

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 | <03/20/24> | Josh Davila | Initial creation |
| 2.0 | 4/5/24 | Josh Davila | Added more to the Evaluation section |
| 3.0 | 4/19/24 | Josh Davila | Added more to the Recommendations section |

**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 has reached out to Creative Technology Solutions (CTS) with an opportunity to develop a web-based version of their existing Android game, Draw It or Lose It. Our ideal solution involves using software design templates and patterns to efficiently address the client's requirements. By using a singleton pattern, we make sure only one instance of the game exists at any time, guaranteeing unique game, team, and player identifiers. Our design allows for multiple teams with multiple players each with unique names to avoid conflicts. Our approach focuses on scalability, maintainability, and efficiency, ensuring a seamless gaming experience for users across all platforms.

## Requirements

* Support one or more teams with multiple players assigned to each.
* Create team names.
* Implementing a singleton pattern to manage game instances.

## [Design Constraints](#_2et92p0)

* Developing the game application in a web-based platform has certain design constraints.
* Making sure the game can be scalable and has good performance across different platforms and devices.
* Security measures must be used to protect user information and game integrity.
* Compatibility with various operating systems and browsers needs to be considered to ensure a seamless gaming experience for all users.
* Make sure the program adheres to web standards.
* Use best practices to future-proof the application.

## [Domain Model](#_8h2ehzxfam4o)

The classes include ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player. These classes show object-oriented programming principles such as encapsulation, inheritance, and association. The singleton pattern is used in the GameService class to make sure only one instance of the service exists so it can manage game instances efficiently. Associations between classes show relationships such as one-to-many between Game and Team, and Team and Player, fulfilling the requirement for multiple teams with multiple players each.

**"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** | Offers stability and security for hosting web-based applications, but it may have limited scalability compared to Linux or Windows | Great performance and scalability for hosting web applications, making it a great choice for server-side deployment | Offers ease of use and compatibility but may have higher licensing costs and may be less efficient for web hosting compared to Linux | Limited capabilities for hosting web applications directly but can interact with server-side components effectively |
| **Server Deployment Method** | Offers server-based deployment methods such as macOS Server. | Various server deployment methods including Apache, Nginx, and others. | Server deployment methods suitable for hosting web applications. | Mobile devices do not offer server-based deployment methods for hosting web applications. |
| **Potential Licensing Costs** | Potential licensing costs for macOS Server may vary based on the version and usage but can range from $19.99 to $299.99 per license. | Linux distributions are generally open-source and free to use but may incur support costs depending on the distribution and vendor. | Windows Server licenses vary based on the edition, ranging from $501 for Windows Server Standard to $6158 for Windows Server Datacenter. | Mobile devices do not have server operating system licensing costs for hosting web applications. |
| **Client Side** | Development for Mac clients requires consideration of software tools and frameworks compatible with macOS, potentially increasing development time and cost. | Like Mac, development for Linux clients involves selecting tools and frameworks compatible with Linux distributions, potentially requiring additional expertise. | Windows clients benefit from a wide range of development tools and frameworks, reducing development time and cost. | Developing for mobile devices requires expertise in mobile development frameworks and consideration of platform-specific requirements, potentially increasing development time and cost. |
| **Compatibility with Web Browser Platforms and Mobile Devices** | Applications developed for Mac are compatible with major web browsers like Safari, Chrome, and Firefox, but may require additional testing for cross-browser compatibility. | Applications developed for Linux are compatible with major web browsers like Chrome, Firefox, and Opera, ensuring broad accessibility. | Applications developed for Windows are compatible with popular web browsers like Chrome, Firefox, and Edge, providing widespread access. | Mobile applications need to be developed using responsive design principles and tested across various devices and browsers to ensure compatibility. |
| **Development Tools** | Tools for Mac development include Xcode and Swift for native development, or web development tools like Visual Studio Code for web-based applications. | Development tools for Linux include various programming languages like Python, Java, or C++, along with IDEs like Visual Studio Code or JetBrains IntelliJ IDEA. | Windows development can utilize tools like Visual Studio and C# for native applications or web development tools like Visual Studio Code. | Development for mobile devices requires knowledge of platforms like iOS (Swift or Objective-C) or Android (Java or Kotlin), along with IDEs like Xcode or Android Studio. |
| **Impact of the Dev Team** | Development for Mac may require specialized expertise in macOS development, potentially necessitating a dedicated development team. | Development for Linux may require expertise in Linux-specific development tools and frameworks, potentially requiring a specialized team. | Windows development benefits from a wide range of available tools and frameworks, enabling efficient development with smaller or less specialized teams. | Mobile development may require multiple teams specialized in iOS and Android development, increasing development complexity and cost. |
| **Licensing Costs for Development Tools** | Xcode, the primary development tool for macOS, is available for free. Swift, the programming language used for macOS development, is also open source | Development tools for Linux, such as Visual Studio Code and many programming languages, are open-source and free to use. | Visual Studio Code, a popular development tool for Windows, is free to use. Licensing costs may apply for other Microsoft development tools. | IDEs like Xcode and Android Studio are available for free, but licensing costs may apply for certain third-party development tools and libraries. |

## 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 for hosting the Draw It or Lose It game due to its robust performance, scalability, and cost-effectiveness. Linux provides a stable and reliable environment for server applications, making sure there is minimal downtime and efficient resource utilization.
2. **Operating Systems Architectures**: Linux utilizes a kernel architecture, providing efficient resource management and performance for server-side applications. The kernel design allows for direct communication between hardware and software components, this allows faster execution of system calls and improved overall performance.
3. **Storage Management**: MongoDB is recommended for storage management, offering scalability and flexibility for handling game data. MongoDB's document-oriented storage model allows for easy management and retrieval of game-related data, such as player profiles, game state, and leaderboard information.
4. **Memory Management**: Linux allows virtual memory management techniques, which then uses more efficient utilization of system resources for running the game software. By using virtual memory, Linux can dynamically allocate and deallocate memory as needed, optimizing performance, and ensuring smooth gameplay experiences for players.
5. **Distributed Systems and Networks**: Draw It or Lose It can communicate between various platforms through RESTful APIs, enabling seamless integration and interoperability. Using RESTful APIs also allows for easy scalability and extensibility, enabling The Gaming Room to add new features and support additional platforms in the future.
6. **Security**: Linux offers robust security features, including access controls and encryption to protect user information and game integrity across platforms. Using HTTPS and authentication enhances security for the game application. Using authentication mechanisms, such as OAuth or JSON Web Tokens (JWT), helps verify the identity of users and prevent unauthorized access to game resources.