

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 | 09/17/25  10/4/25  10/15/25 | James Ford  James Ford  James Ford | Added Executive Summary, requirements, constraints  Added development requirements for platforms  Added Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The staff of the Gaming Room client have requested that we propose how best to set up an environment that facilitates the development of a web-based version of their gaming app. The environment must have the capability of hosting multiple games, each with one or more teams consisting of multiple players. For that reason, it is important that only one instance of a game exists in memory at one time. To achieve this, a singleton pattern should be implemented alongside the creation of unique identifiers for each instance of a game, team, or player.

## Requirements

1. Design a web-based game platform capable of hosting multiple games.
2. Each game will host multiple teams.
3. Each team will have multiple players.
4. Game, team, and player names must have unique identifiers.
5. Only a single instance of each game may exist at one time in memory.

## [Design Constraints](#_2et92p0)

1. If the games are intended to be played in the browser, the browser is not a dedicated game engine and comes with inherent limitations concerning networking, latency, and individual browser performance, especially since this game will operate in real time.
2. Device compatibility will need to be considered. Desktops, tablets, and smartphones each have different screen sizes, operating systems, processing power, and input methods (mouse, keyboard, touchscreen).
3. Web-based games often require dedicated servers. Does the client have an existing physical server, or will the budget demand cloud-based services?
4. Web technologies chosen for development may impose additional constraints or limits on development.
5. Gaming Room staff may have little to no experience with the web technologies and programming languages necessary to maintain the platform.

## [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)

All classes except program driver and the singleton tester are inherited from the Entity class. Each class practices common principles of object-oriented programming by leveraging the portability of the class object. The classes also utilize the inheritance of methods and variables from a super class, encapsulation of variables and methods by making them private and accessible only through the class methods, and polymorphism with overridden super class methods. The program driver class calls the test singleton method within the singleton tester class which controls object creation and ensures only one instance exists in memory at a time.

"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** | Unix-based environment with robust networking and security features.  Pre-installed with a wide range of development tools and utilities.  More expensive than other OS servers.  Better suited for smaller-scale developments. | Open-sourced.  Highly stable with robust security.  Flexible, customizable, Scalable.  Robust networking features.  Linux servers outperform those of other OS.  Compatibility challenges between other devices. | Seamless compatibility with Microsoft products, servers.  Strong tech support.  Robust server management.  Offers advanced enterprise features like data management and virtualization.  World’s #1 multi-platform game publisher. | Broad market access.  Large user base.  Portability can lead to increased engagement.  Framework and cross-platform flexibility.  Internet Dependent.  Managing servers, databases, and scaling requires time and expertise. |
| **Client Side** | Hardware is expensive.  Licensing costs.  Cost of devs with expertise in Mac OS, Apple frameworks, and third-party tools.  Frequent OS updates require constant cross-platform testing. | Cost to integrate, support, maintain, and test multi-platform devices.  Cost of devs with expertise in Linux, cross-platform development, and integration. Diverse clients with specific needs increase the time needed for development. | High licensing fees.  Security vulnerabilities.  Resource intensive Graphical User interface.  Less scalable than Linux for some large-scale, high traffic deployments. | System resource and screen size constraints.  Mobile device variety and complexity can increase the time needed for development.  Internet connectivity can hinder functionality. |
| **Development Tools** | HTML, CSS, Java, Python, PHP, JavaScript (Node.JS), Ruby, and Go.  IDE tools include Xcode, VS Code, Sublime Text, Git.  Web Servers: Apache, Nginx. | HTML, CSS, Java, Python, PHP, Go, JavaScript, React and Angular.  IDE: Eclipse and NetBeans, VS Code, Vim, Sublime Text, Git.  Web Servers:  Apache, Nginx. | HTML, CSS, Java, JavaScript, C++, C#, PowerShell, Python, TypeScript, VS Code, Sublime Text, Git.  Web Servers:  Windows Server, Apache, Nginx. | HTML, CSS, Java, JavaScript, React, Kotlin, Swift, Dart (with flutter), C++, C#, Objective-C, Python, VS Code, Xcode, Sublime Text, Git.  Web Servers:  Apache, Nginx. |

## Recommendations

1. ***Operating Platform****:*

Draw It or Lose It should be developed using the cross-platform web browsers Chrome, Edge, and Safari. Developing a browser-based game is a low-cost development option that allows for broad accessibility through modern browsers, requires no download for users, centralizes updates and maintenance, and retains full control over all revenue earned (Jamardo, 2018). Modern web technologies like Web Assembly and WebGL now allow for high-performance, visually rich games that run at near native speeds, and the use of standard web technologies lowers the cost of hiring developers with expertise.

1. ***Operating Systems Architectures****:*

The client-side, the browser, handles the user interface, game loop, and rendering. HTML and CSS provide the structural and visual components, including animations and transitions, while a language such as JavaScript processes the game logic, user input, and state management. Finally, there is a rendering layer such as the <canvas> element that leverages technologies like WebGL and WebGPU to provide dynamic 2D and 3D shapes, texts, and visuals.

1. ***Storage Management****:*

A cloud-based server-side storage system is crucial to keep costs low through pricing based on usage, increase operational agility, prevent cheating, store game states, and allow updates on progress across multiple devices (Oracle.com). Cloud-based storage also often comes with collaboration tools for teams, built-in security, and is easily scaled up at need.

1. ***Memory Management****:*

The browser allocates a minimal block of its own memory for the game to process video and control user input while the heavy processing is performed on cloud servers. Browser memory holds uncompressed game data such as code, textures, and other assets. Browsers can also cache assets using services like the IndexedDB API, which is a JavaScript application programming interface that browsers provide for managing a larger database of structured objects.

1. ***Distributed Systems and Networks****:*

Instead of traditional HTTP requests, this game would make use of web sockets which create a persistently open channel which allows the client-server sides to communicate with each other in real time. A single server acts as the source for the game world, sending and receiving input from players, and updating the game's state. To scale up to a larger number of players, the system can be distributed across several servers, with different servers handling different sessions or functionality in the game, like player authentication or updating game state.

1. **Security**:

Security begins with browser features such as HTTP enforcement, which ensures that a website connection is encrypted, and the standard browser sandboxing feature that isolates browser processes and limits access to system resources.  The client can enable firewalls, require unique player identification together with strong passwords and, additionally, teams can take advantage of cloud security technologies such as data encryption and Two Factor Authentication.

**REFERENCES**

Jamardo, I. (2014, July 18). *Developer Nation Community*. Developernation.net. <https://www.developernation.net/blog/benefits-challenges-html5-app-game-cross-platform-development/>

Oracle. (2022, March 3). *Top 10 Benefits of Cloud Computing*. Oracle.com. <https://www.oracle.com/cloud/what-is-cloud-computing/top-10-benefits-cloud-computing/>