

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/12/22 | Ian Winniwicz | Added information related to the software design of the game Rooms “Draw it or Lose it.” |
| 1.1 | 09/26/22 | Ian Winniwicz | Added information related to server side, client side, and development tools comparing and contrasting the differences between one another. |
| 1.2 | 10/10/22 | Ian Winniwicz | Analyzed the characteristics of and techniques specific to various systems architectures to make a recommendation to The Gaming Room. |

**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 Game Room wants to develop a web-based game called “Draw it or Lose it.” Currently this game is only available on android platforms. The client wants to release their product on all platforms. This document will outline the design constraints along with a UML diagram of the product.

## [Design Constraints](#_2et92p0)

There are four major design constraints within this web-based application. One of them being that the game must have the ability to house one or more teams during play. Another being that each team will need to be able to have multiple players assigned to it. This brings us to our third design constraint; each game and team name must be unique and allow users to check whether a name they chose is in use or not. The fourth constraint being that only one instance of the game can exist in memory at any given time.

As far as development goes one design constraint is the cost of housing an internet application on a website and paying for the domain.

## [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 below shows classes that relate to each other through inheritance, the GameService, Game, Team, and Player classes all inherit from the Entity class. The multiplicity of each of these classes is zero to many. The GameService, Game, Team, and Player classes are also all associated with each other by connection. The ProgramDriver class sends a synchronous message to the SingletonTester class. This is indicated by a solid black arrow.

The object-oriented principles that are demonstrated by this diagram are Encapsulation, Abstraction, Inheritance, and Polymorphism. Encapsulation is shown in this UML diagram by the private and public identifiers in each object, Abstraction is shown by leaving some methods private so that the application only shows the user what is needed for them to see to make the application less complex for the user. Inheritance is show by other objects being created by other objects for example the GameService class creates the Game class, and the Game class inherits from the GameService class. Polymorphism is shown by the Player class because there can be more than one player and the Team class because there can be more than one team.

**"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** | Offers a lightweight HTTPS-based protocol that has low-data traffic impact, this makes Mac good for cloud-hosted programs as well as local-hosted programs. Unfortunately, these are only natively supported on Mac devices (anything running MacOS or iPhone.) | Most popular OS for server hosting, Linode for cloud hosting which is a great alternative to AWS. Obviously can’t use Windows applications or Mac applications. This OS seems to be the preferred choice and is very low cost and doesn’t use a lot of resources.  Uses MySQL   * Open source | Uses specific Microsoft software to run and manage data.  Uses MSSQL, this option is less secure than Linux. | More common on Android devices than on Apple devices. Open source. This option seems to be more expensive than any other option from what I am seeing. Fees to distribute through mobile phones. |
| **Client Side** | This OS is sleek and simple but very costly, doesn’t have native support for everything that Linux or Windows has support for, the user-friendliness of this OS doesn’t make up for what it lacks in comparison to a Unix based OS or Windows. | Developing on Linux is more niche than Mac and Windows. You must be skilled in using the terminal. I would say that the benefit of the cost being low would be outweighed by the time it would take to learn how to navigate this OS. | More expensive than Linux less than Mac, a happy medium. Easy to understand from a client-side perspective. I think this would be the preferred option in my opinion. | This platform can’t perform as well as the other three platforms. You must have a lot of time and be skilled to incorporating a mobile device as it is more difficult than the other three. |
| **Development Tools** | Supports Swift, Objective-C, HTML, CSS, JavaScript, Perl, Python, Ruby, and Java. Has support for Visual Studio as well as VSCode. XCode which is specific to MacOS for developing iPhone, iPad, and Mac applications.  Releasing on the app store for $99 a year | A lot of free opensource alternatives to graphic software that is important to different dev’s. Support for C, C++, C#, Perl,Python,HTML, CSS, JavaScript, Java, Ruby, can’t use languages specific to Mac.  Good IDE’s like VSCode and has Android Studio support unlike Mac. | C, C++, C#, HTML, CSS, JavaScript, windows batch scripting supports executables whereas Linux doesn’t have an equivalent to EXE. Other than that, all the coding languages are pretty much the same. Windows has .NET and ASP, but you can use these on Mac and Linux as well. Has Support for Android development and can run a VM for Mac development. | Mac applications.  Releasing on the app store for $99 a year  Developing for android is free developing for iPhone requires a system running MacOS. |

## 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**: Personally, I believe that windows would be the best bet here, it’s easy, almost everyone has experience with it and it has a lot of flexibility in terms of development. It has some of the best IDE’s and software to work with as well.
2. **Operating Systems Architectures**: Windows has different software that can be used at all levels as well as GUI support which is easier than the other options.
3. **Storage Management**: Since Windows 8.1 Microsoft has released and been supporting a feature called storage sense. Storage sense can automatically free space on any drive in your pc by removing temp files and automatically cleaning your recycling bin.
4. **Memory Management**: Creating a library to host pictures is easy to do and will help in storage management for our project. This lets us store the entire project on one computer for easy access.
5. **Distributed Systems and Networks**: Porting games from windows to iOS and android may not be the simplest task, but it isn’t the end of the world I think the hardest part would be getting the server set up to support multiplayer, something that I have done in Unity engine before. It is not easy at all and implementing cross-platform would be very difficult if you have never done something like that before.
6. **Security**: Thankfully, windows does come with pre-installed defenses. Windows is fairly good at protecting you and scanning itself for viruses or any malicious content.