

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 4**](#_Toc115077324)

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/19/23 | Anna Lauffer | Initial submission of Design Document |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants their Android gaming app, Draw It or Lose It, to be developed as a multi-platform web application. The new web version will have the identical functionality of the mobile app including flexible numbers of teams and players. However, this project will need to be coded from scratch – based on the Android app code but not reusing it – and will utilize both OOP principles and established design patterns.

## Requirements

To duplicate the Android app functionality in the web-based version:

- The code must be cross-platform.

- Each instance of the game must be able to have one or more teams.

- Each team needs to have multiple players.

- Each individual game and team must have a unique name and id.

- Only one game instance may exist in memory at a time.

## [Design Constraints](#_2et92p0)

- Use Java for the backend and HTML/CSS/JavaScript for the frontend to code the web application.

- Use the Singleton design pattern to ensure that only one instance of Draw It or Lose It exists in memory at a time.

- Use the Iterator design pattern to ensure that duplicate names and/or IDs cannot exist for game, team, or player instances.

- Use OOP principles to create and organize objects, assigning player objects to team objects and team objects to game objects.

- Determine internet connection speed needed to run the web app and develop graphics quality to produce the client’s desired minimum speed requirement.

- Determine the server computer and hardware requirements based on data storage and memory for the new app.

## [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)

This program is built on the OOP principle of modularity, meaning that it is made up of classes. The ProgramDriver class houses the main method from which the program will be run. ProgramDriver uses the SingletonTester class in a one-way association. The Entity class is the parent class from which the Game, Team, and Player classes all extend, showcasing the OOP principle of inheritance. Having a parent class groups together common attributes and behaviors of the child classes, avoiding repeated code and making any future adjustments to the children easy to manage. Additionally, the inheritance hierarchy represents polymorphism – meaning that all child-type objects are also Entity objects and can be treated as such. The GameService class does not inherit from a parent class, but instead is at the head of a chain of associations (each with a multiplicity of 0 to many): GameService is associated with the Game class, which in turn is associated with the Team class, which is lastly associated with the Player class. The member access symbols (private [-] and public [+]) demonstrate the encapsulation of this program which will prevent the creation of more than one instance of the web app running in memory.

**"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** | - Supports LDAP  - Supports ADP  - Supports cloud  - Less malware/ security threats  - Server OS discontinued (server features in macOS)  - Cost = Mac computer/macOS license | - Supports LDAP (OpenLDAP)  - CData Linux/UNIX ODBC Driver for ADP  - Supports cloud  - Less malware/ security threats  - Server OS options (Ubuntu Server, RHEL, etc.)  - Cost = free to use (open source) | - Supports LDAP (pair w/ AD)  - Supports ADP  - Supports cloud  - Most threats (viruses, worms)  - Server OS (Windows Server 2022)  - Cost = $500 standard license | - Supports LDAP (Virtual LDAP)  - ADP Mobile Solutions  - Supports cloud  - iOS less viruses / Android more  - Android app: Servers Ultimate  - iOS device w/ iSH and Python  - simple sites only | |
| **Client Side** | - Cost = Apple Developer account, below dev tools  - Time ≈ 4 months  - Expertise = REST APIs, HTML5, App Store, dev tools | - Cost = below dev tools  - Time ≈ 4 months  - Expertise = REST APIs, HTML5, Snap Store/Flat Hub, dev tools | - Cost = below dev tools  - Time ≈ 4 months  - Expertise = REST APIs, HTML5, Microsoft Store, dev tools | - Cost = below dev tools  - Time ≈ 4 months  - Expertise = REST APIs, HTML5, Google Play/App Store, dev tools | |
| **Development Tools** | - Visual Studio ($45 a month)  - RAD Studio (C++) ($2600+ one-time)  - Uno Platform (C#/XAML) = free  - Xamarin (C#) = free  - .NET MAIU (C#) = free | - Visual Studio ($45 a month)  - RAD Studio (C++) ($2600+ one-time)  - Uno Platform (C#/XAML) = free | - Visual Studio ($45 a month)  - RAD Studio (C++)  ($2600+ one-time)  - Uno Platform (C#/XAML) = free  - Xamarin (C#) = free  - .NET MAIU (C#) = free | | - Visual Studio ($45 a month)  - RAD Studio (C++)  ($2600+ one-time)  - Uno Platform (C#/XAML) = free  - Xamarin (C#) = free  - .NET MAIU (C#) = free |

## Recommendations

1. **Operating Platform**:

To provide a reliable, affordable solution to the Gaming Room for developing and deploying “Draw It or Lose It” as a multi-platform app, Creative Technology Solutions (CTS) recommends the use of the Ubuntu Server operating system. The most recent release is Ubuntu Server 22.04 LTS (Long-Term Support). This open-source, Linux distribution is free for business use. (Canonical, the publisher of Ubuntu, offers subscriptions for infrastructure and security support which can be considered in the future, if needed).

