

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc115077317)

[Table of Contents 2](#_Toc115077318)

[Document Revision History 2](#_Toc115077319)

[Executive Summary 3](#_Toc115077320)

[Requirements 3](#_Toc115077321)

[Design Constraints 3](#_Toc115077322)

[System Architecture View 3](#_Toc115077323)

[Domain Model 3](#_Toc115077324)

[Evaluation 4](#_Toc115077325)

[Recommendations 5](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/21/2024 | Kevin Caballero | Updated executive summary, requirements, and the design constraints for the intended software development. |

## [Executive Summary](#_sbfa50wo7nsh)

“The Gaming Room” wants to develop a web-based game, that runs on multiple platforms within a browser using a server dependent infrastructure. The project would be based on their current game “Draw It or Lose It” which is only available as an Android app at the time. Teams compete to guess what is being drawn as the application renders images from a large library of stock drawings as clues. The game consists of four rounds of play lasting one minute each, with drawings being rendered within a 30-second mark. If the team does not guess the puzzle before the timer expires, every other team will gain the opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## Requirements

* 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

* Currently, the game’s application will contain a multiplayer feature involving multiple teams and players that will interact with one another within the game. Since the application will need to be capable of running on multiple platforms, different operating system interactions must be considered to ensure stability.
* The game will take place in a web-based environment where all the data provided to players will derive from a framework of servers. These hosted servers will have to provide an adequate amount of data that will be used whenever a sudden possible influx of players go online. Otherwise, players could experience connectivity issues once there are too many players active at the same time.
* Every player’s internet connection is another factor to be considered to deter any connection issues certain player bases may experience if they do not have enough download/upload speeds to process the application’s data. If the game requires exceptionally more data processing, players with slower internet speeds will experience performance issues.
* Different browsers that may be used that are unique to certain platforms such as Safari, which is only available on IOS devices, should also be considered to develop the game to be played on multiple browsers as well. This can drastically increase the number of players reached with a versatile application.

## [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 following diagram begins with a GameService class that contains a set of attributes that will develop a single instance of a game’s service running. It uses encapsulation to specify certain attributes that will only pertain to the class itself declared as both private and static attributes and methods. Furthermore, it has a none-to-many relationship beginning with the Game class who also has a none-to-many relationship with a Team class that subsequently has a none-to-many relationship with the Player class. Each use polymorphism with an overwritten toString class that will be used to print their unique attributes from an instance deriving from every class. All three classes, Game, Team, as well as Player, use inheritance to inherit their attributes from an Entity class declaring their name and identifier attributes. Finally, ProgramDriver and SingletonTester are two classes that will be used to test and run the program’s complete functionality. The SingletonTester is used exclusively to test the singleton design pattern that ensures only one instance of a class exists but is capable of being accessed externally. Since our program is broken up into several different classes with unique characteristics there is sufficient portability in place to troubleshoot and make required modifications with minimal issues.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac hosting provides users with plenty of support and resources for hosting on a Mac’s operating system. It is one of the most secure web hosting for developers with a user-friendly interface that is easier to use. Mac hosting is not case-sensitive whenever users are attempting to access a site with its URL. It also allows easy integration with other Apple devices from IOS to other computers using the Mac operating system. It does, however, possess limited customization options for developers. | Linux is an open-source operating system that the public can use, modify, and share. There is versatility in hosting through Linux as it supports both new and older devices which means users can run an application on outdated systems. Also capable of running on most setups without comprising performance. Linux is often targeted less due to its lower popularity, but still offers a highly secure system. Unfortunately, Linux isn’t game or proprietary software friendly since it is not a widely adopted operating system. | Windows is the most popular operating system used by many users. It contains the most compatibility among devices and many websites that rely on Microsoft technologies. Hosting on Windows leans towards developers requiring Microsoft specific technologies like ASP, .NET, and MSSQL. Security poses an issue since Windows hosting services are often vulnerable to ransomware attacks and other threats due to its popularity and accessibility. | Web-based hosting of mobile applications requires external resources and data that would be stored on dedicated servers, shared hosting, VPS hosting, or cloud hosting through different host providers. Mobile applications communicate with these servers using APIs (Application Programming Interfaces) which create a communication link between the application and server. Reliability on security depends on the measures offered by providers through encryption, firewalls, or other security features. Mobile devices do however come with limited performance since processing power is limited due to size constraints along with the possible smaller screen sizes the application will have to accommodate. |
| **Client Side** | Since Mac provides high-quality proprietary software, it comes with a higher price tag with a range of plans to accommodate a client’s needs. Mac hosting offers an isolated environment with infrastructure resources, such as memory, storage, CPU, and bandwidth to host any Mac applications. Due to its developer-friendly interface, it does not require much time to learn and acquire complete control over a client’s virtual environment. Mac also allows hosts to install custom software applications. | There is a steeper learning curve to Linux because it requires more technical knowledge to operate the system. It often uses command-line interfaces to run software installations or other configurations. As a result, it does offer more freedom to modify any application’s files and make changes as desired. Since Linux is open sourced, servers require less set-up costs without requiring extra costs like licensing fees. Capable of using cloud operating systems with major cloud computing providers that work with Linux without sacrificing more hardware for increasing server capacity and performance. | Windows costs more to host with plans offering higher price rates from providers. Servers on Windows may also lack stability since they struggle to execute several applications and processes at once. Which may lead to frequent crashes and other performance issues users may face when accessing the web application. Due to Window’s user-friendly interface, it is easier to learn and manage using the Plesk control panel the system offers. Many users also tend to be more familiar with the OS interface which makes it easier and less time consuming to deploy updates on a website. Windows servers also tend to be more hardware demanding possibly requiring pricier devices to operate the servers. | The amount of time or costs vary depending on the mobile platform developers choose to primarily create the application for. Since there are several platforms including IOS, Android, Windows phone OS, and Symbian. Pricing will vary depending on the host providers along with any portion of fees involved for distributing software on application stores provided by the mobile device OS. All with different capabilities with a few containing similarities with their primary counterparts like Windows and IOS. Mobile devices also possess the best portability, there is far more reach towards users offering far more integration within other apps. |
| **Development Tools** | Mac allows inline scripting through SSI’s (Server-side includes) and PHP by providing a simple way to dynamically add information to web pages within a web application’s HTML code. Java’s virtual machine (JVM) can also run on dynamic websites using platform-dependent Java code hosted on Mac. The open-source SQL database is also preinstalled onto servers allowing database control. Other scripting languages such as Perl, Ruby, and Python are available. Some IDE’s used on Mac devices consist\ of Xcode, Visual Studio Code, Sublime Text, Nova, etc. | IDEs that can be used to build software deploying on Linux include Spacemacs, Vim, Kate, CodeLite, Visual Studio Code, Eclipse, PyCharm, and others. Linux is not limited to any specific programming languages as it can use Python, C++, Java, JavaScript, or Rust to name a few of the most notable languages. Also enabling the use of CGI, Pearl, and PHP for web applications. | Since Windows is the most popular operating system used to date, it offers nearly every single programming language that a developer may use to deploy their web application. Microsoft does offer a variety of web application development tools with ASP.NET, MSSQL (Microsoft SQL Server), C#, Visual Studio, Microsoft Access, and others that can foster a flexible application compatible across other platforms. | Software deploying on mobile devices will usually require external hardware using other systems to develop the application before deploying on any mobile device. Programming languages available to use may consist of JavaScript, Java (mostly used on android apps), Swift (primarily for IOS app development), Python, PHP, and Ruby. Any IDE’s used will pertain to their availability based on the operating system developers choose for deploying the software on mobile devices. |

