

Draw It of 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 | 11/8/23 | Logan Martha | Following CTS’s software requirements, develop a web-based version of the gaming app “Draw It or Lose It.” |

**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)

CTS has taken on a new client, The Gaming Room. They are looking to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It. It is currently only available on Android.

## Requirements

1. *A game will have the ability to have one or more teams involved.*
2. *Each team will have multiple players assigned to it.*
3. *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
4. *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)

1. Cross-Browser Compatibility: Web-based games must work consistently across different web-browsers.
2. User Experience: Game must provide and enjoyable experience for the users.
3. Updates/Maintenance: Code must be clean and concise to provide the necessary updates and maintenance.
4. Game must clearly follow those of Draw It of Lose It.

## [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)

Entity is the ‘parent’ class of Game, Team, and Player. Entity has two private variables of ID which is a long variable, and Name which is a string. It has a default constructor that is private and 4 public methods. The Game class, which inherits Entity, has one private object which is a List called teams. It then has 3 additional public methods. Team also has one private object which is also a List called players. It also has 3 public methods. Player Class has no unique objects and has 2 public methods. Game Service Class inherits Game. It has 5 unique private objects. A list called games, a long called nextGameId, a long called nextPlayerId, a long called nextTeamId, and a companion object called service. It also has a private default constructor and 7 additional methods. GameService, Game, Team, and Player are all have “zero to many” when regarding multiplicity.

**"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** | -**Characteristics**:  -Centralized Control  -Strong Security  -**Advantages:**  -Enhanced Security  -**Disadvantages:**  **-**Complex Implementation  -Not very adaptable | -**Characteristics**:  -Open Source  -Customizable  -Great Community  -**Advantages:**  -Cost Effective  -Performance  -Scalability  -**Disadvantages:**  **-**Complex for beginners  -Limited Driver Support | -**Characteristics**:  -User-Friendly  -Compatibility  -**Advantages:**  -Easy to Use  -Good Vendor Support  -**Disadvantages:**  -Expensive  -Complex | -**Characteristics**:  -Portable  -Touch Interface  -**Advantages:**  -Accessibility  -Growing User Base  -**Disadvantages:**  -Limited Resources  -Security Concerns  -Interface Constraints |
| **Client Side** | **Cost:**  Hardware  Software  Training  Ongoing Support  **Time:**  Need time for initial setup, configuration, maintenance, and troubleshooting.  **Expertise:**  MacOS features and compatibility w/ other software applications | **Cost:**  Free to use, but has hardware expenses and licensing.  **Time:**  Need time for initial setup, configuration, maintenance, and troubleshooting.  **Expertise:**  Linux features and compatibility w/ other software applications | **Cost:**  Windows Operating System, Microsoft Office suite, specialized software, hardware.  **Time:**  Need time for initial setup, configuration, maintenance, and troubleshooting.  **Expertise:**  Requires understanding of multiple Windows versions | **Cost:**  The physical devices themselves, mobile device management solutions, mobile application license, security, and support.  **Time:**  Need time for setup, device programming, configuration, updates, app installations.  **Expertise:**  Understand the many types of phones and their systems. (IOS, Android) |
| **Development Tools** | **Languages:**  Swift  Objective-C  **Tools:**  Xcode  Interface Builder  Git and Github | **Languages:**  C/C++  Python  Java  **Tools:**  Visual Studio Code  Eclipse  Git and Github  Package Managers  Docker | **Languages:**  C#  C/C++  Java  Python  Java Script  **Tools:**  Visual Studio  Eclipse  .NET Framework  MSBuild  Windows SDK | **Languages:**  Swift  Objective-C  Java  Kotlin  **Tools:**  Xcode  Android Studio  Android/iOS SDK  AppCode  Unity and Unreal Engine |

## 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**: My recommendation is using the Windows Operating Platform.
2. **Operating Systems Architectures**: I believe that Windows User-Friendly and great Compatibility sets up for a great operating platform to use for this project. It is also an easy to use platform that offers a lot of support from vendors.
3. **Storage Management**: I believe a cloud-based storage solution would support this application well. Utilizing cloud storage allows the most amount of flexibility and scalability while still being reliable. Since Draw It or Lose wants to be integrated across multiple platforms, being able to have that flexibility if they wish to expand more in the future is important.
4. **Memory Management**: The core component responsible for managing memory, processes, I/O requests, and interfacing hardware is the modular Windows kernel. Windows is great in utilizing dynamic memory management. Windows dynamically allocated and deallocates memory to processes as needed. This allows the system to optimize performance. Windows also uses virtual memory to allow to possess more memory than physically available.
5. **Distributed Systems and Networks**: Windows utilizes TCP/IP allowing for communication over network. Windows also is compatible with REST APIs. This allows them to communicate with other systems that are also compatible. REST APIs utilize HTTP/HTTPS for communication.
6. **Security**: Windows has many different security measures within their operating system. They use User Account Control (UAC) for access control, a Windows Defender for anti-virus and malware protection.