

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 | 06/23/2020 | Dustin Haugh | <Brief description of changes in this revision> |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

Creative Technology Solutions (CTS) has entered into an agreement with The Gaming Room to provide technology consulting to further their development of the game DRAW IT or LOSE IT!.

The Gaming Room would like Creative Technology Solutions to:

* Explain the operating system architectures for different operating systems.
* Analyze the differential file system components that represent a collection of data.
* Explain the functions of memory and storage management.
* Identify memory management techniques.
* Describe distributed systems and the networks that interconnect them.

The operating systems that will be included in this review will be Unix type systems such as Linux, Mac OS X, and the newest iteration of Windows 10.

## [Design Constraints](#_2et92p0)

The proposed solution would adapt the app DRAW IT or LOSE IT that has been produced primarily for the Android operating system, but adaptations need be made for other operating systems with similar functionality, design, and behavior.

Specific Server Design Constraints

* Each operating platform must offer a server-based deployment method where the website will be hosted.
* Potential licensing costs to the client (The Game Room) need to be listed and considered.

Specific Client Design Constraints

* Integrate various operating platforms, such as iOS, Android, and traditional desktop-based operating systems; Windows 10, Linux, and Mac.
* Capable of communicating with the back-end web application running on the server.
* Completed as a modern, responsive HTML interface running inside the web browser.
* The software development considerations (cost, time, expertise) necessary for supporting multiple types of clients needs to be evaluated.
* The development process must ensure the application is compatible with all web browser platforms and mobile devices.

Specific Design Requirements

* Programming languages must be supported cross platform.
* IDE’s used must be capable of deploying on each platform.
* Multiple teams might be necessary to deploy.

## [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)

<Describe the UML class diagram provided below. A text version of the diagram can be found [here](https://learn.snhu.edu/d2l/lor/viewer/view.d2l?ou=6606&loIdentId=24238). Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

****

## [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**  **Back End Development** | Mac is a proprietary operating system from Apple, is expensive, has less third-party software options, but better driver support than Linux.  Java can also be developed and deployed using various IDEs. | Linux terminal is easy to work with and contains a library of BASH commands. Linux is free and has server support as well. However, it does not have good driver support. Open source, so the kernel can be modified. With that, it also requires the most expertise to work with.  Linux does not have the largest selection of third-party software, but it does have the necessary IDE’s for development and deployment.  Java and Java APIs can be used for development in Linux to create the server utilizing HTML protocol. | Both Windows 10 and Windows Server are good operating systems to host online. More people are familiar with these operating systems and Windows comes with more third-party software options. | A server should be hardwired and immobile, so a mobile device would not be suitable for a server application. It could easily drop service resulting in a loss of the distributed system in its entirety. |
| **Client Side**  **Front End Development** | For this desktop OS there are the same  web browsers available as other desktop operating systems.  Java is a multi-operating platform language that can use eclipse for each operating system.  Development costs, time, and expertise can all be reduced if we utilize a similar language and IDE for each operating system platform.  <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | Linux might be harder to support due to a reduced availability of development tools.  This may not be true depending on our needs, but less third-party software is developed for Linux. | We can expect the costs associated with developing software on windows to be a little higher than Linux, but about the same as Mac OS.  We can expect to have better support and more available development tools.  The difference might be insignificant. | Java is all that is needed to develop apps for Android  Objective-C and Swift are the two main software development tools for iOS. Costs might be a little higher for development along with a smaller list of users than Android. |
| **Development Tools**  **Full Stack Development** | Most programming languages can be used on different operating systems. I would imagine this is because they all work with the same CISC processor architecture these operating systems work with, so they would compile largely the same with differences in file allocation.  For full stack development we will be using HTML for server/client interfaces. This will be used within our APIs.  Full stack development languages can include HTML, CSS, JavaScript.  I believe that we can still utilize Java APIs to easily create our server architecture and authentication protocols. | The same development tools are used for Linux | The same development tools are used for Windows | I believe that Java is the main development language for Android, but other languages are available to use as well.  To keep things simple, and where we are already going to stick with Java for server architecture, we will just use Java again for this.  Java is also one of the most supported language for google (who developed android).  Also, we will need Android Studio and Xcode for iOS. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

There are two options that stand out as an operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments. That would be to either use a Linux distro, such as Ubuntu or Debian to set up a Linux server or to migrate to a cloud service if the user demand gets too high or unpredictable.

