The Gaming Room

# Software Design Template

Version 1.0

## Table of Contents

[Software Design Template 1](#_Toc21086455)

[Table of Contents 2](#_Toc21086456)

[Document Revision History 2](#_Toc21086457)

[Executive Summary 3](#_Toc21086458)

[Requirements 3](#_Toc21086459)

[Design Constraints 3](#_Toc21086459)

[Rationale 3](#_Toc21086460)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 04/15/2025 | Evgenii Chulkov | Final draft |

## Executive Summary

The Gaming Room aims to expand its existing game, "Draw It or Lose It," to multiple platforms, including traditional desktop operating systems (Windows, Mac, Linux) and mobile platforms (Android, iOS). Currently, the game runs only on Android, and the client wants a web-based version that allows players on different devices to participate seamlessly. This requires evaluating the feasibility of deploying and running the game across various platforms while considering cost, scalability, and security factors.

## Requirements

**Server-Side Requirements:**

* The game must be hosted on a scalable web server that handles thousands of concurrent players.
* The server should support secure data transmission and user authentication.
* The solution should minimize licensing costs where possible.

**Client-Side Requirements:**

* The game should be accessible via modern Windows, macOS, and Linux web browsers.
* Mobile users (Android and iOS) should have a seamless experience.
* The web interface must be responsive and optimized for different screen sizes.

**Development Considerations:**

* The development process should use efficient tools and frameworks to ensure cross-platform compatibility.
* The application should follow best security practices to protect user data and prevent unauthorized access.

## Design Constraints

* **Cross-Platform Compatibility:** The game must function consistently across different operating systems and browsers (Linux, Windows, macOS).
* **Scalability:** The server infrastructure should handle large traffic loads while maintaining performance.
* **Security:** User authentication and data protection measures must be implemented.
* **Cost Efficiency:** The solution should minimize expenses related to licensing and hosting.
* **Development Complexity:** The technology stack should allow for efficient development without requiring separate codebases for each platform.

## Rationale

* **Cross-Platform Compatibility:** Since the game must support multiple OS environments and mobile devices, a web-based architecture ensures accessibility without requiring platform-specific applications.
* **Scalability:** The game will have multiple concurrent players, making it necessary to use cloud-based hosting solutions like AWS, Google Cloud, or Azure, which provide auto-scaling features.
* **Security:** Implementing encryption (TLS 1.3), OAuth 2.0 for authentication, and secure storage solutions (such as Android Keystore and Apple Secure Enclave) ensures data protection.
* **Cost Efficiency:** Linux-based servers can reduce licensing costs compared to Windows servers. Open-source frameworks (such as Node.js, React, and Firebase) can lower development expenses.
* **Development Complexity:** Web technologies like HTML5, JavaScript, and WebSockets allow for real-time interaction without the need for separate applications per platform, streamlining the development process.

**Platform Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| Platform | Characteristics | Advantages | Weaknesses |
| Linux | Open-source OS widely used for server environments. Supports common web technologies such as Apache, Nginx, Docker, Node.js. | Free to use, highly customizable, stable and secure. Excellent community support. Preferred for scalable web applications. | Requires technical expertise to configure and maintain. Limited support for some commercial development tools. |
| Mac | Unix-based OS with good developer tools. Less commonly used as a production server. | Great development environment for iOS and cross-platform applications. Built-in security and system stability. | Expensive hardware. Limited server deployment use. Less community support for server-side production compared to Linux. |
| Windows | Popular OS with strong support for .NET technologies and enterprise environments. | Wide usage in enterprises. Easy GUI-based server management. Good compatibility with Microsoft services. | Licensing costs. Resource-intensive. Less efficient for high-performance web server hosting compared to Linux. |

**Recommendations**

**Operating Platform**

A Linux-based platform is recommended for server deployment due to its widespread use in hosting scalable, secure, and cost-efficient web applications. It provides a stable environment with support for web servers like Apache and Nginx and containerization tools like Docker. Its open-source nature ensures no licensing costs, aligning with The Gaming Room's cost-efficiency goals.

**Operating Systems Architectures**

Linux operating systems use a monolithic kernel architecture that efficiently manages system calls, process scheduling, and memory. This design allows for modular extensions, enabling developers to add or remove components without rebooting. Such flexibility is ideal for customizing and optimizing the server environment for "Draw It or Lose It."

**Storage Management**

Linux offers a flexible storage management system with the Logical Volume Manager (LVM), enabling dynamic resizing of file systems and efficient disk use. Reliable file systems like ext4 and XFS support journaling and scalable storage, ensuring effective management of user data and game assets.

**Memory Management**

Linux effectively manages memory using virtual memory management techniques, like paging and demand paging. Combining physical memory (RAM) with swap space ensures applications like "Draw It or Lose It" run smoothly under varying loads. The kernel also includes an out-of-memory (OOM) killer to keep critical services operational during memory pressure.

**Distributed Systems and Networks**

To enable real-time gameplay across multiple platforms, "Draw It or Lose It" should utilize a distributed software architecture with RESTful APIs or WebSockets for client-backend communication. Key services such as load balancers, cloud databases (like Firebase or MongoDB Atlas), and scalable application servers (e.g., Node.js in Docker containers) can be hosted on cloud platforms like AWS or Google Cloud. These systems provide redundancy and failover mechanisms to ensure connectivity, and essential component dependencies, like session state and user data, should be centralized for consistency.

**Security**

Security should be applied at all application levels. Use Transport Layer Security (TLS 1.3) for client-server communications and secure user authentication with OAuth 2.0 or similar standards. Ensure data is encrypted at rest and in transit using AWS KMS or GnuPG solutions. Protect sensitive credentials with Android Keystore and Apple Secure Enclave on mobile platforms. Regular security updates, role-based access control (RBAC), and audit logging are essential to mitigate threats.