

Draw It or Lose It! Application.

# **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 |
| --- | --- | --- | --- |
| 1.1 | 3/23/2024 | Alexander Feeney | Includes additional technical game 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)

Draw It or Lose It is a game based on a classic television game series “Win, Lose, or Draw”. This application will have team-gameplay and a unique and strategic guessing-game element. This fun and interactive game will surely captivate the players and ensure a thrilling experience. This game features twists and turns to the original series, enhancing the experience with fresh modern experience.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

A design is necessary for the way we are going to hand the front-end work. For now, the back-end logic will be written purely in Java. Our game will have to structurally be made to handle more and more players, so we will have to focus on scalability as well as modularity for ongoing changes to the codebase as we maintain the application. User experience is a top priority, so we have to ensure the application has a clean and easy to use interface. Security is also a major threat, so we will ensure the code is written securely and intricately structured.

## [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 GameService class serves as the median of the model. It manages instances of the Game class, implementing the singleton design. This directly promotes code efficiency and avoids duplication. The Game class is an actual instance of a game being played. It also has association to Team entities. This utilizes the one-to-many relationship. The addition of the entity class makes game, team, and player classes all incorporate code encapsulation. Overall, the utilization of the singleton design pattern, structured design, and the incorporation of encapsulation and inheritance, and efficiently structure the model and all its relationships.

**"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** | Mac is very secure and has great architecture and integrates well with Apple products, but it is more limited in hardware options. Less common route for server hosting. Moderate licensing costs. | Linux is very well known for its stability and security. It is extremely reliable. One downside is its complexity. Robust support for servers. Licensing costs are low. | Extremely user friendly. Its support for Microsoft is fantastic. Windows is going to want to update every 10 minutes on you. Moderate to high licensing costs and server. | Portability is unrivaled with mobile devices. Phones also have features and hardware to utilize like a camera. Mobile development, on the other hand, is very complex. Low to high costs depending on service. |
| **Client Side** | Still slightly lower costs than other options for deployment. Specialized development environments are a plus and would also make deployment much faster for Mac end-users. | Linux is open source and has plenty of libraries available. It is also heavily documented. You need an extensive skillset and a lot of training to use Linux. | Windows has an abundance of tools and frameworks. You would have extensive documentation available as well. Some expertise needed in Windows environment. | Costs are considered when dealing with mobile development, as they can vary. Compatibility-wise, it takes a lot of time to make sure it is well optimized for different devices. Familiarity with Android-specific technology is required. |
| **Development Tools** | Primarily Swift and C are used with Apple. Interface Builder is used for mobile development. Moderate cost for tools. | Linux utilizes a wide variety of languages and tools, from Java to Ruby. It also embraces Eclipse. Costs tend to be on the lower end compared to other platforms. | Also encompasses Java. Also, many frameworks are available to aid in building windows applications. Moderate to high cost for tools and additional fees from Microsoft. | Mobile development is extremely specific. It has a lot of tools and frameworks specific to the platform. However, there are many frameworks available for targeting both android and iOS. Moderate costs for mobile dev tools. |

## 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 recommend using Microsoft Azure for hosting the game. It’s scalability for expanding our game to other environments and has great features to help in development. This platform’s scalability caters to user demands as well as other requirements. Microsoft’s integration with different tools such as visual studio also makes the development process much easier. Azure’s Kubernetes Service can be used to deploy and manage the game’s infrastructure. Additionally, the platform’s robust security infrastructure will ensure our user’s data is protected and secure.
2. **Operating Systems Architectures**: Microsoft Azure’s architecture has a very reliable network infrastructure. Its data locations are strategically spread out globally, allowing for low-latency and great performance. The Azure Virtual Network also helps promote good connection and better performance by providing dedicated network resources for game traffic. It supports a range of operating systems like Windows and Linux to ensure cross-platform compatibility. The deployment will be very straightforward due to its user-friendly interface and variety of integration tools.
3. **Storage Management**: Azure SQL Database is extremely reliable and secure. If we need to handle more data at any point, then the platform will be able to do so. My recommendation would be to utilize Azure’s Blob Storage system. With this, data can be stored securely as well as hierarchically. For example, data can be stored in multiple tiers based on its usage and requirements. This will be important to handle high volumes of data.
4. **Memory Management**: Virtual machines and containers are used for memory management. This means the application will have great performance and excellent scaling capabilities. When these tools are utilized, memory resources will be allocated appropriately to ensure efficiency. This helps spread the workload out evenly and consistently to prioritize performance.
5. **Distributed Systems and Networks**: Azure has distributed systems to provide a strong network infrastructure to work seamlessly across different platforms. Azure distributes their data centers to cover large areas and allow consistent connection regardless of geographical location. This means its global presence ensures low latency regardless of where the player is located. Azure Traffic Manager has great balancing capabilities by distributing player traffic accordingly and optimally to ensure performance. This adds to performance as well as gains resistance against network outages because of the wide regions of server accessibility.
6. **Security**: This is a top priority. With Azure’s encryption capabilities data will be secured efficiently. Real-time security management and monitoring will be available with the Azure Security Center. Its advanced active directory security features and proper access controls make it a strong and robust choice. Azure is continuously updated, allowing for new and better features to be added consistently over time. Azure also implements the power of machine learning and AI to detect security threats, such as different algorithms that analyze data and different machine learning models that identify patterns that could be a threat. Overall, this will be a great choice for our requirements.