

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 | 11/12/2023 | MK Fowlkes | <Brief description of changes in this revision> |

## [Executive Summary](#_sbfa50wo7nsh)

We are to create an online version of Draw It or Lose It for the Gaming Room company. We're creating a game that features the following: Games feature many teams, each with multiple players, and only permit unique team names so that users can verify whether a name is already in use. Additionally, only one instance of the game is ever allowed to run in memory at a time.

## [Design Constraints](#_2et92p0)

For the game to function properly in any web-based environment, it needs to be built in a web-based language like Python or JavaScript.-By using team and player objects, the game must be designed to support several teams, each with multiple players.-When generating games and teams, the game must use a singleton pattern to ensure that each game is unique by running checks on the names of the teams and games.-The game must create distinct IDs for each instance of the game, team, or player using the iterator pattern in order to prevent more than one instance of the game from existing in memory.

## [Domain Model](#_8h2ehzxfam4o)

Each class that inherits from the Entity superclass—the Game, Team, and Player classes—allows the properties and operations found in the Entity class to be written only once and utilized by all of the other classes. There is an association between the GameService, Game, Team, and Player classes. They employ a zero to many association thus they can use any number of occurrences of the other or none at all. In the process of driving the package, the ProgramDriver class inherits the SingletonTester 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Hosting a web-based software application on a Mac can be a viable option, especially if you value a Unix-based environment, high-quality hardware, and a developer-friendly platform. However, the cost, limited hardware options, and software compatibility issues might make it less practical for certain scenarios. It's important to carefully assess your specific needs and budget before deciding to host your web application on a Mac. | Linux is a powerful and cost-effective choice for hosting web-based software applications. Its strengths in stability, security, scalability, and open-source nature make it a popular choice for web hosting. While it may have a learning curve and some software/hardware compatibility challenges, these can often be outweighed by its flexibility and community support. Carefully consider your specific needs and the technical expertise of your team when deciding whether Linux is the right choice for hosting your web application. | Windows Server is a suitable choice for hosting web-based software applications, especially if your application is built using .NET technologies or if your organization relies on Windows-based infrastructure. It offers a familiar interface, commercial support options, and seamless integration with Microsoft technologies. However, the potential higher licensing costs, resource requirements, and security concerns should be carefully weighed against the advantages when making your decision. Consider your specific application requirements, budget, and the technical expertise of your team in making the choice between Windows and other hosting platforms. | While mobile devices have certain advantages in terms of accessibility, portability, and cost, they are generally not well-suited for hosting web-based software applications, particularly those intended for production use. The limitations in terms of performance, connectivity, power, security, and scalability make them an impractical choice for most web hosting scenarios. Instead, traditional server environments or cloud-based solutions are more appropriate for hosting web applications that require stability, reliability, and performance. Mobile devices are better suited for testing, development, and demonstrations rather than hosting web applications for actual use. |
| **Client Side** | Dedicated development resources, time, and experience are needed to support various client types on macOS. To guarantee a flawless user experience, platform-specific components of Mac development, such as hardware and software requirements, must be taken into account. It is possible to simplify the development process and lessen some of the difficulties involved in serving Mac clients by utilizing cross-platform development tools and best practices. | Software developers must take into account the unique requirements and features of the Linux platform in order to support Linux clients. Although Linux's open-source nature might reduce development costs, its complex ecosystem and requirement for compatibility testing may lengthen the development process. Success requires familiarity with Linux development tools, packaging, and distribution techniques. Linux software development also involves responding to user comments and engaging the community. | In order to support Windows customers in software development, platform-specific characteristics and requests must be considered. The expenses of development might vary depending on the tools and licensing required. High-quality software requires knowledge of Windows programming and user experience design. Cross-platform development tools can make the development process more efficient. Furthermore, for a successful software product on Windows, strong distribution, support, and maintenance methods are required. | Several factors must be considered while developing mobile applications for a variety of clients. Costs can vary depending on the tools used, platform requirements, and whether or not cross-platform development is chosen. Due to platform-specific work and intensive testing, development time may be extended. It is necessary to have knowledge of both iOS and Android coding, as well as design and app store guidelines. Cross-platform development can aid in the process. Prepare for app store guidelines and budget for customer support, feedback, and ongoing updates. |
| **Development Tools** | These are the key frameworks for creating Mac (Cocoa) and iOS (Cocoa Touch) applications. Interface Builder, a component of Xcode, is a visual tool for creating user interfaces for Mac and iOS applications. Homebrew is a macOS package manager that makes it simple to install open-source apps and libraries. The Swift Package Manager is a command-line utility for managing Swift project dependencies. | These are the building blocks for Linux development. C and C++ are used to write many system libraries and applications. Python is frequently used in the development of Linux software, particularly for scripting, web applications, and automation. JavaScript is critical for web-based applications. Node.js is a popular runtime environment for developing server-side JavaScript. VSCode is a cross-platform code editor that supports a wide range of programming languages and is frequently used in Linux development. This IDE is specifically built for GNOME application development. It supports programming languages such as C, Vala, and JavaScript. These tools (Autoconf, Automake, and Libtool) are commonly used on Linux to configure and build software. | C# is a popular language for developing Windows applications, particularly desktop applications, gaming, and enterprise solutions. On Windows, C++ is often used for system-level and performance-critical applications. JavaScript is utilized for web development, and TypeScript is commonly used for building front-end web apps with tools like Angular. Visual Studio is the official integrated development environment (IDE) for Windows development, providing a full collection of tools for developing desktop, web, and cloud applications. VSCode is a cross-platform code editor for Windows that supports a variety of programming languages and is frequently used for web development. ASP.NET is used for developing web applications on Windows, and you can utilize. | Swift is the primary programming language for creating iOS apps. It was created by Apple and is well-known for its security and performance features. Objective-C is an outdated but still useful programming language for iOS. Many existing iOS apps and libraries make use of it. Traditionally, Java has been the dominant programming language for Android development. Java is used to write many Android apps. Kotlin is an official language for Android development that has grown in popularity due to its conciseness and better security features over Java. React Native enables developers to create iOS and Android mobile apps from a unified codebase, mostly utilizing JavaScript and React. |

