Chat Away

# Software Design Template

Version 1.0

## Table of Contents

[Software Design Template 1](#_Toc21086455)

[Table of Contents 2](#_Toc21086456)

[Document Revision History 2](#_Toc21086457)

[Executive Summary 3](#_Toc21086458)

[Requirements 3](#_Toc21086459)

[Design Constraints 3](#_Toc21086459)

[Rationale 3](#_Toc21086460)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/26/2024 | Christian Wallace | Initial draft of software design document for Chat Away |

## Executive Summary

Chat Away, a successful social media company aims to expand its reach by developing a mobile application for iPhone and Android smartphones. This project is crucial for marketing their product to a larger customer base and increasing revenue. The client seeks to outsource this development project to stay within their budget constraints.

## Requirements

* **Business Requirements:**
  1. Develop a mobile application to complement the existing website.
  2. The app should be available on both iPhone and Android platforms.
  3. The project must stay within the client’s budget.
* **Technical Requirements:**
  1. The app must provide a seamless user experience consistent with the website.
  2. Integration with existing backend systems and APIs.
  3. Ensure compliance with app store guidelines for both iOS and Android platforms.
  4. Utilize a cross-platform development framework to optimize resource use and maintain a single codebase.

## Design Constraints

1. **Budget Constraint:**
   * The development must be cost-effective. Using a cross-platform framework like Flutter or React Native can reduce costs by allowing a single codebase for both iOS and Android apps, minimizing the need for separate development teams and reducing maintenance costs.
2. **Platform Compliance:**
   * The app must comply with both Apple App Store and Google Play Store guidelines. This involves adhering to each platform’s UI/UX standards and submission requirements, which can constrain design and development choices to ensure approval and smooth deployment.
3. **Integration with Existing Systems:**
   * The app must integrate seamlessly with Chat Away's existing website and backend infrastructure. This requires using compatible technologies and possibly updating or adapting existing APIs to ensure smooth data flow and user experience continuity between the web and mobile platforms.

## Rationale

1. **Budget Constraint:**
   * **Rationale:** Staying within budget is critical for the client. A cross-platform development approach is both a business and technical constraint, as it limits the choice of development frameworks and offers a practical solution to meet financial requirements.
2. **Platform Compliance:**
   * **Rationale:** Ensuring compliance with app store guidelines is essential to prevent rejection during submission. This constraint is both business and technical, as non-compliance can delay the app’s launch and incur additional costs for revisions.
3. **Integration with Existing Systems:**
   * **Rationale:** Integration is crucial to maintain a consistent user experience and leverage existing infrastructure. This constraint is primarily technical, as it dictates the choice of development tools and technologies to ensure compatibility with current systems.

**Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Server-Side** | **Licensing Costs** | **Client-Side** | **Development Tools** |
| **Linux** | Offers robust server-based deployment with high scalability and security. Widely used for web servers (e.g., Apache, Nginx). | Generally low as Linux is open-source and free. Some enterprise distributions (e.g., Red Hat) may have costs. | Compatible with modern web browsers (Chrome, Firefox). Can run web apps efficiently. | Development tools (GCC, Clang, Make) are free. IDEs like Eclipse, VS Code are also free. Minimal licensing costs. |
| **Mac** | Can host web servers using macOS Server or open-source solutions (Apache, Nginx). Less common than Linux for server use. | Higher cost due to macOS licensing and hardware requirements. | Compatible with Safari, Chrome, Firefox. Excellent for development of macOS and iOS apps. | Xcode is free, but macOS hardware is required, increasing costs. Other tools (Homebrew, VS Code) are free. |
| **Windows** | Widely used for hosting web applications via IIS or open-source servers (Apache, Nginx). Good support for enterprise environments. | Licensing costs for Windows Server. Potentially higher costs for enterprise solutions. | Compatible with Edge, Chrome, Firefox. Widely used in enterprise environments. | Development tools like Visual Studio can be costly. Free versions available. Licensing for Windows OS adds to costs. |
| **Android** | Typically not used for hosting web servers. Focuses on mobile client-side application. | No direct licensing cost for Android OS, but app store fees apply. | Requires development for various screen sizes and OS versions. Uses Chrome browser. | Android Studio is free. Other tools (Gradle, Kotlin, Java) are also free. Google Play Store registration costs apply. |
| **iOS** | Not used for hosting web servers. Focuses on mobile client-side application. | Higher licensing costs due to Apple’s ecosystem. App store fees apply. | Requires development for various devices and iOS versions. Uses Safari browser. | Xcode is free, but macOS hardware required. Apple Developer Program registration costs apply. |

#### Linux

**Server-Side:**

* Pros: Robust, scalable, secure, widely used for web hosting with open-source solutions like Apache and Nginx.
* Cons: May require more technical expertise for setup and management.

**Licensing Costs:**

* Low, as Linux distributions are typically free. Enterprise distributions (e.g., Red Hat) may have costs.

