

Draw it or Lose it.

# **CS 230 3-1 Project One Submission**

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 | 2022/09/18 | Gerardo Gonzalez | First Revision |
| 1.1 | 2022/09/29 | Gerardo Gonzalez | Second Revision (Security) |
| 1.2 | 2022/10/10 | Gerardo Gonzalez | Third Revision |

## [Executive Summary](#_sbfa50wo7nsh)

The new client, “The Gaming Room” wants to develop a web-based game, this game is named “Draw it or Lose it” and as of now it is only on android market (Google Play Store). This came consists of teams guessing what is being drawn. Each team is made up of several people and they have a specified amount of time to guess before it is passed down to different team. The images have 30 seconds and if time is up the other teams have 15 seconds to submit one guess each.

## [Design Constraints](#_2et92p0)

One constraint is having this web application work on all operating systems. In order for this to work properly, one way to do this is by using another application e.g.(a web browser) to run that web application. Other constraints are the number of teams in a single game. Within that team, there is also a number of people. Each team also has to follow the specified rules of the game such as unique names, for all the members of the team, and each player having one instance in memory for that given game/time. Mobile constraints with a concentration in Apple mobile IOS will need to be investigated, the reason is that apple devices have an exclusive OS system that limits developers without the proprietary hardware and development tools.

## [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)

With the given UML diagram, you can see that the ProgramDriver class is where the main loop is run, this class has an interaction with the SingletonTester class and has a uses relationship with the ProgramDriver class. Then you can see that the Entity creates a hereditary relationship with three classes, the Game, Team, and Player class. This makes the three classes children of the “Entity” class.

The relationship that each child class has a specific relationship with each other, for example, the game class has zero instances in the team class, but the team class has multiple instances in the game class.

This continues with the team class and the player class. Then there is the Game service class which does not have any form of inheritance to the Entity class but does have many Game instances within itself with a zero to many instances.

**"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** | **Pros:** MacOS devices work exceptionally well with MacOS servers. Easier Navigation terminal.  **Cons:** Discontinued as of this year so no newer macOS server versions.  Proprietary hardware is necessary to complete daily tasks. Costly to maintain and develop application. | **Pro:** No GUI needed, and hardware is readily available and free of charge in most cases and cost effective.  Can run on almost any machine which makes it light and portable.  Has a reputation for being Powerful, Stable, and Secure.  **Cons:** A variety of distros might be overwhelming. Can be harder to navigate through. Not fully supported in some software applications. | **Pros:** Most used OS on client’s devices. Gaining popularity. A higher number of resources. Beginner friendly, Supportive on most web and software applications. Long-term support.  **Cons:** Not as secure and is primarily the target of cyber-attacks. Resource intensive. Is not open source and can be expensive to implement. A “Heaver OS”. Troubleshooting can be more difficult. | **Pros:** Highest portability. Cloud-based servers are typically used.  **Cons:** workload is not as expansive as other OS. Hardware capabilities are not as robust as other OS systems as mobile devices typically also have poor security. Different OS with both apple and android makes it difficult to develop an application for both Operating Systems. |
| **Client Side** | **Pros:** More mainstream devices are used. Typically labeled as premium equipment.  **Cons:** expensive and proprietary OS comes with devices. Costly to implement application. | **Pros:** Cost is free to use. A large variety of Distros. Minimum Space needed for OS. Compatible with almost all web/mobile platforms. Secure to use.  **Cons:** Has a learning curve. Maintenance is needed more often to ensure that servers are running properly. And has a low user adaption rate. | **Pros:** Most popular used OS for work environments. Easy to use with GUI. A large amount of hardware. Primarily used for gaming.  **Cons:**  New versions often have bugs. More likely to suffer from spyware/viruses. Trouble shooting can be difficult for clients to solve. | **Pros:** the largest number of potential clients (everyone has a phone).  **Cons:** has to be created with some constraints. Like utilizing the screen size. Can be harder to implement than other OS devices. |
| **Development Tools** | Swift and Objective C are the only two languages used specifically with this OS built-in software editing OS XCode for developing native applications. Supports large amounts of Programming languages. Hardware apple silicon is powerful for and desired for large workloads. | Supports almost all languages, and is complimented by tools that are also open source.  All free applications can help to support almost all languages. | Is easier to use. And can run just as well as other OS devices. Built-in tools such as Visual studio help make this OS a great contender for C/C++ programming. A large number of additional tools and extensions are available. | Swift is used for mobile app applications. as Java is the primary language for android phone platforms. For apple devices as mentioned swift/swiftic is used for development with apple OS platforms and is compatible with Mac OS devices. |

## Recommendations

1. **Operating Platform**: My Operating Platform recommendation for “The Gaming Room” is to start with Windows OS devices as windows is primarily used for game development. Although since this application will be more of a cross web-based platform you would want to make sure you have the proper document types. This will help the application run more smoothly in different browsers. So, If the client chooses to proceed with the application in an Apple device. Application can run as intended thanks to the document type and Cross-platform UI’s.
2. **Operating Systems Architectures**: Windows also shares services used by all of windows-based applications. This makes creating your applications more convenient and can reduce the amount of time to lunch your applications. This is done by the abundance of resources and User interfaces that Windows OS offers. Windows is also the most popular OS in terms of running games. Paired with powerful industry-leading architecture in Azure makes this an excellent choice for hosting applications.

1. **Storage Management**: With the Windows Storage system primarily accessible and sortable by its GUI you can organize and customize the storage of the application as efficiently as any other Storage management system, utilizing Azure as mentioned earlier can help manage storage even further by having the data housed in a cloud-based system, which the client can upgrade/downgrade the appropriate storage amount as they see fit. And based on the client’s needs this would help make it easier for the client because it will all the storage maintenance will be taken care of by Azure leaving the client with zero maintenance responsibilities for the storage devices.
2. **Memory Management**: For this application to work were are going to need to store images and display them as the time counts down. One way is to store the images in memory allocations in the program. To Store an archive of pictures/images, the program will need to have a source of images, like a folder or directory in the program. Packaging them in the same directory or folder as the program will make the loading process much easier and faster. Ideally, this application will benefit from having more emphasis on speed so this means utilizing cache memory as a main contender when managing memory.
3. **Distributed Systems and Networks**: having the right tool for the job will control how this program is developed research will be done to find the most efficient way for this program to run on all Operating Platforms, this way it can be portable and accessible to the users. In order for this application to run smoothly, you will also need to have the proper hardware to support all users in that application. My recommendation would be to have a peer-to-peer architecture to help keep the application running more reliably during connectivity issues.
4. **Security**: To help prevent any security breaches, we are going to develop/code an application that promotes built-in security protection. Other security measures will be to encrypt the program and prevent data leaks in the program and place filtration barriers in the web host to help protect the client and server hosts. Using different protocols like Access matrix protocols with the variant of Rollback protection restricts the users based on their role in the organization. This will help control the access of players, and teams. Finally, we would prompt the user to validate their identity with a two-factor identification method and have a singleton method for that user so only one version based on the user’s account details, such as email and username, can exist at any given time. They will also have the option to enroll in two-factor identification with a phone number which can make it convenient for mobile users.