

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 | 03/22/25 | Eleanor Shamble | Executive Summary, Requirements, Design Constraints, Domain Model |
| 2.0 | 04/06/2025 | Eleanor Shamble | Evaluation |
| 3.0 | 04/20/2025 | Eleanor Shamble | 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)

Draw It or Lose It is a game where teams compete against each other to guess what is being drawn. Currently, Draw It or Lose It is an Android app. The Gaming Room would like to adapt the game Draw It or Lose It into a web application, which would expand access to the game.

## Requirements

* Each Game will have a list of Teams, with no duplicate Team names.
* Each Team will have a list of Players, with no duplicate Player names.
* GameService must be a singleton.

## [Design Constraints](#_2et92p0)

If this app becomes a web app, then there will have to be a lot more attention paid to responsive design. The web app would be expected to be compatible with phones, tablets, laptops, and larger desktops. There may also be issues with different browsers- there may be a bug on Chrome but not Firefox, or Firefox but not Edge. More browsers and device views will need to be tested during Quality Assurance, which may make it easier for problems to slip through the cracks.

## [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)

Game, Team, and Player are all subclasses of Entity. This means that id and name are created and handled within the code for Entity, without Game, Team, or Player’s code needing to explicitly contain them. This is an example of inheritance- they all inherit from Entity.

Game, Team, and Player also all override the toString() method, and Entity does as well. Additionally, they all have two constructors, one with no parameters, and one with parameters. These are examples of polymorphism.

Most classes have both private and public variables, which show encapsulation.

ProgramDriver and SingletonTester both have runnable test methods. GameService is a service class for Game, which ensures through a singleton design pattern that there are no duplicate Games or Game lists.

**"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** | Resistant to viruses. Expensive. | Linux is incredibly flexible and open source. The costs are lower. Resistant to viruses. | Windows has higher operating costs and licensing fees. They have high software compatibility. | Mobile devices would be very inefficient to do web hosting on. It can technically be done but it is not recommended. |
| **Client Side** | Mac specific keybinds would need to be accounted for when determining user controls. The general size of a MacOS toolbar and taskbar would need to be accounted for on the app’s y axis. | Linux users are tech savvy and may prefer more niche browsers than the average Windows or Mac users, so a wider net may need to be cast when doing QA. Linux specific keybinds would need to be accounted for when determining user controls. The general size of a Linux toolbar and taskbar would need to be accounted for on the app’s y axis. | Windows specific keybinds would need to be accounted for when determining user controls. The general size of a Windows toolbar and taskbar would need to be accounted for on the app’s y axis. | Mobile devices have very different proportions than desktop computers, so custom CSS will be needed to support them. Additionally, the user controls are different. Users would not be as quick with their keyboards, or have access to arrow keys, for instance. Running a game web app would need many special considerations. |
| **Development Tools** | This program will likely be using HTML, CSS, and JavaScript in addition to Java. MacOS is compatible with less IDEs than Windows, and more than Linux. | This program will likely be using HTML, CSS, and JavaScript in addition to Java. Linux is the most limited in software compatibility, but it still has IDEs that can be used. | This program will likely be using HTML, CSS, and JavaScript in addition to Java. Windows has the highest level of IDE compatibility, so if The Gaming Room has an IDE of choice, it can be used (this provides the most flexibility when evaluating functionality vs. price). | Doing development work besides QA testing on a mobile device would not be recommended, but it would be good to have a mobile device (both Android and Apple) on hand for QA purposes. |

## 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 would be the most suitable for the server, but Windows would be most suitable for the development environment. I would recommend that the web server be run on Linux due to its flexibility and cost, but I would recommend that the *development* work would happen on Windows due to its high software compatibility, which would mean that the Gaming Room could use any IDE they would choose. Additionally, when developers are running the web app and Quality Assurance is checking it, the bugs for Windows users would be found first, and Windows is the most popular operating system, so that would cover the greatest amount of users.
2. **Operating Systems Architectures**: I will focus on Linux, because that is where the server would be run. The core of Linux is the Kernel, which queues processes and allocates resources to them (GeeksforGeeks, 2025). The System Library contains “predefined functions”, which are the “foundation upon which any software can be built” (GeeksforGeeks, 2025). Then there is the Shell, which is how the User interacts with the Kernel and sends commands (GeeksforGeeks, 2025). System utilities are what users use in the Shell for essential functions like file management (GeeksforGeeks, 2025). Then, there is the hardware where all this functionality exists physically.
3. **Storage Management**: Using an enormous amount of storage is inevitable, due to the game requiring 200 high quality pictures. Both the images and the user data should be stored in database(s). A database would enable programmers to avoid hardcoding data and streamline any retrieval of the data. For example, the code could generate a random number in a set range (which could be determined by finding the size of the table the image IDs are in), and the application could choose an image from the database with an ID of that number. When a user is signing up, the database could be searched for the username and not allow the user to sign up with a taken username.
4. **Memory Management**: Virtual memory can be used to ensure that the only memory being used is memory that is being actively used in the application (The kernel development community, n.d.). Because the amount of images could easily expand far beyond 200, zones could be used to group the memory into specific uses if the amount of physical space exceeds the amount of virtual space (The kernel development community, n.d.). When an image is used in the game, it can be put into a page cache to make subsequent access to the image easier and less time consuming (The kernel development community, n.d.).
5. **Distributed Systems and Networks**: Draw It or Lose It would operate with client-server architecture, where different users can all access Draw It or Lose It’s servers. This would require that all users be connected to the internet, and for their connection to be high speed enough to load the images, but this is not an uncommon requirement for web applications by any means. Because of the adaptability of web browsers and responsive designs, users on different types of machines could all access the web application and play together.
6. **Security**: In the prototype I created, there was username and password-based authentication for the users, and role-based authorization. If Draw It or Lose It would like for players to retain their data between games (which would be ideal for the user- having records of their games may be something users may enjoy) or connect with other players virtually, usernames and passwords would be a must. When storing the usernames and passwords, I would recommend that sensitive data about players be encrypted, and for users to provide their email for 2 factor authentication and potential password recovery. Additionally, role-based authorization would allow for admin users to moderate the platform by banning problematic users, or enable admins to add images.

References

GeeksforGeeks. (2025, April 4). *Architecture of Linux Operating System*. <https://www.geeksforgeeks.org/architecture-of-linux-operating-system/>

The kernel development community. (n.d.). The linux kernel. Concepts overview - The Linux Kernel documentation. <https://docs.kernel.org/admin-guide/mm/concepts.html>