

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 4**](#_Toc115077324)

**Evaluation 6**

**Recommendations 9**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/07/2023 | Phillip L Moffett II | Initial Draft of Document |
| 1.1 | 11/08/2023 | Phillip L Moffett II | Write down Executive Summary, Requirements, and Design Constraints |
| 1.2 | 11/09/2023 | Phillip L Moffett II | Write down Evaluation and Recommendations |
| 1.3 | 11/10/2023 | Phillip L Moffett II | Write down Domain Model |
| 1.4 | 11/11/2023 | Phillip L Moffett II | Update all to the Final Draft |
| 1.5 | 11/23/2023 | Phillip L Moffett II | Evaluate the characteristics, advantages, and weaknesses of various platforms |
| 1.7 | 12/08/2023 | Phillip L Moffett II | Update the Recommendations (COMPLETE) |

**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)

Creative Technology Solutions (CTS) proposes to develop a web-based version of The Gaming Room’s existing game, Draw It or Lose It. This game involves teams guessing a phrase based on rendered images, akin to Win, Lose or Draw. Our solution ensures adaptability for one or more teams, each with multiple players. Crucially, uniqueness is prioritized for game and team names, addressing the client’s need for a seamless user experience. We plan to implement unique identifiers for managing a single instance of the game in memory, meeting the client’s request efficiently.

## Requirements

The client has requested that the following **software requirements**be met for the game application:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

Design constraints for the web-based distributed environment include real-time communication challenges, data consistency across platforms, and limitations on client-side processing. These constraints emphasize the need for robust networking, a well-structured database, and efficient server-side logic to ensure a scalable and responsive architecture.

## [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 provided UML Class Diagram illustrates the relationships and structure of classes within the software design.

**ProgramDriver Class:**

• Represents the entry point of the program with a main() method.

**SingletonTester Class:**

• Contains a method testSingleton().

**Entity Class:**

• Represents a base class with attributes id (long) and name (String).

• Demonstrates object-oriented principles like encapsulation with private attributes and provides public methods (getid(), getName(), toString()) for accessing and displaying information.

**GameService Class:**

• Manages games and their details.

• Utilizes the Singleton pattern, ensuring a single instance of GameService exists.

• Maintains lists of games and generates unique identifiers (nextGameId, nextPlayerId,

nextTeamId).

• Exhibits encapsulation with private attributes and exposes public methods to add

games, retrieve game information, and manage identifiers.

**Game Class:**

• Represents a game with a list of teams.

• Demonstrates the composition relationship with Team, as a game consists of one or

more teams.

• Uses encapsulation by having private attributes and provides methods to add teams and

generate a string representation.

**Team Class:**

• Represents a team with a list of players.

• Exhibits a composition relationship with Player, indicating that a team comprises one or

more players.

• Implements encapsulation and provides methods to add players and create a string

representation.

**Player Class:**

• Represents a player.

• Demonstrates encapsulation with private attributes (id, name) and provides a method

to generate a string representation.

**Connections:**

• Solid lines with “0…\*” indicate associations between GameService and Game, Game and

Team, and Team and Player.

• Open arrow lines pointing to Entity from Game, Team, and Player signify dependencies

on the Entity class.

**Object-Oriented Programming Principles:**

• Encapsulation: Private attributes and methods encapsulate the internal details of each

class, allowing controlled access to data and behaviors.

• Composition: The relationships between Game and Team, as well as Team and Player,

demonstrate the composition principle, emphasizing the structure of the whole (game

composed of teams, team composed of players).

• Singleton Pattern: The GameService class utilizes the Singleton pattern, ensuring a single instance is responsible for managing game-related operations, enhancing efficiency and avoiding unnecessary duplications.

