

Draw It or Lose It

# **CS 230 Project Software Design**

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 | 3/19/2023 | Thi Hong Van Nguyen | This version provides an overview of the software's context, including the client's initial business and technical requirements, as well as any design constraints that are related to them. |
| 1.1 | 4/02/2023 | Thi Hong Van Nguyen | This version provides evaluations of each operating platform (Linux, Windows, Mac, and mobile platforms) in terms of building server-side and client side applications, as well as tools used to deploy the game app on each operating platform. |
| 1.2 | 4/16/2023 | Thi Hong Van Nguyen | This version provides an analysis of the specific characteristics and techniques associated with various system architectures, along with recommendations for The Gaming Room. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game, Draw It or Lose It, which can be accessed on multiple platforms. The game will be modeled after the classic 1980s TV game show, Win, Lose or Draw, where teams compete to guess the drawing being presented. The client has specified certain software requirements for the game application, such as the capacity for multiple teams with multiple players, the use of distinct game and team names, and the ability to restrict the game's memory to a single instance.

## Requirements

The requirements of Draw It or Lose It from business perspectives includes developing a web-based version of the game that can serve multiple platforms, allowing for one or more teams to participate in a game, assigning multiple players to each team, ensuring that game and team names are unique, and creating one single instance of the game at any given time when the program is running.

The Gaming Room’s technical requirements are develop a web-based application for the game, implementing a system for rendering images from a library of stock drawings, designing a system for teams to guess the puzzle within a one-minute time limit and developing a system for other teams to offer one guess each to solve the puzzle if time expires.

## [Design Constraints](#_2et92p0)

Since the game would be run in a web-based environment, it requires it to be compatible with multiple web browsers such as Mozilla Firefox, Google Chrome, and Microsoft Edge. The game should be run smoothly without breaking or malfunctioning in different browsers. Another design constraint would be user interface as web browsers can be used in different devices and platforms with different sizes. The development team needs to design and develop an intuitive interface that is responsive and accessible in any devices and screen sizes. The third design constraint is scalability as there might be a large number of users playing the game at the same time. The application should be able to handle a large traffic. Security should also be put into consideration as the application will be accessed by users on the Internet so protecting data from being hacked or breached is extremely critical.

## [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 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.**

The ProgramDriver class contains the entry point for the application, the "main" method. The ProgramDriver class uses the SingletonTester class as a utility to test whether a Singleton design pattern is correctly implemented.

The Entity class serves as a base class for other classes and contains attributes and methods related to an entity's ID and name. The GameService class manages the game-related functions and contains a list of Game objects, as well as the next available IDs for players, teams, and games. The Game class represents a game instance and contains a list of Team objects. The Team class represents a team in a game and contains a list of Player objects. Finally, the Player class represents a player in a team.

In terms of relationships, the GameService class has a composition relationship with the Game class, as it contains a list of Game objects. The Game class, in turn, has a composition relationship with the Team class, and the Team class has a composition relationship with the Player class. The "0...\*" symbol on the connections between classes indicates that a game can have zero or more teams, a team can have zero or more players, and the GameService can manage zero or more games.

In terms of object-oriented programming principles, the diagram demonstrates the principles of inheritance and encapsulation. Inheritance is demonstrated by the use of the Entity class as a base class for the other classes, allowing them to inherit attributes and methods related to an entity's ID and name. This simplifies the code and allows the classes to implement the software requirements for unique IDs and names without duplicating code which helps reduce redundancy. Encapsulation is implemented in the use of access modifiers like private and public which control access to the classes' attributes and methods. This helps prevent external code from directly accessing or modifying their internal state. This enables more reliable and secure software by limiting the scope of what each class can access, making it easier to maintain and debug the system.

