BAB II

TINJAUAN PUSTAKA

2.1 Optimasi WebApp

Optimasi merupakan suatu proses memaksimalkan atau meningkatkan ketercapaian dari tujuan yang diharapkan sesuai dengan kriteria yang telah ditetapkan. Optimasi merupakan proses untuk mengoptimalkan suatu solusi agar ditemukannya solusi terbaik dari sekumpulan alternatif solusi yang ada. Optimasi dalam *website* yaitu belajar bagaimana mengolah konten, meningkatkan performa sebuah situs *website* (Fadli, 2019).

Kecepatan akses *website* ditandai dengan durasi loading time yang cepat dan kehandalan web server memiliki pengaruh signifikan dalam memberikan kenyamanan bagi user ketika mengakses konten *website*. Kestabilan akses web sangat bergantung pada *performance* web server serta besar ukuran halaman *website* (Kurniawan & Widiyanto, 2016). Hasil yang didapati dari optimasi yang dilakukan pada penelitian sebelumnya adalah mengurangi ukuran berkas, mengurangi request file, mengurangi *bandwidth*, dan mengurangi waktu muat halaman (Amrullah, 2021).

2.1.1 Teknik Optimasi WebApp

Untuk melakukan optimasi WebApp agar lebih cepat dalam mengakses dapat dilakukan dengan cara konfigurasi server tersebut meliputi pembaruan *library* yang ada pada server dan aplikasi meliputi penggunaan *Image Compression*, *Browser Cache*, *Minify* CSS, HTML dan *Javascript file*, *Anti Render-Blocking Javascript*, serta penempatan *stylesheet* CSS dan Javascript, sehingga mampu meningkatkan

kecepatan dan kestabilan akses *website* (Kurniawan & Widiyanto, 2016). Optimasi dilakukan pada sisi *front end* relatif lebih murah dan mudah jika dibanding dengan optimasi pada sisi server ataupun jaringan internet. Metodenya adalah dengan meminimalkan jumlah HTTP *request*, meminimalkan ukuran file dan membuat halaman *cacheable* (Arumoadi, Wawan Laksito Y.S., & Susyanto, 2019).

Kompleksitas *query* dapat mempengaruhi pada penambahan beban kinerja CPU, *bandwidth* dan memori, untuk meminimalisir beban tersebut dapat dilakukan perbaikan pada *pagination* dan *query* (Kartika & Rinartha, 2020). Dari kutipan tersebut dapat disimpulkan optimasi website dapat dilakukan dengan penambahan *pagination* pada data dengan jumlah baris yang banyak serta merampingkan *query*. Selain itu pengaruh ukuran gambar juga memberikan dampak pada kecepatan aplikasi. Dikutip dari halaman GTmetrix, gambar yang besar dan tidak fleksibel menyebabkan penurunan pada performa aplikasi.

2.2 *Stress Tools*

Stress Tool adalah sebuah aplikasi ampuh untuk menguji HTTP-client/server yang dirancang untuk menentukan kinerja *website* ketika sedang mengalami masalah kritis bila sebuah *website* atau pun *server* milik *website* tersebut yang sedang mengalami lonjakan pengunjung (Delta & Asmunin, 2016). *Stress tools* memiliki banyak jenis diantaranya yaitu Loadview, Grinder dan GTmetrix. *Stress tools* tersebut memiliki kekurangan dan kelebihan. Setelah dilakukan perbandingan untuk memilih *stress tools* sesuai kebutuhan penelitian optimasi didapatkan beberapa kekurangan pada *stress tools* Loadview yaitu adanya batasan waktu mencoba pengetesan website (*trial*) sehingga untuk mengetahui teknik optimasi yang harus dilakukan harus berlangganan dengan harga yang cukup mahal

($199/bulan). Dengan harga tersebut LoadView memberikan informasi berkala pada aplikasi yang dites, namun dokumentasi yang ada merujuk pada peningkatan server (*hardware*) sedangkan pada penelitian difokuskan pada peningkatan aplikasi dengan server yang ada.

Pada *stress tools* Grinder didapatkan kekurangan pada saat melakukan optimasi memiliki keterbatasan pada dokumentasi, sehingga optimasi website belum tepat sasaran. Selain itu Grinder merupakan aplikasi desktop *open source* namun untuk melakukan optimasi yang lebih kompleks harus memiliki paket berbayar. Sedangkan pada GTmetrix terdapat kekurangan dalam memberikan informasi pada permasalahan waktu respons awal server. Pada dokumentasi tidak diberikan penjelasan kriteria suatu aplikasi agar dapat dilakukan pengetesan. Sehingga ketika didapatkan hasil analisis eror harus melakukan analisis diluar dokumentasi resmi GTmetrix. Namun secara keseluruhan GTmetrix dapat memberikan hasil pengetesan performa dengan kompleks meskipun tanpa berlangganan. Perbedaan fitur berlangganan tidak terlalu berpengaruh seperti pada Loadview karena GTmetrix hanya memberikan keterbatasan pemilihan lokasi dan hasil pengetesan tidak dilakukan secara *realtime*.

