

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 3**](#_Toc115077324)

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

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/08/2025 | Everett De Bree | Initial software design focused on early system structure and requirements prepared for client review. |
| 2.0 | 06/16/2025 | Everett De Bree | Expanded Evaluation section and added platform analysis, file management, and security considerations. |
| 3.0 | 06/21/2025 | Everett De Bree | Final Revision. Added development tool and language alignment, and client/server distinctions. |

**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 Gaming Room is expanding their Android game, Draw It or Lose, to a multi-platform, web-based format this compatible with multiple platforms. The goal of this project is to provide a software design solution that provides consistency, scalability, and cross-platform functionality through Java. The core components will include unique identification for games, teams, and payers, and will enforce a single service instance using the Singleton pattern. The Iterator pattern will be used to traverse collections and maintain uniqueness. This design will allow a solid and maintainable structure to build upon.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Java requires the JVM to achieve platform independence because the JVM hides operating system-specific features. The portability advantage of this approach restricts developers from accessing system calls and hardware-specific optimizations which need proper planning for file management and memory handling features (Silberschatz et al., 2008, Ch. 10). The stateless requirement of RESTful web services built with Dropwizard forces developers to use external methods for data persistence and identity management (Dropwizard Auth, 2020). The system architecture takes shape during its initial development phases because of these limitations (Dennis et al., 2009, p. 85).

## [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 UML class diagram displays a base Entity class with common attributes inherited by Game, Team, and Player. GameService handles the creation and retrieval of these entities. A Singleton pattern is used to ensure only one instance of GameService exists in memory to maintain a consistent application state. The Iterator is used in the addGame() and getGame() methods by looping through collections and enforcing uniqueness.

**"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** | The operating system macOS enables the operation of Apache and Java-based services including Dropwizard but it remains unpopular for production use because of its expensive hardware requirements and restricted server optimization capabilities. Developers can use macOS for building and testing local servers (Silberschatz et al., 2008, Ch. 14). | The operating system Linux (e.g., Ubuntu Server) is best suited for hosting RESTful APIs using Dropwizard. It supports strong security practices like RBAC and iptables, and containerization tools such as Docker and Kubernetes make it scalable and lightweight (Dennis et al., 2009, Ch. 3). | Windows Server operates with IIS and enables Java application hosting through specific setup but requires additional resources. The high resource requirements and licensing expenses together with increased exploit vulnerability make Linux a better choice for cost-conscious environments (Silberschatz et al., 2008, Ch. 15). | Not viable for hosting because of limited processing power and OS restrictions. Has exclusive use for client-side roles (Dennis et al., 2009, Ch. 3). |
| **Client Side** | The operating system macOS enables JavaScript frontends (React, etc.) in browsers and is necessary for developing native iOS applications through Xcode with Swift or Objective-C. The sandboxing policies of Apple and App Store restrictions determine which client applications can access data (Silberschatz et al., 2008, Ch. 14). | Linux supports browser-based frontends and can run Electron or Java GUIs. Android development via Android Studio is fully supported. Developers can test front-end features using browsers and emulators. However, UI tools are often more fragmented than on Windows/macOS (Dennis et al., 2009, Ch. 3). | Windows supports a wide range of front-end tools, including React in Visual Studio Code or IntelliJ. It provides a robust testing environment with access to browser dev tools, emulators, and OAuth/TLS libraries for secure client-side authentication (Dropwizard Auth, 2020). | The final client applications operate on mobile devices. The Android operating system uses Java/Kotlin programming languages while iOS operates with Swift. The clients depend on secure API calls and local storage and UI responsiveness. The combination of different screen sizes and permissions demands extensive testing according to Silberschatz et al. (2008, Ch. 13). |
| **Development Tools** | The operating system macOS enables Java development through IntelliJ and Eclipse while Xcode (Swift/Objective-C) serves as the exclusive tool for native iOS development. Android Studio enables developers to work with Java and Kotlin programming languages. The operating system macOS supports Java-based development and Docker containerization but requires more expensive hardware (Silberschatz et al., 2008, Ch. 14). | The preferred environment for backend development using Java in Eclipse or IntelliJ is Linux. It also supports Android Studio (Java/Kotlin) and Docker for containerized builds. Linux offers strong CLI tools and compatibility with JVM languages, making it ideal for enterprise-scale Java services (Dennis et al., 2009, p. 85). | The Windows operating system enables Java development through Eclipse and IntelliJ while using Android Studio for Android target development. The platform does not have built-in support for iOS development although it enables backend code development and testing. Visual Studio provides additional tools for multi-language development environments yet its licensing model leads to higher expenses (Silberschatz et al., 2008, Ch. 10). | Mobile devices do not develop applications but they are developed for them. Android apps are developed using Java/Kotlin in Android Studio on desktop OSes while iOS apps require macOS and Xcode. The testing tools like Appium and emulator environments simulate mobile behavior. OAuth and TLS libraries secure APIs (Dropwizard Auth, 2020). |

