BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, kesimpulan yang diambil di antaranya sebagai berikut:

a. Dalam analisis *Static Site Generation* (SSG), penulis membuat 2 *website* serupa dan mengembangkan salah satunya dengan menggunakan SSG. Penulis juga membuat *REST API* untuk menggunakan data yang disimpan pada *database*, *API* diuji dengan menggunakan *POSTMAN*. Setelah pengembangan, *website* diuji dengan menggunakan metode *Black Box Testing* dimana *website* berhasil melalui seluruh skenario pengujian.
b. SSG berhasil diimplementasikan, dimana penulis menggunakan 2 fungsi SSG dari *Next.js*, yaitu *getStaticPaths* dan *getStaticProps*. Langkah ini menghasilkan rute statis dari rute dinamis serta objek statis yang di-*fetching* menggunakan API pada waktu pembangunan. Meskipun data sudah ditarik pada waktu pembangunan, *Website* tetap dapat memperbarui data dengan melakukan *fetching API* pada kondisi tertentu Setelah pengembangan, performa kedua website diuji dengan *Google Lighthouse*. Hasil pengujian menampilkan pada 4 dari 6 item uji, terjadi peningkatan performa *website* dengan SSG dibandingkan dengan *website* awal. Skor performa meningkat dari 41 menjadi 99, LCP dari 30,8 detik menjadi 0,9 detik, TBT yang sebelumnya 1.730 milidetik menjadi 20 milidetik, dan *Speed Index* meningkat dari 2 detik menjadi 0,7 detik. Hasil tersebut menunjukkan peningkatan performa dari implementasi SSG pada *website* dinamis.
c. Banyak cara untuk memanfaatkan SSG, umumnya SSG hanya digunakan untuk membuat objek dan rute statis. Pada penelitian ini, fungsi *getStaticProps* dielevasi dengan penggunaan bersama dengan fungsi untuk mengonversi gambar yang sebelumnya disimpan *database* dengan tipe data base64 menjadi jpeg/png. Pada fase pembangunan, objek statis kemudian disimpan dalam bentuk jpeg/png. Hal ini bertujuan untuk meningkatkan

performa di sisi *TotalBlockingTime*, ini adalah salah satu penggunaan SSG secara spesifik dengan terfokus pada objek gambar yang merupakan konten terberat pada *website* ini. Fungsi ini tidak efektif digunakan pada website tanpa SSG, karena waktu rendering justru akan meningkat jika mengolah objek terlebih dahulu setiap kali melakukan penarikan data. Algoritma berbeda juga dapat digunakan pengembang dengan kasus sesuai letak beban *website* sebagai salah satu solusi meningkatkan performa agar nyaman digunakan pengguna, misalnya sisi pengembang memiliki keterbatasan akses untuk mengubah *store procedure* ataupun *view* yang tidak efektif dari tender proyek sehingga membutuhkan pengolahan data di sisi *client* sebelum diakses pengguna akhir. Penelitian ini diharapkan dapat menjadi acuan solusi terhadap kasus tersebut.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, penulis memiliki beberapa saran dalam penggunaan SSG, peningkatan performa, serta pengembangan penelitian selanjutnya:

a. Memperbanyak jumlah buku/data yang digunakan untuk mendapat hasil yang lebih signifikan.
b. Menggunakan data yang lebih besar seperti video atau gambar dengan resolusi tinggi
c. Dalam penggunaan SSG perlu memerhatikan waktu *rendering* terutama pada penggunaan *React Hook.*
d. Menggunakan *page router* pada saat pembuatan proyek *Next.js,* karena fungsi SSG *getStaticPaths*, *getStaticProps*, dan lainnya tidak ada pada *app router.*
e. Menggunakan SSG pada rute yang membutuhkan banyak data, konten berat, dan pengolahan data yang membutuhkan waktu lama baik di *frontend* maupun *backend.*
f. Menyiapkan penyimpanan yang cukup. Hal ini dikarenakan kelemahan SSG adalah membutuhkan lebih banyak *resource* pada penyimpanan server.

Pada penelitian ini ukuran *file* tanpa menggunakan SSG sebesar 366MB *on disk,* sedangkan setelah menggunakan SSG sebesar 547MB *on disk*. Ukuran dari penggunaan SSG akan trus meningkat seiring dengan meningkatnya jumlah data.