

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 | 03/13/2023 | Cody Lockerby | Initializing and first draft. |
| 1.1 | 4/2/2023 | Cody Lockerby | Completed evaluation section |
| 1.2 | 4/16/2023 | Cody Lockerby | Recommendation and complete |

## [Executive Summary](#_sbfa50wo7nsh)

This design summary is for our client, The Gaming Room, and their aim is to create a web-based game based on their current game called Draw It or Lose it. The client wants their game to be available as a web-based application and currently is only accessible on Android systems. Each game consists of 4 one-minute rounds. If a correct guess is not made opposing teams are given 15 seconds to guess. Drawings must be rendered at a steady rate and completed by the 30 second mark.

## Requirements

Requirements from the client are as follows:

* Each game will be able to have one or more teams included.
* Each team will have multiple players assigned to it.
* Each game and team name must be unique so players can see whether or not a name is currently in use when selecting their team name.
* Only one instance is allowed in the memory at any given time.

## [Design Constraints](#_2et92p0)

The design constraints I have found are as follows:

* Client wants game to be web-based to access as many people as possible.
* Game consists of four rounds each round being timed to 1 minute each round
* Client needs drawings rendered at a steady rate and must be fully complete at the 30 second mark.
* If no correct guesses are made the remaining teams have a chance to guess with a timer of 15 seconds.
* Each game having multiple teams included.
* Each of those teams having multiple players.
* Ensuring each team name is unique so no duplicates between games.
* Creating unique identifiers for each game, team, and player.

## [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)

Below is the UML model previously created. Some of this program has already been created, but we will be required to adjust it to what the clients needs are currently. We will need to create the Entity class in which the Game, Team, and Player class will inherit from. The Team, Game, Player, and additionally the GameService classes all share association of 0 or many objects between them. Our main will be in our Program Driver and this will be where our code is written and ran as it contains all properties of the classes listed before. Also included is the Singleton Tester, and this is what we will use to help with part of the constraints as it will test for duplicate team and player names and will only allow one game in the memory as required.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac has become increasingly popular in the last 10 to 15 years as their technology improves. From their website they have discontinued to new uses of their MacOS server. Any existing clients can continue use but who knows when that could stop. From what else I could find it seems to have been a higher cost with an easy set up. It also had fewer virus attacks and good support, but it lacked hardware customization. | Linux is less popular in today’s technologic climate but still an option we should investigate. From what I can find this appears the be the cheapest option as we would not have to pay for licensing fees and more reliable than windows or mac. It has low requirements so it can prioritize performance rates. While the cheapest it also seems to be the least user friendly. It also seems that it may be more intricate to upgrade between major versions. | Windows is the middle ground for pricing as it relates to computer OS. We would have to pay for licensing fees but not as expensive as MacOS was. It is the most user friendly as most individuals have some experience with windows. Windows does support third party apps, but it is very prone to cyber-attacks because of its popularity. | Not much could be found for server side for mobile apps. One big disadvantage is that there are many mobile devices we will need to be compatible with. Unless we make the choice to only be compatible with Android and iOS we will need to research more like Amazon app store and so on. If we can do that an advantage is that we could perhaps reach a larger audience of those without computers but may own a tablet or phone. |
| **Client Side** | Developing for MacOS will not break the bank and comes with a good support team. Popularity is increasing so some extra work may mean more customers. A downside is having to develop something and spend more time for specifically MacOS. | Linux has more of an open-source situation. We would also need to find someone with knowledge of Linux since it is not as popular. Python seems to be the most popular which is good and could cut down on development time. | Again, this is very user friendly and easy to do test runs but has the chance to be less safe. Development time would probably be the lowest. It would be very difficult test for Mac on windows. | We will need someone with expertise in mobile development since it is a beast of its own. This could prove to be in development longer since it is so much different than web-based application. |
| **Development Tools** | Apples website recommends swift for writing Mac applications. Also, able to use cross platforms languages like python and java. | C++, Python, and Java have become the most popular for Linux programmers. So Visual studios, PyCharm, and Eclipse will be needed. | C# and C++ are popular among windows programmers so we will need Visual studios. | Mobile apps are more difficult because we will need to adapt code for multiple variations of phone. For example, if we code for an Android we can use Java but that code will need to be calibrated for iOS for it to be as efficient as possible. |

## 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**: For the operating platform my recommendation would be to use Windows. This platform is the most user friendly while also allowing developers many tools to be able to create unique and creative programs. Windows is also easily upgraded. It will cost more than a Linux system, which was the second choice but will not cost as much as a Mac system. Another reason for selecting this system is the Azure system within Windows which I will explain further in later sections.
2. **Operating Systems Architectures**: The architecture of a Windows system is very simple and efficient. It utilizes a two-mode system: a user mode and a kernel mode. In the user mode is where almost all the application will run and allow interface from a user. Kernel mode is where most of the functions of the core system will run to allow space and more efficient running of the application. Most of the drivers run in kernel mode with a few exceptions.
3. **Storage Management**: With how technology is advancing I recommend using a cloud-based storage option. There are so many on the market from Amazon to Google storage but this is one place where Microsoft Azure comes in. Azure has a cloud storage system that is included with its service. It is backed by Microsoft and compatible with Windows systems. Windows is also equipped with built in storage options that will help keep storage clean and tidy while Azure will store the bulk of what you need for Draw it or Lose it.
4. **Memory Management**: Windows will utilize both physical and virtual memory to ensure smooth play. In the case of memory management this is where kernel mode will come in to help in a big way. Behind the scenes kernel mode will manage the allocation of memory both physically and virtually by using the memory manager.
5. **Distributed Systems and Networks**: This is another critical part where Azure will come in to use. The Gaming Room will want this game to be able to be played on multiple platforms. Azure is a cloud based system and will be able to communicate with all platforms with access to an internet connection. Crashes and outages happen to even the biggest Triple A games so they are something that will happen no matter what. Azure is backed by Microsoft and has a team meant to deal with outages and will be able to help make sure the game is operational as much as possible.
6. **Security**: This is the section that almost made me consider switching operation platforms. Windows does not lack security, but it is the lease secure of the three main systems. To combat this, we can again lean on Windows Azure. Azure has high end features built in to protect its clients with unique features only available to Azure. We can also ensure the Gaming Room clients are making their passwords as secure as possible and enable two factor authentication.