<Chat Away>

**Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 05/19/2024 | Lilaisha Harris | Initial draft with executive summary, requirements, design constraints, and rationale. |

Instructions: Fill in all bracketed information on page one (the cover page), in the Document Revision History table, in the footer, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

**Executive Summary**

*<*Chat Away, a successful social media company, seeks to develop a mobile application to expand their market reach and increase revenue. They require the app to be available on both iPhone and Android platforms. The company aims to outsource this project to stay within their budget.*>*

**Requirements**

*<*Business Requirements:

The mobile app must be available on both iPhone (iOS) and Android platforms to maximize market reach.

The app should be developed within a limited budget as the project is outsourced to stay cost-effective.

Technical Requirements:

The app must be compatible with the latest versions of both iOS and Android operating systems.

The app should integrate seamlessly with the existing Chat Away website and backend systems.

It must include features that are standard for social media applications, such as user profiles, messaging, notifications, and media sharing.*>*

**Design Constraints**

*<*Cross-Platform Development:

Constraint: The need to develop the app for both iOS and Android platforms.

Rationale: This requirement can be addressed by using cross-platform development frameworks like React Native or Flutter. These tools allow for a single codebase to be deployed on both platforms, reducing development time and cost.

Budget Constraints:

Constraint: The project must be completed within a tight budget.

Rationale: Outsourcing the project and using cross-platform frameworks can help manage costs. Additionally, leveraging open-source libraries and tools can reduce licensing fees and development expenses.

Integration with Existing Systems:

Constraint: The app must integrate with the existing website and backend.

Rationale: Ensuring compatibility with existing APIs and data structures is crucial. This may require thorough documentation of current systems and possibly adapting the backend to support mobile app functionalities.*>*

**Rationale**

*<*Cross-Platform Development:

Why it's a constraint: Developing separate native apps for iOS and Android would significantly increase costs and development time.

Business or Technical Constraint: Both.

Budget Constraints:

Why it's a constraint: Staying within budget is essential to prevent overspending and ensure project feasibility.

Business or Technical Constraint: Business.

Integration with Existing Systems:

Why it's a constraint: Seamless integration is necessary for data consistency and user experience across the web and mobile platforms.

Business or Technical Constraint: Technical.*>*

**Evaluation**

**Server Side**

**Linux**

**06/09/2024**

- Characteristics: Open-source, widely used for server environments, strong community support.

- Advantages:

- Cost-effective (no licensing fees for many distributions).

- High stability and security, making it ideal for web servers.

- Flexible and customizable to specific needs.

- Supports a wide range of server applications (e.g., Apache, Nginx).

- Weaknesses:

- Requires expertise in Linux system administration.

- Potential compatibility issues with certain proprietary software.

**Mac**

- Characteristics: Unix-based, known for its robust performance and security.

- Advantages:

- Stable and secure with a user-friendly interface.

- Integration with other Apple products and services.

- Weaknesses:

- Higher cost due to expensive hardware.

- Less common in server environments compared to Linux.

- Limited customization options compared to Linux.

**Windows**

- Characteristics: Widely used, user-friendly, extensive support for various applications.

- Advantages:

- Easy to use and manage, especially for those familiar with Windows.

- Strong support for enterprise applications and integration with Microsoft services.

- Large user base and extensive documentation.

- Weaknesses:

- Licensing costs can be significant.

- Historically more vulnerable to security threats.

- Requires more frequent updates and patches.

**Client Side**

**Development Considerations:**

- Cost:

- Developing a responsive web application that runs on all platforms is generally more cost-effective than developing native applications for each platform.

- Using frameworks like React Native or Flutter can further reduce costs by allowing code reuse across mobile platforms.

- Time:

- Developing a web application can be faster due to the use of existing web technologies and frameworks.

- Cross-platform frameworks accelerate the development process by enabling simultaneous deployment on iOS and Android.

- Expertise:

- Requires proficiency in web technologies (HTML, CSS, JavaScript) and familiarity with responsive design principles.

- Knowledge of cross-platform development frameworks (React Native, Flutter) is beneficial.

**Linux, Mac, Windows (Desktop Clients)**

- Characteristics: All three platforms support modern web browsers (Chrome, Firefox, Safari, Edge).

- Advantages:

- No additional software installation required; applications run directly in the browser.

- Consistent user experience across different desktop environments.

- Easier maintenance and updates since changes can be deployed to the web server.

- Weaknesses:

- Performance may vary depending on browser efficiency and system resources.

- Some advanced functionalities might be limited compared to native applications.

**Mobile Platforms (iOS and Android)**

- Characteristics: Mobile devices require a responsive design to ensure usability on smaller screens.

- Advantages:

- Wide reach due to the popularity of mobile devices.

- Potential for better user engagement through mobile-friendly interfaces.

- Weaknesses:

- Requires thorough testing to ensure compatibility across various devices and screen sizes.

- May need optimization to handle performance constraints of mobile hardware.

**Development Tools**

**Programming Languages and Tools:**

- Web Application:

- Languages: HTML, CSS, JavaScript.

- Frameworks: React, Angular, Vue.js for front-end development.

- Tools: Visual Studio Code, WebStorm, browser developer tools.

