

Draw It or Lose It

# **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 | 05/26/24 | Kameron Coolong | Changes were made to the cover, document revision history, executive summary, design constraints, system architecture view, domain model, and recommendation. |
| 1.1 | 06/07/24 | Kameron Coolong | Changes were made to the evaluation table. Added characteristics of each platform for server side, client side, and development tools. |
| 1.2 | 06/17/2024 | Kameron Coolong | Changes were made in the recommendation section of this document. |

## [Executive Summary](#_sbfa50wo7nsh)

The client has come to us asking to create a design template and start making the game “Draw It or Lose It.” They want help creating the environment for the game and to streamline the development. The game will need to be cross platform. The game will also need to allow one or more teams, with multiple people assigned to each team. They also ask that only one game instance is in memory at a time.

## Requirements

Each game requires one or more teams, each with multiple players assigned. One instance of a game should exist at a time. This should be done using unique identifiers for games, and team names, ensuring the uniqueness of both. The users will need the ability to check if a name is already in use.

## [Design Constraints](#_2et92p0)

## The biggest constraint for this project will be making this all cross-platform. We need to implement a way to manage each game so that they are unique and only one instance of a game is running in memory at a time.

## [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 entity class serves as the parent class for Game, Team, and Player classes. This means that the Game, Team, and Player classes all inherit the attributes from the Entity class. The Game service class will be used to incorporate the requirements that the client has specified about the uniqueness of the games and team names as well as player names. The Game class will be done by managing a Teams list. This class also contains a constructor, a method to add a team to the game, and a helper method. The team classes will contain Player lists. It will also contain a constructor, a method to add players to the team list, and a helper method. The player class only contains a constructor and a helper function.

**"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** | Mac is very expensive to run as it comes with expensive licensing options, and expensive hardware options as well. However, apple has discontinued macOS server in 2022, and will likely be discontinuing the macOS Monterey in late 2024 as well. | Linux is open source, and very cheap with almost no hardware limitations. Linux is very versatile with tons of open-source tools premade which can make setting up a webhost very easy. | Windows is also very expensive. Windows offers Azure, a web hosting platform with multiple services for storage, database and networking. You can also make your own webhosting platform on windows. | Mobile devices are very cheap. You can host a webserver on a mobile device by using the following open source software:  Termux, a terminal emulator, hackers keyboard, allows you to use special keys like ctrl and alt, a webserver such as node.js http-server, and OpenSSH client. |
| **Client Side** | MacOS is easy to use. To access the web server, you can use any supported browser available, and this would mean having a REST API that can handle requests from a browser, | Linux has many web browsers available that can be used to access the website and access a REST API that can be used to interact with the server side. | Windows also has web browsers available that can be used to access the website. A REST API can be used to interact with the webserver. | On mobile devices, you would need to make two mobile clients on IOS and Android devices that can use the REST API as well. This will ensure that across all platforms, the same server backend can be used. There are also ways to make a client that can be used for both IOS and android such as flutter and react native. |
| **Development Tools** | When creating the webapp, you won't need to worry about the requirements of the operating system as you will simply use the available browser on that platform. The server could be set up on any three of these platforms (mac, Linux, windows) and accessed on any three. | As with mac, if you have a compatible browser installed, all you need to do is navigate to the website. The same team can develop browser-based applications for all three platforms. | Again, each of the browsers on mac, Linux, and windows should be able to access the web app with no compatibility issues. | To develop the webapp for the flowing platform would require a framework to be used such as flutter or react native that can create a cross-platform application capable of communicating with the web server via REST API. |

## Recommendations

1. **Operating Platform**: I would recommend windows for this project, as it is a game, and most users will probably be using a mobile app or windows to play. Windows offers great resources to make mobile apps that are built into Microsoft's Visual Studio IDE. This IDE provides many tools and templates that can be used to easily develop and launch a webapp that can be accessed on various platforms.
2. **Operating Systems Architectures**: Windows 10 is a great option as you can develop with Win32 API’s, .NET framework, and windows RT API’s, making it easy to create cross platform applications. These are also easily accessible within the Visual Studio IDE.
3. **Storage Management**: Windows server storage management is the best choice for this project, Windows provides many disk management tools that can be used to clean disk drives, defragment, and even partition drives easily.
4. **Memory Management**: Windows offers additional built-in virtual memory that can help with memory management. Windows also has heap functions and file mapping. This allows the processor to work with memory more effectively, which could be a crucial aspect of this application as this will be memory intensive.
5. **Distributed Systems and Networks**: There are many options for cross-platform frameworks that would streamline the Devlopment and usability of the applications, I would suggest using a client-server distribution system like flutter as it is well known and commonly used. This would allow for apps in macOS, Linux, Windows, IOS, and android to all run and communicate with the server side.
6. **Security**: Using windows, I would recommend paying extra attention to protocols and safe coding practices being used to ensure that the application and server are secure. There are services that can be used to help increase security, however this would increase costs. Windows also comes with windows defender. However, since windows is a very widely used proprietary software, there are many known and unknown vulnerabilities that people can take advantage of. It would be crucial that all back-and-forth traffic between client side and server side are encrypted to prevent man in the middle attacks. It is also crucial that all user data is encrypted before it is stored on our systems. Another action that needs to be taken is strict logging on all functionality on the server, this way if any vulnerabilities were to be exploited, we have a system log that can trace it back to the source and we can patch the bug.