

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/22/2024 | Adrienne Sturgeon | I created the initial software design template. |
| 1.1 | 04/06/2024 | Adrienne Sturgeon | I added the chart showcasing the advantages and disadvantages of each operating system. |
| 1.2 | 04/21/2024 | Adrienne Sturgeon | I completed the software design template by adding recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

The client The Gaming Room currently has an android based app for their game Draw It or Lose It. The game has players work in teams to guess what image is being drawn by a computer from a collection of stock images. The client would like to create a web-based version of this game but need direction in developing this application. They’ve asked for help setting up the game’s environment, and jumpstarting development for the web-based version of their existing game application.

## Requirements

The client needs to translate the android application of their game to a web-based version of this game. The game should allow for at least one team, and should allow for multiple player teams. The teams should have multiple players assigned, and should have unique names. Players should be able to check if a name is currently in use, and only one instance of the game should exist at a time.

## [Design Constraints](#_2et92p0)

* The game should work with at least one player team, but should allow for multiple teams if there enough user players.
* There should be a check to make sure that each team has at least two players on it.
* The team’s name should serve as a unique ID. When new teams are created, a check should run to make sure that no other team currently has that team name.
* Using the singleton design pattern, only one instance of the game should be running at a time. The iterator design pattern can also be used to make sure that the teams and players are unique.
* This should be a web-based game that is able to run on multiple browsers without issue.

## [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)

There is a greater gamingroom environment that all classes are contained within. The main problem runs in the program driver class and uses a singleton tester to ensure that only one instance of the game is running. The entity class is meant to use the object-oriented design concept of portability. The methods and attributes are unique to the Entity object meaning that the class could easily be used in another project. There is also inheritance displayed between the Entity and Game, Team, and Player subclasses. The parent class Entity’s methods are inherited by its child classes. Some classes have private attributes that cannot be changed without using that class’s methods. This makes the data stored more secure and is an example of encapsulation at work. There are also relationships of multiplicity shown in the UML diagram below. A game may have zero to many teams, and each team may have zero to many players. This creates a better idea of how these pieces are meant to interact with one another when the program is running. Finally, polymorphism is shown using the child classes Game, Team, and Player. An object created using the Game class would also be considered an Entity object. This makes Entity polymorphic.

**"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)

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** | MacOS has excellent tools for finely management the operating itself. It has great system administration and the ability to manage servers.  It is contained within the Apple ecosystem, and has compatibility with its own mobile operating system iOS.  Apple is more restrictive with their software, and a mac OS server would only be available on Apple devices. | This is an open source operating system. This means that it is free to use, and can be utilized for provide excellent performance.  Linus servers can be controlled using a GUI, but it is less user-friendly than other operating systems’ GUIs.  Linux systems are built with security in mind, and this extends to the servers. | The high market share of this OS naturally means it experiences high levels of security threats. This will need to be considered when servers are set up using Windows.  This provides excellent support for developers, utilizing a comprehensive GUI for server management.  Windows has been consistent in its use which means it will almost certainly continue to be supported. | Much of the software’s systems will need to be run through a sever due to the hardware limitations of mobile devices.  There is a greater need to provide regular updates to users as the hardware often changes more quickly than desktop computers.  Mobile devices are owned by most people, and there is larger reach for applications running on mobile devices. |
| **Client Side** | Apple is very user-friendly, and information flows easily between devices within Apple’s ecosystem. This means that if an iOS app is developed, it will work well with a Mac OS application.  Apple products are generally more expensive, so there is a price barrier to users.  Compared to open-sourced products like Linux, there is less ability to fully modify the operating system. This may not be appliable to all users however. | Linux is mostly aimed at users with technical experience. It is not as user friendly as operating systems designed for general markets.  For experienced users, Linux is very secure, and offers a flexible approach that provides unbeatable freedom.  Linux does not have as much easily available support for users. There isn’t a centralized support number or char system for users experiencing issues.  Many types of software are not developed with Linux in mind due to it having a lower share of the overall market. | Windows is very user-friendly, and many users will already be familiar with the operating system.  There is extensive support for users experiencing issues with the hardware or operating system itself.  Windows is often expensive if users do not already have a licensing key.  Though support is available, there are many viruses meant to target specifically Windows systems. | Mobile devices are very common, and most users are already familiar with their operations.  There are already many free or low cost entertainment applications available on mobile stores, so users likely already use these devices for entertainment.  Though web-browsers are available on mobile devices, they do not always display properly on mobile phone screens.  Since phones are often very affordable when compared to desktop PCs there is a lower price barrier to users. |
| **Development Tools** | This shares the UNIX file structure with Linux, so developers familiar with the file management of one operating system will find it intuitive in the other.  Homebrew is an intuitive way to manage installations or file management on Mac OS.  Most IDEs are available for development using common languages, but Objective-C will be used frequently when development for Mac OS. | Like Mac OS, a UNIX based file management system makes it familiar to developers experienced with Mac OS.  The open-ended nature of Linux may make it less accessible for new developers, but support for various languages gives more flexibility to developers.  Like MacOS homebrew can be used to simplify file management and installations. | The drive system of storing files might be unfamiliar to developers experienced with Mac OS or Linux.  Many IDEs work natively on windows including the popular Visual Studio.  Windows primarily uses C++, though other languages can be used.  Due to its prevalence, it is very easy to utilize resources like GitHub to further develop code. | There is more limited support for IDEs due to the limited languages being available for mobile development.  Development is primarily done using Java for Android and Swift for iOS. Unless developers are familiar with iOS development they may need to learn Swift for the first time.  There is a wide variety of hardware on the mobile phone market, and applications will need to run with acceptable performance on these 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**:

In order to reduce costs for the Gaming Room’s servers, I would recommend Linux as the operating platform for Draw It or Lose It web application. Linux is free to use and an open source (Oddy, 2022) meaning that the Gaming Room would not need to buy expensive licenses in order to manage servers using Linux. An open-source operating system also means that the OS itself can be modified if need be to suit the needs of the business (Oddy, 2022) offering greater flexibility than other operating platforms.

