

<Draw It Or Lose IT>

# **CS 230 Project Software Design**

Version 2.0

## 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 |
| --- | --- | --- | --- |
| 2.0 | 08/17/2024 | Luis D. Lara | <Final Draft> |

**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 project we have been tasked with is to set up a web-based environment for The Gaming Room’s app Darw It or Lose It. The Gaming Room has asked that the following requirements be met. More then one team, one or more players per team and that no multiples of the same game team or player.

We plan to tackle these requirements by implementing a singleton design pattern and iterator design pattern in order to sift through the program and make sure that there are no multiples of the same games with the same team names and id’s.

## Requirements

The project must have the ability for one or more teams to be involved.

That each team will have multiple players assigned to it.

That game and team names be UNIQUE to allow users to check whether a name is in use when choosing

team name.

Only ONE instance of the game exists in memory at any given time.

## [Design Constraints](#_2et92p0)

On top of those requirements, we face the following constraints.

Make it available in the different platforms of Mac, Linux, Windows and Mobile devices.

Make sure that there are two different time limits met.

Make sure that there is a library of stock drawings to be pulled from when playing.

Consider what type of storage management practices to use to hold this library.

Lastly the budget, skill level, and timeframe for the project.

## [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 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.**

## In the UML provided above we see how the ProgramDriver is associated with SingletonTester through generalization relationship. We see that the ProgramDriver <<uses>> the SingletonTerster. SingletonTester is where the singleton design pattern will be tested to make sure that there is not another instance of the game that’s already been created allowing us to fulfill one of the requirements.

## GameService is bidirectionally associated with Game noting that it can have 0 to many games. The 0 to mant can also be seen in the relationship between Game Team and Player stating that there can be more then one of each. Within the GameService all the attributes and one of the methods within it are private while all the methods except one are public. Since the GameService allows the ability to create the game and give it a name and an id. Because of its private attributes, GameService also protects from having the repeats of games, players, and teams. Encapsulation is used within the Game class and the Team class when we want to add a new team or player

