

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 | 01/26/24 | Aneudi Matos | Added the summaries and design constraints |
| 1.1 | 01/27/24 | Aneudi Matos | Added development requirements for different OS |
| 1.2 | 02/25/24 | Aneudi Matos | Edited development 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 wants to develop a web-based game called Draw It or Lose It. This game will require creation and management of multiple teams with unique names and players. Also, only one instance of the game must exist in memory at any given time while maintaining scalability.

## Requirements

The business requirements are to launch this game as a web-based and make it compatible and available on different web browsers. This can ensure that it can be enjoyed by many potential users. The program should perform efficiently and be scalable. The technical requirement would be to make sure the game can pull from the database of the stock images. Also, the creation of unique names and teams so players can identify themselves and their team easily. And to make sure only a single instance of the game exists in memory.

## [Design Constraints](#_2et92p0)

Some design constraints could include making sure that the web-based game works well on different browsers, ensuring a robust user experience. Also in distributed systems latency can be an issue because of the differences in network performance from each player which affects responsiveness. As mentioned before, another constraint is the web browser compatibility which limits the game design to widely available web technology. The game must be compliant with HTML5 and CSS3.

## [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)

In the UML diagram we see encapsulation in the GameService class where we have private static class attributes. Also, present is the private GameService method, which is the singleton design pattern, this ensures that only one instance of game exists in memory. The entity class is a base class for Game, Team and player showing inheritance and they all share id and name and the id and name fulfill the requirements of unique names. There are multiple examples of association by the 0…\* showing that more than 0 teams and players. Polymorphism is shown by inheritance where methods from Game, Team and player can be overridden. Abstraction by the entity class where the ID management is hidden.

**"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 must work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

**Server Side:**

**Client Side:**

**Development Tools:**

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Consistent user experience, easily integrated with other apple hardware, secure, more expensive hardware cost. | Open-source, highly customizable, stable, and reliable, strong community support, widely used in server apps with lots of documentation. | Has its own server versions, lots of users are familiar with windows, active directory to easily integrate with other Microsoft products, not ++as secure more targeted by malware. | Most widely used are android or IOS, Uses ARM architecture not x86, compact and portable, lots of development tools, not typically used for servers, less computing power than servers. |
| **Client Side** | If the hardware is not already owned this could prove costly, will need to be tested for use in safari, if standard web practices are used testing time can be reduced between other browsers. | Hardware can be cheaper than Mac, will need to be tested for use in safari, chrome, and Firefox, if standard web practices are used testing time can be reduced between other browsers. Most tools needed are open source and free. | Hardware can be cheaper than Mac, will need to be tested for use in safari, chrome, and Firefox and edge if standard web practices are used testing time can be reduced between other browsers. | The initial investment in different OS and hardware with different size screens can be a problem, optimizing for the network speed and data usage, perfecting the touch functionality can use up more time and resources. |
| **Development Tools** | For the front end Javascript, HTML, CSS as for backend Node.js, Ruby, python, and PHP. IDEs and tool for mac Visual Studio code, Xcode, and safari developer tools. | For the front end Javascript, HTML, CSS as for backend Node.js, Ruby, python, and PHP. For IDE Visual studio code, android studio for web apps on mobile, developer tools for browsers such as Firefox and chrome. | For the front end Javascript, HTML, CSS. As for Backend .NET, Node.js, python and PHP. For IDE visual studio code and Microsoft edge developer tools. | Javascript, HTML, CSS with a combination React Native. For IDE and tools Visual studio code or android studio and tools like chrome developer tools and safari developer 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**: After all the analysis and consideration of the operating systems, the one recommended is windows, for its cost efficiency, familiarity, community support. Also, it would be easy to integrate with other Microsoft products like Microsoft azure for server hosting.
2. **Operating Systems Architectures**: Windows offers the advantage of a hybrid kernel having a user mode and a kernel mode which boasts reliability and the kernel providing threading, memory management and network stack.
3. **Storage Management**: As mentioned above Microsoft azure can be deployed for hosting and cloud storage making the scalability of storage easy and fast.
4. **Memory Management**: Widows systems are efficient at handling memory, a 64-bit windows system can where half of the memory gets split between the kernel and user programs. Which makes a windows system able to multitask by prioritizing important processes.
5. **Distributed Systems and Networks**: Fault tolerance and load balancing are strategies suggested to handle server errors and failures. With load balancing the workload the requests are distributed to optimize the use of resources. Also fault tolerance uses recovery and backup methods to restore to keep the system operating in the event of an error or failure.
6. **Security**: Role based security would work perfectly for this game because it can separate admins, players, and teams from accessing things they do not need to access.