

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 |
| --- | --- | --- | --- |
| 1.1 | 05/27/25 | Keo Foxx | The Gaming Room wants a multi-platform rendition of the game Draw it or Lose it. |
| 1.1 | 06/08/25 | Keo Foxx | Multiple platform evaluations |
| 1.2 | 06/22/25 | Keo Foxx | Add 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 game that can be accessed across multiple platforms. Currently, the game is only available on Android devices. The game they are looking to develop is “Draw it or Lose it”. This game is quick-paced with 4 rounds that last a minute each. There are multiple teams, with each team consisting of several players. Unlike the original television game “Win, Lose or Draw” from the 1980s, this would be done using stock images completed within a 30-second. At the end of the round, the other teams would be given 15 seconds to make one guess at the image.

## Requirements

* Multiple platform ability
* Ability to create unique team names
* Multiple players on each team
* Only one game instance exists at a time

## [Design Constraints](#_2et92p0)

The game application should be developed for a web-based distributed environment, which requires addressing key constraints such as network communication reliability, security measures, and compatibility across different web browsers and devices. To enhance the user experience during game creation and joining, the system must enforce unique names for games, teams, and players to prevent conflicts. Additionally, to ensure stable performance and correct operation, the architecture must be designed to allow only a single active instance of the game service in memory at any given time.

## [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)

**An Entity class was created that is considered to be a superclass that helps to create a relationship between the Game, Team, and Player classes. Also, each class share common identifiers as references to certain unique traits in each game, such as, the name. This make it possible for the Player, Team, and Game class to inherit information for the Entity class.**

**"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** | Easy to access server configuration  Easily upgradable  Not preferred for web hosting  Pricey | Less costly  Quick response time to security flaws  Preferred web hosting  Difficulty to find applications | Large amount of software available  Easily accessible  User friendly  Easily susceptible to virus | Portable and touchscreen  High demand |
| **Client Side** | Expensive  Moderate time and expertise required  Intuitive and user friendly interface | Minimal cost  Maximum expertise and time required.  Free to distribute steep learning curve | User friendly  Mid range price | Flexibility of use. Harder to implement than other devices. |
| **Development Tools** | Macs support front-end languages such as HTML, CSS, and JavaScript, along with the libraries and frameworks commonly used for web development. In addition, they can run general-purpose programming languages such as Java, Python, C++, and many others, making macOS a versatile environment for developers. | It supports a wide range of programming languages, including but not limited to HTML, CSS, and JavaScript for front-end development, along with the necessary libraries and frameworks. Additionally, Linux is well-suited for general-purpose programming languages such as Java, Python, PHP, and Ruby, making it a powerful and flexible platform for developers working across various domains. | Windows supports a variety of languages, including but not limited to HTML, CSS, and JavaScript for front-end development, as well as general-purpose languages like Java, Python, PHP, and Ruby. It also supports the necessary libraries and frameworks to enable a full range of development capabilities across different platforms. | These platforms support a wide range of programming languages, including but not limited to HTML, CSS, and JavaScript for front-end development, as well as general-purpose languages such as Java, Python, PHP, and Ruby. Additionally, libraries and frameworks are available to enhance functionality and streamline development across all three operating systems. |

## 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

After evaluating several options, I recommend using the Linux operating system for this project. I appreciate that Linux is open-source, which helps us keep costs down. It also supports a wide variety of development tools, and from my experience, it’s a flexible and secure platform—ideal for building and deploying robust applications like Draw It or Lose It.

2. Operating System Architectures

Most operating systems I've worked with follow a similar structure. They serve as a bridge between users and the computer’s hardware (cs2.edu). The three main components are:

Hardware: This includes memory, the CPU, I/O devices, and any other physical components.

Kernel: I see this as the heart of the OS—it handles communication between hardware and software.

Shell: This is the interface that allows users to interact with the system (c2.edu). Understanding this structure helps me plan how our game will interact with the system resources efficiently.

3. Storage Management

Since Draw It or Lose It will be web-based, I recommend going with cloud storage. I prefer this approach because it gives us the flexibility to scale as needed—we only pay for what we use. As user traffic grows, we can expand storage without worrying about physical hardware upgrades. Based on what I’ve reviewed, Google Cloud seems like the best fit for our needs.

4. Memory Management

For this project, we’ll be using Linux and developing the backend with Java. One reason I like Java is its automatic memory management using a garbage collector. This takes the burden off us as developers since we don’t need to manually handle memory allocation or deallocation (javapoint.com). It keeps our code cleaner and more efficient.

5. Distributed Systems and Networks

Given that the game is web-based, we don’t need to worry much about which operating system users are on. I recommend designing the game to run in Google Chrome, which is available on nearly every platform. This ensures anyone with internet access can play, regardless of their device or OS.

6. Security

Security is a big priority for me, and by using Google Cloud, we can rely on their secure infrastructure instead of managing our own data centers. On the software side, I’ll make sure we use Google’s tools to automate patches and updates. I also believe in strong QA testing, so I’ll be running thorough tests before we launch any new version to make sure everything is secure and stable.