

# **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 | 07/15/22 | Nick Franklin | Updated the Executive summary, design constraints, system architecture view, domain model, development requirements, and recommendations |
| 1.1 | 07/30/22 | Nick Franklin | Minor rewordings but no real changes in Recommendations |
| 1.2 | 08/12/22 | Nick Franklin | Minor rewordings in 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 Gaming Room is looking to develop a web-based version that will work on multiple platforms of their game that is currently only available on Android. The game is called Draw It or Lose It and it consists of four, one-minute rounds of play where teams try to guess the puzzle (generated from a library) before the time is up. If a team is unsuccessful at guessing their puzzle, then the opposing team has 15 seconds to correctly guess.

## [Design Constraints](#_2et92p0)

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* The game must work on multiple platforms.

## [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 Game, Team, and Player classes are all related and inherit from the Entity class. This means that these classes share some commonalities from the Entity class. The Team, Game, and GameService classes are all “has a” type since they reference other classes such as the Team class has a Player, the Game class has a Team, and the GameService has a Game. This helps with efficiency instead of making instances of these in each of these classes. They can call on them from another class. The ProgramDriver is shown to have a relation to the SingletonTester as well.

**"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** | Mac has good flexible terminal commands for setting up the server and editing it and it is upgradeable. It is not the most popular web hosting service though. | Linux is very secure from viruses, less costly than say Mac, and has quick loading times but has less applications for support than say Windows. | Windows has a lot of applications and software available and is very widely used but is more vulnerable to viruses. | Mobile Devices offer mobility but the hardware is not as powerful and for the most part, they have poor security. But it’s best for the server to not be mobile for tracking. |
| **Client Side** | Mac is not as commonly used as Windows so some expertise is needed. Cost may be similar to Windows but time may be more due to less available Software and Applications. | Linux is the least common platform and thus takes the most expertise and it is time consuming. The cost is however low. | Windows is very commonly used and needs probably the least amount of expertise and time with cost being somewhat on par with Mac. | Mobile takes a bit more expertise but is mobile so less time consuming and the cost is low. |
| **Development Tools** | Mac can use most all languages like JS, HTML, and CSS for libraries and front end and more general use languages such as Python and Java. It is likely that Swift would be used. | Linux can use many IDE’s like Eclipse and many languages such as those mentioned in the Mac section. | Windows has basically all IDE’s available for use and supports most all of the languages. | Mobile is compatible with most of the languages and some IDE’s. |

## 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 developing the game on Windows since it is very commonly used meaning that it won’t be difficult to find developers with experience on the platform. Windows also has a lot of available software and at minimum cost.
2. **Operating Systems Architectures**: X86 is the best option because of the power and hardware compatibility. Windows has encompassing applications that are useful for easily creating the graphical portions of an application and easy access to system resources.
3. **Storage Management**: Windows has built-in features for storage management that make it easy and intuitive and adjustable. The cloud is also available for large amounts of storage just like on most other systems.
4. **Memory Management**: Memory allocation will be useful for the game’s library of puzzles to easily be stored and retrieved from a choice location. All processes have their own address space on Windows so as not to corrupt any applications using a space.
5. **Distributed Systems and Networks**: There are some IDE’s that will transfer code written for one platform to that of another platform making it easier to have cross-platform compatibility so that all versions can be connected and playable together. Dedicated good servers are necessary to provide a stable game environment for users across networks. There are also network based databases that can be shared by the different versions.
6. **Security**: Security can be a risk even though Windows has built in protection. It is best to have a limited amount of data collection so there is less risk and to use another form of anti-virus protection. Data encryption is also a must for information not only in the databases but being transmitted between client and server.