

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 05/24/2024 | Faris Malik | Added info for design project |
| 1.1 | 05/25/2024 | Faris Malik | Added evaluation and recommendation sections |
| 1.2 | 06/05/2024 | Faris Malik | Added more recommendations and OS compatibility info |

**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 current project aims to transition The Gaming Room’s "Draw It or Lose It" from an Android-only app to a web-based game that can serve multiple platforms. The challenge lies in designing a scalable, efficient, and secure software solution that provides to a diverse user base across various devices. Our solution leverages modern web technologies to create a responsive, real-time gaming experience that maintains the essence of the original application while introducing the flexibility of cross-platform compatibility.

## [Design Constraints](#_2et92p0)

The transition of The Gaming Room's "Draw It or Lose It" from an Android-only app to a web-based game accessible on multiple platforms introduces several design constraints that must be carefully considered:

* **Cross-platform Compatibility**: To ensure the game is accessible from operating systems such as MacOS, Linux, Windows, and others, we will develop it as a web application. This eliminates the need to select a language compatible across all platforms, as the application will communicate using a REST API over HTTP. This approach requires careful consideration of differences in browser behavior and feature support.
* **User Interface**: The existing Android application sets a precedent for the user experience. We have the option to either replicate this interface for web users, ensuring familiarity for current users, or to redesign the interface for a fresh experience. In either case, the interface must be responsive and adaptable to various screen sizes and input methods.
* **Multi-Team and Multi-Player Support:** The game must support the participation of multiple teams with multiple players each. This requires a robust client-server architecture capable of managing concurrent connections and synchronizing game state across all clients in real-time. The server must efficiently handle multiple requests without latency or data loss.
* **Unique Identifiers**: Game and team names must be unique, and only one game instance can exist in memory at a time. This necessitates a system for generating and managing unique identifiers for games, teams, and players. Additionally, memory management strategies must be devised to handle the single-instance constraint, especially as the application scales across multiple platforms.
* **Image Compatibility and Copyrights:** The game relies on a library of images as part of the gameplay. We must ensure that images from the Android platform are compatible with web standards and that their quality is maintained across all platforms. Any new images added will require appropriate licensing or copyright permissions. Furthermore, we must implement efficient image rendering and caching to maintain performance across platforms.

## [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)

The UML diagram, or Unified Modeling Language diagram, serves as a blueprint for the structure of the "Draw It or Lose It" game system. It visually represents the relationships and hierarchies between various components within the software. At the heart of the diagram is the Entity class, which acts as a superclass for the Game, Team, and Player classes. This inheritance relationship is denoted by arrows pointing from the Game, Team, and Player classes to the Entity class, indicating that they inherit common attributes and behaviors from Entity. The Entity class encapsulates the shared properties, such as a unique identifier (id) and a name (name), thereby promoting code reusability and reducing redundancy. The ProgramDriver class, situated prominently, is connected to the SingletonTester. This relationship signifies that the ProgramDriver will utilize the SingletonTester to verify the singleton pattern implementation, ensuring that only one instance of the game can exist in memory at any given time. This is a critical requirement as it prevents multiple instances of the game from being created, which could lead to inconsistent game states and data conflicts. The GameService class is depicted as the core engine of the game, containing complex methods that define the game's functionality. It is responsible for orchestrating the game logic, managing game sessions, and enforcing the rules. The GameService class uses the singleton pattern to ensure it is the sole instance managing the game state.

Each class designed to represent the game's entities—Game, Team, and Player—is essential for fulfilling the requirement of having unique identifiers for games, teams, and players. The associations between these classes are indicated by the connecting lines, with multiplicities specified at each end. For instance, the line between the GameService and Game classes with the multiplicity "0..\*" indicates that the GameService can be associated with zero or more instances of the Game class. The UML diagram will guide the development process, providing a clear map of the object-oriented structure required to build the game. While the diagram serves as an initial design, it is flexible and may be adjusted as development progresses and new requirements or constraints are identified. This iterative approach ensures that the final product will meet The Gaming Room's needs while adhering to the principles of software engineering.

