

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 |
| --- | --- | --- | --- |
| 1.2 | 08/11/2023 | David Hughes | Added more detail to the recommendation 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 at hand is to develop a game application called "Win It or Lose It" for The Gaming Room. The client requires a web-based distribution environment for the game application. The proposed solution is to design and implement the game application using Java and web technologies to ensure cross-platform compatibility. The application will allow users to play games, join teams, and interact with other players. By utilizing the singleton pattern, the application will ensure that only one instance of the GameService class exists at any given time. This will provide centralized management of game data and ensure consistency. The use of the iterator pattern will allow efficient searching for existing games, teams, and players based on their names. These design decisions will enable the application to meet the client's requirements effectively.

## Requirements

The client's business and technical requirements for the game application are as follows:

The game application should be web-based and accessible from different computing environments.

Users should be able to play games, join teams, and interact with other players.

Game and team names should be unique to avoid conflicts.

The application should ensure data consistency and centralized management of game data.

## [Design Constraints](#_2et92p0)

Developing the game application in a web-based distributed environment introduces the following design constraints:

Cross-Platform Compatibility: The game application should be compatible with different operating systems such as Mac, Linux, and Windows. This requires using platform-independent technologies and libraries.

Scalability: The application should be designed to handle a potentially large number of concurrent users. The design should consider load balancing and performance optimization techniques.

Security: The game application should prioritize security to protect user information and prevent unauthorized access. This includes implementing secure authentication and data encryption mechanisms.

Network Communication: The application must efficiently handle network communication between the client and server components. Proper protocols and data transfer methods should be employed to ensure reliable and efficient communication.

These design constraints impact the development process by requiring the selection of appropriate technologies, architectural patterns, and security measures to meet the client's requirements in a distributed environment.

## [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 provided represents the Domain Model for the game application. It consists of the following classes: ProgramDriver, SingletonTester, GameService, Entity, Game, Player, and Team. The Entity class serves as the base class for other entities and provides common attributes and behaviors such as id and name. The Game class inherits from the Entity class and represents a game in the application. The Player and Team classes also derive from the Entity class and represent players and teams, respectively.

The UML diagram demonstrates object-oriented programming principles such as inheritance and encapsulation. Inheritance allows the Game, Player, and Team classes to inherit attributes and behaviors from the Entity class, promoting code reuse and maintainability. Encapsulation ensures that the internal state of each class is protected and can only be accessed through appropriate methods. These principles enable efficient organization and management of game-related entities within the application.

**"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 provides a stable and secure foundation for hosting a web-based application. With robust server capabilities, it is designed for performance but may have compatibility issues with certain enterprise software. Licensing costs are moderate but must be considered. | Linux offers exceptional stability, flexibility, and cost-effectiveness. It can be customized to suit specific needs, making it ideal for various server configurations. Open-source options can minimize costs, but more technical expertise may be required for setup and administration. | Windows has widespread usage and extensive developer support, providing compatibility with a wide range of software and tools. Licensing costs vary, and while it has powerful server capabilities, it can be more vulnerable to security threats, requiring careful configuration. | Mobile devices offer portability and accessibility but face challenges like limited processing power and screen size. The server-style configuration may require optimization techniques for efficient resource utilization. |
| **Client Side** | Supporting Mac clients involves compatibility with different versions of macOS, rigorous testing, and potential platform-specific issues. Expertise in macOS development is necessary, and thorough planning will offset higher development costs. | Linux clients require expertise in various distributions. Detailed knowledge of different environments and thorough testing ensures broad compatibility. Consideration of various Linux flavors may add time but ensures a high-quality product. | Windows clients need an understanding of specific development environments, tools, and security considerations. Development might be more streamlined due to extensive support, but attention to security can't be neglected. | Supporting mobile devices demands knowledge of responsive design and optimization for performance. It includes iOS and Android-specific considerations, ensuring universal compatibility and an optimal user experience. |
| **Development Tools** | Java, HTML, CSS, JavaScript; Xcode, IntelliJ IDEA, Eclipse, NetBeans. Tailored tools for macOS development enhance productivity but may have associated costs. Coordinated efforts with other platform teams enable synergistic development. | Java, HTML, CSS, JavaScript; IntelliJ IDEA, Eclipse, NetBeans. Open-source tools can reduce costs, but variant-specific tools may be required. Collaboration with Mac and Windows teams ensures consistency. | Java, HTML, CSS, JavaScript; IntelliJ IDEA, Eclipse, NetBeans. Tools are widely available, some at a cost, but provide efficient development. Coordination across teams ensures product uniformity. | HTML, CSS, JavaScript; React Native, Flutter. Mobile development tools offer flexibility for cross-platform development but require specialized skills. Collaborative development with traditional platforms ensures a cohesive product. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

Operating Platform:

For an optimal balance between performance, flexibility, and cost, it's recommended to deploy "Draw It or Lose It" on a Linux-based server environment. Linux is revered for its stability and cost-effectiveness due to its open-source nature. Choosing a web-based application design allows The Gaming Room to effortlessly expand the game across various computing landscapes.

Operating Systems Architectures:

Linux, our recommended operating platform, offers a versatile architecture. The Kernel, at its core, interacts directly with system hardware. Above this layer is the Shell, which interfaces with the Kernel, allowing users and administrators to interact with the system. With standard libraries and system utilities complementing it, Linux ensures a robust foundation. A web-based frontend means our game will be accessible across multiple systems including Mac, Linux, Windows, and mobile devices without requiring individual native builds.

Storage Management:

For our storage needs, PostgreSQL, an open-source relational database management system (RDBMS), is recommended. PostgreSQL provides the right blend of performance, extensibility, and reliability. It's particularly suited for applications like "Draw It or Lose It", ensuring that game data, user profiles, and other key information are stored securely and can be retrieved quickly.

Memory Management:

The choice of Java for backend development comes with built-in memory management. Java's garbage collector autonomously manages memory, discarding unreferenced objects and ensuring efficient use of server resources, hence reducing potential memory leaks and enhancing the overall gameplay experience.

Distributed Systems and Networks:

To ensure seamless communication between various platforms, "Draw It or Lose It" will leverage HTTP/HTTPS protocols. Universal support for these protocols ensures compatibility across devices. Redundancy mechanisms and failover strategies will be implemented to safeguard against potential service interruptions, while load balancers will distribute incoming game traffic, ensuring servers operate efficiently without being overwhelmed.

Security:

Ensuring robust security is paramount for any gaming application. By incorporating Maven, a build and dependency management tool, we can manage software dependencies and ensure that all used libraries are up-to-date and free from known vulnerabilities. Advanced encryption standards will be employed to protect user data both in transit and at rest. OAuth 2.0 will be utilized to provide secure authentication, and HTTPS will ensure all data transmitted between the server and clients remains encrypted. Regular security audits and the continuous monitoring and updating of dependencies via Maven will help mitigate potential threats and vulnerabilities.