

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)3/4

[**Evaluation**](#_2o15spng8stw)4

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/22/22 | Violet Moore | Updated Executive Summary, Design Constraints, and Domain Model |
| 1.1 | 06/05/22 | Violet Moore | Updated Development Requirements for Server side, Client side, and Development tools |
| 1.2 | 06/017/22 | Violet Moore | Updated Recommendations |

**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 currently has a game, Draw It or Lose It, available as an Android app only. The Gaming Room would like to develop a web-based game to serve multiple platforms and be based on their current Draw It or Lose It app. The Gaming Room needs CTS to begin the development of the web-based gaming environment.

## [Design Constraints](#_2et92p0)

* The game will allow one or more teams to participate with multiple players assigned to each team.
* Team names, player names, and game names must be individual.
* Users need to be able to check if a name is in use when choosing the game, player, and team names.
* Only one occurrence of a game can exist in memory at any time. Unique identifiers need to be created for each occurrence of a game, team, or player.
* The game will follow industry standard best practices by using private and public variables and only allowing one instance of the program to run at a given 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)

In the UML diagram below, we can see how The Gaming Room's program will work.

There are 7 classes represented by the 7 boxes shown. Encapsulation is the object oriented programming principle demonstrated by the classes being separated. Instead of writing one large amount of code, encapsulation separates the code by classes which creates easier to maintain and debug code. Entity is a base class (or parent class) and this class contains the mutual information between the Game, Team, and Player classes. This object oriented programming principle is called inheritance and prevents the program from being repetitive. Similarly to encapsulation, inheritance improves the readability of the program for future writing and debugging. The GameService class sets up one GameService instance, meaning that only one GameService program can be running at a time. This is especially important in an online game because it prevents duplicate games, teams, or players from being created and causing the program to crash or more than one user playing from the same player profile. The GameService, Game, Team, and Player classes have multiplicity with no limits. Noted by the 0...\* meaning zero to many objects of those classes can be created and the program will still operate as normal.

The ProgramDriver is where the functionality of the program is written. ProgramDriver calls on each of the classes available to get the information needed for a user to be able to play the game. The SingletonTester is used by the ProgramDriver to test that the GameService class is indeed a singleton and will prevent the above mentioned program errors if more than one instance of GameService is created. While this is a brief overview of how The Gaming Room's program will work, it should give us a better idea of how the program will function and provide a good experience to its users.

**"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’s server side can automatically sort users into groups. However, if the groups are not used, new users are automatically granted access to all areas until revoked by admins. Software also needs to be kept up to date. | Linux has less vulnerabilities than Windows and Mac. Linux also needs simpler code so that it does not have to process as much information to run efficiently. Software also needs to be kept up to date. | Windows systems need to have a strong firewall set up to help protect the server and users from being affected by viruses and attackers. Software also needs to be kept up to date. | Mobile devices must be set up to not enter a sleep cycle. Usually requires additional apps or software to be installed and kept up to date on the mobile device. The mobile device also needs to have adequate memory to process the server programming. |
| **Client Side** | Macs work well processing programs with multiple programming languages working together. They can also iterate over larger programs more efficiently than Linux or mobile devices which provides more flexibility in the development phase of the program. | Linux requires skilled programmers to write concise code to again ensure that Linux does not spend unnecessary time processing lengthy code. Programmers also need to set up the code as efficiently as possible for the Linux operating system. | Windows operating systems have more vulnerabilities than other operating systems which need to be protected against with secure coding practices. Windows systems work well processing large programs with multiple programming languages as well as Mac. | Similarly to Linux, mobile devices need concise code written in order for the system to be able to process the code most efficiently for its operating system. |
| **Development Tools** | Mac will likely use Java or C++ with an IDE like XCode or Eclipse. These programming languages will work with a Graphical User Interface (GUI) for users to interact with during gameplay. | Linux will likely use Java or C++ with an IDE like Visual Studio or Eclipse. However, Visual Studio can sometimes be too large for Linux to run efficiently using it. These programming languages will work with a Graphical User Interface (GUI) for users to interact with during gameplay. | Windows will likely use Java or C++ with an IDE like Visual Studio or Eclipse. These programming languages will also work with a Graphical User Interface (GUI) for users to interact with during gameplay. | Swift is a programming language that will be used for Apple mobile devices. Android studio is a programming language used for Android mobile devices. These programming languages will work along with programming languages like Java or C++. |

## 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**: The recommended operating platform that will allow Draw It or Lose It to continue to expand to other computing environments is Mac. This is because of Mac's flexibility in processing different programming languages quickly and its compatibility with different user operating systems.
2. **Operating Systems Architectures**: The Mac architecture is a multiprocessor system. This means that there is more than one processor in the computer. Having a multiprocessor system provides peace of mind because if there is an error in one processor and it crashes, the remaining processors can take over the work of the crashed processor. Doing this only causes the computer to run slightly slower rather than crash the entire program or computer. Another benefit of multiprocessor systems is that they are able to work faster than single processor computers. Each processor takes care of a specific task allowing the best user side and server side experiences.
3. **Storage Management**: Both a storage area network and cloud storage should be utilized in Draw It or Lose It. The recommendation for information in cloud storage would be the photo library for the game. While user and game information should be stored in a pre-allocated server disk to optimize server communication speed and improve user experiences.
4. **Memory Management**: The memory management recommended for Draw It or Lose It is a storage-area network. This type of memory management has multiple hosts and multiple storage arrays because it has more ports for the storage and server to be accessed through. This allows communication to flow more efficiently from the user to the server to the storage area. Information travels quickly on its own host or storage array and does not get backed up or cause system crashes like other memory management has the potential for.
5. **Distributed Systems and Networks**: To have Draw It or Lose It be able to communicate between various platforms, the recommendation is to set up a server with multiple ports to check and verify which operating system a user is on while playing Draw It or Lose It. The server will then connect them to the version of the game that is compatible with their operating system. Compatible code will also need to be written so that the different devices can communicate with the network and interact efficiently. If one devices' code is not properly written, users may not be able to interact from different device types. Inefficient code, for example, can also cause a user on a Windows computer to have a poorer experience while a user on a Mobile Device may have a great experience. However, if these users are playing a game together from their individual devices, both experiences will be poor because each devices must interact with the server. If one device is not performing optimally, it can affect other user experiences.
6. **Security**: Multiple levels of authentication requirements is the most important security recommendation. Frequent checks of authentication access as different security levels are accessed instead of only requiring one log-in at level one. Ensuring best practices are used in setting up administrator accounts and passwords is another security recommendation. Removing inactive administrator accounts to prevent their information from being compromised and used to access the system. Ensuring that unused features of the program are removed so that attacks cannot be accomplished through those program areas. Another security consideration with Mac is to have new users on the server sorted into a group with restricted access. This prevents server users from being given all permission immediately. Instead, an admin can grant access to areas that the new user requires.