

# **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 4**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 7**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 22/06/25 | Omar Barajas | Executive Summary, Design Constraints and Recommendations on different operating systems for web based environment. |

**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 current Android game from The Gaming Room is a guessing game played in groups called Draw It or Lose It. To suit a number of users, the client is now working hard to increase the application's compatibility on several platforms such as iOS, Windows, Mac, Linux, and mobile phones. The program needs to be restructured into a Web offering, with server-side and client-side components, to achieve that. This article discusses the differences between desktop and mobile platforms in the needs of tools, development issues, and hostability. It also tries to suggest an architecture using cross-platform Web technologies so that it can remain scalable, secure, and cheap. In an attempt to balance performance, compatibility, and future expansion, the proposed solution tries to employ a Linux server for back-end processing, with a responsive HTML frontend for the client machines.

## Requirements

With a web-based, distributed platform, the Draw It or Lose It game must be able to run on various platforms like Windows, Mac, Linux, Android, and iOS, according to The Gaming Room. Being one of the most important business needs is to have a cutting-edge, responsive HTML-based user interface with the ability to host thousands of users and allow teams to play simultaneously. The application software should be cross-platform with a consistent user experience on all devices. Data integrity, user authentication, deployment of the software on a scalable server, and latency tolerant enough for real-time play are among the technical requirements. The solution should help reduce redundancy in development by sharing unified development frameworks and technologies wherever reasonable.

## [Design Constraints](#_2et92p0)

Cross-Platform Compatibility: The game should be compatible across various devices and operating systems using contemporary web browsers, necessitating rigorous browser testing and adaptive user interface design.

Real-Time Performance: The application should handle user interactions and graphics rendering in real-time. Server responsiveness, scalability, and fast data exchange using web sockets or APIs are thus a requirement.  
  
Scalability: This solution will require a distributed and horizontal scale server configuration with probably load balancers and cloud providers to accommodate thousands of players concurrently.  
  
Security: Being the data of the user, the game state, and communication should be secured across all platforms and network levels. HTTPS enforcement and user-authentication will be mandatory, and APIs should remain secure.

Development Resources: Creative developers will have to support numerous platforms alongside frameworks such as React or Flutter that will help provide maximum elimination of duplication.

## [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 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** | Despite as much as it can host web applications, macOS Server is not widely utilized in corporate settings. Less hosting companies utilize Mac servers and there isn't considerable server support, so it is more expensive and less scalable.  Licensing cost is actually substantial. | Because it is cost-effective (free and open-source), scalable, fast, and dependable, Linux is widely used for web hosting. It is great for large distributed systems and supports popular stacks like LAMP and MEAN. | Windows Server is dependable and Microsoft technology-friendly (IIS, ASP.NET). Although it offers business functionality and excellent support, it also has license fees and may use more resources. | Because of their limited hardware and operating system restrictions, mobile devices (iOS/Android) cannot operate web-based applications. They are not servers; they are clients. |
| **Client Side** | Web browsers that can support responsive HTML interfaces are Safari. Development is required to be performance and compatibility tested inorder to necessitate a good grasp of the features exclusive to macOS. Cost is higher due to Apple hardware, | Firefox or Chrome are largely installed by desktop users on Linux. Low market share and high browser compatibility are development considerations. Low price, but perhaps little support for non-technical users. | The most significant desktop market share is Windows. Chrome and Edge are used most. has a huge following but needs extensive testing. The development cost and time are medium. | It requires scrupulous testing on devices, screen resolutions, and operating systems, along with responsive web design, to accommodate Android and iOS. While cross-platform frameworks like React Native reduce development time, they require one to be familiar with other mobile-first design. added cost and time investment due to mobile device testing. |
| **Development Tools** | Common tools are IntelliJ IDEA, Visual Studio Code, and Xcode (if developing for iOS) support Python, JavaScript, and Swift, among others. Though the licensing is generally free, there is a Mac hardware requirement for iOS development. | Common tools are Android Studio, Eclipse, and Visual Studio Code. Node.js, JavaScript, and Python languages. All tools are free or open-source. Though the development is flexible, additional setup might be needed. | Mostly utilized are tools like Unity, Visual Studio, and VS Code. Excellent support for C#, JavaScript, and .NET. Some of the products, such as Visual Studio Pro, are licensed and need to be bought. | Popular tools are Flutter, React Native, Xcode, and Android Studio while common Languages are Swift (iOS), Dart, and Java/Kotlin (Android).  Cross-platform tools or separate environments are required for building the program. A Mac is required to program iOS. Licensing is free except for the Apple Developer Program. |

