

# **Draw it or lose it!**

# **CS 230 Project Software Design Template**

Version 3.0

## 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 4**](#_Toc115077323)

[**Domain Model 5**](#_Toc115077324)

**Evaluation 6**

**Recommendations 7**

**Platform Evaluation and Recommendations 8**

**Detailed Client-Side Evaluation** [**8**](#_Toc115077326)

**Development Tools Insight 9**

**Final Recommendations 10**

**Citations 11**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/28/2024 | John Vowcicefski | Initial creation of the software design document for the "Draw It or Lose It" web-based game. |
| 2.0 | 2/11/2024 | John Vowcicefski | Platform Evaluation and Client-Side Development Tools Recommendations |
| 3.0 | 2/15/2024 | John Vowcicefski | Recreation of Evaluation and Development Tools Recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

This project aims to expand "Draw It or Lose It," originally an Android-exclusive game, into a web-based application accessible across various platforms. Our objective is to maintain the game's engaging experience while adapting it for broader accessibility, including desktop and mobile users.

The development focuses on ensuring cross-platform compatibility, efficient resource management, and scalability to accommodate a large user base. By leveraging modern web technologies and implementing robust software design patterns, we aim to create a seamless and enjoyable gaming experience for all users. This transition not only enhances the game's reach but also aligns with The Gaming Room's vision of inclusive and versatile gaming solutions.

## Requirements

The client, The Gaming Room, requires the game to:

* Be accessible on various platforms, not just Android.
* Support multiple teams and players.
* Have unique identifiers for games, teams, and players to avoid conflicts.
* Ensure that only one instance of the game is in memory at any given time for efficient resource management.

## [Design Constraints](#_2et92p0)

The development of the web-based game faces several constraints:

* **Cross-Platform Compatibility**: The game must function well on different devices and browsers.
* **Scalability**: It should support a large number of users simultaneously.
* **Unique Naming System**: A mechanism is needed to ensure unique names for games and teams.
* **Singleton Pattern:** We need to implement this pattern to manage game instances effectively.

## [System Architecture View](#_ilbxbyevv6b6)

## 

## What We Use to Play: Everyone plays the game on their devices, like phones or laptops.

## Servers Doing the Heavy Lifting: We've got servers that run the game and store all our game stuff like scores and player info.

## Three Main Parts:

## The Look and Feel Part (Presentation Tier): This is the game's website or app that we see and use.

## The Brainy Part (Logic Tier): It's on the server, making sure the game rules are followed.

## The Memory Part (Data Tier): This is where all our game data lives, like who's playing and their scores.

## Talking to Each Other: The game uses the internet to talk to the server securely, and the server uses special commands (SQL) to handle the game data.

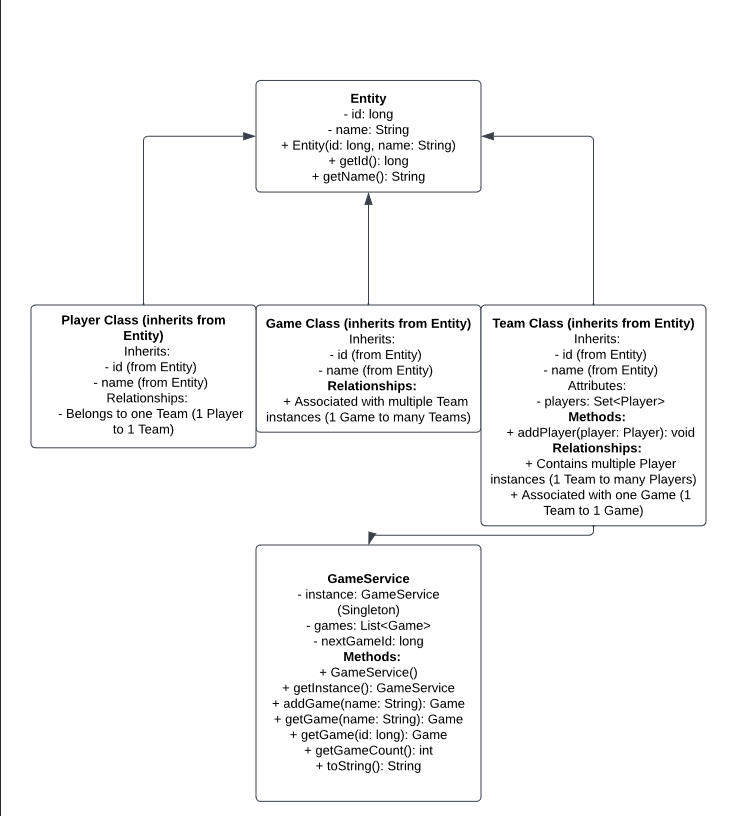
## Keeping Things Smooth: We've got tools to manage lots of players at once and keep the game safe from hackers.

