

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.333gyl104xf6)

[**Table of Contents 2**](#_heading=h.7sslyt2qc7lf)

[**Document Revision History 2**](#_heading=h.wqfl72estmiy)

[**Executive Summary 3**](#_heading=h.apugxvxzzk2g)

[**Requirements 3**](#_heading=h.i8iokmwpwg5b)

[**Design Constraints 3**](#_heading=h.sj5ad1rntu6q)

[**System Architecture View 3**](#_heading=h.bl0k87rlqnb4)

[**Domain Model 3**](#_heading=h.7hm2xas1qi8y)

[**Evaluation 4**](#_heading=h.i90sdkq8h6gj)

[**Recommendations 5**](#_heading=h.hgdlfsre9lss)

## [Document Revision History](#_heading=h.fyhn1uunndmc)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <03/27/2025> | Alejandro Exarcheas-Ibarra | Document Created |
| 1.1 | <04/12/2025> | Alejandro Exarcheas-Ibarra | Adjusted contents of Evaluation, and cleared contents of Recommendation, as I had initially misunderstood the scope of the assignment |
| 1.2 | <04/26/2025> | Alejandro Exarcheas-Ibarra | Completed document |

**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](#_heading=h.iu944ikdhpxi)

The Gaming Room has an Android application that they would like to turn into a web application so that it can be playable across multiple operating systems.

## Requirements

The client explicitly stated the following requirements:

* Games will have one or more teams.
* Teams will have multiple players assigned.
* Game and team names must be unique.
* Only one instance of a game can exist at a time.

## [Design Constraints](#_heading=h.g0g3ud2xqbhj)

The greatest constraint is that the game should be compatible with most Browsers. Most modern browsers are built off of Google’s Chromium, however some notable and popular exceptions are Mozilla’s Firefox and Apple’s Safari, which will require specific testing and support.

The game will require testing on various hardware. While the user’s web browser should be able to handle the bulk of the compatibility, considerations need to be made for varying screen sizes and aspect ratios. Additionally, the game should be able to handle the game window changing size.

## [System Architecture View](#_heading=h.mhnsyo1atjvu)

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](#_heading=h.wz7de13qqafq)

The GameService class is a singleton which manages multiple running games of Draw It or Lose It. It can store any number of Games in a list.

There is an abstract Entity class. This Entity class exists for the purpose of inheritance. Other classes, namely the Player, Team, and Game classes inherit the Entity class to utilize shared methods and attributes. The entity class encapsulates these attributes and methods, while also containing the method toString(), which each inherited class overrides. This example of polymorphism has every class able to output a customized description of its contents.

The three inherited classes are composed of each other. The Game class contains a list of Teams, which in turn contains a list of Players.

