

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 | <03/23/24> | Christopher Manley | Filled Executive Summary, Requirements, Design Constraints, and Domain Model sections. |
| 1.1 | <04/07/24> | Christopher Manley | Filled Evaluation section. |
| 1.2 | <04/21/24> | Christopher Manley | Filled Recommendations section. |

**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 has an Android app that they want to convert to a web-based game. To accomplish this, we will develop a game based on object-oriented programming principles. To this end, classes will be used as objects to facilitate the functions of the game.

## Requirements

A game can have one or more teams playing. A team has multiple players. Game and team names need to be unique, and names need to be checked when choosing a name. Only one instance of the game can exist at a time.

## [Design Constraints](#_2et92p0)

There needs to be a team class to hold team information while participating in a game. A player class will hold individual player information. A name checking function will confirm that a name is available when choosing a name. Finally, games will use the singleton pattern to ensure that only one game exists 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 use of classes demonstrates portability, allowing each class to function as a self-contained object. Game and Player are both associated with Team. All three inherit from the Entity class, which helps eliminate redundancies. Encapsulation is shown through the use of private variables, allowing each class to protect its data.

**"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** | macOS Server is available for a one-time purchase fee. A Mac computer must run the macOS Server, so that will need to be purchased. | Various server hosting options are available on Linux, both free and paid. Linux is generally more flexible than Mac and Windows so scaling for more users may be easier. | Windows offers Internet Information Services for a licensing fee. A Windows compatible device will be needed. | Hosting the mobile app on the Google Play Store requires a one time purchase, while the iOS store charges an annual fee. |
| **Client Side** | Because the application will be web-based, not much alteration needs to be done for each client OS. For Mac, the app should be optimized to take advantage of macOS features, along with being made compatible with the macOS browser, Safari. | The work on Linux is similar to Mac. The app should be optimized for Linux, along with tested on various browsers. | The app should be optimized for Windows, along with being tested on Microsoft Edge to ensure compatibility. | The app is already designed for Android, so that is taken care of, but will need to be modified for iOS. It will need to be compatible with iOS Safari, along with making sure to function on Apple sized mobile devices. |
| **Development Tools** | The Swift programming language is built for Mac and could be helpful. Xcode is an IDE for macOS development that supports Swift. | Linux allows various programming languages that would work, along with options for hosting the app. | C++ is a common language for developing Windows applications, while Visual Studio is an IDE for Windows development. | Similar to Mac, Swift and Xcode are available for iOS development, while the iOS App Store is used for apps. |

## 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 would recommend Windows for the server operating platform due to Window’s accessibility and ability to easily scale for more users.
2. **Operating Systems Architectures**: Windows has a variety of tools available for use, including Visual Studio, Internet Information Services, and support for a multitude of languages.
3. **Storage Management**: Windows’ default file system, New Technology File System, should be more than adequate to handle the storage for Draw It or Lose It, especially with careful management of stored data.
4. **Memory Management**: One important memory management technique will be the removal of unnecessary things from memory once they are no longer needed. Windows also uses virtual memory to mimic the behavior of larger RAM.
5. **Distributed Systems and Networks**: One aspect that will allow Draw It or Lose It to communicate between various platforms is the use of RESTful APIs. The use of the internet will allow clients to communicate with the server and play from anywhere. This obviously necessitates the need for a stable network connection, as the connection to the server can be compromised with a loss of internet connection.
6. **Security**: One security measure in Draw It or Lose It will be an authentication system to prevent data breaches. Other measures, such as the use of roles and encryption techniques will help secure user information as well.