

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 3.0 | <08/11/21> | <Aaron Shipley> | Revised Document- Architecture Reccomendations |

**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 needs a web-based-game that can serve multiple platforms based on its popular game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show from the 1980s. The game will be available on all mobile and web based operating systems.

## [Design Constraints](#_2et92p0)

* Needs to be available on multiple platforms (IOS, Android, Windows, Linux, Mac)
* Multiple teams and multiple players present within the game
* Falling within budgetary constraints to meet multi-platform designs
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time.

## [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 is our super class, and the Game, Team and Player classes all inherit directly from it. The GameService has a reference to the Game class, which has a reference to the Team class which has a reference to the Player class. All of these classes having a reference to another is done through aggregation. Also included are our SingletonTester and our ProgramDriver class. Our ProgramDriver class is where any executions of our application take place, and it uses our SingletonTester class. The SingletonTester class also has a use relationship with the ProgramDriver class. The way these classes are set up allows for one game to go on at a time with multiple teams (from a list) with multiple players (from a list).

****

## [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 does have the Mac OSX server for use. The price ranges vary on usage rate and storage needs. Looking at larger storage needs for storing stock drawings, the price goes up quite a bit to be more expensive than other options. Security for this is top notch and constantly evolving to protect from unauthorized use / auditing. | Linux contains many distributions, such as Mint, Ubuntu, Puppy Linux, etc that are server capable. Since Linux is open source, there are endless resources out there. The open-source factor does make it more susceptible to unauthorized use or tampering. Linux is the least used of the OSes, so it would be more difficult to find skilled people to work on it. | Windows is the forerunner for OS servers to use. Their moderate pricing for server use and storage is middle of the pack. The frequent updates to the system allow windows OS to stay on the forefront of any breaches, hacks or other unauthorized use. The usability and popularity make windows servers commonplace among businesses. This makes their OS a key field of study, so there are many people that are capable of working on it. | Since most mobile OS like Android or IOS do have dedicated servers to use for an application that has been made available to mobile, one may say this could be the most financially inefficient route to go. Service like AWS EC2 and other cloud-based services offer cloud servers to use and cloud storage. These can add up, but with the pay as you go method, the cost can be justified by how much you use. It is easily adaptable to growing demand and easy to scale back when demand ebbs as well. |
| **Client Side** | Pricing of servers makes this the more expensive option up front. Cost of development will be moderate, since Mac is widely used and taught. There are numerous people that are well trained in Mac OS. The added security of Mac will help offset the initial pricing and possibly be more beneficial financially in the long run. | On the front side this will be the most cost-effective option, due to the resources being open source. The costly part will be development. Finding well trained people in Linux will be costly in order to ensure the application is done correctly. | Overall, Windows will be the middle of the road choice on cost. Server pricing is moderate, while popularity of Windows OS means more people are trained on it, cutting development time down. Windows is widely used, so it will fit most client’s needs. | Cost could be a factor in the long run. Since services like AWS are pay as you go, cost will fluctuate. Cost to develop will be higher as well due to the multiple OS on mobile devices, Mac, Windows, Android, Chrome, etc. Trying to mobilize an application and get the same experience out of it can be costly as well. |
| **Development Tools** | Swift is the common language used for Mac. An IDE like Atom is common to use in building software for Mac OS. | Popular languages to use for Linux would be Java, C++ and C#, which would be able to use the IDE Eclipse. Python is most widely used in Linux and PyCharm IDE is common for this language | Languages like HTML, C# and JavaScript would be used. Eclipse and Visual Studio IDEs are most commonly used. Visual Studio would be the best option for IDE | For android devices, Java language and Visual Studio IDE would be a clear choice. For IOS, Swift or Objective C are the languages used, while using AppCode from JetBrains (people who make PyCharm IDE) is the front runner for an IDE to use |

## 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 Gaming Room should use a cloud-based architecture with a Linux OS. Serverless / cloud- based architecture solutions transfer system maintenance and other cost-based responsibilities over to a party that works to offer custom pricing models based on leveraging scale. This pay-as-you-go type model will ensure The Gaming Room can scale up or down on a need basis for servers, storage, and other cost-based decisions that would otherwise be the responsibility of the company.

Middle layer service solutions can significantly reduce the scope and

complexity of the app’s traditional logic layer. Some functionality (e.g., user authentication, file uploads, etc.) can be accessed directly from the client, reducing latency and promoting more simplistic app architecture. A service like this is widely offered by multiple suppliers. Services that would fit the need of The Gaming Room could by achieved by AWS (Amazon Web Services) through their EC2 service, offering a scalable server service. The client can scale up or down the number of servers needed to run the application with a few clicks and be up and running within minutes. This ease-of-use style infrastructure makes it ideal in saving time and money for the client.

1. **Operating Systems Architectures**:

Traditional three tier solutions include a middle layer that executes logic and passes information between the user view and the data store. A cloud-based architecture can build on this model by way of modularization of this layer. Modularization is similar to that of microservices architecture, but on a larger scale. This will allow The Gaming Room to only have to pay for the services needed at the time, but change throughout if need be.

