS the application will be running on. The likely hood of a total outage drops significantly due to the use of multiple participants.
6. **Security**: A role-based security system will be best to prevent parties from accessing information they have no business looking into. It also allows separation of roles and different protocols for each role providing a sense of organization for application management and debugging.