## 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 server OS I suggest to scale up Draw It or Lose It is Linux. Linux has web-based apps in distributed environments with deep support, scalability, stability, and performance. It is open-source, low-cost, heavily supported by cloud providers (AWS, Google Cloud, Azure), and highly compatible with leading development stacks such as LAMP (Linux, Apache, MySQL, PHP/Python). The scalability and future technological maintenance for The Gaming Room will be supported by its strong enterprise and community backing.

1. **Operating System Architectures:**

Linux's kernel structure is massive. Linux's every component executes in kernel space and gives excellent performance. It combines file systems, device drivers, memory management, and process management into an integrated codebase. This leads to efficient resource management and quick system calls for applications like Draw It or Lose It. To enable making the environment fit for distributed web applications' requirements and provide lightweight deployment in terms of taking advantage of speed improvement, modular design also makes components customizable.

1. **Storage Management:**

What I would suggest for storage is Ceph or GlusterFS distributed file system combined with cloud storage block or object like Google Cloud Storage or Amazon S3. All three of these technologies are supported by redundancy, high availability, and horizontal scaling that are mandatory for an online multiplayer game. The architecture guarantees that user information, game sessions, and multimedia content are stored securely when integrated with database systems like PostgreSQL or MongoDB. They include backup and replication support and also have excellent support on Linux platforms.

1. **Memory Management:**

Linux's sophisticated memory management system encompasses all the capabilities of virtual memory, paging, and segmentation. It effectively uses demand paging, swap space, and process isolation at the kernel level to manage memory. By using caching layers (e.g., Redis or Memcached) that store cache-bypassable game data, Draw It or Lose It can reduce memory usage while reducing latency and database load. Efficient use of memory and crash tolerance under heavy loads will also be facilitated by Linux's ability to garbage collect in high level languages such as Python or JavaScript in Node.js and memory overcommit.

1. **Distributed Systems and Networks:**

Draw it or lose it would be better serviced by a microservices architecture deployed atop container orchestration tools like Kubernetes for cross-platform operation. WebSockets and RESTful APIs may hence be exploited for real-time communication between several distant services. The system is scalable and reliable owing to failover configurations, load balancers, and service registries. Cross-platform networking (web, desktop, and mobile) is sustained through CDN-accelerated static content delivery secured with HTTPS over the web, along with VPN/tunneling for inter-service communications. Dependencies such as session management and inter-service coordination can be managed via message brokers and service meshes.

1. **Security:**

Since multi-layered security is critical in serving as a protection to user data as well as in enabling cross-platform communication, all network connections should be encrypted using HTTPS/TLS while encryption of data at rest is done using AES-256 encryption to ensure safe handling of user data across platforms. OAuth 2.0 or JWT tokens may be used for user authentication control, and role-based access control (RBAC) will do the denial based on the user roles. Linux also adds more system protection with built-in security features like firewalld and SELinux. Rate-limiting API gateways prevent DDoS attacks, and intrusion detection systems, vulnerability scanning, and regular upgrades add more strength to overall security. Compliance with regulations like the CCPA and GDPR is key to user privacy. For Draw It or Lose it, they take these measures to develop a secure environment on all of their websites.