

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.3

## 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)5

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/16/2022 | Easton Kyhl | Completed the Executive Summary and Design Constrains sections of the document |
| 1.1 | 09/17/2022 | Easton Kyhl | Finished the Domain Model section and cleared up any sections of text that were no longer needed |
| 1.2 | 09/30/2022 | Easton Kyhl | Filled in the evaluation table discussing client side, server side, and development tools for different operating systems |
| 1.3 | 10/16/2022 | Easton Kyhl | Made recommendations based on the application and operating system on how to host their expanding game |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room would like a web-based game developed that will be available on multiple different platforms. This game is going to be based on a current application they have developed, Draw It or Lose It, that is only available as an Android app. Since the game is already developed on a different platform, our goal will be to transfer those ideas in a way that we can make the game available on all platforms. The main points that we know of the game are:

* It can have one or more teams
* Each team will have multiple players
* Each game and team name will need to be unique
* Only one instance of the game can exist in memory at any time

## [Design Constraints](#_2et92p0)

* App needs to be available on different platforms
  + Making the game available on different platforms, in this case, means the game needs to be remade.
* Should be based off their already developed app: Draw It or Lose It
  + Since the game was already designed and running as an Android app, we have the structure to make a remake as easy as possible and could use this as an opportunity to refine and make changes to the game if needed.
* Will be created as a web-based application
  + Will need to be created for different sizes of screen, varying levels of hardware, and different browser sources to ensure it stays available on every platform possible.

## [Domain Model](#_8h2ehzxfam4o)

In the UML diagram, we have one superclass that three subclasses inherit from. This will make it easier to ensure we only have one game, team, or player that has the same name or id since everything will be a unique entity before a new game, team, or player. Each child class is related to each other and another class that stores the game instances to ensure there is only one game active with the same name or id. They are all related as the GameService class can have multiple different games, the Game class has at least one team, and each Team has at least one player playing the game. The ProgramDriver uses the Singleton Tester class so we can ensure there is only one instance running at a time and no game shares the same id or name before developing an otherwise functional 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](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | As of April 2022, macOS Server has been discontinued for new OS updates. In order to host a server on this platform, it would require the older update macOS Monterey, which I would consider a significant disadvantage, largely for security purposes.  In order to utilize all services from macOS server, the client would need a machine dated back to an OS before September 2017. | Most popular for hosting web servers. Since it is more popular, it has a lot more support and features that web designers would come to expect from servers  Usually much cheaper and has a wide range of uses. It would be difficult to find a reason to not use one unless the client were to desire to run a windows specific application | More people are familiar with Windows as an operating system, lowering the barrier of entry than that of a Linux server. Easy to find and incorporate different third-party software  The cost of a windows server is usually what drives people away as many of the windows-specific applications require a license to use. Many applications have even turned to a renewal subscription rather than a lifetime license | A mobile device would not be well suited to run a server. What makes them not as well equipped is mainly that they do not have the hardware capability to do so as performance does typically have to suffer in favor of portability, especially in this era of technology where it is getting increasingly more difficult to make things smaller.  Portability of a server could also pose serious safety concerns for data stored as servers are quite fragile |
| **Client Side** | To run with macOS requires very specific technology, potentially increasing costs for the project overall and provides less flexibility than some other operating systems. Working on macOS would also require knowledge of specific, slightly lesser-known languages, potentially creating more complications and cost in maintenance | The client would need likely desire to have a couple people ready and knowledgeable on the Linux operating system since it is generally less used overall, but finding people for the position would likely not be difficult  While it may be more difficult to set up and navigate the operating system and server, development should be rather simple as languages used are more well-known and used | Windows would likely be easier to maintain and run over Linux and easier to develop than macOS with more common languages used and a simpler OS in general  It would also be expected to have the best support and development tools available and easily accessible, especially with the relatively high cost invested to get to this point | Mobile platforms would likely also require a specific group of people for development and maintenance as less development tools are available for this platform. iOS would run into similar problems as macOS with lesser-known languages and tools |
| **Development Tools** | macOS contains support for the languages is uses often, like Swift, and has other tools that provides support for other popularly used languages through XCode which is initially free, but costs money later on to upload to the app store in the form of a yearly subscription | Linux has a wide availability of development tools, virtually all of them being able to be found for free like Eclipse or PyCharm, tools used quite often and are very well-known. Some of these tools even come preinstalled on many Linux distributions | Windows is similar to Linux in that there are a variety of free development tools available, especially their own Visual Studio environment. Visual Studio is often considered almost standard at this point for developing Windows applications. | Trying to use development tools on a mobile OS would not be advised.  Development tools for mobile applications would include XCode as mentioned before and Android studio for the major mobile operating systems |

## Recommendations

1. **Operating Platform**: The recommended operating system for The Gaming Room to host their game Draw It or Lose it is the Linux Operating System. The major benefits of using Linux to expand their game to other environments are cost and popularity. There are lower overall costs to operate on Linux in terms of licensing, it may just take a couple more people that can ensure.
2. **Operating Systems Architectures**: At the center of the Linux architecture is the hardware of the computer. It then expands to include the kernel, responsible for primary tasks of the operating system. Linux architecture then spans further to include the shell where, through all the different types of shells, the operating system can provide and run kernel services, allow for a graphical user interface, and enable command line interfaces. Last comes the application and utilities like any libraries needed.
3. **Storage Management**: Since the client is already expanding to different platforms, I would recommend using cloud storage. Cloud storage is ideal because The Gaming Room will not get limited by hardware if it comes time to expand the size of the game and also will not need to have people specifically trained in setting up and maintaining their own storage system.
4. **Memory Management**: For memory management, the recommendation would be to utilize virtual memory on behalf of the Linux Operating System, allowing the machine to use more memory than actually provided. The Linux kernel manages this swap space between these processes.
5. **Distributed Systems and Networks**: Since cloud storage was already recommended, it makes utilizing the cloud again a sensible choice. Preventing outages and maintaining uptime has been a considerable point for cloud-based services and for good reason. This again makes expansion and maintenance a breeze as full training or a new team need not be created just for this sole purpose. The client could also implement a REST API with a client-server architecture to ensure multiple platforms would be able to communicate with the selected server provider.
6. **Security**: With Linux being such a widely supported open-source project, it comes with a rather large community of supporters making the likelihood of vulnerabilities being found much less likely. If any flaw is found, that community is rather quick to ensure the vulnerability is patched as quickly as possible. The application could implement its own security as well in the form a role-based login system where users would have to log in and could only have permissions based on a certain role they are given.