

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 |
| --- | --- | --- | --- |
| 1.2 | Feb 9, 2022 | Gianna Screen |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room requested that CTS provide a web-based version of their Android application, Draw It or Lose It. Instead of solely serving Android, the customer wants to support multiple platforms.

## [Design Constraints](#_2et92p0)

The customer requires the application be web-based so that more users can access it. Draw It or Lose It features an Android-based deployment that is now in use at The Gaming Room. Since CTS has been asked to expand this to the web, the software must support web-deployment.

## [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 Game, Team, and Player classes are controlled by the Entity class. The four classes that were created—Game, Team, Player, and GameService—utilize one another. To develop the project to our client's specifications, we use the ProgramDriver class. The project can proceed in accordance with the design limitations for each class that is set up, allowing for the simultaneous use of several teams and players in a single gaming session.

**"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)

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** | Although the OS X Server is supported for Mac, locating hosts will be challenging and costly unless the client wishes to buy their own hardware. | Opensource operating systems such as Linux typically have lower maintenance and license expenses than closed operating systems like windows. | Windows servers are ideal since they are GUI based and support a wide variety of office applications, offering a great deal of familiarity. In contrast to Linux, license charges are frequently very costly per user. | Although it’s possible for mobile devices to be used as personal webservers or file servers, they lack the capacity to serve several users at once. They are not expandable like blade servers and often have more constrained hardware. |
| **Client Side** | A Mac computer running the most recent version of XCode is required to create for Macs. | The GNU/Linux operating system supports multiple users. The lack of widespread use means that GNU/Linux development may not be of much benefit. | C# or.NET, which are both commonly used, are frequently used in Windows development. Because of this, the creation of a Windows client program would not present many obstacles. | Mobile devices are not built to support multiple users. Nevertheless, creating a client application for iOS or Android is simple. |
| **Development Tools** | For programming languages, Mac uses SWIFT and Objective-C. The most widely used IDE for Mac development is XCode. | Python, Java, or C/C++ can all be used in Linux development. | C# and.NET are primarily used in the development of Windows applications. | The most popular Android IDE is Android Studio, which was created by Google as the primary development platform. Also, android SDK is Java-based. |

## 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**: Linux-Dependent servers are the recommended choice despite whatever software is selected. In comparison to Windows servers, which can restrict access to data centers, Linux servers have lower license prices. Linux provides significant functionality and security.
2. **Operating Systems Architectures**: A backend server that controls the gameplay and client-side graphics is the suggested architecture. There is no requirement that there be a short response time between the frontend and backend.
3. **Storage Management**: There is no need to choose between HDD and SSD as a storage medium unless the Game Room intends on purchasing their own hardware. Depending on whether the application uses caching or client-side rendering, either HDDs or SSDs should be able to meet its performance requirements.
4. **Memory Management**: Client-side rendering necessitates a minimum amount of RAM on the server but with a newer structure that employs microservices, the price will increase with the as users increase. The amount of RAM used by the client should likewise be kept to a minimum because only one or two images need to be kept in memory at any given time.
5. **Distributed Systems and Networks**: Several applications are being developed using cloud native architectures due to concerns about reliability. To avoid widespread failures, cloud providers can duplicate and move services between several implementations/updates.
6. **Security**: Position-based authorization will be a major component of the security system. This implies that in order to effectively administer the responsibilities and accounts, an administrative interface must be developed. Also, A firewall will likely be an additional component of the server.