

Draw It or Lose It

**CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.30j0zll)[1](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.30j0zll)

[**Table of Contents**](#_1fob9te)2

[**Document Revision History**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.2et92p0)[2](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.2et92p0)

[**Executive Summary**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.tyjcwt)[3](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.tyjcwt)

[**Design Constraints**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.3dy6vkm)[3](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.3dy6vkm)

[**System Architecture View**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.1t3h5sf)[3](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.1t3h5sf)

[**Domain Model**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.4d34og8)[3](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.4d34og8)

[**Evaluation**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.2s8eyo1)[3](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.2s8eyo1)

[**Recommendations**](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.3rdcrjn)[5](https://docs.google.com/document/d/1t01n6hRaGCuE4gd_hS77wThUGJfkFR9D59aAS65yqd0/edit#heading=h.3rdcrjn)

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 09/20/20 | Timothy Crowell | Added support for additional platforms |

**Executive Summary**

The game application we are working on is a web-based multiplayer team-based game called Draw It or Lose It. Teams will take turns guessing what each image is, while the image slowly renders. The game is currently on the android platform, but The Gaming Room wants to bring it to more platforms. We can do this by creating a version of the game that will run on Mac, Linux, or Windows. The client requirements are to evaluate each platform and determine what is necessary to bring the game to each platform.

**Design Constraints**

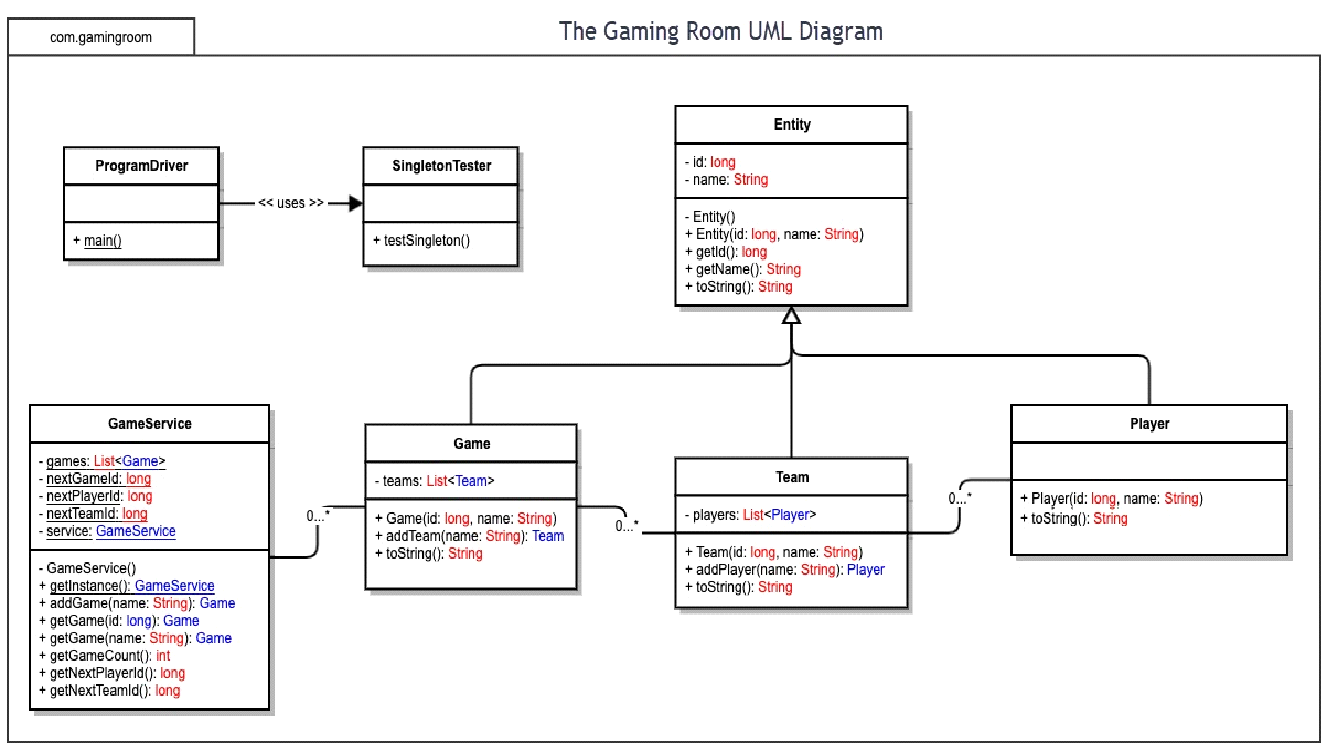
Web-based distribution models require a server or service to host the files for download. They also require servers to maintain online multiplayer connectivity. The client requires the game to be web-based, therefore there needs to be a website to host the game and servers that can communicate with the client on a cloud-based system. Another constraint that constricts development is if the game client is too large, It will cost more resources for the user to support it. We will need to make sure that images and files can be loaded quickly for all platforms equally.

