

<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 | 03/23/2025 | Sungho Yoon | Initial version |

**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 purpose of this document is to outline the solution for web-based game application “Draw It or Lose It”. The game is designed by our client The Gaming Room. Inspired by the classic TV show Win, Lose, or Draw, this game renders image from a large library of stock drawings, and teams will compete to guess what is being drawn. The client needs to develop web-based version of the gaming app, which will enable various devices to access the game.

## Requirements

1. The game should support web-based access and ensure compatibility with different operating systems and devices.
2. The game will have ability to have one or more teams involved.
3. Multiple players will be assigned to each team.
4. The app should remain game and team names as unique. Users will check if a name is in use when choosing a team name.
5. Only one instance of the game can exist at the same time.
6. Each round should have time limits, and rendered image will be revealed progressively until the 30-second.
7. Once a team fails to guess the answer, the other teams should have 15-second to guess the answer.

## [Design Constraints](#_2et92p0)

1. Web-based environment

Web-based platform has possible constraints related to compatibility with various operating systems and devices. Also, security and network consistency is the other constraints.

1. Unique Name

Team, player, and game names should remain unique. The program should be designed to prevent naming conflicts.

1. Singleton Instance

Only one instance of the game service being active in memory at any given time is required. The application should implement singleton data structure to support the feature.

## [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 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.**

The UML diagram above shows the classes that exist in “Draw It or Lose It” application. The ProgramDriver class contains the main() function of the application. This class serves as the entry point of the application. ProgramDriver class also uses SingletonTester class, which ensures only one instance of the GameService is created at one time. By being entry point, ProgramDriver is responsible for adding games, teams, and players using the instance. Each subclass will have Id and name attributes. Additionally, default constructor and constructor with parameter id and name are declared to ensure proper Entity class is created. Accessors such as getId and getName is declared in Entity class, which can be accessed by subclasses. Each subclass Game, Team, and Player have their own constructor with parameters so each attribute can be created to their specific classes.

Game has list of Team attribute and has 0 to many connections with Team class. Game class also has addTeam method, which will add a Team variable to the list. Team class has similar structure as Game class. It consists of list of Player attribute and has 0 to many relationships with Player class. It also has addPlayer method, which will add a Player variable to the list. Player class, however, does not have any attribute besides from inherited attribute of id and name from Entity class. All classes have overridden toString method to print the data when asked.

The GameService class has a composition relationship with the Game class. GameService class can manage the Game instances. By having list of Game class, GameService has access to multiple Game objects.

This model shows several concepts of OOP by encapsulation, polymorphism, abstraction, and inheritance. Entity is an abstract class that cannot be instantiated. Game, Team, and Player class are extended from Entity, which demonstrates the inheritance. Entity, Game, Team, and Player class have overloaded toString method, which demonstrates the polymorphism. These concepts of OOP enhance code readability and reduce the code complexity by removing duplicate codes. The attributes in each class are declared as private attribute, which demonstrates the encapsulation. The attributes can be accessed by accessors which are publicly declared. Encapsulation provides the privacy and security desired for the system.

.

## [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** | UNIX based environment. Developer friendly environment. Web hosting applications can be supported stable and secure. Higher cost for hardware. Limited scalability and compatibility compared to another environment. | Open-source operating system. Highly scalable and able to support various range of tools and software. Limits to GUI and hardware. | Biggest advantage is compatibility and developer ecosystem. Various hardware is supported. As it is more compatible, it is more open to security and vulnerabilities. | Mobility. Touch screen. User inputs are not keyboard and mouse but interacted with touch. Various hardware capabilities. Different environment may need to be supported such as iOS and Android. |
| **Client Side** | User friendly interfaces. Expertise required to develop and maintain multiple clients. It also increases costs. | Open-source environment, so it is free to use. Using the operating system has steep learning curve, so expertise is needed. | Higher licensing cost compared to other open-source operating systems. | Mobile devices often connect with wireless connection. Unique design needed to support unique input touch system. |
| **Development Tools** | Node.js, JavaScript. IDEs such as VSCode and Xcode are used. | IDEs such as VSCode, Atom, and Sublime Text are used. Command line interface | Popular frameworks are C# and .NET for Windows-based web applications.  IDEs such as Visual Studio and JetBrains are used. | Kotlin, Swift, Objective-C, Java, and JavaScript. |

## 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**: In order to support various devices, web-based operating platform is recommended. Windows can be an optimal operating system for the application as it supports various range of hardware and have highest compatibility among the other operating platforms. Adopting a web-based environment in Windows operating system will allow multiple devices to access the application, which can result in more customers.
2. **Operating Systems Architectures**: The architecture of the application will be based on client-server architecture in web-based environment. HTML, CSS, and JavaScript will be used to server client-side web environment. Multiple architecture should be implemented in the server side to cover data structure and application side.
3. **Storage Management**: This data structure does not need a flexible data structure. A noSQL or non-relational database is not needed. Relational database system such as SQL can be used. The system can be implemented in cloud-server to store the images and user data. Cloud-server supports high scalability and accessibility across various platforms.
4. **Memory Management**: The operating system utilizes the memory management system provided by web browsers. Microsoft edge, Chrome, and other modern browsers support garbage collection to handle memory management such as allocation and deallocation.
5. **Distributed Systems and Networks**: An implementation of cloud-based server can serve as a central point of system. Modern cloud service providers such as AWS support distributed systems in the cloud model. The synchronization can be handled inside the server, which enables rea-time updates and exchange between players across multiple devices. Error handling method inside the cloud model can be set up to ensure clients have stable connectivity.
6. **Security**: Secure protocols must be used in all data transmissions such as TLS, HTTPS, and VPNs. Using VPN will create private instance for networking users to ensure the security of the data. Encryption method such as AES-256 should be used to protect end data. Authentication and authorization method should be established on every user access to ensure authorized access. To enhance the security measure, user can be asked with one-time passcode (known as OTP).