1. **Operating Systems Architectures**:

There are several “pieces” to Linux that allow applications or servers to run. The hardware interacts directly with the kernel. The kernel is responsible for managing input/output devices, allocating memory, and managing the internal components of a computer’s hardware (Oishi, 2023). There are many ways that commands can be given directly to the operating system. Shells allow the user to enter commands as text using their keyboard directly into the operating system (Oishi, 2023). For the Gaming Room’s server, applications with graphical user interfaces, or GUI’s, can be used to simplify this process and make it easier to understand how information or resources are being used.

1. **Storage Management**:

I would recommend a serverless system for the Draw It or Lose It Program. Rather than the Gaming Room managing physical servers themselves, third-party software can be used to control and access contracted server space (Bashir, 2019). To make images more accessible by users they can be cached closing to the end user. Though this doesn’t eliminate the need for the servers used for hosting the website needed for the application, it can allow for faster transfer of assets and improve the application’s performance (*What is a Content Delivery Network (CDN)?*). There are many other advantages of using a Content Delivery network, but reducing the cost to the Gaming Room is likely very important. By caching commonly used assets, the application can ultimately use less bandwidth and reduce the costs of running the web application itself (*What is a Content Delivery Network (CDN)?*).

1. **Memory Management**: A big challenge for the Gaming Room will be the prevention of fragmentation. Fragmentation occurs when small pieces of memory become unusable as information is stored in blocks of system memory (Silberschatz et al., 2009). Periodically a cleanup can be performed that checks for “lost” memory, and organizes the information stored so that all of the system memory can be used again. This process is called garbage collection, and a large benefit of this is that it is done automatically (Wright, 2022). By preventing fragmentation, the Gaming Room can ensure that no unexpected downtime or performance issues relating to system memory impact the end user’s experience.
2. **Distributed Systems and Networks**:

In order to allow multiple users on various platforms to access the same information, a server client system will need to be implemented. Servers can be added as needed meaning that there is room for the company to grow and scale as Draw It or Lose It becomes more popular (What is Distributed Computing?). There is also better fault tolerance, or risk of failure, using a server client system. If one server fails, there are backups that can continue allowing users to run the program. This means that an individual server experiencing an outage will not cause the entire program to cease operations (What is Distributed Computing?). The main server can contain several versions of the program that are called depending on the user’s operating platform. This makes is easier to possibly implement cross-platform compatibility as all instances of the game are ran on the same server. Using a grip computing network can allow for greater scalability in the future. Rather than each instance running across all processors in the network, it can be limited to a few within a grid. These grids may be loosely connected and can be added to as the business needs dictate (What is Distributed Computing?).

1. **Security**:

Using a client server system will help improve security as users will not have direct access to the game’s files or inner workings. The server acting as an intermediary between the user and the game’s files minimizes potential changes to the program. Linux itself is very secure as it uses passwords to access the root directory making it easier to notice attempts to access the server by malware of malicious users (Memel, 2022). Rather than an unwanted program being able to run in the background automatically, a secure password can be configured to prevent unauthorized access to the servers themselves. If the Gaming Room noticed that an unrecognized attempt was made to access the servers, they can use that information to identify potential security threats. Another large advantage of Linux is that developers from all over the world have free access to the code running the operating system itself. This means that there are many pairs of eyes scanning and identifying potential security issues as they arise (Memel, 2022).

References:

Bashir, F. (2019, July 13). What is Serverless Architecture? What Are Its Pros and Cons?. freeCodeCamp.org. <https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/>

Memel, T. (2022, August 21). 5 reasons why linux is more secure than windows. Medium. <https://medium.com/codex/5-reasons-why-linux-is-more-secure-than-windows-1d036c3d3324>

Oddy, C. (2022, March 30). The Advantages & Disadvantages of Linux for programming. KO2 Recruitment. https://www.ko2.co.uk/advantages-and-disadvantages-of-linux/

Oishi, A. Z. (2023, November 20). Architecture of Linux Operating System. LinuxSimply. <https://linuxsimply.com/linux-basics/introduction/architecture-of-linux-operating-system/>

Silberschatz, A., Galvin, P. B., & Gagne, G. (2009). Operating Systems Concepts (8th ed.). Wiley.

What is a Content Delivery Network (CDN)?. Cloudflare. (n.d.). <https://www.cloudflare.com/learning/cdn/what-is-a-cdn/>

What is Distributed Computing?. aws. (n.d.). https://aws.amazon.com/what-is/distributed-computing/

Wright, J. (2022, March 29). The Basics of Application Memory Management. Medium. <https://medium.com/dvt-engineering/the-basics-of-application-memory-management-19f060c2d0f>