Draw It or Lose it

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

## Document Revision History

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | 01/29/2023 | Daniel Dobbs | Added executive summary, design constraints, and domain model. |

**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 Gaming Room wants to developer a web-based game that servers multiple platforms based on their game Draw it or Lost it. The staff at The Gaming Room does not know how to set up a web-based development and will need help streamlining the development. The game needs to have the ability to have one or more teams involved, in addition to multiple players on each team. The game and team names must be unique to allow users to check whether the name is in use when choosing a team name. Also only one instance of the game can exist in memory at any given time which can be achieved by a singleton design pattern. In addition to this to be able to assure there is no duplicate players or teams, we can turn these into a singleton pattern using a list with an iterator pattern.

## Requirements

< Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client's business and technical requirements in a clear and concise manner.>*

## Design Constraints

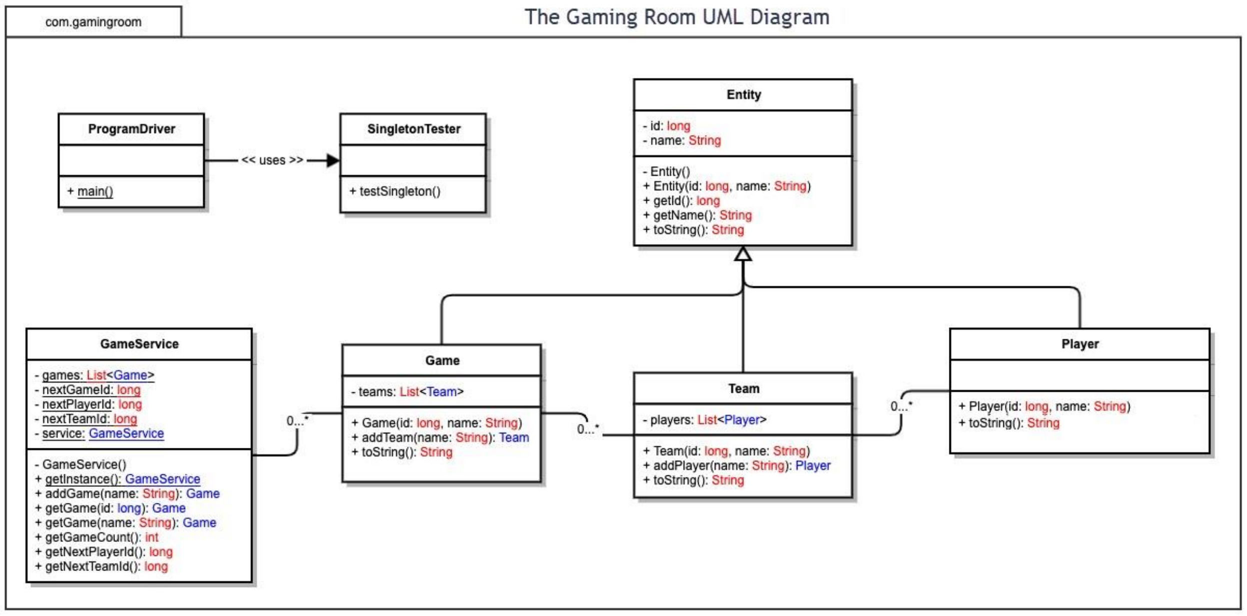
The design constraints of a web-based distributed app involve the development of the game application itself. For example, if the number of users increases the staff needs to consider scalability of the backend services. The multi-platform of web-based is great but it means we have to configure the application to work on every platform and make sure it runs correctly.

## 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

Encapsulation: There is encapsulation here as every time there is private modifier that means that the classes is hiding the data and not exposing it.
Abstraction: There is abstraction through the use of Entity while not an abstract class its methods get override by its child's Team, Game, and Player but all the logic for these methods is in entity.
Inheritance: This is seen by that Team, Game, and Player all inherit from the Entity class and this is shown by the arrow from Entity.
Polymorphism: This is shown to be used in the GameService class we have multiple methods named the same thing but have different parameters to do find a different way to the solution of the method. For example, we have getGame with an input of both a long and String to find a Game by an id on one method and Game by a name on the other method.



