

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 | 09/21/2024 | Danielle Franklin | Recommendations for the Draw It or Lose It game. |

**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 wants to expand their game Draw It or Lose it to multiple platforms including web and mobile. They want to maintain smooth performance and secure data management. Currently Draw It or Lose it is only on the android app store. Designing a distributed web-based application that can run on different platforms will help increase the Gaming Room’s customers and revenue.

## Requirements

These are the business requirements to be considered: the game should be accessible on multiple platforms, it needs to be able to support a large user base, and it should offer consistent gameplay across devices. These are the technical requirements to consider: the software must be web-based, scalable, secure and support cross-platform functionality.

## [Design Constraints](#_2et92p0)

These are the design constraints that need to be considered: The game needs to handle many concurrent users, so the design must ensure server-side scalability. There may be differences in how platforms handle memory, storage, and user interfaces. User data and game progress must be secured across platforms, requiring encryption and secure communication protocols. Each game must have the ability to have one or more teams involved. Each team must be able to have multiple players assigned to it. The game and team names must be unique, so that a user can check if a name is already in use. There needs to be only one instance of the game at a time.

## [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)

This program includes seven classes. The classes Game, Team, and Player are in a generalization relationship, inheriting from a common superclass called Entity, demonstrating multiple inheritance. These three classes and the class GameService are directly associated, and their multiplicity is indicated using curly brackets, showing that these classes may share zero to many objects. Multiple players, each identified by an ID and name, can be added to a team. Similarly, multiple teams, each identified by a name, can be associated with a game. The GameService class maintains a list of games and includes a singleton method called to service. This class also holds references from the Game class. The SingletonTester class is used to check if only one instance of the Game is running at a time, in connection with the ProgramDriver class. The main method, which controls usage, is housed within the ProgramDriver class.

**"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 is Unix-based and provides stability and security. macOS can still function as a web server. The advantages is that stability makes it a viable option for development and integrating with iOS apps. It can offer a consistent environment for developers who work in the Apple Ecosystem. The weaknesses include that macOS is not widely used as a production server platform when compared to Linux. The Licensing cost could be higher, and the hardware required could be more expensive. | Linux is a popular server operating system because of its scalability, security, and performance in web hosting environments. The advantages include that it is open-source, highly customizable, and supports a wide range of sever software. It is stable and secure. The weaknesses include that it requires technical expertise for configuration and maintenance, and that there may be a learning curve for those unfamiliar with it. | Windows servers are common in enterprise environments and offer integrations with other Microsoft products. The advantages include that it offers a familiar environment for developers who have used Windows-based tools and applications. It is ideal if for the game to be integrated with of the Microsoft services. The weaknesses include that the licensing cost may be higher compared to Linux and Windows servers can be considered less stable and efficient compared to Linux when handling heavy loads in web hosting. | Mobile devices are typically not used as servers, but considering mobile client-server architecture the connection to cloud servers is crucial. The advantages include that it is lightweight front-end data processing can be handled locally while heavier processing can be done on cloud-based servers. The weaknesses include that latency and network reliability can affect mobile device performance when communicating with the server. There may be more security concerns with mobile devices. |
| **Client Side** | When considering costs developing applications for macOS requires access to Mac hardware, which can be expensive. When considering time: developing for macOS might take more time due to Apple’s strict guidelines for UI/UX, especially if creating macOS app alongside web applications. When considering expertise: knowledge of macOS, Swift, and Objective-C may be required, however, for web-based clients, knowledge of common front-end technologies like HTML, CSS, and JavaScript might be sufficient. | When considering costs: Linux is open-source and free, which reduces costs, but developing for Linux clients may require more time to address platform-specific issues. When considering time: Time can vary depending on how much customization is needed. When considering expertise: expertise in Linux system administration and development tools, such as Python, JavaScript, or Java, may be required. | When considering costs: Developing for Windows is relatively cost-effective, but there may be some licensing costs. When considering time: Time to market can be quicker due to a large pool of developers familiar with Windows tools. Considering expertise: knowledge in Windows-specific libraries is necessary for developing Windows applications. Web-based clients on Windows don’t pose major challenges and use common web technologies. | When considering costs: developing for both Android and iOS requires investment in tools and possibly hiring developers with mobile app expertise. Considering time: developing responsive web applications can save time compared to app development, but developing a truly native mobile app for multiple platforms can increase development time. Considering Expertise: knowledge in development languages is necessary for apps, while knowledge of cross-platform frameworks like can streamline development. |
| **Development Tools** | Common languages like JavaScript and Python can be used for web-based development, while Objective-C will need to be used for macOS. When considering IDEs: Xcode is the primary IDE for native macOS and iOS development while for Web-based development can be done using IDEs like Visual Studio Code or JetBrains'. Other tools that can be utilized can including testing frameworks like XCTest or web-based testing frameworks. | Commonly used languages like Python, Java, JavaScript and C++ can be used for web-based and server-side applications on Linux. When considering IDEs: common IDEs can include Eclipse, Visual Studio Code, and PyCharm. Other tools to consider are that Linux provides a wide range of tools for server-side development and testing tools like Junit for Java applications. | Programming languages like JavaScript are commonly used for Windows-based development, but Python and Java can also be used for cross-platform development. When considering IDEs: Visual Studio is a common IDE for Windows development and Visual Studio Code can be used for web-based development. When considering other tools: Windows offers a range of development tools and testing frameworks. | When considering programming languages Swift can be used for iOS while Java can be used for Android. For cross-platform development, JavaScript-based frameworks can be used. When considering IDEs: Xcode can be used for iOS while Android Studio can be used for Android. Visual Studio or Visual Studio Code can be used for cross-platform development. When considering other tools testing tools for both iOS and Android are useful for mobile app development and testing. |

