

Draw It or Lose It

# **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> | Oshard Henry | <Brief description of changes in this revision> |
| 1.1 |  |  |  |

**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 wants to create a multiplatform web-based game. To make this work we will need to manage the creation of each game play in memory, making sure that each component within memory belongs to a different player or team.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Developing a web-based application, the number one constraint would be browser type and version. Each browser has their way of compiling code, but by utilizing their API we could implement best practices to lessen the margin for error during run-time.

In addition, user interface design should also come into consideration when designing for browsers can be complicated when finalizing how the vision of the project should be viewed by users. Adequate end user testing and reviewing trends can help with this.

## [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 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.**

This program first makes use of the ProgramDriver class which is the main class where the application will be tested and run, and the SingletonTester class which ensures that only one instance of the GameService is created. This program follows the Singleton Design Pattern, which means it ensures that only one instance of a class, the GameService class, is created or exist at any time in memory. This is done by setting the GameService constructor to private and employing a method, getInstance() to check whether GameService is already in memory. If it isn’t this method will create it.

If GameService is properly implemented, the program can make use of the addGame() method which employs the iterator pattern to check the instance of Game object being created to manage the repetition of Game object names being created. If the Game object name isn’t already created, the method will add it to a ArrayList games, which can be printed.The addTeam method is then responsible for adding team members to the game. The iterator pattern is used here as well to prevent the duplication of similar team objects within the program. If the object doesn’t exist, it is then added to the ArrayList teams.The addPlayer method does the same as the addGame and addTeam method, it checks for duplication using the iterator pattern and adds its unique instance variable to its respectable ArrayList—Player.

The Entity class acts as base class for all three classes, except for GameService, with 2 private attributes, private constructor which helps to prevent creation of new objects just like the GameService constructor.

## [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** | **Advantages**  Developed on Linux like server, GNU is like Linux, which means high security.  Mac environment is user-friendly.  **Weakness**  Isn’t commonly used in cloud computing or deploying servers.  Primarily consumer-facing services | **Advantages**  Dominant force in cloud computing, with many cloud service providers this is due to it enhance security, scalability, and stability.  Affordable  Open-source  **Weakness**  Must be skilled to use this system, compared to mac or windows. | **Advantages**  Majority of the population is familiar with the Windows system.  **Weakness**  Isn’t a major play in cloud computing but can run some services and runs some in a cloud environment, so like a hybrid situation. | **Advantages**  Isn’t known for cloud computing, mostly consumer facing.  Native applications doesn’t require a server  Sites hosted on Mac, Linux, or Windows will run on mobile browsers of any operating system.  **Weakness**  Some features or functionality on computer based website might not be available on a mobile website |
| **Client Side** | **Cost**  Mac OS development can be costly and scales depending on experience.  **Time**  Around average.  **Expertise**  Languages used in mac, like objective-c might require more expertise compared to other languages. | **Cost**  Moderate cost, but scales depending on experience.  **Time**  Around average  **Expertise**  Junior- mid level. | **Cost**  Moderate cost, but scales depending on experience.  **Time**  Around average  **Expertise**  Junior- mid level. | **Cost**  Moderate cost, but scales depending on experience.  **Time**  Around average  **Expertise**  Junior to mid level |
| **Development Tools** | Can utilize most, if not all IDEs, but most native developers stick to XCode Ide. Programming languages for developing mac OS apps includes: Java, C++, Objective-C, Swift, React, and C# (and it’s framework tools) | Can utilize most, if not all IDEs as well, with languages ranging from Java, c++, python, JavaScript, swift, and C | Can utilize most, if not all IDEs as well (except for XCode which is mostly for Mac OS), with languages ranging from Java, c++, python, JavaScript, swift, Kotlin, and C | Cannot utilize IDEs, or programming languages. However, languages like java, kotlin, swift, and python can be used to development applications that can be ran on this device. |

## 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**: For scalability potential, security, and wide preferred usage I would recommend Linux system, mostly for the backend development side of things, and server management. The downside would be the lengthy configuration, but it is a investment that will pay off in the end. For front-end development any system can be used since it is primarily user facing.
2. **Operating Systems Architectures**:

The Linux architecture consists of the Application, Shell, Kernel, Hardware, and Utilities. The kernel is the main component of Linux and it visualizes the common hardware resources and provide each process with necessary virtual resources. This means It makes the process to wait in the ready queue and execute in consequently to avoid any kind of conflict.

System libraries are some predefined functions by using which any application programs or system utilities can access kernel’s features. These libraries are the foundation upon which any software can be built.

The shell is like the kernel’s interface which hides the internal execution of functions of kernel from the user. Users can just enter a command and using the kernel’s function to perform a task.

The hardware is the lowest level and plays the role in managing all the hardware components. This includes drivers, kernel functions, memory management, CPU, and I/O operations.

Utilities are command line tools that perform various tasks provided by the user in order to make system management and administration more efficient. These tasks include file management, system monitoring, network configuration, user management, etc.

**Storage Management**:

I would recommend cloud storage for this application due to its flexibility, stability, and scalability. Cloud storage varies in cost and requires less maintaining compared to a fixed physical storage drive that would probably be hassled to adjust depending on the growth of the application.

Since our virtual machine will be Linux, I would recommend DigitalOcean for its simplicity and reliability. In addition, there are also options to migrate to another cloud infrastructure like AWS if DigitalOcean can’t suit your needs.

1. **Memory Management**:

For Linux the virtual memory abstracts the details of the physical memory from the application software and keeps only the needed information in the physical memory and provides a mechanism for the protection and controlled sharing of data between processes. This also means that physical system memory is divided into page frames or pages with architecture specific sizing and selection which can be controlled at kernel build time by configuring the required kernel option.

Each physical memory page can be mapped as one or more virtual pages. These mappings are described by page tables that allow translation from a virtual address used by programs to the physical memory address. The page tables are organized hierarchically with tables at the lowest level of hierarchy containing physical addresses of actual pages used by the software and tables at the highest level containing physical addresses of the pages belonging to the lower levels.

On a loaded machine memory, the kernel might be exhausted and the kernel will be unable to reclaim enough memory to continue operation. In such a case, to save the rest of the system, it invokes the *OOM killer*.

The *OOM killer* selects a task to sacrifice for the sake of the overall system health. The selected task is killed in a hope that after it exits enough memory will be freed to continue normal operation.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
2. **Security**:

Linux offers a mechanism for numerous security options through providing security checks to be hooked by new kernel extensions. The required configuration for security checks are selectable at build time and can be overridden at boot-time. Numerous factors ensure that Linux is more secure than most other operating systems. For example, to ensure Linux system security, users are automatically assigned lower privileges, it tracks activity and access with system logs, and offers administrators flexible control over file access by setting defined privileges with SELinux.

The service that will also be hosting the application, DigitalOcean, also offers another added layer of security which secures the infrastructure of the application and the information it obtains from users.