

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/28/24> | Kholood Alkohali | Initial draft for client review. |

**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](#_heading=h.35nkun2)

The Gaming Room aims to transition its Android-based game, Draw It or Lose It, into a web-based platform accessible across multiple devices. This endeavor necessitates a meticulous software design strategy to ensure seamless gameplay and user experience. By leveraging modern web technologies and adhering to specific design constraints, we can develop a robust, scalable, and user-friendly gaming application. Our proposed solution involves creating a web-based game application with support for multiple teams, each comprising several players. To achieve this, we will implement unique identifiers for games, teams, and players, ensuring data integrity and optimal memory management. By addressing the client's requirements comprehensively, we aim to deliver a dynamic gaming experience that captivates players across diverse platforms.

## Requirements

The client's business and technical requirements encompass the following:

1. Support for multiple teams and players within a game session.
2. Unique identification mechanisms for games, teams, and players to maintain data integrity.
3. Compatibility across various platforms, ensuring seamless gameplay experiences.
4. Implementation of robust memory management techniques to optimize resource utilization.
5. Security measures to protect user information and ensure data privacy.

## [Design Constraints](#_heading=h.1ksv4uv)

Developing the game application in a web-based distributed environment imposes several design constraints. Firstly, compatibility across different web browsers and devices is essential to ensure a consistent user experience. Secondly, the reliance on web technologies necessitates adherence to web standards and protocols. Additionally, the distributed nature of the application demands efficient data synchronization and communication protocols to facilitate real-time gameplay. Moreover, security concerns regarding data transmission and storage must be addressed to safeguard user information and prevent unauthorized access.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The UML class diagram provided depicts the domain model of the game application. It comprises classes such as GameService, Game, Team, Player, and Entity, interconnected to represent the relationships between games, teams, and players. The use of object-oriented programming principles such as inheritance, encapsulation, and association facilitates the efficient management of game instances, teams, and players. The diagram demonstrates a clear separation of concerns and enables modular development, allowing for scalability and extensibility in the game 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](#_heading=h.z337ya)

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** | Mac systems offer a stable and user-friendly environment for hosting web-based software applications.  Mac systems provide robust security features  but can be relatively expensive compared to other platforms | Linux is an open-source operating system renowned for its stability, security, and flexibility. Linux offers cost-effective hosting solutions but Linux distributions may have a steeper learning curve for beginners | Windows operating system is widely used in both personal and enterprise environments.Windows provides robust development tools and frameworks but Windows hosting solutions may be more expensive compared to Linux. | Mobile devices encompass smartphones and tablets running operating systems such as iOS (Apple). Mobile devices offer portability, accessibility but Mobile devices may have limited processing power and screen real estate |
| **Client Side** | Developing software for deployment on Mac requires familiarity with macOS development tools, including Xcode and Swift programming language. | Developing software for deployment on Linux involves using a wide range of programming languages and tools, including Python, Ruby, PHP, and Java. | Developing software for deployment on Windows typically involves using Microsoft Visual Studio and languages such as C#, .NET, and JavaScript. | Developing software for deployment on mobile devices requires expertise in mobile app development frameworks such as Swift (iOS), Kotlin/Java (Android), and cross-platform solutions like React Native or Flutter. |
| **Development Tools** | Relevant programming languages and tools for Mac development include Swift, Objective-C, Xcode IDE, and macOS-specific frameworks like Cocoa. | Programming languages and tools for Linux development include Python, Ruby, PHP, Java, C/C++, and popular IDEs such as VS Code, Sublime Text, and Eclipse. | Relevant programming languages and tools for Windows development include C#, .NET Framework, Visual Studio IDE, and JavaScript frameworks like React and Angular. | Programming languages and tools for mobile development vary based on the platform. For iOS development, developers use Swift or Objective-C with Xcode IDE. |

## 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**:Given the requirement for cross-platform compatibility, we recommend leveraging web technologies and hosting the game application on Linux-based servers. Linux offers a cost-effective solution with robust hosting capabilities.
2. **Operating Systems Architectures**: Linux-based servers offer a scalable and reliable architecture for hosting web applications, ensuring seamless performance and scalability.
3. **Storage Management**: Implementing a distributed database system such as MongoDB or MySQL would facilitate efficient storage management across multiple platforms.
4. **Memory Management**: Utilizing memory caching techniques and efficient resource allocation strategies will optimize memory usage and enhance application performance.
5. **Distributed Systems and Networks**: Implementing RESTful APIs and WebSocket protocols will enable seamless communication between various platforms, ensuring real-time updates and synchronization.
6. **Security**: Employing SSL/TLS encryption, user authentication mechanisms, and secure data transmission protocols will enhance user protection and safeguard sensitive information across distributed systems. Additionally, regular security audits and updates are recommended to mitigate potential vulnerabilities.