

<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 | <02/23/2025> | <Steven-Mathias> | <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 client, The Gaming Room, wishes to create a web-based game that operates on multiple platforms and is based on their current game, Draw It or Lose It. Currently, the game is only available as an Android app. In this game, the application will render images from a stock library for the players to use as clues to guess the puzzle. Each game lasts four, one-minute rounds, and the drawings are fully rendered at the thirty-second mark. After the minute has expired, if the team does not guess correctly, the remaining teams have fifteen seconds each to offer one guess.

## Requirements

To achieve this, the application will need to be expanded to multiple platforms. Next, it will need to support features including multiple teams of many players, unique names or identifiers, the ability to store and pull images, and set time limits.

## [Design Constraints](#_2et92p0)

* The first design constraint is that only one instance of the game can be run at one time. Meaning that one instance of an image going through the rendering process can happen at one time. This also includes the constraint of time allotment. This image must be rendered consistently for over thirty seconds, and this must be the same for each image.
* The team guessing must be able to enter their guess within a minute. If they have not guessed, or guess incorrectly, the application must cycle through the other teams, giving them each fifteen seconds to enter a guess. Therefore, the primary team’s ability to enter a guess must only be available during the initial minute. Similarly, the other teams cannot make a guess during the initial minute, and only during their fifteen second turn.
* Each instance of a game, a team, and a player must have a unique identifier. The game and team names must allow users to check if the name is already in use. Also, the application must allow one or more teams.

## [Domain Model](#_8h2ehzxfam4o)

First, the entity class uses inheritance to create a relationship between the “Game”, “Team”, and “Player” classes. This is represented by the joining lines concluded by an open unfilled arrow. This entity class is going to hold attributes and methods that the other classes will inherit. This allows us to only establish these items once, and the other classes can use the information.

Second, the program driver is what holds the main class. This will run first. The program driver is linked to the singleton tester that verifies if a game, team or player is already in use, and mandates that one game instance is applicable at one time and team names are unique and/or already in use.

Finally, the Game class has an association relationship with the Game Service class. This relationship is zero to many, represented by the “0…\*”. This means that the elements can be 0, or any number above zero. The game service class manages many of the functions. The game service class will directly utilize the information it receives from the game, team, and player 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** | Mac OS has a Unix foundation that supports stability and security. However, it is limited by cost and less customization. From the server side, MacOS seems to have removed any licensing fee. The MacOS server is available with the general operating system software. | Linux is great specifically for web-based applications. It is highly secure, customizable, and cost effective.  Linux requires a licensing fee of about $300 per server, per year. | Windows also performs well regarding web-based applications. Windows is much more user friendly and easy to learn, however generally less secure. Windows has a higher price for the server use license, however, seems to have more options than Linux and MacOS. | The biggest benefit is being portable and app-based but much lower security and limited processing power.  Mobile devices have a much lower server license fee, however the same applies to the license as with most mobile features in being very limited. |
| **Client Side** | The cost may be a large issue for the client when considering Mac OS, also Mac is great for iOS applications, but the client wishes to not just switch to iOS devices but to expand across multiple. Other than cost, Windows can be more user friendly as the Windows OS is more ubiquitous. | The client will like the cost effectiveness of using Linux, feel better about its high security, and the programmers can more easily make changes and adapt as the project progresses. A potential issue for the programmers is that Linux may take much longer to learn and get up and running due to it not being as widely used as MacOS and Windows. | Windows is also cost effective; however, the weaker security could mean a larger cost to solve problems down the road. Also, Windows is very user friendly and easy to learn. This is great for the programming team as work can be started immediately. | Beneficial for testing the app version of the game across multiple platforms (i.e. MacOS and Android) but will require cross platform tools. Many of the programming requirements are the same, so the team should not have much difficulty creating for a Mobile platform. |
| **Development Tools** | The primary development environment is Xcode, used to develop software for Mac OS and IOS. Also, VS code, which is a code editor available across multiple platforms. | Linux also utilizes VS code while also implementing the Eclipse code editor. Linux is open to a lot more tool opportunities than Mac OS. | Windows offers a large array of software and languages covering many preferences and workflows. Many systems like Python, Java, VS code, and others work extremely well on Windows, and windows OS creates homogeneity for a team when sharing data and working together. | The most limited of the other options. Mobile devices are more for testing what has been created by one of the other operating systems. |

## 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**: My recommendation is to utilize the Linux operating platform. From the research in the table above, this platform offers the best balance of cost effectiveness, customization, security, and processing power.
2. **Operating Systems Architectures**: Linux is used for both web-based and app-based applications. It can be used to build applications for websites, desktop and mobile systems. In addition, it includes flexibility and customization to allow the dev team to create a workflow and process that works best for them and the client.
3. **Storage Management**: Linux offers logical volume management (LVM). This is a storage manager that allows system admins to allocate storage resources. This management can be across multiple physical hard disks and increase system security.
4. **Memory Management**: Linux primarily uses virtual memory, where each process has its own virtual address mapped to the RAM. This allows for a large memory space by swapping data between RAM and the hard disk.
5. **Distributed Systems and Networks**: Ubuntu is software for the Linux operating platform that allows cross-platform distribution. Ubuntu is widely used and easy to use. This will be crucial as the client wishes cross-platform distribution to be a major factor in this version of the game.
6. **Security**: Security is very important for all applications and operating systems. Linux is not as secure as MacOS but is more secure than Windows and mobile devices. Some of the ways Linux can implement security features are through Firewalls to protect against network attacks, restricting access to certain functions/resources, and consistent updates to manage changing software and new vulnerabilities.