

The Game Room

# **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 | 03/22/2024 | Maria Duhamel | Added my name to cover sheet and replaced brackets with answers to questions asked. |

**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 challenge lies in The Gaming Room staff’s unfamiliarity with the environment setup. To enable the development of the web-based gaming app, we’ll streamline the process. Our approach involves creating a software design document and developing the necessary game applications to meet the software requirements.

## Requirements

The client’s business and technical requirements play a pivotal role in the project’s success. Engaging in client interviews and workshops, asking probing questions to gain a deeper understanding of their goals, and meticulously documenting their responses are essential steps. Equally important is comprehending the client’s needs and expectations. The game must accommodate one or more teams, with each team having multiple assigned players. Ensuring unique games and team names allows users to verify name availability during team selection. Additionally, the system should maintain only a single instance of the game in memory at any given time.

## [Design Constraints](#_2et92p0)

In the context of developing a web-based distributed game, several design constraints come into play. First, ensuring unique game and team names poses a challenge; the system must rigorously validate and enforce uniqueness during the selection process. From a technical perspective, balancing name validation without compromising performance requires a robust mechanism. Additionally, managing teams with multiple players necessitates a flexible data model to handle relationships and maintain data consistency. Proper memory management is crucial, as the game should exist in memory as a single instance at any given time. Creating unique identifiers for games, teams, and players is essential, ensuring uniqueness across instances and throughout the game’s lifecycle. Providing user feedback during name selection impacts the overall user experience. Lastly, legal and compliance constraints dictate avoiding trademark violations and offensive or inappropriate names.

## [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 **ProgramDriver**class has a dependency on the **SingletonTester** class, indicating that the **ProgramDriver** class relies on the functionality provided by the **SingletonTester** class. The solid black arrow signifies a dependency relationship between these classes. The **GameService**class interacts with instances of the **Game** class, illustrating connections, cardinality, and pertinent specifics. Similarly, the **Team**class and the **Player** class connect to the **Entity** class with a hollow arrow. This represents that the superclass **Entity** serves as a general template for the other classes. The **Entity** class provides common attributes, methods, and behaviors that the **Game**,**Team**, and **Player** classes inherit. Essentially, the latter three classes are specialized versions of the **Entity** class.

