

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 11/17/2024 | Cydnie Fisher | The following sections were completed: executive summary, requirements, design constraints, and domain model. |
| 2.0 | 12/1/2024 | Cydnie Fisher | The evaluation section was completed. |
| 3.0 | 12/11/2024 | Cydnie Fisher | The recommendations section was completed. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has a game, Draw It or Lose It, which is currently only available on Android. They are looking to develop a web-based version of this game, which will be available on multiple platforms. The game should consist of a large library of images of drawings, which will be randomly rendered to the users. The Gaming Room requires that only one instance of a game can exist at a time, and that game and team names should be unique. This requirement will be met by implementing a singleton pattern to manage the games efficiently. Games, teams, and players will be stored in an array list, and given a unique ID. An iterator pattern will be implemented to search through those array lists, searching for the respective IDs to ensure that each is only implemented once.

## Requirements

* Games should allow either one or multiple teams to play
* Teams will consist of multiple players
* Game and team names should be unique
* Users should be able to check if a name is in use when choosing their team name
* Only one instance of a game should exist in memory at any given time

## [Design Constraints](#_2et92p0)

* Now that the application will be available on multiple platforms, more people will have the opportunity to use it, meaning the server will need to be able to handle a larger amount of traffic.
* Each platform that the application will be available on has different development requirements that will need to be accounted for to ensure that the application works as intended on all platforms.
* Assuming that players can play together or against each other over the internet, the application will have to continually communicate amongst multiple platforms, ensuring that players from all platforms can play rounds of the game together.
* If also assuming that players can play together or against each other over the internet, the gameplay data will need to be synchronized and maintained amongst multiple clients.
* The application will need to be created using OOP principles to ensure maintainability. This will allow for new updates and features to easily be implemented.

## [Domain Model](#_8h2ehzxfam4o)

**"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.**

The above UML diagram represents the current state of the application. The Entity class serves as the base class with the Game, Team, and Player class extending Entity. This allows for those classes to reuse the name and id attributes, which are protected but can be obtained by using public setters. The Game class also contains an array list for teams and a method allowing teams to be added to the game. While the Team class also contains an array list for players and a method that allows for players to be added to the team. The player class represents the individual players on the team. The GameService class manages the game. It contains the list of games, with methods to add games and identifiers for game, player, and team. It also contains the singleton pattern, ensuring that only one instance of a game exists at a time.

The UML diagram showcases three of the main OOP principles: inheritance, polymorphism, and encapsulation. The Game, Team, and Player classes all inherit from the Entity class, ensuring that attributes in the Entity class can be accessed in those classes, allowing for less code to be written. The entity class also allows for polymorphism, with its constructor and toString methods being able to be overridden by the classes that inherit from it. Encapsulation is commonly seen throughout as well, with private attributes and lists of games, teams, and players only being managed through public methods.

