

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 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

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

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/28/2024 | Brenda Sanchez | Update recommendations per additional information and feedback |
| 2.0 | 2/11/2024 | Brenda Sanchez | Updated per additional requirements |
| 3.0 | 3/3/2024 | Brenda Sanchez | Updated per final requirements |

**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 its Android-exclusive game, "Draw It or Lose It," into a web-based multiplatform offering. This game involves teams competing in rapid drawing-guessing rounds. The project's aim is to adapt the game for diverse platforms, ensuring it supports multiple team play, retains unique identifiers for games and teams, and operates a single game instance concurrently.

In the latest update, the Gaming Room is enhancing its game "Draw It or Lose It" with a comprehensive approach to memory and storage management, leveraging advanced technologies for optimal performance across various platforms. This update focuses on ensuring scalability, performance, and security through cloud-native architecture, Linux containers, Kubernetes, and microservices, addressing both current and future technology demands.

**Requirements**

The client, The Gaming Room, seeks to expand their Android-based game app, "Draw It or Lose It," to multiple platforms. This game, reminiscent of "Win, Lose or Draw," challenges teams to guess drawings within a time limit. The client requires an evaluation of Linux, Mac, Windows, and mobile platforms to understand how the app can be deployed and operate across these, focusing on each platform's characteristics, advantages, and weaknesses. This assessment aims to guide their decision-making regarding multi-platform expansion in a distributed environment.

The expansion requires a nuanced approach to memory and storage management, employing platform-specific optimization and cloud-native technologies, including Linux containers and Kubernetes, to efficiently manage resources and enhance the game's multi-platform capabilities.

## [Design Constraints](#_2et92p0)

The development must enable multi-team participation and guarantee unique identifiers for games and teams. The game will be singular in instance and universally accessible across varied devices, demanding compatibility with various operating systems and potential cross-language integration.

Constraints include the need for efficient image compression, dynamic storage allocation, and leveraging distributed file systems, object storage, and advanced caching techniques to manage memory and storage effectively while ensuring the architecture supports scalability and performance optimization.

## [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.

The architecture employs a cloud-native approach with Docker or Podman containers, orchestrated by Kubernetes, to ensure stability, security, and performance. A microservices architecture enhances complex application delivery, supported by distributed file systems and object storage for scalable, efficient data handling.

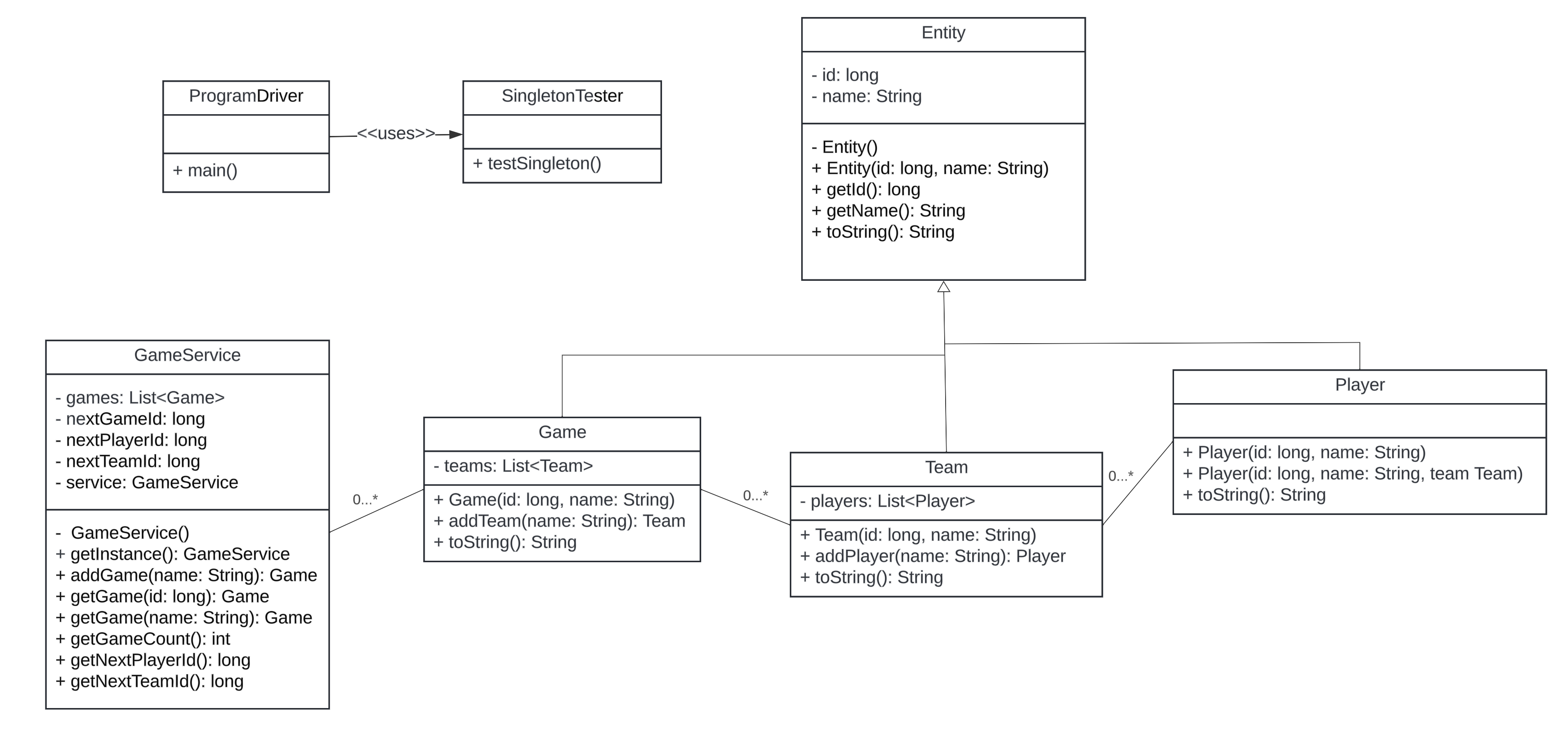
## [Domain Model](#_8h2ehzxfam4o)