**"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**, akin to Linux, is built on a Unix-based system, ensuring stability and robustness—qualities essential for hosting. Mac computers feature high-quality displays that enhance user experience by providing accurate color representation and clarity. Additionally, Macs do not necessitate additional graphics cards for professional creative tasks. However, macOS has its limitations: it lacks Internet Explorer (IE), which can be a challenge for testing web applications; setting up ASP on macOS may require more involvement due to tools like Mono; and transitioning to macOS from another operating system may involve a learning curve. Choosing an operating system for web hosting depends on specific requirements and familiarity with the platform. | **Linux**, an open-source system, offers transparency and customization. It is renowned for stability, making it ideal for servers. Thanks to its modular design, Linux can be customized to specific requirements and runs efficiently even on older hardware. Its architecture and community-driven security efforts minimize exposure to malware. Advantages include cost-effectiveness (it’s free and includes most software packages). However, Linux has limited commercial support, a learning curve for new users, and some hardware/software may lack official Linux compatibility. | **Windows servers** offer a stable and reliable hosting environment, supporting a wide range of applications and third-party software. They facilitate patch updates and hardware maintenance. Advantages include effective application handling, seamless integration with directories and SQL Servers, and familiarity with the Windows environment. However, licensing costs can be high, and Windows servers may struggle with multiple applications and processes simultaneously. While comfortable, they have resource limitations. | **Mobile devices** are easy to carry and lightweight. Most have touchscreens and are smaller than laptops, relying on battery life, which can impact availability. The advantages of mobile devices include app-friendliness, easy access to GPS, cameras, and other features. They can be faster due to local databases and can reach a larger audience through app stores. Additionally, some mobile devices can function offline, enhancing user convenience. However, mobile devices also have weaknesses. They can be costly, with some being more expensive than computers. Different operating system versions can complicate development, and web apps may have limited access to certain features. Managing updates across various devices can be complex, and designing for small screens requires careful layout planning. |
| **Client Side** | When developing software for macOS, consider crucial factors: development costs (including hiring skilled developers and purchasing tools), testing costs (across various macOS versions and devices), and distribution costs (especially via the Mac App Store). Rigorous testing is essential to address bugs promptly. Expertise in macOS technologies, design patterns, security practices, and localization ensures success. | **Linux development** leverages cost-effective, open-source tools and community support. While testing demands additional resources, budgeting for ongoing maintenance and updates is essential. Ensuring compatibility across Linux distributions takes time, and deploying on Linux servers involves specific configurations. Developers need expertise in Linux security practices and familiarity with distribution-specific nuances. Linux excels in cost-effectiveness, stability, and customization, but thorough testing remains crucial. | When developing software for **supporting multiple types of clients on Windows**, consider these crucial factors: cost-effectiveness, stability, and customization. Rigorous testing ensures compatibility across Windows versions. Expertise in Windows technologies, security practices, and distribution-specific nuances is essential. Windows excels in hosting web-based software but requires thorough testing and familiarity with the platform. | **Mobile device platforms** vary based on audience, budget, and objectives. They should be visually appealing and user-friendly, with **user testing** to enhance satisfaction. **Security** is critical; regular updates are necessary to patch vulnerabilities. **Performance optimization** involves efficient coding practices, caching, and utilizing local databases for faster data retrieval. **Cost** depends on user requirements, and balancing development time is crucial. Engaging skilled developers with mobile app experience ensures a successful outcome. |
| **Development Tools** | **Swift**, Apple’s official language, excels in building native applications across various Apple platforms. It’s user-friendly and well-suited. Additionally, **Objective-C** remains relevant for maintaining existing codebases and interacting with macOS frameworks. For development, **Xcode** provides a comprehensive environment. If you work with **C or C++,** consider **CLion.** Meanwhile, **Atom**, a popular code editor, supports multiple languages. Of course, it all depends on your specific needs to enhance macOS app and web-based software development. | **Linux** supports a variety of programming languages, including **C/C++, Java, Python, JavaScript, Ruby, and Vala**. Popular **IDE tools** for Linux development include **VS Code**, a powerful code editor with extensions for various languages; **Sublime Text**, an updated code editor suitable for Linux development**; Eclipse**, a robust IDE with features for C/C++ development on Linux; and **NetBeans,** which also supports C/C++ development and other languages. | When developing software for **Windows,** you’ll use various programming languages. **C#** is common, **VB.NET** is less frequent, and **C++** excels for critical applications. **PowerShell**is essential for scripting and automation. For **IDEs, Visual Studio** offers a comprehensive environment, while **Visual Studio Code** is a lightweight editor with extensions for JavaScript and TypeScript. Deployment tools include **PDQ Deploy, Bamboo**, and **Jenkins**, with options like **Ansible, NinjaOne**, and **Chef** based on project needs. | When developing software for mobile devices, you’ll encounter various programming languages and tools. Native iOS apps use **Swift** or **Objective-C**, while Android apps utilize **Java** or **Kotlin**. For cross-platform development, consider **React Native**, which allows you to write code once and deploy it on both Apple and Android platforms. Additionally, **Flutter,** developed by Google, uses **Dart** and provides a rich set of widgets for building UIs, **Ionic**, built on **Angular** and **TypeScript**, or the **Quasar** framework for cross-platform development. |

## 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 10
2. **Operating Systems Architectures**: Windows 10 supports both 32-bit and 64-bit architectures. The 32-bit version is limited to 4 GB of RAM, while the 64-bit version allows access to a larger memory space beyond 4 GB, resulting in better performance. Windows 10 is compatible with both Unified Extensible Firmware Interface (UEFI) and Basic Input/Output System (BIOS), allowing installation on various devices. It operates in both User Mode and Kernel Mode. Windows 10 supports a wide range of hardware through device drivers, facilitating communication with graphic cards and printers. The Windows registry stores configuration settings, user preferences, and system information. Security features include BitLocker for disk encryption, Windows Defender for antivirus protection, and Windows Firewall. Additionally, Windows 10 bridges the gap between Windows and Linux development environments, providing a robust platform for diverse applications and devices.

