

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## 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 | 03/22/24 | Chris Weaver | Added details to Executive Summary, Design Constraints, Domain Model and Recommendations. |
| 2.0 | 04/04/24 | Chris Weaver | Updated Development Requirements section to include more detail. |
| 3.0 | 4/19/2024 | Chris Weaver | Updated Recommendation section. |

**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 intended to take an existing web-based game, Draw It or Lose It, and make it compatible with multiple operating platforms. The game is currently only available on Android devices. The game is loosely based on the 1980’s television game called Win, Lose, or Draw. Multiple teams compete to guess what an image is that is being drawn. Instead of a person drawing the image on an easel and letting the teams guess the image, the application renders images from a library of stock drawings as clues. One game consists of four rounds lasting one minute each. The drawings rendered from the library of stock images should display at a steady rate and be complete at the 30 second mark. If the guessing team does not guess the image correctly before time expires, the other teams will take turns making one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

* Application must be distributed across multiple platforms. This is a challenge due to having to make the application compatible with different operating systems and interfaces.
* Multiple players on multiple teams must be supported by implementing client-server architecture. The application must run smoothly over a network connection between a client and the server and the server must be able to efficiently handle multiple players per game instance.
* Different platforms require different interface options. The app was previously designed for Android devices so making the app compatible with other devices will challenge developers to make the interface as similar as possible with each interface.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram below demonstrates inheritance between the Entity superclass and the Game, Team and Party subclasses. The Entity superclass inherits data and information from the subclasses below it. The relationship between the Game, Team and Player subclasses represents a “has a” type meaning each subclass has an instance in the other. The GameService class also has an instance in the Game class.

**"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 OP has easy accessibility and configurability while also having graphical user interface that is easy to use. Also has flexible terminal commands. Mac offers server-based deployment methods (macOS Server) that is easy for Mac clients to utilize. MacOS Server also provides the tools necessary to manage macOS and iOS devices. The cons are that it can be quite expensive since it only runs on Apple hardware. | Less expensive but also more difficult to navigate compared to other OP’s. Linux offers several server-based deployment methods such as LAMP stack but it requires more technical knowledge through the entire cycle from installing to debugging and maintaining. Requires no license to use. | More expensive but also a more user friendly graphical user interface. Windows Server OS is the server-based deployment method for Windows. It is widely used and allows multitasking as well as support for peripheral devices. The cons are that due to its widespread use it is more susceptible to security threats. Requires user-based licenses, driving up costs. | Mobile devices vary based on device being used meaning the specifications of each device being used can be different. Some of the choices of server-based deployment available to mobile device app developers are DigitalOcean, AWS, and Microsoft Azure. Choosing the best method depends on how much control the developers wish to have over the application server. |
| **Client Side** | Somewhat expensive for clients, takes a medium amount of time to navigate and clients need to be experienced to navigate OS effectively. The MacOS needs dedicated support from experienced developers which can potentially drive costs up due to hiring more experienced developers. | Most expensive and requires plenty of expertise and knowledge. Due to the complexity of the Linux OS, this OS suffers from the same issue as Mac requiring developers with more experience to support. | Least expensive but also easiest to learn and use while not requiring as much time as other OS. Due to the ease of use of Windows OS, the cost should not be quite as high compared to other platforms. | Due to varying devices mobile is flexible for clients but slightly more difficult and timely to implement than other OS. |
| **Development Tools** | JavaScript, CSS, HTML. Tools include things such as GitHub, PyCharm and Visual Studios. License required?: No | JavaScript, CSS, HTML. Tools include things such as Gedit, Vim, and Netbeans. License required?: No | JavaScript, CSS, HTML. Tools include things such as Eclipse and PyCharm. Licenses required?: Yes | JavaScript, CSS, HTML. Tools include things such as Android Java, Swift and Windows .NET. License Required?: Yes |

## 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 would recommend using Windows as it has more IDE’s available to utilize for development. Windows is also vastly popular across the world and is compatible with many software and hardware configurations.
2. **Operating Systems Architectures**: Windows, developed by Microsoft, is an operating platform providing a graphical user interface that gives users the ability to do multiple tasks such as store files, watch videos, run software and play games that are compatible with the operating platform.
3. **Storage Management**: The Windows operating platform comes with a system utility called Disk Management that allows for advanced storage operations such as setting up a new drive, extending volume on the same drive and shrinking a partition.
4. **Memory Management**: Windows allows users to store and manage photos and names of games and players for Draw It or Lose It. All of this data can be stored in memory where the user chooses.
5. **Distributed Systems and Networks**: Draw It or Lose It will allow users on multiple platforms to connect to a shared database via network connection. The database will store the data needed for each instance of the game in process and each device should be able to interact with others connected to the same game instance. The developers will need to implement a shared database as well as player communications for this application to work as intended.
6. **Security**: The Windows operating platform has built-in security software that protects and secures user data, however using a trusted third-party security software is a good recommendation.