IMPLEMENTASI ALGORITMA BOYER MOORE UNTUK PERNYARINGAN *EMAIL*

Skripsi

diajukan sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Pendidikan Program Studi Pendidikan Teknik Informatika dan Komputer

Oleh:

Wahyu Prakoso

NIM. 5302415028

PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SEMARANG

2020

PERSETUJUAN PEMBIMBING

Nama	: Wahyu Prakoso
NIM	: 5302415028
Program Studi	: S-1 Pendidikan Teknik Informatika dan Komputer
Judul Skripsi	: Implementasi Algoritma Boyer Moore Untuk Penyaringan *Email Spam*

Proposal skripsi ini telah disetujui oleh pembimbing untuk diajukan ke panitia ujian skripsi Program Studi S-1 Pendidikan Teknik Informatika dan Komputer Fakultas Teknik Universitas Negeri Semarang.

Semarang, 20 Mei 2020

Dosen Pembimbing,

Arief Arfriandi, S.T., M.Eng.
NIP.198208242014041001

PENGESAHAN

Skripsi dengan judul "Implementasi Algoritma Boyer Moore untuk Pernyaringan *Email*" telah dipertahankan di hadapan sidang Panita Ujian Skripsi Jurusan Teknik Elektro Fakultas Teknik UNNES pada tanggal 17 Juni 2020

Oleh

Nama	Wahyu Prakoso
NIM	5302415028
Program Studi	Pendidikan Teknik Informatika dan Komputer

Panitia:

Ketua Panitia

Ir. Ulfah Mediaty Arief, M.T., IPM
NIP. 196605051997022001

Sekretaris

Budi Sunarko, S.T., M.T., Ph.D.
NIP. 197101042006041001

Penguji I

Dr. Ing. Dhidik P., S.T., M.T.
NIP.197805312005011002

Penguji II

Arryo B. U., S.T., M.T.
NIP. 198409092012121002

Penguji III/Pembimbing

11/09/2020

Arief A., S.T., M.Eng.
NIP.198208242014041001

Mengetahui
Dekan Fakultas Teknik UNNES

Dr. Nur Qudus, M.T., IPM
NIP. 196911301994031001

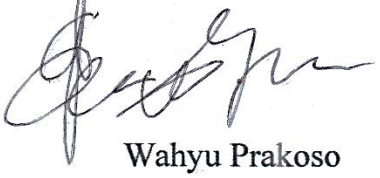
PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa:

1. Skripsi/TA ini, adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (sarjana, magister, dan/atau doktor), baik di Universitas Negeri Semarang (UNNES) maupun di perguruan tinggi lain.
2. Karya tulis ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan Pembimbing dan masukan Tim Penguji.
3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya ini, serta sanksi lainnya sesuai dengan norma yang berlaku di perguruan tinggi ini.

Semarang, 20 April 2020

Yang membuat pernyataan,



Wahyu Prakoso

NIM. 5302415028

MOTTO DAN PERSEMBAHAN

Motto:

"Bahagia"

"Apapun yang terjadi adalah yang terbaik"

"*Never stop learning because life never stop teaching*"

"Dalam hidup ini harus mempunyai tujuan hidup agar bisa terus melangkah dan tidak kehilangan arah"

"Hadapi! Jangan melarikan diri! Hidup itu melangkah ke depan bukan melangkah ke belakang. Manfaatkanlah waktu dengan baik! Waktu dapat digunakan untuk menghasilkan uang tapi uang tidak bisa membeli waktu dan tidak bisa membeli kesempatan kedua"

"Jangan mau terpedaya oleh rayuan menunda-nunda dan berangan-angan panjang sebab setiap detik umur yang terlewatkan dari umur tidak akan tergantikan"

(Hadratussyaikh KH. M. Hasyim Asy'ari)

Persembahan:

1. Bapak dan ibu saya tercinta yang senantiasa mendoakan dan memberikan dukungan kepada saya. Mohon maaf untuk selama ini, saya harap masih mempunyai sisa waktu untuk membahagiakan dan berbakti kepada kalian.
2. Teman-teman seperjuangan Pendidikan Teknik Informatika dan Komputer angkatan 2015.
3. Orang-orang yang hadir di hidup saya yang telah memberikan pelajaran tentang kehidupan yang sangat bermakna.

