

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 12/9/2021 | Rosario Robinson | Discussing methods of improving Draw It or Lose It app via design and software requirements given by the client, The Gaming Room. |

**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)

Draw It or Lose It is a game that a new client, The Gaming Room, has created. Draw It or Lose It focuses on teams that compete to guess what is being drawn via a rendered image. The game has four rounds of which each lasts one minute. There are other time limits included, such as the 30-second mark for the rendered drawings, and a 15-second time limit for a final guess for each team. The Gaming Room wishes to expand its game’s environment to implement each of these ideas, but this requires the help of developers and updates of their software (and hardware).

The solution for this software design problem would be to create an environment that not only includes the information listed above but to be accessible on all major platforms for customers. This will require time and a look into the design constraints given.

## [Design Constraints](#_2et92p0)

Design constraints for this game application include having the ability to have one or more teams involved, having multiple players assigned to a team, having unique IDs for each game and team name, and having only one instance of the game exist in the memory at any given time.

An additional design constraint that The Gaming Room has on Draw It or Lose It is that is currently only available as an Android application. This means that expanding it to operating platforms on Mac, Linux, and Windows is important for them to serve as many customers as possible.

## [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 class diagram consists of seven classes: ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player. The program driver has the main method code that runs and tests the game application's functionality. The singleton tester focuses on displaying the singleton pattern, which contains a local reference that uses the program driver class to execute the code. Next, we have Entity, which is one of our main classes. This class is extended to other classes: Game, Team, and Player. By being the main class, this class will hold all the common attributes found in the other classes and will essentially be the umbrella for the other classes. This also makes it easier for running the game, team, and player classes. Game, Team, and Player are all classes that not only have specific attributes acquired for each but also focus on the client’s requirement of having unique identifiers within the methods. GameService is its own unique class within this package and is a singleton service for the game engine. This class includes identifiers for team, player, and service, as well as attributes that allow for new games to be added when the program runs.

This entire UML diagram focuses on many object-oriented programming principles. One that is evident is inheritance, where attributes from entity are inherited through the other four classes, as shown in the diagram below. The same goes for ProgramDriver and SingletonTester – SingletonTester inherits code from ProgramDriver in order to run the main code provided.

**"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** | With Mac, they have a built-in server, especially through their IDE XCode, which can allow for web-based hosting to occur. In addition, Macs come with an Apache web server that allows for hosting and allows Mac users to host web-based software. The built-in terminal allows for multiple functionalities as well. | Linux servers are relatively easy to navigate, like Mac servers. Linux is great to use for hosting web-based software applications because it has great built-in security and allows for more customization features, unlike Mac’s system. Linux, also like Mac, can have certain limitations based on what will be required on the back end. For instance, Linux uses more of MySQL and PHP, while Windows would use more features like .NET that works best on their servers. The Gaming Room would utilize Linux for more robust features over Mac but would have to decide if it’s a better option than Windows. | Considering Windows is the most well-known and popular out of the three operating platforms, they have full capabilities to host a variety of web-based server applications like one for The Gaming Room. However, one pitfall, similar to Mac is that their servers are not as robust as Linux is. Although they allow for more features outside of Microsoft-based products, they don’t have as many features and applications available as Linux does. | Mobile devices are most likely the most limited out of the operating platforms since this is highly dependent on what type of phone you are using and what software and programming language to create the application. With Mac, they are restricted to Apple’s App Store and would most often use XCode and Swift to host web-based software for iPhone users. With Linux and Windows servers, they have more flexibility with using Android devices to upload and host web-based software applications. |
| **Client Side** | For the cost, Macs are considerably more expensive to work on versus their Windows and Linux counterparts. Mac computers and desktops are quite expensive, so working on them would be more costly, which is something that The Gaming Room will have to take into consideration. In addition to this, Mac systems may not be able to support multiple features that may be available on other operating platforms.  Time spent could be faster and more efficient working on Mac since its software makes it easier to work on for developers. Expertise, however, would require someone to know Mac systems specifically, as they have certain features that only work or are available on their platform than what Windows and Linux will have available. | Cost and timewise, Linux would be better compared to Mac’s operating platform. The reason for this is that Linux is significantly cheaper than Mac, especially since it can be installed on a variety of computers and is not limited to being only on one type of system like Macs are. Expertise on this could be less than what it is for Mac-based systems since Linux and Windows developers can have more overlap in comparison to Mac developers. | Similar to Linux systems, Windows has much more flexibility in terms of cost and time constraints in the software development process. Windows are the most widely manufactured out of the three operating platforms, which means they have more readily accessible computers and devices available to develop and test on. Expertise is similar as well, but it is necessary like with Mac developers to know Windows’ system well in order to perform and create the best software. | As previously stated, mobile devices are dependent on the type of device the user(s) will have and the software and programming languages featured on them. This means that the cost, time, and even expertise can vary, which can sometimes be inefficient for The Gaming Room’s application. However, depending on if they want to focus on a specific operating platform, that will allow for developers to manage the code and software that can be used to be used across multiple devices. Since there are native applications and web applications, this allows for developers to create the best product outcome for The Gaming Room’s application. |
| **Development Tools** | For Mac platforms, the programming languages that they most identify with is Swift, as it is native to Mac’s operating system. In addition to this, HTML, CSS, JavaScript, C++, Java, Python, and Ruby can be utilized here as well.  IDEs that would be best used on Mac are XCode, which is a native IDE to Mac, and additional IDEs that are supported are Atom, Visual Studio Code, PyCharm, and Eclipse. Mac does support a variety of IDEs, but these are the most used and would be best for the type of software for The Gaming Room’s software. | For Linux platforms, the programming languages that can be utilized are similar to that of Mac: Python, C++, C, Perl, Java, and many more. The main difference here is that Mac developers tend to handle the Swift language, although it is based on Objective-C language. As for IDEs, it is also like Mac, where Eclipse, Visual Studio Code, Pycharm, and IntelliJ are utilized. XCode is the only IDE that cannot be run on Linux systems. | Windows is also very similar to Mac and Linux platforms in terms of programming languages and IDEs. Java, JavaScript, C#, PHP, and Ruby are the most used programming languages within Windows systems.  Since Visual Studio Code was created by Microsoft, it is the top choice for Windows developers specifically. Eclipse, PyCharm, IntelliJ, and even XCode can be used on Windows. XCode can be run on Windows operating systems, but only via a virtual machine. | Depending on if developers are doing native applications, hybrid, or web-based, the programming languages and IDEs can vary more than the three specific operating platforms.  For instance, depending on the operating platform chosen, hybrid applications would be the best option. In this case, using JavaScript, HTML, and CSS is the best option(s). Python, Java, and Swift can also be used, but utilizing the languages best for hybrid applications is the best choice (as well as frameworks). The IDEs that would work best here would again, depend on the operating system itself. Android developers use Android Studio, IntelliJ, and Eclipse, while Apple developers may use XCode. |