## 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 Linux as the main operating platform to use to set up a versatile server framework that would expand Draw It or Lose It to other environments. It is open source, which already reduces the set-up costs without requiring extra licensing fees. Also able to accommodate all the operating systems both old and new, while still offering a stable but highly secure system as it is not commonly adopted in comparison to other platforms.
2. **Operating Systems Architectures**: Linux operating system’s architecture consists of the following main components: The kernel, system library, hardware layer, system, and shell utility. The kernel manages system resources and will provide services to the application or processes running on its system. System libraries are either static or dynamic, with dynamic libraries capable of being shared on multiple platforms. They provide pre-written code modules that can be used to efficiently develop applications and optimize their code. The hardware layer revolves around the system’s peripherals such as the CPU, RAM, SSD, or HDD. System calls allow applications to communicate with the kernel to manage system resources, files, and other devices. Lastly, the shell is the command-line interface that Linux operating systems use to run software installations or perform configurations to modify a system’s resources.
3. **Storage Management**: One of the best storage management systems that can be implemented using Linux is with a web-based server management tool such as Webmin. It offers server control options to configure operating system internals, like users, disk quotas, services, and configuration files. Using a RAID (Redundant Array of Inexpensive Disks) would be the suggested approach to store data on multiple hard drives combined within a single device or virtual unit. This method facilitates a secure environment that prevents data loss in case of disk failure, which leads to the duplication of the loss data that would be duplicated onto another disk. There is also increased performance by allowing data to be written onto several devices simultaneously through balanced I/O (input/output) operations.
4. **Memory Management**: Linux uses virtual memory that would act as a logical layer between the Draw It or Lose It application memory requests and its physical memory (RAM). Using virtual memory will allow the program to use more than is physically available within the system. It uses a hardware component called the memory management unit (MMU), which translates virtual memory addresses to physical memory addresses. And a software component, virtual memory manager (VMM), responsible for the allocation and deallocation of virtual memory.
5. **Distributed Systems and Networks**: Using distributed systems and networks would allow an environment where various platforms are able to process all the tasks and components Draw It or Lose It requires to run the application. Multiple servers should be established in multiple regions where players with geographical differences are still able to connect with others despite their server’s location. It would also be advantageous to set up multiple servers in each chosen region to allow a continuous, reliable, and highly accessible network that wouldn’t be impacted by interruptions to the service. If one server requires maintenance or is experiencing a temporary outage, then players can quickly be directed to another neighboring server. In an event where high network traffic occurs as a result from a high player count, a distributed system can also offer stability in which several servers tackle the excessive amount of server requests without overloading a singular system.
6. **Security**: Users will require proper authentication before accessing their information whenever accessing the Draw It or Lose It application. The correct combination of username and password will be required before a user is permitted to access the application. Two-step authentication will be highly recommended as an extra measure to protect a user’s account, requiring an external verification method such as an authenticator app or their phone number where a randomly generated key will be provided for them to enter after a successful login. A password expiration policy can be another measure that requires users to change their password every 6 months to a year. Servers should enforce regular backups to all encrypted data on servers in case of potential losses influenced by security breaches or malfunctions. Firewall protection measures should also be implemented to monitor incoming and outgoing network traffic in case there is the need to block potential threats. Lastly, using an approach such as the principle of least privilege (PoLP), restricts certain users or entities to access only specific data, resources, or applications to accomplish any given task(s). It is especially useful for onsite preventative measures where privileged credentials given to certain roles may be exploited by unauthorized users.