

Draw It or Lose

# **CS 230 Project Software Design Template**

Version 1.2.

## 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**

**Evaluation 5**

**Recommendations 6**

**Citations 8**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0. | 09/19/2024 | Anthony Ferguson | Document Creation |
| 1.1. | 10/6/2024 | Anthony Ferguson | Updated Evaluation Section & Recommendation Section. |
| 1.2. | 10/20/2024 | Anthony Ferguson | Updated Table of Contents, Recommendation section, and Citations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has requested a web-based version of their current Android game "Draw It or Lose It." This web-based game should serve multiple platforms, incorporating team and player management, game logic, and performance tracking. The solution proposed involves implementing a scalable, distributed system using Java, which allows multiple users to interact with the game in real-time, ensuring efficient performance and maintainability. The design follows the Singleton pattern to ensure only one instance of game management exists at a time, with the use of unique identifiers to manage games, teams, and players. We will build upon the provided UML to meet the client’s requirements.

## Requirements

* *A game will have the ability to have one or more teams involved.*
* *Each team will have multiple players assigned to it.*
* *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
* *Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.*

## [Design Constraints](#_2et92p0)

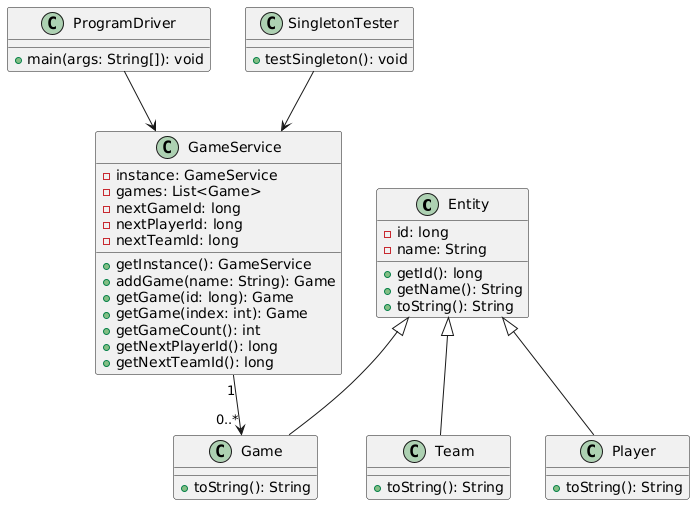
This project requires the game to run in a web-based distributed environment, which introduces challenges such as concurrency management, real-time synchronization between users, and scalability. The distributed nature also requires a strong focus on security, especially for user data. Memory and resource efficiency will also be essential, particularly when managing multiple simultaneous games and user sessions. The use of Java and the Singleton pattern ensures that only one instance of the game management system runs in memory, reducing overhead and preventing duplicate game sessions.

## [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 diagram consists of several core classes: GameService, Game, Team, and Player. All entities inherit from the Entity class, which stores an ID and a name, ensuring each game, team, and player has a unique identifier. The GameService class follows the Singleton pattern to ensure only one instance is in use. The addGame, addTeam, and addPlayer methods will use the Iterator pattern to prevent duplicate names, as per client requirements. Object-oriented programming principles such as inheritance, encapsulation, and composition are evident and effectively used to achieve flexibility and simplicity.



## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac offers strong stability but may face higher licensing and hardware costs. It supports Apache and other web servers for hosting the application. | Linux is ideal for hosting due to its flexibility, strong security, and cost-effectiveness. Most servers run on Linux, and it supports Apache and Nginx. | Windows supports IIS for hosting and may integrate better with certain enterprise software, but it can be costly due to licensing. | Mobile devices typically don’t serve as servers but access the server via a web application. |
| **Client Side** | Development for Mac clients may require specific knowledge of Apple systems, but the cost is moderate. | Linux client support is essential but may require specific expertise in Linux environments, which can increase development time. | Windows clients are widely used, so supporting them is critical but straightforward, though it may increase cost and time due to its widespread use. | Supporting iOS and Android requires developing responsive web design and ensuring cross-platform compatibility, increasing cost and development time. |
| **Development Tools** | Xcode is a popular IDE for Mac development, and there may be licensing costs associated with some tools. | Linux tools like Eclipse and IntelliJ are generally free and open-source, keeping costs down. | Visual Studio is the most commonly used tool for Windows development, which can be costly. | Development for mobile platforms will require tools like Android Studio for Android and Xcode for iOS, both of which are free but require specialized expertise. |