**"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](#_heading=h.r2ic4snmcx0h)

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** | The greatest advantage MacOS possesses for web hosting is security. The operating system is known to be resilient, and this makes the operating system an attractive option if security is a great concern. However, the operating system also has generally worse software compatibility for web hosting compared to other operating systems. In addition, MacOS may be a more expensive operating system to build a web server off of. | Linux has a number of advantages for server hosting, making it a popular choice, with widespread software compatibility. In addition to a reputation of reliability, Linux is also free and open-source. It bears mentioning, however, that Linux is the most difficult operating system to pilot of the popular options. | Windows has a wide range of software compatibility, as well as access to Microsoft support for their officially supported software and tools. However, it should be noted that Windows is especially vulnerable to security breaches across the major operating systems. | While it is possible to host a server using a mobile device, their comparatively weak hardware and reliance on wireless connection makes this an unattractive option for projects intended to grow in scale. In addition, mobile devices face unique challenges such as a limited battery life and reliance on a mobile network that can introduce instability. It should be noted as well that with lower storage capability, mobile devices will prefer some form of cloud storage solution for hosting the images the game needs, rather than store them on the device itself. |
| **Client Side** | The bulk of compatibility issues will be in supporting common browsers for the operating system. While it is difficult to source usage data for different operating systems, deployment for MacOS certainly means testing support for Safari. | Testing for individual popular distros may be necessary. Linux as a platform is unique in how varied people’s individual machines and operating systems are. Therefore, testing for Linux will necessitate a wide array of hardware and software arrangements to ensure that images display properly and all parts of the website are functional across popular browsers. | Similar to Mac, the operating system itself should not be a major hurdle. Rather, the main concern is browser compatibility. Google Chrome is extremely popular, so it should be prioritized for compatibility testing, as well as other chromium based browsers that build on its framework. | Mobile devices have notably smaller screen resolutions and dramatically different aspect ratios. They will likely require a specialized user interface. Additionally, modern mobile devices have worse access to keyboard controls than other operating systems, so nearly all functionality should be accessible via touch. |
| **Development Tools** | With the game being a webpage, it is important that the game is playable on common Mac browsers, such as Safari, Firefox or Google Chrome. To that end, I believe the best choice would be to program the game in Javascript. For this, Visual Studio, Eclipse, or IntelliJ are common IDEs. In particular, Eclipse is free and open source, even for businesses, making it a potentially attractive option on a budget. It would be remiss of me to neglect to mention Xcode, macOS’s built-in IDE. However, being an IDE optimized for deploying on the App Store, it is less appropriate for the client’s purposes. | Much like Mac, I believe that the best language to program the game will be in Javascript, due to its in-browser capability and ease of integration. Linux in particular has an extremely wide range of IDEs to pick from, while the common options such as Visual Studio and Eclipse remain available. Notably, due to its open source culture, many IDEs on Linux will be completely free of licensing fees. | Once again, Javascript being ideal for the game has not changed, and the tools have not dramatically changed either. The popular candidates for IDEs remain across all major operating systems. Though Windows does not have the amount of choice that Linux does, the major IDEs are still available, such as Visual Studio, IntelliJ, and Eclipse, with licensing costs varying from free to several hundred dollars per user per month. | Mobile, as usual, is an interesting case for development tools. As always, the game is a web page, however, Android devices in particular often have Javascript disabled by default. For this reason, it may be worth considering updating the existing Android application to connect to the new infrastructure. This caveat is exclusive to Android devices, and other mobile devices should have no problems displaying the website properly. |

## 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**: My recommendation would be to operate based on the Linux operating system. Between its tight security and widespread compatibility for server hosting, I think it is a good match for what the client wants with Draw It or Lose It.
2. **Operating Systems Architectures**: Linux is made up of multiple layers of architecture. Past the hardware level is the Kernel, which controls the hardware. Above this is the Shell, which serves as an interface to the kernel to operate the computer (GeeksforGeeks, 2025).
3. **Storage Management**: Draw It or Lose It is reliant on a large repository of images. Rather than store the images directly on the server, my recommendation would be to opt for a cloud storage solution. Cloud storage has some notable advantages, such as price and scalability (Bonuccelli, 2024). This does run the risk of a dependency on factors outside the control of the company, however I believe the benefits outweigh this risk.
4. **Memory Management**: The Linux Kernel contains its own memory management unit. Within this MMU, Linux has a concept of Virtual Memory to simplify memory management and what parts of memory are being accessed. In addition to typical Paging techniques, Linux has a concept of Huge Pages, in the case that a piece of software requires excessive amounts of memory (*The linux kernel*). This capability, I believe, will make the operating system more than capable of managing any amount of games as the project scales.
5. **Distributed Systems and Networks**: There are 2 main methods by which players can connect to the game. There is an existing app on Android devices, and in addition, players will be able to connect via their web browsers. In either case, the game will need to connect to both game servers and the cloud hosting platform. It will be important to monitor potential server issues with the cloud host, as the game will be unplayable if this host suffers downtime.
6. **Security**: One important first step is to maintain a firewall for the server, to filter what information is able to come in and out of Draw It or Lose It servers. This will help ensure that malicious packets are unable to be uploaded to the servers. While this will help to some extent, it is also important to protect user data and information being sent across the network. This is accomplished by encrypting data as it is being transferred to and from the servers, so that even if the data is intercepted, it is unreadable and useless to any attackers (SILBERSCHATZ et al., 2008).

Citations:

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

Bonuccelli, G. (2024a, May 13). *Cloud vs server: Learn the key differences and benefits*. Server and Cloud Blog. <https://www.parallels.com/blogs/ras/cloud-vs-server/>

*The linux kernel*. Memory Management - The Linux Kernel documentation. (n.d.). <https://docs.kernel.org/admin-guide/mm/index.html>

SILBERSCHATZ, A., GALVIN, P., & GAGNE, G. (2008). *Operating System Concepts, 8th edition*. John Wiley & Sons.