

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 | 09/13/2021 | Andrew Riley | Executive Summary and Design Constraints added |
| 1.1 | 09/30/2021 | Andrew Riley | Evaluation added |
| 1.2 | 10/12/2021 | Andrew Riley | Recommendations added |

**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)

Based off the Gaming Room’s current Android mobile game, Draw It or Lose It, we will design a web-based version of the game. The game will be played by one or more teams consisting of multiple players assigned to each team. The game will render stock images from a library as clues. Teams will take turns for four 1-minute rounds attempting to guess the puzzle. The stock image will be fully rendered at the 30-second mark. If the team does not correctly guess the puzzle, other teams will have 15 seconds to offer one guess and solve the puzzle.

## [Design Constraints](#_2et92p0)

* Ability to have one or more teams
* Each team will have multiple players
* Game and team names must be verified as unique
* Only one instance of the game can exist in memory at any given time
* Must be able to run on multiple platforms
* Must stay within budget set by The Gaming Room
* Must stay on schedule as determined by the budget

## [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)

**Entity is a super class. Game, Team, and Player all have an “is a” relationship, or inherit, from Entity. This means that Game, Team, and Player all share attributes with Entity, such as id and name. The UML diagram also contains “has a”, or aggregation, relationships. Team has a Player, Game has a Team, and GameService has a Game. This means that one class references an instance of the other class. The UML diagram also indicates the classes can have multiple instances of other classes. For example, GameService can have multiple instances of Game, there can be multiple Teams in Game, and multiple players in Team.**

**"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** | More expensive. User interface is easy to use and allows for easy server configuration. | Cheapest option due to free licensing. Very good for web-based hosting. | User friendly. Vast amount of software available. Can be costly, similar to Mac. | Inexpensive. Not ideal for server due to lack of power. Can lead to compatibility issues due to compatibility issues between various device specs. |
| **Client Side** | Time and skill needed are moderately high, with the cost being higher than other platforms. | Time needed and skill level are much higher than other platforms, but at a minimal cost. | Time and skill needed are lower than other platforms, with higher cost (similar to Mac) | Time and skill needed to develop for multiple platforms are highest due to compatibility issues between platforms. Developing for multiple platforms would increase cost. |
| **Development Tools** | Most coding will be done using swift, although Macs can also be used to code in Python, Java, HTML, etc. | Common languages such as Java, Python, HTML, Ruby, C++, etc. Tools include, but are not limited to, Visual Studio, Eclipse, and the command prompt. | Common languages such as Java, Python, HTML, Ruby, C++, etc. Tools include, but are not limited to, Visual Studio, Eclipse, and the command prompt. | Apps can be created using Android and Swift (iOS). Python, Java, HTML, Ruby, C++, etc. can also be utilized. Coding with Android and Swift creates compatibility issues between the two. |

## Recommendations

1. **Operating Platform**: I would recommend using Windows for the development of Draw It or Lose it.
2. **Operating Systems Architectures**: Windows is user friendly and has a wide array of software available, as well as a variety of software development tools.
3. **Storage Management**: Windows allows the user to customize the storage configurations as well as offers cloud storage.
4. **Memory Management**: Windows has numerous memory management techniques for both physical and virtual memory that can be utilized to ensure the app runs smoothly.
5. **Distributed Systems and Networks**: Distributed systems and networks allow for higher performance and flexibility. It also is reliable because a system crash on one service doesn’t affect the other servers. However, distributed systems can be more difficult to troubleshoot due to multiple servers being involved. It is also more expensive to set up and maintain the network.
6. **Security**: Windows has numerous security features designed to protect user information. It allows the user to customize their security settings, offering complete control of information going in and out of the system, as well as requiring authorization before system modifications can be made. Windows also encrypts sensitive user information (passwords, credit card numbers, etc.). Windows also has built protection from malware, viruses, and other security threats.