**Client-Side:**

* Compatible with all major web browsers. Efficient for running web applications.

**Development Tools:**

* Tools like GCC, Clang, Make are free. IDEs like Eclipse and VS Code are also free. Minimal licensing costs.

#### Mac

**Server-Side:**

* Pros: Can host web servers using macOS Server or open-source solutions. Good for development of macOS/iOS apps.
* Cons: Less common for server use, higher hardware costs.

**Licensing Costs:**

* Higher due to macOS licensing and hardware requirements.

**Client-Side:**

* Compatible with Safari, Chrome, Firefox. Excellent for macOS and iOS app development.

**Development Tools:**

* Xcode is free, but macOS hardware is required. Other tools (Homebrew, VS Code) are free.

#### Windows

**Server-Side:**

* Pros: Widely used for hosting web applications via IIS or open-source servers. Good enterprise support.
* Cons: Higher licensing costs for Windows Server.

**Licensing Costs:**

* Higher due to Windows Server licensing.

**Client-Side:**

* Compatible with Edge, Chrome, Firefox. Widely used in enterprise environments.

**Development Tools:**

* Visual Studio can be costly, though free versions are available. Licensing for Windows OS adds to costs.

#### Android

**Server-Side:**

* Not typically used for hosting web servers. Focus is on mobile applications.

**Licensing Costs:**

* No direct cost for Android OS. App store fees apply.

**Client-Side:**

* Requires development for various screen sizes and OS versions. Uses Chrome browser.

**Development Tools:**

* Android Studio and tools like Gradle, Kotlin, Java are free. Google Play Store registration costs apply.

#### iOS

**Server-Side:**

* Not used for hosting web servers. The focus is on mobile applications.

**Licensing Costs:**

* Higher due to Apple’s ecosystem and app store fees.

**Client-Side:**

* Requires development for various devices and iOS versions. Uses the Safari browser.

**Development Tools:**

* Xcode is free but requires macOS hardware. Apple Developer Program registration costs apply.

### Conclusion

Each platform has unique characteristics, advantages, and weaknesses. For server-side deployment, Linux offers a cost-effective and scalable solution, while Windows provides strong enterprise support at a higher cost. Mac can serve both server and client roles but at a higher hardware cost. Android offers flexibility and lower costs for mobile platforms, while iOS development requires investment in Apple's ecosystem.

**Recommendations**

**Operating Platform**

For The Gaming Room's expansion of "Draw It or Lose It" to multiple platforms, Linux is the recommended operating platform for the server-side deployment.

**Operating Systems Architectures**

Linux offers a robust, scalable, and secure environment ideal for web hosting. It is widely used for web servers with open-source solutions such as Apache and Nginx. It supports extensive customization and optimization for different workloads, providing flexibility for the gaming application to scale up to thousands of players. Additionally, Linux's open-source nature means lower licensing costs, making it a cost-effective choice for the client.

**Storage Management**

The appropriate storage management system for the Linux platform is the Ext4 (Fourth Extended Filesystem). Ext4 is a widely used filesystem for Linux that offers high performance, reliability, and large storage capacity. It supports journaling, which helps protect data integrity in case of power failures or system crashes and provides features like extents for better performance and reduced fragmentation.

**Memory Management**

**Linux uses several sophisticated memory management techniques, including:**

Virtual Memory Management: Allows the system to use disk space as an extension of RAM, providing the illusion of a larger memory capacity.

Demand Paging: Loads pages into memory only when they are needed, which optimizes memory usage.

Swapping: Moves inactive pages to a swap space on disk, freeing up RAM for active processes.

Buffer and Cache Management: Uses buffers and cache to store frequently accessed data, reducing I/O operations and speeding up access times.

These techniques ensure efficient use of memory resources, which is crucial for the smooth operation of the "Draw It or Lose It" application, especially under high loads.

**Distributed Systems and Networks**

The game can be implemented as a distributed system using the microservices architecture to enable communication between various platforms. This approach involves breaking down the application into smaller, independent services that communicate through APIs. Each service can be developed, deployed, and scaled independently, allowing for greater flexibility and resilience.

For the network infrastructure, RESTful APIs over HTTP/HTTPS can be used for communication between the client-side applications (web browsers and mobile devices) and the server-side services. This setup ensures compatibility across different platforms and devices.

**Security**

**Ensuring the security of user information is paramount. The recommended security measures include:**

Encryption: Use HTTPS to encrypt data transmitted between clients and the server. Employ SSL/TLS certificates to secure the communication channels.

Authentication and Authorization: Implement robust authentication mechanisms (e.g., OAuth, JWT) to verify user identities and control access to resources.

Data Protection: Store sensitive user data, such as passwords, using strong hashing algorithms (e.g., bcrypt). Ensure regular backups and secure storage solutions to protect against data loss or breaches.

Regular Security Audits: Conduct regular security audits and vulnerability assessments to identify and mitigate potential risks.

These measures will help protect user data and ensure the integrity and confidentiality of information as it travels between various platforms.