Draw It or Lose It

CS 230 Project Software Design

Version 3.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	5/19//24	Drew Sibila	Initial Version
2.0	6/7/24	Drew Sibila	Added detailed descriptions for system architecture view and domain model. Updated evaluation section to include more comprehensive platform analysis.
3.0	6/20/24	Drew Sibila	Recommendations Section Completed

Executive Summary

Client: Draw It or Lose It

Problem Summary: The Gaming Room wants to open the reach of its main game, "Draw It or Lose It," beyond Android to Linux, Mac, Windows, iOS, and Android platforms. We need a good look through of an assessment of each platform's strengths, weaknesses, and unique features to make informed decisions. Key challenges include selecting server deployment methods, assessing licensing costs, and addressing development needs across different client platforms. Our goal is to make the gameplay better across multiple platforms, boost market competitiveness, and optimize deployment strategies.

Requirements

- Cross-Platform Compatibility: Support for Windows, Linux, Mac, iOS, and Android.
- Scalability: Ability to integrate new features and handle increasing user loads.
- Performance and User Experience: Ensure high usability while meeting performance targets.
- Security and Privacy: Compliance with data protection laws and prioritization of user data security.
- Cost-Effectiveness: Maintain a budget-friendly deployment approach.

Design Constraints

Cross-Platform Development:

- Constraint: Support iOS and Android from a single codebase.
- Rationale: Use cross-platform frameworks like React Native or Flutter to reduce development time and costs effectively.

Budget Limitations:

- Constraint: Adhere to a fixed budget.
- Rationale: Utilize open-source libraries and pay-as-you-go cloud services for cost-effective project management.

Performance and User Experience:

- Constraint: Meet specific performance benchmarks.
- Rationale: Optimize architecture, employ efficient coding practices, and conduct rigorous testing for a seamless user experience.

Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	MacOS can host web applications effectively due to its Unix foundation, but it's less commonly used as a server OS. This leads to limited support and higher costs compared to other platforms like Linux.	Linux is ideal for hosting web applications due to its stability, low cost, and strong community support. Its a preferred choice for many web servers because of its flexibility and scalability.	Windows Server is a popular choice for hosting web applications. It offers a user-friendly interface and good integration with other Microsoft products. However, it can be expensive due to licensing fees.	Mobile platforms do not host web applications directly. Instead, they rely on backend services hosted on other platforms like Linux or Windows. The mobile apps communicate with these backend services via APIs.

Client Side	For Mac, software development requires using tools like Xcode, which supports Swift and Objective-C. The cost can be higher due to the need for Apple hardware. Developers need expertise in macOS and iOS development.	Linux is highly flexible and cost-effective for development. However, it may require more expertise due to its open-source nature.	Windows development uses tools like Visual Studio, which supports multiple languages like C#, JavaScript, and HTML/CSS. Developers benefit from extensive support and integration with other Microsoft services.	Mobile development requires expertise in native languages (Swift for iOS, Kotlin/Java for Android) and cross-platform frameworks (React Native, Flutter). Development costs and time can be significant due to supporting multiple platforms.
Development Tools	Xcode for macOS and iOS development using Swift and Objective-C.	Eclipse, PyCharm, Visual Studio Code for Python, Java, C++, and JavaScript development.	Eclipse, PyCharm, Visual Studio Code for Python, Java, C++, and JavaScript development.	Xcode for iOS and Android Studio for Android; cross-platform tools like Flutter and React Native enable unified development.

Recommendations

Operating Platform

Recommendation: For server-side hosting, use Linux because it is stable, flexible, and cost-effective. For client-side, use cross-platform frameworks like Flutter or React Native to support both iOS and Android, as well as desktop systems like Windows, Mac, and Linux.

Rationale: Linux offers robust, scalable, and cost-effective server-side solutions. Cross-platform frameworks like Flutter or React Native reduce development time and costs by allowing a single codebase for multiple platforms.

Operating Systems Architectures

Description: Linux-based server architectures support modular and microservices-based designs, enabling easy scalability and maintenance. This architecture facilitates seamless updates and integration of new features without significant downtime. Additionally, it supports various databases and web servers, ensuring compatibility with different components of the game application.

Rationale: The stability, flexibility, and cost-effectiveness of Linux make it an ideal choice for server-side deployment. Its strong community support and vast array of available tools further enhance its suitability for hosting web applications.

Storage Management

Recommendation: Use cloud storage solutions like AWS S3 or Google Cloud Storage. They are scalable, reliable, and cost-effective, providing strong security and easy integration.

Rationale: Cloud storage solutions offer high availability, fault tolerance, and scalability, essential for handling large volumes of game data. They provide flexible, on-demand storage that scales with the user base, ensuring cost-effectiveness and reliability.

Memory Management

Explanation: Linux employs advanced memory management techniques such as paging, segmentation, and virtual memory to optimize the use of physical memory. These techniques ensure efficient allocation and management of memory resources, which is crucial for the performance of the "Draw It or Lose It" game. The use of containerization (e.g., Docker) further enhances memory management by isolating processes and minimizing overhead.

Rationale: The efficient memory management capabilities of Linux, combined with containerization, ensure optimal performance and resource utilization, which are critical for maintaining a smooth user experience.

Distributed Systems and Networks

Explanation: To enable communication between various platforms, the game can leverage RESTful APIs and WebSockets for real-time interaction. These technologies allow different components of the game to communicate efficiently over the network. Using a content delivery network (CDN) can also improve the game's performance by caching and delivering content from servers closer to the end-users, reducing latency and improving load times.

Considerations: Ensuring high availability and fault tolerance is critical. Implementing load balancing and redundancy strategies will mitigate the impact of connectivity issues and outages, ensuring a seamless gaming experience for users.

Security

Explanation: Security is paramount for protecting user data and maintaining trust. The recommended platform should implement strong encryption protocols (e.g., SSL/TLS) for data in transit and at rest. Additionally, regular security audits, intrusion detection systems (IDS), and firewalls will safeguard the infrastructure from potential threats. Employing OAuth2 for authentication and authorization ensures secure access control across different platforms.

Considerations: The platform should comply with relevant data protection regulations (e.g., GDPR) and incorporate measures to protect against common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Conclusion

By adopting the recommended Linux-based server platform, along with the outlined memory and storage management techniques, The Gaming Room can successfully expand "Draw It or Lose It" to various computing environments. Implementing robust distributed systems and network strategies, coupled with stringent security measures, will ensure a scalable, high-performance, and secure gaming experience for users across different platforms.