BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi, pengujian, dan analisis statistik terhadap sistem pengisian kuesioner pada SIAKAD yang telah dilakukan dengan menggunakan pendekatan message broker (Apache Kafka) dan caching (Redis), maka dapat disimpulkan hal-hal sebagai berikut:

1. Sistem pengisian kuesioner berhasil dikembangkan dengan menerapkan arsitektur terdistribusi yang mengintegrasikan komponen message broker dan caching, sebagaimana ditunjukkan pada diagram rancangan sistem. Pendekatan ini dirancang untuk meningkatkan skalabilitas, keandalan, dan efisiensi pengolahan data dalam sistem.
2. Implementasi sistem terintegrasi dilakukan dengan melakukan konfigurasi Kafka sebagai message broker untuk memisahkan beban pemrosesan data secara *asynchronous*, serta Redis sebagai caching layer untuk mempercepat pengambilan data. Konfigurasi ini diterapkan pada beberapa endpoint API utama dalam sistem pengisian kuesioner.
3. Integrasi message broker dan caching terbukti meningkatkan performa sistem, khususnya pada lingkungan server UPA TIK yang merepresentasikan kondisi lingkungan *production.* Peningkatan ini dapat dilihat dari perubahan waktu respons, *throughput*, dan *error rate*:
   - Penurunan Waktu Respons (*Response Time*):
     a. Di lingkungan local development, terjadi penurunan waktu respons yang drastis pada endpoint GET /notifications (dari 51-59 ms menjadi 3 ms, dengan p-value < 0.001) dan GET /questionsmeeting (dari 3-4 ms menjadi 2-3 ms, dengan p-value 0.038). Endpoint POST /attendances juga menunjukkan penurunan pada beban tinggi (dari 1486 ms menjadi 14 ms) meskipun secara statistik tidak signifikan (p-value 0.192). Endpoint POST /answers mengalami sedikit peningkatan waktu respons (dari 7-9 ms menjadi 10-12 ms), namun tetap dalam kategori optimal dan secara statistik signifikan (p-value 0.009).
     b. Di lingkungan server UPA TIK, endpoint POST /attendances menunjukkan penurunan waktu respons rata-rata yang sangat drastis (dari 17585 ms menjadi 170 ms pada 5000 request), mengindikasikan efektivitas message broker dalam menangani pemrosesan *asynchronous*. Demikian juga, GET /questionsmeeting menunjukkan perbaikan signifikan (dari 4913 ms menjadi 61 ms pada 5000 request, dengan p-value 0.043), menandakan efektivitas caching untuk data statis. Meskipun secara statistik POST /attendances dan POST /answers tidak menunjukkan perbedaan

signifikan pada data yang diuji, nilai rata-rata waktu respons menunjukkan adanya penurunan waktu respons. Namun, endpoint GET /notifications menunjukkan peningkatan waktu respons yang setelah integrasi (p-value 0.673), mengindikasikan adanya isu kompleksitas data.

- Peningkatan Stabilitas dan Penurunan *Error Rate*
  a. Di lingkungan local developmentx, tingkat kesalahan konsisten 0% di semua endpoint baik sebelum maupun sesudah integrasi.
  b. Di lingkungan server UPA TIK, integrasi ini berhasil menurunkan error rate pada endpoint POST /attendances dari 23.02% menjadi 0% pada beban tertinggi, serta menghilangkan error yang sebelumnya muncul pada GET /notifications dan GET /questionsmeeting. Meskipun secara statistik error rate pada endpoint-endpoint ini tidak menunjukkan perbedaan signifikan pada data yang diuji secara statistik, penurunan error rate hingga 0% ini secara praktis menandakan peningkatan keandalan sistem yang dalam menangani beban tinggi.
- Peningkatan *Throughput* (Kapasitas Pemrosesan Sistem)
  a. Di lingkungan local development, throughput stabil dan meningkat linier pada semua endpoint baik sebelum maupun sesudah integrasi, dan secara statistik tidak ada perbedaan signifikan.
  b. Di lingkungan server UPA TIK, throughput meningkat secara signifikan pada endpoint POST /attendances (dari 26.7 menjadi 83.2 permintaan per detik pada 5000 permintaan) dan GET /questionsmeeting (dari 62.8 menjadi 83.1 permintaan per detik pada 5000 permintaan, dengan p-value 0.043 untuk GET /questionsmeeting). Ini menunjukkan kemampuan sistem untuk memproses lebih banyak permintaan per detik setelah optimalisasi. Throughput POST /answers dan GET /notifications relatif stabil, tanpa perubahan secara signifikan secara statistik.

Secara keseluruhan, penerapan arsitektur terdistribusi dengan message broker dan caching ini efektif dalam mendukung kinerja sistem yang lebih responsif dan andal, terutama dalam skenario beban tinggi di lingkungan server produksi, meskipun ada beberapa *endpoint* yang memerlukan optimalisasi lebih lanjut.

5.2 Saran

Untuk penelitian dan pengembangan selanjutnya, berikut beberapa saran yang dapat dipertimbangkan:

1. Lakukan analisis cost-benefit terkait investasi dalam infrastruktur message broker dan caching. Ini akan memberikan gambaran mengenai efisiensi

biaya yang dicapai versus peningkatan kinerja, membantu dalam pengambilan keputusan investasi teknologi di masa mendatang.

2. Lakukan pengujian terkait metrik performa server seperti penggunaan CPU, memori, atau I/O disk, yang tidak tercakup dalam penelitian ini. Hal ini akan memberikan gambaran mengenai dampak implementasi terhadap sumber daya infrastruktur.
3. Mengoptimalkan endpoint yang mengalami penurunan performa karena banyaknya data yang dikembalikan, seperti pada endpoint GET /notifications di server UPA TIK, dengan menerapkan mekanisme *pagination* dan *filtering*.
4. Pertimbangkan untuk mengembangkan fitur-fitur lain di luar konteks peningkatan performa dan skalabilitas, seperti integrasi dengan modul SIAKAD lain (misalnya, sistem akademik, keuangan). Ini dapat memberikan nilai tambah yang lebih besar pada sistem kuesioner.