

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 | 07/20/25 | Talia McCarthy-Wielenga | Information Update |

**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 purpose of this software design is to outline the needs of the software design for Draw It or Lose It, a application that is loosely based on the 1980s television game Win, Lose, or Draw, for the client The Gaming Room. The game will consist of four rounds lasting a minute each, where the goal of the player is to guess what the drawing is going to be by the 30-second mark. The Gaming Room needs help with setting up the environment to streamline the development.

## Requirements

*The Gaming Room has listed the following software requirements for the game: the ability for one or more teams to be involved, each team must have multiple players assigned to it, the game and team names must be unique to the teams to check if a name is in use when picking their team’s name, and only one instance of the game can exist in memory at any given time to be done by creating unique identifiers for each game, team, and player instance.*

## [Design Constraints](#_2et92p0)

* The game is web-based which leads to the constraint of network communications and security.
* There needs to be unique names for the players to be able to play, but the company did not specify if they wanted a name generator to help players choose a team name.
* The game needs to account for the single instance limitation for the game only being able to have one active game instance in the memory.

## [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 diagram below shows what classes exist in Draw It or Lose It. Entity is serving as the base class for all the other entities in the app. The GameService class has a composition relationship to the Game class. The Game class, Team class, and Player class are all subclasses of the Entity class. The Game class has a composition relationship to the Team class and the Team class has a composition relationship to the Player class. The main loop exists in the ProgramDriver class and a SingletonTester is used in the ProgramDriver.

**"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 is unix-based and developer friendly. Mac has a limited scalability compared to Windows and Linux. Mac is also higher costing as far as hardware goes.  **Project Two:**  To minimize costs the client may choose to utilize a client side API which allows for them to connect directly into the browser using node.js as a backend server or another similar program for the backend server. | Linux is open-source, which is also known for it’s security. Linux is also very scalable and flexible.  **Project Two:**  Linux being incredibly secure would allow for an admin team to control access while including certain features like password protection or multifactor authentication. With Linux the development team should maintain software updates and use backup logs, the team itself would need to be knowledgeable in Linux as it is not as user friendly as something like Windows, with Linux the user can also choose the security parameters which can increase security. | Windows is relatively cost efficient for users. With Windows there is lots of support options and has many options for developers.  **Project Two:**  While yes Windows is a lower cost option, and is user-friendly there is also a risk of security features. Windows starts off low cost but the cost can increase depending on security features used and if a cloud needs to be used to increase computing power. | Mobile devices are portable, but have different capabilities based on the type of device it is (IOS or Android). While Mobile Devices are portable they limit users based on size of device (phone vs tablet) it may not be as easy to work on certain projects on such a smaller device.  **Project Two:**  Being that mobile development would require ground up coding this option is not preferred for this specific project as we are looking at creating a solution based on the website and web development. |
| **Client Side** | Macs are Apple products that are continuously being developed and upgrading to meet client needs which makes them more expensive than other options. Macs work in IOS so people new to IOS may have more of a learning curve using one than someone who has used IOS in the past.  **Project Two:**  There are most cost effective options if using Mac specifically the app could be made hybrid rather than native this would allow for portability and keep the cost low, however some instances for the app could be left out of not work as intended if the cost is the only thing taken into consideration. | Linux is considered a much harder program to use but it is free to use and can provide extra security even when just running it in the background.  **Project Two:**  Linux provides front-end client interaction and can connect to databases. Linux could provide the extra security that a client may want and does not have to be the only OS running at a time, if the client wanted a more user friendly approach but have the security of Linux that would be possible. | Though Windows is relatively less expensive than Mac, users may experience more security issues. Windows does not require that much extra knowledge for doing simple functions.  **Project Two:**  Visual Studio can be used to create the code and add some protection in when working in the Windows OS. Personally, I do not like Visual Studio I find it to be glitchy, but it is user friendly and is a good starting place for a base code. | Cost of Mobile Devices vary from brand to product type, there also may be problems utilizing features when in dead zones, which can be a setback for users.  **Project Two:**  While I have mentioned the downsides of using mobile devices, the client would be able to allow for responsiveness if using react. |
| **Development Tools** | JavaScript is commonly used in Mac with IDEs like Xcode.  **Project Two:**  Flask, mySQL, or node.js could be used. | IDEs like VSCode are used with Linux, Linux also has many different packages that can be worked in.  **Project Two:**  Firebase and Heroku can be used. | Cft is commonly used with Windows and one of the more popular IDEs used is Visual Studio.  **Project Two:**  Javascript, HTML, CSS, or React can be used in the Windows platform. | Java and JavaScript are used commonly for mobile applications.  **Project Two:**  Most browsers work on all mobile devices, however some work better than others since they were made for that system (Safari for IOS). |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

*Project Three Updates in italics.*

1. **Operating Platform**: Utilizing a web-based operating platform would allow for The Gaming Room to expand to other computing platforms.
   1. *The Gaming Room should utilize a cloud-based operating platform this would lower cost and can reduce complexity. Using this sort of operating platform would be useful for multiple different operating systems while allowing for less maintenance needs.*
2. **Operating Systems Architectures**: JavaScript and HTML will be used for the software side of the game.
   1. *Using Linux would be more secure and promotes consumer safety with their personal information.*
3. **Storage Management**: Cloud storage would be a good storage management system as there is much cloud space to allow for larger documents.
   1. *Cloud storage is still a good option for storage since it allows for larger documents and has (theoretically) endless space while a local drive would eventually run out of space. Clouds can be increased to suit the company needs while remaining cost effective.*
4. **Memory Management**: Cloud storage auto saves to the designated save area, this allows for the developers to avoid having to complete saving without having to manually save at every point.
   1. *Cloud storage once again would be good for memory management as it would auto save and remain cost effective for the company. Cloud storage can also have the total amount of storage adjusted to meet the company needs so it can be larger or smaller depending on how much memory is needed and budget restraints.*
5. **Distributed Systems and Networks**: To achieve communication between various platforms, the developers can utilize a centralized server to maintain the communication between clients. Utilizing a centralized server will allow for users to be on different operating systems while still communicating with other user.
   1. *If the company opts for a cloud server then they can give admin rights to certain servers while users would only have access to what they’re allowed to have access to. By utilizing this type of server the company would not have to worry about shutting down everything to fix a problem in a central server they would be able to shift over to a different server so the game can remain up while fixes are being made.*
6. **Security**: A secure communication protocol can be utilized to protect user information on and between various platforms, a firewall could also be used in the central hub to protect against hacking, including logins for users would also help to protect their information, another technique that could be used is encryption to better protect user information.
   1. *The Gaming Room should be using encryption to protect user information, they should also be assigning admin rights to maintain who can access what. Another thing The Gaming Room could implement is running background code to monitor if someone is trying to access information they shouldn’t be this would allow the company to flag those accounts and adjust their security measures as needed while shutting the problem down right away.*