

<Name-of-Software-Application>

# **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 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this 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)

The Gaming Room aims to expand Draw it or lose it into a web-based version that supports multiple platforms other than android. By implementing a singleton-based game service and utilizing object-oriented principles we ensure a scalable, maintainable and secure system.

## Requirements

The game application must support multiple teams with each team having multiple players ensure unique game maintain a single instance of the game service in memory use object-oriented approach with Entity base class for game team and player.

## [Design Constraints](#_2et92p0)

Developing a web-based game the following design constraints Concurrency management the system must handle multiple players simultaneously. Scalability the backend must support growing user traffic. Cross platform compatibility the app must function on different operating systems and devices. Data integrity unique identifier and name validation mechanisms must prevent duplicate entries. And Security.

## [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)

Entity Base class provides common attributes id name for game team and player. Game Service Singleton manages the creation of games ensuring unique names Game contain teams and ensures each game has a unique identifier

Team inherits from entity and represents individual users

**"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** | MacOS is less commonly used for hosting web applications | Linux is highly preferred due to stability scalability and open-source flexibility | Windows is viable but has a licensing cost and requires more maintenance | Not suitable as primary server but can run lightweight services |
| **Client Side** | Development tools are available but fewer users make it lower priority | Requires additional compatibility layers for GUI based game development | Most widely used for gaming making it a strong client side | Accessible and portable |
| **Development Tools** | Java IntelliJ VS code Spring boot tom cat | Java Eclipse IntelliJ Springboot Apache | Java VScode Eclipse Spring boot IIS | Andriod Studio xcode javascript |

## 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 is the recommended platform for development because it has wide support for IDEs and tools used in gaming. It’s the most popular operating system, which makes it the ideal choice for backend infrastructure.
2. **Operating Systems Architectures**: Windows is known for its easy-to-use graphical interface and support for a variety of IDEs like Visual Studio, Eclipse, IntelliJ, and JetBrains. This makes it a solid choice for developing and managing the game application.
3. **Storage Management**: Windows makes managing storage simple, and its cloud server options offer flexibility for scaling up in the future. While the client will need to factor in the extra cost of a Windows cloud server, it provides the ability to expand storage as the game grows and more assets and data are added.
4. **Memory Management** Windows supports both virtual and physical memory, with virtual memory being more efficient for managing large programs. This is especially useful for running complex game processes, ensuring smoother performance and more memory to handle the game.
5. **Distributed Systems and Networks** While Windows does have some challenges, like potential server overloads and lag, these can be handled through good communication and teamwork. Solutions like optimizing server performance and managing queuing issues will help ensure the game runs smoothly for users.

.

1. **Security**: Windows comes with built-in security features like Windows Defender and VPN services to protect user data. To keep things secure, regular security checks, encryption, and training for developers on user data protection are important to prevent unauthorized access and keep user information safe.