

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

[**Evaluation**](#_2o15spng8stw)3

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

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 7/17/2021 | John Mullen | Initial proposal |

**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)

With the success that The Gaming Room is seeing on their Android exclusive app Draw It or Lose It, the company is looking to expand to a web-based format. This expansion will retain the current functionality and carry those features to a new user base. The game must adhere to the same game regulations and the user processes as the Android app. Keeping the rules of the game such as one or more teams being involved, multiple players, unique names, time limits, and a large library of drawings for the users to use as clues.

## [Design Constraints](#_2et92p0)

Each platform environment has its own way of storing, processing, and displaying data from within the program. In our goal to maintain the familiarity of our already established and successful product in the Android version of Draw It or Lose It, we must find the necessary languages and processes capable of conveying these features to other platforms. Often times, we will be looking to multiple languages and hybrid environments of them in order to adhere to the requirements of different OS and mobile environments while maintaining end-user uniformity.

## [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)

When reviewing the UML Diagram, the first thing I notice is that Game, Team, and Player classes all share an “is a” relationship with Entity. This means that they all inherit from Entity. Game is an Entity, Team is an Entity, and Player is an Entity. This means that Game, Team, and Player all have common attributes in their ID and name. This means that it can be defined as Super Class Entity, with only variations being handled within the Game, Team, and Player classes.

Similarly, we see that Game, Team, and Player share a “has a” relationship. Unlike them all sharing the common attributes with Entity, they build upon each other. Games have Teams, and Teams have Players. In addition to these three classes, we also see that GameService has Games, another class that is not defined by Entity, but utilizes its attributes via the Game, Team, and Player classes. Via this relationship, we can also refer to it as a reference, since GameService references the information in Game, Game references attributes in Team, and Team references data stored in Player.

****

## [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** | Flexible terminal commands to configure the server, access, or make changes. | Flexible terminal commands to configure the server, access, or make changes while being more cost friendly than Mac. | More software options available compared to other OS. | For the best results, the server should not be mobile and should have the ability to be tracked at a single place. |
| **Client Side** | Moderate expertise and time required. Cost comparable to Windows. | Maximum expertise and time required. Minimum cost associated. | Minimum expertise and time required. Cost comparable to Mac. | Provides the clients with flexibility of use, and developers the ability to see updates at any place. Slightly more difficult to implement than other options. |
| **Development Tools** | HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github, Notepad++, databases (eg. mongoDB,SQL,Cassandra) | HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github, Notepad++, databases (eg. mongoDB,SQL,Cassandra) | HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github, Notepad++, databases (eg. mongoDB,SQL,Cassandra) | HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github, Notepad++, databases (eg. mongoDB,SQL,Cassandra) |

## 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**: When it comes to operating systems for the game, we need to look beyond the definitive platform variations. We also need to account for different systems such as iOS vs Android, or Linux vs Windows. These will all require personalized variations to how they store, read, and run the data. These platforms are in a constant debate in order to provide the best games to their community, and the best community to game companies. The most beneficial choice would be the community that would be able to provide the most benefits to the developers and would be able to receive the most in return. With the unparalleled size of the Windows OS community, I believe that this would be the best OS to target.
2. **Operating Systems Architectures**: For the OS architectures, we have two main options; ARM and x86. For this application, the limitations experienced utilizing ARM would not make it the best choice. X86 provides us with a much more reliable and capable foundation. We will also have a significantly larger hardware selection and a large support group for assistance and knowledge.
3. **Storage Management**: When it comes to storage management options, we have two main choices that we can utilize. The most common option that we see when it comes to storage options is an HDD. Using an HDD will be a much cheaper option, but at the expense of speed and being an outdated option for some newer processes. The alternative is utilizing an SSD. This newer choice is significantly faster than an HDD and is easier adapted in modern and upcoming processes.
4. **Memory Management**: When we look at memory management in terms of the Windows OS, since it is our recommended choice, we need to understand how the system chooses to allot its memory space. Windows utilizes both 32-bit and 64-bit system options. These different versions reserve 4 and 8 gigabytes of memory respectively. These memory allotments ensure that the system keeps the ability the utilize those sections of memory without fear of conflict or overwriting unnecessarily.
5. **Distributed Systems and Networks**: The most common format that we see when it comes to multi-platform games, we usually see a database that is distributed to each user’s device and stores the data locally. It then shares the communication from the local user’s device, as well as the input from each user and conveys inter-game communications from one end point to another. Though this two-way communication format can be challenging, storing the database on each user’s computer will free up a large portion of the communication that will allow the system to prioritize the user’s input per game.
6. **Security**: Security needs to be a priority for any multi-user program that we see today. With all of the communication going back and forth for each user, there are many of opportunities for data to get into the wrong hands. Data controllers are a great choice for the format of this application to minimize data collection and processing. Being able to anonymize and integrate pseudonyms into user profiles can help disassociate information with users.