Nama : Azriel Aprieliant

Nim : I.2310086

Studi Kasus 1

1. Masalah kinerja pada aplikasi kemungkinan disebabkan oleh overload pada database akibat query lambat, kurangnya caching untuk data yang sering diakses, dan kapasitas server yang tidak cukup untuk menangani pertumbuhan pengguna.

2. Untuk mengukur kinerja dan mengidentifikasi masalah, kita dapat menggunakan metrik seperti waktu respon (response time), jumlah permintaan yang diproses per detik (throughput), dan pemakaian sumber daya server (CPU dan memori).

3. Isolasi masalah dapat dilakukan dengan menggunakan alat monitoring seperti New Relic, melakukan pengujian beban menggunakan JMeter, dan menganalisis alur permintaan melalui tracing tools.

4. Solusi utama yang dapat diterapkan adalah menggunakan caching dengan Redis/Memcached, menambahkan load balancer untuk mendistribusikan beban, dan mengoptimalkan query database dengan indeks atau perbaikan logika query.

5. Untuk mengimplementasikan solusi tanpa mengganggu layanan, kita bisa menggunakan metode canary deployment, di mana perubahan diterapkan bertahap pada sebagian kecil pengguna, atau rolling updates untuk memperbarui server satu per satu.

6. Untuk memastikan solusi tidak menimbulkan masalah baru, lakukan pengujian menyeluruh seperti unit testing dan load testing, serta siapkan rencana rollback jika terjadi kegagalan

7. Pencegahan masalah serupa di masa depan dapat dilakukan dengan rutin melakukan stress testing untuk menguji batas beban sistem, serta mendesain aplikasi agar lebih skalabel menggunakan pendekatan microservices.

8. Untuk membangun sistem yang lebih tahan beban, kita dapat menggunakan auto-scaling untuk menambah sumber daya secara otomatis, serta menerapkan arsitektur berbasis event-driven untuk mendukung pemrosesan asynchronous.

Studi Kasus 2

1. Jenis serangan yang mungkin menjadi penyebab kebocoran data meliputi SQL Injection yang memungkinkan penyerang mengakses data sensitif, Man-in-the-Middle (MITM) yang mencuri data selama transmisi, Cross-Site Scripting (XSS) yang memanfaatkan celah pada input pengguna, serta phishing yang mengecoh pengguna untuk memberikan informasi rahasia.

2. Kebocoran data dapat berdampak serius, seperti hilangnya kepercayaan pengguna, potensi penyalahgunaan data untuk penipuan atau pencurian identitas, serta kerugian finansial dan hukum bagi perusahaan akibat denda regulasi dan tuntutan hukum.

3. Untuk mencegah kejadian seperti ini, langkah-langkah keamanan yang harus diterapkan mencakup enkripsi data secara menyeluruh, penggunaan firewall dan IDS/IPS, pengamanan API, validasi input, serta autentikasi multi-faktor (MFA) untuk melindungi akses pengguna.

4. Tindakan segera yang perlu dilakukan mencakup menutup celah keamanan yang ditemukan, memberlakukan reset kata sandi pengguna, melakukan audit sistem secara menyeluruh, dan meningkatkan pengawasan terhadap aktivitas mencurigakan di sistem.

5. Sistem perlu diperbarui dengan pengamanan yang lebih ketat, seperti penerapan TLS untuk komunikasi, pengamanan database dengan hashing pada data sensitif, dan penggunaan mekanisme rate-limiting untuk mencegah brute-force attack.

6. Untuk meningkatkan keamanan secara keseluruhan, perusahaan harus mengadopsi pendekatan keamanan berbasis lapisan (defense in depth), mengadakan pelatihan keamanan untuk tim, melakukan pentesting secara berkala, dan selalu mengikuti perkembangan teknologi keamanan terkini.