**"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** | Mac OS X is known for its stability, robust performance, and minimal downtime. Offers excellent security features and is less prone to cyber attacks; integrates well with other Apple products and services. Hardware and software for Mac can be more expensive; less market share compared to Linux and Windows servers might mean fewer community resources. | Linux is open-source, widely used for servers, and supports a vast array of software and programming languages. High customizability, strong community support, and cost-effectiveness (many distributions are free). May require more technical expertise to manage and configure compared to Windows; different distributions can lead to fragmented experiences. | Windows Server is a widely used operating system that integrates well with other Microsoft products. User-friendly interface, comprehensive support from Microsoft, and widespread familiarity among developers and administrators. User-friendly interface, comprehensive support from Microsoft, and widespread familiarity among developers and administrators. | Mobile devices typically run on iOS or Android and are not traditional hosts for web applications. Can be used for testing mobile responsiveness and features; useful for demonstrations or on-the-go administration. Not suitable for hosting due to limited resources, less stability, and scalability issues. |
| **Client Side** | Ensuring compatibility with Safari and other browsers on Mac; design must be responsive to cater to various Mac devices. Development tools for Mac can be expensive, but the platform offers a high-quality user base. Testing and optimization may take additional time due to the need to accommodate Mac-specific features. Developers should be familiar with macOS and its nuances, especially for Safari browser compatibility. | Must ensure the game runs smoothly on various Linux distributions and browsers. Generally lower due to open-source tools and software availability. May require additional time for testing across different distributions. Developers need to be comfortable with the Linux environment and its various distributions. | Compatibility with multiple versions of Windows and browsers like Edge and Chrome. Development tools can vary, with many free options available, but licensing for software may incur costs. Testing on different versions of Windows is essential and can be time-consuming.  Developers need to be familiar with the Windows ecosystem and its user interface guidelines. | Must ensure a responsive design and touch-friendly interface; testing across different devices and screen sizes. Development and testing tools for mobile platforms can be costly. Optimization for mobile devices requires significant testing and tweaking for performance.  Expertise: Requires knowledge of mobile platforms, touch interfaces, and performance optimization. |
| **Development Tools** | Swift, Objective-C, Xcode for native applications; JavaScript, HTML5, CSS for web applications; cross-platform tools like React Native or Flutter. | Commonly used languages include Python, Java, PHP, and JavaScript; tools like Eclipse, Sublime Text, and Git; containerization with Docker; and orchestration with Kubernetes. | C#, .NET framework, Visual Studio; PowerShell for scripting; JavaScript, HTML5, CSS for web applications; cross-platform tools like Xamarin. | Swift for iOS, Java or Kotlin for Android; integrated development environments like Android Studio and Xcode; cross-platform frameworks like React Native, Flutter, and Apache Cordova. |

## Recommendations

1. **Operating Platform**: For The Gaming Room to expand "Draw It or Lose It" to other computing environments, I recommend using the Windows operating system for both server-side and client-side operations. Windows offers a wide range of development tools and is familiar to many users and developers, which can streamline both development and deployment processes.
2. **Operating Systems Architectures**: Windows Server is a robust platform for hosting web applications and offers a comprehensive ecosystem for development and deployment. It supports .NET and .NET Core for building web services and applications, which can be advantageous for "Draw It or Lose It" considering its performance and scalability features. The architecture can be configured for a multi-tier setup, separating the web server, application, and database layers for better manageability and security.
3. **Storage Management**: On Windows, SQL Server is a powerful and widely-used database management system that can be used for "Draw It or Lose It". It offers advanced features for data analysis, transaction processing, and business intelligence. For file storage, Windows Server provides integrated services like File and Storage Services, which can be scaled out with additional hardware if needed.
4. **Memory Management**: Windows has built-in memory management features that automatically handle memory allocation and garbage collection for applications. For "Draw It or Lose It", this means the game can run efficiently with optimized resource usage. Windows also offers tools for developers to monitor and fine-tune the performance of their applications.
5. **Distributed Systems and Networks**: "Draw It or Lose It" can leverage Windows Network Load Balancing (NLB) services to distribute client requests across multiple servers, ensuring reliability and availability. For cross-platform communication, Windows supports a range of protocols and services, including WebSockets for real-time interaction. The distributed architecture can be designed using Windows Communication Foundation (WCF) for robust networked applications.
6. **Security**: Security on Windows can be managed using a combination of Windows Defender, firewall settings, and regular updates to protect against vulnerabilities. For "Draw It or Lose It", implementing application-level security measures such as HTTPS, secure authentication, and authorization protocols is critical. Windows also provides Active Directory services for user management and access control, ensuring that only authorized users can access sensitive game functions and data.