Research Document: Security and Performance Optimisation for Social Networking Platforms

This document explores critical strategies for enhancing security and performance in social networking platforms. It covers three core topics: implementing safe and restricted HTML/CSS customisation to mitigate security risks, applying best practices to secure databases effectively, and optimising system performance for real-time user interactions such as messaging and posting.

Each section provides detailed recommendations, backed by industry insights and practical examples, aiming to guide developers and platform administrators in designing robust, scalable, and secure social networking environments.

Date: **04.06.2025**

Author: **Andrii Matviienko**

Implementing Safe and Restricted HTML/CSS Customisation

Allowing users to customise HTML and CSS on a platform introduces risks, notably cross-site scripting (XSS) and cross-site request forgery (CSRF). To prevent these vulnerabilities, enforcing a strict content security policy (CSP) is essential. This policy should restrict frame ancestors, block mixed content, and provide mechanisms to report violations. A sandboxed iframe environment can safely isolate user customisations.

Utilising a whitelist approach for permissible tags and CSS properties is effective; libraries like HTML Purifier have demonstrated high effectiveness at preventing XSS attacks. Popular platforms, such as GitHub, apply similar restrictions by limiting CSS to specific classes. Relying on DOM-based sanitisation libraries such as DOMPurify or HTML Purifier provides robust protection against XSS, outperforming traditional regex-based filters. Additionally, rate limiting user customisation actions and auditing changes prior to deployment further reinforce security.

Best Practices for Database Security in a Social Networking Platform

Database security is paramount due to sensitive personal data handled by social networks. The principle of least privilege should be strictly followed for database accounts, as misuse of privileges remains a significant contributor to data breaches. Data encryption both at rest (using AES-256) and in transit (via TLS 1.3) is recommended by authoritative bodies like the UK's NCSC.

Strong authentication and authorisation protocols, including OAuth 2.0 and OpenID Connect, safeguard access. Preventing SQL injection through prepared statements and parameterised queries is critical. Activity monitoring with SIEM systems, consistent backups adhering to the 3-2-1 rule, data masking of personally identifiable information (PII), and multi-factor authentication (MFA) for administrative access greatly enhance security. Tools like HashiCorp Vault assist in secure secrets management, with database firewalls adding an extra defensive layer.

Optimising Performance for Real-Time Interactions

Real-time responsiveness is a key user expectation in social networking. Implementing WebSockets enables persistent connections that significantly reduce latency compared to traditional HTTP polling. Asynchronous message processing using queuing systems such as RabbitMQ or Kafka ensures smooth handling of concurrent operations.

Performance further benefits from caching frequently accessed data with in-memory stores like Redis or Memcached, and utilising Content Delivery Networks (CDNs) for static assets. Database query optimisation and indexing can reduce query execution times by up to 80%. Connection pooling decreases overhead, while load balancers distribute traffic efficiently. Compression techniques like Gzip and Brotli mitigate data transfer sizes, complemented by minimising and refining JavaScript code. Continuous performance monitoring with tools like New Relic enforces proactive optimisation.

Optimising Messaging Performance

Messaging systems require high efficiency to provide seamless communication. Pagination of message history significantly reduces initial load times, letting users access recent messages quickly while older ones load gradually.

Applying compression (e.g., Gzip) to message data significantly lowers bandwidth usage, speeding up delivery. At the database level, optimising queries with proper indexing and caching active conversations (using Redis or similar) improves responsiveness.

For real-time updates, WebSockets enable persistent connections, cutting latency compared to HTTP polling. Adding delivery confirmations ensures messages reliably reach recipients.

Rate limiting prevents spam and abuse by controlling message frequency, protecting system stability. To handle high loads smoothly, distributed message queues like RabbitMQ or Kafka support scalable, asynchronous message processing.

