

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 | 12/11/2024 | Jacob Segarra | Final draft. Recommendations for Draw It or Lose It software requirements. |

**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 Gaming Room desires to expand their popular Android game, "Draw It or Lose It," into a web-based application accessible across multiple platforms. The goal is to broaden the game's audience and enhance its accessibility, allowing teams to compete in guessing puzzles based on a library of stock drawings.

To meet this demand, Creative Technology Solutions (CTS) proposes developing a scalable, web-based version of the game, leveraging modern web technologies to ensure compatibility across various devices and platforms. The solution will prioritize user engagement through intuitive design, ensure the uniqueness of game and team names, and maintain a singleton instance of memory to manage resources effectively.

## Requirements

The Gaming Room’s business and technical requirements can be segmented as such:

Web-Based Distributable:

* Cross-platform
* Scalability in-mind

Gameplay:

Team Management:

* Multiple players assigned to each team
* Create multiple teams within a game
* Assign a unique identifier to each team
* Assign a unique identifier to each player

Game Creation:

* Ability to create a new game instance
* Assign a unique identifier to each game
* Prevent multiple instances of the same game from running concurrently

## [Design Constraints](#_2et92p0)

Cross-platform compatibility:

Compatibility across multiple platforms is a significant constraint for web-based games. A consistent and optimal experience across various devices and browsers is paramount. The development team will need to verify across all major web browsers as part of testing. Features such as consistent rendering, flexible layouts, and device detection should be considered user accessibility.

Scalability and performance:

As with most businesses, the goal of The Gaming Room is growth. The product should scale to accommodate an increasing player base without significant performance loss or network latency. This constraint requires optimizing database access, minimizing resource consumption, and considering caching mechanisms.

Network infrastructure:

If the application is to become a cross-platform, web-based distributed environment, it will require investment into increasing server capacity and assuring the reliability of network infrastructure. This can be done by increasing server count, upgrading existing servers, utilizing cloud providers, and using low latency networks.

## [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 base class for the Game, Team, and Player classes and provides inheritance to them. Everything in the game (players, teams, and game names) start with this building block. The child classes can share Entity’s common attributes and methods.

Game Class:

This class inherits from the Entity class. It possesses its own attributes in addition to the ones inherited by the parent class (Entity). It also has relationships with the Team and Player class. In what is known as a zero-to-many relationship, a game can have multiple teams and multiple players or none.

Team Class:

This is another child to the parent class Entity. It has its own attributes along with those given to it from Entity. This class has relationships with the Game and Player class. Multiple teams can belong to one game giving this class a many-to-zero relationship with the Game class. A team may have multiple players establishing a zero-to-many relationship with the Player class.

Player Class:

Lastly, the Player class. Another class that inherits from the Entity class, this class has its own attributes along with those given to it by Entity. The Player class has a many-to-zero relationship with the Team class. There may be multiple players on a single team.

GameService:

A service class to provide functionality to “Draw It or Lose It” by way of game-related operations with the application. GameService has a direct relationship with the Game class and an indirect relationship with the Team and Player class (Through the Game class). It is a zero-to-many relationship with Game class.

The UML diagram depicts classes and their relationships. The hollow arrowhead connecting Entity, Game, Team, and Player shows the three classes inherit from Entity. The lines between GameService, Game, Team, and Player show “zero or more” relationships between the classes as described above. This is known as multiplicity and refers to the number of instances of one class that can be related to a single instance of another class.

"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: Mac can be used as a server for web-based software applications, but it's less commonly chosen compared to Linux or Windows. It offers a Unix-based environment, which is favorable for hosting web applications.  Advantages: Unix-based system provides robust security and stability. Mac OS Server includes features like Apache web server and PHP support.  Weaknesses: Limited hardware choices for server-grade Macs. It may be more expensive compared to Linux-based solutions. | Characteristics: Linux is a popular choice for hosting web-based software applications. It offers a wide range of distributions suitable for server use, with excellent stability and scalability.  Advantages: Cost-effective, highly customizable, and offers a wide range of server software options. Linux is known for its security and reliability.  Weaknesses: May require more technical expertise to set up and manage compared to Mac or Windows. | Characteristics: Windows Server is a commonly used platform for hosting web applications, especially applications built using Microsoft technologies.  Advantages: Excellent integration with Microsoft tools and technologies. Offers a range of web server options like IIS. Good support for .NET applications.  Weaknesses: Licensing costs can be high. May not be as well-suited for open-source software development. | Characteristics: Mobile devices don't typically host web applications. Servers are used for mobile app data.  Advantages: Proximity to data for reduced latency. Cost-efficiency for specific use cases. Easy deployment for small-scale applications.  Weaknesses: Limited resources can lead to performance issues. Scalability challenges. Reliability issues like battery drain. Network dependencies and potential bandwidth limitations. |
| Client Side | Developing for Mac clients typically involves using Apple's development tools such as Xcode. Costs can be moderate, and expertise in Swift and Objective-C may be required. | Linux clients can vary depending on the distribution and desktop environment. Costs are typically low, but expertise may be required for distribution-specific considerations. | Windows clients often involves using Visual Studio. Costs can vary, but expertise in .NET languages like C# may be required. | Developing for mobile devices involves platform-specific development. Costs can vary depending on the number of platforms targeted. Expertise in languages like Swift (iOS), Kotlin (Android), or cross-platform tools like Flutter may be required. |
| Development Tools | Xcode is the primary integrated development environment (IDE) for Mac applications. It supports Swift, Objective-C, and C++. | Linux supports a wide range of programming languages. Popular IDEs include Visual Studio Code, IntelliJ IDEA, and Eclipse. | Visual Studio is the primary IDE for Windows development, supporting many languages such as C# and C++. | For iOS, Xcode and Swift/Objective-C are used. For Android, Android Studio and Kotlin/Java are common. Cross-platform tools like Flutter (Dart) can streamline development for both platforms. |

## 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:

To expand The Gaming Room’s “Draw It or Lose It” application, the operating platform suggested is Windows OS. Using Windows provides users with ease of use and access to a variety of software packages. Windows also offers a plethora of software engineering abilities for developers.

1. Operating Systems Architectures:

Windows provides services used by all Windows-based applications that enable applications to show a Graphical User Interface (GUI) while accessing system resources and much more. These applications also refer to Graphics and Multimedia, messaging, and web services. These services can be implemented with a user account or a server.

1. Storage Management:

Storage recommendations will vary with anticipated growth. Windows Storage Server in conjunction with Microsoft Azure are two well-paired options. Azure is cloud based and potentially viable on its own. Both products are scalable while Windows Storage is easier to manage.

1. Memory Management:

Windows uses several techniques to manage memory. Virtual address space, memory allocation, and memory protection all work to allocate and regulate memory use.

1. Distributed Systems and Networks:

Distributed systems and networks can have common issues like queuing and congestion problems. Common problems, in relation to using the distributed system, include independent failed components, absence of a global clock, simultaneous computations of components which can result in lagging computing performance and connection problems among individual users. These problems may begin to show as the application scales. The Gaming Room should look to increase network and server capacity before it’s an issue.

1. Security:

Windows offers WAF (Web Application Firewall), regular security audits, encryption, and secure communication protocols for server-side security. As company growth continues, regular user authentication is recommended for maintaining security on the client-side.