## Adjusting as Needed: If more people play, the system can handle it, and we always check to make sure everything's running fast.

## [Domain Model](#_8h2ehzxfam4o)

**Describe the UML class diagram:**

* **Entity Class**: A base class with common attributes like ID and name.
* **Game Class:** Inherits from Entity, representing a game instance.
* **Player Class:** Inherits from Entity, representing individual players.
* **Team Class**: Inherits from Entity, representing teams with multiple players.
* **Game Service Class:** A singleton class managing game instances and operations.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Macs are solid and secure for servers, but they can cost a lot. They're great if you want something stable and have the budget for it. | Linux is awesome because it's free and you can do a lot with it. It's super customizable, but you might need to know more tech stuff to use it well. | Windows is easy to use and lots of people have it. But it can get expensive because of the licenses and sometimes has more virus issues. | Using mobiles as servers is kind of unusual. They're not really built for that and don't have much power for big server tasks. |
| **Client Side** | Making stuff for Mac users can be pricey since you need specific tools and machines. But Mac users really like their stuff, so it's good to include them. | Linux can be tricky because there are so many versions. It's flexible but can be a headache to make sure your game works on all of them. | Tons of people use Windows, so you got to make sure your game works on it. It's got lots of tools but watch out for security stuff. | Designing for phones and tablets means making sure your game looks good on all sorts of screen sizes. It's extra work but super important. |
| **Development Tools** | You'd use Xcode and Swift for Mac stuff. They're powerful but can be tough to learn if you're new to Apple things. | Linux has loads of tools like Eclipse and GCC. It's great because you can really make it your own. | Windows has tools like Visual Studio, which is super versatile and useful for all kinds of development. | For mobile, you need Android Studio for Android phones and Xcode for iPhones. Or you can use something like React Native to cover both. |

**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 seems like a good choice for the server because it's flexible and not too expensive. But we should make sure the game works on Windows and Mac for players, and on mobile phones too.
2. **Operating Systems Architectures**: Linux is strong and secure for the server stuff. Windows and Mac are good for players using computers, and we can't forget about iOS and Android for mobile gamers.
3. **Storage Management**: We could use cloud storage like AWS or Azure with our Linux server. This way, we can store lots of game data without worrying about running out of space.
4. **Memory Management**: Linux is good at managing memory, especially when lots of people are playing our game at the same time.
5. **Distributed Systems and Networks**: Utilizing cloud services and Content Delivery Networks (CDNs) can ensure efficient communication between various platforms. This setup will help in managing connectivity issues and potential outages.
6. **Security**: Implementing SSL/TLS for data transmission, using secure cloud services, and regular security updates are crucial across all platforms. For mobile devices, additional security measures like app sandboxing should be considered.

**Revised Platform Evaluation and Recommendations**

**Server-Side Evaluation**

**Linux**

* **Characteristics:** Linux's open-source framework and robust cloud compatibility make it an exemplary choice for server-side applications, from startups to large-scale enterprise solutions.
* **Advantages:** Cost efficiency through the absence of licensing fees, superior security mechanisms, and system stability. The vibrant Linux community provides unparalleled support and resources.
* **Weaknesses:** The steep learning curve and the necessity for technical proficiency in Linux administration could challenge teams lacking in-house Linux expertise.
* **Licensing Costs:** None, enhancing its appeal for cost-effective server solutions.

