

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/23/2022 | Andrew Menard | Executive Summary, Design Constraints, Domain Model, Evaluation, Development Requirements, Recommendations |
| 1.1 | 02/10/2023 | Andrew Menard | Evaluation – Server-side, client side, development tools |
| 1.2 | 02/20/2022 | Andrew Menard | Finalized recommendations |

**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 requires their mobile Android application Draw It or Lose It be available as a web-based program. Draw It or Lose It is a multiplayer game where the teams have to guess the images that are rendered by the program within a certain time. This requires formatting the pre-existing game to multiple operating systems and allowing them to sync.

## [Design Constraints](#_2et92p0)

First the game will have to be available as a web-based application on all operating systems. This will require us to either utilize the current code or recode for the appropriate operating systems. These include Windows, Linux, and Mac OS/iOS. We need to determine if it is better to rewrite the code in Swift for Apple devices or utilize multiple languages within each other to make the program work. The game requires the ability to have one or more teams, each team can have multiple players, and the game and team names need to be unique and check whether they are already in use. Additionally, only one instance of the game can exist in memory at a given time.

## [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)

In this diagram we see that the Game, Team, and Player objects all inherit from the Entity class. The relationship between Game, Team, and player is zero to many as depicted by 0..\*. This shows that The Game object may have 0 to many teams, and teams may have zero to many players. This is an example of Aggregation because they have a “has a” relationship. The Game Service Object also has a 0 to many relationship with those three objects, however it does not inherit from 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** | The accessibility and configuration for the server are simple. The terminal commands are versatile for accessing and making changes, and designing the server. Expensive | Lower cost to make changes. Has command shell like Mac for easy server configuration. Navigating platform not cohesive. | Also expensive but mass software availability. User friendly and has command prompt. | Wide range in mobile devices where some dive specifications are better than others. |
| **Client Side** | Moderately expensive and requires moderate expertise needed and familiarity to to navigate OS | Lower cost but maximum expertise required to navigate OS. More time and proficiency may be required to support Linux. | Cost similar to Mac. Minimum expertise needed as OS is user friendly and user does not need much time to familiarize with Windows setup. | More complex to implement due to many updates required. Users need to commit more time and support. Provides flexibility to clients to be able to see updates any time. |
| **Development Tools** | Mac can support all languages. Swift would be the Apple Specific language. HTML/CSS/JavaScript support Front End development. Java, Python, and Ruby on Rails for other programming needs. Some tools within Mac are NotePad++, Eclipse, PyCharm, Visual Studio and Github. | Linux support but are not limited to HTML/CSS/JavaScript to support Front End development. Java, Python, and Ruby on Rails for other programming needs. Other development tools are NotePad++, Eclipse, PyCharm, Visual Studio and Github. | Windows OS support but are not limited to HTML/CSS/JavaScript to support Front End development. Java, Python, C++ and Ruby on Rails for other programming needs. Other development tools are NotePad++, Eclipse, PyCharm, Visual Studio and Github. | Android and iOS support but are not limited to HTML/CSS/JavaScript to support Front End development. Java, Python, C++ and Ruby on Rails for other programming needs. Other development tools are NotePad++, Eclipse, PyCharm, Visual Studio and Github. |

## 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**:

We recommend utilizing a Linux OS. Linux is a popular server OS that is open sourced. The cost of using Linux OS relatively low and it being open sourced not only helps keep costs down but also provides for a helpful community that can be utilized. The popularity and open-sourced nature of Linux allows for easing pairing with applications on all platforms. A cloud provider can be utilized to reduce costs and provide dedicated expertise. When developing with Linux OS it allows for local testing. We can have different versions of the same software running in the development OS as it would run on the actual server whenever the application or website is hosted. Linux also usually comes with most of the compilers and interpreters that you would need. If not, then it is easy to use the OS command line to get them.

1. **Operating Systems Architectures**:

The Basic architecture of Linux OS consists of the following layers: the Kernel, system libraries, system utility programs, hardware layer, and the shell. The kernel stores all the core operating system operations. These modules cooperate with the hardware directly. “The kernel facilitates required abstraction for hiding details of low-level hardware or application programs to the system” (Javatpoint, 2022). The system libraries are applied for implementing the functionality of the OS and don’t need code access rights of the kernel modules. The system utility programs are responsible for doing “specialized level and individual activities” (Javatpoint, 2022). There are several peripheral devices that are part of their hardware layer that include the CPU, HDD and RAM. The shell is the interface between the user and the kernel. The structure of the Linux OS architecture provides a lightweight solution that occupies less disk space, which therefore improves efficiency and processing.