## [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** | MacOS is compatible with Unix-based tools which allows it to be developer oriented and secure.  MacOS is not commonly used as a web server, and does not offer their own server-based deployment method. MacOS use offer their own server-based deployment method, but hasn’t since April of 2022. Although, Apple does support other servers, such as Apache. The license for Apache is free, but a physical or cloud server will be needed. These servers are not free, and cost will depend on server type and the size of the server needed.  Disadvantages of using MacOS to host a webserver include no native server-based deployment, expensive hardware, and potential compatibility issues with other software. | Linux is easy to use for web hosting due to it being open-source, customizable, and having a strong community support.  Linux offers their own server-based deployment method which comes preinstalled with Linux and can be used to host the website. Linux servers are known to be stable, secure, and reliable, with long runtimes and scalability. The license for Linux servers is free, but a physical or cloud server will be needed. These servers are not free, and cost will depend on server type and the size of the server needed.  Disadvantage of Linux servers include still having to pay for either a physical or cloud server, and being difficult to use do to needing to understand Linux commands. | Windows is a well-known and commonly used operating system. It is also commonly supported by other applications.  Windows offers their own paid server-based deployment method called Windows Server. Windows Server is user-friendly providing a graphical user interface, integrates well within the Windows ecosystem, is easy to deploy, and offers scalability. Windows Server costs $1176 for the base standard edition, and $6771 for the base datacenter edition.  Disadvantages of Windows Server includes a high licensing cost, security concerns due to user errors, and being resource extensive. | Mobile devices are not designed to handle hosting web-based software, and subsequentially do not have their own server-based deployment methods. Therefore, there are no advantages, disadvantages, or costs for hosting a web-based software application on mobile devices. |
|  |  |  |  |  |
| **Client Side** | MacOS is a proprietary software only available on Apple computers. MacOS is free with any Apple computer purchase, but the hardware it runs on can be expensive.  Testing will need to occur to ensure that the web app works properly in Safari, the native web browser on Apple devices.  Expertise in creating a web app with a responsive HTML interface is required. Knowledge in how MacOS functions, and how to test web apps in Safari is also required. | Since Linux is open source, it can be downloaded onto most computers for free.  Testing will need to occur to ensure that the web app works properly in commonly used Linux web browsers such as Chrome and Firefox.  Expertise in creating a web app with a responsive HTML interface is required. Knowledge in how Linux functions, and how to test web apps in Chrome and Firefox is also required. | Windows is a proprietary software, but is commonly found in computers. If the computer does not come with Windows preinstalled, it costs $120 for Windows 11 home, and $160 for Windows 11 Pro licenses.  Expertise in creating a web app with a responsive HTML interface is required. Knowledge in how Windows functions, and how to test web apps in Edge, Chrome, and Firefox is also required. | Testing will need to occur to ensure that touch input functions correctly.  Expertise in creating a web app with a responsive HTML interface is required. Knowledge in mobile devices and their perspective web browsers is required.  **iOS:**  iOS is a proprietary software available on Apple iPhones.  Testing will need to occur to ensure that the web app works properly in Safari.    **Android:**  Android is an open-source software that is commonly used in smartphones, by numerous brands.  Testing will need to occur to ensure that the web app works properly in Android web browsers such as Chrome |
| **Development Tools** | Programming Languages: HTML, CSS, JavaScript  Tools: VS Code, Xcode, and React  Cost: VS Code costs $45 per user per month for a professional subscription, and $250 per user per month for an enterprise subscription | Programming Languages: HTML, CSS, JavaScript  Tools: VS Code, React, and Angular.  Cost: VS Code costs $45 per user per month for a professional subscription, and $250 per user per month for an enterprise subscription | Programming Languages: HTML, CSS, JavaScript  Tools: VS Code, React, and Angular  Cost: VS Code costs $45 per user per month for a professional subscription, and $250 per user per month for an enterprise subscription | Programming Languages: HTML, CSS, JavaScript  Tools: Xcode, VS Code, React, and Angular  Cost: VS Code costs $45 per user per month for a professional subscription, and $250 per user per month for an enterprise subscription |

## 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 operating platform that I would recommend to The Gaming Room would be Linux. While using Linux over a system like Windows presents a learning curve, the benefits of using it outweigh the drawbacks. Firstly, Linux is open source, meaning it is readily available to use for free. Linux is also commonly used for its server capabilities due to Linux Servers being secure, stable, and reliable with long runtimes and high levels of scalability.

1. **Operating Systems Architectures**:

The operating system architecture for Linux consists of multiple layers: utilities, hardware, kernel, shell, and user applications. The utility layer consists of command line tools that enable user performance, such as file and user management and network configurations. The hardware layer manages the hardware components and includes device drivers, memory management, and kernel functions. The kernel layer is the main component of Linux and can be used to visualize hardware resources and to provide processes with the necessary resources. The shell layer is the interface to the kernel, ensuring that the internal execution of functions is hidden from the user. Lastly, the user applications are the programs that users interact with while using the system.

1. **Storage Management**:

For storage management, I would recommend using a cloud-based storage system, such as Amazon Simple Storage Service, where the pricing is dependent on the amount of storage used. Cloud storage services can be more cost-effective since you only pay for what you use and don’t have to worry about buying and maintaining hardware. This also leads to higher levels of scalability, with immediate access to extra storage if necessary.

1. **Memory Management**:

Linux has its own memory management subsystem, which is responsible for managing virtual memory, garbage collection, and caching. Linux utilizes virtual memory to ensure that memory is properly allocated for applications to use. Caching is used for frequently accessed files and can reduce the access times for requests to these files. Finally, garbage collection occurs when unused memory is reclaimed by the operating system to ensure it is available for use.

1. **Distributed Systems and Networks**:

Since Draw It or Lose It will be available on multiple platforms, it will need to communicate with various platforms at once. The code is already written in a way that allows for this by using API endpoints and HTTP methods to communicate between the client and server sides of the application. This separates client and server logic and allows the program to be easily integrated among various platforms. Load balancing should also occur to distribute the data requests, which ensures the server functions efficiently and doesn’t become overwhelmed. Lastly, monitoring tools should be used to continually monitor and analyze server performance. This can help discover issues before they occur, allowing these issues to be prevented.

1. **Security**:

Security is important for ensuring the protection of user data and can be achieved by combining various methods. Firstly, role-based access control, which is already implemented into the code, will ensure that only users who possess certain roles may access certain functionality. Communication between the client and server should be done through HTTPS to ensure that the data remains encrypted during transmission. Lastly, the server and any dependencies used should be updated regularly so that discovered vulnerabilities can be patched.