<https://learn.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model>

1. **Storage Management**: **Windows 10’s powerful utility, Disk Management, simplifies storage management. You can access it easily through methods like Start Menu search, the Power User Menu, Run Menu, and Command Prompt. With Disk Management, you can initialize new hard drives, adjust volumes, and modify drive letters as needed. It’s a handy tool for efficient storage management in Windows 10.**
2. **Memory Management**: **Windows 10 optimizes memory utilization for the Draw It or Lose It software through a combination of strategies. These include leveraging the memory hierarchy (RAM, cache, and storage), utilizing virtual memory to extend RAM, and ensuring efficient file management. The result is smooth performance and seamless execution.**
3. **Distributed Systems and Networks**: **Effective communication between platforms involves several key aspects:** **Componentization**, divide the software into smaller, modular components that can run independently on different devices or servers. **Protocols** are crucial, Implementing standard APIs for smooth interaction between software components across platforms. **Service Discovery** uses dynamic mechanisms to locate and connect to relevant services. **Network Connectivity**, when deciding whether communication will occur over the public internet or within a private intranet. Prioritize security using encrypted channels. Consider a client-server model. The software acts as a client and communicates with a central server (hosted on a server or cloud). Ensure accessible and properly configured databases, utilize load balancers, caching servers, and minimize latency. Handle asynchronous communication and plan for fault tolerance and Scale horizontally and ensure compatibility across operating systems and devices. By addressing these aspects, the software can provide a consistent user experience across different platforms.
4. **Security**: Windows 10 has a few security practices to consider, Windows information protection helps protect data by encrypting it and separating it from personal data, it will automatically classify data based on organization policies, another option is User Account Control, which prompts users for permission before allowing applications to make system changes and prevents unauthorized actions and ensures trusted processes, Windows firewall restricts network traffic, limiting communication to authorized channels and prevents unauthorized access. Data Encryption encrypts the data during transmission and storage it uses HTTPS for network communication and protects user credentials, payment information and other sensitive data, another great tool is regular security audits, which conducts periodic assessments to identify vulnerabilities and addresses issues promptly.

[Windows Privacy Compliance Guide - Windows Privacy | Microsoft Learn](https://learn.microsoft.com/en-us/windows/privacy/windows-10-and-privacy-compliance)

**References:**

Chan, L. (2023). What Language Is The Windows Operating System Written In

<https://robots.net/tech/what-language-is-the-windows-operating-system-written-in/>

Gasim, (2023). Discovering the Best Linux IDEs: Our Top 10 Recommendations

<https://stepofweb.com/what-language-does-linux-use/>

Grigoras, D. (2024). The Language Behind Linux: Exploring the Programming Language Used in Linux

<https://stepofweb.com/what-language-does-linux-use/>

Meredith, J. (2023). The Best IDE Apps for Mac

[The Best IDE Apps For Mac In 2022 - AppleToolBox](https://appletoolbox.com/the-best-ide-apps-for-mac/)

Possamai, M. (2023). What programming languages are used at Apple?

[What programming languages are used at Apple? | by Matteo Possamai | CodeX | Medium](https://medium.com/codex/what-programming-languages-are-used-at-apple-a36261c715da)

Robyn, (2021). 6 Reasons Mac Computers Are Good for Web Design

[6 Reasons Mac Computers Are Good For Web Design - MacSources](https://macsources.com/6-reasons-mac-computers-are-good-for-web-design/)