

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 | 09/19/2020 | Noah Sherry | Initial Design Template review |

**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 would like to create a web-based version of their current Android game, Draw It or Lose It. Draw It or Lose It is a multiplayer picture guessing game, similar to Win, Lose or Draw from the 1980’s. The web-based version of their game needs to serve multiple platforms, not just Android exclusively. The Gaming Room has outlined a few requirements, which will be described in the Design Constraints section below.

## [Design Constraints](#_2et92p0)

* The game must be a web-based application and run in multiple different environments (iOS, Android, Windows, Mac, etc.)
* The game must utilize network connections to allow for multiple players to connect to the same game instance.
* The game must have unique team-names and must allow users to check name validity before submitting a team name for themselves.
* Only one instance of the game may exist in the system memory at any given time, so that all players will be connected to the same game.

## [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 Entity class defined below will serve as a base class for all objects in the game. Entity contains the properties id and name, which each object will use to uniquely identify itself. The Game, Team, and Player classes will all extend from the Entity class so that they may inherit these properties. The GameService class will serve as the system manager for coordinating and driving the individual game instances. As such, the GameService may be related 0 to many instances of Game objects. The Game class is related to the Team class in the same manner and thus may be related to 0 to many instances of Team Objects. As expected, the Team class carries this same relationship to the Player class and may be related to 0 to many Player objects. Finally, The ProgramDriver class initializes the one and only instance of the GameService, as well as uses the SingletonTester class to perform validation at run-time.

****

## [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 software licensing is very expensive, but Macs also have great OS security. Cloud-hosted Macs are possible, but can get pricy with higher performance needs. Some example of mac hosting are <https://www.macincloud.com/> and <https://www.macstadium.com/> . Mac virtualization is also possible, but this circumvents most of the inherent OS security that MacOS provides. | Linux has no licensing fees, and UNIX-based kernels are open-sourced. Cloud hosting for Linux servers is very popular and extremely cheap, due to no licensing fees. Linux OS Distros tend to have fewer security vulnerabilities in the OS itself due to them being open-source. Virtualization/containerization are common and provide great benefit here. | Windows has lower licensing fees that MacOS, but still some. Windows OS tends to have many more security vulnerabilities than either Linux Distros or MacOS, due to its popularity. Containerization/virtualization is possible, and a great fit for Windows OS. | While it technically is possible to run a web server out of a mobile Android or jail-broken iOS device, its incredibly unsecure and not worth mentioning further. |
| **Client Side** | As the program will be running out of a web-browser on all of the client devices, the only considerations when using MacOS that needs to be made is that the game supports most modern web-browsers such as Firefox, Google Chrome, Safari, and Microsoft Edge. Safari has some special browser rules regarding CSS and HTML tags that will need to be considered. | Much like MacOS, so long as the client-side web applications runs in modern web browsers, no special client-side precautions or considerations need to be made for Linux Distros specifically. | Echoing the previous two platforms, Windows needs no special considerations for a web-based client application. Internet Explorer support is a possibility but would severely limit the scope of web frameworks possibly used to make support. | There are two possibilities for developing mobile client-side web-based applications. The first is to simply have the mobile devices run their web browser to connect to the game. The second is to use mobile app web-views to wrap the site in a native application. In either case, a mobile version of the website will have to be made, or the desktop site will need to be responsive to all screen sizes. |
| **Development Tools** | For server side development, a Java runtime environment / development kit will be required. We’d also need a web-hosting application like XAMPP to host the site. A Java IDE like Eclipse will be needed. Client side development will simply be a website, so most generic IDEs will work. We could use ReactJS, AngularJS, or countless other JS frameworks to create the web client site. | Much like MacOS, a Java runtime/development environment will be required. XAMPP or LAMP or other web-hosting programs will be required for server deployment. A Java IDE will be needed. Linux shares the same client requirements as MacOS for web-based client site. | Windows also needs a Java runtime/development environment. Windows can use programs like XAMPP, but Windows also comes with it’s own web hosting program called IIS. Like Linux and MacOS, a Java IDE will be needed. They all share the same client development reqs. | Mobile client side development tools needed depends on whether or not native web-view applications will be made. If not, no additional development tools are needed beyond devices to test the website on. If native applications are being used, then a Mac with Xcode will be needed to develop the iPhone wrapper and a PC with Android Studio will be needed for the Android web-view wrapper. |

## 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**: The operating system I recommend for the Draw It or Lose It application that will allow it to expand into other computing platforms is the Linux Operating System (the specific distro isn’t as relevant). This is because the Linux OS can be compiled and run on any system architecture, instead of being restricted to Intel’s x86 architecture like MacOS. This would allow for the application to run on any hardware system, even ARM based processors. This provides the maximum possible amount of scaling to other systems, with the added bonus of no licensing fees like Windows or MacOS.
2. **Operating Systems Architectures**: Linux-based OS’s are UNIX-based operating systems that use the Extended File System and Unified File System storage formats. Linux Operating Systems are open-sourced and community-developed and scrutinized, thus producing a more refined and secure platform for production-scale applications.
3. **Storage Management**: Because the Draw It or Lose It game application uses roughly 1.6 GB of files within the application (not including executable size), the application should simple take advantage of the NFS file shares available within Linux to create a small partitioned repository of images to pull from. This ensures that the game will be the only application to access these files and will reduce the chances of file-locks taking place.
4. **Memory Management**: Linux operating systems generally use paging over segmentation within memory management, as certain RISC instruction sets don’t generally have support for segmentation all that well. To help with this, Linux also creates memory “zones” to classify sections of memory and tell the system what it will have access to versus what the applications will have access to.
5. **Distributed Systems and Networks**: Linux operating systems work exceptionally well with containerization, which allows for the use of Kubernetes or other container frameworks to create many server nodes that the application can run on. A distributed system of server nodes running the game can severely mitigate the risk of network or system outages, if done correctly. For example, I would recommend creating clusters of containers using separate hosting services so that if any one cluster or node inside those clusters goes down, the worst effect to the user might be a disconnection from the game.
6. **Security**: Linux operating systems treat everything as a file. Access to mounted drives is obtained via a special file in the filesystem. As such, the traditional security model for Linux OS’s is to assign read, write, and execute access to every file or folder within the OS. To protect the security of the game, the application can be given exclusive access to the images it needs for each game. As for network security, Linux web hosts have complete HTTPS support as well as native support for end-to-end encryption in some distributions.