

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 08/01/2021 | Guilherme Pereira | Added an Entity, Team, and Player class. |
| 1.1 | 08/14/2021 | Guilherme Pereira | Added information regarding storage, memory, and security. |

**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 recreate a web-based version of the game: *Draw It or Lose It*. This is based off a similar television game *Win, Lose, or Draw*, where multiple teams compete to interpret an image quickly. The client has requested to unique names for the games, teams, and players added in the game. Each game may have one or more unique teams and each team may have one or more unique players.

## [Design Constraints](#_2et92p0)

Developing a game in a web-based environment has certain constraints such as security and load times. Due to the nature of a web-based game, any guest or player may be able to play the game at any given time without setting up a server room that limits the number of players who can join. Additionally, using basic authentication is risky as anyone can decode the username and password by accessing Authorization under the Request Header. This constraint can be avoided by using a certificate-based authentication in which a digital signature proves that someone is accessing the server and has the correct private key to the associated certificate. One other authentication that could be use is token-based in which providing credentials to log into the site will give the person a uniquely encrypted token which can be used by RESTful APIs. I believe moving forward, token-based authentication would better suit the needs of the web-games security.

Draw It or Lose It is also based on the Java programming language which has a few constraints in terms of speed. Java utilizes the Java Virtual Machine (JVM) which communicates between the language and compiler, essentially double checking that the code runs correct, and those exceptions are filtered through it. While this does increase the stability of the game, it does mean that it will run slower with the extra checks. Additionally, Java is not used much for games such as the Xbox One or PS4, so adding other clients into this game may prove limiting only using Java.

## [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 UML class diagram below consists of 7 classes being: ProgramDriver, SingletonTester, GameService, Entity, Game, Team, and Player. The pattern being demonstrated below is the Singleton Pattern, which emphasis that only one instance of a class is ever present during the execution of the application. The GameService class is the singleton with the Game class being dependent on it, the Team class depends on Game, and the Player class depends on Team.

Due to Game, Team, and Player all sharing similar fields and accessors, an Entity class has been created so this information is not redundant and can be accessed from one class. The Entity class contains private variables, accessors, and an overridden toString() method. This is an example of **polymorphism**, as each class inheriting from Entity has their own way to outputting the toString() method**. Inheritance** is used as Game, Team, and Player all extend from Entity. This means that they share the same fields, accessors, and method as Entity does, but can put their own values instead.

GameService can create 0 to multiple instances of a Game object. A Game class can create 0 to multiple instances of a Team object. Finally, a Team class can create 0 to multiple instances of a Player object. GameService is an example of **encapsulation**, as it provides private fields and accessors that are wrapped up together to be used. The ProgramDriver class is where the code will execute through as it includes a main method. Additionally, ProgramDriver will use the SingletonTester to test whether the code has multiple instances of a class. GameService has a special method called getInstance() which will check if an instance of the class is already at play. If no instance is found (the instance is set to null), then a new one will be created. If the instance is found, however, then the same instance will be returned, and nothing will be created. This is efficient as it saves up on memory by having one instance of a game service running up at a time instead of multiple game services.

****

## [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 claims to have the most security of all other OS and can configure the server. | Linux is very secure and versatile on the server side. There are more possible configurations to the operating platform which could make it more light-weight and faster than other operating platforms. | Better program availability and frequent updates keep servers secure. Corporate servers means a license is needed to operate. | While a server is able to be set up from a mobile device, it is not practical to do so and on iOS devices it requires extra steps that teeter legality. |
| **Client Side** | Some knowledge is required to operate the Mac terminal and use the tools provided. Cost is higher than most OS. | Linux requires the most expertise with operating client-side, as there are many things available only through the terminal. Despite difficulty, cost is less than most OS. | Not much expertise or tools are required. Most of the tools are given to the user. | Provides flexibility to clients by allowing mobility and accessing the site any place with a stable connection. |
| **Development Tools** | Xcode, Tower, FlexiHub, Homebrew, and Atom by GitHub. | Komodo, Code::Blocks, Atom, WebStorm, and Atom by Github. | Intellij, Eclipse, NetBeans, Visual Studio, PyCharm, PHPStorm, and Atom by GitHub | Android Studio, Xcode, Visual Studio, and Intellij. |