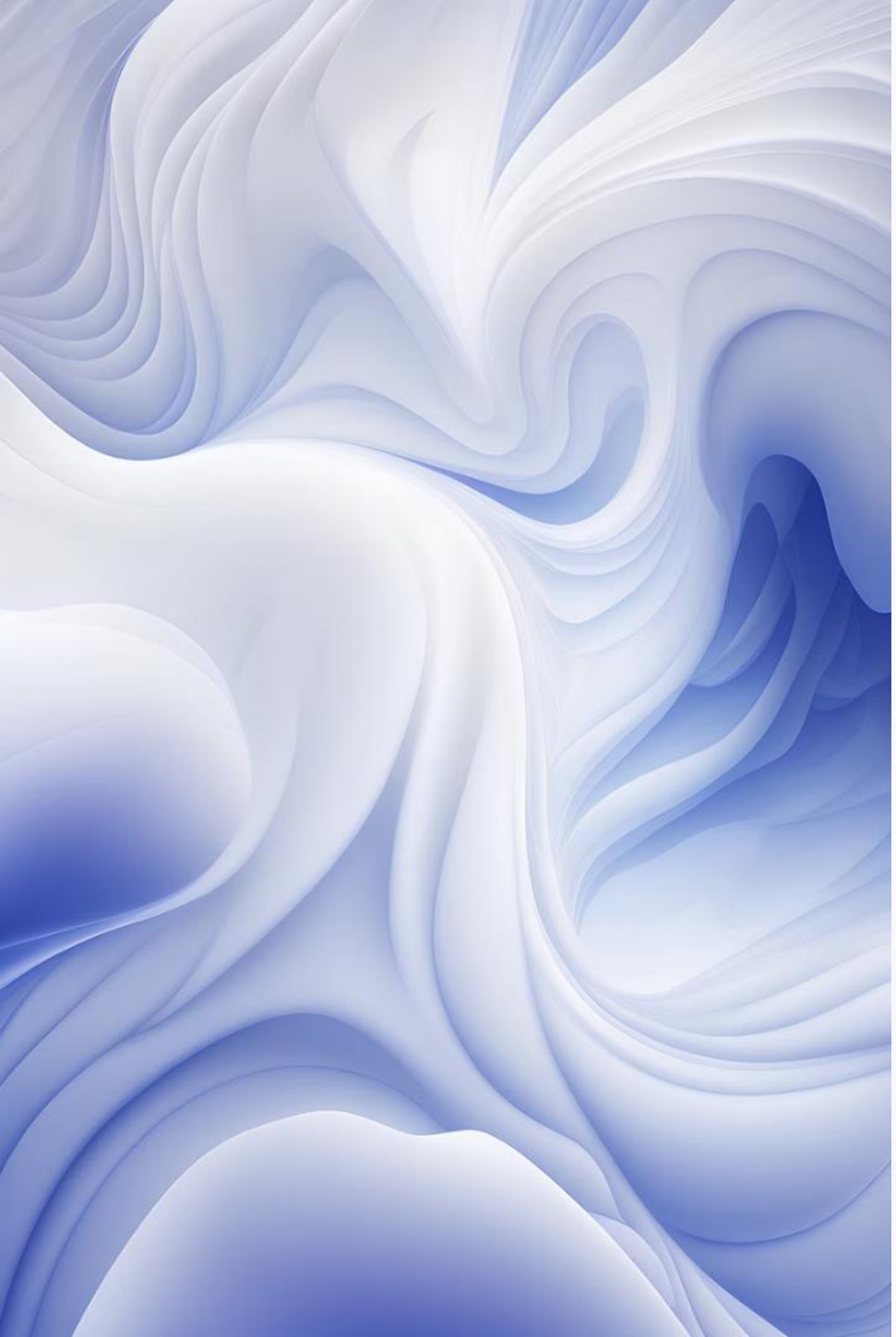
Optimising Posts Performance

Efficient post display management is essential for platform scalability. Pagination can reduce initial page load times, while lazy loading of images and videos can accelerate content rendering. Caching popular posts helps significantly decrease repeated database hits, reducing server load.

Optimising database queries is crucial for timely post retrieval, and Content Delivery Networks (CDNs) ensure rapid distribution of media assets to users globally. Implementing search indexes, such as Elasticsearch, enhances post discovery and navigation by enabling fast and relevant search results. Additionally, distributed caches and message queues aid in handling high traffic volumes and offloading processing tasks, maintaining optimal responsiveness during peak usage periods.

Case Studies and Examples

- **Facebook:** Leveraged Memcached caching, reducing database load, enhancing performance during traffic spikes.
- **Twitter:** Utilises Kafka message queues to manage an immense volume of tweets efficiently without throughput loss.
- **LinkedIn:** Combines caching, load balancing, and database optimisation techniques to maintain responsiveness at peak usage times.
- **Slack:** Employs WebSockets and distributed message queues to facilitate real-time team communication with low latency.



Conclusion and Recommendations

Ensuring both security and performance is indispensable for the success of any social networking platform. Establishing safe, restricted HTML/CSS customisation policies alongside rigorous database security practices is foundational. Attention to real-time interaction optimisation enables smooth user experiences critical for engagement and retention.

Continual monitoring, security audits, and penetration testing are essential to proactively manage vulnerabilities. Implementing a robust incident response plan enhances preparedness against potential breaches. This comprehensive approach fosters a secure, scalable, and resilient platform capable of supporting dynamic social networking demands.

References

- **BigID:** Principle of Least Privilege Access - Why It Matters (bigid.com)
- **CURBSTONE:** What We Learned from the 2023 Verizon Data Breach Investigation Report (curbstone.com)
- **Verizon:** DBIR 2024 (verzion.com)
- **NCSC UK:** Data Protection (nscs.gov.uk)
- **NIST:** Digital Identity Guidelines (nist.gov)
- **OWASP:** CSP Cheat Sheet (cheatsheetseries.owasp.org), XSS Prevention Cheat Sheet (cheatsheetseries.owasp.org)
- **Mozilla:** The Websockets API (developer.mozilla.org)