

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_jp56deqxufev)

[**Table of Contents 2**](#_rrcarf3pqift)

[**Document Revision History 2**](#_ctbry31am0ty)

[**Executive Summary 3**](#_8pwh80o59398)

[**Requirements 3**](#_yw18re9y120c)

[**Design Constraints 3**](#_f9tbdnjpomwy)

[**System Architecture View 3**](#_ahkf75cpp6zw)

[**Domain Model 3**](#_h4s25c8cvfs2)

[**Evaluation 4**](#_4h08qyr5d6dl)

[**Recommendations 5**](#_b7c5dv8lqbgx)

## [Document Revision History](#_yic58viqo0to)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | 6/8/2025 | Thomas Comer |  |
| 1.1 | 5/24/2025 | Thomas Comer | Created the entity class, and refactor the rest of the classes to utilize this new class. |
| 1.0 | 5/18/2025 | Thomas Comer | Implemented the singleton pattern onto the GameService Class, and added the iterator pattern to the addGame() and getGame() methods. |

**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](#_cuj4grortz5c)

Draw It or Lose It is a mobile application for android, published by The Gaming Room, based on the game show Win, Lose or Draw. The Gaming Room wishes to develop their application into a web based desktop version of the game, in order to allow it to run on operating systems such as Windows, OSX, and Linux. This design document will describe the general design of the game, as well as addressing the software requirements and design constraints of the project.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_dzn2k8ope4e)

The Design Constraints for the Draw It or Lose It game are as follows:

* **Cross-platform:** Since the game already has an android version, any kind of online multiplayer support will require consideration of cross platform compatibility. Also worth consideration, is the fact that any desktop environment should be able to run the game, so testing will need to be done on multiple different operating systems in order to ensure the game will always run correctly.
* **Desktop User Interface:** The currently existing user interface that is in use for the mobile version of the game must be completely redesigned with the desktop use in mind. What works well for a small touch screen doesn’t work as well for a larger, mouse controlled monitor, so that should be put into consideration.
* **Multiple teams and players per game:** The game must be able to support multiple teams, with the ability for either one or multiple players on each team. Beyond this, the team names and the player names must be unique in every single game session.
* **Unique IDs:** Each game, team, and player must be tracked via a unique ID, in order to avoid duplicates, so a system of generating unique IDs, assigning them to each element, and then be tracked to ensure that the game only utilizes unique IDs.

## [System Architecture View](#_ld7p5ebvyib5)

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](#_7dpc15t8csn)

Below is The UML (Unified Modeling Language) Diagram of the different objects in the program for the Draw It or Lose It game. UML Diagrams are used to display the attributes in all of the classes of a specific Java Program, and how the classes relate or inherent the attributes to the other classes. The first class to discuss is the Entity class, which the Game, Team, and Player classes all inherent attributes from. The Game, Team, and Player classes also all reference each other, as well as the GameService Class. This makes the Entity class a superclass in relation to all of the other classes. Next, we see the ProgramDriver class, which points to the SingletonTester, in order to fulfill its function to test the code of the program.

