

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 | 5/20/23 | Justin Osman | Executive summary, design constraints, domain model, evaluation, and our recommendations. |
| 2.0 | 5/21/23 | Justin Osman | Evaluating different operating systems |
| 3.0 | 6/17/23 | Justin Osman | Recommendations |

**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 client, The Gaming Room, is asking for our assistance in creating a web-based gaming application called “Draw It or Lose It” that is closely related to the 1980s TV game “Win, Lose, or Draw”. The solution will involve a client-server architecture, that will allow multiple teams to guess what the picture is being drawn that will generate from stock drawings. Unique team and game names will be required, and a database will store the game data. The client will need to provide stock drawings, choose a hosting environment, be involved in the testing and responsible for maintenance after development. In doing so we will be able to provide an engaging game experience for the user.

## Requirements

Business Requirements:

* Create a web-based game where teams guess drawings generated from stock drawings.
* Provide an engaging user experience to attract and retain players, that ensures high user engagement and entertainment.

Technical Requirements:

* Implement a client-server architecture.
* Create unique team and game names to avoid any sort of conflict.
* Create and use a database that will store game data, including games, teams, players, and drawings.
* Get stock drawings for the game.
* Choose a hosting environment for necessary and ensure proper storage, memory and processing power.

## [Design Constraints](#_2et92p0)

Design constraints in a web-based environment:

* The application will need to be able to handle many users and game instances effectively and without any errors.
* Security measures need to be in place to protect user data and unauthorized access.
* Must be able to run on multiple platforms, devices, and web-browsers.
* The application needs one or more teams involved and teams need to have multiple players.
* Must make sure that game and team names are unique by using singleton pattern in their creation.
* Only one instance of the game can exist by using a unique Id for the player, team, and game instance.

Design constraints for application development:

* Application needs to not only run on Android but also Mac, Windows, Linux, and Apple.
* May need to use multiple programming languages to be compatible across all devices and browsers.
* Testing will need to be done to ensure the application runs smoothly across all devices.
* Have systems in place to handle any unexpected errors and backup important data.
* Security measures will need to be in place to protect user data and unauthorized access.

## [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)

In The Gaming Room UML Diagram, the Entity class is a superclass that establishes relationships with the Game, Team, and Player class. This shows inheritance, one of the object-oriented programming principles, which means that these 3 classes share common attributes such as “name” and “id” that are directly inherited from the Entity class. In the UML diagram, the Team class and the Player class have a “has a” type of relationship, which means that they are instances of one class and have references to instances of another class. This is a representation of aggregation, another object-oriented programming principle. The Game class has a reference to the Team class and the GameService class has a reference to the Games class. We can also see that the GameService class holds a reference to the Games class, the Games class holds a reference to the Team class, and the Team class holds a reference to the Player class. We can also see that the ProgramDriver class drives the package and inherits the SingletonTester class while doing so.

