

<The Gaming Room>

# **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 | 03/20/2023 | Angelica Randall | Summary description, design constraints |

**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 loosely similar to the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. However, the application will render images from an extensive library of stock drawings as clues rather than a player drawing images on an easel to help team members think about the puzzle (a phrase, title, or thing). A game consists of four rounds of play lasting one minute each. Graphics are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## Requirements

*The client requires that the game can have one or more teams involved. Each team should be able to have multiple players assigned to it. All game and team names must be unique to allow users to check whether a name is in use when choosing a team name. Also, only one instance of the game must exist in memory at any given time.*

## [Design Constraints](#_2et92p0)

## The constraints are as follows:

Thedevelopmentof the application needs toensurecompatibility with modern web browserswhile maintaining a responsive design for various screen sizes**.** It must alsoimplementthe proper security measures to protect user data. These constraints will impact the development process by requiring certain technologies and techniques to ensure compatibility and security.

## [Domain Model](#_8h2ehzxfam4o)

An example of the Object-Oriented principles present in the game is as follows:

The “Game,” “Team,” and “Player” classes exhibit the principle of inheritance because these attributes are also listed in the "Entity" class. This allows the other child classes to inherit those attributes as well. Another Object-oriented principle expressed in the diagram is encapsulation; "Entity" protects data applied to the program by limiting access to public methods. This, in turn, protects the privacy of each player’s information. Entity is the super-class or Parent class, while Game Service, Game, Team, and Player are the child classes that inherit from it. The singleton pattern is also present, with only one instance of each 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) and 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** | It is Expensive but includes GUI allowing ease of use for developers and other users. Flexible terminal commands. Popular characteristics in web hosting.  Has a built-in web server with easy setup and use.  The latest Macs have built-in Apple T2 security chips—low licensing costs.  To scale thousands of players, have multiple load-balanced servers for the same zone.  Moderate licensing costs for OS X Server. | This option is the most cost-effective. Needs more intuitive site navigation. Includes a command shell for simple server configuration as well as accessibility.  Has a built-in server on Linux open-source operating system. Considered a cost-effective option.  To scale, thousands of players have multiple load-balanced servers for the same zone.  Licensing costs vary for the time needed. | User a friendly platform but could be more cost-effective.  Deployment handled by (WDS) Windows Deployment Services. It gives administrators the ability to deploy remotely. In addition, it can be used for network-based installations—very cost-effective licensing.  To scale thousands of players have multiple load-balanced servers for the same zone.  High licensing costs. | Most inexpensive, with device specifications varying from user to user. This will present a challenge while seeking to be compatible with all other platforms.  Deployment can be done through the mobile backend as a Service. (BaaS), (Paas), or (IaaS)  It could be costly to develop and maintain over time.  To scale thousands of players have multiple load-balanced servers for the same zone.  Affordable licensing costs. |
| **Client Side** | Moderate expertise and time are required for the development. Not cost-effective.  It must be developed with HTML responsive web design to operate on all platforms. | This requires minimum cost but the most expertise for the development.  It must be developed with HTML responsive web design to operate on all platforms.  Ample development time and Max proficiency are needed. | Minimal time and proficiency are required to support. Cost similar to that of Mac.  It must be developed with HTML responsive web design to operate on all platforms.  Short development time for a simple app, longer for more complex apps. Very Costly. | The most time-consuming.  It must be developed with HTML responsive web design to operate on all platforms.  High development costs. Short development time. High proficiency is needed. |
| **Development Tools** | The tools and languages needed to execute are as follows:  JavaScript, Python, CSS, HTML. IDEs would be VS code and Eclipse.  Multiple teams are required for the development.  Low-Cost licensing. | The tools and languages needed to execute are as follows:  Ruby on Rails, Java, Python, CSS, JS.  Tools would be VS, GitHub, node.js, and command line.  Multiple teams are required for the development.  Costly licensing. | The tools and languages needed to execute are as follows:  Ruby on Rails, Java, Python, C++, C#. VS, eclipse, Repl. It, command line  Multiple teams are required for the development.  Low-Cost licensing. | The tools and languages needed to execute are as follows:  Android, Swift, HTML, JS, Python, Ruby, CSS  Multiple teams are required for the development.  Licensing Costs vary according to the type of application. |

## 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 developing the app on the Windows platform to allow ease of use for the users and access to several different software packages. Many of the applications can also be used on other operating platforms. To meet the expectations of the game, a cloud-based server platform will be the best option. It will provide scalability and flexibility and is the most cost-effective option for continued development. A few platform examples are Amazon Web Services, Microsoft Azure, Google Cloud, and Windows 365. Their broad range of services can be used for storage, computing, and networking.
2. **Operating Systems Architectures**: Windows offers ease of use with its intuitive GUI design. Developers also have more resources available, as well as the use of the command prompt for server configuration. I recommend a multi-tier architecture that separates the application’s logic, data management, and user interface into separate layers. It provides the scalability, flexibility, and security needed for this application.
3. **Storage Management**: Windows offers cloud storage abilities that give developers adequate resources for allocation. An additional recommendation is a relational database management system, which provides the ability to store and retrieve data in a structured and efficient manner. Other features include data backup, recovery, and security.
4. **Memory Management**: Memory management includes physical and virtual address spaces, which allow up to four gigabytes of memory. Virtual memory management techniques should be used in development. This will enable more memory than is physically available by storing the data in a hard drive or some secondary storage. This will optimize memory usage and improve application performance.
5. **Distributed Systems and Networks**: Distributed systems and networks can have joint issues such as queuing, routing, and congestion problems. These systems offer accessible communication and coordination between one another. There is an IDE named Develop four that allows cross-platform development that would be an excellent resource for this application. You will also need to ensure the servers can handle many players and have backups in place in case of outages. RESTful APIs and web services will enable data transfer and connectivity.
6. **Security**: Windows comes with built-in security measures, but to ensure the protection of your users, there should be additional security in place. Input validation, data encryption, and secure coding practices must be implemented in development. SSL/TSL encryption should be used for data transfer and user authentication. I also recommend the implementation of firewalls, intrusion detection systems, and many other security measures to prevent unauthorized access and ensure user data protection. A role-based security system would also suffice, making it so a user cannot access unauthorized information.