The Domain Model utilizes an Entity superclass to centralize common attributes for the Game, Team, and Player classes, employing inheritance. This model demonstrates 'has-a' relationships indicative of class instances containing references to instances of another class, illustrated through UML aggregation.

While focusing on entity relationships within the game, the architectural choices such as microservices and cloud-native components significantly impact the logical structure and interactions of game components, fostering a scalable and flexible game environment.

**The Gaming Room UML Diagram**

com.gamingroom

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Adaptable with robust terminal commands. | Similar flexibility as a Mac at a more economical price. | Extensive software support, and mobile devices prioritize a centralized server model for better tracking. | Not recommended. It is better if the server is immobile for security purposes. |
| **Client Side** | Demands moderate level of skill required with higher costs than Linux. | Demands maximum level of skill required, but more cost effective. | Demands minimum level of skill required with similar costs to Mac. | Offers the advantage of on-the-go updates, though slightly more complex to develop. |
| **Development Tools** | Supports a variety of languages including Java, Python, PHP, and Ruby, with Swift being a prominent choice.  **Mac** supports languages like Swift and Objective-C, primarily through Xcode for macOS and iOS development. Xcode streamlines development for Apple's ecosystem but necessitates Mac hardware, which could influence overall costs despite the free availability of Xcode. | Linux and Windows are compatible with a broad range of tools like Visual Studio and Eclipse, and they support languages such as HTML/CSS/JavaScript.  **Linux** offers a wide array of open-source development tools and IDEs, such as Eclipse and Visual Studio Code. This flexibility benefits teams by providing multiple options, though it might require extra setup time and open-source expertise. Most tools are free, potentially lowering costs unless enterprise support is needed. | Broad range of tools like Visual Studio and Eclipse, and they support languages such as HTML/CSS/JavaScript.  **Windows** caters to a broad range of programming languages, with Visual Studio being the primary IDE for desktop and web applications. It offers a user-friendly environment, though non-Windows testing may be necessary. Licensing costs vary with Visual Studio's edition and team size. | Versatile, accommodating languages like Java and Python across various operating systems.  **Mobile Platforms** have specific tools: Android Studio for Android and Xcode for iOS, alongside cross-platform frameworks like React Native. This setup may require platform-specific teams or knowledge of cross-platform nuances. Xcode and Android Studio are free, but additional tools or services might incur costs. |

Each platform has unique strengths and considerations, from Linux's open-source flexibility and Mac's ecosystem integration to Windows' comprehensive development environment and the specific requirements for mobile development. The choice of operating system and tools should align with project goals, budget constraints, and target audience, considering cross-platform frameworks for efficient development across multiple platforms.

## 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**: After evaluating all options, Windows emerges as the optimal choice for the operating platform due to its widespread adoption and familiarity among users. It offers a variety of Integrated Development Environments (IDEs) at a generally lower cost.
2. **Operating Systems Architectures**: The Windows architecture enables applications to leverage the platform's core functions, like graphics and memory handling, without disrupting the system's fundamental operations.
3. **Storage Management**: Windows incorporates essential tools like Disk Management for sophisticated storage tasks and features like Storage Sense and Disk Cleanup to optimize space by removing redundant files.
4. **Memory Management**: Windows includes inherent utilities for memory management. A database for the game's images would facilitate swift access within the application.
5. **Distributed Systems and Networks**: The game will function on a client-server model, allowing client applications to capitalize on their system's capabilities while relying on a robust server network for interconnected gameplay.
6. **Security**: Windows Defender provides intrinsic security measures for Windows systems. Data transmission will be safeguarded through established encryption techniques.

A cloud-native approach is strongly recommended, utilizing Linux containers and Kubernetes for operational excellence. Emphasizing distributed systems, hybrid cloud, and edge computing for scalability, and advocating for robust security practices like end-to-end encryption and Zero Trust architecture.

**Additional Considerations**

Performance testing, cost analysis, comprehensive monitoring and logging, disaster recovery, high availability, and user experience optimization are crucial components of the deployment strategy, ensuring the game's success in a competitive digital landscape.