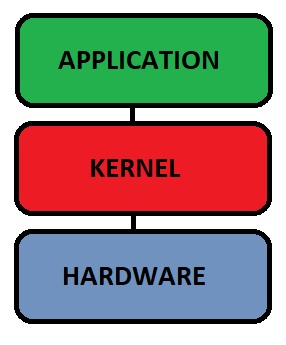
Linux stands out among other operating systems when it comes to stability and reliability, security, flexibility, and total cost of ownership.

We will set up a Linux server that will allow us to set up a client/server distributed system allowing us to operate on multiple client operating systems.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

The first operating system chosen was Windows 10, but after learning about its architecture and vulnerabilities, I have changed to recommend a Linux plus GNU distro, either Ubuntu or SteamOS which is a Debian-based distro.

At the base of any computer architecture we begin at the hardware. Above that we will have the kernel that implements memory isolation for applications, handle system calls, implement a file system, schedule processes, and talk to hardware utilizing drivers. Drivers are a grey area not completely defined as part of the kernel or not, but some would classify it as part of the kernel. Then, a user application can run within the kernel.



Windows based operating systems have a few extra layers between the kernel and the application. These would be DLL (Dynamic Link Library), System Services, and run-time libraries.

The DLL helps facilitate dynamic allocation of memory. This allows different processes/applications to share memory, thus utilizing resources more efficiently. However, the DLL layers create opportunities for malware to cause problems by injecting itself into executables via the DLL.

Windows utilizes a micro-kernel which tries to limit how many processes are being run within the kernel at a time while Linux utilizes a monolithic kernel where many services are run at a time.

Linux begins with the boot loader that instantiates the kernel. Just as in the simple diagram above, in Linux the kernel starts and manages the CPU, memory, and other hardware resources. Operating system tools are used for applications to interact with the kernel. These tools are separated from the Kernel and are distributed in different versions. From here there might be a windowing system and environment “shells” that interact with the operating system tools. And very last, applications will work within the shell. The different shells are what differentiate between different distributions of Linux.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

For this application, if we have a small user base and we are able to utilize a Linux server then we can set up a local Linux database and file system utilizing Bash. However, we can also migrate our server to a cloud and utilize a cloud database as well. This would be perfect for irregular client usage and more traffic than a hardwired server might efficiently handle. Cloud based servers and databases are versatile and can be more cost effective.

Linux is a stable operating system that can stay on for years. Updates can be made to the system without restarting as well, but for more stability the cloud is still a better service. Within a cloud service multiple game services can be instantiated at any given time on a distributed system giving it an even better stability.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

As mentioned before, Windows utilized dynamic memory linking, making it possible for multiple processes to share limited memory resources. However, this poses a security problem as malware found a way to inject into the DLL.

Linux on the other hand utilizes virtual memory which is a combination of both RAM and disk space (called swap space). The kernel keeps memory separated between processes.

Both operating systems utilize a hierarchical file system that can be given access permissions and are fairly secure.

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).>

Utilizing a client-server architecture over a cloud might be the best overall plan to avoid outages and guarantee connectivity. Local servers, even while running Linux, run the risk of power outages and interruptions. Cloud services can be distributed among different locations and on multiple servers, thus increasing their reliability to the client.

Cloud migration also means more resources, more game services that can be instantiated concurrently, and a pay-as-you-use model that might make it more cost effective.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

The main reason why I have decided to go with a Linux based operating system to use as a server or a cloud service are the security risks of Windows.

Linux has some advantages. The biggest advantage might be that it has been supported by thousands of individuals and an incredible number of invested companies, so it is supported and tested by an incredible workforce that is less likely to let security threats go unnoticed. A user can also choose to update with new security and other features at any time, or never.

Linux has more stability as well. It has a reputation for running without having to restart and will not likely crash.

Also, applications do not have access to memory allocation as they do in Windows operating systems. This gives an added layer of protection if we choose a Linux server.

Where this game system does not pose a large security risk to the individual if it is compromised, I do not believe that we need to have a secondary authentication service, but if the client would like it added, that is an option to add an extra layer to user account security.

Many cloud services that are offered today are utilizing real-time, monitored firewalls and security to protect the cloud infrastructure itself. However, it can be recommended to incorporate security layering to add protection to internal resources. This is where secondary security services can be obtained, or added built-in-in security controls or cloud firewalls.