

Draw It or Lose It Web

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/17/22 | Raphael Coloma | Prototype build to ensure client requirements are met. Environment was established allowing multiple game, team, and player instances. Unique names and IDs are issued on creation. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room would like to develop a web-based version of their Android app, Draw It or Lose It, to serve multiple platforms. They have hired Creative Technology Solutions to set up the environment and facilitate its development. The Game application, Draw It or Lose It Web, can have multiple games but allow for only one instance to exist in memory at any given time. A game must have the ability to support one or multiple teams and each team can consist of one or more players. Game and team names must be unique. The Game application will check validity of game/team names on creation.

Draw It or Lose It Web will implement a singleton design pattern to ensure only one game instance is active at a time. Game, team, and player data will exist in their respective array lists. As new elements are added to the arrays, the Game application will implement iteration patterns to compare the new data to existing data to ensure names and IDs are unique.

## [Design Constraints](#_2et92p0)

Migrating from an Android only app to a web-based app to support multiple platforms will be challenging. Android only apps are installed on the device where its local data will be stored. Since the Game application will be web-based, it is important to address where the data will be stored and hosted. The host site will need to allow for large amounts of data traffic between the server and clients.

To support multiple platforms, we must consider how the platform will interpret the data it receives from the Game application. Will a shell program be necessary to translate the data for the platform or will the Game application be browser based with the processing handled by the server? We will be developing Draw It or Lose It Web in Java to utilizing its object-oriented programing features. Not all platforms are optimized for Java and thus this will need to be addressed.

It is expected that the web-based Game application will be an upgrade to its Android app counterpart. It is best to determine any improvement/features that will be implemented, if at all. Will Creative Technology Solutions be developing those as well? Will we be expected to maintain the Game application?

## [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 application consists of six classes and a SingletonTester class to test the application. The Game, Team, and Player class inherit from the Entity class and are instances of it. This is represented by the arrows coming from their classes to the Entity Class. The relationship between the GameService Class, Game Class, Team Class and Player Class is zero to many – Zero to many players to a team, team to a game, game to a service, but only in that manner. How they are connected in the UML shows their relationship to one another.

The Classes hold their respective data, but also its relation to the other classes. The Entity class is a base class introduced to hold common attributes and behaviors. The Team Class holds the Team name and ID as well as the players part of the team. The Game Class holds the Game name and ID, but also the teams participating in the game. The GameService Class holds the instances of the games.

It is important to note the methods that are private and public in each class. This is an example of encapsulation and ensures that only the data that is necessary for the other classes to operate is seen. The array lists for the respective classes, the next iteration identifiers for the elements, and the GameService instance are private to ensure that data is not altered while the application is running and ultimately cause errors.

**"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** | Seamless interconnectivity with other Apple products  Inherent Unix command environment.  Accustomed Terminal commands to access/manipulate servers.  First Party exclusivity – Apple software are optimized for Apple hardware. This limits integration of third-party support  macOS Server has been discontinued, but many of the functions have been integrated in macOS. Licensing will be high as OS can only be run on Apple computers. | Offers optimal system logic for web server security since it prevents unwanted operations while fostering vigilance.  Inherent Unix command environment.  Free Open-source applications offered – Reliant on community support.  File System allows easier access with servers and organizing their file structures  Linux OS is free and requires no licensing costs | Powershell and Windows Terminal for command line access to server. Not as intuitive as macOS or Linux – Environment setup can be cumbersome.  First party support with Windows server applications.  I/O operations are slower than macOS or Linux – due to file handles. Significant performance decrease with file heavy projects.  Licensing costs can be expected to use Windows Server OS. | Though possible, it is not the optimal means of running hosting a web-based software application.  The range of hardware specifications on mobile devices is vast and performance can be sporadic. |
| **Client Side** | Access to a wide range of  supported  web browsers that offer many  tools/plugins for developers and clients.  Data can be cached, and code can be executed on client to improve application performance  Client will utilize device network connection. Minimum data transfer rates can be expected and optimized for use with application.  Proper use of Web browser APIs is required to ensure compatibility and communication.  Applications must be reviewed and approved by Apple. | Access to a wide range of  supported  web browsers that offer many  tools/plugins for developers and clients.  Data can be cached, and code can be executed on client to improve application performance  Proper use of Web browser APIs is required to ensure compatibility and communication. | Access to a wide range of  supported  web browsers that offer many  tools/plugins for developers and clients.  Data can be cached, and code can be executed on client to improve application performance  Proper use of Web browser APIs is required to ensure compatibility and communication. | Mobile browsers lack plugin support for media types.  All operations must be executed on the server. Mobile processing will vary between devices.  Data compression will be necessary to accommodate mobile data transfer rates. |
| **Development Tools** | Apple Owned/Affiliated tools/software – no open source  Swift is Apple’s programing language and would be the primary language for developing software for macOS and iOS. Developers will be required to know how to utilize this language  Xcode and Visual Studio Code are two IDE that would help in the application development. Xcode is built-in to macOS.  Visual Studio Code is maintained and designed by Microsoft, but compatible with macOS systems.  The Adobe Creative Suite products can be used to create graphics that can be used for the application. Licensing will be required  Multiple development team may be needed. One to develop on Safari, Apple’s web browser.one for the others, i.e. Chrome, Firefox, etc. | Lack of propriety software – Heavy reliance on open-source tools, software, and community.  No restrictions to the programing languages that can be used with Linux. C++ and Java are commonly used with the operating system and would suffice for development.  Visual Studio Code and Eclipse are the two IDE that I am familiar with and are compatible with Linux. Both offer good language support and are free to use.  GIMP is an open-source web designing tool and an alternative for Adobe Photoshop. GIMP can be used for creating graphics to be used in the application. No licensing is required  Multiple development teams will not be necessary. | Because of the ubiquity of Windows there are no limitations to the tools and software that can be used for the application development.  No restrictions to the programing languages that can be used with Windows. C++ and Java are commonly used with the operating system and would suffice for development.  Visual Studio Code and Eclipse are the two IDE that I am familiar with and are compatible with Linux. Both offer good language support and are free to use.  The Adobe Creative Suite products can be used to create graphics that can be used for the application. Licensing will be required  Multiple development teams will not be necessary. | Swift is necessary for building applications on iOS devices.  JavaScript, although not the only language for programing Mobile apps, is commonly used by application developers for Android OS.  Mobile application testing software such as TestComplete or Appium would be necessary to test in the environment.  Developers will need to know Swift.  Native app testing will be required.  Multiple development teams will be required to build the application for iOS and Android. |

