

The Gamin Room

# **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**](#_Toc115077324)

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

**Recommendations 7**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/04/2024 | Michael Kinful | Revisions have been implemented in several key sections of the document, including the cover page, the document revision history, the executive summary, design constraints, system architecture view, domain model, and recommendations. These updates are intended to enhance the clarity, accuracy, and overall quality of the document. |
| 1.1 | 02/11/2024 | Michael Kinful | Revisions made based on updated Project guidelines. |
| 1.2 | 02/24/2024 | Michael Kinful | Final revision. |

**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)

CTS has been commissioned by The Gaming Room to develop a web-based version of their Android game, Draw It or Lose It. This new version will support multiple teams, each comprising several players. To ensure the uniqueness of each game instance, team, and player, a singleton design pattern will be utilized for the creation of game instances. Additionally, the iterator pattern will be implemented to manage team and player entities, avoiding any conflicts or duplications.

## [Design Constraints](#_2et92p0)

In transitioning "Draw It or Lose It" to a web platform, we aim to ensure cross-platform accessibility and a seamless user experience by utilizing a REST API for smooth integration across devices. Our focus is on either refining the current interface for user familiarity or introducing innovative designs to enhance engagement. Our development strategy emphasizes multiplayer support and efficient session management through unique identifiers for games, teams, and players, within a solid client-server framework. Image integration, crucial for the game's appeal, will prioritize cross-platform compatibility and copyright adherence. This approach is designed to broaden the game's audience, improving playability and accessibility while preserving its core mechanics and visual allure.

## [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 application's structure features a main driver class that triggers the creation of games, teams, and players through the singleton GameService class, ensuring a single instance in memory. This class's private constructor and public getInstance() method enforce its singleton nature, with creation methods like addGame(), addTeam(), and addPlayer() employing the iterator pattern to prevent duplicates and maintain uniqueness across entities.

Each entity—Game, Team, and Player—derives from the Entity superclass, which provides protected id and name attributes and a protected constructor to ensure entities are properly instantiated with necessary attributes.

The UML diagram illustrates key object-oriented principles such as polymorphism and inheritance through entity class extensions, encapsulation, and abstraction in the entity creation process, ensuring a clear, efficient architecture for the application.

**"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)

Several platforms present viable options for development, including Macintosh, Windows, Linux, and various mobile platforms, each serving as potential hosts/servers or clients. The subsequent analysis explores the merits and limitations of each platform in detail.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Using a MacOS server benefits Mac clients with seamless application support and easy management through its graphical interface. However, it has drawbacks like higher maintenance costs, and for large organizations relying on third-party and custom apps, it might not be the ideal option. | Linux is favored for web hosting by leading companies, including Google and Microsoft, due to its cost-effectiveness and open-source nature. This reduces expenses related to deployment and tools. Furthermore, its customizable design bolsters security, permitting organizations to adapt their security measures. | Windows, as a proprietary OS, necessitates license purchases, marking a potential cost concern. However, its benefits for web hosting are significant, providing extensive compatibility with various applications and third-party software, and smooth updates for software and hardware. Additionally, developers familiar with Windows benefit from robust support for ASP.NET and popular databases such as MySQL. | Employing mobile devices as web servers, while not typical, becomes possible with technologies such as Oracle's Database Mobile Server. This system facilitates the comprehensive administration of apps, users, devices, and data, catering especially to extensive mobile environments and supporting development tools for iOS and Android. |
| **Client Side** | This choice incurs a greater expense for users and necessitates a significant investment of time and expertise for optimal utilization. Skillful navigation of the operating system is essential.  Developing applications for Macs requires a Mac computer equipped with XCode. | There is a lot of  expertise and  time required.  Linux data is  required to use  the operating  system.  Maximum cost for  Linux users  There is a lot of  expertise and  time required.  Linux data is  required to use  the operating  system.  Maximum cost for  Linux users  Considerable expertise and time investment are necessary for operating the system. Proficiency in Linux data is essential for effective use.   Linux development offers a smooth experience, with a variety of language options like Python and Java being widely adopted and well-suited for the task at hand. | Although it comes with a higher price tag compared to Linux systems, this choice boasts an intuitive framework that simplifies the support for Windows setups, requiring minimal expertise to manage effectively. | Efficient setup is enabled by automated tools, ensuring proper device detection and content adaptation for optimal display across different screens. Android SDK, Java-based, allows code developed for Windows and Linux to serve as a foundation. However, for iOS, Swift-based development necessitates Mac hardware. |
| **Development Tools** | Mac OS X Server serves as the core development tool for Mac OS, offering advanced web hosting and tailored essential tools. Development on Mac employs Objective-C and Swift, with Xcode being the prevalent IDE. | The platform supports HTML, CSS, and JavaScript for web development, while Linux development includes C/C++, Java, or Python, with free Python IDEs like NotePad++ and PyCharm. | The platform seamlessly handles essential web languages like HTML, CSS, and JavaScript, complemented by front-end development libraries. | Android development relies on Java with Android Studio as the primary IDE, freely available from Google. Meanwhile, iOS development predominantly employs Objective-C and Swift within Xcode. |

