

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 | <05/25/24> | Julianne Takaya | First Revision |
| 1.2 | 6/10/24 | Julianne Takaya | Second Revision: Added evaluation of server side, client side, and development tools needed and provided recommendations on which OS would be best to use. |
| 1.3 | 6/22/2024 | Julianne Takaya | Third revision: added recommendations for OS, OS Architectures, Storage and Memory Management, Distributed systems and network, and Security. |

**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 currently has an gaming app available on Android called “Draw It or Lose It”. They are looking to expand the game into a web-based service. The game consists of multiple games, which host multiple teams, each with multiple players. For the game to function properly in a web-based environment each game name, team name, and player name must be unique. Draw It or Lose It hosts a large library of stock drawings for the game to show to players. The team at The Gaming Room do not know how to set up a web-based environment.

## Requirements

* The game must have cross-platform functionality
* Players on different platforms should be able to play together.
* Each game and team names must be unique
* Players should have the ability to check if a name is available before starting a team
* Only one instance of the game can exist in memory at a time
* Each team should have multiple players
* Each game should host one or more teams

## [Design Constraints](#_2et92p0)

* Android, iOS, and the web have different software development requirements
* The API needs to be made to work with different platforms
* Game and team names must be unique
* Use of unique game id’s limits instances of games to one

## [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 contains the main method. ProgramDriver uses Directed Association with SingletonTester Class to check if there is already an instance of the GameService. The child classes, Game, Team, and Player, all inherit required attributes from the parent class, Entity. Multiplicity is shown between classes. The GameService may have zero to many Games, Games may have zero to many Teams, Teams may have zero to many 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 includes built in Apache web-server commands. Mac’s client creation policies are more intuitive than those of Windows, but it runs on limited hardware configurations.  The cost to license the server is also high. | Linux has a powerful command system and web-server while being cost effective. Linux servers are stable, secure, and reliable. | Windows servers have the advantage of being familiar to most users. They are also very secure, but carry a cost to use. | Although mobile devices can be used as servers, they are not optimized for it. They lack the power of other options. |
| **Client Side** | Mac is more expensive than Windows, but ease of use is relatively equal between the two. | Linux is very popular among web-developers because of its cost-effectiveness and open source programs. However, the learning curve is steep for those not familiar with it. | Windows has many unique tools that can only be virtualized on other systems, and has extensive support for web-app and website development. | Provides the flexibility of having the app available anywhere at any time, but requires adjustments for displaying on smaller screens. |
| **Development Tools** | VSCode could be used for syntax and code previews, In addition platform non-specific languages such as Javascript, HTML, CSS, React. | VSCode, Atom, git, node, flask. Additionally, Javascript, Java, HTML, CSS | VSCode, git, node, javascript, HTML, CSS, React | Firefox, opera, Samsung browser, chrome. Websites should work across all mobile browsers. |

## 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**: I recommend The Gaming Room to use Linux to start this project. Linux is open source, reducing costs, while remaining secure. While there is a slight learning curve for those unfamiliar with the system, Linux supports many IDEs, and is very secure.
2. **Operating Systems Architectures**: The Linux architecture generally has four layers: Application, Shell, Kernal, and Hardware(Oishi, 2023). The Hardware consists of physical components of the system, like CPU, RAM, and input and output devices such as keyboards and monitors or printers. Linux supports a wide range of input/output devices. The Kernal is the core of the operating system and handles things like device, resource, and memory management, system calls, and performance optimization. The shell is the interface that processes and executes commands given by the user. Applications are the programs the user runs on top of the architecture. These include media players and web browsers(Oishi, 2023).
3. **Storage Management**: For storage I recommend a cloud storage solution. With cloud servers, The Gaming Room can use only the space they need with room to grow at any given time. This could save money in the long run as they are only paying for what they use. At this time, I recommend Google Cloud Functions.
4. **Memory Management**: Linux uses several memory management techniques to optimize memory performance. Virtual Memory is a technique that allows a program to use more memory than is physically present in the system through a combination of hardware and software. It also uses Memory Mapping, which allows a process to access a files contents as if it were part of the processes memory, allowing it to access large files efficiently (Kumar, 2023). Linux also uses shared memory, which allows multiple processes to access the same portion of memory. These are all techniques that will be useful in the application Draw it or Lose it.
5. **Distributed Systems and Networks**: As Draw It or Lose It must communicate between various platforms, the a distributed system may be used to accomplish this. A distributed system is a collection of independent computers that appear to the user as a single coherent system (Singh, 2023). To accomplish this, a system’s components can communicate using various protocols and tools, including TCP/IP, HTTP, or message queues (Singh, 2023). As Draw It or Lose It is a web based game, the web browser acts as distributed system, allowing anyone with access to the internet access and play the game across different OS and Mobile devices.
6. **Security**: Security should be built into the game from the ground up. Use of Role Based Security in the game’s code will separate Admin and Player access to the various parts of the game. On to of that, user data that is required by the game such as user name should be protected by encryption, an only data necessary for game operation should be gathered. Furthermore, a service such as Cloudflare could be used to protect against outside attacks. Finally, Google Cloud Functions also has inbuilt security to protect user data against unauthorized access.

# References

Kumar, S. (2023, march 24). *Process Memory Managment in Linux*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/process-memory-management-in-linux

Oishi, A. Z. (2023, Nov 20). *Architecture of Linux Operating System*. Retrieved from LinuxSimply: https://linuxsimply.com/linux-basics/introduction/architecture-of-linux-operating-system/

Singh, O. (2023, March 18). *What are distributed systems and how do they work?* Retrieved from Cointelegraph: https://cointelegraph.com/explained/what-are-distributed-systems-and-how-do-they-work