

Draw It, or Lose It

# **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 | 11/13/2022 | Ron Bal | Implementation of Entity, team, and player classes |

| 1.1 | 11/22/2022 | Ron Bal | Updated client/server side evaluation |
| --- | --- | --- | --- |

| 1.2 | 12/5/2022 | Ron Bal | Updated recommendations |
| --- | --- | --- | --- |

**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 customer is developing a web-based game called Draw it, or Lose it. The game is currently only available on Android OS but would like to expand this to include multiple platforms.

## [Design Constraints](#_2et92p0)

* Application must render images from a library of stock drawings.
* Consist of four rounds, one minute each.
* Images rendered at steady rate being completed at thirty seconds.
* Remaining teams given opportunity to offer one guess if previous team didn’t solve it. The limit here is fifteen seconds.
* Must be web-based.
* Support multiple teams.
* Each team contains multiple players.
* Game and team names are unique.
* One instance of game running at any time.

## [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)

Use the class Entity to encompass information that the Game, Team, and Player classes will inherit from. Game, Team, Player, and GameService will all reference one another. A ProgramDriver class will be used to build the project to meet the requirements outlined by the customer. All the classes that are created will be accessed by the ProgramDriver class and have the ability to execute them. We will use a SingletonTester class to test that the program is being executed per the customers design and allow for multiple teams and players during a single game.

"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** | There are multiple options for the server side for both self hosted or cloud hosted.  Licenses are $20-$70 per month depending on which route you go. Xcode (apples development tool) is easy to set up and cheap. Certificiate and profile provisioning are easy, there’s built in code coverage. If something goes wrong, the code will run indefinitely, there are extra servers and bots for management. | License to develop for Linux is free but is dependent on the server requirements. Not many people are familiar with how to use Linux which may cause issues if using it for development. | A license must be purchased to run a server. The cost is dependent on the size and uses for it. The cost would likely be insignificant if you’re hosting applications or using it using it for file management. | Server costs are relatively low and is dependent on the CPU, memory, or disk space. |
| **Client Side** | Software is developed using Xcode. Xcode has a complete set of tools for developing, project management, code and visual editors, debugging and more. The environment is free. | It’s open source which creates a lot of options and support and have low costs. Linux isn’t as popular as Windows or Apple so barrier to entry would likely be high. | Costs are relatively high and not open source. There’s a general high familiarity with Windows which allows for a low barrier of entry when developing. | Offers low cost and simplicity while providing flexibility to developers. |
| **Development Tools** | Apple Developer account $99 a year. Uses the Xcode IDE and supports Swift and C programming languages. | The IDE GNOME is used and supports Python, C, C++, Rust, and JavaScript. | Uses VS Code as an IDE. Many languages are supported such as C#, C++, Python, and JavaScript. Costs are either $45 or $250 depending on what you may require as a business. | HTML, CSS, and JavaScript can be used. Several IDE’s are available and may be dependent on the platform the app is created for. Some examples are Xcode, Android SDK, and Eclipse. |

## 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**:

The Windows operating system is widely used and is the operating system we use here. Since we’re all familiar with the platform and use it daily, it would be the best option for development on this project. Windows has tons of IDE support and offerings which will help if we need to develop the application in multiple languages to support the different platforms the app will run on and future platforms the application may be ported to such as Xbox or PlayStation. Windows also has inherent security through regular updates and Windows Security that provides antivirus protection which will be an added layer of security.

1. **Operating Systems Architectures**:

A Windows process consists of an executable program that consists of initial data and code, a private virtual address space, system resources accessible to all threads within the process, a process id, a minimum of one thread of execution, and security through what is known as an access token. The thread is where Windows schedules execution within a process. A program used by a process cannot run without the use of threads. The contents of the registers represented by the state of the processor, a private storage area utilized by the subsystems and DLL’s, a thread ID are all components of a thread. A thread consists of two stacks, one that is used for executing instructions from the kernal and the other for the user-mode.

Windows utilizes their own API that provides services to all windows-based applications which provides a Graphical User Interface, system resource access, and many other services. With the user in mind, a process can be a process that is loaded by the operation system in user-mode which can be independent from a logged-in user. These services are controlled by the Windows Service Manager, which can be used to handle the credentials of the game the Gaming Room has built.

1. **Storage Management**:

A serverless storage solution deploys applications in containers that are automatically launched when the application is called upon. In a standard Infrastructure-as-a-Service cloud computing model, capacity units are prepurchased supplying server components that are “always on”. In this infrastructure, the user has to scale server capacity as the applications demand fluctuates.

With a serverless architecture, the application is launched only when necessary. When application code is triggered to run, the cloud provider allocates resources for that application to run. When the code of the application stops running, the user stops paying for the resources. This will help keep the cost of running and maintaining the application down as well as free up developer time and resources having to scale an application to combat its demands. We should move forward with a serverless architecture to get the memory and security capabilities of cloud storage while also not having to pay for resources not being used.

1. **Memory Management**:

Aspects of the game that need to be accessed often or quickly should utilize the devices virtual memory while anything else can be stored and accessed on physical memory. This will allow the game to run efficiently without demanding unnecessary resources from either the users device or the cloud service or applicable API’s.

1. **Distributed Systems and Networks**:

As the game grows, demanding more resources, it would be beneficial to handle the application with a serverless architecture which will allow the game to be run across multiple servers to accommodate the client demand. This will allow the game to be efficiently supported with the increase in client use.

This will be handled with load balancing which will spread client requests evenly across the applications servers, which will improve the games response and availability. Load balancers often provide cost effective and efficient security measures that can be implemented by the network administraotr through a load balancing algorithm. This will allow optimal server performance.

Within Windows, using a distributed network and system allows for greater performance over a single system. If one PC malfunctions the other PC’s will pick up the slack. This also allows the ability to add more resources. Because of the information being shared among other computers, this lends itself to security issues.

1. **Security**:

This would be best handled by role-based access control as well as typical login credentials. The role based access control will ensure users are only granted the amount of access necessary to run and play the game. This will ensure users can’t make any changes to the application. We can tie in the users login credentials to the biometric data saved within their phone if available to save a user from having to type their password into the application where they wouldn’t need to in others.

The users login credentials should be saved using a hash map in a database to prevent the information to be viewed in plaintext supposing the database gets accessed by an unauthorized party.

Windows updates regularly to patch potential breaches and continually scans for malware, viruses, and security threats. There are also third party offerings for security such as Avast that can protect the computers running the cloud servers.