

The Gaming Room Design

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 |
| --- | --- | --- | --- |
| 3.0 | 08/14/2022 | Abiel A. Zamora | Added new info to fields |

## [Executive Summary](#_sbfa50wo7nsh)

The company The Gaming Room wants to create a website-based game that serves multiple platforms based on their current game. The name of the game is “Draw It or Lose It” and it has 4 rounds that last 1 minute each where a picture will be rendered for the player to guess. If the player guesses incorrect then the other teams have 1 chance to guess with 15 seconds. The game currently only runs on android.

## [Design Constraints](#_2et92p0)

The design constraints are:

-must run on multiple platforms

-must have one or more teams involved

-must have multiple players on each team

-must have one name used per game and team so no duplicates

- must have only one instance of the game exist in memory at any given time

These are what needs to be included in our software to complete what the company wants. In regards to application development I believe wanting the game to run on multiple platforms will mean having to make sure it the code is correct for each platform and can be run correctly with no errors.

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram below has an inheritance between the Entity class and the Game, Team, and Player classes which means that the Entity class shares attributes or methods to the listed classes. We can also conclude that the Entity class is a parent class. We can also see that the Game, Team, and Player classes have a zero to many multiplicities which means that you can have a game with no teams or many or a team with no players or many. With these relationships it can make running the software efficient by not having to rewrite each attribute or method for each class multiple times if that is the case.

**"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** | Has a large popular fan base which means servers should run well. Having servers run for this platform can cost money from licensing costs. | Has a large popular fan base which means servers should run well. Having servers run for this platform can cost money from licensing costs. | Has a large popular fan base which means servers should run well. Having servers run for this platform can cost money from licensing costs. | Has a large popular fan base which means servers should run well.  Having servers run for this platform can cost money from licensing costs. |
| **Client Side** | Medium time, not much expertise, high cost. Keeping code simple and continuously test for cross-browser support then you can easily fix errors given on another browsers. | A lot of time, a lot of expertise, high cost. Keeping code simple and continuously test for cross-browser support then you can easily fix errors given on another browsers. | Medium time, not much expertise, high cost. Keeping code simple and continuously test for cross-browser support then you can easily fix errors given on another browsers. | Medium time, not much expertise, high cost. Keeping code simple and continuously test for cross-browser support then you can easily fix errors given on another browsers. |
| **Development Tools** | Not many development tools. But Swift is a known development tool that can be used. I believe multiple development teams would work well for this with agile methodology. I do believe many development tools for this platform cost money to use with licensing. | Many development tools that can be used. Like visual studio, eclipse, etc. Some can cost money while others are free. I believe multiple development teams would work well for this with agile methodology. | Many development tools that can be used. Like visual studio, eclipse, etc. Some can cost money while others are free. I believe multiple development teams would work well for this with agile methodology. | Many development tools like Framework 7, Sencha, Xamarin, etc. I believe one development teams would work well for this. I do believe many development tools for this platform cost money to use with licensing. |

## 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 believe that Windows would be best suited to allow expansion. This is because many people are proficient in coding for Windows applications and there are many development tools.
2. **Operating Systems Architectures**: Windows has two layers which are user mode and kernel mode with modules within each layer.
3. **Storage Management**: Windows allows for something called “Storage Spaces” which effectively allows for two drives become merged into one. Using the capacity of the drives to create virtual drives that allow for better memory storage which can help storage management. Windows also has OneDrive installed within every system which acts as a virtual drive and can allow users to have access to more storage.
4. **Memory Management**: Windows has a virtual address space that allows for 4 gigabytes of memory to be viewed which is helpful in terms of memory management. Also, there are many third-party memory management products for windows that can be used for this application.
5. **Distributed Systems and Networks**: We would have ensured the companies servers can support the number of players that will be playing. With making sure the servers are strong enough we can make sure that outages and connectivity are not a problem. Finding a development tool that can export code to different platform will help with distributing the software.
6. **Security**: Windows has a firewall and built-in security feature that comes installed with every system. And if this is not enough, we can always download from private companies to ensure maximum security.