

Gaming Room Software

# **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 | 5/22/2023 | Eric Buchanan | <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 is a software application aimed at providing a gaming platform for users to play games, form teams, and interact with each other. The goal is to develop a web-based distribution environment that allows users to access the application from different computing environments. The proposed solution involves designing and implementing a game application using object-oriented programming principles and industry-standard best practices.

## Requirements

The client's business requirements include providing a unique identifier and name for each entity in the application, ensuring that game and team names are unique, and implementing functionality to add and retrieve games, teams, and players. The technical requirements involve using design patterns such as singleton and iterator patterns, adhering to industry best practices, and ensuring code readability and modularity.

## [Design Constraints](#_2et92p0)

The design constraints for developing the game application in a web-based distributed environment involve considerations for hosting the application on different operating platforms such as Mac, Linux, Windows, and mobile devices. These constraints impact the choice of development tools, compatibility issues, deployment considerations, and the ability to support multiple types of clients.

## [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 class diagram provided in the design demonstrates the relationship between the Entity, Game, Player, and Team classes. The Entity class serves as the base class with common attributes and behaviors. The Game, Player, and Team classes inherit from the Entity class, enabling code reusability and adherence to object-oriented programming principles. The singleton pattern is applied to the GameService class to ensure only one instance exists, and the iterator pattern is used to implement functionality for adding and retrieving games using unique names.

**"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.

1. **Server-Side Evaluation**: Here you will need to assess the feasibility of each platform (Linux, Mac, and Windows) for hosting a web-based application.
   * Does the platform offer server-based deployment?
   * What are the potential licensing costs involved?
   * What kind of scalability options does each platform offer to support thousands of players?
2. **Client-Side Evaluation**: This part will require an understanding of the development process to support multiple client types (Android, iOS, Mac, Linux, and Windows).
   * What will it take to develop a responsive HTML interface compatible with all web browser platforms and mobile devices?
   * What are the potential costs, time, and expertise needed?
3. **Development Tools Evaluation**: This involves identifying and evaluating the programming languages and tools necessary for deployment on each platform.
   * What impact will the technical requirements have on the development team(s)?
   * Will multiple teams be needed for different platforms?
   * Are there licensing costs related to the development tools used?

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS, with its Unix-based foundation, is known for its stability and reliability. It also has excellent support for a variety of development tools. However, it's not commonly used for servers due to its higher cost and lack of enterprise features compared to Linux and Windows. | Linux: Linux is widely used in server environments due to its stability, security, and flexibility. It's open-source, which means it's free and customizable. However, it requires more technical expertise to manage and configure. | Windows: Windows Server is user-friendly and integrates well with other Microsoft products. It's a good choice for organizations heavily invested in the Microsoft ecosystem. However, it can be more expensive due to licensing costs. | Mobile Devices: Hosting a web-based software application on mobile devices isn't typical due to their limited resources and power. However, mobile devices can serve as clients to access the application. |
| **Client Side** | Mac: Developing for Mac clients requires knowledge of Swift or Objective-C for native applications. Macs are popular among certain user groups, but supporting them might increase development costs. | Linux: Linux clients are diverse, and developing for them might require knowledge of various technologies. However, Linux's open-source nature can help reduce costs. | Windows: Windows has a large user base, so supporting Windows clients is almost always necessary. Development might require knowledge of languages like C# and .NET framework. | Mobile Devices: Supporting mobile devices is crucial due to the increasing number of mobile users. This requires knowledge of Java or Kotlin for Android and Swift or Objective-C for iOS |
| **Development Tools** | Mac: Common tools include Xcode for native applications, and languages like Python, Java, and JavaScript for web applications. | Linux: Tools vary widely, but may include text editors like Vim or Emacs, and languages like Python, Java, and JavaScript. | Windows: Common tools include Visual Studio for .NET applications, and languages like Python, Java, and JavaScript for web applications. | Mobile Devices: Android Studio is typically used for Android applications (Java, Kotlin), and Xcode for iOS applications (Swift, Objective-C). |

## 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 a detailed analysis of various platforms, Linux is recommended as the appropriate operating platform for The Gaming Room to expand Draw It or Lose It. Linux servers are robust, reliable, and capable of supporting high-performance applications. Its open-source nature provides an opportunity for customization and modification according to the game's specific requirements.
2. **Operating Systems Architectures**: Linux operates on a monolithic kernel architecture. In this model, the entire operating system works in the kernel space, making it highly efficient due to direct access to hardware. The kernel includes system server processes that handle aspects like process, memory, and file management. The monolithic nature of the Linux kernel contributes to its performance, making it an excellent choice for a high-demand environment like gaming..
3. **Storage Management:** For storage management, the Linux file system ext4 is recommended. Ext4 is known for its robustness and reliability, offering features such as journaling, which helps in preventing data corruption in case of a power failure or system crash. Moreover, ext4 supports large individual files and filesystem sizes, making it suitable for modern gaming applications that often involve large files and data sets.
4. **Memory Management:** Linux uses several memory management techniques. It utilizes paging to avoid the limitations of simple base and limit registers. It also employs a method called swapping to maximize the use of main memory by moving inactive processes to the disk when memory is low and bringing them back when needed. Furthermore, Linux uses a Page Cache to store frequently accessed disk data in memory to reduce read operations, improving performance.
5. **Distributed Systems and Networks**: Linux uses several memory management techniques. It utilizes paging to avoid the limitations of simple base and limit registers. It also employs a method called swapping to maximize the use of main memory by moving inactive processes to the disk when memory is low and bringing them back when needed. Furthermore, Linux uses a Page Cache to store frequently accessed disk data in memory to reduce read operations, improving performance.
6. **Security:** Security is an essential aspect of any online platform. Linux offers various tools and practices to ensure the security of user information. For instance, SELinux (Security-Enhanced Linux) can be used to provide access control security policies. On top of that, SSL/TLS protocols can be employed to encrypt the communication between the game server and the clients, protecting the users' data in transit. To protect user data at rest, encryption techniques like dm-crypt can be used to encrypt the file system. Regular updates and patches should also be applied to keep the system safe from known vulnerabilities.