## The Entity class has a relationship with the Game, Team, and Player classes through Inheritance. There is also a relationship of Polymorphism for Entity and the child classes. Meaning that the Entity class is the parent/super class and the other 3 are the child classes. The child classes inherit the public methods from Entity. However, the child classes may not access the attributes and the private Entity method.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | With the Mac operating platform, it is known by everyone but expensive. An advantage to this platform is the use of servers like Apache and NGINX which are compatible with Windows and Linux. A disadvantage to Mac is that it only deals with apple products and hardware which as I said are expensive. | For the Linux operating platform there are the advantages of lots of freedom in customization. Because of these freedoms it can be arranged to be compatible with both Windows and Mac. One of the biggest advantages for this platform is that it can be free. The only cost to the client would be for the server that would run the application on which could be either Apache or NGINX like mac. | For the server side of Windows there are several server-based deployment methods and many servers to choose from where the game can be hosted. However, a disadvantage to having many servers to choose from is that they don’t all accommodate everything that we might need, and the prices vary. There are some that are compatible with the other two OS and others that are only compatible with Windows and Linux. | I believe that most mobile device games are saved in cloud-based storage. There are many options of servers to choose from for the hosting of the web game but the services that each one offers also affects the prices. A thing that a lot of the server hosts I investigated have in common was the management part of the website. Easily managed depending on the project. |
| **Client Side** | I think the biggest thing to consider for the client side of things will be the server host that will be used. Depending on what we choose it will have to be compatible with the other 3 systems. This is difficult to overcome since Mac is annoying when it comes to sharing with devices that aren’t Apple. It must also fall within the budget which will also be affected by the language that it is programmed in. | I do not think that there will be a lot to overcome where the Linux OS is involved. Since the app started as an Android app, and Android uses a Linux based OS there should be a smoother transition when looking into what server hosts will be used. We want one that will allow availability to Mac, Windows, and iOS. | Some software development consideration for the Windows platform would be the choosing of which server will be hosting the web game. This will affect the time and price that goes into the project. As mentioned in the server side for Windows, there are options for servers to host the website. Some of the cheaper ones I’ve investigated have limitations on the foot traffic that then affect the speed of the site. Of course, those that don’t have the constraints and better benefits are more costly. I believe that the IDE and language part of the preferred host might also be a barrier we have to overcome. Since I do not have a grasp on Azure or the language that it uses. Meaning that we may have to spend some time learning how to navigate the IDE or spend some money on a team that has more knowledge in this area. | For the mobile part of the client side It is my understanding that the game started out as a Android app. This would mean that we would have to focus on turning what we have into something that will be compatible with the iOS system. This will take probably more time if the people working on the project are not familiar with the Xcode IDE or the language that it uses. Or there could be another team that is used but this might mean more money spent by the client. |
| **Development Tools** | The development tools for Mac OS would be Xcode IDE and the Swift programming language for Apple Developers which has the advantages of supporting cross platform with Windows and Linux. There is a price to be paid for the IDE per month. I do not think that more then one team would be required for the development | The best part of Linux is that it is free. The tools for web development such as the language that would be used for the application would be Java. I don’t believe that you would need multiple teams but you would need multiple people. The IDE that I would suggest would be Visual Studio Code. In the past I have used this product which I found to be user friendly and free of cost for what I needed. | Some of the development tools for Windows would be the IDE Microsoft Azure. Azure has many features that would help deploy the web-based app. One of the ones that I liked was the App Services which would help create this project for the PC and Mobile. I do think that there would be multiple development teams needed for this because of some of the services that Azure offers which in turn might drive up the cost for the client. | Since for phones there is only two options, Apple or Android. The development tools for these would be the Xcode for Apple and Android studio for Android. Another option that I have also experimented with would be the Pycharm IDE for python which is free and beginner friendly in the creation of apps for Android although I’ve never tried it for Apple. I do not believe that using these tools would require multiple development teams since mobile apps are not as complicated as pc apps. Since there are limited things that can be done for the OS of the mobiles to run the games properly. I think that this would be more cost effective to our client. |

## 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**:

I recommend that the game Draw It Or Lose It be hosted on the Windows Operating Platform server Azure.

1. **Operating Systems Architectures**:

**The Windows architecture consists of a layered design which has the User and Kernal mode. Since our app is a web-based game our players will more then likely be on user mode. This will limit what they can access within their device hardware. There are some good safety features while in user mode, one being that if the device were to experience a crash it is recoverable. The user will be using the graphical user interphase to access the resources for the system.**

1. **Storage Management**: Since I am recommending Azure, the storage management that would be used for the game will be cloud based. With the Azure servers I would recommend the blob storage that is offered. With the blob storage the client will have the option to store multiple types of files such as the data for the users profiles and the images of the library for the game. The client pays for what they use and it is scalable in case they would like to expand in the future.
2. **Memory Management**:

While using the Azure servers to host I would use the paging system for memory management. I feel like this is the best option for fast loading of the images as requests are made for them by the gamers.

1. **Distributed Systems and Networks**:

**Since Azure servers are compatible with Windows, Mac and Linux there should be no issues with connecting to the website to play the game**. I feel that access to the site has positive and negatives effects to the cloud-based storage. An example of the connectivity issues would be the load time limitation by the individual user’s internet speed. A person with normal internet speeds should have no issues, however someone using something like hotspot pf like 5mps to ger access to the web might have a harder time loading the game properly. A big dependency for using the cloud-based storage is that the client is at the mercy of the third party as far as managing the servers if they go down.

1. **Security**:

**Since this is a web**-based game I would implement the creation of a profile which would require a specific log in ID and Password. A criteria that must be met for a password to be valid would be 7 case sensitive characters and 1 special character. If the game continues to grow and the client decides to add more things to it, such as importing ones own images or allowing the user to change the rules then I would add different rolls within those games to allow for only the creator/host of the game to have control to the settings. For my final suggestion on security, I would suggest that the client add an option for the individual user to have a two factor authentication system if they would like.