

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 | 20240127 | Brandon Murphy | Summary, requirements, constraints, UML diagram, Evaluation, recommendations |
| 1.1 | 20240209 | Brandon Murphy | Updated/ added information |
| 1.2 | 20240224 | Brandon Murphy | Recommendations sections of the software design |

**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 client “The game room” wants to develop a web-based game that serves multiple platforms based on their current game, “Draw it or lose it”, which is only available on android now. However, the client seeks to make this cross-platform. The game and team name needs to be unique; the game should allow one instance of the game to exist in memory at a time. The game will need to be able to have multiple or single teams playing per game.

## Requirements

The game will require one or more teams involved and each team be assigned players with both players and teams having unique names and ID’s. The game must be multi-platform and only one instance of the game should exist at a time. The application should provide a responsive drawing interface that allows users to create and submit drawings in real time.

## [Design Constraints](#_2et92p0)

The application must be developed as a web-based platform to ensure compatibility across different operating systems and devices. The application should be designed to handle many concurrent users and be scalable to accommodate future growth in user base and demand. The application should deliver a responsive and smooth user experience, particularly for the real-time drawing interface and evaluation process. The application must implement robust security measures to protect user information and prevent unauthorized access or data breaches. The application should be compatible with various operating systems and devices, ensuring a consistent user experience across different platforms.

## [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 UML class diagram demonstrates essential object-oriented programming principles such as encapsulation, inheritance, and association. The Entity Class is the parent (super) class of the Game, Team, and Player classes. This means that Game, Team, and Player class, as Entity’s child classes, will inherit Entity’s attributes, while each being assigned attributes of their own, that are separate to the parent class. Encapsulation is evident in the class’s attributes and methods, encapsulating related data and behavior within each class. Inheritance is demonstrated through the relationships between classes, where the Game and Evaluation classes inherit properties and methods from their parent classes. Associations between classes establish relationships and enable data exchange between objects, as seen in the association between User and Drawing.

**"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** | Development on Mac platforms is well-suited for developers familiar with macOS environment. The team should have expertise in web development languages and macOS-specific tools. | The development team should have expertise in web development languages and frameworks suitable for Linux. If developers are already familiar with open-source tools and technologies, it may streamline the development process. | Windows developers should be proficient in web development languages and may have expertise in .NET technologies if applicable. | The two most common mobile devices are android and IOS. For Android development requires expertise in Java or Kotlin for native development, and web development skills for the web-based parts. For iOS developers should be proficient in web development languages and have expertise in iOS app development using Xcode and Swift or Objective-C for native features. |
| **Client Side** | Some tools, like Xcode, may have licensing costs for certain features, but many web development tools on Mac are free or have affordable licensing options. | Most development tools for web-based applications on Linux are open-source and free to use, reducing licensing costs. | Some development tools on Windows, like Visual Studio, may have licensing costs, but there are also free or open-source alternatives available. | Android Studio and the Android development environment are free to use, which can help minimize licensing costs. Xcode and the iOS development environment are available for free to Apple developers at reduced costs, reducing licensing costs for iOS development. iOS is known for its secure and user-friendly environment. It provides a high-quality user experience and offers strong security features. iOS is known for its secure and user-friendly environment. It provides a high-quality user experience and offers strong security features. |
| **Development Tools** | Mac developers often use Xcode, Visual Studio Code, or Sublime Text for coding and web development. For server-side development, they can utilize tools like PyCharm or WebStorm. | Developers may use text editors like Visual Studio Code, Sublime Text, or Vim for coding. For server-side development, they can use IDEs like PyCharm for Python or WebStorm for JavaScript. | Windows developers commonly use Visual Studio, Visual Studio Code, or Sublime Text for coding. For .NET development, Visual Studio is a popular choice. | Android Studio is the official IDE for Android development. For web-based components, developers can use standard web development tools like Visual Studio Code or WebStorm. iOS developers primarily use Xcode, the official IDE for iOS development, which includes robust tools for building web-based components. |

## 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 recommend the Gaming Room to use Linux servers for web hosting due to their low cost and high level of security. Proper configuration/setup may take a bit longer, but it is worth the time investment. Amazon Web Services (AWS) provides cloud computing options that meet our needs. These Linux distributions will be hosted on AWS Elastic Compute Cloud (EC2) virtual server instances.
2. **Operating Systems Architectures**: Targeted browsers are listed below allowing for current version support. Support for legacy versions will be determined based on market share but generally will not cover browsers more than two years old from the present date.

· Safari (WebKit)

· Chrome (Blink)

· Chrome (WebKit)

· Firefox (Gecko)

· Edge (Chromium)

· iOS (App Store)

1. **Storage Management**: Because the Draw It or Lose It game application uses roughly 1.6 GB of files within the application (not including executable size), the application should simply take advantage of the NFS file shares available within Linux to create a small, partitioned repository of images to pull from. This ensures that the game will be the only application to access these files and will reduce the chances of file-locks taking place.
2. **Memory Management**: Powerful and quick is Linux. Excellent memory management. This is because it makes system management easier while providing storage, backup, and recovery.

Several recommendations to ensure efficient utilization of memory resources:

1. Sizing game images correctly to their container size

2. Using the appropriate format; JPG is recommended

3. Using image compression to limit size

4. Use proper network request headers to enable image catching where appropriate

Note: use less RAM for management of memory

1. **Distributed Systems and Networks**: Using a cross-platform development environment will help limit the need for multiple expertise and make for a smoother application development process. To address connectivity or outage issues, I suggest ensuring the servers are built with the capacity that meets the client’s needs based on forecasting game usage/users once the application is launched to the new environments.
2. **Security**: From a backend and database perspective, hosting on Linux machines affords a high level of security. AWS allows manual security tasks to be automated and is a reputable industry leader with many clients that have top-secret workloads.

Authentication and authorization will be required for making changes to our stack in AWS. This strategy of authenticating users and allowing role-based access is also recommended on the client side. On the front end, these user roles limit who can edit game settings, allowing control only to team captains’, for example.

As far as native mobile devices are concerned, iOS comes with Data Protection, a built-in encryption/decryption feature that is automatic and hardware accelerated for storing files in the app directory.

Note, browser security comparisons are shown below:

· Safari

* good:

Prevents suspicious sites from loading and alerts potential danger.

* bad:

Not open-source, updates at irregular intervals

· Chrome

* good:

Largest market share at almost 80%

Auto updates every six-to-eight weeks.

* bad:

Google tracks everything to monetize information.

Close source, no way to see what is hidden in the code.

· Firefox

* good:

Best, latest security features

Allows “Content Blocking” to block trackers that the browser detects.

Completely open source

* bad:

Doesn’t update as frequently as Edge or Chrome but still at regular intervals.

· Edge

* good:

Now Chromium-based, open-source, and auto updates at least once a week

* bad:

IP addresses can be linked to backend servers via identifiers.