

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 | 08/13/2024 | Christopher Gaceta | Original Draft |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, would like to develop a web-based game that serves not only their Android user base, but multiple platforms based on their current game, Draw It or Lose it. CTS was tasked to help streamline the development and help develop the game application. The game’s software requirements that have been requested must be met for this application.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. Unique identifiers for each instance of a game, team or player must be created.

## [Design Constraints](#_2et92p0)

Due to the game already published on the Android platform, CTS will need to utilize a full stack for a distributed system. Distributed systems are used and widely adopted by social media platforms, video streaming and gaming. Due to Android using Java to develop games, we would need to find a way to utilize Java when deploying the web-based game and ensure it runs on multiple platforms.

## [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)

We are utilizing a Singleton pattern to ensure that there is one GameService existing at any time. GameService is the instance in the GameService class which holds the single instance of the class. Private constructor is created to prevent instantiation from outside the class.

GameService class is able to call the method addGame which then creates a new game. The addTeam() method allows a team name to be stored as a String. A player made be added to the team using addPlayer() method and the Player class stores the player. Game, Team and Player classes all fall under the Entity Class and stores the id and name private.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## 

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Macs have a program called Apache built-in and can host a web application. Hardware can be expensive as well as setting up security such as a firewall. | If a client wants to host using Linux, they can utilize Virtual Private Servers and pay for hosting. They can pick from different distributions such as Ubuntu and Debian. Files would have to be uploaded and the server configured. | Windows Servers can be utilized to host the web based application. Internet Information Services would need to be installed as well as security needs to be implemented via Firewall. | Mobile devices are unable to host web based applications due to the limitations imposed on the product itself. Due to hardware restraints and capabilities, this would not be feasible |
| **Client Side** | If the client would like to develop on a Mac, they can use a Mac computer and utilize Xcode. | Cost can be fairly cheap as there are plenty of Linux VPS services available. Learning Linux isn’t the hardest and is widely supported via many operating systems. | Windows programs would have to utilize C# for developing applications on the .net framework. | Due to different operating systems for mobile devices, we would have to develop using Android software such as Android SDK or Swift for iPhones. |
| **Development Tools** | Xcode is the IDE used which supports the Swift programming language. Xcode allows the developers to utilize C, Java, and other languages if needed. | IDE’s such as Visual Studio Code and Eclipse are easily available and used to build software in Java for these web-based applications. | Visual Studio can be used to develop Microsoft programs due to being created by Microsoft. Eclipse can also be used on Windows as well. | We can use android studio or XCODE to develop for the mobile platforms. |

## 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**: CTS recommends utilizing Linux Servers due to the affordability and flexibility of the servers. The Gaming Room can pay for web hosting to handle the bandwidth needed for people to utilize the application.
2. **Operating Systems Architectures**: Cloud based systems such as AWS or Amazon Webb Services can be utilized as it is scalable due to the pricing of the service. AWS offers volume based discounts as well as pay as you go depending on business needs.
3. **Storage Management**: AWS Cloud-Based Architecture allows data to be stored on servers as well as the option for Centralized backup across AWS services.
4. **Memory Management**: Due to the nature of the AWS Cloud-Based services, there are virtual machines which can be utilized to optimize each instance of the game created by the application based on the workloads. The company can even define limits for memory storage.
5. **Distributed Systems and Networks**: Having consistent connectivity to the application will be important for players who are trying to connect to the game. Being a cloud-based architecture, AWS can ensure high server uptime and prevent outages.
6. **Security**: The client will have high level security for their servers that are hosting the web based application. The cloud infrastructure itself such as the hardware servers and networking will be controlled by AWS. The client can install VPN and Firewalls to control traffic and what is being sent out of their network. Users will be able to have limited access besides creating a game, team name or player name.