

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 | 05/17/2022 | Cameron Lee | Completed the Executive Summary and Design Constraints of the document |
| 1.1 | 5/19/2022 | Cameron Lee | Completed the Domain Model |
| 1.2 | 5/22/2022 | Cameron Lee | Completed Evaluation and 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 wants to develop a web-based game application that serves multiple platforms that is based on their current game, Draw It or Lose It. The game is currently only available on Android devices. The application will render stock drawing images as clues steadily and are fully complete at the 30-second mark. Players must guess what is being drawn. If the team does not guess the puzzle before time expires, the remaining teams each have a chance to guess once, with a 15-second time limit.

## [Design Constraints](#_2et92p0)

1. The game must be available on multiple platforms.
2. The game application needs to have one or more teams.
3. Each team will have multiple players assigned to it.
4. The team names must be unique, and players must be able to search to see if a team name is available when choosing their own team’s name.
5. Only one instance of the game can exist in memory at any given 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)

ProgramDriver is the only main() method in the program and uses the singleton design pattern in SingletonTester. Game, Team, and Player all inherit from the Entity class, meaning they have an inheritance relationship with each other. GameService references Game, which references Team, which references Player using association between the classes.

**"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** | Very secure software.  Most expensive option.  Limited hardware options to run. | Not resource hungry.  Relatively cheap to run.  Incredibly stable to run. | Lots of software options.  Less stable.  Not as strong performing. | Cost-effective option.  High portability.  Poor security compared to other options. |
| **Client Side** | High cost due to the cost of the hardware to run.  Moderate time and expertise required due to the specificity of the software to the OS. | Minimum cost due to the simplicity of the OS.  Moderate to low expertise and time required due to the efficiency of the OS. | Moderate cost due to the cost of the hardware to run.  Minimum expertise and time required due to the popularity of the OS. | Minimum cost due to the structure of the option.  Moderate expertise and time required due to the need for Android/iOS compatibility. |
| **Development Tools** | Very specific to the Mac operating system. If issues arise, not as many resources to fix the issue. Still supports most major tools. | Similar to Windows, but a little more niche. Has great development tools support. | Contains the best development tools out of any of the options due to the support and popularity. | Easy to create for both Android and iOS individually. |

## 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 the Windows operating system to develop the application. The popularity, combined with cost, and simplicity make it a great option for this project.
2. **Operating Systems Architectures**: 32-bit for older machines and software, 64-bit architecture for the best performance and most features.
3. **Storage Management**: Windows 10 has a built-in utility called Disk Management that can partition drives, set up new drives, and other storage functions.
4. **Memory Management**: Windows, like Linux and Mac, has memory compression that can compress memory and store it in the RAM of the machine.
5. **Distributed Systems and Networks**: The Qt Creator application is a cross-platform IDE that can develop for both Android and iOS. This is available on Windows devices and can be used to created good reliable code that works with the opposing mobile operating system.
6. **Security**: Windows comes with a built-in malware protection utility called Windows Defender that can provide upfront security to the machine. On the more backend of things, Windows has its own App store built into the core of the operating system that provides a direct way to distribute applications to users with added security.