**References**

Nelson, T. (2021, April 25). *How to use Mac's built-in Apache Server to host a web page*. Lifewire. Retrieved November 28, 2021, from <https://www.lifewire.com/use-your-mac-to-share-web-site-2260400>.

Rothman, E. E., Rosen, R., & Jepson, B. (2008, September). *Mac OS X for unix geeks, 4th Edition*. O'Reilly Online Learning. Retrieved November 28, 2021, from <https://www.oreilly.com/library/view/mac-os-x/9780596520625/>.

Weikel, J. (2019, April 20). *Guide to linux server OS & hosting*. WHdb. Retrieved November 28, 2021, from <https://whdb.com/guides/linux-server-os/>.

Williams, M. (2021, June 25). *Best windows hosting services of 2021*. TechRadar. Retrieved November 28, 2021, from <https://www.techradar.com/news/best-windows-hosting>.

Pam. (2021, July 8). *Best language for mobile app development-what you need to know*. Webiotic. Retrieved November 28, 2021, from <https://www.webiotic.com/best-language-for-mobile-app-development-what-you-need-to-know/>.

Gewirtz, D. (2019, April 8). *Windows, Mac, or linux? we compare the pros and cons of these computing platforms*. ZDNet. Retrieved November 28, 2021, from <https://www.zdnet.com/article/windows-mac-or-linux-we-compare-the-pros-and-cons-of-these-computing-platforms/>.

## 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**: In terms of operating platforms, the best option for The Gaming Room to expand Draw It or Lose it to other computing environments would be Linux. My reasoning for choosing Linux over other platforms such as Mac and Windows are because Linux has the most flexibility in terms of what it can handle versus the other servers available. For instance, aside from its great built-in security system, Linux also has customization features that I believe can essentially be used to build and be accessible on other servers/platforms.

If I were to use Mac's server, it wouldn't be as easy to utilize in other computing environments. Their servers are based around being used for Mac systems only. With Windows, although it has more features and can be more accessible than Mac, it is not as robust as Linux servers can be in terms of capability and features available. Overall, Linux is the best option here to use to be as adaptable as possible on other platforms, including mobile devices.

