

**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 | 12/10/2023 | Rotina Staples | Updated Executive Summary for enhanced clarity and conciseness.  - Refined Requirements section to articulate business and technical needs more clearly.  - Analyzed and explained Design Constraints, and highlighted their impact on application development in a web-based environment.  - Incorporated feedback for a more comprehensive and reader-friendly Document Revision History 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 software design problem involves developing a web-based distributed game application named "Draw It or Lose It" for The Gaming Room. The solution focuses on implementing a robust system architecture, incorporating the Entity class for common attributes, and ensuring unique names for games, teams, and players. The proposed design addresses the client's requirements for a scalable, efficient, and secure gaming platform.

## Requirements

This section outlines the client's business and technical requirements, supporting the outline of design constraints below.

## [Design Constraints](#_2et92p0)

Developing the game application in a web-based distributed environment imposes certain constraints. The primary constraint is the reliance on web technologies, which influences factors such as platform independence and client-server communication. The choice of web-based architecture introduces challenges in ensuring seamless cross-platform functionality, which necessitates careful consideration during development.

## [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 class diagram represents the Domain Model for "Draw It or Lose It." The diagram illustrates relationships between key classes such as Game, Player, Team, and the base Entity class. Object-oriented principles, including inheritance and encapsulation, are evident. The Entity class acts as a base class, promoting code reusability, while derived classes (Game, Player, Team) showcase relationships and specialization. The diagram efficiently fulfills software requirements by organizing entities and their interactions.

**"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)

Recommendations

Operating Platform: Considering the need for cross-platform compatibility, it is recommended to adopt a web-based operating platform to allow seamless expansion to various computing environments.

Operating Systems Architectures: Opt for an architecture that supports web-based applications, ensuring compatibility across diverse operating systems such as MacOS, Linux, and Windows.

Storage Management: Employ a robust storage management system compatible with web-based platforms, ensuring efficient data storage and retrieval.

Memory Management: Leverage memory management techniques inherent in modern web-based operating platforms to optimize resource utilization for the Draw It or Lose It software.

Distributed Systems and Networks: Implement a distributed architecture to facilitate communication between various platforms. Consider network dependencies and component interactions to ensure smooth connectivity.

Security: Emphasize security features provided by the chosen web-based operating platform to safeguard user information across different platforms. Consider encryption and authentication mechanisms to meet the client's security requirements.

Server Side

| **Operating System** | **Server-Based Deployment** | **Licensing Costs** | **Comments** |
| --- | --- | --- | --- |
| Linux | Yes | Low/None | Linux is a robust choice for server-side hosting. It supports server-based deployment, has low licensing costs, and is scalable. Considered optimal for hosting web-based applications. |
| Mac | Yes | Moderate | Mac offers stable hosting with Unix-based architecture. While suitable, licensing costs can be moderate, and scalability for enterprise-level applications may be limited. |
| Windows | Yes | Moderate to High | Windows provides a user-friendly environment for hosting web applications. Licensing costs can be a concern, and it may not be as optimal as Linux for high-performance tasks. |

Client Side

| **Platform** | **Compatibility Considerations** | **Development Considerations** |
| --- | --- | --- |
| Linux | Varied desktop environments, testing on different distributions, expertise in Linux development. | Linux clients are diverse, and development should consider the varied desktop environments, testing on different distributions, and expertise in Linux application development. |
| Mac | Mac-friendly UI design, Mac development tools, expertise in macOS application development. | Developing for Mac clients requires consideration of Mac-friendly UI design, the cost of Mac development tools, and expertise in macOS application development. |
| Windows | UI adaptation for Windows, cost of Windows tools, expertise in Windows application development. | Supporting Windows clients involves UI adaptation for Windows, accounting for the cost of Windows development tools, and expertise in Windows application development. |
| Mobile Devices | Responsive UI design, compatibility with iOS and Android, cost of mobile development tools, expertise in mobile app development. | Development for mobile devices requires responsive UI design, compatibility with iOS and Android, consideration of the cost of mobile development tools, and mobile app development expertise. |

Development Tools

| **Platform** | **Commonly Used Tools** | **Impact on Development Team** | **Licensing Costs** | **Comments** |
| --- | --- | --- | --- | --- |
| Linux | VSCode, Python, C, Java | Dependent on the chosen programming language and distro | Often open-source or low cost | Linux development commonly utilizes open-source tools, and the impact on the team depends on language and distribution choices. |
| Mac | Xcode, Swift, Objective-C | Tailored to the macOS environment, expertise required | Moderate | Mac development involves the use of Xcode and languages like Swift, with tools tailored to the macOS environment. |
| Windows | Visual Studio, C#, .NET | Comprehensive tools for Windows development | Moderate to High | Windows development involves tools like Visual Studio, providing comprehensive support for Windows applications. |
| Mobile Devices | Android Studio (Java/Kotlin), Xcode (Swift) | Depends on the chosen framework and language | Often free | Mobile app development involves platform-specific tools like Android Studio and Xcode, with varying licensing costs. Cross-platform frameworks may provide cost benefits. |

**Recommendations**

Analyzing the characteristics and techniques specific to various system architectures, the following recommendations are made for The Gaming Room:

1. **Operating Platform:** Considering the need for cross-platform compatibility, it is recommended to adopt a web-based operating platform for hosting "Draw It or Lose It." This choice will facilitate seamless expansion to various computing environments, ensuring accessibility across diverse operating systems.
2. **Operating Systems Architectures:** Opt for an architecture that robustly supports web-based applications. This includes compatibility considerations for operating systems such as MacOS, Linux, and Windows. It is crucial to choose an architecture that ensures smooth execution and performance on each platform, given the varied nature of the client's user base.
3. **Storage Management:** Employ a robust storage management system compatible with web-based platforms. The selected storage management system should ensure efficient data storage and retrieval, catering to the dynamic nature of a gaming application. Consideration should be given to scalability, reliability, and ease of maintenance in the chosen storage solution.
4. **Memory Management:** Leverage memory management techniques inherent in modern web-based operating platforms to optimize resource utilization for the "Draw It or Lose It" software. This involves efficient allocation and deallocation of memory resources to enhance the overall performance and responsiveness of the gaming application.
5. **Distributed Systems and Networks:** Implement a distributed architecture to facilitate communication between various platforms. This involves designing a network that connects devices across different operating systems. Consideration should be given to network dependencies, potential outages, and ensuring continuous connectivity. The distributed system should be designed to handle data synchronization and communication seamlessly, enhancing the multiplayer gaming experience.
6. **Security:** Emphasize security features provided by the chosen web-based operating platform to safeguard user information across different platforms. Encryption and authentication mechanisms should be implemented to meet the client's security requirements. The goal is to ensure the confidentiality and integrity of user data, both within the application and during communication between devices.