**"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** | Evaluating Mac for hosting a web-based software application reveals its strengths lie in stability and security. Mac’s UNIX-based architecture provides reliability, making it suitable for server-side hosting. However, potential disadvantages include higher hardware costs compared to Linux-based alternatives. | Linux excels in hosting web-based software applications, offering stability, scalability, and cost-effectiveness. Its open-source nature and robust security features make it a preferred choice. However, Linux may require more expertise for configuration and maintenance. | Windows, widely used in enterprise environments, presents advantages such as user-friendliness and compatibility. However, licensing costs can be a concern, and it may not be as optimized for server-side web hosting as Mac or Linux. | Mobile devices pose challenges for server-side hosting due to their diverse platforms. Utilizing cloud-based solutions and cross-platform development tools can enhance compatibility. Consideration must be given to optimizing server-side logic for efficient communication with mobile applications. |
| Considering the client's requirement for a web-based application, all three operating platforms—Mac, Linux, and Windows—provide server-based deployment methods. However, Linux stands out for its cost-effectiveness and open-source nature, making it a strong candidate for hosting the Draw It or Lose It software. Mac's stability and security make it a reliable option, though higher hardware costs might be a drawback. Windows, with its user-friendly interface, can also serve as a suitable platform, but potential licensing costs should be considered. | | | | |
| **Client Side** | Developing for Mac clients requires considerations of cost, time, and expertise. While Mac users are generally accustomed to a seamless experience, development costs may be higher due to Apple’s ecosystem requirements. Timeframes for development should account for Apple’s stringent app review process. | Supporting multiple types of clients on Linux involves considerations of cost, time, and expertise. Linux users often appreciate open-source solutions, but ensuring compatibility across various distributions may require additional development time and expertise. | For Windows clients, development considerations include factors like cost, time, and expertise. Windows users expect a user-friendly experience, but addressing compatibility across different Windows versions may impact development timelines and require diverse expertise. | Supporting multiple types of clients on mobile devices necessitates considerations of cost, time, and expertise. Cross-platform development tools, like React Native or Flutter, can streamline the process, reducing costs and development time. Expertise in mobile app development is crucial for optimal results. |
| Supporting multiple types of clients on mobile devices necessitates considerations of cost, time, and expertise. Cross-platform development tools, like React Native or Flutter, can streamline the process, reducing costs and development time. Expertise in mobile app development is crucial for optimal results. | | | | |
| Development Tools | Building software for Mac involves utilizing programming languages such as Swift and Objective-C. Integrated Development Environments (IDEs) like Xcode are essential tools for Mac application development, providing a comprehensive environment for coding, testing, and debugging. | Linux software development often involves languages like C, C++, and Python. Popular IDEs like Visual Studio Code and Eclipse, along with robust command-line tools, support efficient development on Linux. | Developing for Windows requires languages like C#, .NET, and C++. Visual Studio is a prominent IDE offering a range of tools for Windows application development, ensuring a productive development environment. | Developing for mobile devices involves languages like Swift or Kotlin for native development, or JavaScript with frameworks like React Native or Flutter for cross-platform development. IDEs such as Android Studio and Xcode support mobile app development, providing tools for designing, coding, and testing applications. |
| Developing for mobile devices involves languages like Swift or Kotlin for native development, or JavaScript with frameworks like React Native or Flutter for cross-platform development. IDEs such as Android Studio and Xcode support mobile app development, providing tools for designing, coding, and testing applications. | | | | |

## 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:**

* *Recommendation:* The recommended operating platform for hosting the Draw It or Lose It web-based software application is Windows. Given its widespread use and user-friendly interface, Windows stands out as a suitable choice. Its compatibility with a large user base familiar with Windows environments aligns well with the project's goals. Despite potential licensing costs, the stability and robust compatibility offered by Windows make it a reliable choice for the scalability and expansion of Draw It or Lose It to other computing environments.

1. **Operating Systems Architectures:**

* *Chosen Architecture:* Windows employs a hybrid kernel architecture, combining elements of both monolithic and microkernel structures. This design choice ensures a balanced approach, supporting robust performance and efficient resource management for the Draw It or Lose It software. The hybrid kernel architecture of Windows provides a solid foundation for the scalability and adaptability required for expansion into diverse computing environments.

1. **Storage Management:**

* *Recommendation:* Opt for a relational database system, specifically Microsoft SQL Server, for efficient and structured data storage on the Windows platform. Choosing a database system compatible with the Windows environment ensures seamless integration and reliable storage management. Microsoft SQL Server provides a proven and scalable solution, supporting the anticipated growth of Draw It or Lose It.

1. **Memory Management:**

* *Explanation:* Windows utilizes virtual memory techniques to optimize memory usage, contributing to the overall performance of the Draw It or Lose It software. Efficient memory management on the Windows platform ensures a responsive and scalable application, critical for providing a smooth user experience.

1. **Distributed Systems and Networks:**

* *Approach:* Implementing RESTful APIs is recommended for cross-platform communication on the Windows platform. This approach facilitates communication between devices and supports the client's desire for Draw It or Lose It to operate seamlessly across various platforms. Consideration of potential network outages and connectivity issues is essential. Leveraging Windows-specific networking protocols ensures robust communication, and careful consideration of dependencies between components addresses challenges related to connectivity and outages.

1. **Security:**

* *Security Measures:* Windows offers robust security features that can be harnessed to protect user information. Implementing HTTPS for secure data transmission is crucial, and employing encryption mechanisms compatible with Windows security protocols enhances user protection across various platforms. The chosen operating platform provides a secure foundation, and additional security measures such as regular updates and monitoring should be implemented to ensure a high level of protection for user information.