

The Game Room

# **CS 230 Project Software Design**

Version 1.1

## 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 | 7/21/24 | Jordan Isaac | Make changes to cover page, add executive summary mention design constraints as per needed for project, list design constraints, topology. |
| 1.1 | 8/4/2024 | Jordan Isaac | Added evaluation for both server side and client side and tool list. Added recommendations for operating systems, architecture, storage management, memory management, distributed networks and systems, and security. |
| 1.2 | 8/18/2024 | Jordan Isaac | Added Recommendations for The Gaming Room Project |

## 

## Evaluation

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Macs are reliable and perform well, making them good for hosting web apps. They have a strong foundation for development. However, Mac servers are more expensive and not as common as Linux and Windows servers. | Linux is popular for web servers because it’s free, flexible, and secure. It supports many server applications and is very customizable. However, it requires more technical know-how to manage. | Windows servers are easy to use and work well with other Microsoft products. They are great for .NET or Java applications and easy to manage. But, they can be costly, and sometimes less flexible than Linux. | Mobile devices aren’t used for hosting web apps because they’re not powerful enough. They are mainly used to access web apps hosted on servers. Ensuring the app works well on different mobile platforms (iOS, Android) is important. |
| **Client Side** | Developing apps for Mac clients can be expensive and needs special skills in macOS development. Apple’s rules ensure high-quality apps but can take more time. Supporting Mac users is good because they value design and performance. | Supporting Linux clients means dealing with many different versions, which can be tricky. It’s cheaper because Linux is free, but finding skilled developers can be hard. Linux users like customizable and fast apps. | Windows has a lot of users, so it’s important to support. Developing for Windows is usually cheaper and faster because there are many tools and developers available. It’s practical and widely used. | Developing for mobile needs attention to both iOS and Android, which can be costly and time-consuming. Skilled mobile developers are needed to make responsive and user-friendly apps. The mobile market is huge, so it’s crucial to reach these users. |
| **Development Tools** | For Mac, developers use tools like Xcode and languages like Swift or Objective-C. You can use Java/Springboot/ Eclipse as well to develop applications. These tools are powerful but need a Mac, which can be more expensive. | On Linux, developers use tools like Eclipse or IntelliJ IDEA and languages like Java, Python, or C++. These tools are often free but require more technical skill. | On Windows, developers use tools like Visual Studio and languages like C#, .NET, or C++. Visual Studio makes building, debugging, and deploying apps easier. Along with those Java can be also be used for developing applications by using Java/Springboot and Eclipse. | For mobile, developers use Android Studio for Android and Xcode for iOS. Languages like Java and Kotlin (for Android) and Swift (for iOS) are common. Each platform needs its own development tools and environment. |

## Recommendations

1. Operating Platform:

For the backend development of "Draw It or Lose It," I recommend using Java, specifically with the Spring Boot framework. Java is a robust and versatile programming language known for its platform independence, allowing the same codebase to be executed across various operating systems, including Windows, macOS, and Linux. This cross-platform capability is essential for a game like "Draw It or Lose It," which aims to reach users on diverse computing environments such as web browsers, mobile devices, and desktop computers. The Spring Boot framework further enhances Java's capabilities by providing a streamlined development process, allowing developers to build and deploy applications efficiently. Its built-in features and pre-configured settings simplify the integration of various components, making it an ideal choice for scaling the game across multiple platforms while ensuring ease of maintenance and consistency in performance.

1. Operating Systems Architectures:

Operating systems (OS) are designed with different architectures to meet various performance, stability, and security requirements. Monolithic architectures are traditional systems where the entire OS, including device drivers, file management, and system calls, runs in a single large kernel. This design, used in Linux and early UNIX systems, offers high efficiency due to minimal communication overhead between components. However, a failure in one component can crash the entire system, making it less resilient to errors.

In contrast, microkernel architectures take a more modular approach. Only the most essential services, like basic inter-process communication and low-level hardware management, run in the kernel. Other services, such as drivers and file systems, operate in user space. This separation, found in systems like QNX and Minix, enhances security and stability, as failures in user-space services do not affect the core system. However, the performance can be slower due to the overhead of communication between the kernel and user-space services.

Hybrid architectures, seen in modern operating systems like Windows and macOS, combine elements of both monolithic and microkernel designs. These systems allow for modularity and stability by running some services in user space while keeping performance-critical components in the kernel. This approach provides a balance, offering both the efficiency of monolithic systems and the reliability of microkernels, making it suitable for a wide range of applications.

