

<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 | <07/14/2023> | <Jamie Javis > | <The executive summary, cover page, document revision history, design constraints, system architecture view, domain model, and recommendation all underwent revisions.> |
| 1.1 | <07/25/2023> | < Jamie Javis > | <Changes to Development Requirements under Evaluation Tab.> |
| 1.2 | <08/12/2023> | < Jamie Javis > | <Changes to Recommendation Tab> |

**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 loosely similar to a TV game “Win, Lose or Draw”, where teams compete to guess what is being drawn. The Gaming room is looking to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is only available as an Android app. The application should render images from a library of stock drawings. A game consists of four rounds, lasting one minute per round. Images need to be rendered at a set rate and are fully complete after 30-second mark. If the object is not guessed before time expires, the other teams have the opportunity to offer one guess each to figure out the puzzle in a 15-second time limit.>

## Requirements

* *Each team has multiple players*
* *Game will need to have a timer*
* *Will have to access a library of images*

## [Design Constraints](#_2et92p0)

<For design constraints, we need to move the app from android and convert it to other platforms and OS. The objective is to reach a larger audience with success and little to no differences. Only one instance of the game can exist and any given time. Each team needs to have multiple players. Game and team names need to be unique. Users should be able to check if a name is 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)

<The model below shows that we created a package containing the classes for Entity, Team, GameService, Game, Player, Program Driver, and SingletonTester. We developed the Entity class as a base class that has an inheritance relationship with Game, Team, and the Player classes. There is an associative relationship between GameService, Game, Team, and the Player classes. Each the instance of GameService “has-a” (addressed by 0…\*) instance of Game class, which has an instance of the Team class, which also has an instance of the player class. The importance on the instances throughout the program creates the structure needed to iterate through lists that exist within the Team and Player classes that help ratify the requirements from the Gaming Room to guarantee that there is one instance of each team and player during any given game. The program driver class is shown to use the SingletonTester class to confirm that there is only one instance of the GameService at any given time.>

**"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 simple availability and server configurability with simple to utilize GUI (Graphical UI), highlighting adaptable terminal commands. Only available for Apple products. | Linux is inexpensive, free, and open source in most cases. It tends to be challenging to explore in view of client experience. It has an organized shell for straightforward server designs. | Licensing charges can be expensive, making it more costly than Linux, yet is more available than macOS. Easy to use GUI and contains an order brief. Available and collaborates with server administrations like Microsoft Azure. | Specifications for mobile devices vary from model to model and person to person. Because the majority of applications are not cross-platform compatible, the ability to run software can be impacted by hardware. |
| **Client Side** | Apple is the maker of MacOS. On the desktop computer, it controls about 10% of the market. This makes it less alluring for outer assaults and adds a component of security to its clients. Its sleek design makes it easy to integrate with other Apple-branded products. However, Apple's hardware tends to be closed and uncustomizable. Mac OS isn't good for gaming because it doesn't have enough graphics memory, and it's expensive because it only works with Apple-branded products. | Linux can be customized, is extremely cost-effective, is open-source, and is dependable. Linux has an absence of programming and equipment accessibility, business support, a lofty expectation to learn and adapt, and similarity issues. It is not really the most client friendly to those not of an IT/PC foundation and has less openness when contrasted with other major working frameworks. | The operating system for personal computers that is used the most is Windows. It is easy to use, has excellent software and hardware compatibility, and it can be customized. Most games delivered for Linux or Mac are additionally delivered for Windows. The majority of games made for older versions can still be played on modern operating systems. Operating Windows can be costly due to licensing fees. Windows can likewise be casualty to security issues because of such a wide client base. | Cell phones have more fragile equipment than what customary laptops contain. They are profoundly advantageous and multipurpose with highlights for correspondences, area following, NFC installment choices, cameras, and different others. Cell phones are progressively normal and are perfect for reasons for diversion. Cost can differ in light of explicit gadgets. can provide experiences that are easy to transition to from one operating system to another (such as iOS' seamless compatibility with macOS). |
| **Development Tools** | Languages that include JavaScript, HTML, and CSS. Libraries for front-end development support other advancement instruments incorporate PyCharm, GitHub, Visual Studios and so forth. | Languages that include JavaScript, HTML, and CSS. Language and frontend support libraries Other. Ruby, PHP, Python, and JavaScript are examples of Linux systems. | Languages that include JavaScript, HTML, and CSS. Language and frontend support libraries Engineer instruments incorporate Overshadowing, order brief, PyCharm, Eclipse, and so forth. | Languages that comprise of HTML, CSS and JavaScript. Language and frontend support libraries HTML, PHP, C++, and Python are the four programming languages supported by IDEs. |

## 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 suitable OS that will permit The Gaming space to extend Draw It or Lose It to other platforms is Windows. Windows is best as you will not run into a lack of IDEs to work with.
2. **Operating Systems Architectures**: Microsoft Windows is a graphical operating system that Microsoft developed and released. It gives a method for putting away records, running programming, mess around, watching recordings, and interface with the web. The storage and processing for most windows devices is sufficient enough to run, Linux is the next best thing.
3. **Storage Management**: A feature known as "Storage Sense" can be found in Windows 10, one of the most recent versions of the Windows operating system. The motivation behind this component is to permit the client to permit windows to naturally let loose drive space when things that are not required are jumbling the drive space, for example, brief documents and things in your reuse receptacle that you might disregard.
4. **Memory Management**: Windows utilizes strategies, for example, Swapping (By the utilization of the trade in, or moving the program from the hard plate to the RAM, and trade out, or moving the program from the RAM to the hard disk.), what's more, memory compaction ( Duplicating all pages that are not being used to one huge region, composing all pages that are at present being utilized into recently liberated space, diminishing or taking out fracture of the information and make it simpler for projects to distribute memory) .
5. **Distributed Systems and Networks**: You can use hyperlinks to load web pages by using a protocol like HTTP (Hypertext Transfer Protocol). It operates on top of other layers of the network protocol stack and serves as an application layer protocol for information transfer between networked devices. Utilizing a HTTP permits data among clients and servers to be moved. HTTP is dependent on TCP's reliability. It is more effective because it makes use of multiplexing, which sends multiple requests through a single TCP connection.
6. **Security**: Since Windows is the most broadly utilized OS, security can be a worry. Windows does come with a basic firewall and anti-virus system to help fight malware and computer viruses. In the event that you decide to not utilize Windows pre-installed software, hostile to malware programming would be suggested for security.