

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 | 03/26/23 | Margaret Snyder | First Version |
| 1.1 | 4/9/23 | Margaret Snyder | Version 1.1 |
| 1.2 | 4/23/23 | Margaret Snyder | Version 1.2 |

**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 would like to develop a web-based version of their game Draw It or Lose It. The game is currently only available as an Android application, so they need CTS to set up an environment that can serve multiple platforms. The game needs to have the ability to have one or more teams involved and each team will have multiple players. The game and team names require unique names and only one instance of the game can exist at one time.

## 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)

The current Draw It or Lose It game is designed for Android devices, so the new application will need to be developed for a website. Android applications typically use Java as the language to build apps, so using Java to develop the website will make the process easier, as much of the code can be reused.

## [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 main driver class is ProgramDriver and initiates the creation of games, teams, and players. This process runs through the GameService class. Once GameService is running, ProgramDriver can call addGame to add a new Game (as long as there is not another game by that name). Then that Game is added to the list games.

Once a Game is created, a Team can be created and added to that Game with the addTeam() method within the Game class. The addTeam() method checks the current list of teams within that Game to prevent duplicates, then creates the Team if that name is not in the list, and adds the Team to the list of teams in the Game.

Once a Team is created, a Player can be created and added to that Team in that Game with the addPlayer() method within the Team class. The addPlayer() method checks the current list of players within that team to prevent duplicates, then creates the Player if that name is not on the list, and adds the Player to the list of players in the Team of that Game.

The Game, Team, and Player classes are subclasses of Entity and inherit their protected attributes (id & name). The inheritance is an example of an object-oriented program technique. The overloading of constructors within the classes is an example of polymorphism.

**"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** | Currently there is no server offered by Mac. The Mac OS X Server was discontinued on April 21, 2022.  There are websites that offer to host servers, but adding a third party into the business increases chances for security breaches and limits a company’s ability to control their own server. | Linux is free and opensource, so it is a cost-effective option for hosting a website. It is also popular with top cloud providers.  It also gives more freedom in terms of development of the source code, which is not an option for proprietary software. | Windows is widely used and has a lot of applications that can be used on the server.  Windows licenses are much more expensive than Linux but are a very popular option for those who don’t need software catered to their use. | Mobile devices can be a server for personal use or for small websites, but they are not designed with the bandwidth to handle multiple users. |
| **Client Side** | In order to develop for Macs, you will need to use the XCode IDE on a mac computer.  The SDK for macOS is in Swift or Objective-C, which are not common languages.  Macs only account for 15% of operating systems on the market, so it | Linux can be developed in some of the more popular languages (Python, Java, C/C++).  The market share for Linux is just under 3%, so there is not a lot of usage of Linux among users. | Windows typically is developed with C# or .NET. Windows has also been offering it’s server for multiple users for a long time, as it is a tried and true plaform.  Windows make up 75% of the operating systems on the market, so developing for these systems would cover the majority of desktop users. | Mobile devices wouldn’t be able to handle multiple clients as they are not designed for that much usage. |
| **Development Tools** | Swift is the most popular language for Mac, but Objective-C and C++ are also used.  XCode is the official IDE of macOS, as it was developed by Apple. It is free to download and then costs fees for uploading to the App Store. | Linux can be developed in many different languages, including Java, Python, and C/C++.  There are many free IDEs that support Linux development (i.e., Eclipse, which can handle all the languages mentioned above). | Windows typically uses C# and .NET for development.  Visual Sudio is an IDE offered by Microsoft and is widely popular. Businesses can utilize its services for $45-$250 a month, depending on the company’s needs. | Android’s can be written in Java, Kotlin, and C++. The Android SDK uses Java and is the most popular IDE.  iOS uses Swift, like it’s desktop counterpart, and can be developed in XCode. |

## 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 is the most common server platform as it is an extremely cost-effective option and offers a lot of flexibility in development. It is the best server for this application and is not limited in the ways iOS and Windows are. It offers great security and as it is widely popular, there are many resources that can be used in the development and maintenance of the application. The frontend of the application will be written in multiple languages to support the different user operating systems. SWIFT will be used for iOS and Java will be used for Android.

1. **Operating Systems Architectures**:

The architecture recommended for this operating platform would be divided into two parts: a backend server to handle the main aspects of the game and the frontend image rendering for the client facing side. The backend should run with a more modern cloud-based service, such as Docker, as it would allow for scalability.

Having the frontend render the images takes some of the work off of the server, which would save on costs for the data center. It would also ensure a smooth gameplay experience if the images are preloaded into a cache, so the user wouldn’t be waiting for images to load. The game is timed, so any delays will ruin the user experience and could lead to loses.

1. **Storage Management**:

Draw It or Lose It contains a memory bank of 200 images, so the main focus on memory management will be with the user data. Both HDD and SSD will offer the performance needs that the game requires, so it is up to the client which they would prefer. SSD is more expensive, but faster than HDD. The game does not require the speed that an SSD offers, so either would be sufficient.

1. **Memory Management**:

For the backend, Linux uses demand paging which requires less memory usage and won’t load pages that aren’t necessary into the memory. The page replacement is based on an algorithm called the Least Recently Used algorithm. For the frontend, minimum RAM will be required from the user as the rendering will be done on their devices. It will require minimal RAM as we are only rendering images and not many will be cached at a time, but there will need to be space on the user’s device.

1. **Distributed Systems and Networks**:

A cloud service should be used to communicate between the frontend and the backend of the app. Many applications are utilizing cloud-based architectures to help prevent outages to maintain user satisfaction. The frontend and backend can communicate through RESTful APIs to keep the data secure.

1. **Security**:

Security should be Role-based so that only certain users have complete access to the entire game, while users will only have access to their own game accounts through the front end. This will require an interface to retrieve entitlements for each user based on their access level. The users can access their own accounts and use them to create new games or change their individual account settings. Users will not be allowed to have access to the backend.

A firewall should be installed on the server to protect user account information and to protect the code. The game does not require or contain any confidential information, so a basic firewall is sufficient.