

Draw it or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 03/13/2025 | Sarah Louise Harrison | Setting primary expectations for the development of the application Draw it or Lose it. |
| 1.1 | 04/03/2025 | Sarah Louise Harrison | Elaborate on the Evaluation Section. |
| 1.2 | 04/16/2025 | Sarah Louise Harrison | Add Platform, Architectures, Storage Management, Memory Management, Distributed Systems and Networks, and Security to 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 Client, The Gaming Room, would like to develop an application for a game called Draw it or Lose it. An environment to facilitate the development of a web-based version of their current Android only mobile application. It will need to be able to render drawings at a steady rate within game-situation time limits. We will discuss hardware needs once the software decisions have been made.

## Requirements

The client’s requirements are as follows: a game with one or more teams, each team with multiple players, games and teams with unique names so there is no other iteration of that game and team together.

## [Design Constraints](#_2et92p0)

Moving from an application environment to a web-based software environment include a few constraints. Some of these constraints include needing to expand from Android only operating system, to include any browser (Chrome, Microsoft Edge, Firefox, Safari, etc.) to work with any browser (MacOS, Windows, or Linux). Another constraint would be where these unique team, game, and ID names are to are stored so that they can be recalled upon game creation. There also needs to be higher bandwidth to support steady rate rendering of the drawing in real-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 UML Diagram shows a Program Driver and Singleton Tester within the com.gamingroom package. The Game service, which has zero to many instances of Game, Team and Player classes which all recall to the Entity. The Game Service class includes a singleton pattern that allows for only one instance of GameService that includes the methods of GameID, TeamID, and PlayerID to exist at a time.

**"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 Requirement** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is more user friendly than its counterparts, but significantly more expensive to produce in their OS. Its inflexibility in adding modular-style software development makes it far less programming-friendly. Mac requires specialty servers to run, costing more in licenses to make the program cross-compatible with Windows and Linux. | Linux’s customization and modular abilities make it much easier to program and create the cross-compatibility needed. It also allows for pre-engineered software to be implemented. However, because it is a free OS, system crashes are possible, and tech support is open sourced and only available from Linux users, not a dedicated team. | Windows, much like Linux, allows for easier cross-compatibility through customizability. It’s more affordable than Mac (although, not free like Linux), and can be used with pre-packaged programs. Windows can also be used on any hardware available, including Mac through programs like Parallels and VMware Fusion. | Scaling down to a mobile format is doable with cross-platform frameworks like Flutter or Ionic. However, it is harder to confront and fix security issues. Also, it would require two separate servers to run both Android and iOS. |
| **Client Side** | Mac is user-friendly, and with little need for updating or anti-viruses, it is a great choice for consistent web-based gameplay. However, it is the most expensive option, as to use a MacOS, you would need a Mac computer. That being said, it has far more advanced security measures. | Although Linux is the cheapest option and is useable on most devices, it has a large learning curve to download programs on to, needing basic knowledge of software development (specifically Python). The client would need to have hired a dedicated team to translate issues coming from the server. | Windows is far more common than Linux, and can be used on far more diverse products (as opposed to Mac, which only works on Mac products). However, Windows is notorious for their forced updating, possibly interrupting gameplay as well as a more costly security system to keep the program running as issues arise. | Because smart phones and other mobile devices are more readily available, having a mobile port that is cross-compatible with players on Mac, PC, or Linux makes usability and playability much broader. The possibility of having one dedicated person to program both iOS and Android would be slim, so you may have to hire more than one developer to keep the mobile applications running, in addition to those who are taking care of the desktop programs. |
| **Development Tools** | MacOS can use Swift to develop applications, while also implementing languages such as C++ and Python. Swift is easy to use and user-friendly. IDEs like Xcode allow for integration into Swift. | Linux uses similar IDEs as Windows, such as Eclipse (most accurate option) or KDevelop (a free option). Linux reads languages such as Java, C, and C++. | Like Linux, Windows inherently reads Java, C, C++, and Python, and can use the similar IDEs to Linux: Eclipse, PyCharm, or Visual Studio (the most preferred). | Because the two common types of mobile devices run on either iOS or Android, it will need to be developed in both of these. iOS can use the same IDE as it’s Mac cousin, with XCode being the most efficient. Android, however, has its own system in Android Studio (recommended by Google), but IDE Xamarin is better for cross-platform development. |

## 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 as the main operating platform. With integrated security measures and anti-viruses, while also allowing for modular programming, Windows is going to be the most efficient platform to begin cross-compatible development.
2. **Operating Systems Architectures**: Keeping in mind that the UI would need to change based on whether the application is being used on a mobile device or a desktop/laptop, there are similarities that can easily be integrated into both architectures. Because we are building native through Windows, we can use their integrated systems to create a basic architecture comprised of a search system (to find and recall the user’s specific game through Player ID, Game ID, and/or Team ID), a menu including Start, End, Pause, Continue, etc., as well as the title of the application, a window for the drawings to be recalled, and another window to draw and send what is input. This would create the most accessible organization of the program.
3. **Storage Management**: Since Draw it or Lose it is planned to be a web-based, cross-platform game, I suggest using a cloud-based storage management system. Because of its easy upgradability and integrated security systems (depending on the storage site used), it is the safest and fastest option for keeping ongoing unique-named games stored. I would suggest Google’s Cloud because of its scalability and price; as the needs grow, the storage is able to grow. With the expandable price point ranging from $0.026-$0.015/gb for data storage, and $0.0010-$0.0100/1000 operations for data processing, Google has become a contender in best practice for storage management with built in security measures.
4. **Memory Management**: Because the requirements include real-time drawing rendering to be shared amongst multiple platforms at 200 high-definition images around 8mb each, a short-term memory solution will be needed as the game progresses. Also, a larger memory need will be necessary to keep all iterations of game, player, and team available for recall to make sure that there will be no repetition. I would suggest another cloud-based system, such as Dragonfly Cloud, to be able to handle the job. It stores all RAM across their own servers, with low-latency read times of 1260 megabytes per second (well within the 30 second rendering time the client has asked for). With a lower price point than most of their competitors, Dragonfly Cloud is a great starter option that also has the ability to grow along with the application.
5. **Distributed Systems and Networks**: By creating the system to work web-based, we should focus development that is available via browser. I would suggest a browser that is usable on any platform, such as Google Chrome. Using Chrome will also make it more seamless when communicating with the Google Cloud for storage management. That way, it uses one company’s integrated systems for any issues (like security, bugs, or storage) that may arise.
6. **Security**: As repeated previously, when using one integrated storage system, such as Google Cloud and Google Browser for Windows, Google’s security system is taken care of. Therefore, we do not need to spend time, resources, and money on creating our own security system. As for memory, Dragonfly Cloud offers multi-factor authentication as well as data transfer through Transport Layer Security, which is a protocol that provides integrated secure encryption system that requires a decryption key to open any intercepted data.