1. **Operating Systems Architectures**: Linux's architectural system has four specific layers - hardware, kernel, shell, and utilities (tutorialspoint). The hardware consists of the disks, such as the hard disk drive and RAM. The kernel is the main component of Linux's operating system, which interacts with the hardware. The shell essentially takes commands from the user and executes functions from the kernel (geeksforgeeks). Lastly, the utilities are comprised of functionalities for the user and what the user would be able to see. Linux has a complex but efficient system that will improve Draw It or Lose Its software capabilities.
2. **Storage Management**: An appropriate storage management system to use with Linux's system is indexed allocation. I referred to this storage management system in last week's module to answer the question of which storage system would be best for Draw It or Lose Its application, especially while expanding to other platforms. Indexed allocation supports direct access to files while also handling more free space when needed for the application (Silbershatz, et. al, 2009, section 11.4.3). This means that developers and users alike will be able to access each file, when necessary, especially when dealing with high-definition files in the game.
3. **Memory Management**: Memory management is an important part of how Draw It or Lose It will utilize memory during the game. The operating platform that I have recommended, Linux, uses memory techniques slightly different from servers like Mac and Windows. Linux uses a system to handle memory via demand paging and virtual memory (javatpoint). The operating system also allocates physical memory when needed to help free up memory space (Silberschatz, et. al, 2009, section 21.6). Demand paging refers to a technique that is used in virtual memory systems that are loaded only when needed during the execution of the program (Silberschatz, et. al, 2009, section 9.2). Utilizing virtual memory allows for Draw It or Lose It to use Linux's memory management system to ensure that each user will have enough memory while playing the game, as well as being on multiple platforms. Overall, Linux's operating platform and its memory management techniques will allow the Draw It or Lose It software to run at its best performance.
4. **Distributed Systems and Networks**: The Gaming Room wishes to have Draw It or Lose It to communicate amongst different platforms, even while using the platform that I have chosen, Linux. In the case of Linux, however, it can support not only UNIX-TO-UNIX communications but also non-UNIX operating systems like Mac, which uses an entirely different network (Silberschatz, et. al, 2009, section 21.10). Networking within the kernel is implemented via the socket interface, protocol drivers, and network-device drivers. Each of these layers deals with a different part of the network in order to communicate properly with Linux's system. Socket interface deals with the user endpoints of communication and helps access protocols within the system (Silberschatz, et. al, 2009, section 3.6.1). Protocol drivers deal with framework and packets, which includes the important TCP/IP protocol, which allows hosts to route anywhere on the network and ensures that reliable connections are made (Silberschatz, et. al, 2009, section 21.10). Lastly, we have the network-device drivers that deal with the IP packets that support routers (Silberschatz, et. al, 2009, section 21.10).

There are also popular distributed systems that Linux has available now. A few systems are Red Hat, Debian, and Caldera. There are many more distribution systems available, but through these systems via Linux, they have package-tracking databases implemented to help packages to install, upgrade, and remove without hassle. This is another convenient part of Linux's system that will benefit Draw It or Lose Its ability to communicate across multiple platforms.

1. **Security**: Security is always a component that must be taken into consideration. With Linux, there are two specific models they focus on: authentication and access control. Authentication focuses on the user's password and identification to ensure that no one else can access the system aside from the user. Access control provides a mechanism to check to see if the user has a right to access specific information (Silberschatz, et. al, 2009, section 21.11.1). With authentication, to ensure that no one can access a user's password information, Linux implemented the pluggable authentication modules, or PAM system to allow authentication modules to load on-demand to a system-wide configuration file (Silberschatz, et. al, 2009, section 21.11.1). This essentially means it allows for updates to the users' passwords and important information when necessary. This will be important for protecting users not only on Linux-based systems but on various platforms as well.

With access control, Linux uses unique numeric identifiers to access certain rights or objects (based on if it's for a single user or a group of users) (Silberschatz, et. al, 2009, section 21.11.2). Essentially, these security protocols that Linux has established helps the user to only access information needed to log in or access their personal information. With Draw It or Lose It, Linux's secure systems allow the client to ensure that their users' private information is being kept protected while running the gaming software. Linux also has firewalls implemented for additional security to ensure a user's information stays safe. This secure software will also protect user information across multiple platforms, including those on Mac and Windows servers. Although each of these servers has its own security protocols in place, Linux's server ensures that security measures will be taken, even while distributing software and being utilized on multiple platforms.

**References**

Silberschatz, A., Galvin, P. B., & Gagne, G. (2009). Operating System Concepts, 8th Edition. O'Reilly. John Wiley & Sons. Retrieved December 5, 2021, from <https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/>.

Linux memory management - javatpoint. [www.javatpoint.com](http://www.javatpoint.com). (n.d.). Retrieved December 9, 2021, from <https://www.javatpoint.com/linux-memory-management>.

Operating System - Linux. Operating system - linux. (n.d.). Retrieved December 9, 2021, from <https://www.tutorialspoint.com/operating_system/os_linux.htm>.

Introduction to linux operating system. GeeksforGeeks. (2021, February 18). Retrieved December 9, 2021, from <https://www.geeksforgeeks.org/introduction-to-linux-operating-system/>.