Draw It or Lose It

**CS 230 Project Software Design Template**

Version 1.2

**Table of Contents**

## Document Revision History

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | 09/18/22 | Vidall Williams | Doing executive summary, design constraints, and Domain Model sections |
| 1.1 | 10/2/22 | Vidall Williams | Adding evaluation to report |
| 1.2 | 10/15/22 | Vidall Williams | Offering final recommendations to client |

**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

The problem being addressed is the creation of a multi platform web client version of the client's android only game "Draw It or Lose it". This requires a full rewrite of the frontend codebase into something compatible with Web Browsers/Web Clients. The recommended language is javascript with HTML canvas.

## Design Constraints

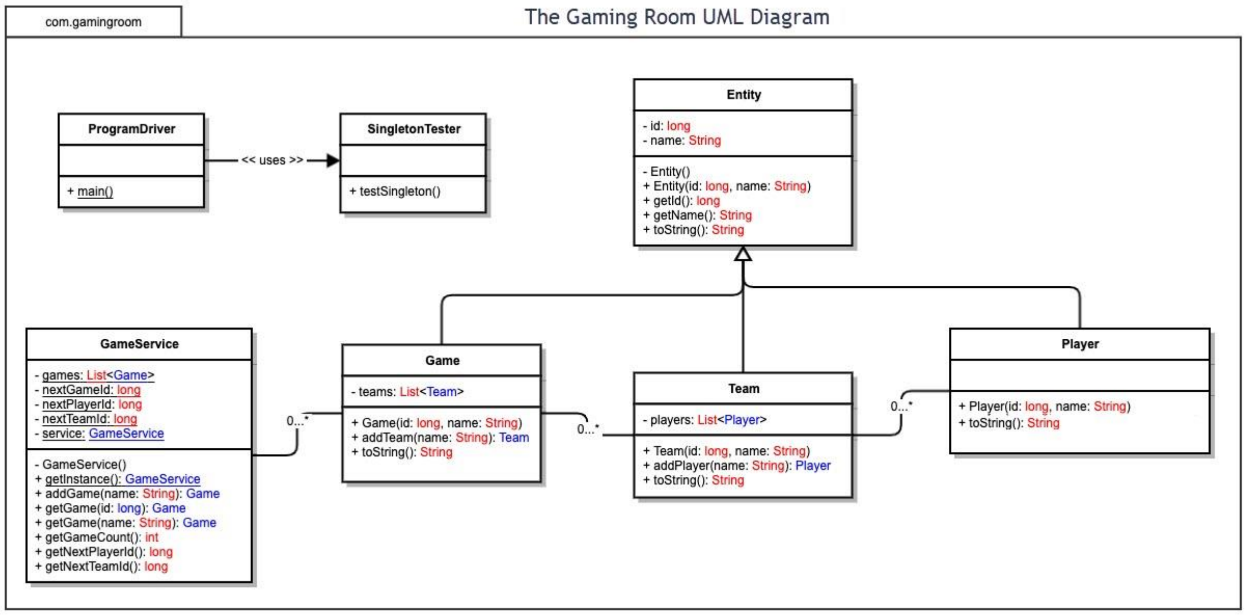
The design constraints are mostly compatibility based. There currently is not a way to directly run android games in a web client. This limits the frontend language choice. Since it is supposed to be a replica of the android game there actual visual design must use the same assets so that the user base isn't alienated. This means if an asset isn't compatible with javascript it must be remade.

## 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 UML Diagram contains a programDriver and SingletonTestor class. The have an association style relationship where SingletonTester is used by the ProgramDriver which also contains the main class. Entity Class has 3 children (game, team, and player) which are connected to it with the open arrow signifying they are indeed Inheritance classes. They all practice encapsulation with each one containing getter methods to retrieve private data instead of exposing the data. The gameservice class is associated with the game, team, and player classes. They are also associated with each other. In order each class from left to right can be created outside the class and contain many of the class to the right of it but cannot be contained in any class to the left. This means one GameService class can contain many Game classes, one Game class can contain many Team classes, one Team class can contain many player classes. However, game cannot contain any gameService classes, team cannot contain any game classes, and player cannot contain any team classes.



