

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/21/2024 | Aidan Sayles | First creation and initial changes |
| 1.1 | 07/30/2024 | Aidan Sayles | Second Project Changes |
| 1.2 | 08/13/2024 | Aidan Sayles | Recommendations added |

**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)

The Gaming Room has hired Creative Technology Solutions to design and program a web-based game, taken from their android version, Draw It or Lose It. Each game will have a multitude of players separated into multiple teams, which all must be unique having only one instance each. This is why a singleton pattern has been adopted to alleviate this issue.

## 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.>*

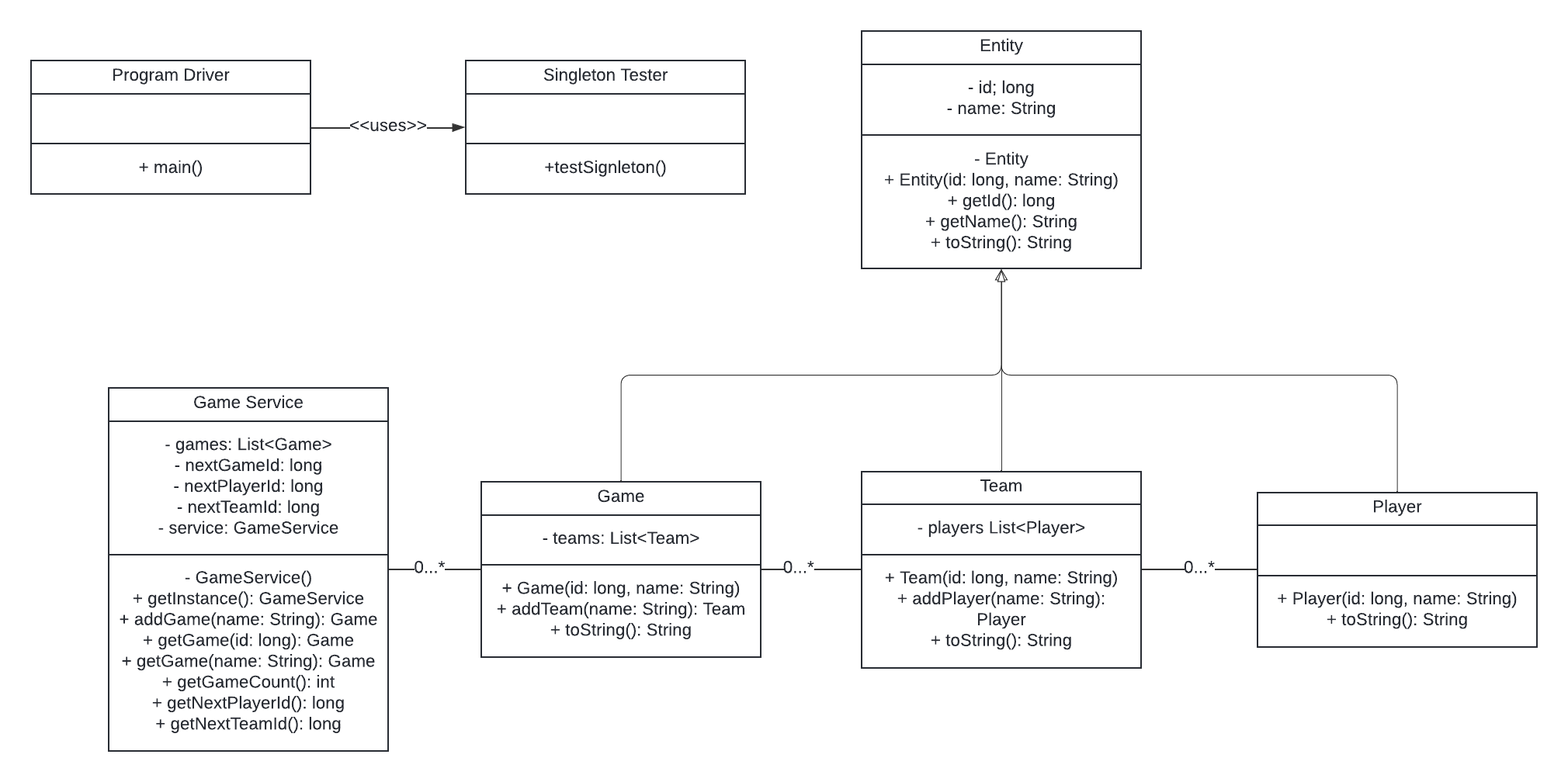
## [Design Constraints](#_2et92p0)

It is assumed that the servers must be the same as the ones for their app, and it may be required that the game be cross-platform, allowing players on the web-based version to play with those on the app. If any coding can be taken from the original app that would save time and money and it would be useful to do rather than coding everything from scratch.

## [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)

****

The Program Driver initiates the Game Service. Using the singleton design pattern, it is designed that only one instance of Game Service may exist at a time. Once the Game Service is created, the Program Driver uses the Game Service’s methods to create the players, teams, and games. Polymorphism is displayed by these three classes all extending the Entity class. Abstraction is best showed by the private methods such as the non-overloaded constructor being private to not allow any objects with null variables.

## [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** | AWS allows for EC2 M1, M1 Ultra, M2, and M2 ProMac instances to enable Apple Silicon macOS environments making AWS the best route for Mac servers. | Due to its open-source nature, it will probably be cheaper to create and maintain.  Amazon and Google cloud work better with Linux. | License costs are higher.  Hosting platforms are more limited.  Because Windows servers are GUI based, there is ease of use between the servers and the office applications. | Costs are hard to estimate.  They can be webservers or fileservers, but they aren’t able to provide multi-user server.  Much less powerful hardware. |
| **Client Side** | Mac computer is needed.  Mac uses much lesser-known programming languages. | Most popular coding languages work in Linux.  Multi-user support is available.  Very small margin of the customer base uses Linux. | Uses common programming languages, not as common as Linux, but more well known than Mac languages.  Multi-user support is available.  Windows is used be most of the customer base. | Android uses common programming languages, however iOS does not.  Massive customer base. |
| **Development Tools** | XCode IDE is the most common for Mac.  Objective-C and SWIFT are the most popular programming languages for Mac. | PyCharm and Eclipse are popular and free.  C/C++, Java, and Python are the most popular languages for Linux. | Visual Studio Code is extremely popular. | Android Studio is the most popular IDE for Androids.  XCode is the most popular IDE for iOS. |

## 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**: Servers should be Linux based due to their ease, scalability, and low cost. However, if the client doesn’t own hardware, AWS would be a great option for server, which also uses Linux. The front-end should most likely be Windows as that would expand the customer basis the most.
2. **Operating Systems Architectures**: The game environment should be managed by the servers while the front-end should do the rendering. This would be cheaper, however it would increase latency. This increase in latency is not significant especially due to the game’s nature, and therefore that downfall is quite negligible.
3. **Storage Management**: If no hardware is owned by the company, AWS offers great storage management for relatively low cost and high dependability.
4. **Memory Management**: Linux memory management is extremely efficient and not a worry. However, if mobile devices are chosen, they have very minimal capabilities. This is another reason that Windows would be an excellent choice.
5. **Distributed Systems and Networks**: Due to AWS’s reliability and outage prevention, it would be the system to make sure the game stays available at all times.
6. **Security**: Information should be encrypted and not sent out or available to every device in the game server. Also, no unnecessary information should be collected or transmitted in any way. Linux and AWS also offer great server security options.