

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 | 05/26/2024 | Tashi Anderson | This document has been updated to contain the game program and 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 client, The Gaming Room, wishes to develop a web-based game that can be played on a multitude of different OS. The game is only available for Android currently. This game, which is like the 1980s game Win, Lose, or Draw, is a guessing game that allows players to view a rendered drawing and guess as to what the puzzle is.

## Requirements

The requirements are:

* 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.

## [Design Constraints](#_2et92p0)

The design constraints of this software would be expanding the compatibility to different OS types. By extension, there would have to be testers who have availability different platforms, which may significantly raise this project’s cost. When making flowcharts to help simplify the programming process, there would have to be different flowcharts that correspond to the different platforms, increasing the overall workload of project planners.

## [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)

This model shows how the code will look and work, highlighting the inheritance. This also showcases which classes are the parent and subclasses, and which classes depend on eachother.

**"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** | Characteristics: Simplistic, locked down.  Advantages: The design is sleeker, which may be appealing to consumers.  Weaknesses: Though it is more popular than Linux, there isn’t widespread support. | Characteristics: Has many distros, easier to program in, fully customizable  Advantages: Linux allows the user to fully customize their experience, more so than windows and mac.  Weaknesses: there is less support for Linux than mac, and driver availability may be a problem | Characteristics: most popular, familiar design to most  Advantages: Windows is the top OS to use and has the most support.  Weaknesses: though this problem isn’t limited to windows, there are multiple windows systems not on the latest windows 10/11 update. This means the dev team would have to potentially extend their support to older systems. | Characteristics: easily accessible, cheaper than other OS types  Advantages: Many people have access to a phone with some sort of OS.  Weaknesses: there are many types of OS for phones, meaning there needs to be support for the top few OS. |
| **Client Side** | Learning how to navigate the Mac interface may be an issue to new users, and the cost for an iMac/MacBook is a luxury. | Linux is the least used but can be installed on any system for free. Many people will be unwilling to switch to an OS with less overall support. | Many entry-level systems have Windows installed, which means more accessibility and widespread availability. | Most people have some type of mobile device and would not be forced to buy a new device when using this software. |
| **Development Tools** | Xcode, PyCharm, eclipse, GitHub | GitHub, eclipse, atom, kdevelop | PyCharm, visual studio, eclipse, ruby | Android studio, xcode |

## 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 best OS to use, as many people will have access to a cheaper device that uses Windows. Windows is constantly updated and has the best support.
2. **Operating Systems Architectures**: the windows architecture has kernel mode, allowing code to be executed with unrestricted access. There is also core OS, memory management, process and thread management, i/o, interprocess communication, and executive.
3. **Storage Management**: Network Attached Storage should be used, as it allows for centralized storage, expandability, and data protection.
4. **Memory Management**: Windows uses RAM and VM, which manages the memory that the game uses. If the memory usage is too much, portions of the memory can be stored on the users ssd/hdd.
5. **Distributed Systems and Networks**: There should be servers that the game is hosted on, containing the users’ data as well as the game data to allow for syncing. The user will have their game connect to the server client, thus syncing their data. This will make the user rely on either wifi or their data, which can easily go out in a power outage.
6. **Security**: Windows has a built-in virus defender, which can work hand in hand with this game. Also, allowing the user to set up 2FA and other means of connecting their account may be beneficial in preventing hacking. User data should also be encrypted.