## 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**: Windows, renowned for its extensive compatibility with Integrated Development Environments (IDEs) like Visual Studio, is the optimal platform for extending "Draw It or Lose It" to diverse computing environments. Its robust ecosystem and seamless integration capabilities not only facilitate efficient development but also ensure smooth deployment across multiple platforms, making it the prime choice for scaling our application.
2. **Operating Systems Architectures**: Amidst the dynamic evolution of technology, our steadfast reliance on Microsoft Windows as the cornerstone of our computing infrastructure endures. Beyond its surface, Windows stands as a beacon of user-friendly functionality, offering a robust ecosystem that seamlessly orchestrates tasks ranging from mundane file storage to intricate software execution and immersive gaming experiences.

Yet, its significance goes far beyond that, as Windows serves as the backbone of our operations, enabling the seamless execution of diverse functionalities vital to our productivity and success. With each iteration, Windows not only adapts to our evolving needs but also anticipates them, embodying a synergy of reliability, adaptability, and innovation that forms the bedrock of our digital endeavors.

1. **Storage Management**: Transitioning to cloud-based storage management offers numerous benefits for our future endeavors. It allows us to allocate storage resources efficiently for smooth game operations, adapting to fluctuating demands while minimizing costs. Additionally, cloud storage scalability enables us to easily accommodate the growth of our game's user base and evolving needs.

With robust data management features, including redundancy and disaster recovery measures, we ensure data integrity and continuity. Embracing cloud technology not only optimizes our storage but also future proofs our game infrastructure for sustained success and innovation.

1. **Memory Management**: Windows, renowned for its versatility and customization options, is our preferred operating system for game development. Employing Java for backend development brings notable benefits, particularly its automatic memory management through the garbage collection system. This feature eliminates manual memory management tasks, streamlining development and enhancing application stability.

In addition to providing a user-friendly environment, Windows fosters efficient collaboration and productivity within our development team. Paired with Java's automatic memory management, this setup allows us to focus on implementing innovative game features and refining gameplay mechanics, ensuring an exceptional gaming experience for our players.

1. **Networks**: Developing a network game often requires sophisticated database solutions for real-time interaction, a task streamlined by cloud infrastructure's scalability and reliability. While Google Chrome ensures accessibility across platforms, prioritizing Windows enhances performance and user experience. With its widespread adoption and robust support for web applications, optimizing for Windows promises a superior gaming experience, leveraging its advanced features and stability. By focusing on Windows, we can tap into its rich ecosystem of tools and resources tailored for game development, ensuring optimal performance and smooth gameplay, ultimately delivering an immersive and enjoyable experience for our players.
2. **Security:** Transitioning to Azure, we adopt a comprehensive security strategy that capitalizes on Microsoft's advanced data center defenses. This includes leveraging their state-of-the-art infrastructure, continuous surveillance for threats, and automatic updates to keep security tight. Such measures significantly reduce the likelihood of security breaches and the challenge of dealing with obsolete systems, while also providing the flexibility needed to respond to dynamic security requirements. We enhance our system's security further by enforcing detailed access restrictions, employing top-tier encryption and protocols for API protection, and setting up firewalls in accordance with industry-leading standards. This holistic security model ensures our system remains robust, agile, and well-equipped to tackle future security challenges.

**References**

**Scalability in Cloud Computing: A Deep Dive**

<https://www.mega.com/blog/what-is-scalability-in-cloud-computing>

Estrach, P (2023, August 18) Mega

**What Is API Security? The Complete Guide**

<https://brightsec.com/blog/api-security/>

Maric, N (2022, April 4) BrightSec

**What is API Security?**

<https://www.cloudflare.com/learning/security/api/what-is-api-security/>

N.A. (N.D.) Cloudflare

**Windows Memory Management**

<https://www.geeksforgeeks.org/windows-memory-managment/>

N.A. (N.D.) Geeksforgeeks

**What is Cloud Storage?**

<https://cloud.google.com/learn/what-is-cloud-storage>

N.A. (N.D.) Cloud.Google