

# **Draw It Or Lose It**

# **CS 230 Project Software Design**

Version 1.1

## 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 | 01/29/2025 | Christopher Clark | First Iteration |
| 1.1 | 02/10/2025 | Christopher Clark | Second Draft |
| 1.2 | 02/22/2025 | Christopher Clark | Final report |

## [Executive Summary](#_sbfa50wo7nsh)

My team and I at CTS were asked by “The Game Room”, to design a web-based game that works on multiple platforms called “Draw It or Lose It.” Currently the game only works on the Android platform, however the company wants the game to run on multiple different operating systems. The game will allow for multiple teams with multiple players per team. The game will have 4 rounds lasting one minute each.

## Requirements

The client requires the following requirements be met for the game. The game will have the ability to have one or more teams involved. Each of the teams will have multiple players assigned to each team. The game and team names must be unique, and the players will have the ability to see if the name they chose is taken. Finally, only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The design constraints for the game consist of:

* Cross-platform. The current game runs on the Android system only at the moment. The new design of the game will work on multiple platforms by being converted to a web-based application compatible with MACOS, Windows, and Linux.
* User interface. With the existing game, a similar interface design will be used for familiarity for the users. A new design is an option to incorporate a new look and feel.
* Unique Usernames. Each game and player will have a unique name that cannot repeat. The players must have the ability to see if the name they want to choose is taken. This will be accomplished by incorporating unique IDs and identifiers to games, players, and teams.
* Multiple Teams. The game will have the capability to have multiple teams withs multiple players in each team. A client-server architecture will help accomplish this as well as verifying that the servers can handle that much traffic.

## [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)

A UML diagram, such as the one below, offers a visual design of the game system and the process of the code. Looking at this diagram, the entity class creates a relationship between the game, team, and player classes. The arrow that links the classes indicate that the game, team, and player classes will inherit from an attribute from a super class. The driver class uses the singleton tester to test the code indicated by the two classes linked at the top left of the diagram. One of the restrictions for this game is that only one instance of the game can exist in the system memory which is tested by the singleton tester. The functionality of the game and complex methods are held in the Game service class. This class follows the singleton pattern and is the main starting class. The game, team and player classes are subclasses of the entity class. ID and name are both protected attributes with the default constructor also being protected, so null objects are blocked at the creation, making it so that only overloaded constructors may be used.

The UML diagram shows multiple examples of object-oriented program techniques such as: polymorphism and inheritance in the entity class and abstraction and encapsulation in the techniques used to add teams. The diagram shows how useful of a tool it is when it comes to creating a project.

**"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** | Apple offers:   * Server-based environments * Having a mac server offers advantages. * Mac has a well-established interface and easy administrations.   Weakness:   * The price to maintain a mac server is higher than most. | The advantages of Linux are:   * The servers capabilities which is highlighted by the companies that use their servers. * Linux servers are a free and open source, making it cheap and easy. * The customization of the security settings makes it great for securing data.   Weaknesses include:   * The difficulty in learning the system * The lack of supported applications * The potential migration issues between other operating systems. | The advantages of windows include:   * The overall well-roundedness of the system. * There are plenty of supported applications * The ease of updating and patching the system increasing security, and the familiarity the developers will have with the system.   The main weakness of using windows is:   * The expense due to needing to purchase the licensing. | The difficulty and lack of knowledge with using mobile devices as servers makes it an unviable option.  Oracle is an example company with great capabilities. The advantages would include:   * The IOS dev tool support * Androids dev tool support * Synchronization to Oracle. |
| **Client Side** | * Easy to use with the intuitive and user-friendly interface. * Helps make it capable to manage multiple clients saving development cost and time. | * Being free to use makes it cost friendly. * May require special equipment, diverse expertise, and training to learn the system. | * May be the most expensive licensing cost especially compared to open-source systems. * The system is easy to use and has plenty of security and 3rd party applications. | * This option may require major changes to design and connectivity due to the lack of capabilities and limitations. * Include native features such as camera, GPS, and notifications. |
| **Development Tools** | JavaScript and Node.js are the most common languages with VScode and Xcode being the most common IDEs | It has a robust command line interface like Apt and Yum. VScode, Atom, and Sublime text are examples of IDEs used. | Languages include C# and .NET framework. With IDEs such as Visual Studios and JetBrains. | There are a vast number of languages with common ones like Java and Script. IDEs include Android studios, Xcode, and device emulators. |

## 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**: A web-based operating platform is recommended for expanding the game to other environments. The Linux servers would be the most favorable due to the reduced license costs, the non-limited data center access, and the good security.
2. **Operating Systems Architectures**: The architecture will be centered around the client-server and web-based technologies. For the game interface and user interaction, JavaScript and HTML will be used as programming languages. A multitier architecture will be used on the server-side.
3. **Storage Management**: A relational database management system and cloud storage services would make for an appropriate storage management system. The RDBMS handles data related to the game progress, profiles, and stats, where the cloud storage stores media assets.
4. **Memory Management**: Modern web browsers provide automatic memory management techniques and a garbage collection mechanism to handle the memory allocation and deallocation.
5. **Distributed Systems and Networks**: The game will use distributed software architectures and leverage network connectivity to enable communication between the various platforms. A centralized server or cloud infrastructure will serve as a communication hub between the game clients and would handle the real-time updates, internal messaging exchange between the various devices, and game synchronization.
6. **Security**: Security is a major concern when it comes to anything online and dealing with multiple platforms. Several security measures can be implemented such as user authentications, secure communication protocols, and data encryptions. Implementing secure encryptions to the users data and passwords ensure all data is inaccessible to outside threats.