

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/19/23 | Jasmine Bierschbach | Recommendation updates |
| 1.0 | 04/12/23 | Jasmine Bierschbach | Added information about the design |

**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 would like to make a multi-platform game. The game will be called “Draw It or Lose It” and will be available to play on Android devices. The goal of the game is to have teams go a total of four rounds, one minute per round, trying to guess what a picture given from a library is each round. If the first team can’t guess the picture, then the other team has 15 seconds to try to guess the picture.

## [Design Constraints](#_2et92p0)

* One or more teams required to play
* Multiple people are needed for each team
* Both teams get names and the games have to be unique
* At any point only one instance of a game can happen
* It must run 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)

Team, Player, and Game all receive information from Entity. This appears in UML via inheritance, turning Entity into a super class. The relationship between Game, Team, and Player is a “Has a” relationship to one another, while they all have an “is a” relationship to Entity. Encapsulation is utilized by Entity since it guards data given to the program by restricting access to public methods within the program, protecting the privacy of players’ information.

**"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** | Costly, but well worth the price. GUI lets devs ease of use, this also gives easy server configuration in a workstation. | Cheap and affordable, but very difficult to navigate platform. Linux does have a command shell for basic server composition and convenience. | Expensive, but very user-friendly platform. Plenty of software options. Easy command prompt. | Fairly inexpensive, but mobile device guidelines vary user to user. (Android, iOS, etc all have different specifications) Difficulty making sure it’s compatible on all platforms. |
| **Client Side** | More expensive compared to others, but with decent support time for mac users, a similar amount of skill is needed. | Not too costly, but a lot of time and an almost master level of skill is needed to support these users. | While on a similar cost scale to Mac, less time and skill are needed to actually support these users. | An absolute maximum time and ability is needed to support mobile device users, especially with a lot of OSs constantly changing. |
| **Development Tools** | The usual dev tools on Mac are JavaScript, CSS, Python, HTML, etc.  Visual studio is just one of the tools used that supports libraries of different languages. | Usual Linux languages have Java, CSS, Python, Ruby, etc. Dev tools in Linux are Github, command prompt, and Repl.it. | The common languages on windows include C++, Ruby, #, Python, and Java as well. Visual Studio, Command Prompt, and Eclipse are the common dev tools | Languages for mobile apps include CSS, Java, Python, HTML, C++, etc, with the common dev tools being VisStu and Githuub. |

## 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**: Windows is the recommended platform for the expansion of platforms for the game. While it is expensive, the ease for users and access to multiple software types allows the compatibility of different platforms when more expansion development is needed.
2. **Operating Systems Architectures**: Windows software gives services utilized by windows-based applications that allow apps to display a GUI whilst using system resources. These apps also attribute to Graphics and Multimedia, messaging, and web services.
3. **Storage Management**: Windows lets devs work with storage configuration and memory management, and cloud storage as well, which provides devs as much storage as they could need
4. **Memory Management**: Windows allows for virtual and physical space for memory management, giving 4 GB, in addition to how to run apps well.
5. **Distributed Systems and Networks**: Routing and bottlenecking complications, including queuing problems, are natural on distributed networks. Nonetheless, the above-mentioned systems may comfortably communicate and harmonize with one another. Faltering computing work and connection issues amid users are usually caused via concurrent computations of components and failed pieces.
6. **Security**: Windows gives clients settings that allow guarded information and data that goes in and out of the system. Administrator permission is required to change the platform. Anti-spyware is recommended by Windows for extra security purposes.