

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)4

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | March 30, 2021 | Matthew Powers | Final view based on full knowledge employed from Operating Systems Concepts book |

**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, our newest client, wants us to develop a web-based game based on their existing game, Draw It or Lose It which is currently only available on Android. Their new game should be portable to a few new platforms starting with iPhones.

## [Design Constraints](#_2et92p0)

* It must be compatible with each relevant marketplace
* Must be playable by each platform on which it is played
* Should be based on a similar style to their existing game
* Should likely use Java to communicate between platforms, even if other languages are used to develop the GUI and handle the client-side work.
* Possible to build upon it with future updates as usability in each language improves

## [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)

Each of the following classes, game, team and player are all inheriting the id and name fields from Entity along with the get<info> functions. Starting with GameService, each of these has a list of the smaller classes leading down toward Game, Team and Player classes, respectively. The GameService class maintains the solitary service which is used for the course of the game. It is also responsible for providing identifying characteristics to each of the subclasses of Entity. By using that single class to provide said identifying characteristics, we can maintain a level of uniqueness between each separate entity, whether they are on at the same time or not. This in turn, improves the security which was mentioned in the design constraints. By employing that uniqueness, we can minimize the memory resources which may be in use at any given 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.

In order to appropriately follow

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Although the listed [requirements](https://www.apple.com/macos/server/specs/) don’t ask for much, the construction of the OS may not be the best choice. The kernel structure of Mac’s OS reduces security by allowing more direct communication with the core kernel. The structure will, however, reduce the amount of time necessary for communication with any required components of the OS. | Down below, I outline some of the major strengths and weaknesses associated with hosting a system. Among them include the creative use of the file system and the employment of the security system. The major drawback to using Linux comes down to unallocating memory. | When looking at the capabilities of a [Windows Server](https://download.microsoft.com/download/B/F/0/BF00B689-94F5-4E12-8159-7804D0C27C09/Windows_Server_2019_Feature_Comparison_Guide_EN_US.pdf), we can see that it eases transition to a cloud server, receives support from Microsoft directly and allows for security through authorization practices | Although most mobile devices have more memory, storage and processing speed than early computers, the expected level of server calls from a growing marketplace may be too much for the system to handle |
| **Client Side** | Just as with any programming initiative, the total cost, whether it may be in money, manpower, time, or something else entirely, will come from understanding of the tools available. Thankfully, the [ease](https://www.computerworld.com/article/3219732/12-reasons-to-learn-apples-open-source-swift-language.html) of learning and understanding the tools make it rather simple to find somebody who can follow the concepts. | As the language of choice is typically C++, a relatively old language which constantly receives updates, the costs associated with building the system come down to complexity of said system. | Once again, the popularity of the related languages lends the cost of development a savings not just in money, but also in time. | Due to the recent growth in the Mobile market, the difficulty to code something of value is slowly shrinking as the number of amateurs willing to get real hands-on experience grows. |
| **Development Tools** | Mac is built on a combination of languages, but the most used one is Swift. Swift is originally developed by Apple, but over time, it’s been transitioned into an open source language. This custom language, coupled with the [development environment](https://developer.apple.com/xcode/) available for download by Apple, will ensure an easy road to follow when heading toward final deployment. | Linux is primarily developed using C++ and as such can be deployed in a number of development environments including eclipse and visual studio to name a few | Just as with Mobile Devices, Windows deployment will be fairly straightforward. The primary languages in use include C++, Java, Python, and C#. Due to their popularity, most IDE’s should handle the languages. Within each of those are plenty of networking libraries capable of constructing the inter-device needs of the application. | The primary tools and languages will depend largely on how the system is deployed. If it is dependent on a web-interface, it will require maven or some other form of dependency which allows for online connectivity. Otherwise, most major languages will do so long as the program can be scanned for viruses by the relevant app store. |

## 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 most appropriate operating platform on which to host the game is Linux. This is based on its current existing application as a leader in [security](https://www.linux.com/what-is-linux/). In addition to this, the speed at which the file management system will be able to proceed with function calls outshines other major platforms.
2. **Operating Systems Architectures**: Much like the original build of Linux, the current one is built on a main kernel which manages most system functions. Next, the system libraries define many of the functions which do not need kernel authorization. Finally, the system utilities handle individual specialized tasks such as configuring aspects of the system or managing network traffic.
3. **Storage Management**: The file management system outlined with respect to Linux in the text, Operating Systems Concepts, 8th Ed. Ch. 21, is the second extended file system. As part of that file system, it places logical adjacent blocks into physically adjacent places on the HDD. As a result, a function call at one point in a program will be next to the following function call. By doing this, the HDD spends less time searching for the machine equivalent of methods or programs and more time sending those methods into logical memory and the processor.
4. **Memory Management**: As described in the text, Operating Systems Concepts, 8th Ed. Ch. 21, the version of Linux outlined uses a buddy paging system to maximize the available memory and allocates memory for various function calls using a system function. The primary downside though, is that the memory allocated must be manually deallocated using another system function.
5. **Distributed Systems and Networks**: In order to address the potential loss of data mid-game/service, one such way is to employ a cloud service to manage such costs. My recommendation there is to use Amazon Web Services. Based on their own [claims](https://aws.amazon.com/lambda/), they monitor traffic down to the millisecond and allows for easy scaling between platforms. They also have the lowest [cost](https://www.parkmycloud.com/blog/cloud-storage-cost-comparison/) amongst various cloud server operators. Further, by diverting the costs of managing the server onto a third party such as Amazon, our client, The Gaming Room can focus more of their time on marketing their products.
6. **Security**: Based on the current Linux architecture outlined in Operating Systems Concepts, 8th Ed. Chapter 21, the authentication and authorization system uses a one way cryptography method employing a one-way “salt” value to encrypt the information. The high number of character combinations along with the timeout protocol increase security and would deny most automated attacks as well as man-made attacks.