

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 | 03/24/24 | Michael LaBrutte | Breaking down information for the Draw It or Lose It application development |

**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)

Draw It or Lose It is a game that is loosely like the 1980s television game *Win, Lose or Draw*, where teams compete to guess what is being drawn. Instead of players drawing images on an easel to guess what the puzzle (a phrase, title, or thing), the application will render images from a large library of stock images to present as clues.

## Requirements

* The game must have the ability to have one or more teams involved.
* Each team must have the ability to have multiple players assigned to them.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. Must create unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

* In a web-based distributed environment, we need to account for the game application being developed for a web-based platform. We need to account for the constraints when it comes to security, communication between the servers and the compatibility between the different browsers available.
* Ensure the game provides a single instance of the game service being active in memory at any given time.
* Enforcing unique names of game, team, and player names to prevent any conflicts while maintaining an optimal user experience.

## [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)

When we look at the diagram, we can see that the Entity class serves as the base superclass for all the entities within the program. It contains attributes such as ‘id’ and ‘name’ that are shared by all the subclasses. This identifies that the Entity will have an identifier and a name. Game, Team, and Player are subclasses that extend to the Entity superclass. The game consists of multiple Teams, and each Team consists of multiple players. The GameService class has a composition relationship with the Game class, while the Team class has a composition relationship with the Player class. The ProgramDriver class holds the main function and serves as the entry point of the application. The ProgramDriver also has SingletonTester that indicates that there will be only one instance of the GameService class throughout the application. Additionally, the ProgramDriver class has a dependency on the SingletonTester class since it has <<uses>> arrow indication. The diagram showcases several object-oriented principles. The diagram showcases the inheritance relationship between the Entity superclass and its subclasses, Game, Team, and Player. This shows that they inherit attributes and behaviors from Entity. Lastly, abstraction is shown by the way the classes manage the Game, Team, and Players for implementation of details for the players interaction.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | **Characteristics**  Mac hosted website will run on any operating system, flexible terminal commands  **Advantages**  User friendly, high security, integrates with Xcode server  **Weakness**  Less preferred for web hosting services | **Characteristics**  Secured, most preferred  **Advantages**  Free and open source, the ability to customize security features, supports multiple coding languages  **Weakness**  Learning curve for software utilization, lack of pre-built machines with Linux capabilities | **Characteristics**  Well rounded operating systems, dominant compared to other platforms  **Advantages**  Supports a magnitude of different applications, easy to provide updates  **Weakness**  Large user error potential, security protocols are lacking which makes it susceptible to malware, spyware etc. | **Characteristics**  Easily accessible to a wider audience, no server necessary for native apps  **Advantages**  Low-cost web servers, no server necessary for native apps  **Weakness**  Hosting options are cloud based which can leave companies to be more vulnerable to hacking, rigorous approval process for applications on to the market |
| **Client Side** | Supporting multiple types of clients on Mac would require additional development efforts and cost due to the platform-specific functions required. The amount of time that would be needed to develop on this operating system would be determined by the expertise level of the developer | Supporting multiple types of clients on Linux would require additional development efforts and cost due to the platform-specific functions required. Additionally, the developer would need a wide variety of knowledge when utilizing this operating system due to the limited use | The price range would vary depending on the needs of the project and the accessibility with multiple applications. Time would be dependent on what the client needs, and the developer would need a moderate level of expertise due to the common accessibility of the windows operating system | Developing cost would be moderate depending on the knowledge of the developer. Average to long time depending on what phone operating system that the user is developing on due to the requirement constraints. Developer would need to be knowledge on multiple phone operating systems and the functionality of all platforms |
| **Development Tools** | When utilizing Mac, we can use SWIFT while also using notepad++. We can also utilize HTML/CSS/JavaScript while utilizing supporting libraries like Java, Python, PHP | When utilizing Linux, we can utilize visual studio eclipse, along with notepad++. Linux can also utilize HTML/CSS/JavaScript while utilizing supporting libraries like Java, Python, and PHP | When utilizing Microsoft, you can also utilize visual studio eclipse to utilize multiple coding languages to develop. HTML/CSS/JavaScript can be utilizing while also using supporting libraries like Java, Python, and PHP | When it comes to developing on mobile devices, you will need to utilize platform-specific programs like Android Studio and Xcode. |

## 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**: I would recommend the Windows operating system for this project due to its optimal ability to integrate with the current Android application and Windows large userbase. Due to this, we will be able to find a wide variety of developers that can utilize their skillsets to ensure the most efficient development for a cross-platform application at a good price.
2. **Operating Systems Architectures**: I would utilize a hybrid architecture, for example, Microsoft Windows NT kernel. This would provide a layered approach that would streamline the system’s complexity and enhance its manageability for future updates. Additionally, the hybrid architecture provides flexibility to expand the application while still maintaining security and protection of information.
3. **Storage Management**: Since we recommended Windows, I recommend Microsoft Azure due to its pricing. It also overs readily available customer support, and consistent updates, while providing reliable store management. Additionally, Azure offers multiple scaling options for storage while being cost efficient.
4. **Memory Management**: I recommend Windows due to the variety of storage and memory management options. The operating system includes virtual and physical space for memory allocation such as OneDrive, Visual Studio, Azure Cloud etc. Utilizing industry standard best practices when developing this project will help mitigate runtime and memory allocation issues. Windows will allow for continuous updates and mitigation for memory allocation when expanding about the product.
5. **Distributed Systems and Networks**: To enable communications between various platforms, the game can utilize distributed software architecture and leverage network connectivity. Additionally, develop platform-specific client applications based on the platform to allow for communicating with the server using the chosen protocol while providing a consistent user experience.
6. **Security**: To protect user information on and between various platforms, we would need to ensure that several security measures can be implemented. We need to ensure secure communication protocols between the operating platforms, user authentication and authorization mechanisms that allow for username and password authentication. Also, allowing for data encryption techniques to safeguard sensitive user information from being saved or transferred between networks.