

Draw It or Lose It

# **CS 230 Project Three Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project One Software Design** 1](#_Toc122269839)

[Table of Contents 2](#_Toc122269840)

[Document Revision History 2](#_Toc122269841)

[Executive Summary 3](#_Toc122269842)

[Design Constraints 3](#_Toc122269843)

[System Architecture View 3](#_Toc122269844)

[Domain Model 3](#_Toc122269845)

[Evaluation 4](#_Toc122269846)

[**Recommendations** 6](#_Toc122269847)

[Operating Platform 6](#_Toc122269848)

[Operating Systems Architectures 6](#_Toc122269849)

[Storage and Memory Management 6](#_Toc122269850)

[Distributed System and Networks 7](#_Toc122269851)

[Security 7](#_Toc122269852)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 0.5 | 09/25/22 | Jacob Abts | Filled out the Executive Summary, the Design Constraints, the Domain Model, the Evaluation, and the Recommendations. |
| 0.7 | 11/20/22 | Jacob Abts | Revised the Executive Summary and the Design Constraints. |
| 0.8 | 12/4/22 | Jacob Abts | Refined the Evaluation and Recommendations. |
| 0.9 | 12/11/22 | Jacob Abts | Created a section to elaborate further on the memory management. |
| 1.0 | 12/18/22 | Jacob Abts | Refined the Recommendations section of the document |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, has requested the development of a web-based game based on their current game, Draw It or Lose It. The application exists only on the Android platform and must handle multiple clients and platforms.

A game will have one or more teams while each team will have a variable number of players. Each game and team name must be unique and allow for users to verify if a name is used. Only a single instance of the game can exist, requiring unique identifiers for every player, team, and game.

## [Design Constraints](#_2et92p0)

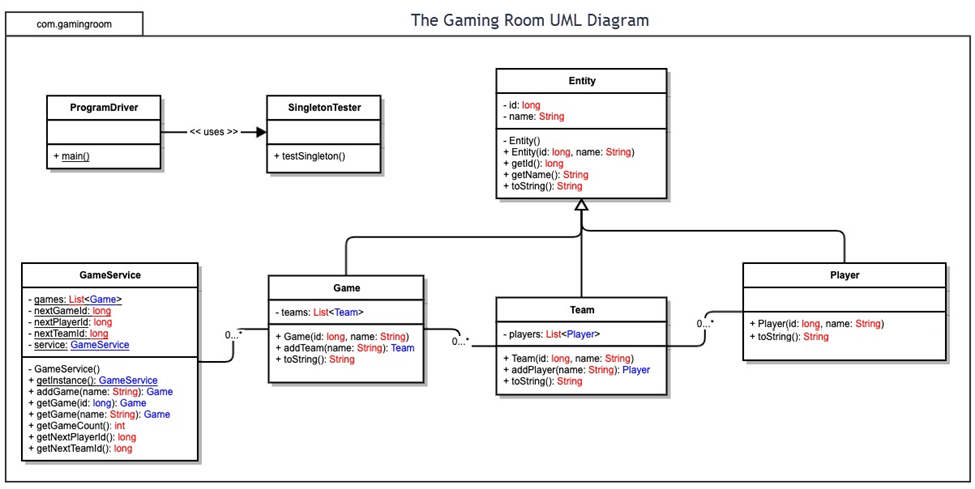
There will be three environments required for development on the hardware and operating systems, they are a development environment, a test environment, and a production environment. The hardware required for the servers can be located on-site or in cloud infrastructure. The application will require more storage to maintain the local assets for the game to load efficiently. Additionally, user authentication will change based on the platform and device, requiring a restructure of the application’s current security. This will include a login interface that is able to register teams and players. Similarly, the game may require users to be limited to being logged in on a single device. Due to the complexity of the game, there may be additional interfaces required to simplify user interaction.

## [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 ProgramDriver uses the SingletonTester to verify there is only one instance of an entity in memory at any given time. The Entity component is an abstract component that provides a base class for the Game, a Team, and a Player, this allows for the verification of each child class being unique. The GameService can handle a variable number of games beginning with none and not limited outside of hardware. The same range of Games in the GameService can be applied for Games to Teams as well as Teams to Players.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The terminal commands are flexible, this allows the server to be configurable, accessible, and maintainable | The terminal is even more flexible than a Mac and can have significant cost advantages | This OS has more software available than the Mac OS and has a larger presence than a Linux OS | A server would be more reliable and practical being anchored to a single location and on a stronger machine |
| **Client Side** | There is a decent amount of time and skill required for this OS. The cost is identical to Windows | Although this OS would be the cheapest to develop for, the difficulty and time required are significantly higher than other OS | This requires the least expertise and time required while having similar costs to a Mac | Extremely flexible for users, cheap, and still has a slightly higher level of expertise required |
| **Development Tools** | HTML, CSS, and JavaScript are excellent from end languages. Java, Kotlin, and Python are excellent all-around languages. SQL, mongoDB, and other SQL varieties are great for databases. | HTML, CSS, and JavaScript are excellent from end languages. Java, Kotlin, C++, and Python are excellent all-around languages. SQL, mongoDB, and other SQL varieties are great for databases. | HTML, CSS, and JavaScript are excellent from end languages. Java, Kotlin, C++, and Python are excellent all-around languages. SQL, mongoDB, and other SQL varieties are great for databases. | HTML, CSS, and JavaScript are excellent from end languages. Java, Kotlin, and C++ are excellent all-around languages. SQL, mongoDB, and other SQL varieties are great for databases. |

