

The Gaming 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 | 07/19/2025 | Samantha Vernoski | 1st Draft |
| 2.0 | 7/23/2025 | Samantha Vernoski | 2nd Draft |
| 3.0 | 8/20/2025 | Samantha Vernoski | Final Project |

**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 goal of this project is to design a web-based game management system for CTS’s upcoming app “Draw it or Lose it” The application must support multiplayer functionality where games consist of multiple teams, and each team has multiple players. The system must ensure that all game, team and player names are unique, and it must ensure that only one instance of the game management service exists in memory at any time. By using object-oriented design patterns such as Singleton and Iterator, we can create a project that is capable of the client’s needs.

## Requirements

* *The game must support one or more teams*
* *Each team can have multiple players assigned to it*
* *Team and game names must be unique to prevent duplicates*
* *The system must ensure that only one instance of the game management service is ever created during runtime*
* *Each entity must have a unique ID assigned*
* *The system must support object-oriented design for maintainability and reusability*

## [Design Constraints](#_2et92p0)

1. Scalability – The app must manage multiple game sessions without duplication.
2. Memory Management – A singleton patter is used to ensure that only one instance of the GameService class exists in memory at a time.
3. Maintainability – The use of inheritance reduces code duplication across the classes
4. Uniqueness – Game names and team names must be unique to avoid duplicates when users add new ones

## [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 diagram illustrates the class structure of the gaming application. The Entity superclass defines the shared attributes and methos used across all major entities in the system; Game, Team and Player. These subclasses inherit from Entity, demonstrating the object – oriented principle of inheritance which improves maintainability and reduces redundancy. (Lavieri, 2019) The game service class follows the singleton patter to ensure that only one instance exists during runtime. It manages all created game objects and provides methods for adding and retrieving games. This supports that only one instance of game service exists in memory. Each game object maintains a list of the Team objects and each Team manages a list of the Player objects. This models a composition relationship. (Lavieri, 2019). This supports scalability and aligns with the client’s requirements that a game can have multiple teams and each team can have multiple players.

**"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** | Mac is known for their stability, easy to use interface and security. However, the hardware is more expensive and has a limited selection of software compared to windows. Mac can serve as a server host through MacOS Server. Licensing costs may be higher as well using MacOS. (Whatley, S. 2024). Mac is UNIX based but is not open sourced so this would factor in the licensing cost of developing on Mac OS. | Cost effective, can use any application or software you desire. Secure, reliable and flexible. Weaknesses include a steeper learning curve and hardware compatibility. Another added benefit is no licensing cost for Linux. (*TezHost Editorial. 2023.) Linux is also UNIX based and is open sourced so this is a great option to host the server cost wise.* | Highly compatible with a wide range of software. Simple interface and more cost efficient. The main weakness is security due to its over popularity, and it does involve licensing costs. | The trick question.  Mobile devices are not built to work as servers, the other OS systems mentioned will be used to run the application on the mobile devices. |
| **Client Side** | Key factors include ensuring platform compatibility, cross-platform testing. Time and cost can depend on what kind of version is used. | Testing is a major factor as special testing might be needed for certain developments. Finding developers with Linux expertise can be costly. | The more windows versions and hardware that needs to be used to support the system the longer the testing process is. Windows is relatively cost effective, but some testing might bring up the cost due to the range of hardware and systems that may be in place. Although windows client support is common (java, C++ etc.) so finding expertise for windows shouldn’t be an issue. | For mobile devices we would develop these on the other three operating systems. |
| **Development Tools** | To develop this type of software we would need Visual Studio or Xcode(Apple’s IDE). Xcode is free but only runs on MacOS. (Finoit Technologies. October 2024). Kwatee is a developmental tool that helps with testing and is compatible with OS, Linux and Windows. (GeeksforGeeks. September 2024.) | To develop this type of software on Linux we could use Java as the programming language. Java would be the universal coding language that can be used on all the OS systems. Visual studios is a tool used that can be used in Microsoft and Mac OS to develop the code.  Tools like Qt for graphical user interfaces (GUIs) and Wine for running Windows applications on Linux exemplify the operating system’s support for cross-platform development. (Çintay, E. 2024, February 25). | We could use Python, Java Script or C++. There is a wide range that can be used on Windows. Microsoft Visual Studio or Visual Studio Code supports these programming languages. These tools are relatively free or low costing. | We would implement the tools discussed in the other OS systems to run the mobile application. |

## 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**: Linux would be the best operating system to host Draw It or Lose it. It is cost effective and is known for its security scalability and its compatibility. Its open-source flexibility, wide range of software to choose from and ability to be compatible cross-platform makes this the obvious choice.
2. **Operating Systems Architectures**: Linux consists of software, computer programs, documentation and hardware. The main components in Linux are the applications, shell, kernel, hardware and utilities. (GeeksforGeeks. 2025, July 23). The kernal is the main software that manages the hardware and software. The shell is also software; this is what takes the commands from the user and interprets them.
3. **Storage Management**: Cloud based storage database will be the best option to store the data as the application can be stored through the cloud and can be accessed anywhere as well as getting needed updates. This also allows for added storage when needed for updates, this prevents from having to upgrade hardware. This also makes the application compatible with cross-platform as Draw It or Lose It intends to be.
4. **Memory Management**: Linux uses virtual memory to give each process its own memory space. This also protects the game server from interference or crashes. Linux also uses paging where physical memory is divided into little units called pages frames. This helps not overwhelm the system when trying to process the application as fast as Draw It or Lose It wants to process it.
5. **Distributed Systems and Networks**: Linux offers strong networking capabilities, open-source flexibility and reliability. Linux is well suited for hosting the distributed game server. It ensures seamless real time between users on different platforms while handling system dependencies, communication protocols and distributed workload management effectively.
6. **Security**: We would implicate user authentication and enforce multi-factor authentication to protect user information. We could use the least privilege principle to further protect user accounts. We could use data encryption to protect data when using cross-platform with secure file transferring. We could use API tokens, API tokens are code built to secure API access. They act as identification, proving whether the user or application has access to the API. (API *Token. N.d) Using* secure data sharing practices can also ensure that the Draw It or Lose It application will be secure across all platforms when Linux is the chosen operating Platform.

Sources:

Whatley, S. (2024, March 23). *Comparing Mac and Windows Servers for Web Hosting*. MamboServer; MamboServer. <https://www.mamboserver.com/blog/comparing-mac-and-windows-servers-for-web-hosting/>

*TezHost Editorial. (2023, June 27). Pros and cons of Linux hosting © - TezHost. TezHost. http://tezhost.com/linux-web-hosting/pros-and-cons/*

Finoit Technologies. (2024, October 29). *Software Solutions: definition, design, requirements and more*. https://www.finoit.com/blog/software/solutions/

GeeksforGeeks. (2024, September 19). *10 most useful development tools for software developers*. GeeksforGeeks. <https://www.geeksforgeeks.org/blogs/10-most-useful-development-tools-for-software-developers/>

Çintay, E. (2024, February 25). *Linux for Developers: Tools, Tips, and Why It’s the Preferred OS for Development*. Medium; Medium. <https://medium.com/@Chinacolt/linux-for-developers-tools-tips-and-why-its-the-preferred-os-for-development-ce7cb3cb218f>

GeeksforGeeks. (2025, July 23). *Architecture of Linux Operating System*. GeeksforGeeks. <https://www.geeksforgeeks.org/linux-unix/architecture-of-linux-operating-system/>

*API Token - What is it and how does it work?* (n.d.). https://getstream.io/glossary/api-token/