

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 | <07/18/24> | Robert Fjellin | Initial draft |
| 1.1 | <08/01/24> | Robert Fjellin | Expanded the Evaluation section |
| 1.2 | <08/14/24> | Robert Fjellin |  |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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 servers are reliable and secure but may not be the most cost effective and not very common. | Linux servers offer high performance, security, and flexibility, making them ideal for web-based applications. | Windows servers are user-friendly and widely utilized but can be costly and resource intensive. | Mobile devices are not typically used for hosting but can support lightweight server functionalities. |
| **Client Side** | Developing for Mac requires expertise in macOS development and can be costly due to higher hardware prices as Apple products tend to be pricier. | Linux development is cost-effective and offers a variety of open-source tools but requires expertise in Linux systems. | Windows development is widely supported with numerous tools available, but licensing costs can potentially be pretty high. | Mobile development requires consideration of different operating systems (iOS, Android) and user interfaces. |
| **Development Tools** | macOS development uses Xcode and Swift, which are robust but limited to Apple hardware. This will require an order of new hardware if the current hardware is not already Apple. | Linux offers a wide range of development tools (Eclipse, PyCharm, etc.) and supports many programming languages. | Windows development is supported by tools like Visual Studio and .NET, offering a rich development environment. | Mobile development uses tools like React Native, Flutter, and platform-specific IDEs like Android Studio and Xcode. |

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | It is possible to use for production but generally only used for development and testing. This is because there is very limited support for large-scale server deployments. There is also higher hardware and OS costs. | Linux is used widely for server deployments due to its stability and performance. Linux is open-source, very customizable and a lot of support is offered. Requires technical knowledge and skill to be able to setup and maintain. Linux is free, no licensing costs. | Most commonly used for enterprise environments and great integration with leading Microsoft services. Very user friendly and strong support for .NET apps. Licensing costs and requires more resource consumption compared to Linux. | Usually not used for server hosting, not ideal at all. The hardware and software are the big limiting factors for these devices. |
| **Client Side** | Requires Safari, Chrome and Firefox compatibility. Just like all OS will require knowledge in web development but needs to know macOS development tools. | Only needs to ensure compatibility with Chrome and Firefox. The cost is very low as most tools are open source. Linux knowledge is required in addition to Web Dev. | Edge, Chrome and Firefox compatibility. Licensing costs will be moderate probably more cost-effective than Mac. Needs to do cross browser testing. | Needs to ensure responsive design and compatibility. Without quick and fast response time, it could easily turn into a disaster. High cost as there is now a need for mobile-specific features which the others did not need. Also needs a knowledge of mobile web standards. |
| **Development Tools** | Swift, JS, HTML and CSS and possibly Objective C will need to be known. Xcode and visual studio can be used for development. Dev team will need a knowledge of Swift for iOS. Xcode is free but it comes at the cost of high costing hardware. | Languages, JS, HTML, CSS, Python, Ruby and PHP. VS Code, Sublime Text, Atom and other open source tools for IDE. Team will need to know Web Dev skills but most importantly experience in Linux environments. | C#, JS, HTML, CSS.. VS Code and Visual Studio. Team will need to know C# for .NET apps, may possibly even need specialized devs just for .NET.  VS is free so not too high in cost but enterprise editions will cost. | Java, Swift, JS, HTML, CSS..  Mobile is different because many languages will need to be known for the different mobile OS. Android Studio, Xcode, and VS Code. May end up needing multiple teams to work on each mobile OS. Costs will be high because of the need for more than one type of hardware. |

## 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 using Linux as the primary operating platform for expanding "Draw It or Lose It" due to its flexibility, performance, and cost-effectiveness.
2. **Operating Systems Architectures**: The chosen Linux architecture supports a wide range of hardware platforms and offers robust networking and security features.
3. **Storage Management**: Use a distributed database system like MongoDB or PostgreSQL, which provides scalability and high availability for the game data.
4. **Memory Management**: Linux uses advanced memory management techniques such as caching and swapping, which will help optimize the performance of the game application.
5. **Distributed Systems and Networks**: Implement microservices architecture to facilitate communication between various platforms. Use RESTful APIs and WebSockets to ensure real-time interaction and data synchronization. Consider using cloud services like AWS or Azure for reliable and scalable infrastructure.
6. **Security**: Implement SSL/TLS to secure communication between clients and servers. Utilize OAuth2 for user authentication and access control. Regularly update and patch the operating system and application to safeguard against vulnerabilities. Encrypt sensitive data during storage and transmission to protect user information across platforms. Despite a recent major outage affecting many companies, CrowdStrike could still offer excellent security.

## 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 using Linux as the primary operating platform due to its widespread use in server deployments, stability, performance, and cost-effectiveness.
2. **Operating Systems Architectures**: Linux provides robust resource management and security features, while its open-source nature allows for high customization, and scalability.
3. **Storage Management**: A distributed file system like Ceph or GlusterFS is ideal for providing scalability and reliable storage. They are also open-source falling in line with Linux.
4. **Memory Management**: Linux uses virtual memory and efficient RAM management techniques, like shared libraries and caching, to optimize memory usage for apps.
5. **Distributed Systems and Networks**: Implement RESTful Aps or gRPC for cross-platform communication, with cloud services and load balancers to ensure scalability and reliability.
6. **Security**: Secure data with SSL/TLS encryption, employ firewalls and intrusion detection, ensure regular updates and use strong authentication and RBA control to protect user information.