BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis hasil penelitian dan pengembangan yang telah dilakukan mengenai pengaruh *code splitting* dan *server-side rendering* pada aplikasi *website* berbasis Vue.js, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Implementasi *code splitting* pada kasus aplikasi *point of sale* di penelitian ini terbukti menjadi strategi yang paling efektif untuk mengoptimalkan kecepatan muat awal. Teknik ini secara efektif mengurangi ukuran unduhan rata-rata dari 1.445 kB menjadi 854,4 kB, yang secara langsung berdampak dengan kecepatan pemuatan visual seperti *First Contentful Paint* (FCP) rata-rata dari 1,05 detik menjadi 0,67 detik.
2. *Server-Side Rendering* (SSR) pada kasus aplikasi *point of sale* di penelitian ini menunjukkan keunggulan pada dua aspek, yaitu SEO dan interaktif pengguna. Dari sisi SEO, SSR berhasil meningkatkan peringkat rata-rata di Google dari 1,7 menjadi 1,3 dan menaikkan rasio klik-tayang (CTR) dari 35,3% menjadi 64%. SSR secara drastis menurunkan *Total Blocking Time* (TBT) rata-rata dari 33,75 ms menjadi hanya 3,75 ms, menjadikan halaman terasa lebih cepat responsif.
3. Secara fungsional, aplikasi POS telah lolos uji *black box* dan bekerja sesuai harapan. Hasil *load testing* menunjukkan *backend* cukup kuat menghadapi beban tinggi, meski proses berat seperti laporan dan ekspor data masih membutuhkan optimasi karena waktu respons yang tinggi.
4. Penelitian ini menyimpulkan adanya pertukaran antara kedua arsitektur, CSR dengan *code splitting* unggul dalam menyajikan konten visual pertama dengan FCP/LCP tercepat. Sementara itu, SSR mengorbankan sedikit kecepatan *render* awal demi keunggulan pada SEO dan interaktif (TBT).

5.2 Saran

Berdasarkan kesimpulan yang telah diuraikan, penulis memberikan beberapa saran untuk penelitian yang lebih lanjut.

1. Penelitian ini berfokus pada Vue.js. Peneliti selanjutnya disarankan untuk melakukan perbandingan performa SSR dan *code splitting* pada *framework* lain seperti React, Next.js, atau Nuxt.js untuk mendapatkan pemahaman yang lebih luas terhadap efisiensi teknik tersebut di berbagai ekosistem.
2. Melakukan analisis teknik optimasi lainnya seperti *threeshaking, minification, compression* dan teknik optimasi lainnya.
3. Optimalisasi pada *backend* dengan menerapkan strategi *caching* untuk data yang sering diakses dan optimalisasi *query* pada basis data.