

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 | 02.07.25 | DeAndrea Stephens | Initial Software Design |
| 1.1 | 02.14.25 | DeAndrea Stephens | Updated evaluation |
| 1.2 | 02.22.25 | DeAndrea Stephens | Updated 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 looking to develop a web-based version of their already existing Android game, “Draw It or Lose It”. The goal is to have the game work well on different devices. The goal is also to keep team and player names unique to keep players and teams from using the same name, and runs one version of the game at any time.

## Requirements

* *Allows multiple teams that has multiple players*
* *Ensures unique names for games, teams, and players*
* *Maintain a single instance of the game in memory*

## [Design Constraints](#_2et92p0)

* Web-Based: Must work on multiple platforms
* Scalability: Supports growing user demand
* Performance: Displaying images smoothly and at a steady rate

## [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)

Entity class: A base class contain id and name

Game class: Inherits from entity and represents an instance of the game

Team class: Inherits from entity. Ensures the use of unique identifiers and names

Player class: Individual players and associated with a team

GameService class: Uses the singleton pattern. The singleton pattern will ensure only one version of the game is used at one time.

**"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 stable and has good security. Can host-web-based applications but less commonly used for servers. | Best for web hosting, security and scalability. Commonly used for servers.  Free, but enterprise versions may have support costs. | Windows may be user-friendly but have higher cost. | Can connect to cloud-based services. iOS requires development through Apple’s App Store. Android is already supported but requires web compatibility updates. |
| **Client Side** | Mac requires software to be specifically compatible. Compatible with Safari, Chrome, and Firefox. | Users have flexibility but require other setups. Fully compatible with web browsers. | Windows is compatible with many browsers such as Chrome, Microsoft Edge, and Firefox. | iOS runs on Safari and Chrome mobile browsers.  Android runs smoothly on Chrome an any other Android supported browser. |
| **Development Tools** | Support Java, IntelliJ and Eclipse | Supports Java, Eclipse, IntelliJ, and other development tools | Supports Java development using Visual Studio and Eclipse | iOS uses Javascript and Swift in Xcode while Android uses Java and Javascsript in Android Studio for development. |

## 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**: Linux is recommended for hosting because it is free, reliable, and secure. It is commonly used for web servers and can also handle a large number of players without slowing down or interruptions.
2. **Operating Systems Architectures**: Implement a client-server model for efficient performance. This model allows many users to connect at one time. A client-server model should be used. The game logic will run on a central server. Players will be able to connect using their web browsers. The game will run smoothly and be easier to update.
3. **Storage Management**: Use a storage management system such as MySQL with the recommended operating platform. MySQL is powerful and a widely used database that helps keep data organized and ensures it isn’t lost.
4. **Memory Management**: Optimize caching for the Draw It or Lose It. It will make the game run faster. It stores frequently used data. With caching, the game will be more responsive.
5. **Distributed Systems and Networks**: Use cloud-based APIs for real time interactions. This will allow real-time communication between players. With this, the game can update instantly without needing players to refresh.
6. **Security**: Security is a must-have for the client. Implement HTTPS, authentication, and encryption for user data protection. Use HTTPS to ensure that data sent between players and server is secure. Store passwords and personal data in an encrypted format to prevent them from being stolen.