## Evaluation

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** | Very expensive hardware and software licenses. Limited to apple ecosystem. Easy to use gui and high stability and security due to its unix roots. | Cheap and very versatile. Can run on most hardware. Can scale almost infinitely. High degree of customizability and versatility. Lots of people working on lots of problems leading to more security coverage. Extremely stable due to unix. Has a steep learning curve. Slower updates. More research required to find specific fixes | Limited to Microsoft services and languages like .NET. Somewhat expensive because of licenses and cost of windows and pcs that can run its server software. Familiarity as a bonus. Windows server looks like regular windows design language. Friendly to people with non technical backgrounds. Dedicated support from Microsoft. | Not easy to near impossible to setup on ios. Lots of power required and constant source of energy needed which is hard to reliably provide through mobile. Not recommended |
| **Client Side** | If done through a website only someone proficient in Front End web development necessary. HTML, CSS, and Javascript. | Relatively cheap and uncomplicated. If done through a website only someone proficient in Front End web development necessary. HTML, CSS, and Javascript. | Relatively cheap and uncomplicated. If done through a website only someone proficient in Front End web development necessary. HTML, CSS, and Javascript. | Can be relatively cheap. Front End Web developer or developers with high level react native experience can create the game on both platforms simultaneously on cheap hardware. Must also be published to respective app stores which costs money |

|  |  |  |  |  |
|---|---|---|---|---|
| **Development Tools** | Visual Studio Code with HTML, CSS, and Javascript. Frameworks like react can also be used. React pairs well with react native. | Visual Studio Code with HTML, CSS, and Javascript. Frameworks like react can also be used. React pairs well with react native. | Visual Studio Code with HTML, CSS, and Javascript. Frameworks like react can also be used. React pairs well with react native. | Visual Studio Code with HTML, CSS, and Javascript paired with Visual Studio Code. Can be made for mobile with react native or dedicated languages swift(ios) and java(android). Android requires eclipse for testing. Swift development requires a Virtual Machine to run mac OS |

**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**: The operating platform needs to be simple enough to have inexperienced developers able to work on it and support server need for an online multiplayer game. The best fit for this is Amazon Web Services. It has the most market share and a plethora of services for developers who cant set them up on their own.

2. **Operating Systems Architectures**: The best operating system architecture for porting the game is the Windows Operating System. It has a very easy to use file system, is the most popular platform by far, and is overall easy to work with and supports a plethora of languages and code environments.

3. **Storage Management**: Since the operating platform being used is Amazon Web Services it would be best to take advantage of one of the services they provide. Amazon offers a serverless cloud storage model that makes it possible to implement security and maintenance services with the knowledge your data is backed by multiple services and professionals. Keeping a hard copy of the server using a traditional file system is also recommended so that youre not 100% dependent on an outside entity and have a backup of the game.

4. **Memory Management**: Amazon Web Services provides lots of memory management services so that the backend doesn't get overwhelmed with requests from the front end. The front end the user will be interacting with will need to utilize their systems RAM. This will allow fast reads and writes so that the visuals and actual player data can be rendered to the player smoothly and in time.

5. **Distributed Systems and Networks**: Amazon Web Services provides a suite of services that include security and authentication with the backend. As long as proper authentication is provided likely through an api it is possible to access the backend no matter what platform the front end is using. This allows the files for the game to accessed across multiple platforms and lets user data be updated very easily. Since the game is running on a large corporate platform like amazon services if the server you've been assigned in the cloud ever goes down another is immediately available for use. This keeps connectivity at maximum and true outages at a minimum.

6. **Security**: The best choice for a security system is the role based security system. This prevents players and admins from accessing data they shouldn't be able to through a login system. I suggest implementing some form of two factor authentication into the game. All verification of passwords should be handled on the backend where malicious users cannot meddle with the data