MK Fowlkes

Project Three CS 230

[Paul.fowlkes@snhu.edu](mailto:Paul.fowlkes@snhu.edu)

Recommendations:

Operating Platform:

Recommendation: Linux-based Server

Linux is a robust and widely used operating system that offers excellent scalability and performance. It is open-source, which means cost-effective implementation, and has a vast community support for troubleshooting and updates.

Operating Systems Architectures:

Details of the Chosen Operating Platform Architectures:

1. Linux Kernel:

- Linux supports a monolithic kernel architecture, providing efficiency and control over system resources.

- For scalability, Linux supports multi-core processors, enabling parallel processing for better performance.

Storage Management:

Identify an Appropriate Storage Management System:

1. File System:

- Utilize a robust file system like ext4, which is commonly used on Linux systems.

- Implement a distributed file system for scalability and fault tolerance, such as GlusterFS or Ceph.

2. Database Management System (DBMS):

- Consider using a reliable DBMS like MySQL or PostgreSQL for structured data storage.

- For unstructured data, implement a NoSQL database like MongoDB.

Memory Management:

Explanation of Recommended Memory Management Techniques:

1. Virtual Memory:

- Linux provides efficient virtual memory management, allowing the system to use disk space as an extension of RAM.

2. Caching:

- Utilize caching mechanisms to enhance performance, such as memcached or Redis.

Distributed Systems and Networks:

Communication Between Various Platforms:

1. RESTful APIs:

- Implement RESTful APIs to enable communication between different platforms.

- Use JSON or XML for data interchange to ensure compatibility.

2. Message Queues:

- Utilize message queues (e.g., RabbitMQ or Apache Kafka) to decouple components, ensuring asynchronous communication and fault tolerance.

3. Load Balancing:

- Employ load balancing techniques to distribute traffic efficiently across multiple servers.

Considerations for Dependencies:

1. Connectivity:

- Implement redundant network connections to mitigate connectivity issues.

- Use protocols like MQTT for reliable communication in intermittent network conditions.

2. Outages:

- Implement failover mechanisms to ensure uninterrupted service during server outages.

- Regularly test disaster recovery plans to minimize downtime.

Security:

User Information Protection and Security Capabilities:

1. Encryption:

- Implement end-to-end encryption for data transmission.

- Use protocols like HTTPS to secure communication channels.

2. Access Control:

- Employ robust access control mechanisms to restrict unauthorized access.

- Implement role-based access control (RBAC) to manage user permissions.

3. Authentication and Authorization:

- Use strong authentication methods (e.g., OAuth, JWT) for user validation.

- Implement authorization checks to ensure users have the necessary privileges.

4. Regular Audits and Monitoring:

- Conduct regular security audits to identify vulnerabilities.

- Implement real-time monitoring tools to detect and respond to security threats promptly.

By following these recommendations, The Gaming Room can ensure the scalability, reliability, and security of the Draw It or Lose It software across various computing environments.