

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 | 10/27/2024 | Sarah Short | First Draft |

**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 would like to develop a web-based version of an existing Android app called Draw It or Lose It. The game requires multiple teams of players. It is essential that the game has only one instance of it running at a time and all team and player names are unique. A solution to these requirements would be using object-oriented design principles and a Singleton pattern to make sure only one instance is running at a time.

## Requirements

The game’s requirements includes allowing multiple teams to play at the same time, unique names of the teams and players, only one instance of the game can exist at a time, and a web-based environment.

## [Design Constraints](#_2et92p0)

1. Scalability: the game must allow multiple teams and players to play at once.
2. State management: The game must allow only one instance of the game to exist at a time.
3. Unique names: All names for teams and players must be unique.
4. Web-based environment: The application requires a web-based environment to ensure cross-platform compatibility.

## [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 UML diagram, the entity class will contain common attributes for game-related entities like id and name. This will allow efficient reusability across different classes, such as Game, Team, and Player. The GameService class will implement the Singleton pattern to make sure only one instance of the game is running at a time. The Game Class will manage multiple teams in a list and will be linked to the Team class to allow multiple teams. The Team Class will manage multiple players in a list and will be linked to the Player class with a 0…\* relationship to allow each team to have 0 or more players. The Player class will store each user’s details. The diagram uses object-oriented principles like inheritance and encapsulation. Inheritance can be seen as an example where the entity class has shared properties with the game, team, and player classes. An example of encapsulation is where each class has private fields to protect the game’s internal state.

**"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 known for its security and reliability, but it is not usually used for large-scale web hosting.  It does have a good environment for developing server-side applications.  Licensing fees can be expensive for MacOS, as well as higher costs for server hosting. | Linux is known for its stability and security and is often used for web servers. It allows for excellent scalability due to its lightweight and efficient resource management. It is open source, so there are no licensing fees. | Windows is often used for web hosting, but it is more expensive than Linux.  Windows Server is less efficient than Linus for high scalability but can be a good choice if integration with other Windows-based technologies is needed. | Mobile devices aren’t typically used for web hosting. They play a bigger role on the client side. |
| **Client Side** | Mac has high-performance hardware but is used less for gaming compared to Windows. Since the game is web-based, that is less of a concern. | Linux is not commonly used for gaming, however since the game is web-based, it should be compatible. | Windows is very popular for gaming. Since this game is web-based, we shouldn’t need to make many specific to Windows requirements though. | Mobile devices are very commonly used for gaming. Typically, an app is developed for this purpose, though games can be played through a web app such as chrome or safari. |
| **Development Tools** | Mac is compatible with web development languages like JavaScript, Python, and Java, and has a variety of programs available for web development like Xcode, Eclipse, and VS Code.  Mac has many tools for web development, but licensing costs can be high. | Linux can support tools such as Eclipse and VS Code and is compatible with languages such as JavaScript, Java, Python and MySQL for backend development. Linux has no licensing fees due to open-source development. | Visual Studio is most often used for Windows development. JavaScript, Java, Python and other web development languages are all supported.  Licensing fees can be high for Windows-based development environments. | Mobile development can be performed by Xcode for iOS development and Android Studio for Android, but there are cross platform tools like React Native available for web-based mobile development. React Native is open source, so there are no licensing fees. |

## 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 recommend Linux for the hosting of Draw It or Lose It. This operating platform is known for its performance, cost, scalability, and security. Linux has the power to handle thousands of concurrent users, which makes it ideal for large-scale deployments and scalability. Linux is also highly customizable which allows for flexibility and optimization in configuration of technical requirements.

Windows or Mac could both be used for development and testing based on user preference. Either is a strong candidate.

1. **Operating Systems Architectures**: Linux operating systems can support multi-threaded and distributed architectures, which can handle web-based games requirements, such as hosting multiple users at once. Multi-threaded architectures allow for efficient processing of multiple concurrent requests. Linux also uses a process-based architecture that has the benefit of strong support for threads and processes to run in parallel. This architecture will allow for large amounts of simultaneous users, which is essential for a web-based multiplayer game. Linux can also integrate with cloud services if needed to provide further scalability and redundancy.
2. **Storage Management**: I recommend MySQL as a storage management system for the game. MySQL is an open-source, commonly used storage management system. It is a relational database that can handle large amounts of data and allow quick retrieval for the game. It supports ACID transactions which can help to ensure data integrity, especially when multiple users are accessing the system at once. As it is open-source, there are no licensing fees but it still has been proven to be a great choice performance wise as a database.
3. **Memory Management**: Linux uses advanced memory management techniques, such as virtual memory and paging. This ensures efficient utilization of system resources. Virtual memory allows Linux to use both physical memory (RAM) and disk space to handle large workloads without running into memory issues. Also, Linux supports caching mechanisms that store frequently accessed data in memory, which will reduce response time for user requests. This is essential for a game like Draw It or Lose It, which requires rapid access to game data
4. **Distributed Systems and Networks**: A microservices architecture will allow multiple services like player and game state management to communicate over the network, allowing redundancy, scalability, and reliability. Each game component, such as player management and game state management, will be deployed as separate services that communicate over the network using RESTful APIs. The network could be supported by cloud-based infrastructure, ensuring scalability and reliability. In case of outages or network issues, failover mechanisms will ensure that services remain available to players. The distributed nature of the system ensures that the game can be accessed from multiple platforms, such as desktops and mobile devices, without interruption.
5. **Security**: SSL/TLS encryption will ensure secure communications between the client and server by encrypting sensitive information such as login credentials. Regular patches and updates are also essential to protect the system from vulnerabilities.