2.2.1 GTmetrix

Dikutip dari dokumentasi (gtmetrix.com), GTMetrix adalah sebuah *tools* gratis yang menilai kecepatan situs *website*. *Tools* GTMetrix dapat menganalisis *grade* kecepatan halaman situs *website* dan memberi saran bagaimana agar situs *website* menjadi lebih cepat. Untuk melakukan pengetesan dapat dilakukan dengan cara memasukkan link halaman yang akan dilakukan pengetesan pada kolom analisis. Kemudian GTMetrix memeriksa total *loading time of page* dan total *page*

size. Hasil analisis tersebut berupa *grade* berupa angka dan diikuti persentase dari *performance* dan *structure*. Selain itu GTmetrix juga memberikan permasalahan utama yang menyebabkan aplikasi yang dites mendapatkan hasil *grade* dan prosentase. Permasalahan tersebut dipetakan dari dampak paling tinggi sampai tidak berdampak sama sekali, permasalahan tersebut yang dilakukan optimasi agar aplikasi menjadi optimal secara kecepatan. GTMetrix menyediakan struktur secara rinci dari komponen-komponen yang berkontribusi pada ukuran total situs.

Ruang lingkup permasalahan yang diteliti meliputi pengujian dan pemilihan web server yang handal dan implementasi tindakan optimasi terhadap web *template* meliputi *Image Compression*, *Browser Cache*, Minify CSS, HTML dan Javascript *file*, *Anti Render-Blocking* Javascript penempatan *stylesheet* CSS, penempatan *script* Javascript (Kurniawan & Widiyanto, 2016). Pengujian kestabilan dan kecepatan akses WebApp diukur dengan bantuan *website stress tool* GTmetrix dengan parameter pengukuran meliputi *performance* (%) dan *structure* (%).

A. *Performance*

Menurut (gtmetrix.com) grade *performance* pada dasarnya adalah *grade performance* utama aplikasi, seperti yang ditangkap oleh pengujian GTmetrix, dengan *browser*, spesifikasi perangkat keras, dan opsi analisis yang ditentukan (AdBlock, Kecepatan Koneksi, dll.). Grade ini terdiri dari 6 metrik utama dengan bobot sebagai berikut:

1. ***Loading performance* (45%)**
    a. *First Contentful Paint* (10%)
    b. *Speed Index* (10%)
    c. *Largest Contentful Paint* (25%)

2. ***Interactivity* (40%)**
    a. *Time to Interactive* (10%)
    b. *Total Blocking Time* (30%)
3. ***Visual Stability* (15%)**
    a. *Cumulative Layout Shift* (15%)

Setiap metrik diukur dan dihitung sebagai *grade*, kemudian dibandingkan dengan ambang batas, dan hasil agregat (dengan bobot yang sesuai) membentuk *perfromance grade* akhir.

B. *Structure*

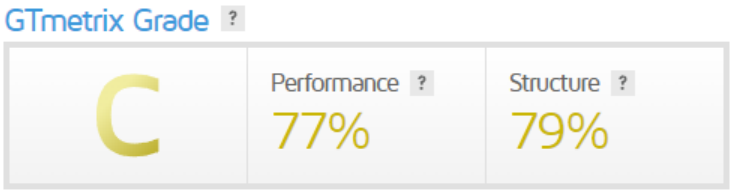
Menurut (gtmetrix.com), *grade structure* memberi tahu seberapa baik halaman yang dibuat untuk kinerja yang optimal. Gtmetrix telah menetapkan nilai poin berdasarkan berbagai faktor, termasuk potensi penghematan dan kepentingan, seperti yang dirasakan oleh tim GTmetrix. *Grade* ini seberapa baik kinerja aplikasi pada *structure grade* mencerminkan seberapa baik situs dibangun untuk kinerja yang optimal. Kriteria yang diberikan oleh GTmetrix yaitu sebagai berikut:

1. *Enable Keep-Alive*
    a. Ini hanya akan dipicu pada halaman yang menggunakan HTTP/1.1 tanpa direktif *Keep-Alive ON* yang terdeteksi.
    b. Jika halaman aplikasi terdeteksi menggunakan HTTP/2, audit ini tidak akan memengaruhi aplikasi.
2. *Combine images using* CSS *sprites*
    a. Jika halaman aplikasi menyajikan beberapa gambar kecil yang dapat diubah, audit ini akan dipicu.

b. Perhatikan bahwa audit ini memiliki ambang batas yang berbeda bergantung pada apakah laman aplikasi menggunakan HTTP/1.1 atau HTTP/2.

