

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 | 9/17/2023 | Connor Hudson | Initial Software Design Draft |

**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) is tasked with developing a web-based version of the game "Draw It or Lose It" for our client, The Gaming Room. The game is designed to allow multiple teams with multiple players to participate. Key requirements include unique game and team names and the limitation of only one instance of the game in memory at a time. This document outlines the software design to meet these requirements and recommends an appropriate operating platform for future expansion.

## Requirements

The client's requirements include supporting multiple teams and players, ensuring unique game and team names, and enforcing a single instance of the game in memory. Technical requirements include using web-based distributed architecture.

## [Design Constraints](#_2et92p0)

The design constraints include developing in a web-based distributed environment. This constraint impacts application development by requiring server-side hosting and cross-platform compatibility.

## [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 of the software. It shows classes for Game, Team, and Player, all derived from the base Entity class. This design ensures efficient handling of game instances, teams, and players. Object-oriented principles such as inheritance and encapsulation are utilized to fulfill software requirements.

**"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 offers a stable and user-friendly environment, suitable for hosting web-based software. May be more expensive and need specific optimization. | Cost-effective and customizable, providing flexibility and stability. May require more technical expertise for setup and maintenance. | Robust solutions with excellent compatibility. Familiar and widely used. Potentially higher costs with some complexity in server management. | Limited server-side capabilities. Wide user base for client applications. |
| **Client Side** | Platform-specific considerations to ensure compatibility. Investment in macOS devices for testing and development. Time varies depending on the project complexity. Will need developers with expertise. | While flexible, will require customization for different distribution efforts. Likely most cost-effective, though significant testing will be needed to ensure compatibility. Will likely need developers with expertise. | Large user base. May need consideration based on platform version. Potential licensing fees for platform specific development tools. Time can be reasonable. Expertise in Visual Studio preferred. | Development involves creating platform-specific apps. Each platform has its development requirements, cost considerations, and expertise needs. Can be costly due to the need for platform-specific development. Time varies based on the complexity of the mobile app and the need to target multiple platforms. Developers with expertise in mobile app development for iOS (using Swift) and Android (using Java/Kotlin) will be required. |
| **Development Tools** | Development for Mac may involve using Xcode and languages like Swift for iOS and web technologies (HTML, CSS, JavaScript) | Linux development often involves scripting languages like Python, along with web technologies (HTML, CSS, JavaScript). | Windows development tools may include Visual Studio and languages like C#. Web technologies (HTML, CSS, JavaScript) are also commonly used. | Mobile app development requires platform-specific languages and tools, such as Swift for iOS and Java/Kotlin for Android. |

## 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**: For the optimal expansion of Draw It or Lose It onto various computing environments, we recommend employing a Linux-based operating platform. Linux is well-suited for hosting web-based applications, offering a cost-effective solution with high customizability. It provides a stable environment for web hosting, which aligns with the client's requirement for a web-based game.
2. **Operating Systems Architectures**: To leverage the benefits of Linux, we recommend utilizing a 64-bit architecture. A 64-bit architecture enhances system performance, memory addressing capabilities, and compatibility with modern hardware. This architecture is well-suited for handling the resource-intensive aspects of the Draw It or Lose It software.
3. **Storage Management**: We propose the implementation of an open-source relational database management system (RDBMS) for efficient storage management. PostgreSQL is a highly recommended choice due to its robustness, reliability, and performance. RDBMS offers data organization capabilities that align with the client's requirements for unique game and team names.
4. **Memory Management**: To ensure optimal memory management for the Draw It or Lose It software, we recommend leveraging Linux's native memory management capabilities. Linux provides sophisticated memory management tools that can be harnessed to optimize memory usage. This is crucial for maintaining the limitation of only one instance of the game in memory at a time, a key client requirement.
5. **Distributed Systems and Networks**: Considering the client's goal of facilitating communication between various platforms, we propose the implementation of distributed systems using RESTful APIs. RESTful APIs ensure efficient and standard communication between different platforms, enabling the game to operate seamlessly across various environments. Load balancing and redundancy should be considered to ensure network stability and minimize the risk of connectivity issues.
6. **Security**: Security is a paramount concern for the client, and we recommend implementing robust security measures to protect user information on and between various platforms. This includes the use of encryption and secure communication protocols to safeguard sensitive user data. Regular security audits and updates are essential to ensure user safety and maintain a secure gaming environment.