1. **Storage Management**:

The Linux storage and file directory structure is different than in Windows OS. “On Windows, an application might store all its files in C:\Program Files\Application. On Linux, its files would be split between multiple locations – its binaries in /usr/bin, its libraries in /usr/lib, and its configuration files in /etc/” (Secur, 2021). Linux utilizes a multi-tiered indexed allocation of the file system structure. The files and/or directories each have a unique identifier. “These identifiers are referenced across multiple index tiers to conserve memory space by avoiding large, single indexes” (Silberschatz, 2009). Linux OS’s efficiency of its storage management is beneficial in this use case.

Additionally, utilizing a cloud computing provider will increase scalability and efficiency of the application. Google Cloud Storage is a viable option that will provide great performance across a wide distribution of clients at an affordable price.

1. **Memory Management**:

When a process is loading, everything will first go through the RAM, and then to the CPU. To make this process faster, “the CPU has level one, level two, level three cache. That is like RAM, but on the CPU” (GoLinuxCloud, 2020). Cache plays an important role in Linux memory management because “Data won't be written immediately from RAM to hard disk, but to optimize write to the hard disk, Linux works with the concept of dirty cache” (GoLinuxCloud, 2020). Virtual memory plays an important role in Linux OS as well because it only writes to memory when the process demands. Physical memory size constraints are eliminated by using virtual memory and allows multiple programs to run in parallel.

1. **Distributed Systems and Networks**:

An important requirement for Draw It or Lose It is that the application is available and can be used across multiple platforms. It is important that this traffic and processing power is balanced. Load balancing will do this across a distributed system. This spreads the client requests evenly across app servers. There are several load balancing algorithms available to implement to ensure optimal server use. Some examples of those are: least connection method, round robin, weighted fair queueing, etc.. Linux uses an implementation of weighted fair queuing (WFQ) called Completely Fair Scheduling (CFS) (Colyer, 2016).

In order to promote multi-platform support, the application should use an API that serve all clients uniformly. REST API utilizes a client-server design pattern that separates the user interface concerns and the data storage concerns. This allows the client and server components to evolve independently. This improves “the portability of the user interface across multiple platforms and improve scalability by simplifying the server components” (What is REST…, n.d.).

1. **Security**:

One of the security advantages of Linux is that default user privileges are limited. Linux enforces the concept of least privilege. This is different from windows where default users are generally given administrative access. This is important because user education is very important in keeping a secure operating system. Limiting unnecessary access can mitigate harmful changes. An article by Jeff Goldman outlines how “84 percent of hackers leverage social engineering in cyber-attacks” (Goldman, 2021). The open-source nature of Linux OS is another reason for its security. This may seem contradictive, but the fact that coders can collaborate and review each other’s work reduces potential for bugs in code. “Linux and Unix-based operating systems have less exploitable security flaws known to the information security world. Linux code is reviewed by the tech community, which lends itself to security: By having that much oversight, there are fewer vulnerabilities, bugs and threats” (Taylor, 2018). Linux also utilizes process independence, which allows each process to have a unique virtual address space. This independent address space prevents memory areas from potentially being overwritten by rogue applications.

**References:**

“Architecture of Linux - Javatpoint.” *Www.javatpoint.com*, https://www.javatpoint.com/architecture-of-linux#:~:text=The%20Linux%20operating%20system's%20architecture,actions%20of%20the%20Linux%20OS.

Colyer, ~ Adrian. “The Linux Scheduler: a Decade of Wasted Cores.” *The Morning Paper*, 26 Apr. 2016, https://blog.acolyer.org/2016/04/26/the-linux-scheduler-a-decade-of-wasted-cores/.

“Filesystem and Storage Device Management in Linux.” *Secur*, 10 June 2021, https://www.secur.cc/filesystem-and-storage-device-management-in-linux/.

Goldman, Jeff. “Fully 84 Percent of Hackers Leverage Social Engineering in Cyber Attacks.” *ESecurityPlanet*, 28 Jan. 2021, https://www.esecurityplanet.com/threats/fully-hackers-leverage-social-engineering-in-cyber-attacks/.

Silberschatz, Abraham, Peter Baer Galvin, & Greg Gagne. (2009). *Operating System Concepts, Eighth*

*Edition*. John Wiley & Sons, Inc. https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/.

Taylor, Dave. “Why Linux Is Better than Windows or Macos for Security.” *Computerworld*, Computerworld, 6 Feb. 2018, https://www.computerworld.com/article/3252823/why-linux-is-better-than-windows-or-macos-for-security.html.

“Tutorial: Beginners Guide on Linux Memory Management.” *GoLinuxCloud*, 10 Sept. 2020, https://www.golinuxcloud.com/tutorial-linux-memory-management-overview/#Overview\_on\_Linux\_Memory\_Management.

*What is Rest?* (Not Dated.). RESTfulAPI.net. Retrieved from https://restfulapi.net/.