## 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 Gaming Room should implement Linux as its main operating system for the Draw It or Lose expansion. Ubuntu Server stands as an excellent choice because it operates as a free open-source distribution which supports thousands of players (OpenSource.com, 2018) and provides stability and security for Java-based web applications. Linux controls server environments as the leading platform according to June 2025 statistics which includes TOP500 supercomputers and supports modern hardware configurations like 8-core CPUs and 32GB RAM for high traffic. The selection of Linux as the operating system decreases licensing expenses while benefiting from its strong community support which AI tools enhance during 2025. The platform will reach its maximum potential when developers receive Ubuntu administration training while IT operations handles hardware provisioning.
2. **Operating Systems Architectures**: The recommended architecture follows Linux strengths and matches the Java-based backend of the project. The architecture divides responsibilities into separate sections (presentation, business logic, data access) which enables better maintenance and scalability for Draw It or Lose across multiple platforms. The Linux kernel operates efficiently through containerization (e.g., Docker) which is expected to become popular by June 2025 (Arsov, 2017, p. 5) for deploying Java components in a modular fashion. The developers should implement this approach while IT ops monitors performance of the monitoring layer to optimize resource utilization between web and mobile clients which supports the design goal of state maintenance.
3. **Storage Management**: The recommended storage solution combines cloud-based SQL database (AWS RDS with PostgreSQL) for main game, team and player data with local caching on Linux servers. The platform-independent JDBC interface allows access to AWS RDS which provides scalability and durability at 2019 prices (e.g., $0.10/GB-month for EBS) that will likely decrease due to 2025 price wars—check https://aws.amazon.com/rds for current rates (Chapel, 2019, p. 6). The Linux-based Redis caching system improves system performance when dealing with repeated data access. The finance team should check cloud expenses every quarter while developers use JDBC drivers to maintain platform consistency according to Chapel (2019) storage overview.
4. **Memory Management**: The Java Virtual Machine (JVM) operating on Linux enables memory management through its garbage collection and optimization features which prevent memory leaks in web applications. The JVM updates scheduled for June 2025 including Java 21 will enhance memory pooling which decreases latency in Draw It or Lose's multi-user system according to https://openjdk.java.net/ (Silberschatz et al., 2008, pp. 13-15, on I/O memory optimization). IT operations should adjust JVM parameters (e.g., -Xmx32g for 32GB RAM) according to traffic levels while developers conduct peak load tests to verify system stability which matches the project requirements for scalability.
5. **Distributed Systems and Networks**: A distributed system should be built with Linux servers that support HTTP/HTTPS and load balancing and multi-threaded Java servers. Nginx on Ubuntu should be used for load balancing across game instances to ensure scalability for thousands of players as OpenSource.com (2018) supports. The microservices approach can distribute team and player management across nodes (Arsov, 2017, p. 6). Kubernetes will enhance multi-threading and scalability by June 2025 so a pilot deployment should be considered as X notes this trend. Network admins should configure HTTPS with TLS and test latency (target <100ms) to meet real-time gameplay needs as Bashir’s (2019) serverless insights on event-driven systems suggest.
6. **Security**: Linux provides strong security features through its user/group permissions system and iptables firewalls and Java libraries for TLS/SSL encryption which reduce the system's exposure to malware. The system will implement Role-Based Access Control (RBAC) and Multi-Factor Authentication (MFA) for player accounts through Linux security features that will be enhanced by 2025. The addition of serverless functions such as AWS Lambda for login validation creates an extra security layer but this benefit is limited by vendor lock-in risks that open-source alternatives like Kubeless (Bashir, 2019, p. 13) can address. IT security teams need to perform penetration testing by July 2025 while developers should use TLS libraries (e.g., OpenSSL) to encrypt data according to industry standards.

References

Arsov, K. (2017, November 10). Microservices vs. SOA — Is there any difference at all? Medium. https://medium.com/microtica/microservices-vs-soa-is-there-any-difference-at-al-2a1e3b66e1be

Bashir, F. (2019, July 5). What is serverless architecture? What are its pros and cons? freeCodeCamp. https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/

Chapel, J. (2019, July 1). Cloud storage cost comparison: AWS vs. Azure vs. Google. Medium. https://jaychapel.medium.com/cloud-storage-cost-comparison-aws-vs-azure-vs-google-844df3d324

OpenSource.com. (2018, May 31). What is a Linux server and why does your business need one? Linux.com. https://www.linux.com/news/what-linux-server-and-why-does-your-business-need-one-0/

Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). Operating system concepts (8th ed.). Wiley. (Chapter 13: I/O Systems).