- Cross-Platform Mobile Development:

- Frameworks: React Native, Flutter.

- Tools: Android Studio, Xcode (for iOS), Visual Studio Code.

- Server-Side Development:

- Languages: JavaScript (Node.js), Python, Ruby, PHP.

- Frameworks: Express.js (Node.js), Django (Python), Ruby on Rails.

- Tools: VS Code, PyCharm, RubyMine, Git for version control.

**Impact on Development Team:**

- Multiple teams may be required to handle different aspects of the project (e.g., front-end, back-end, mobile development).

- Cross-training team members on different frameworks and tools can improve flexibility and efficiency.

- Using open-source tools and frameworks can minimize licensing costs and reduce overall expenses.

**Conclusion:**

By evaluating the characteristics, advantages, and weaknesses of Linux, Mac, Windows, and mobile platforms, Creative Technology Solutions can make an informed decision on the best deployment strategy for "Draw It or Lose It." This comprehensive analysis ensures that the application will be scalable, cost-effective, and compatible across multiple platforms, meeting the client's requirements for expanding their game application.

Here's a detailed recommendation for The Gaming Room, focusing on the characteristics of various systems architectures and how they can support the expansion of "Draw It or Lose It" across different computing environments:

**Recommendations**

**Operating Platform**

**Recommendation: Linux Server Platform**

**06/23/2024**

**Rationale:** Linux is widely regarded as a robust, secure, and cost-effective server platform, suitable for scaling and supporting a multi-platform application like "Draw It or Lose It." Its open-source nature, extensive community support, and flexibility in customization make it ideal for both web and game server environments.

**Operating Systems Architectures**

**Linux Architecture Details:**

- \*\*Kernel:\*\* Monolithic kernel, providing efficient performance and comprehensive control over hardware resources.

- \*\*File System:\*\* Ext4, Btrfs, or ZFS can be used for high performance and reliability.

- \*\*Process Management:\*\* Utilizes a preemptive multitasking system, allowing efficient management of game server processes.

- \*\*User Interface:\*\* Command-line interface (CLI) primarily, with graphical user interface (GUI) options like GNOME or KDE available if needed for certain administrative tasks.

- \*\*Security:\*\* Implements various security features like SELinux, AppArmor, and built-in firewall capabilities to protect the server and user data.

**Storage Management**

**Recommendation: Network Attached Storage (NAS) with ZFS**

**Rationale:** ZFS offers advanced storage management features such as high data integrity, compression, snapshots, and support for large storage capacities. A NAS setup provides scalable storage solutions, allowing easy expansion as the game's data grows.

**Memory Management**

**Linux Memory Management Techniques:**

- \*\*Paging and Swapping:\*\* Efficient use of physical memory and disk space, enabling large game data to be managed effectively.

- \*\*Caching:\*\* Utilizes disk caching to speed up read/write operations, enhancing game performance.

- \*\*Buffering:\*\* Reduces latency in game data processing by buffering inputs and outputs.

- \*\*Virtual Memory:\*\* Extends the available memory using disk space, allowing the game to handle more data than what is physically available in RAM.

**Distributed Systems and Networks**

**Communication in Distributed Systems:**

- \*\*Middleware:\*\* Use of middleware like Apache Kafka or RabbitMQ for message brokering to ensure reliable communication between game instances on different platforms.

- \*\*API Gateway:\*\* An API Gateway can manage and route API requests between different microservices, ensuring seamless integration and interaction.

- \*\*Load Balancing:\*\* Implement load balancers to distribute incoming game traffic evenly across multiple servers, ensuring high availability and reliability.

- \*\*Redundancy and Failover:\*\* Set up redundant systems and failover mechanisms to handle server outages without affecting the game's availability.

**Dependencies and Considerations**:

- \*\*Connectivity:\*\* Ensure high-speed, reliable network connections between servers and clients to minimize latency and provide a smooth gaming experience.

- \*\*Scalability:\*\* Plan for horizontal scaling to accommodate increasing numbers of players by adding more servers.

- \*\*Monitoring and Management:\*\* Use tools like Prometheus and Grafana for monitoring system performance and identifying potential issues proactively.

**Security**

\*\*User Protection and Security Measures:\*\*

- \*\*Data Encryption:\*\* Implement SSL/TLS encryption for all data transmitted between clients and servers to protect user information from interception.

- \*\*Authentication and Authorization:\*\* Use robust authentication mechanisms (OAuth, JWT) and role-based access control to ensure that only authorized users can access the game services.

- \*\*Intrusion Detection:\*\* Employ intrusion detection systems (IDS) to monitor and respond to suspicious activities in real-time.

- \*\*Regular Updates:\*\* Keep the operating system and all software components updated to protect against known vulnerabilities.

- \*\*Firewall and Security Groups:\*\* Configure firewalls and security groups to restrict access to critical services and data.

**Conclusion**

By leveraging a Linux-based server platform, implementing advanced storage and memory management techniques, and utilizing robust security measures, The Gaming Room can effectively expand "Draw It or Lose It" to various computing environments. This setup ensures high performance, scalability, and security, providing a seamless gaming experience across multiple platforms.