## 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**:

Comparing the computing environments, Linux would be the best Operating Platform to support and expand The Gaming Room web-based version of Draw It or Lose It. Linux offers the same benefits as Windows or macOS with minimal drawbacks. Linux is the most cost-effective choice while providing more control over the server and application. Linux has more flexible integration with different hardware and software interfaces and offers built-in security controls and scale-out manageability.

1. **Operating Systems Architectures**:

The Linux system basically works on 4 layers: Hardware, Kernel, Shell and Applications. The kernel is the heart of the operating system and manages the communication between the hardware and the rest of the system. In Linux, everything is a file, meaning all input/output resources are simple streams run through the filesystem name space. This is advantageous because the same set of tool, utilities and APIs can be used on a variety of resources. This also makes the system highly configurable to fit the needs of the developer. Linux also executes many functions by way of system libraries that do not have direct access to the kernel providing additional security by limiting access to device hardware. User space is segregated from kernel space and root users can access that kernel space. User space is segregated from kernel space and root users can access that kernel space. User space is separated from kernel space. Only root users can access the kernel space.

1. **Storage Management**:

With scalability in mind, I would suggest a cloud provider for the storage and database needs of the Draw It or Lose It application. Despite only needing roughly 1.6 GB to store the image files, the original mobile application was a success and warranted enough notoriety to establish a web-based version to be played on different platforms. Starting with a cloud provider for storage and database services allows for peace of mind regarding security, reliability, and scalability. The library of image files can be expanded, and the database of users would not need to be worried about. AWS (Amazon Web Services) is an industry leader and can accommodate for the needs of The Gaming Room. Storage, backups, and the database would be taken care of. AWS supports API-driven code promoting lightweight stateless interactions between the server and the cloud. AWS also provides another layer of security for the user database and storage files. Storage management would be maintained by the Amazon and would not require extra effort.

1. **Memory Management**:

Memory management for the Draw It or Lose It application will be performed on the server side. The resources for the site will be pre-loaded and server responses will be cached on the user’s local memory. The APIs requests/responses are intended to be lightweight interactions between the client and server. Since requests are client driven, the server will need to execute and appropriately allocate memory to the client requests. Luckily, the REST interface promotes statelessness and self-descriptive messages, so the server will be able to respond accordingly. These features will help make the application’s memory management efficient and will only use memory when necessary. The application will randomly select images for the game correlating with the number of rounds established by the players. These image files will be pushed to the stack and transmitted to the client to be stored on the client’s web browsers caching. Once the image files have been transmitted, they are popped from the stack and that memory is freed. The image files are pushed onto the client’s local memory and allows the server application to reallocate resources for other requests. This reduces load times and allows the application to perform more efficiently.

1. **Distributed Systems and Networks**:

Users will be able to access the Draw It or Lose it web-based version from the web browsers of their chosen operating platform. The application will operate through Web APIs conforming to the REST architecture. The Client-Server separation of concerns allows for user interface across multiple platforms by simplifying and standardizing server components. Application installation on the client side will not be necessary. The use of a web browser allows for users to easily access the game without the need to distribute or install software. Requests are client driven, thus storage and memory management will be handled mainly on the server-side, though responses may be cached. This will help with portability and reduce strain on the client-side as well as compensate for connectivity issues. The server will require regular maintenance and updating. Since the application is web-based, the user will need to have an internet connection of some sort to access the application. The RESTful application interactions should be simple, lightweight, and fast so data transfer rates are not bogged down by requests/responses.

1. **Security**:

Authentication and authorization will be important to secure user information and protect the application software between the various platforms. Login credentials will be utilized to authenticate users and authorization will be handled through role-based access control (RBAC) that promotes least privilege. Administrator accounts should be established before launch and will follow a different creation process than that of standard player accounts. Accounts created by users will be assigned player roles allowing them access to play games and manage teams only. Administrators will have access to the server application and manage library files and user information on the database. Integrating Security-Enhanced Linux enforces mandatory access control policies that confine user programs and system services, as well as access to files and network resources. The Web APIs will work with the Client browser security and encryption to the user’s information is protected as well.