

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 09/17/2023 | William Rosales | Added information into regarding design, summary, product model, and recommendations |
| 1.1 | 10/1/2023 | William Rosales | Added information into the requirements area and added into the evaluation of the different operating systems that can run applications. |
| 1.2 | 10/14/2023 | William Rosales | Added information into the recommendations area to cover about different aspects of the background of the web-application and its security and architecture. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is a web-based version of the popular game “Draw It or Lose It” as it is expanding beyond the currently active Android app. This project aims to have multiple teams with multiple players while ensuring isolated names for games and teams. This document will outline the project with its software design and operations on how the game will work.

## Requirements

Unique names for games and teams

Teams with multiple players

Device that will be able to connect to game server

Device that is compatible with the hardware requirements

## [Design Constraints](#_2et92p0)

Ability to be used across different platforms supporting Cross-platform compatibility

Efficient memory and storage management

Security that protects user information

Only one game can be exist at any time

Game names and team names are all unique as the program will not allow multiples of the same name

## [System Architecture View](#_ilbxbyevv6b6)

A UML Class diagram is displayed to showcase the model of the different classes that are incorporated into the project.

## [Domain Model](#_8h2ehzxfam4o)

There are relationships between the classes that are displayed below this summary. The Entity class is an extension of the Game, Team, and Player classes. We identify the id and name variables as super since they will be coming from the extension of the Entity class. There are also references from GameService to Game, Game to Team, and Team to Player. This shows that there are relationships within those classes within the code of the project. This can be thought of as a food chain in which the GameService needs a Game, which a Game needs Teams, and Teams needs Players.

**"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** | Capable to run for hosting and has advantage with ease of use but tends to run at a higher price. | Lower end on the scale of price, but expertise for setup and maintenance is critical. | User-friendly, but can lean on the expensive side but is also very common to use. | Mobile Devices vary in capabilities for server-side use. Specs for mobile devices vary as well. Ideally, mobile devices are not used as a server platform due to limited specs. |
| **Client Side** | Expensive for users as well. Development requires expertise on macOS. The default browser is Safari which is integrated into the Apple software. Safari has its own cloud utilization tools as it is independent from other cloud servers. Google Chrome and Mozilla Firefox are two other browsers that are common to use on a Mac software. | Requires high expertise in Linux development and can be a time consumer as it is not as user-friendly. Mozilla Firefox is the top choice for Linux users. Firefox allows for open-source work and is very easy to customize to your own needs. Google Chrome is also used with Linux, but Chromium is the open-source browser that is very similar to Google Chrome. | Easier to learn due to the exposure of Windows OS systems used around the world. Expertise is at a minimum. Windows normally has Microsoft Edge pre-installed with the software, but other programs like Google Chrome, Mozilla Firefox, Opera are all very popular. Google Chrome is probably the most popular since it allows for high customization to the user interface. | Development varies from platform on iOS and Android. Also supports access to web-applications through browser applications. For example, Safari and Google Chrome. On iOS platforms, Safari is the primary web browser with many more search engines available in the App Store. Android platforms default to Google Chrome since it is pre-installed. Other web browsers are used like Opera, Brave, and Samsung Internet on select phones. |
| **Development Tools** | Utilizes programming languages such as Swift and JavaScript.  Also has programs to assist with app development. | Utilizes programming languages such as HTML, Java, or C++. Has development tools such as Visual Studio Code that are open source. | Utilizes programming languages such as C# or JavaScript. Also has programs to assist with app development. | Uses programming languages specific to its OS, such as iOS (Swift) and Android (Java/Kotlin). |
| **Licensing Costs** | Higher initial cost for hardware and for licensing costs. | Tends to be minimal to no costs for development on Linux. | Licensing costs tend to be high for a Windows server. This can be due to the popularity and user-friendly OS. | iOS and Android both have development costs, but do not tend to be very high. |

For application development for both iOS and Android, it is common for companies to create their applications through another OS. Mac users can use a program called Xcode that has certain features and development tools that makes it easier for the developer to create an application. Mobile devices usually lack the potential hardware to be able to create an application. This is why most development is done on a system that has better processing hardware.

## 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**: We recommend a Windows-based platform for the Draw It or Lose It web application. Windows is more user-friendly, highly compatible, and will allow for expansion as well. Windows is also a widely used operating system that already has a large user base. Accessibility to users will be easier due to the large range of players the game will attract.
2. **Operating Systems Architectures**: As an introduction to a new game, the ease of development on the Windows operating platform will allow for consistent performance and will allow a great number of clients to have access to the game. It is also important to begin early development for compatibility to run on mobile devices and tablets. This will help reach a wider audience since we are expanding beyond just Windows and Mac users.
3. **Storage Management**: Cloud-based storage with Windows servers may be the best option as it won’t call for any physical storage. This will allow for scalability in the application as the user count or game count continues to increase. As the game gets bigger and the user count continues to increase, the system will need to be able to adapt to the number changes. Using cloud storage will also be able to store data in the most efficient way considering that we will be targeting multiple platforms all at once.
4. **Memory Management**: The Windows platform allows many features that are already implemented into the OS to review application load on your hardware. Using these features, performance will be monitored in a user-friendly way in which developers and end users can understand how much memory the application is using. Being able to review and test the usage of memory on a system will allow the Gaming Room to either add or remove certain features to a game. This can be a matter of too much memory workload on a game, or room to expand into a game with new features and designs. Using the Windows software will also allow developers to see memory usage in live time since the software comes with a Task Manager. This will promote resources allocation and better performance.
5. **Distributed Systems and Networks**: Interactions systems within the game using APIs. This allows players to interact with one another while they are on their teams. This will make teams that are made up of players that are not physically together still remain in a team and play together. Cross-compatibility will be able to be used when communicating between different types of devices. Cross-platform testing will ensure that the user experience is not being affected and that the game is operating smoothly regardless of the platform the user is playing on. Another feature for the application could be the implementation to switch from online to offline mode if connection is lost during the play of the game. It could be beneficial to the user if they were able to still remain in the game if they lost connection for a temporary amount of time.
6. **Security**: The Windows OS has security features already implemented which protect user information by securing the user’s access. Windows Defender and Windows Firewall are just two examples of security measures that come with the Windows OS. Additional security measures would not hurt the game process either as a second verification for the user. This is common practice when it comes to connecting players remotely to other players on a network. Securing APIs with token-based authentication will allow users to secure their account on an application that is being used around various platforms and connecting with those users. It is also important to add expirations to these tokens so that the user has to refresh a new token in order to keep their account secure. Remaining with the same token for long periods of time can result in data breaches to the user’s account and possibly to other unauthorized access in the game.