## 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**: I would recommend using Windows as the choice operating platform to host servers, as well as the recommended operating platform to play *Draw It or Lose It.* Windows brings a great amount of support when it comes to the tools it provides. According to statistics provided by W3schools from their log-files, 71.7% of all users utilize some Windows operating platform, making it the ideal target to gather more players. While Windows is not as versatile in hosting a server compared to Linux, it does not require constant attention and manual modifications to every component of the infrastructure and files to make these changes. The operating system itself is robust and complete enough to help support a wide range of users while also providing other related applications that help with storing information, such as Microsoft SQL.
2. **Operating Systems Architectures**: The Windows operating system uses Windows NT Server to sustain the game web-application. Windows NT has plenty of powerful features, such as supporting up to 32 processors, connecting to clients without restriction heavy network tasks, supporting up to 256 remote clients, and supports Macintosh files (MiniTool). Windows NT is a Layered Structure, in which each layer performs a specific task such as presenting the user with an interface to interact with the program up until the deepest layer which is the hardware itself. This layered system offers organization through modularity, an easy way to debug certain parts of the layer, and abstraction so that each layer has its own role to accomplish. Windows NT has a User mode and Kernal mode.
3. **Storage Management**: Alongside using Windows as the operating platform, I recommend using Microsoft SQL Server 2019 or MSSQL for short. MSSQL is one of many online databases that can store relational and structural data into containers while also being able to modify and visualize that information for others to see. *Draw It or Lose It* needs to store around nearly 2 gigabytes (GB) of data online so that the web-browser has a means of reaching this information without having to client download the game onto their computer. MSSQL also has an extremely large database capacity at 524,272 terabytes (TB), so if the game were to update to hold more information in the future such as additional images or a score-keeping system for different users, MSSQL would have no issues doing so (Ray).
4. **Memory Management**: *Draw It or Lose It* is a web-based game, therefore making memory management an important factor of providing a suitable experience for clients. While clients do not have to worry about storage management, clients should have a minimum 4 GB of RAM to play the game at a decent speed. For the Windows operating platform, memory is automatically managed by the system instead of manually having to change everything, such as Linux. One way Windows manages memory is by using the paging scheme, which retrieves data from a secondary storage for use in main memory (Fortuna). Another way Windows manages memory is by caching data which is operated by the cache manager. While Windows is running, the system file cache is written onto the disk to be used for later, which essentially improves the speed and access. Instead of loading an entirely new web-page, the system’s cache is able to reference the address to that file through cache and load everything more quickly.
5. **Distributed Systems and Networks**: Distributed network systems allow for computers to communicate with other computers connected over a network. Since the application will be hosted on a server with multiple clients connecting to it, the client and server model is used for *Draw It or Lose It*. Hypertext Transfer Protocol (HTTP) is what the web page will be using to transfer data between the client(s) and server such as the image being rendered or the input being read. Major operating platforms such as Windows, Mac, and Linux utilize HTTP to communicate a client to the server and back. If a client were to disconnect from the server due to an outage or poor Internet connection, this model would work well to not interrupt the gaming session for other players, as the client cannot impact the server’s run-time, but the server can impact all the clients.
6. **Security**: My recommendation for security is using MSSQL to store the user’s login information as well as their profile data such as wins and losses. All players must create a login account to play *Draw It or Lose It* and solve a captcha to ensure that a human is playing and not a bot. Player information and login credentials will be stored on MSSQL where the data will always be encrypted. This data can be decrypted by the client with an encryption key, which is never exposed to SQL database (Jaszymas). As for the program written for this game, I would recommend using Token-based authentication as it is more secure than basic authentication. Token-based authentication provides a unique encrypted key for the user logging into the web page and is supported by RESTful APIs, which is what the program is currently using.

**References**

Fortuna, A. (2019, March 13). *Some thoughts about Windows memory Management: Andrea fortuna*. AndreaFortuna. <https://www.andreafortuna.org/2019/03/13/some-thoughts-about-windows-memory-management/>.

Jaszymas. (2020, October 26). *Security overview - azure sql database & azure sql managed instance*. Security Overview - Azure SQL Database & Azure SQL Managed Instance | Microsoft Docs. https://docs.microsoft.com/en-us/azure/azure-sql/database/security-overview#:~:text=The%20data%20is%20always%20encrypted,or%20in%20Azure%20Key%20Vault.

MiniTool. (2020, December 2). *What is windows nt & main features*. MiniTool. https://www.minitool.com/lib/windows-nt.html.

Ray, M. (2020, March 5). *Maximum capacity specifications for SQL server - SQL Server*. SQL Server | Microsoft Docs. https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-ver15.

W3schools. (2021, July). *OS platform Statistics*. OS Statistics. https://www.w3schools.com/browsers/browsers\_os.asp.

Volico. (2021, April 16). *Linux vs. Microsoft WINDOWS servers: The Official volico blog*. Linux vs. Microsoft Windows Servers. https://www.volico.com/linux-vs-microsoft-windows-servers/.