

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 | 07/16/2023 | Sebastian Jimenez | Template for design of the Draw It Or Lose It game |

**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 client “Gaming Room” has a developed a game called “Draw It Or Lose It”, the issue is the game is only on the android app store so they would like to port their game to other platforms via a web-based version.

## 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)

In a web-based application for the program to function, it must be always connected online to communicate with the servers. Making offline play impossible. All of the methods and functions will be executed in the backend instead of locally so the program will not run as efficiently but will be available to play on many platforms.

## [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 the diagram, there is one parent class which will exemplify one of the OOP principles which is inheritance. 3 subclasses will be created that will inherit the public members and methods of the Entity class. Private variables and methods will not be inherited due to their private status but can be accessed indirectly through the parent’s public methods which is encapsulation. Another OOP principle that is shown is polymorphism, although toString() method is inherited by Entity, the subclasses override this method with their own toString() method to format the information that is appropriate for each class. GameService is not a subclass but is associated with the other 3 classes. Of course, there is the ProgramDriver class which is the main class that will be used to run the program. The final class is the SingletonTester which will be called from the main class in order to test whether the singleton pattern is working as intended.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | macOS currently does not have a standalone server OS as its been discontinued, instead it is integrated into macOS. They weren’t considered popular as other server OSs people usually opted for both Linux and Windows OS | Linux is the most popular OS to host web servers. Due to its reliability and the large variety of distros that are specific to servers like Red Hat Enterprise which is used by many companies. Since some Linux distros are specifically made for server machines, they do not contain unnecessary processes making it reliable and stable. Stability is important when hosting a web server. One negative about Linux is it’s not as supported as Windows which means it has less in terms of tools. | Window’s first server was the NT and currently Windows server 2022. Its strength lies in its file hosting servers. The Windows server OS differs from regular desktop Windows, it can scale better, and it has features that help manage servers. Although the command line interface is utilized more, it is considered more user friendly making it easier to train personnel to manage and maintain servers easier. | Although it is certainly possible to host servers from the phone, it is not ideal. The hardware and storage that is in today’s phone cannot compare to server hardware. One of its strengths is power utilization, due to their size they consume less electricity making them cheaper to maintain. |
| **Client Side** | Developing for macOS will take more time to develop since it uses its own IDE and programming language. The macOS hardware is typically weaker than Windows, but still very good, so more considerations must be considered to make compromises. It is better to utilize developers that are already familiar with the macOS architecture to speed up the process | Linux is open source and so are most of its applications. Although it is open source you are still able to monetize apps if you like. Even though Linux has many distros it is still possible to make the Linux version of the app compatible with most distros by packaging. | As the most supported OS out of all the platforms. Developers have many tools at their disposal when it comes to developing applications. Unlike Linux, Windows has fewer versions, so development is more uniform. One negative about Windows is it is not as resourceful with its hardware as Linux meaning that you might need better hardware to run Windows stable. | The other big mobile platform would be iOS. Since development is already being made for macOS, it means that we can save time on development time since they both use the same tools like the IDE and programming language. Someone who’s familiar with the macOS architecture can develop for both platforms in tandem. It might take a little longer to get approved since apple is stricter about their apps than android is |
| **Development Tools** | Building apps on macOS is more streamlined, the official IDE to develop macOS apps is Xcode and using the Swift programming language | When developing for Linux to cover most distros you might want to build the application through a appimage, which helps with compatibility with many environments and distros. There is more tools that Linux offers other tools to make apps made for Windows compatible with Linux like Wine or Proton which is a compatibility layer that can work with video games. When it comes to IDEs and languages, most Windows-compatible tools should be compatible in Linux. | The many tools that can be used include eclipse IDE, Visual Studio Code among other ones. And even though it is written with C and its variants the apps can be run in many different programming languages including python, Java, or C. | Most tools will be the same as the macOS version using Xcode, although it’ll have to be scaled down to be optimized for mobile devices. |