## 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 recommended as the primary operating platform for hosting Draw It or Lose It. Linux is recommended because it is an open-source, stable, and highly scalable system that is widely used in web hosting environments. It can offer cost savings when compared to other platforms and it provides better performance for server-side operations because of its lower resource consumption and ability to handle high traffic efficiently. Linux has many development tools and support for many programming languages, and this makes it flexible for deployment in web-based distributed environments.
2. **Operating Systems Architectures**: Linux supports multi-tier architecture. This is beneficial for web-based distributed applications. A standard architecture could include a Linux server using the LAMP stack (Linux, Apache, MySQL, PHP/Python), which will allow for efficient separation of the application layers. The web server, Apache, will handle requests, while the database server, MySQL, manages the game’s data. The application logic, PHP or Python, will be executed on the server side. This architecture supports scalability, as additional can be added easily to handle increased loads as the game grows.
3. **Storage Management**: For storage I recommend using a scalable cloud-based solution. Linux servers integrate seamlessly with cloud services, allowing for storage that can grow with the game’s user base. Cloud storage provides automatic backups, failover systems, and high availability, ensuring that game data such as user profiles, scores, and game states are always accessible.
4. **Memory Management**: Linux uses efficient memory management techniques such as paging and swapping, which allows the operating system to handle large amounts of memory-intensive tasks while keeping system performance stable. Linux server will be able to manage multiple simultaneous game sessions without degrading performance, ensuring smooth gameplays for all users.
5. **Distributed Systems and Networks**: Linux provides support for distributed systems which can support the game’s expansion across various platforms. By using cloud infrastructures, the game can be distributed across multiple servers to ensure minimal downtime and maximum availability. Linux’s built-in support for networking tools and protocols helps with reliable communication between platforms. Cloud-based load balancers can be used to handle traffic efficiently between servers and this can minimize the risk of outages and improve responsiveness for players connecting from different platforms.
6. **Security**: Linux has great security features. It has built-in support for various encryption techniques and advanced firewalls. Linux security protocols can help ensure that user information is protected across platforms. Regular security updates and the ability to customize security configurations can provide protection.