In order to design the app, CTS recommends the use of the Uno open-source UI platform paired with the Visual Studio IDE. Uno uses C# and XAML on top of .NET to provide high-quality, native performance for Windows, Mac, Android, iOS, and Linux clients. Using Visual Studio and C# to handle the back-end code as well seems the ideal choice, increasing ease of coordination between server- and client-side developers.

1. **Operating Systems Architectures**:

The distributed system CTS is recommending for the “Draw It or Lost It” app is a simple client-server model, with an in-house, dedicated server computer as the web host. This option gives The Gaming Room high performance, intimate IT control, and increased security. Ubuntu Server 22.04 LTS is built on top of the Linux kernel and runs on many computer architectures, including x86-64, ARM (v7 and 64), and IBM zSystems. (Canonical provides kernel updates twice a year.) This provides a lot of flexibility in choosing the physical server machine. Minimum system requirements to operate this OS include 4GB RAM, a 2GHz dual core processor, 25GB disk space, and a 4GB USB drive. CTS recommends a quad core processor for improved multitasking and overall client delivery speed.

1. **Storage Management**:

CTS recommends utilizing flash memory in the server computer via a 1TB solid-state drive (SSD) for storing the Ubuntu Server OS along with all the files needed to deploy the web app (including the 1.6 GB of images for teams to guess). Although SSDs are more expensive per gigabyte to purchase, they significantly out-perform hard-disk drives (HDDs) in speed. Additionally, SSDs do not have issues with disk fragmentation, which is a constant issue in HDDs.

The SSD will also provide storage for the game database of user accounts plus existing teams and players. MySQL is an open-source database systems that Ubuntu supports, and we recommend pairing it with the MySQL Workbench GUI to set up and manage the “Draw It or Lose It” database.

1. **Memory Management**:

RAM is at the core of Linux memory management. All data that the app will need to access from storage is loaded to RAM before being processed and served to the client. Linux takes advantage of the page cache management techniques, which can increase read-access speeds. A page cache allows the OS to store recently accessed files in order to retrieve them more quickly if they are requested again. Certain GUI-related files could end up in the page cache, providing each user faster transitions between rounds of game-play.

Since the server computer for “Draw It or Lose It” will need to service up to 4000 players on 1000 running instances of the app, CTS recommends 16GB of server-grade RAM. Also, instead of loading all 200 image files to RAM for every game instance, the server-side code should be written to create a randomly selected array of 10 images from the library to cover the first 10 rounds. Every time a game completes 10 rounds, a new array of randomly chosen images should be sent to RAM to overwrite the original array.

1. **Distributed Systems and Networks**:

In this client-server model, the Gaming Room’s dedicated server machine will store, process, and web-host the “Draw It and Lose It” app data, serving all client requests to run the app in various browsers. The server will need to be configured with the TCP/IP network protocol. To make certain each browser can interact with the server, a cross-platform REST API should be written with .NET and C#, utilizing HTTP commands and JSON data formatting.

To provide uninterrupted service to clients in the case of power disruptions, the server computer should be connected to a back-up generator.

1. **Security**:

The first line of defense for the “Draw It or Lose It” application will be utilizing the principle of least-privilege in both server management and user system-access. Within the REST API, the use of roles in conjunction with authentication and authorization functions will set this foundation. Access to the server machine, where game user account information will be housed on the MySQL database, should be password protected. As an extra layer of password security, Ubuntu Server 22.04 LTS allows authorized administrators to set up password expiration, forcing everyone with server access to routinely update their passwords. Additionally, Linux-based systems include Netfilter, which is a subsystem for managing network traffic/packets via the iptables user interface or ufw (Uncomplicated Firewall) tool.

In the end, the application will be accessed by clients through username/password authentication and authorization via the browser; the server machine and its database contents will be protected by malicious or unnecessary access through password and firewall configuration already provided by the Ubuntu Server OS.