## 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**: When it comes to servers one of the most popular OS used for servers is the Linux-based operating system, out of the many different Linux distributions I would recommend using the Debian Distro.
2. **Operating Systems Architectures**: The Debian distro created in the 90’s is still going strong with its newest iteration, Debian 12 “bookworm”. The non-profit development organization that supports Debian believes that their software should be free to everyone and supports each iteration with up to 5 years on average. As expected Linux-based software does not require the newest hardware and can work on older machines just fine. Debian 12’s hardware requirements are 2 GB of RAM, a dual core Processor or higher, and 20 GB of disk space minimum. Although for hosting web servers you will want to upgrade to better hardware in order to meet client demands. When it comes to a user-side interface, Debian 12 is more user-friendly than ever with its desktop environment which is more traditional with Windows desktops or if you’d like it supports the bash command line interface with is traditional with Linux-based OS. It also supports kernel 6.1, the kernel is the interface that connects the hardware to the OS. Kernel 6.1 brings many improvements like new features to filesystem formats. When it comes applications there is the Debian repository that has a total of 64,419 packages according to the official Debian website, these packages contain applications and the dependencies that are required to run them.
3. **Storage Management**: Although the minimum requirement to run Debian 12 is 20 GB it would be prudent to increase it, especially with the storage of images you would want to prepare in case the developers add more images or add new features to the game. When first installing an OS, each hard drive attached to the host computer will have to be formatted, one of the most popular filesystem formats is ext4, which is the newest iteration of the Ext (extended file system) file format first created in 1992 and written by Remy Card. The ext4 has improved capability and it attempts to reduce fragmentation and instead opts more for contiguous file blocks or extents in order to decrease head seeks. You can also create different partitions on the same disk in case you need to separate files for security. The Ext4 contains the inode, which is an index table that points to the data file location, if the file data is updated then the inode also must also update. The issue with having both of these update is if something happens in between updates then they both become desynchronized creating errors and corruption. This is where the journal comes in and solves that issue. There are several types of journaling methods but essentially when a data file is updated it creates an entry in the journal and only after the inode is fully updated then the journal entry is removed, this reduces the chances of desync between the inode and the file. Not only is the file system important but also the physical attributes of the drive. Modern SSDs are superior to the older HDDs but when it comes to dedicated servers it is more cost effective to use HDDs due to their price and capacity.
4. **Memory Management**: Although the recommended operating platform is Debian, the Linux kernel is mostly responsible for memory management, it is shared among different Linux-based distros. In order to load up the files that are related to the Draw It or Lose It software (images in this case) ideally you would want a file system like Ext that prioritizes contiguous data allocation, this requires less seeks from the disk head since it doesn’t have to look for the data files across several sections. Once the data is found it is stored into the RAM and then fed to the CPU for processing. Memory components such as RAM and CPU cache are volatile meaning that memory will be stored on them as long as there is power being fed into them, otherwise the data will be wiped from these components.
5. **Distributed Systems and Networks**: Out of the many distributed systems, the most appropriate system and simplest I would recommend is the Server-Client system. Porting the game to more platforms will ensure that your user base will grow exponentially. You would want a system that is easily scalable that could handle the oncoming traffic and if the user base goes down, you can also scale back by reducing the number of running servers in order to save costs. This centralized system comes with some cons. If the electricity were to go out in the building, then the game would go offline since all the information is stored in one place. Someone will also have to manage the servers, so you will have to train a team to maintain the servers. You will also need a web server, one of the most popular ones are Apache and NGINX. Again, since the application has the potential to host several thousands of clients at a time it’s important to pick a web server that can handle the traffic and has flexible scalability. I recommend using NGINX, as Apache slows down when it approaches 10,000 requests. NGINX can handle many requests by distributing requests among itself using the Master-Slave system. When it receives requests from a client it will delegate the process to a slave and then continues to look out for more requests while the slave does the processing. Since the game will keep track of the players scores and player names, a stateful server would be most appropriate. Creating an instance of the game ensures that the players information and progress stay consistent throughout the session.
6. **Security**: When it comes to security, a lot of users will agree that Linux based OS are the most secure compared to other OS. Open-source software means that anyone can look at the source code to check for any malicious code. With many eyes on the Debian source code and the rigorous testing done on packages for the stable version it’ll be difficult to get away with inserting your own virus. On the client-side features like authentication and two-factor authentication can be implemented. When users log in to a new computer it can trigger two-factor authentication that’ll require the user to enter a second form of authentication in order to prevent malicious users from hijacking your account. Passwords alone are not enough since there are algorithms that can crack easily guessable passwords. Ultimately security falls on the people both on the server and client side to protect themselves from phishing attacks. You can have the most sophisticated security system but if an outsider gets private information from someone inside, it can be disastrous as we have seen in the past. This is why many operating systems use the Principle of Least Privilege with Linux giving Sudo privileges to only those that require it.

Works Cited

B., Richard. “What Is Apache? An in-Depth Overview of Apache Web Server.” *Hostinger Tutorials*, 6 Feb. 2023, www.hostinger.com/tutorials/what-is-apache.

Both, David. “An Introduction to Linux’s Ext4 Filesystem.” *Opensource.Com*, opensource.com/article/17/5/introduction-ext4-filesystem. Accessed 9 Aug. 2023.

“Debian 12 Bookworm Released.” *Debian*, www.debian.org/News/2023/20230610. Accessed 9 Aug. 2023.

Tucakov, Dejan. “Linux vs. Microsoft Windows Servers, the Ultimate Showdown.” *phoenixNAP Blog*, 5 Dec. 2022, phoenixnap.com/blog/linux-vs-microsoft-windows-servers#:~:text=Advantages%20of%20Linux%20Servers&text=Due%20to%20its%20low%20infrastructure,without%20exposing%20the%20program’s%20core.

“What Is Nginx? A Basic Look at What It Is and How It Works.” *Kinsta®*, 26 Jan. 2022, kinsta.com/knowledgebase/what-is-nginx/.