BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut.

1. Penerapan server web berbasis Laravel Octane terbukti secara signifikan meningkatkan performa aplikasi web, ditandai dengan peningkatan RPS *(Request Per Second)*, penurunan R*esponse Time*, dan peningkatan T*hroughput*. Namun, peningkatan performa ini diikuti dengan kenaikan *CPU Utilization* dan M*emory Usage* karena sistem *caching* dan *worker* yang di implementasikan oleh Laravel Octane menyimpan proses PHP dalam memori.
2. Dari segi arsitektur aplikasi, arsitektur modular dengan protokol gRPC menunjukkan unjuk kerja terbaik dengan RPS tinggi, *Response Time* rendah, serta *CPU Utilization* dan *Memory Usage* yang efisien. Keunggulan ini disebabkan oleh penggunaan *binary* dengan *protocol buffer* dalam pertukaran data, serta dukungan bahasa Go sebagai *gRPC Server* yang memiliki kinerja lebih cepat dibandingkan PHP.
3. Arsitektur monolitik menunjukkan performa paling rendah karena proses *rendering* file PHP dan HTML dilakukan dalam satu basis kode yang sama, berbeda dengan arsitektur modular yang memanfaatkan JavaScript untuk *rendering frontend* secara terpisah.

4. Dari perspektif server web, Swoole menunjukkan performa terbaik berkat arsitektur berbasis *event-loop* yang meminimalisir *blocking* I/O dan dibangun dengan bahasa C++ yang lebih dekat dengan bahasa mesin. RoadRunner menempati posisi kedua dengan fitur *asynchronous* berbasis bahasa Go, sementara FrankenPHP di posisi ketiga dengan *throughput* tinggi namun memiliki *error rate* yang mengindikasikan kurangnya stabilitas transfer data karena belum mendukung fitur *asynchronous*.
5. Server web konvensional seperti Nginx dan Apache menunjukkan performa terendah, dengan Nginx masih unggul dibandingkan Apache karena menggunakan modul php-fpm untuk manajemen proses PHP yang lebih efisien.

5.2 Saran

Berdasarkan penelitian dan analisis yang telah dilakukan, berikut adalah beberapa saran yang dapat dilakukan pada penelitian selanjutnya.

1. Melakukan pengujian dengan skenario yang lebih mendekati kondisi nyata khususnya terkait dengan *load test* yang lebih tinggi.
2. Untuk memudahkan dalam proses pengujian, dapat dibuat otomasi seperti membuat *script* khusus, menggunakan docker, atau menggunakan aplikasi otomasi lainnya.
3. Melakukan eksplorasi terhadap arsitektur *microservices* ataupun *microfrontend*, karena saat ini sudah banyak diterapkan dalam pengembangan aplikasi skala besar di industri.