

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 |
| --- | --- | --- | --- |
| 3 | 10/05/2025 | Travis Erwin | Third Edit: Server 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 would like to expand their reach with the game “Draw it or Lose it” by releasing it as a web-based game that is available on all platforms. It is only available on android devices in its current form. The game is similar to Win, Lose, or Draw where teams guess what is being drawn in a competition. Instead of team members drawing an image for others to guess, drawings are steadily rendered to completion in 30 seconds. If the time expires without the team guessing the puzzle, the remaining teams have one guess each to solve the puzzle within a 15 second time limit.

The new version of this game will be playable on multiple platforms, host multiple unique teams with multiple unique players, and unique game instances.

## Requirements

* A single game can have one or more teams involved
* All teams will have multiple players assigned to it
* Game and Team names will be unique, allowing users to check if a name is in use or not.
* Only one unique game instance can exist in memory at any given time. It is accomplished by creating unique identifiers for each game, team, or player.

## [Design Constraints](#_2et92p0)

* Games must be web-based.
* Games must be compatible on all platforms.
* Must Handle multiple players assigned to teams in a game instance.
* Data Security
* Minimizing latency.
* Resource management.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram shows how each class and driver in the program works with each other. There is a base/parent class called Entity that provides the ID and Name attributes that will be inherited by the sub/child classes Player, Team, and Game.

The three sub classes that extend the entity base class are Game, Team, and Player. The Player class represents an individual player and creates an ID for them. The Team class holds the list of player objects to help create a unique team. The Game class holds a list of team objects to help create a unique game instance with them.

The GameService class manages active games, it holds a list of games and Tracks available game Ids, player ids, and team ids. It is equipped with the singleton design pattern that ensures only one unique game can exist at a time. It creates new games with names that don’t already exist, returns specific games, and number of active games. It does the same with player and team IDs.

The Program driver holds the main function of the program. It calls in the GameService and adds games, teams, and players.

The SingletonTester is a class that tests if the singleton patterns in the Game, Team, and GameService classes are working correctly

Object Oriented Programming principles are demonstrated in the UML diagram showing inheritance as the Game, Team, and Player instances all inherit from the Entity class. Each class has private attributes with public getters and setters demonstrating encapsulation. Abstraction is seen as well with keeping info hidden in some cases. For instance, Entity itself is an abstract class. You never play as “Entity” but it is the parent class that can define an object with an ID and a name. The game service manages games privately and that isn’t seen by the other classes.

**"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.

| **Server Side** | Macs are quick, stable and secure. They are user friendly and connect seamlessly with other apple devices.  The hardware can be expensive doesn’t offer much in the way of upgrading making scaling somewhat difficult and expensive. | Linux tools are free and open source. It is highly secure and stable and efficient with resource management. It has a wide range of cheap server software as well.  On the downside it can be complex to use and not compatible with some software. | Windows servers are user friendly and support many tools for web hosting.  It is easy to navigate for admins, and widely compatible. Security is not as strong as linux or mac, and licensing can be costly. | Using mobile devices a server is not typica and would require cloud hosting. The only advantage would be mobility. The stability is limited and control over the hosting environment could be tricky. |
| --- | --- | --- | --- | --- |
| **Client Side** | Costs can increase with hardware. Time can be a factor when making a cross-platform application. Expertise in cross platform frameworks will be needed to ensure cross-platform compatibility is seamless. | The cost is low since the OS is open source. Time may be higher to test cross platform compatibility, and expertise in open-source frameworks is needed. | The costs are higher than linux but likely less than mac. Time is lower with windows large development community. As this product is intended to be cross platform, then expertise in cross platform frameworks will be necessary. | Costs will be high due to the need to use cross platform frameworks. Time can be higher for needing to test on device sizes. Expertise in mobile languages and cross platform framework will be necessary. |
| **Development Tools** | Cross platform languages that mac supports would be Java, JavaScript, and HTML. VS and Eclipse could be viable tools used on mac for this project. | Viable code we could use on a linux based server for this program would also be Java, JavaScript, and HTML. VS code and Eclipse would also be ideal IDEs for the project. | Java, JavaScript, c#, c++, python, and HTML would also be the go to here for the language we could use. IDES like eclipse and Visual studio are great for the project. | JavaScript and HTML would be used if hosting on mobile. IDEs like Android studio or Xcode could be an option. |

## 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**: It is our recommendation to use a windows-based server for Draw it or Lose it. Its versatility, large pool of programming languages and developing tools will be useful for scaling up as the game grows.
2. **Operating Systems Architectures**: Using a client-server architecture style is recommended. Managing the core logic and player data on the server side while the client handles the game interface works best for online games. Windows will run as a process with a thread pool handing requests.
3. **Storage Management**: To store the game source code, player info, and images on the server side, a physical drive with cloud backup service is how we could do this with a windows based server. The images will be stored in a tree-structured directory. The user data will be stored in a SQL database. For client-side storage, we would use localStorage, sessionStorage, and cookies to handle data.
4. **Memory Management**: Windows uses virtual memory paging, swapping to keep the server’s workload smaller and streaming images instead of buffering to ram. Using dynamic loading to load routines into memory only when they are needed will allocate assets efficiently for a game that will have multiple players and multiple instances.
5. **Distributed Systems and Networks**: Clients will connect to our Windows server through their browser using RESTful API to communicate cross platform over HTTP. Multiple servers acting as backup servers for each other can help mitigate any full crash on a local server level. For ISP outages on a server side, we will need a client-side response to alert the user that systems are down and will be back up as soon as possible.
6. **Security**: Role based security will be the best route for all game mechanics and enforcing checks of roles on the server side after every user action. The principle of least-privilege will be implemented in all roles from users to admins. For all users, there will be a multifactor authentication login process to prevent any bad actors from gaining access to any sensitive information. On the server side we will run Windows Defender Firewall at all times, patch systems regularly and keep backups encrypted.