

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 | 05/26/2024 | Sarah Giannini | The executive summary, design constraints, and domain model for The Gaming Room program. |
| 1.0 | 06/09/2024 | Sarah Giannini | Evaluation table. |
| 1.0 | 06/23/2024 | Sarah Giannini | Recommendations |

**Instructions 6/30/20246/30/2024**

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 is a new client that wants to hire CTS to develop a web game that is supported by multiple platforms. It will be based off their game that is called "Draw It or Lose It" and is only on Android app right now. The game will have users play on teams that will compete by guessing what is being drawn from a library of stock images.

## Requirements

* The game will support multiple teams.
* The teams will support multiple players.
* Each game and team name must be unique, letting users check if names are available.
* Only one instance of each game can exist at any time.

## [Design Constraints](#_2et92p0)

* The web game must be able to scale to multiple team and player amounts.
* With only one instance being allowed at a time, multiple people must be able to access that one instance at the same time.
* The Gaming Room wants it to be supported by multiple platforms, which means it must allow cross-platform gameplay.
* Team and game names must be stored to keep each of them unique and secure.

## [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 UML diagram displays the classes that are in the Draw It or Lose It app. The base class for the other classes is the Entity class. The subclasses Game, Team, and Player all inherit from the Entity class, meaning they will share the attributes from the Entity class. The GameService class ensures there will only be one instance at any given time and contains the methods for creating the games. The classes also demonstrate encapsulation by storing data privately, such as lists for teams or players, with public methods. There is also association between the classes, as players are in the teams, and teams are in the game.

**"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 not very commonly used for hosting web-based software applications. It would work very well between all the apple products, but they tend to be more expensive. | Linux is a very commonly used operating platform for hosting web-based software. It is also free, unlike Mac. It requires more to set it up and maintain it, but there is a large community of Linux users that can help with this. | Windows is common within businesses. It has strong software support and security. There is not as much to set up and maintain compared to Linux. A downside is that it is not as customizable as Linux and is more expensive. | Mobile devices do not host web-based software. |
| **Client Side** | The developer would need to be familiar with macOS and iOS. The hardware cost would be high initially. | Linux can support a lot of systems, while being very low cost. It may take more time or people with more knowledge to set up the software. | Windows supports many systems, but still requires developers to have knowledge. It can be costly for developing certain things. | Mobile devices would need developers to create a web app that works with iOS and Androids. It would require a lot of knowledge of cross platform development and can be costly. |
| **Development Tools** | You must be using Mac hardware. Relevant languages include Swift, JavaScript, html, Python, and Ruby. Xcode is a popular tool that is used with Mac. | Linux requires more maintenance than other software. Popular languages used are html, CSS, Python, C++, and C. A common tool used is Docker, and IDE's used are eclipse and VS code. | Windows has Visual Studio, which is an IDE that has a lot of tools and supports multiple languages. Relevant languages are Python, C++, C#, html, JavaScript, and SQL. | Mobile developers commonly use Xcode and Android Studio. Relevant languages used are JavaScript, Java, Swift, and Dart. |

## 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**:

For the expansion of Draw It or Lose It to other computing environments, my recommendation would be Windows. Windows is popular, making it familiar to most users and developers. It offers much support for a variety of development tools and IDEs, helping developers and keeping costs lower. The Windows ecosystem also allows for the app to have integration and expansion across other computing environments, such as iOS and Android for mobile, web browsers like chrome, and Linux and Mac.

1. **Operating Systems Architectures**:

For operating systems, the Windows architecture is a viable choice for Draw It or Lose It because of its modular design and strong support it has for multitasking. It provides comprehensive APIs and system services, allowing efficient use of resources like CPU and memory. Windows also ensures stability and security for users with built-in features for errors.

1. **Storage Management**:

A good system for storage management would be to utilize a scalable cloud storage option like Google Cloud Storage. This would provide flexibility, reliability, and scalability that would then help with managing data while expanding the app Draw It or Lose It across multiple platforms. Cloud storage would reduce the need for certain hardware investments, and it would allow data storage to be easy to manage. This will also help it work efficiently throughout the different platforms.

1. **Memory Management**:

Windows is a good operating platform for memory management. It uses dynamic memory allocation and virtual memory management. These methods of memory management optimize the memory usage of the app while it is running and allow for the overall performance of the game to be better.

1. **Distributed Systems and Networks**:

To make sure that Draw It or Lose It can communicate across various platforms, communicating through microservices like RESTful APIs and message queues can be used. Using both services together will ensure effective communication across the different platforms that the app will expand to and its users, reducing chances of connectivity issues and maintaining the network through any issues.

1. **Security**:

Security protection for the users is extremely important to keep their data safe, especially with the app's expansion between different platforms. Windows has built-in security against malware and threats with Windows Defender to help keep the clients secure. Data transmitted between clients and the server should also be encrypted to prevent any data from being stolen. Google Cloud Storage offers more security features, which include encryption at rest and when data is being transmitted. It also offers role-based access control for those who can access the data. These methods, combined with passwords and double authentication, will all work together to protect user information across the multiple platforms.