## Recommendations

**Operating Platform**

I recommend Linux as the primary server-side platform for "Draw It or Lose It." Linux offers scalability, strong stability, and cost-effectiveness, making it an ideal platform for high-performance distributed systems. As an open-source platform, Linux allows extensive customization, which is crucial when handling the large-scale real-time multiplayer gaming environment required for this game. According to Adekotujo et al. (2020), Linux stands out due to its compatibility with various hardware systems and strong performance in concurrent environments​. Furthermore, Linux’s inherent security and flexibility make it a preferred choice for hosting large-scale web applications​.

**Operating System Architectures**

Linux’s architecture, featuring robust multitasking and process management capabilities, is highly suitable for handling the concurrent operations that "Draw It or Lose It" will require. Its monolithic kernel structure allows efficient resource allocation and management of user processes, making it a strong candidate for a system that demands real-time performance and synchronization​. Linux's ability to handle distributed environments, as demonstrated in Appuswamy et al. (2018), ensures reliable performance and minimal disruptions, even during peak usage​. This architecture will ensure seamless management of player sessions, server load balancing, and resource optimization.

**Storage Management**

For storage management, cloud-based solutions such as AWS or Google Cloud should be integrated with the Linux server to manage user data and game assets. These platforms offer elastic storage capabilities that scale according to demand, ensuring high availability and quick access times. As highlighted by Adekotujo et al. (2020), Linux servers can efficiently integrate with such cloud solutions, maintaining performance while ensuring the security of stored data​.

**Memory Management**

Memory management in Linux is highly efficient due to its use of demand paging and memory segmentation. These techniques ensure that memory is allocated and freed as needed, preventing memory leaks and optimizing the game's performance during periods of high traffic. The use of Java's garbage collection mechanism alongside Linux’s paging techniques ensures smooth memory management, reducing the risk of crashes due to memory overload​. The Linux architecture also enables fine control over memory allocation, helping to manage real-time operations efficiently.

**Distributed Systems and Networks**

A distributed system architecture will be employed to allow "Draw It or Lose It" to function across multiple platforms. The game will use a client-server model supported by robust networking protocols such as TCP/IP to ensure communication across devices in real-time. Load balancing will be implemented to distribute the system's workload across multiple servers, improving reliability and reducing the chances of downtime. The distributed nature of Linux systems, as explained by Appuswamy et al. (2018), supports seamless communication between different nodes in the network, even under high traffic or during outages​.

**Security**

Security is a top priority for protecting user data in a distributed environment. SELinux or AppArmor, both native to Linux, will be employed to enforce security policies, ensuring only authorized access to sensitive game and player data. These tools provide mandatory access control, limiting the system's exposure to vulnerabilities. Additionally, SSL/TLS encryption will be implemented to secure data in transit, while strong password hashing techniques will protect sensitive information​. Architectural safeguards such as sandboxing and security tickets, as described in Narayanan et al. (2017), will further protect the system from external threats​.

**Citations**

Adekotujo, A., Odumabo, A., Adedokun, A., & Aiyeniko, O. (2020). **A comparative study of operating systems: Case of windows, unix, linux, mac, android and ios**. International Journal of Computer Applications, 176(39), 16-23.

<https://www.researchgate.net/profile/Adedoyin-Odumabo/publication/372400705_A_Comparative_Study_of_Operating_Systems_Case_of_Windows_UNIX_Linux_Mac_Android_and_iOS/links/64b41d62c41fb852dd7b65e1/A-Comparative-Study-of-Operating-Systems-Case-of-Windows-UNIX-Linux-Mac-Android-and-iOS.pdf>

Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., & Rowstron, A. (2018). **Architectural design for a secure Linux operating system**. In 2018 IEEE International Conference on Consumer Electronics (ICCE) (pp. 1-3). IEEE. <https://doi.org/10.1109/ICCE.2018.8299902>

Jaeger, T. (2022). **Operating system security.** Springer Nature.

<https://books.google.com/books?id=a39yEAAAQBAJ&lpg=PR2&ots=Ef47FsksC1&dq=Operating%20System%20security&lr&pg=PR2#v=onepage&q&f=false>