3. *Use a Content Delivery Network* (CDN)
4. *Avoid* CSS `@import`
   a. Menghindari CSS `@import` jika memungkinkan masih merupakan praktik yang disarankan karena setiap arahan impor yang ditemukan oleh *browser* (tanpa `defer` atribut) akan segera diunduh, diuraikan, dan dieksekusi.
   b. Ini dapat memblokir *rendering* sisa halaman aplikasi.

Dari kedua poin tersebut maka diakumulasi ke dalam *grade* sebagai *feedback* dari *testing* yang dilakukan. Grade tersebut seperti pada Gambar 2.1 berikut.



Gambar 2.1 Grade Hasil Testing GTmetrix
Sumber: https://gtmetrix.com/

Pada GTmetrix terdapat 6 jenis *grade* sebagai hasil *testing*, yaitu: A, B, C, D, E, F dengan parameter optimal adalah 70% *performance* dan 30% *structure*. Apabila mendapatkan *grade* A maka aplikasi sudah memiliki performa yang baik, tidak perlu optimasi lebih lanjut. Jika mendapat *grade* B tanpa dampak tertinggi pada permasalahan utama maka tidak diperbaiki tidak masalah karena masih tergolong optimal. Untuk *grade* C-F maka *website* perlu mendapatkan penanganan untuk optimasi agar aplikasi menjadi lebih baik.

2.3 Pengujian Alpha dan Beta

Pengujian alpha dilakukan untuk melihat apakah semua sistem dapat berjalan dengan baik dan dilakukan oleh pembuat sistem atau yang terlibat dalam pembuatan sistem sedangkan pengujian beta digunakan untuk melakukan evaluasi terhadap sistem yang telah dibuat, pihak yang akan melakukan penilaian sistem adalah pengguna atau orang-orang yang tidak terlibat dalam pembuatan sistem (Masripah & Ramayanti, 2020). Tujuan dari beta testing dapat beraneka ragam, seperti memberikan kesempatan kepada media pers untuk menuliskan tinjauan awal dari *software* untuk mendapatkan masukan mengenai *user interface* sebagai usaha terakhir untuk menghilangkan *bugs* (Tjandra & C. Pickerling, 2015).

Menurut (Tjandra & C. Pickerling, 2015) dalam melakukan proses pengujian alpha beta terdapat beberapa hal yang perlu diperhatikan seperti pada poin-poin berikut:

1. Siapa penguji yang akan dipilih. Karena sebuah beta testing dapat mempunyai bermacam-macam tujuan, maka penting untuk mengerti siapa saja para penguji yang akan dipilih tersebut.
2. Pengujian dengan beta testing ini akan juga sangat membantu pengujian dengan metode *usability testing*. Gabungan dari penguji yang berpengalaman akan meningkatkan nilai pengujian metode *usability* testing. Seorang penguji yang baru melihat *software* yang diuji untuk pertama kalinya akan dapat dengan mudah menemukan bagian yang membingungkan atau sulit untuk dioperasikan.
3. Selain *configuration*, *compability* dan *usability testing*, pengujian beta *testing* adalah cara pengujian yang baik untuk menemukan *bug*.

2.4 *Cyclomatic Complexity*

Cyclomatic Complexity merupakan metode pengujian yang dilakukan untuk memastikan kelayakan suatu aplikasi sebelum digunakan, testing ini salah satunya adalah dengan pengujian *white box* dimana pengujian *White Box Testing* adalah metode pengujian menggunakan struktur kendali dan desain prosedur, hasil pengujian ini bermanfaat untuk mengetahui sistem aplikasi setelah tahap implementasi (Setiawan, Ananda, Rayesta, Tasya, & Joko , 2022). *Cyclomatic complexity* adalah *software* yang menyediakan acuan kuantitatif kompleksitas suatu logika dalam program (Suprapto, Fauziah, Fitri, & Hayati, 2020). Untuk melakukan perhitungan *cyclomatic complexity* adalah dmenggunakan formula sebagai berikut.

$$V(G) = E - N + 2$$

Keterangan:
E = jumlah *edges* pada *flowgraph*
N = jumlah *node* pada *flowgraph*