**System Architecture View**

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**

The ProgramDriver class holds the main method and is what starts the program. It uses the SingletonTester class to make sure we have a single instance of the GameService class. The GameService class is a singleton class meaning that only one instance of that class is created and can exist. The Game, Team, and Player classes inherit from the parent class Entity. Each child class of Entity will have an id and name. The GameService class can have any amount of games from the Game class, The Game class can have any amount of teams from the Team class, and the Team class can have any amount of players from the Player class.



**Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| **Server Side** | Mac OS is a very user friendly OS and direct tools for online hosting and managing. Mac is not as modular and slightly less popular than other OSes. | The least common platform for PCs, and the least user friendly. Linux has the most tools for online hosting and management and is very modular. | Windows is the most common platform for personal computers. It’s easy to use yet easily modular. | Mobile Devices have great online environments and many hosting tools. Mobile phones are almost always connected to the internet. |
| **Client Side** | Macs have the most expensive hardware and software. Expertise and familiarity are needed with this OS, but can be gained over time.  Mac is Unix based.  Mac comes with the Safari browser | Most developers are familiar with Linux or can quickly learn. The available tools for development make it an easy and cost-efficient platform. Linux is Unix based. | Windows is the most available and has many tools available, some expensive, others not. Expertise and familiarity are needed with this OS, but that is commonly found. Windows uses Edge browser | Mobile devices come in either android or IOS. Android is based on Linux and IOS is based on mac. They share similarities with their parent language but are streamlined and light. |
| **Development Tools** | Some Mac tools are Homebrew for package management and  Tower, a git client for mac.  Mac OS uses C and C++. | Linux has lots of tools built into the command line. Tools like  Vim- command line, and  Netbeans are useful.  Linux uses C, C++, Java, and Python. | Some great Windows development tools are Eclipse, Filezilla, and GitKraken. Windows uses C#. | Tools for mobile platforms include FlexiHub for IOS devices, and Android has many available SDKs for use. Android uses Java and IOS uses swift. |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: I recommend we adapt our game Draw It Or Lose It to fit on the windows platform. Windows is a common platform that is easy to develop for and adaptable.
* **Operating Systems Architectures**: Windows system architecture is made of two parts, user mode and kernel mode. User mode interacts with the kernel and programs by using an environment subsystem. Windows has many subsystems available to enhance application compatibility. The kernel interacts with the hardware and memory and protects the system’s protected memory.
* **Storage Management**: Windows has a built-in storage management API to manage a large number of storage configurations. This API provides administrative capabilities and storage provisioning. Applications on windows use a storage management provider (SMP) to support this API. With this architecture, storage can remain on one computer or a cloud system and still remain connected to the main system.
* **Memory Management**: Windows manages memory by giving each process its own virtual address space. The virtual address space is separated into two partitions, one partition for user processes to use, and another reserved for the system. Each process only has access to its own memory, unless shared, to prevent corruption.
* **Distributed Systems and Networks**: Draw It or Lose It has a web-based system that can be accessed from the browser. In order for users to connect to the system, the main host server has to be running. Users can connect to the host and create games with each other, even on different platforms. In order for the game to work correctly, players must be able to maintain a stable connection.
* **Security**: In order to protect the system and users, the principle of least privilege should be observed. This means creating a digital identity for each user and restricting that user to only the functions and commands necessary to complete their tasks. This will prevent users from accessing restricted data or misusing the system. The game itself will be able to do this across all platforms making it an easy and effective method of security.