**"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](#_k56wij3xcw3o)

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** | The MacOS environment for server hosting is not the most optimal environment to choose for this project. Unlike the client side application, where developing the application for all versions must be considered, the server side application will only be developed for one OS. MacOS has a closed source system of libraries that often require expensive licensing costs in order to develop large server applications. Another flaw is that the ecosystem of development for MacOs is far more closed off from accommodating support for other platforms, which overall makes it generally a poor choice for developing the server application on. | Linux is generally the best OS for building a server application on. The biggest reason for this is because Linux is an open source OS, meaning the OS can be modified to fit the needs of your application. This can even mean stripping down most functions of the OS so that the only processes it runs are those necessary for your server application, saving resources for running the server. Because it is open source, this also means that there are no costs for licensing any kind of programs for running the program, making the upkeep cost very low in comparison to other options, as well as providing a very good base to scale off of. | Windows is a fine option for server hosting, not the best, but not the worst. The biggest pro for Windows hosting is it is the most popular OS in the world, which also means it has the most tools designed for it. However, since Windows is not open source, it has many of the same flaws that MacOS has, in that these tools can have potentially expensive licensing costs, and the OS itself can’t be customized for your own needs. | A mobile OS environment for the server application is probably the worst option of all the possible options. One of the largest problems with developing on a mobile OS is that mobile devices use ARM architecture for their CPUs, a completely different architecture than every other platform and can often result in difficulties with cross platform server management. Another problem is that there is no real benefit to be gained from running a server client on mobile hardware, as there isn’t really ever a reason that the server machine should need to be moved enough to warrant the need for a mobile device. There is, however, one very good benefit that can be received from mobile server application development, is that specifically Android is open source, and thus gets many of the same benefits as Linux in that regard. |
| **Client Side** | Since the application is being developed as an HTML web app, generally there aren’t very many considerations for platform dependent development that need to be made. This is because most browser environments are platform-agnostic, and will be capable of running the same web code no matter the environment that they are put on. However, if an issue does come up that is Mac independant, it will most likely be costly and time consuming to fix, because Mac development is much more niche of a skillset than every other platform. | Similarly to MacOS, there aren’t many considerations that should have to be made for the platform specific development in regards to Linux for the web client. However, since the server development will already be built by a team with experience on linux, and linux only issues would most likely be not terribly difficult to develop fixes for, as an experienced linux team is already available for the project. | Once again, due to the client’s nature as an HTML web app in a browser makes it so that there shouldn’t be very many platform specific considerations for the windows client. However, due to Windows being the most popular OS to develop for, fixing any problems that are windows dependent shouldn’t be too big of a problem either, making it about the same difficulty as the considerations that go into Linux. | For the mobile versions of the client, there already exists an Android version of the app, and this is what the IOS version will be based off of. Depending on the engine the Android version was built off of, it may be a simple process to port it over, as engines such as Unity have options to directly compile the android version into an IOS version with minimal work necessary. However, if the Android version was custom made, then the game will most likely have to be redone from scratch on an engine level in order to function with IOS libraries. In this regard, an experienced programmer in both IOS and Android will be required in order to ensure accuracy between both versions of the game, and ensure that there are no problems that only exist on one version. |
| **Development Tools** | For the Mac webclient, the game will be built using HTML, CSS, and JavaScript, as well as utilizing Websocket connections to connect to the server program and receive data for any webclient. The IDE that will be used for this is VSCode, an IDE that supports all of those languages very well, and is usable on Mac, Linux and Windows, meaning the Mac version can be developed on any of the Major OS’s | With Linux, there will both have to be the Web Client, and the Server Client. The web client on linux will be made using the same methods as on mac, using the same languages, and the same IDE, built at the same time, as the code for the web client should be the same for all OS’s. As for the server, the program will be built in Java using the Eclipse IDE. This will still allow it to send out packets using websocket and communicate to the client. If you want to make modifications to the Open Source Linux OS that the server is running on, then VSCode can be used to make those modifications as well. | For windows, the web client will once again be built on the same HTML, CSS, and Javascript in VSCode. The site should have the exact same code for every operating system, as all modern browsers are platform agnostic. | For IOS, if the android client was built in an engine that was compatible with compiling in multiple Mobile OS’s, then the game should be built in the same IDE and language as the Android version. An example of this is Unity, which uses C++ in VSCode, and has compilation to both platform options. However, if the android version was not built in an engine that allows for this, then most likely the IOS version will have to be rebuilt from the ground up. For this purpose I’d still recommend Unity, and perhaps rebuilding the android version from the ground up as well, as support for all mobile versions in one code base will reduce costs greatly, and if you are already rebuilding for one platform, then the other platform would be included by default with Unity. |

## 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**: Based on all of the considerations, I believe the most optimal Operating Platform for the Server network of Draw It or Lose It to be a Linux based build. The reasons being, Linux is an open source platform, and thus can be modified for any needs the server network needs, as well as ensuring there are no licensing related issues with choosing this operating system.
2. **Operating Systems Architectures**: The linux architecture system uses what is called a “Layered Model,” a system which has several layers of interfaces that each layer is built on top of. At the center of the model is the hardware that the computer uses to compute all of the operating system processes. Then, the kernel, which is the core of the operating system itself. The kernel is responsible for allowing software and hardware to communicate, and is constantly resident in the memory. The next layer is the shell, which is the front to allow the user to communicate to the kernel, either through just a command line terminal, or through a graphic user interface. The final layer is the application layer, which is where different applications interface with the previous layers. This is the layer the server application will be built upon.
3. **Storage Management**: Due to being a web-based game, the best solution for storage would be a cloud based solution. With a cloud based system, running low on physical storage space isn’t a great issue, as cloud solutions often offer the ability to upgrade for as much storage space as is necessary. This allows for a future-proofed and scalable solution to long term storage. The best fit for Draw It or Lose It would be Google Cloud, as it is very reliable and inexpensive.
4. **Memory Management**: The server program is written in Java. Java utilizes an automatic memory management system, called a garbage collector. This will automatically add and remove memory processes as needed. Beyond this, since the machine the server is running on should not be used for any other processes besides the server, as long as there is a relatively large built in amount of memory on the machine, it should constantly have enough memory to run the program at all times.
5. **Distributed Systems and Networks**: The client program will be distributed through http via a web browser. Almost all modern web browser’s will allow for any site to run the same code via any operating system, and thus will allow for compatibility on all platforms. The server application will communicate to any client via webhooks and send the needed data back and forth to the client via RESTful API.
6. **Security**: The main concern for security is the long term storage data. Google cloud services will automatically encrypt data to ensure it is secure and safe. A few other ways to ensure security are to ensure the site is secure using SSL Certification, and utilizing end to end encryption for user connections, allowing for the user to have a safe and protected connection to the host.