Similarly, the Linux OS is constructed in modular fashion. The kernel of Linux OS is a more bare-bones OS, containing just core components, while separate modules promote dynamic expansion of services. This structure delivers a lightweight solution that occupies less disk space. Modular OS solutions also bypass more latent communication methods used with layered OS solutions (Silberschatz, 2009). Linux also executes many functions by way of system libraries that do not have direct access to the OS. This can limit access to device hardware, a sort of encapsulation.

1. **Storage Management**:

Cloud-based storage, such as AWS S3 simple storage, allowing unlimited amount of object storage to be accessed anytime and anywhere, is a viable option. Having pay-as-you-need system similar to EC2, allows the client to scale up or down the amount of storage based on need at that time. The pricing is an affordable option for rapid growth or decline.

Direct access storage allows for the fastest load time and virtualization support. Virtualizationallows program design without consideration for memory capacity (Silberschatz, 2009) and is most efficient when combined with indexed, direct access storage methods. An indexed allocation of storage avoids the issue of unused storage space caused by fragmentation stemming from contiguous allocation.

**Memory Management**:

The size and amount of code in the application needs to be taken into consideration. The code must be able to be modified or appended based on need, yet not be littered with unused code that will take away from the memory, constraining the application. The CPU need only execute the application existing partially in memory. This is another benefit to Linux OS, which allows partial memory execution through virtual memory and demand paging. This will help with physical memory demand and allow multiple programs to run simultaneously throughout.

Demand paging generates page faults. These faults impact performance when not managed effectively (Silberschatz, 2009). Processes reside on disk and are broken into smaller logical segments (pages) that are only pulled into physical memory when needed for execution (Silberschatz, 2009).

1. **Distributed Systems and Networks**:

A company with an application, such as The Gaming Room and their Draw-It-or-Lose-It game, will base their success off of usage and downloads / plays of their app. Serverless cloud-based system will allow the client to scale up their needs as the popularity and usage of the app increases.

As the app scales, it will need to run across multiple servers to handle client demand. Load balancing and replication are key to successfully supporting increased client use across distributed systems.

Load balancing will essentially do what its name describes. It will balance the demand requests of the client across multiple servers, improving responsiveness and availability.

In a distributed system, the app database is usually run across several machines simultaneously. A strategy such as the Master-Slave strategy would work in this situation. This strategy leverages the idea that adding data to the database (i.e. writes or edits) is much less frequent that retrieving data (i.e. reads) (Kozlovski, 2018). When the data is modified in the master set, it informs slave nodes the change is available for subsequent retrieval.

Communication between the Client and the app will happen over the internet. HTTP is the protocol used in internet communication (Parikh, 2020). HTTP’s stateless nature creates a constraint that developers must work within to simulate a stateful experience. In order to serve multiple clients, the app should expose an API serving all clients in uniform fashion. This instance promotes multi-platform support due to the portable nature of APIs. With serverless architecture, solutions are built on top of individual functions or pieces of business logic (Bashir, 2019).

1. **Security**:

Authentication and authorization are key components to secure code, apps, and programs. They are also key to running across multiple platforms. Login credentials are key to the authentication portion of this app. The username and password are the parts of the authentication that will help authenticate a user correctly and give them proper access to the app.

When it comes to authorization, one may use the model of least privilege. By this model, the user will be given proper access to play the game, manage their profile, and manage their teams. The least privilege model will not give them access to the app in any other portions, such as altering code, making rule changes or other admin-based properties.

Linux promotes security through process independence. Each process in the system has its own virtual address space independent of other processes. This prevents a process running one application from affecting another while also protecting certain memory areas against being overwritten by rogue applications. This prevents a process running one application from affecting another while also protecting certain memory areas against being overwritten by rogue applications. Additionally, each Linux user requires authentication to use the OS and least privilege is enforced automatically. This limits scope of access and potential threats. Linux protects files and directories by way of discretionary access control (DAC). With DAC, file owners can dictate other users or groups who should have access to read or edit the file (Overview of Linux…, 2013).

**References**

*Cloud computing services - Amazon web Services (AWS)*. (n.d.). https://aws.amazon.com/.

Bashir, Faizan. (2019, July 5). What is Serverless Architecture? What are its Pros and Cons? Retrieved from https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-prosand-cons/.

Kozlovski, Stanislav. (2018, April 27). A Thorough Introduction to Distributed Systems. Retrieved from https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems3b91562c9b3c/

Overview of Linux Kernel Security Features. (2013, July 11). The Linux Foundation. Retrieved from <https://www.linux.com/training-tutorials/overview-linux-kernel-security-features/>.

Parikh, Aarti & Albert Agram. (2020). Introduction to HTTP. Launch School. <https://launchschool.com/books/http>.

Silberschatz, Abraham, Peter Baer Galvin, & Greg Gagne. (2009). Operating System Concepts, Eighth Edition. John Wiley & Sons, Inc. https://learning.oreilly.com/library/view/operating-systemconcepts/978047012872