# **Recommendations**

A cloud deployment environment removes the need for concern for many of the standard issues regarding data security in any networked environment. Since servers can be entirely remotely deployed and accessed securely, they stand as an excellent alternative to physical servers.

## Operating Platform

The operating system I’d recommend is Windows as it has a low bar to entry, is not costly, and currently has a wider audience. Windows also currently has a much wider variety of add-ons to streamline development. Windows would allow for developers to have the ability to customize their environments to their needs and wants without the limitations of other operating systems. Due to the availability of the operating system, there is no bar for entry to navigate the platform and many issues are documented by Microsoft due to the large user base.

## Operating Systems Architectures

The target architecture should be x86 as it has an excellent platform for development support and is powerful. This will allow for more influence on older devices, while also being able to be applied to x64 architectures. Since most devices in the last 30 years are compatible with the x86 architecture, there will be much less limitations on specific hardware required to run the application.

## Storage and Memory Management

As technology has progressed, the demand for higher and higher resolution context grows exponentially. This poses the ever-growing demand applications must store, retrieve, and reference large portions of data. Memory can be broken down into 3 subcategories; the relatively instant access provided by caches, the random-access memory to store larger segments of information, and the slower long term storage solutions. Although caches are the fastest in terms of data retrieval, their main priority is to manage the information directly related to what a processor is trying to accomplish. This implies that cache storages are dramatically smaller than their counterparts. This is done intentionally to allow for the drives to be filled and processed significantly quicker by the processor. Similarly, the volatile random-access memory, or ram, provides a user with a staging point. This is where programs prepare to be run by the processor after being retrieved from long term storage. Processes rely on the ram to store data that may need to be retrieved quickly, such as loading a save point for someone who died in a game they are playing. Finally, the remaining memory category is the non-volatile memory storages, such as hard drives and solid-state drives. Although these drives do send and receive reads and writes, due to the slow communication of these drives the objective is to keep the interaction to a minimum. The simple way of deducing how a larger storage is inherently slower is by thinking about the time it takes to read the disk. For example, a human can read many 40-page books efficiently, but at the cost of details the stories may contain. A 300-page novel would allow for the details of the world to be worked into a reader but would also take a significantly larger amount of time to read. The separation in storage types leads too two categories, memory management, for data access frequently, and storage management, for data meant to be retained for a significant period.

For storage, due to the nature of many games being stored in the cloud; I’d recommend Azure

as the chosen OS is another Microsoft product. If a physical storage is desired an SSD is faster and more reliable than a hard drive, specifically the M.2 chips. For memory, due to the relatively simple nature of the game loop, the RAM requirements are extremely low. As a baseline I’d recommend 2 gigabits of RAM at a minimum.

## Distributed System and Networks

For the Draw It or Lose It application, the bulk of the work would be performed through memory, only using storage to store data for later use. The image being portrayed, the time left in the game, and the objects in the system will likely be held in memory. This would include the current games, the current players, the current scores, the time remaining for each game, and the progression of the game at that time. Meanwhile, any information regarding a player would be securely transferred to the storage servers. The large library of clues would be primarily in storage, potentially having the read requests pulled prior to their need to allow them to be stored in the significantly fast ram or cache memory locations. This can still be achieved with the cloud infrastructure, although the management of the data sent and received will have to be secured. This would allow all the relevant data for users to be secured while allowing for semi-quick deployment of that data back to the user if it’s stored regionally.

An ideal system regarding cloud infrastructure would allow for the images to be stored in the cloud and only the relevant images be deployed to a user’s device. This may introduce a slightly longer initial load time, but the storage requirements for the game would be decreased as images could just be maintained in the random-access memory until the application is closed or the memory is released. This would also allow for an identical update deployment system to enforce user’s staying on the most stable patch of the game. A significant portion of the game lacks the requirement to be constantly loading large amounts of information from storage.

## Security

The is significant security in the cloud as everything in Azure is managed by encryption keys to allow for streamlined setup and access. In this case, Azure provides a built-in security service to prevent bad actors from accessing the data. A similar level of encryption is provided with a default secure AWS service permission configuration. If a local storage location is chosen, the server and access need to be restricted. In addition, just as in cloud storage, the data sent to and from the server would also require encryption for user information.