

Jose Munoz

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc173705292)

[Table of Contents 2](#_Toc173705293)

[Evaluation 3](#_Toc173705294)

[Recommendations 4](#_Toc173705295)

## [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** |  **Characteristics**: macOS can be used to host web-based applications but is not typically preferred for large-scale deployments.   **Advantages**: Good for development and testing; integrates well with other Apple products.   **Weaknesses**: Limited server-oriented features compared to Linux; higher licensing costs.   **Licensing Costs**: Higher than Linux, requiring macOS licenses and potentially more expensive hardware. |  **Characteristics**: Highly popular for web servers due to its stability, security, and open-source nature.   **Advantages**: Free and open-source; extensive support and community; highly customizable; robust security features.   **Weaknesses**: Requires more expertise to manage and configure; may have compatibility issues with some commercial software.   **Licensing Costs**: None for most distributions (e.g., Ubuntu, CentOS). |  **Characteristics**: Widely used for enterprise applications and offers good support for .NET applications.   **Advantages**: Easy to manage with a familiar GUI; excellent support from Microsoft; integrates well with other Microsoft products.   **Weaknesses**: Higher licensing costs; potentially more vulnerable to malware.   **Licensing Costs**: High, including Windows Server licenses and potentially additional CALs (Client Access Licenses). |  **Characteristics**: Not typically used to host web servers but important for client-side access.   **Advantages**: Ubiquitous and essential for end-user access; growing support for Progressive Web Apps (PWAs).   **Weaknesses**: Limited server capabilities; more relevant as clients rather than servers.   **Licensing Costs**: Varies based on platform (iOS requires Apple Developer Program subscription). |
| **Client Side** |  **Development Considerations**: Requires expertise in Swift/Objective-C for native apps; HTML5, CSS, JavaScript for web apps.   **Cost**: Higher due to Apple hardware and developer subscriptions.   **Time**: Moderate to high depending on app complexity and integration needs.   **Expertise**: Requires familiarity with Apple's ecosystem and development tools (Xcode). |  **Development Considerations**: Generally involves web technologies (HTML5, CSS, JavaScript); compatibility with multiple browsers.   **Cost**: Low; no licensing fees for Linux OS.   **Time**: Moderate, depends on browser compatibility and testing.   **Expertise**: Requires knowledge of open-source tools and potentially shell scripting. |  **Development Considerations**: Requires expertise in C#/ASP.NET for native apps; HTML5, CSS, JavaScript for web apps.   **Cost**: High due to Windows licenses and development tools (Visual Studio).   **Time**: Moderate to high, similar to Mac.   **Expertise**: Familiarity with Microsoft’s development stack and tools. |  **Development Considerations**: Expertise in Java/Kotlin for Android; Swift/Objective-C for iOS; responsive web design for cross-platform.   **Cost**: High due to need for multiple developer environments and testing devices.   **Time**: High, requires development and testing across multiple platforms.   **Expertise**: Requires skills in mobile development and responsive web design. |
| **Development Tools** |  **Languages and Tools**: Swift, Objective-C, Xcode.   **Impact on Development Team**: May require a dedicated team for macOS/iOS development.   **Licensing Costs**: Apple Developer Program subscription. |  **Languages and Tools**: HTML5, CSS, JavaScript, various IDEs (e.g., Eclipse, VS Code).   **Impact on Development Team**: Requires knowledge of open-source tools and possibly multiple distributions.   **Licensing Costs**: None for most tools. |  **Languages and Tools**: C#, ASP.NET, Visual Studio.   **Impact on Development Team**: May require a dedicated team for Windows development.   **Licensing Costs**: Visual Studio licenses, Windows licenses. |  **Languages and Tools**: Java/Kotlin (Android), Swift/Objective-C (iOS), various IDEs (Android Studio, Xcode).   **Impact on Development Team**: Requires a diverse team with expertise in multiple mobile platforms.   **Licensing Costs**: Apple Developer Program, potentially other tool subscriptions. |

## 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**: Linux is the most suitable operating platform due to its cost-effectiveness, stability, and strong community support. It’s ideal for hosting web applications and can be integrated into a distributed environment with other platforms for client access.
2. **Operating Systems Architectures**: Linux-based servers with potential use of containers (Docker) for scalability and microservices. Front-end development should use responsive web design to ensure compatibility across Mac, Windows, and mobile devices.
3. **Storage Management**: Use of cloud storage solutions (e.g., AWS S3, Google Cloud Storage) to provide scalable, reliable, and secure storage.
4. **Memory Management**: Leverage Linux’s robust memory management features, including virtual memory, and caching mechanisms, to optimize performance.
5. **Distributed Systems and Networks**: Implement RESTful APIs for communication between server and clients. Use load balancers and CDNs to manage traffic and ensure high availability.
6. **Security**: Use HTTPS for secure communication, implement strong authentication (OAuth), and employ encryption for sensitive data. Regular security audits and adherence to best practices will protect user data across platforms.