

Draw It or lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_gjdgxs)

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/27/2023 | Evann Hopkins | Added information related to the software design |

**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](#_35nkun2)

In response to The Gaming Room's request to expand their gaming app, Draw It or Lose It, to multiple platforms using various software patterns in a distributed environment, we have conducted a thorough evaluation of the characteristics, advantages, and weaknesses of each platform. This evaluation aims to assist in the decision-making process by highlighting the benefits and utilities of each operating platform in a distributed web-based application.

## Requirements

The client seeks to create a web-based application that can scale up to accommodate thousands of players. The application must be accessible on traditional desktop operating systems (Linux, Mac, and Windows) and mobile platforms (Android and iOS). Additionally, it should offer a responsive HTML interface capable of communicating with the server-side application.

## [Design Constraints](#_1ksv4uv)

Developing the game application in a web-based distributed environment presents several design constraints. These include ensuring compatibility across different platforms and devices, efficient communication between components in a distributed system, robust storage and memory management, and the utmost importance of security to protect user information.

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

The game system will utilize the Unified Modeling Language (UML) diagram to provide a visual representation of its design. The UML diagram shows the relationships between different classes, such as game, team, and player. These classes inherit attributes from a superclass, as indicated by the arrows.

Within the diagram, we can observe the classes, variables, and methods that will be used during the development process. The programDriver Class, located in the top left corner, is connected to the singletonTester, indicating that the programDriver will use the SingletonTester to test the code. This test ensures the restriction of having only one instance of the game existing in memory, fulfilling one of the program's requirements.

The Gameservice class is responsible for holding all the complex methods that form the core of the game and its functionality. The design of separate classes for game, teams, and players ensures uniqueness for each element, meeting the program's requirements.

The associations between classes are represented by the connecting lines, indicating that they are all interconnected. The numbers between the lines signify the number of associations within each class. For example, the GameService can have zero or more games associated with the Game class, and the same principle applies to the other classes.

Throughout the development process, this UML diagram will guide the creation of the final product. Adjustments may be made along the way to ensure the game system is implemented effectively.

**"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](#_z337ya)

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 (macOS) is a secure and stable platform for hosting web-based software applications. It offers a developer-friendly environment with Unix-based architecture, ensuring good performance and multi-platform testing. However, higher licensing costs and limited market share should be considered, and scalability may be a concern for high-traffic applications. Clients should weigh these factors based on their specific needs and target audience. | Linux is a cost-effective, stable, and secure choice for hosting web-based software applications. It offers scalability, diverse web server support, and benefits from a large community. However, it may have a learning curve for newcomers and potential driver and fragmentation challenges. Overall, Linux is a solid option for web application hosting. | Windows is user-friendly and compatible, making it familiar to many users. It integrates well with Microsoft technologies, but licensing costs and security concerns should be considered. While it can scale, it may not match Linux's scalability for high-traffic applications. Windows is a viable option, especially for Microsoft-centric environments. | Mobile devices offer accessibility and touch interactions, making them appealing for web-based applications. However, limited resources, connectivity, and screen size constraints should be considered. Despite challenges, the growing user base and convenience make mobile devices valuable for reaching a broader audience, but application complexity and target users should be weighed. |
| **Client Side** | Supporting multiple types of clients on macOS involves considerations for development cost, time, expertise, user experience, hardware compatibility, updates, maintenance, security, and user support. Developers need expertise in macOS-specific tools and cross-platform skills to ensure a smooth user experience and compatibility across different Mac devices. Regular updates and bug fixes are essential for maintaining optimal functionality. | Supporting multiple Linux clients involves considerations for cost (lower due to open-source nature), development time (addressing distribution variations), expertise in Linux tools, cross-platform skills, user experience adaptation, hardware compatibility, updates, security, and technical support for Linux users. | Supporting multiple Windows clients involves considerations for cost (moderate with potential licensing fees), development time (addressing version variations), expertise in Windows tools, cross-platform skills, user experience design, hardware compatibility, updates, security, and technical support for Windows users. | Supporting multiple mobile clients involves considerations for cost (higher development and testing costs), time (addressing platform differences), expertise in mobile frameworks, responsive design, hardware compatibility, updates, security, and technical support for mobile users. Developers need skills in Android and iOS development to ensure a smooth user experience across different devices and platforms. |
| **Development Tools** | Relevant languages and tools for deploying web-based software on Mac include HTML, CSS, JavaScript, PHP, Ruby on Rails (or alternatives), and Python/Node.js/Java. IDEs and text editors like Visual Studio Code, Atom, and Sublime Text are used. Git for version control, Homebrew/npm for package management, and Apache/Nginx as web servers. Safari/Chrome DevTools for browser debugging. | Relevant languages and tools for deploying web-based software on Linux include HTML, CSS, JavaScript, PHP, Ruby on Rails (or alternatives), and Python/Node.js/Java. IDEs and text editors like Visual Studio Code, Atom, and Sublime Text are used. Git for version control, package managers like apt/yum, and Apache/Nginx as web servers. Firefox/Chrome DevTools for browser debugging. | Relevant languages and tools for deploying web-based software on Windows include HTML, CSS, JavaScript, PHP/ASP.NET, and Python/Node.js/Java. IDEs like Visual Studio and Visual Studio Code are commonly used. Git for version control, package managers like Chocolatey/NuGet, and IIS as the web server. Edge DevTools/Chrome DevTools for browser debugging. | Relevant languages and tools for deploying web-based software on mobile devices include HTML, CSS, JavaScript, PHP/Ruby on Rails, and Python/Node.js/Java. IDEs like Android Studio (for Android) and Xcode (for iOS), along with Visual Studio Code, are used. Git for version control, npm for package management, and remote web servers (Apache/Nginx) are common. Browser developer tools like Safari Web Inspector (iOS) and Chrome DevTools (Android) aid in debugging. |

## 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 a suitable choice for the operating platform due to its robust server-based deployment capabilities, cost-effectiveness, and openness as an open-source operating system. It provides excellent support for hosting web applications.
2. **Operating Systems Architectures**: The chosen operating platform, Linux, follows a stable and scalable architecture, making it well-suited for hosting a web-based application like Draw It or Lose It.
3. **Storage Management**: For storage management, we recommend leveraging a reliable and scalable database system, such as MySQL or PostgreSQL, which are compatible with the chosen operating platform.
4. **Memory Management**: Linux's memory management techniques, combined with proper application design practices, will ensure efficient memory utilization for the Draw It or Lose It software.
5. **Distributed Systems and Networks**: To enable communication between various platforms, we propose adopting RESTful APIs and WebSocket protocols to facilitate real-time interactions. We must consider the dependencies between components and plan for network outages gracefully.
6. **Security**: Given the client's priority on security, Linux offers robust security features. However, we also recommend implementing secure coding practices, data encryption, and user authentication mechanisms to enhance the overall security of the application.