For the gaming room I recommend the JVM architecture. The architecture of the Java Virtual Machine (JVM) plays a crucial role in enabling the platform independence of Java applications. The JVM serves as an abstraction layer, translating compiled Java bytecode into machine code that is compatible with the underlying hardware of any operating system. This means that "Draw It or Lose It" can be executed on different operating systems, such as Windows, macOS, and Linux, without requiring modifications to the code. Spring Boot, built on top of the JVM, further simplifies the deployment of Java applications by managing dependencies, configurations, and embedded servers. This architecture not only ensures that the game can be easily ported across different environments but also supports high-performance execution, making it suitable for the real-time demands of an interactive game.

1. Storage Management:

For managing the data of "Draw It or Lose It," I recommend using MongoDB, a NoSQL database known for its flexibility and scalability. MongoDB's document-oriented structure is particularly well-suited for storing dynamic and unstructured data, such as player profiles, game states, and real-time updates. This flexibility allows for the easy adaptation of the database schema as the game's features evolve over time. Additionally, MongoDB is designed to handle large volumes of data efficiently, with built-in support for horizontal scaling, which is crucial as the game's user base grows. The seamless integration of MongoDB with the Spring Boot framework through Spring Data MongoDB further simplifies the process of managing database interactions, ensuring that game data is stored and retrieved efficiently.

In terms of storage management techniques, there are several options to consider. RAID (Redundant Array of Independent Disks) is a popular choice for data redundancy and fault tolerance, as it allows data to be distributed across multiple disks, improving performance and reliability. Storage virtualization is another technique that abstracts physical storage resources, improving flexibility and efficiency.

1. Memory Management:

Java's memory management is handled by the JVM, which automatically manages memory allocation and garbage collection. This built-in memory management system is crucial for maintaining the performance and stability of "Draw It or Lose It," especially given the game's real-time nature. By automatically reclaiming memory from objects that are no longer in use, the JVM helps prevent memory leaks and ensures that the application runs smoothly. To further optimize memory usage, the game can implement techniques such as lazy loading, where data is only loaded into memory when needed, and object pooling, which reuses frequently instantiated objects. These strategies help minimize the memory footprint of the game while maintaining high performance, ensuring that "Draw It or Lose It" can handle multiple players and complex game states without slowing down.

In addition to the JVM's memory management, there are several techniques that can be used to optimize memory usage. Lazy loading is a technique that loads data into memory only when it is needed, reducing memory usage and improving performance. Object pooling is another technique that reuses frequently instantiated objects, reducing memory allocation and deallocation overhead.

1. Distributed Systems and Networks:

To expand "Draw It or Lose It" across different devices and platforms, it is essential to adopt a distributed system architecture. This involves breaking the game into smaller, independent components or microservices, each responsible for a specific functionality, such as user authentication, game logic, or real-time communication. These microservices can be deployed across multiple servers or cloud environments, communicating with each other via RESTful APIs. For real-time updates and interactions between clients and the game server, WebSockets can be employed to provide a persistent, low-latency connection. Additionally, utilizing a cloud infrastructure such as AWS or Google Cloud ensures that the game can scale to accommodate a growing number of users while maintaining high availability and fault tolerance. The Spring Boot framework supports the development of microservices and the integration of WebSocket for real-time interactions, making it an ideal choice for building a scalable and responsive gaming platform.

In terms of network architecture, there are several options to consider. Local Area Networks (LANs) connect computers within a close proximity, such as an office or campus, while Wide Area Networks (WANs) connect computers over long distances, such as between cities or countries. The choice of network impacts the system's latency, bandwidth, and overall performance.

1. Security:

Security is a critical consideration for "Draw It or Lose It," particularly in protecting user data and ensuring safe gameplay across different platforms. To secure communication between clients and servers, it is recommended to use HTTPS and TLS, which encrypt data in transit and protect against eavesdropping and man-in-the-middle attacks. For user authentication and authorization, implementing OAuth 2.0 provides a robust mechanism for securing access to game resources, ensuring that only authorized users can access specific features or data. Additionally, input validation and sanitization are essential for protecting the game against common security threats such as SQL injection and cross-site scripting (XSS). Regular security audits and updates will further help maintain a secure gaming environment. Spring Security, a module of the Spring framework, can be utilized to implement these security measures effectively, providing comprehensive protection for both the server and client sides of the game.

Regular security audits and updates will further help maintain a secure gaming environment. Spring Security, a module of the Spring framework, can be utilized to implement these security measures effectively, providing comprehensive protection for both the server and client sides of the game.

1. In conclusion, building a game like "Draw It or Lose It" requires careful consideration of various components, including operating platform, storage management, memory management, distributed systems and networks, and security. By selecting the right technologies and techniques, developers can create a scalable, reliable, and secure gaming platform that provides a seamless and enjoyable experience for users.