## [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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is stable, user friendly and secure. It is not as widely used for server deployments as Linux or Windows, which could limit the availability of support and expertise. It also has higher hardware requirements than Linux, which could increase costs. Mac requires the use of Apple hardware, which can be expensive. Additionally, licensing costs may be incurred for software and tools used in the development process. | Linux servers are known for their stability and uptime, making them an excellent choice for businesses that need their applications to be available 24/7. It can be easily scaled up for large numbers of users. Additionally, Linux provides a high degree of security. However, it may require more technical expertise to set up and maintain compared to Mac or Windows.  There are no licensing costs associated with its use. | Windows is compatible with a wide range of software and hardware. However, it may not be the most secure or stable platform compared to Mac or Linux. It also has higher hardware requirements than Linux, which could increase costs. Windows requires licensing costs for both the operating system and the various software and tools used in the development process. These costs can be significant, especially for larger-scale deployments. | Mobile platforms offer a wide range of tools and software development kits that are well suited for mobile-based gaming applications. They also have a large user base, which makes it easier to reach a broader audience. However, they may not have the same computing power or resources as desktop platforms. |
| **Client Side** | Developing for multiple types of clients on Mac may require using cross-platform development tools or creating separate versions of the software for different platforms. It has user-friendly interface. However, it may require more technical expertise to develop for compared to Windows. Also, developing a web application for macOS can also be costly, as Apple's development tools and frameworks, such as Xcode and Swift, are only available on macOS and require Apple hardware. | Developing for multiple types of clients on Mac may require using cross-platform development tools or creating separate versions of the software for different platforms. Linux is highly customizable. However, it may not have the same level of user-friendliness as Mac or Windows. Developing a web application for Linux can be less costly than Windows, as many of the development tools and frameworks are open source and freely available, such as Python and Node.js. | Developing for multiple types of clients on Windows may require using cross-platform development tools or creating separate versions of the software for different platforms. It may require more technical expertise to develop for compared to Mac. Also, developing a web application for Windows can be costly due to the licensing fees associated with Windows-based software development tools and frameworks, such as Visual Studio and .NET. | Developing for multiple types of clients on mobile devices may require using specialized development tools and programming languages. Costs of acquiring and maintaining the necessary hardware and software for testing the app on different mobile devices and platforms should also be put into consideration. |
| **Development Tools** | Mac has a range of development tools and IDEs available. It also supports a wide range of programming languages, including Objective-C, Swift, C++, or Python. Xcode, a free Apple IDE, includes a wide range of tools for debugging, performance analysis, and testing. | Linux has a range of development tools and IDEs available, including Eclipse and NetBeans. Linux supports a wide range of programming languages, including C++, Java, and Python. These tools are free and do not involve licensing costs. | Windows has a range of development tools and IDEs available, including Visual Studio, which is a popular choice for developing Windows applications. It supports a wide range of programming languages, including C++, C#, and Java. | Developing for mobile devices may require using specialized development tools and programming languages, such as Xcode for iOS development or Android Studio for Android development which could be costly. |

## 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**: Linux is a suitable choice for the operating (server) platform to expand Draw It or Lose It to other computing environments, due to its high level of customization, stability, security, and flexibility. Although it may not be as user-friendly as Mac or Windows, its strong community of developers and users can provide valuable support and resources to The Gaming Room. Additionally, the cost of using Linux would be minimal, making it a cost-effective option.
2. **Operating Systems Architectures**: Linux appears to be the best choice of operating platform for the Gaming Room. It has a modular architecture with three main components: kernel, system libraries, and user space. The kernel manages system resources like CPU, memory, and input/output devices, while system libraries provide essential functions for the operating system and applications. The user space is where user-level applications, such as graphical user interface (GUI) and command-line interface (CLI) applications, run.

Linux is highly scalable and adaptable to various hardware configurations due to its layered architecture, which provides modularity and flexibility. Compared to other operating systems, Linux's modular and layered architecture offers customization and flexibility, which makes it a suitable choice for the Gaming Room's plan to expand Draw It or Lose It to other computing environments.

1. **Storage Management**: All the operating systems have robust storage management capabilities. However, Linux has a significant advantage in terms of storage management. Linux file systems are known for their stability, reliability, and efficiency. The file system provides a hierarchical structure for organizing files and directories, and it supports various types of storage devices, including hard disks, solid-state drives, and network-attached storage.

In addition, Linux offers a variety of storage management systems, including LVM (Logical Volume Manager) and RAID (Redundant Array of Independent Disks), that can be easily configured and optimized for the specific needs of the Gaming Room's storage requirements. Additionally, Linux also has excellent support for cloud storage solutions, such as Amazon S3 and Google Cloud Storage, which can provide scalable and cost-effective storage options for the Gaming Room's needs.

1. **Memory Management**: In terms of memory management, Linux is often considered the best option among the three operating systems. This is because Linux uses a more efficient memory management technique known as demand paging. Demand paging is a technique where pages are only loaded into memory when they are needed, which helps to reduce memory usage and improve overall system performance.
2. **Distributed Systems and Networks**: Linux is the recommended choice for building a distributed system that can enable Draw It or Lose It to communicate across different platforms. This is due to its extensive support for various network protocols and interfaces that can establish and maintain communication between different devices and platforms. Additionally, Linux provides an array of tools and technologies that can effectively manage dependencies and ensure high availability and reliability. For example, the Linux Virtual Server (LVS) is a robust load-balancing solution that can efficiently distribute network traffic across multiple servers, ensuring that the system remains operational, even in the event of a server failure.
3. **Security**: All three operating platforms, Linux, Mac, and Windows, offer strong security features to protect user information. However, Linux has certain advantages that make it a strong candidate for security-conscious applications such as the Draw It or Lose It game. Linux has built-in security features such as SELinux (Security-Enhanced Linux) and AppArmor, which provide mandatory access control and application sandboxing respectively. These features can help prevent unauthorized access and protect sensitive data. Furthermore, Linux has robust password policies and encryption standards which can help protect user information from unauthorized access.