

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 01/25/2025 | Mack Yeddo | The revisions completed in this document are the cover page, the summary, revision history, requirements, design constraints, domain model, system architecture view, and development requirements. |
| 2.0 | 02/08/2025 | Mack Yeddo | Updated Evaluation of Server Side, Client Side, and Development Tools. |
| 3.0 | 02/22/2025 | Mack Yeddo | Complete Recommendations Update. |

**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)

This project design document for The Gaming Room is to propose an outline for developing a game application “Draw It or Lose It”. This game will take images from a library of stock drawings as clues in which the drawings are shown at a steady rate of 40-seconds. This lays out taking the already working android app into a web-based game that serves multiple platforms.

## Requirements

* *The game must run through a web-based platform so that it is available to play on multiple platforms.*
* *The game should have the ability to have one or more teams involved.*
* *The game and team names must be unique so that only one name can be used at a time in a game.*
* *There can only be one instance of the game at any given time, which will be by creating identifiers for each instance of a game, team, or player.*

## [Design Constraints](#_2et92p0)

* The system needs to have unique names
* Game must have the ability to have one or more teams playing at a time
* Only one game can exist in the memory at any time
* Unique identifiers for each instance of game, team, or player
* Each team will have multiple players

## [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 Diagram below shows all the classes that are in the game application Draw It or Lose It. The Entity class is the base for all other entities. Some common attributes between the entities are name, ID, and Player. Every entity is listed as it will have different identifiers or names to differentiate per game. Game, Team, and player entities extend from the main Entity within the Game Service. The class Game Service shares a relationship with Game Class which means that it directly manages it. The program Driver class is where the main function is, which within the class is the Singleton instance. Being the Singleton instance, this means that while the main game function runs there is only one instance of the game active at any time. The Program Driver class uses the Singleton class, so if there is no instance than the game is not running.

**"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 is able to host the web-based application and have up to a few thousand players per server instance. Mac has good security when it comes to their products. Mac is tied into the apple ecosystem which includes iPhone, iPad, and MacBook. Mac does tend to be on the higher cost of things and there is usually less customization available. | Linux has good performance and is also open source. This gives a lot of flexibility with customization. Linux is on the cheaper side but does come with a larger learning curve. Linux will have a server-based method that is free, Linux also can support a few thousand players per server instance on its open source. | Windows is a well-integrated ecosystem. Windows has a lot of tools at hand but lacks customization because of being dependent on Microsoft. The cost is higher on this platform. Microsoft will have licensing fees which can range in price and also get pretty pricy. Having a windows server can still be beneficial even with the extra cost. | Mobile Devices are very diverse with many features like touch screens, a multitude of applications, and a lot of customization. They are, however, dependent on the service you get. When it comes to mobile devices these usually connect to servers hosted on other specific platforms not the mobile devices themselves. These fees could vary greatly depending on the choice of platform. |
| **Client Side** | Compatibility issues because certain tools are specific to mac. Mac does have a good user experience but finding compatibility can be hard. Because of the compatibility issues with certain tools this could mean taking longer to trial run the program and optimize it for the users. | Very flexible within customization and performance. Can have issues with compatibility and security because of being open source. This would need to be tested across multiple browsers and screen sizes to make sure there are enough setting changes available to have a good user experience for the players. | They have a great suite of tools on the windows platform. Compatibility isn’t usually an issue because of how well adopted windows is. Costs could be cheaper because of the knowledge out there already for this platform. Extra time may be needed for testing multiple browsers to make sure the application functions correctly on each and that there are enough options available for the different browser settings. | There are a lot of different mobile devices so support could be difficult with the amount of customization. Users can be very involved at all times being a mobile device. IOS phones have more security with less options than android phones so someone familiar with all the nuances of IOS devices will be beneficial. This may take extra time to get the settings right from porting over from android. |
| **Development Tools** | JavaScript, HTML, PyCharm, Github, CSS. Multiple development teams will be needed that are proficient in apple products. | HTML, CSS, Python, PHP, JavaScript, Ruby. This will need multiple teams involved. | HTML, CSS, JavaScript, PyCharm, Eclipse. This will need multiple teams involved. This won’t be as bad as apple products when it comes to porting the application. | HTML, JavaScript, CSS, C++, Python. This will need a team proficient in how android works and how IOS works because of the security and use differences. |

## 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 Operating Platform that I would like to recommend is Windows. Windows is one of the more flexible operating systems available and is also one of the commonly used platforms. Windows scalability makes it ideal for multiplayer games across different environments.
2. **Operating Systems Architectures**: Windows can be easy to operate and navigate because of the simplified user interface. Windows also supports most of your IDEs for all software development. Windows servers support 64-bit architectures which enhance the processing power, memory addressing, and security features. The architecture of windows can handle multiple game sessions with smooth performance without any issues.
3. **Storage Management**: Windows has cloud storage so that as the game/company grows so can the storage amount you have. Windows also has many other storage management options including NTFS/ReFS file systems which are used for large-scale applications with improved security and scalability. This does come at additional cost, but this allows you the flexibility to grow without needing to find more storage along the way.
4. **Memory Management**: With Windows you have 2 memory options which are virtual and physical. Being able to utilize both is huge, especially when trying to manage larger programs. Virtual memory is the best option because the windows server can efficiently allocate the ram to where it is needed which makes sure the active game sessions get priority to the memory. The system will also optimize itself to make sure it is making the best use of the memory.
5. **Distributed Systems and Networks**: through windows you have many apps such as Firefox, google chrome, and Microsoft edge to use for the web-based game as long as you have internet. You can also use remote desktop which allows you to access windows applications on different devices from your computer.
6. **Security**: Windows has built in security such as windows defender but also has many other application choices for security measures. You can also set up daily security checks through these programs. There is also data encryption which protects the game data from unauthorized access. There is different encryptions that helps secure the communication to and from the clients and servers such as TLS/SSL encryption.