**Mac**

* **Characteristics:** macOS is known for its exceptional performance, reliability, and user-friendly interface, with advanced security features. However, as of April 21, 2022, Apple has discontinued macOS Server, limiting its viability for server-side deployment.
* **Weaknesses:** Higher upfront costs for hardware and software, combined with its discontinuation as a server, may deter its selection for server-side deployment despite its capabilities.
* **Licensing Costs:** Higher due to Apple's ecosystem, impacting the total cost of ownership for server deployment.

**Windows**

* **Characteristics:** Windows Server is favored for its intuitive management interface and extensive support for diverse software ecosystems.
* **Advantages:** Easy set-up process and widespread familiarity within the IT community potentially accelerate deployment and reduce training requirements.
* **Weaknesses:** Licensing fees and the necessity for additional security solutions to mitigate vulnerability risks may increase total costs.
* **Licensing Costs:** Variable, depending on the edition and deployment scale, which can significantly impact budgeting.

**Client-Side Evaluation**

**Web Browsers (Linux, Mac, Windows)**

* **Requirements:** A seamless, responsive HTML5 interface across major browsers like Chrome, Firefox, Safari, and Edge is crucial for engaging a broad audience.
* **Development Considerations:** Emphasize rigorous cross-browser compatibility testing and adopt responsive design techniques to ensure optimal user experiences across diverse devices and screen sizes.

**Mobile Platforms (iOS and Android)**

* **Requirements:** Delivering a user-friendly, responsive application on both iOS and Android platforms is essential, utilizing platform-specific capabilities to enhance functionality and user engagement.
* **Development Considerations:** Cross-platform frameworks such as React Native or Flutter offer efficiency in development, though native development may be required for advanced features.

**Development Tools Insights**

**Linux**

* **Languages and Tools:** A rich ecosystem of development tools including Eclipse, GCC, and Git, available without licensing costs, supports a wide range of development projects.
* **Impact on Development Team:** Proficiency with Linux command-line operations and open-source tools is crucial, highlighting the need for targeted training or specialized recruitment.

**Mac**

* **Languages and Tools:** Xcode and Swift are pivotal to macOS and iOS development, providing robust tools for creating sophisticated applications.
* **Impact on Development Team:** Navigating Apple's ecosystem requires a learning curve, particularly for developers new to Xcode and Swift, underscoring the importance of dedicated training resources.

**Windows**

* **Languages and Tools:** Visual Studio, as a comprehensive IDE, supports an extensive array of languages and frameworks, making it indispensable for Windows-based application development.
* **Impact on Development Team:** The widespread use of Windows as a development platform may streamline onboarding, though consideration of licensing fees for Visual Studio and other Microsoft products is necessary for budget planning.

**IDEs for Programming:**

* According to Okeke's article on TechRepublic, the selection of an Integrated Development Environment (IDE) is critical for the efficient development of software across different platforms. The article lists Visual Studio, Eclipse, and Android Studio among the top IDEs, noting their comprehensive features that support various programming languages and frameworks, which are essential for developing "Draw It or Lose It" across multiple platforms.

**Cloud Services:**

* Borge and Poonia's review on cloud services highlights the strategic importance of choosing the right cloud provider for hosting web-based applications. The article compares Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, noting their distinct advantages in terms of scalability, reliability, and global infrastructure. For "Draw It or Lose It," leveraging these cloud services can enhance the game's performance and availability worldwide.

**Mobile Platforms**

* **Languages and Tools:** Android Studio and Xcode are the primary IDEs for Android and iOS development, respectively. Cross-platform frameworks offer a unified development path across mobile ecosystems.
* **Impact on Development Team:** Choosing between native and cross-platform development paths will significantly influence team composition, training needs, and tooling investments.

