

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 | 03/14/23 | Treyton Davis | Updated descriptions of items in the template |
| 1.1 | 03/25/23 | Treyton Davis | Updated Evaluation Section of document |
| 1.2 | 04/12/23 | Treyton Davis | Updated Recommendations section of document |

**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, our newest client, wants a web-based version of their game Draw It or Lose It. They are asking for our help to streamline development and develop the application for them.

## Requirements

* If a game has begun, it will have one or more teams involved.
* Each team will have more than one player.
* All game and team names cannot match, and each team name must be able to see every team name to not allow duplicates.
* Only one game instance can be active at a time.

## [Design Constraints](#_2et92p0)

* If a game has begun, it will have one or more teams involved.
  + This constraint is easy to handle due to being able to create separate instances on the same class inside the program.
* Each team will have more than one player.
  + This constraint is also easy due to being able to set it up the same way as the teams and having separate instances of the same class.
* Only one game instance can be active at a time.
  + This was a little difficult due to not knowing how to create a Simpleton Class. It did get figured out and was accomplished before the milestone due date.
* All game and team names cannot match, and each team name must be able to see every team name to not allow duplicates.
  + This constraint is easy due to comparing names and seeing if any of them are the same to help the teams choose a unique name.

## [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)

For this UML, it is set up for easy readability. In the top left, we have the program driver which inherits from SingletonTester, which verifies whether everything is being used only once or not. On the bottom, we have GameService, Game, Team, and Player. The Game, Team, and Player classes also extends the Entity Class that was created for the application. The way that the UML is set up is that each GameService can have 0 or more games, each Game has 0 or more teams, and each Team has 0 or more 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)

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 easy to use and is easy to set up server configurations on. The terminal commands are easy to use and are also flexible to design, access, and make changes to the server. | Linux is the most difficult to use out of the main 3 desktop operating systems due to most of the commands requiring the terminal and plain Linux has no user interface to be used. It is the cheapest option for server-side development and is easy to maintain a server on Linux. | Windows server systems are expensive like Mac, but also has more software available to them due to WS being built off Home and Pro system. It also has a command terminal and a user-friendly interface. | Not best suited for hosting a web-based application due to the restricted nature of mobile devices.  Other options are much better for server-side hosting. |
| **Client Side** | MacOS is the most expensive option due to Apple controlling the prices of components. Most people need an average amount of time to get used to the system and need moderate skills to navigate OS. | Hardest OS to use if not used to running commands all the time. Linux is the lowest cost for most OS, some even being free. Not many people will want to use it due to needing the most time and proficiency in using it. | Windows costs less than Mac but also costs more than Linux. Most people will be used to the OS in the least amount of time. | Hard to code for due to some OS having strict requirements to be able to allow users to install applications. Flexible to clients and developers to see updates in real-time. Requires lots of time and skill. More difficult to implement compared to Windows. |
| **Development Tools** | Supports many languages like Windows and Linux. Can program through a terminal, but also has access to Atom, Eclipse, and PyCharm. | Supports many languages like Windows, but also does not require an IDE for programming due to being able to access files through the terminal. Linux does have access to IDEs, like Eclipse, PyCharm, and Atom, but is usually not needed due to the terminal being able to read files with code. | Supports many programming languages like HTML, CSS, PHP, SQL, Java, JavaScript, and Python. IDEs include Eclipse, PyCharm, Android Studio, and Atom. | iOS supports Swift (the main programming language for Apple) and supports Java, C++, HTML, CSS, and PHP. More difficult to program compared to Android and Windows.  Android supports many languages like HTML, CSS, PHP, Java/JavaScript, C++, and Kotlin (a newer language for Android only). Many IDEs for most languages, Kotlin requires Android Studio for Windows or Mac. |

## 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 due to the low turn-around of set-up and the cost is much lower compared to Mac. It has the most applications that can be used to be able to handle storage, web, applications, databases, printing, and mail servers. It also can be run for many days without needing to be restarted.
2. **Operating Systems Architectures**: Windows server includes many ideas for file management to control and manage the memory sticks and allows the developers to try different languages to work with and determine if a different language could help them.
3. **Storage Management**: Windows allows for striping or creating redundancy for hard drives or SSDs together to allow for high amounts of storage to be used while also allowing for data recovery between the hard drives and for cloud storage to be used with data recovery.
4. **Memory Management**: Windows allows virtual and physical address space to be used for handling all system processes and applications. It also allows for programs to be swapped between RAM and the page file that is usually stored on the same drive that Windows is installed on.
5. **Distributed Systems and Networks**: The Windows server operating system facilitates communication between processors and different workspaces, with its distributed systems and networking features. Utilizing networking support is a beneficial technique for applying and taking advantage of software for distributed systems. Communication and guidance between each other are facilitated by these systems, yet there are some shared issues such as routing and congestion problems.
6. **Security**: Windows allows users to create multiple accounts to control who can access what parts of the system to help keep data secured. It also allows for a VPN to keep all data going in and out to be encrypted to protect the company from a cyber-attack. Windows also has Defender to help protect the server from viruses and malware.