

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 | 02-04-2024 | Claude Brewer | Executive Summary, Design Constraints, System Architecture View, Domain Model |
| 2.0 | 02-18-2024 | Claude Brewer | Evaluation |
| 3.0 | 02-25-2024 | Claude Brewer | Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

This design document outlines the design approach for a web-based distributed game app. We plan to provide an engaging and interactive game experience to users. The solution leverages a web-based environment, allowing user to access the game from various web browsers and devices.

## [Design Constraints](#_2et92p0)

A few design constraints are network latency, scalability, and UI design. Since the game operates in a distributed environment then network latency could impact real-time interaction. To combat this, the design should incorporate efficient data transfer protocols and minimize the number of network round trips. We need to ensure that the app can accommodate a growing user base and handle the increase in traffic. Meaning we will need scalable infrastructure, have load balancing techniques, and employ caching mechanisms to optimize performance. Lastly, the UI of the app needs to be intuitive, responsive, and visually appealing. The UI also should provide clear user feedback, a guide through gameplay, and the support for different screen sizes and resolutions.

## [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 class diagram provides an overview of the main classes and their relationships. The Entity class is the base class that holds all the common attributes and behaviors shared amongst the classes. It shows the object-oriented programming principle of inheritance. The player class represents a player participating in a game holding player-specific information. The game class represents and instance of a game within the app. This contains info about the game’s state, players, and game-specific data.

**"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 can be used as a server for web-based software applications, but it's less commonly chosen compared to Linux or Windows. It offers a Unix-based environment, which is favorable for hosting web applications. Unix-based system provides robust security and stability. Mac OS Server includes features like Apache web server and PHP support. Limited hardware choices for server-grade Macs. It may be more expensive compared to Linux-based solutions. | Linux is a popular choice for hosting web-based software applications. It offers a wide range of distributions suitable for server use, with excellent stability and scalability. Cost-effective, highly customizable, and offers a wide range of server software options. Linux is known for its security and reliability. May require more technical expertise to set up and manage compared to Mac or Windows. | Windows Server is a commonly used platform for hosting web applications, especially for applications built using Microsoft technologies. Excellent integration with Microsoft tools and technologies. Offers a range of web server options like IIS. Good support for .NET applications. Licensing costs can be high. May not be as well-suited for open-source software development. | Mobile devices don't typically host web applications. Instead, servers are used to serve mobile app data. Any of the mentioned platforms (Mac, Linux, Windows) can be used for mobile app backends. Proximity to data for reduced latency. Cost-efficiency for specific use cases. Relatively easy deployment for small-scale applications. Limited resources can lead to performance issues. Scalability challenges. Reliability issues like battery drain and hardware failures. Security risks due to potential vulnerabilities. Network dependencies and potential bandwidth limitations. Maintenance and management complexities. |
| **Client Side** | Software Development Considerations: Developing for Mac clients typically involves using Apple's development tools such as Xcode. Costs can be moderate, and expertise in Swift and Objective-C may be required. | Developing for Linux clients can vary depending on the distribution and desktop environment. Costs are typically low, but expertise may be required for distribution-specific considerations. | Developing for Windows clients often involves using Visual Studio. Costs can vary, but expertise in .NET languages like C# may be required. | Developing for mobile devices involves platform-specific development. Costs can vary depending on the number of platforms targeted. Expertise in languages like Swift (iOS), Kotlin (Android), or cross-platform tools like Flutter may be required. |
| **Development Tools** | Relevant Programming Languages and Tools: Xcode is the primary integrated development environment (IDE) for Mac applications. It supports Swift, Objective-C, and C++. | Relevant Programming Languages and Tools: Linux supports a wide range of programming languages. Popular IDEs include Visual Studio Code, IntelliJ IDEA, and Eclipse. | Relevant Programming Languages and Tools: Visual Studio is the primary IDE for Windows development, supporting languages like C#, C++, and more. | Relevant Programming Languages and Tools: For iOS, Xcode and Swift/Objective-C are used. For Android, Android Studio and Kotlin/Java are common. Cross-platform tools like Flutter (Dart) can streamline development for both platforms. |

## 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 choosing a web-based operating platform. The game app can then be accessed and used across different devices and platforms, including desktops, laptops, tablets, and smartphones.
2. **Operating Systems Architectures**: The web-based operating system should support cross-platform compatibility. This will help with having as many people playing the game as possible.
3. **Storage Management**: Cloud-based storage should be used. This offers scalability, reliability, and accessibility. Google Cloud Storage could be used.
4. **Memory Management**: The recommended operating platform, being web-based, utilizes memory management techniques inherent to web browsers and the underlying operating systems. This includes features such as garbage collection, which automatically manages memory allocation and deallocation. By utilizing these memory management techniques, the platform ensures efficient use of system resources and minimizes memory leaks or crashes in the Draw It or Lose It software.
5. **Distributed Systems and Networks**: To enable communication between various platforms for Draw It or Lose It, a distributed software architecture can be implemented. This involves breaking down the application into smaller components or microservices that can run independently and communicate with each other over the network. RESTful APIs can be used for inter-component communication, allowing devices to interact with the game's backend and exchange data. It is important to consider dependencies between the components, handle connectivity issues, and implement fault-tolerant mechanisms to deal with network outages or disruptions.
6. **Security**: To protect user information on and between various platforms, it is crucial to implement robust security measures. The recommended operating platform should have built-in security capabilities, such as secure socket layers (SSL) for encrypted communication and authentication mechanisms. User data should be stored securely using encryption techniques, both at rest and during transmission. Additionally, access controls and user authentication should be implemented to ensure that only authorized individuals can access sensitive information. Regular security audits, vulnerability assessments, and updates should be performed to maintain a secure environment for Draw It or Lose It users