

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[Draw It or Lose It 1](#_Toc115519183)

[**CS 230 Project Software Design Template** 1](#_Toc115519184)

[Table of Contents 2](#_Toc115519185)

[Document Revision History 2](#_Toc115519186)

[Executive Summary 3](#_Toc115519187)

[Design Constraints 3](#_Toc115519188)

[System Architecture View 3](#_Toc115519189)

[Domain Model 3](#_Toc115519190)

[Evaluation 4](#_Toc115519191)

[Recommendations 6](#_Toc115519192)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9-16-2022 | Dustan Baker | Initial entry. Outline of Summary, constraints, model. |
| 1.1 | 10-1-2022 | Dustan Baker | Evaluation |
| 1.2 | 10-11-2022 | Dustan Baker | Operating Platform, System Architectures, Storage/memory management, system and networks, security |

**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 wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. The game is a party game by nature and will need to be reasonably fast, but with no high-end graphical requirements this should not be a problem. The game will need to be written in a way that can be utilized in a web browser and a web API will be the quickest way to implement this change.

## [Design Constraints](#_2et92p0)

The game will need to have the ability to have 1 or more teams with multiple players per team. The game and team names must be unique, and users need the ability to check if their names are already in use. Only 1 instance of the game can exist in memory at any point in time. Having unique identifiers will be crucial if the game is to be played on different devices in different locations so that everyone can play in the exact same game.

## [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)

First the”ProgramDriver” and the “SingletonTester” use each other to run and ensure the program is correct. The classes “Game”, “Team” , and ”Player” have a shared inheritance from “Entity”. “Entity” will be used to log-in the user in order to create the single unique instances of each “Game”, “Team” , and ”Player”. “Game”, “Team” , and ”Player” will have zero or more associations with each other.

**"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** | Apple does not currently produce dedicated server equipment and even if you were to use MacOS, it would be wildly expensive and the “Walled Garden” of the OS would likely prevent some functionality. MacOS is very stable. | Linux is the current “Gold Standard” for server-side applications. Its open-source, stable nature and lightweight kernel means that in the unlikely event you cant find a distro that fits your needs, you can just make your own. | While windows is more accommodating than MacOS, it falls short by being “Heavy” or demanding of resources, comparatively unstable, and basically spyware. Not to mention its more consumer based in design and servers don’t need a polished UI. | While mobile devices can provide “server like” functionality in a hive mind sort of way. Like how Google reports back data from android users. To rely on a battery powered cellular device is inherently a bad idea. Upgradability is not an option and the cost to deploy enough mobile devices to handle scale would be eyewatering. |
| **Client Side** | MacOS can use any browser, but most users will use Safari by default. When accessing the webapp from the safari browser the client-side CSS and HTML will need to identify the browser format then format content. Apple hardware is comparatively expensive | Linus distros typically use Firefox, but other browsers can be used either by default of download. It may be complicated to support more browser styles. When accessing the webapp from the client-side CSS and HTML will need to identify the browser format then format content. Will run on any hardware | Windows users will typically use Google Chrome or Microsoft Edge. These are both Chromium browsers so there shouldn’t be too many differences. Will run on most hardware. | The 3 major platforms for mobile phones are Android, Apple, and Windows. Depending on how large the application is, it might need to be formatted into a downloadable app from their app store rather than a browser-based app. Keep in mind, that a program that gets used rarely should not have to take up space on the user’s device. As most people have a mobile device already, the deployment will be cheap. |
| **Development Tools** | Server side is dependent on the platform it runs on. A universal ide or text editor such as Visual studio code or Notepad++ are most popular for developing the front-end application for a webapp and should provide any of the tools needed to script the app. For back-end development Java is a popular choice because it can be interpreted on all OS’s (write once, run everywhere). | | | Mobile apps have their own IDE like android studio. The server-side will be dependent on the rest of the application framework. |

## 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**: Linux OS, more specifically Ubuntu server.
2. **Operating Systems Architectures**: Ubuntu Server is incredibly light, making it ideal for server-side applications that need to run in the background all the time. Compared to Windows or MacOS, Ubuntu server manages memory and storage better, is generally faster, and uses fewer resources for operating system operations. Unless you have an unique requirement for another platform, you should use it. Use Ubuntu Server unless you already have a server that is running on a different platform and you want to stay using that platform.
3. **Storage Management**: Since we need a certain amount of speed for the game to perform as advertised and a marginal amount of security, we need to configure multiple drives into a RAID array. Raid 5 will give us the best amount of storage with 1 drive being used as a “backup”. We will still need to backup the data elsewhere for security purposes.
4. **Memory Management**: RAW memory management is supported by Ubuntu Server, which has both advantages and disadvantages. Because it isn't operating in virtual memory that needs to be pre-processed by the operating system, it will allow the application to operate a little bit faster. However, if a poorly built application runs amok, the system will experience a stack overflow. For the platform to remain stable, extensive testing must guarantee that the server-side application manages memory effectively.
5. **Distributed Systems and Networks**: Since there will be interaction exchange between all platforms due to the back-end server's design to answer to the requests of all applications. If necessary, the server and application server can be separated, which will necessitate the greatest network connection possible between the two workstations. The server and application server will both run on the same system. The software won't function properly or at all if the network breaks down in any way. To guarantee that a connection can be formed to all reliant services and servers before execution begins, fail-safe measures will need to be incorporated into the code

**Security**: Through the usage of the HTTPS protocol, password and user information will be encrypted during client-server transit. In addition, the data stored in the database, client server communication, and any sensitive password information will be hashed to secure it in the case of a breach.