**"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** | **Characteristics:** Offer a developer-friendly environment with smooth integration of development tools and compatibility with various web technologies. They are a preferred choice for web applications that involve heavy graphics and multimedia tasks.  **Advantages:**  Offers a smooth development experience and has a strong command line interface. Supports various web technologies and frameworks commonly used in web development.  **Disadvantages:**  Mac computers are generally more expensive than other PC options. Some software applications may be limited with MacOS compatibility. | **Characteristics:**  Linux is an open-source operating system. It is known for its reliability which makes it extremely popular for web hosting. Linux has extensive software support.  **Advantages:**  Linux makes it very easy to customize the operating system to the users’ requirements. High focus on security with regular security updates. Linux is free to use and easily available.  **Disadvantages:**  Linux can be harder to use for some users, may take more time to troubleshoot or learn how to troubleshoot issues. May have some hardware challenges. | **Characteristics:**  Very user friendly and most users are more familiar with Windows than other operating systems. Windows has in depth software support with many development tools easily available.  **Advantages:**  Compatible with the more popular software applications. Well established for hosting web applications that are built with Microsoft technologies.  **Disadvantages:**  Windows security is not as great as other operating system choices and has experienced multiple targeted attacks. The operating system license is costly compared to other options. | **Characteristics:**  Allow users to access applications on the move. The interface is touch-based which engages some users more on web-based applications.  **Advantages:**  Mobile devices have the capability of reaching a larger number of users. They can use cameras, GPS, and notifications to enhance the users overall experience with the application and keep them more engaged.  **Disadvantages:**  Mobile devices have performance limitations because of less processing power and memory. Mobile devices have smaller screens compared to laptops or desktops so designing applications for the smaller screen size can be harder to do. |
| **Client Side** | Mac computers may require I more costly upfront investment. Supporting multiple clients will require expertise on Mac specific framework, if the current team does not have that knowledge additional training will be required. To design, code and release the different clients on Mac platform may require additional time because each client may want different development ideas. | Using Linux can have cost advantages because Linux is open source and does not have licensing fees. More expertise and time will be required to learn and train on Linux since it is not one of the more popular operating system. | Using windows will come with minimal cost because of the Windows licenses. Using Windows will require expertise on certain programming languages and developers may need additional training if not familiar but is easier to learn than other operating systems. An additional amount of time will be required to support multiple types of clients based on their specific needs. | Using mobile devices will require some extra costs to get devices to test on, there may be costs to obtain certain software licenses, and certain fees in the app store to be able to distribute to the users. Will require additional time to serve the needs of both Android and iOS. To support multiple clients on mobile devices will require knowledge in mobile-specific technologies which may require additional training. |
| **Development Tools** | When running languages on Mac we can use Swift or Objective-C for application development although Macs can run all languages. As far as IDE’s we can use Visual Studio which is a cross platform IDE or Xcode which is the official IDE for macOS and iOS development. Other tools that could be used are Interface Builder which is a visual design tool or Homebrew which is a package manager. | On Linux we can use C, C++, Python or Java for programming languages. Relevant IDE’s that can be used are Visual Studio Code which supports Linux development and Eclipse that provides support for development in Linux. Other tools that could be used are Git which is a control system or GCC which is a popularly used compiler for Linux that supports multiple languages . | Relevant programming languages on Windows are C-sharp, C++ or VB.Net all three can be used for Windows development. IDEs that can be used are Visual Studio or Visual Studio Code. Other notable tools that can be used with Windows are Microsoft SQL Server which is relational database manager for Windows or .NET Framework which is a software development framework offered by Microsoft. | Programming languages that can be used on mobile devices are Swift, Objective-C or Java. Swift and Objective-C are used for iOS and Java or Kotlin are the languages used for Android devices. IDEs that can be used are Xcode for iOS app development and Android Studio for Android app development. Other tools that can be used are Flutter which is a cross platform framework developed by Google that allows building apps for both iOS and Android. React Native can also be used which is framework used for building mobile applications using Java. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform:** Using a Linux server platform provides several advantages for The Gaming Room. Linux is renowned for its stability, security, and flexibility. It has a large and active open-source community that continuously improves the operating system. Linux offers extensive development tools, frameworks, and libraries, making it suitable for game development. It also supports cross-platform compatibility, allowing Draw It or Lose It to be expanded to other computing environments, such as Windows, macOS, and mobile platforms.

**Operating Systems Architectures:** Linux supports a wide range of architectures, including x86\_64 (64-bit Intel/AMD), ARM (used in mobile devices and embedded systems), and PowerPC (used in some server environments). By choosing Linux, The Gaming Room can ensure that Draw It or Lose It is compatible with different computing devices and platforms. This flexibility allows the game to reach a broader audience and be deployed on a variety of devices, including computers, smartphones, tablets, and even specialized hardware.

**Storage Management**: Logical Volume Manager (LVM) is a suitable storage management system for Linux servers. LVM provides advanced capabilities for managing storage resources. It allows the creation of logical volumes that span multiple physical disks, providing flexibility in terms of storage capacity and performance. With LVM, The Gaming Room can easily resize volumes, take snapshots for backups, and implement striping for improved I/O performance. This scalability and manageability make LVM a suitable choice for handling the storage requirements of Draw It or Lose It.

**Memory Management**: Linux utilizes memory management techniques provided by the Linux kernel. The kernel is responsible for efficiently managing memory allocation and utilization across the system. When developing Draw It or Lose It, programming languages and libraries can be used to handle memory allocation for storing pictures. Dynamic memory allocation techniques allow efficient storage of pictures outside the default picture folder. By properly managing memory, the game can ensure security, optimize performance, and keep the project assets organized in a designated location on the Linux server.

**Distributed Systems and Networks**: Implementing a client-server architecture is an effective way to enable communication between various platforms for Draw It or Lose It. The central server component would handle game logic and data storage, while clients on different platforms (computers, mobile devices) would connect to the server. Communication between the server and clients can be achieved using standard network protocols such as TCP/IP and HTTP. APIs can be developed to facilitate data exchange and synchronization, ensuring consistent gameplay experience across platforms. To handle potential dependencies and network issues, fault-tolerant mechanisms, data synchronization protocols, and robust error handling strategies should be implemented to ensure smooth operation even in the presence of connectivity outages or other network-related challenges.

**Security:** Linux servers offer strong security capabilities that can protect user information on and between various platforms. Security hardening practices, such as applying regular updates and patches, should be followed to address any known vulnerabilities in the Linux operating system and associated software components. Additional security layers, such as firewalls, intrusion detection systems, and data encryption, can be implemented to safeguard the Linux server and communication channels. User authentication and authorization mechanisms should be in place to ensure that only authorized users can access and modify game data. Compliance with relevant data protection regulations, along with a focus on user privacy, should be a priority throughout the development and deployment of Draw It or Lose It on the Linux server.

By considering these recommendations and implementing them effectively, The Gaming Room can leverage the capabilities of a Linux server to expand Draw It or Lose It to various computing environments while ensuring scalability, security, and a consistent gaming experience for users.