

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 | 11/7/2024 | Travis Scott | Changes were made to meet the technical and design requirements from the client. |
| 1.1 | 12/1/2024 | Travis Scott | Optimizations were made for better stability. |
| 1.2 | 12/10/2024 | Travis Scott | Bugs fixes and further optimizations were made. |

**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 Project is to develop a web-based game that serves multiple platforms based on the

game Draw it or Lose it, which is only available on Android devices. The purpose of the game is to have multiple teams consisting of several people going four rounds at a minute each try to guess what is being drawn. When a picture is pulled from a library of images, one team tries to guess the image until time runs out. If time runs out, each opposing team member gets to answer until 15 seconds runs out; this continues until the answer is correct.

## Requirements

- The game must be able to run on multiple platforms  
- Only one instance of this game can exist

## [Design Constraints](#_2et92p0) - Ideally, every team should have multiple players - Ideally, participants should be able to see if a team name is they want is currently in use

## [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)

Entity creates a relationship between Game, Team, and Player class. This means they all inherit or get information from Entity. With UML, we can show this with inheritance. You can look at Entity as the main hub to where all the other classes relate. When we look at their relationship, we see Team and Player is a “has a” type. While Game has a Team and GameService has Games. When we use UML, we call it aggregation (HAS-A). When a user “has a” it means it’s an instance of one class and has a reference to an instance to another class. When we look at the diagram, we see GameService has a reference of Game, Game a reference of Team, and Team a reference of Player.

**"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 has easy accessibility and server configurability. Easy to use graphical user interface. Flexible terminal commands. | Cost friendly. Difficult to navigate the platform. Command shell for simple server configuration and accessibility. | Has an expensive server side, but a user-friendly GUI. Also has a command prompt. | Specifications are better in other devices. Mobile device specifications vary from user to user. |
| **Client Side** | Expensive for users. Moderate time and expertise required. Accurate skills and needed to navigate OS. | There is a lot of expertise and time required. Linux data is required to use the operating system. Maximum cost for Linux users. | More expensive than Linux systems. Easy to learn and understand how to support a Windows setup. Minimum expertise needed. | Provide flexibility to clients or even developers to see updates at any place. Slightly more difficult to implement than other devices. |
| **Development Tools** | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend development. Other development tools include PyCharm, GitHub, Visual Studios etc. | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend and languages. Other. Linux systems include JavaScript, Ruby, PHP and Python. | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend and languages. Developer tools include Eclipse, command prompt, PyCharm, Eclipse, etc. | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend and languages. IDE’s for programming languages consist of HTML, php, C++ and Python. |

## 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**: The most appropriate operating platform that will allow The Gaming room to expand Draw It or Lose It to other computing environments is Windows. Windows is best as you won’t run into a shortage of IDEs to work with.
2. **Operating Systems Architectures**: Microsoft Windows is a graphical operating system developed and published by Microsoft. It provides a way to store files, run software, play games, watch videos, and connect to the internet.
3. **Storage Management**: Windows 10 comes with a nice feature called storage sense. This allows you to scrutinize and manage files on your hard drive, along with how much space it takes up.
4. **Memory Management**: The Windows 10 storage sense would allow for storage and management of Draw It or Lose It photos and game players. It also allows you to keep them together in one secure space in memory.
5. **Distributed Systems and Networks**: Network based multiuser interaction systems such as network games typically include a database shared among the players that are physically distributed and interact with one another over the network. Currently network game developers must implement the shared database and the inter player communications from scratch.
6. **Security**: Windows comes with built-in security protection software. Though to secure user data and information it would be recommended to use another source.