## Recommendations

LINUX

1. **Operating Platform**: By developing a web-based version of "Draw It or Lose It," you may create a unified and accessible game experience across multiple computer platforms. This strategy enables The Gaming Room to grow its player base, boost engagement, and react to changing technological trends.
2. **Operating Systems Architectures**:

A.)Development of the front end:

HTML (HyperText Markup Language) is in charge of arranging the game's content and web page layout.

CSS (Cascading Style Sheets): Controls the visual presentation, such as color, fonts, and layout.

JavaScript: Allows for game interactivity and dynamic features such as user interactions, game logic, and real-time rendering.

B.) Development of the backend:

Server-Side Programming (Node.js, Python, PHP, and other similar languages): Manages game logic, user authentication, and communication with databases or external services.

Databases (SQL/NoSQL): Used to store user data, game state, scores, and any other essential information for gaming.

1. Infrastructure and hosting:

Web servers (Apache, Nginx): Provide users with web pages and gaming content.

Cloud Services (AWS, Google Cloud, Azure): Provide the game with scalable and dependable hosting, storage, and computational capabilities.

1. **Storage Management**: You might examine alternative storage management methods for a web-based game like "Draw It or Lose It" running on a Linux operating platform based on your individual storage and data requirements. Here are several possibilities:

***RDBMS (Relational Database Management System):***

MySQL: MySQL is a well-known open-source RDBMS that is ideal for online applications. It is well-known for its dependability and performance.

PostgreSQL: Another open-source RDBMS that is very adaptable and ideal for complicated data management is PostgreSQL.

***Databases that do not use SQL:***

MongoDB: MongoDB is a popular NoSQL database that can manage unstructured data and provides data modeling freedom. It's ideal for games with dynamic data structures.

Cassandra is an Apache. Cassandra is a NoSQL database that is designed to handle massive amounts of data across distributed clusters.

1. **Memory Management**

To guarantee optimal use of memory resources, the recommended operating platform for "Draw It or Lose It" includes memory management techniques on both the client and server sides. To maintain optimal speed and avoid memory-related concerns, these approaches include trash collection, caching, session management, database query optimization, and resource loading, among others. Memory management is an important part of web-based game creation in order to provide users with a pleasant gameplay experience.

1. **Distributed Systems and Networks**: To enable "Draw It or Lose It" to communicate between different platforms in a distributed system, you must first create a networked architecture that can handle component dependencies, connectivity, and probable outages.

***Server-Client Architecture:***

Implement a server-client approach in which the game server serves as a communication and game state management hub.

Clients on various platforms (web browsers, mobile devices) communicate with the server via the internet.

***Protocol for Networking:***

Select a network connection protocol, such as WebSocket or HTTP/2, that allows for real-time interactions and data synchronization. Bidirectional communication and low-latency updates are possible with these protocols.

***Component located on the server:***

The game logic, including drawing sessions, game data, user profiles, and game rules, is hosted on the server side.

It talks with the database to retrieve user data and uses a memory cache to speed up data retrieval.

1. **Security** It is critical to preserve user trust and protect sensitive data in "Draw It or Lose It" by ensuring the security of user information on and between numerous platforms. Encrypt data in transit and at rest using encryption techniques. To validate user identities, utilize strong user authentication measures. This involves employing safe password hashing and salting methods. Use multi-factor authentication (MFA) to add an extra layer of security, especially for user accounts with critical data access. Store and handle user data securely, adhering to data protection best practices. Access to user data should be restricted to those who have a need-to-know basis. In conclusion the preferred operating platform, a web-based architecture based on Linux, provides solid security features, but extra security layers must be built to secure user information. By following these best practices and leveraging the platform's security capabilities, you can build a secure environment for "Draw It or Lose It" users, protecting their information on and between platforms.