

Karl Maloy

# **CS 230 Project Software Design**

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)**-5**

**Recommendations 6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 12/09/23 | Karl Maloy | Update to Recommendation |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only.

## Requirements

We need to take the already developed app and transfer all its uses to other operating systems.

## [Design Constraints](#_2et92p0)

The design restraints we can see from both a web-based attribute and an application is that in all instances it will involve the use of different coding languages while being expected to perform at the same level across the board.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

Below we have a UML of the gaming room as we can see the Game, Team, and Player classes all inherent attributes from the Entity class. GameServices has a reference to Game. Game has a reference to Team. Team has a reference to Player. The Program Driver has a reference to the Singleton Tester. These references are useful in making the code less redundent and more readable. **"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** | Mac is another popular OS and is user friendly as well as easy to obtain. | Linux is a bit of a harder OS to download and use and is somewhat uncommon. | Windows is a very common OS that is used mostly all around the world and often is very easy to learn and use. | A benefit for the use of mobile devices for web-based applications is that the user can easily use them as most are often touch screen. A disadvantage is that often websites can be harder to view if not tailored to the devices screen. |
| **Client Side** | Mac is another popular OS and can be easy to find people who can operate with it. This is another easy option that could be most cost-effective and save time. Another popular option that is fairly common and user-friendly. | Linux is not a very common OS therefore it can be slightly more challenging to find people who can operate on this additionally it may take more time and money due to its occasional difficulties. While this is still an option it is the least preferred in my opinion. | With being one of the most popular OS in the world it is not hard to find people who can work on this OS it also can be more on the cheaper side from this as well. This is the most common OS in the world it is extremely easy and user-friendly additionally it is familiar to most if not all people with an type of computer experience. | This would possibly be one of the more expensive options as there are many types of OS for mobile devices however the most common are IOS and Android. In our case, an Android app is already created and would just need to be maintained. However, we would need to accommodate the IOS system and create an app for that while also maintaining both. This |
| **Development Tools** | Mac is another diverse OS with many options one of which is the popular Eclipse IDE which allows users to write code in the popular language JAVA. | Linux does not have as many options like other options but it does allow for the use of JAVA and the Eclipse IDE which is in my opinion one of the easiest and most beneficial. | Windows being one of the most diverse OS’s can be easy to code with for example we can continue to use JAVA in the Eclipse IDE but we could also use any other software we choose. | For the mobile devices it is more difficult as the code needs to work for both apple and android which use separate OS. Android uses java while apple uses object-c and swift. They do make some 3rd party software that will allow for code sharing however these options are often not popular or well supported. |

## Recommendations

1. **Operating Platform**: I personally would recommend Windows as the operating platform as it is one of the most common OS in the world and is easy to use and produces a good product.
2. **Operating Systems Architectures**: Windows allows for all services and programs to be displayed to the user. Windows has two types, user mode and kernel. These will allow for the easiest user and server communication to be accomplished.
3. **Storage Management**: Windows has what is called “storage sense” and can allow the user to see how much storage is present as well as what storage is being taken up by each application. This will allow us to ensure that we have enough storage for all 200 photos and the game itself. while also not taking up too much storage and allowing the game to be easily accessed and played by all users.
4. **Memory Management**: Windows uses memory management by both RAM and normal storage. For quick access material, you have the RAM where some game parts can be stored in use, and then you also have your regular storage which can be used to store the game easily along with all the photos. The use of the RAM will allow for the game to run efficiently and smoothly. While having the storage capacity that allows for the holding of all the photos.
5. **Distributed Systems and Networks**: Within Windows, we can download an IDE that will allow us to write the code we need to develop the game then we can export the code into each of the languages we would need to be compatible across all platforms. An excellent example of this is Eclipse IDE with this we can write the code in JAVA language which will work across most of the platforms and can be converted to accompany the others. Allowing the game to run on a server will allow the users to connect and access the game without being dependent on the user's platform equipment. Additionally, we can build backup servers into it to help avoid outages within the game and server.
6. **Security**: Windows has built-in security software but other options such as McAfee are available for use with the system. Both will help the user protect their personal information. Additionally, we can ensure that the game comes with proper security measures such as the usernames and passwords to protect user's profiles. We can also build into the game itself security functions to help protect both the server and the user.