

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 | 02/05/24 | Julia Coronado | Updated name, requirements, domain model, evaluation, 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)

Draw It or Lose it is a game where teams compete to guess what images are being drawn. The application will have a large library of images. The game will consist of 4 rounds, with a 30 second time limit to fully render. If the picture is not correctly, guessed the remaining team will have a change to guess with a 15 second time limit.

## Requirements

*<* This will be a web-based application with hardware requirements not available*.>*

## [Design Constraints](#_2et92p0)

1. The game will need to be multiplayer.
2. Game names must have the ability to be unique and validate if a name is already in use.
3. Only one instance can exist in memory at a 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)

The domain model we see the program driver and the test. Then we have the entity which creates the relationship between the team, player, and the game. Also, we can see the relationship between game service and game which we can see all hold a relationship to the main entity.

**"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** | Better security features less vulnerable to viruses and malware  Easy implementation  Great technical support  **Cons**  Cost  Software availability | Cost effective.  Opensource  Security features are constantly updated making security breaches less frequent.  **Cons**  Can be difficult to use.  Support can be more difficult to navigate | Most widely used operating system.  More availability of updates and more software support  Easier to use development tools and resources.  **Cons**  More prone to security attacks since application is more used.  Compatibility issues with other operating systems requiring additional software to test cross platform | Better online use features  Enhanced security  Easy communication with end users  **Cons**  Cost  The need for updates  And can take longer to develop.  Integration with cloud services |
| **Client Side** | Development time  More technical support  Use of multiple supported browsers  **Cons**  Not as widely used as windows and may be difficult for end users | Cost Effective  Open source  Development tools  Quicker Development  **Cons**  Cumbersome and knowledge required | One of the most used platforms and many users have familiarity with windows OS.  More development tools.  **Cons**  Security  Cross platform testing | Technical support  Development environment and tools  Enhanced security feature for mobile devices  **Cons**  There are so many different mobile operating systems which can make configuration a task |
| **Development Tools** | Eclipse  MAC OS | Eclipse  Java | HTML  CSS  JavaScript  Eclipse | Eclipse  Xcode  Mobile Development environments |

## 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 believe for this application the best operation platform would be Windows for the following reasons:

* Better Storage
* More widely used.
* More development tools available for use

1. **Operating Systems Architectures**: This operating system can provide optimal storage and ways to access many more resources such as:

* Storage
* Network architecture.

1. **Storage Management**: Windows does allow for optimal storage and other ways to store data such as:

* Cloud storage
* File storage within in OS.

1. **Memory Management**: Windows has many ways to manage memory as there are ways to manage what software is running and how much memory it is running such as:

* Ways to optimize memory within OS.

1. **Distributed Systems and Networks**: Windows has the capability to connect to the Web and share database information with players. This is the following recommendation:

* Have a robust server to accommodate multiple users.

1. **Security**: Windows has a security center which can alert end users to threats this important for the following:

* Making sure to alert users to threats this important because of the connection style being used during gameplay.
* Protecting user data