## 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** | Mac is propriety software made by Apple meaning to its very hard to get it installed on any computer unless it an apple product. There are also limited ranges of hardware. Mac servers are also going to be the most expensive option as the hardware itself is usually more expensive. | Linux is the best when choosing a server as it is open source and free. This means it can be installed on any hardware brining down costs. Also, Linux is widely used in the professional field so getting outside help would be easier as finding an expert in this should be easy to find. Linux is also easily customizable and very flexible. Linux also gives you full control of the platform making it so can be very reliable as a server. | Windows has a steeper upfront cost than Linux but lower cost than Mac. Windows is also more user friendly as it has a better CLI than Linux and lot more graphic interfaces. The one problem windows has is it has more potential for security issues than the other platform since it's the most widely used consumer platform. It is also the most targeted platform so having to pay for security is an issue too. | Mobile devices can be used as a personal server or file server, but they are not equipped to handle multiuser serving. The hardware is limited with ram being usually 8GB and they are not very scalable. Costs are unknown as there are no web hosting tools that provide this service so it would have to be bult in house. |

| | | | | |
|---|---|---|---|---|
| **Client Side** | Mac has two main platforms with on being based on x64/86 like windows and also having their new architecture based on ARM. This makes developing for mac very different as mac in in the middle of converting their architecture and if for example we only support ARM macs we might be losing out on significant chunk of mac's market share depending on the time the product releases.<br>Also, to work on mac software you must develop using a mac computer running their own IDE. | Development in Linux should be straightforward as their platform is open source. A lot of developers know how to use Linux so getting new talent on Linux would be relativity easy. Also, in recent times due to companies like Valve, AMD, and a few others Linux has been a better platform to develop on and they've made it easier for developer to use. The only downside of Linux is that the market share is relativity small. | Windows development should be as straightforward as Linux. The windows platform has the highest market share of the PC market so making this our top priority is the best. Since windows has a high market share this makes it easier to find talent that has experience developing on a windows platform. | Mobile devices for client application for iOS and Android are pretty straightforward.<br><br>Android code is going to be similar to Linux as it using the same language so basis off that might help.<br><br>iOS is going to be like coding with mac ARM in mind as lot of games on the apple store that work on an newer iPads run the same technology as the macs and the iPhones run on a similar architecture too. So, if mac client is made it should help tremendously in developing an iOS client. |

| | | | | |
|---|---|---|---|---|
| **Development Tools** | Macs use Objective-C and SWIFT for development languages.<br><br>Xcode is the most common IDE used for Mac development.<br><br>XCode license is $99 per year per developer. | Linux development may take the form of C/C++, Java, or Python.<br><br>Any IDE can be used some commons ones are, PyCharm and the JetBrains IDEs like IntelliJ IDEA. | Windows development can take the same form as Linux but also can be developed using C# and .NET.<br><br>Microsoft has two popular IDEs one being Visual Studio and the other being Visual Studio Code.<br><br>Visual Studio may take some expense if need to use extra features. | Android SDK is Java based and the most widely used Android IDE is Android studio which is developed by Google as the official tool. Kotlin has recently gained support as android language.<br><br>iOS uses Objective-C and SWIFT and are developed using XCode like with Mac.<br><br>Xcode license is $99 per year per developer. |

## <u>**Recommendations**</u>

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 should be chosen for the server as it is open source and very powerful platform when used correctly.

2. **Operating Systems Architectures**: Linux always users to develop and setup the environment the way the want with its command line interface that is extremely powerful. In addition to having a wide range of support already in the field so getting an expert on this would be easy. Linux is also very reliable by giving the user full control. Linux is based on Unix architecture.

3. **Storage Management**: Can use an HDD or SSD but I recommend using SSD as it is much faster and not that much more expensive. To optimize storage and cut down on physical components I believe we should have backup storage using a cloud system and using SQL for it.

4. **Memory Management**: Linux provides various memory management techniques, including virtual memory, swapping, and shared memory. Linux will use virtual memory to management the memory of the software. Virtual memory allows the OS to allocate more memory to the application than physical memory temporarily.

5. **Distributed Systems and Networks**: To enable communication between different platforms, I recommend implementing a distributed system architecture using a messaging queue system such as Apache Kafka. Kafka allows for real-time data streaming and processing, enabling seamless communication between different devices and platforms. The Kafka messaging system is highly scalable, fault-tolerant, and has a low latency, making it ideal for a gamming platform. The network interconnecting the devices should be robust and reliable, with high bandwidth and low latency. The recommended network architecture should have redundancy built-in to ensure the game is always available.

6. **Security**: I recommend using encryption to secure data in transit and at rest. The Linux platform has various built in security features including a secure shell protocol for secure remote access, firewalls for network security, and audit logging for tracking and monitoring system activities. The game should also implement secure user authentication and access control mechanisms to ensure that only authorized users have access to sensitive data. Regular security audits and updates should be performed to ensure continued security of the platform.