

Draw It or Lose It, The Gaming Room

# **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 | 9/15/2022 | Kate Doughty | Updating software design information. |
| 2.0 | 9/28/2022 | Kate Doughty | Adding Evaluation |
| 3.0 | 10/10/2022 | Kate Doughty | 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 has decided to make a web-based version of their android game Draw It or Lose it. They want the game to be available on multiple platforms.

## [Design Constraints](#_2et92p0)

* The game is set for four 1-minute rounds.
* Player interaction: would players submit answers to the game, or just have a button to click on once solved?
* Drawing rendering: set the software to load the picture in increments over 30 seconds.
* Ability to have one or more teams with one or more players and unique team names.
* If time runs out, give other teams 15 seconds to make one guess.
* Memory: only one instance can exist at a time.

## [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 Entity class is the handler for the Game, Team, and Player classes, with the three inheriting from Entity. GameService, Game, Team, and Player reference data used or collected by the other classes. The ProgramDriver class contains the software for the game itself, using the data collected by the other classes to build a game instance. The SingletonTester class ensures that there is only one instance of the game at a time.

**"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** | Frankly, I struggled with finding information on Mac’s hosting. From what I can find, it seems Mac is user-friendly. | Linux is more popular for web servers, and it is more secure. | Windows is well-known and efficient, with less loading time. A downside is security, with Windows being susceptible to viruses. | Mobile devices are popular and portable, but they have poor security. |
| **Client Side** | As a closed source, Apple would be more expensive, as the client would have to pay for the licenses as well as any other requirements, such as support and systems integration. Not everyone is familiar with Mac OS, so time will have to be set aside for learning, though it is relatively easy to pick up. | As an open source, Linux is more cost-efficient, but it is harder to learn. | Cost would be closer to Mac, but it seems harder to use, meaning more time spent learning or working on code. | Mobile devices are easy to use, cost efficient, and flexible. |
| **Development Tools** | Apple has its own language, Swift, to develop apps. Also popular in server development are JavaScript, HTML, PHP, and the C languages. | JavaScript, HTML, PHP, and the C languages seem most popular for development. | Windows uses JavaScript, HTML, PHP, and the C languages for servers. | Mobile apps use JavaScript, HTML, PHP, and the C languages. iOS also has Swift. |

## 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 starting with Windows. The cost and skill level required makes it a happy medium, and it offers many options for IDEs.
2. **Operating Systems Architectures**: Windows can be used by anyone, and includes useful applications for web services, communication, and graphics.
3. **Storage Management**: A cloud would be the best way to store data, as it can be accessed from anywhere, preventing geo-locks. Cost of a cloud includes paying for upkeep and security.
4. **Memory Management**: Virtual memory is my recommendation for managing memory space, as it copies items that have not been used recently to the drive to clear space on the RAM, leading to better efficiency.
5. **Distributed Systems and Networks**: The game would require powerful servers and a way to preserve game information to handle high traffic on the server, crashes, and power outages.
6. **Security**: Encrypting user data is a must. Also, frequent scans for malware and viruses are important to detect problems early and handle them before user data is compromised.