

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.30j0zll)1

[**Table of Contents**](#_heading=h.1fob9te)2

[**Document Revision History**](#_heading=h.2et92p0)2

[**Executive Summary**](#_heading=h.tyjcwt)3

[**Design Constraints**](#_heading=h.3dy6vkm)3

[**System Architecture View**](#_heading=h.1t3h5sf)3

[**Domain Model**](#_heading=h.4d34og8)3

[**Evaluation**](#_heading=h.2s8eyo1)3

[**Recommendations**](#_heading=h.3rdcrjn)5

## [Document Revision History](#_heading=h.2et92p0)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <07/18/21> | <Jordan Stelivan> | Edited cover page, wrote description for UML diagram, Filled out information under each individual header, added very brief amounts of information to the ‘evaluation’ and ‘recommendation’ sections due to limited knowledge. |
| 1.1 | <08/01/2021> | <Jordan Stelivan> | Updated Evaluation and Recommendation sections |

**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](#_heading=h.tyjcwt)

Our problem here is that we need a program that can run an instance of the game ‘Draw It or Lose It’. We need to be able to track the game with one or more unique teams involved. Each team will also hold multiple players. We must have only one instance of the game running at a time. However, we also want multiple unique games to be able to be stored within the system. Not running at the same time, but storing their information for later use. Allowing users to quit out and come back later if need be.

## [Design Constraints](#_heading=h.3dy6vkm)

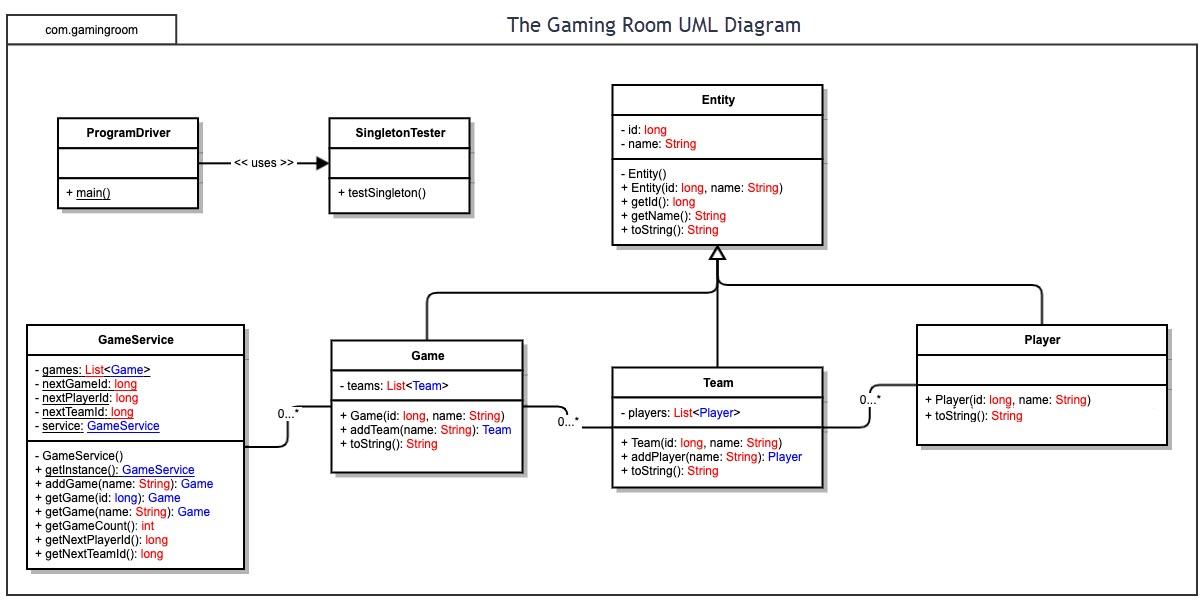
Our application will be web-based. This means that we’ll need to make sure that different systems using different browsers can access and run the game. From a software standpoint, we’ll want to avoid using up too much memory during an instance of the game in order to make streaming the game to various computers at the same time possible.

## [System Architecture View](#_heading=h.1t3h5sf)

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](#_heading=h.4d34og8)

Starting with our GameService class, we can see that only one of this class can exist thanks to the use of a Singleton Pattern. This further extends into our Game, Team, and Player classes. Allowing us to store multiple, but unique, instances of each class. These classes are also able to inherit the same data structure as our Entity class because each class uses the same name and id structure. Our program adheres to the object oriented principles quite well, as only the necessary amount of information for the game to run is provided within the layout of the program.

****

## [Evaluation](#_heading=h.2s8eyo1)

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** | -Server-based deployment  -Shouldn’t run into memory issues while hosting  -May be slightly lacking in security options | -Higher security  -Lacks the same level of support as Windows/Mac  -Server-based deployment | -Server-based deployment  -Excellent levels of security between platforms  -Bloated OS may slow down server side cpu | -Would reach a larger User Base, requiring more host space  -Server-based hosting available, but less reliable in certain parts(mobile network connection(s))  -Lacking security options |
| **Client Side** | -Smaller user base  -Will require niche browser testing (safari)  -Decent client side security options | -Much Smaller user Base than other OS  -Excellent client side security | -Multiple browser options to access application requiring more testing  -Excellent client side security options | -Many different test cases needed  -Functionality of the app varies between devices  -Adds a level of mobility for users |
| **Development Tools** | -Visual Studio Code  -Homebrew  -Atom | -Vim  -C/C++ | -C, C++, C#, Java, Python, etc.  -Visual Studio Code  -Eclipse  -Jira | -iOS (Swift)  -Android (Java)  -Windows Phone (C#)  -Licensing costs for windows, apple, and android |

## 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**: With where the market is at this point and time, I would recommend that the development take place within the Windows OS. It has the most flexible options on almost all fronts for both developing and maintaining software.
2. **Operating Systems Architectures**: The Windows OS allows us to set up our gaming application within the Server-Client architecture that we initially aimed for. However, a cloud based architecture may be something worth looking into further.
3. **Storage Management**: Windows relies on the use of either HDD or SSD/Sata Drives for it’s storage. Storage allocation and Free-space management will be the keys to proper storing the program’s data.
4. **Memory Management**: CPU generated I/O calls will allow the OS to call on data that is otherwise stored outside of the Main memory.
5. **Distributed Systems and Networks**: The largest issue here will be limiting the amount of data needed to be transferred over the network. Many end users will have hardware strong enough to run the program. However, many are also lacking a solid internet connection which may lead to issues with maintaining a stable performance from the game if too much data is transferred at once.
6. **Security**: As all of the game information will be stored on the host side and not the user side. The system’s security will need to be able to prevent infiltration and extraction of user data. Windows has several excellent security options available.