**Recommendations**

1. **Operating Platform Choice:** Linux is highly recommended for server-side deployment due to its scalability, security, and cost-effectiveness. Compatibility across Windows and Mac for desktop clients, alongside iOS and Android for mobile users, is paramount.
2. **Client-Side Development:** Embrace responsive design principles and consider the strategic use of cross-platform tools to ensure a consistent and engaging user experience across all platforms.
3. **Development Tools and Team Impact:** Select development tools that complement the team's existing expertise and meet project specifications. Prioritize training for specialized tools and assess the total cost of ownership, including licensing and maintenance expenses, in tool selection decisions.

**Citations in MLA Format**

* Okeke, F. "The 12 Best IDEs for Programming." *TechRepublic*, 9 Feb. 2024
* Borge, Sneha, and Nidhi Poonia. "Review on Amazon Web Services, Google Cloud Provider and Microsoft Windows Azure." *International Journal of Advance and Innovative Research*, vol. 7, no. 3, July-Sept. 2020, pp. 49-60.

**Recommendations: Additional Areas of Improvement**

**Operating Platform**

Linux is recommended as the primary server operating platform for hosting "Draw It or Lose It." This choice is due to Linux's robustness, security, cost-effectiveness, and broad support for web technologies essential for a seamless gaming experience across various platforms.

Linux offers an open-source advantage, reducing overall costs while providing a stable and secure environment for web-based applications. Its compatibility with cloud services like AWS and Azure facilitates scalable deployment options. However, to ensure cross-platform accessibility, the game's client-side should be optimized for both Windows and macOS, alongside mobile platforms through responsive web design.

**Operating Systems Architectures**

Linux's kernel-based architecture enables efficient resource management and scalability, crucial for handling "Draw It or Lose It's" gameplay data and user interactions in real-time. Its modular nature allows for tailored optimizations, enhancing the game's performance and responsiveness across different hardware configurations.

**Storage Management**

Amazon Web Services (AWS) S3 is identified for storage management. AWS S3 offers scalable, secure, and cost-efficient cloud storage solutions, ideal for managing the game's static assets and dynamic content. AWS S3's global infrastructure ensures fast and reliable access to game assets, reducing load times and improving user experience. Its robust security features and scalability support the game's growth and evolving storage needs.

**Memory Management**

Linux excels in memory management with mechanisms like paging, swapping, and the OOM Killer, which are pivotal for optimizing server performance under varying load conditions. Implementing caching strategies, such as Redis for session management, can significantly reduce database load and enhance the gameplay experience by speeding up data retrieval processes.

**Distributed Systems and Networks**

To support communication across various platforms, a combination of RESTful APIs for server-client interactions and WebSocket for real-time gameplay is recommended. This setup ensures efficient, bidirectional communication essential for multiplayer gaming experiences. RESTful APIs provide a standard, scalable way to interface with the game's backend services, while WebSocket facilitates live data exchange, critical for real-time gaming actions. Utilizing cloud-based solutions enhances reliability and scalability across distributed systems.

**Security**

A multi-layered security approach is recommended to protect user data and ensure secure gameplay across platforms. This includes HTTPS for secure data transmission, OAuth for authentication, and regular security audits to identify and mitigate vulnerabilities. On the Linux server, firewall configurations and regular updates are essential for safeguarding the server environment. Implementing these recommendations requires a detailed understanding of each technology and its application to "Draw It or Lose It." The success of this expansion hinges on careful planning, execution, and continuous evaluation of the chosen technologies and architectures to ensure they meet the game's performance, scalability, and security needs.

**Citations**

OWASP. "OWASP Top Ten Web Application Security Risks." The Open Web Application Security Project, 2021, <https://owasp.org/www-project-top-ten/>.

"Linux Kernel Documentation." Linux Kernel Organization, 2021, https://www.kernel.org/doc/html/latest/.