ABSTRAK

Wahyu Prakoso. 2020. Implementasi Algoritma Boyer Moore untuk Penyaringan *Email*. Skripsi. Pendidikan Teknik Informatika dan Komputer Jurusan Teknik Elektro. Fakultas Teknik Universitas Negeri Semarang.

Pembimbing: Arief Arfriandi, S.T., M.Eng.

Email merupakan media komunikasi di internet, sudah banyak *provider email* yang ada, setiap *provider* memiliki sistem penyaringan *email* tersendiri, namun sistem penyaringan *email* yang dimiliki *provider* masih terdapat kesalahan dalam mengkategorikan *email* sebagai *spam*, serta tidak terdapat pemberitahuan kepada pengguna sehingga dapat dimungkinkan *email* penting terlewat untuk dibaca. Tujuan dari penelitian ini adalah merancang dan membangun sistem penyaringan *email* dan mengimplementasikan algoritma Boyer Moore dengan pengembangan menggunakan *framework* Laravel.

Dalam penelitian ini menggunakan metode *waterfall* yang memiliki lima tahapan yaitu *communication*, *planning*, *modeling*, *construction*, dan *deployment*. Pengujian yang dilakukan diantaranya pengujian performa algoritma, *white box*, dan *black box*. Untuk pengujian *black box* aspek yang diuji adalah *functionality*, *efficiency*, *portability*, dan *usability*. Pengujian menggunakan data *email* hasil dari sinkronisasi melalui *Internet Message Access Protocol* (IMAP) ke *server Email Service Provider* (ESP) yang digunakan pengguna.

Hasil penelitian performa algoritma Boyer Moore pada sistem penyaringan *email* dapat dikategorikan cepat dikarenakan proses pencocokan teks dan *pattern* dalam waktu kurang dari satu detik yaitu rata-rata 20,85 μs. Performa algoritma Boyer Moore dapat maksimal atau lebih cepat dalam proses pencocokan jika teks dan *pattern* memiliki jumlah karakter unik yang banyak pada teks atau *pattern* yang panjang. Hasil pengujian *white box* diperoleh nilai *cyclometic complexity* yaitu 5 yang berarti algoritma Boyer Moore merupakan algoritma yang sederhana dan memiliki resiko rendah. Pengujian *black box* memperhatikan beberapa aspek yaitu *functionality*, *portability*, *efficiency*, dan *usability*. Pada aspek *functionality* memperoleh hasil 100% yang berarti semua fungsi berjalan dan berfungsi dengan baik. Pada aspek *portability* memperoleh hasil 100% yang berarti sistem dapat di akses dari berbagai macam *browser* dan *device*. Pada aspek *efficiency* sistem sudah baik karena memperoleh rata-rata *grade* PageSpeed A (93%) dan Yslow B (84%), pada rata-rata *page size* 362 KB dengan *load time* rata-rata 3,2 detik. Pada aspek *usability* memperoleh hasil skor SUS yaitu 77,84. Jika berdasarkan cara penilaian SUS maka pada aspek *usability* sistem termasuk pada kategori *acceptable*, memperoleh *grade* C, dan *adjective rating* pada kategori *excellent*.

Kata Kunci: Penyaringan *Email*, Algoritma Boyer Moore, Pencocokan *Pattern*, *Framework* Laravel.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Menurut perusahaan manajemen hubungan pelanggan EveryAction, rata-rata 7 persen dari *email* nirlaba dikirim ke kotak *spam*, bukan ke *inbox*. Ada banyak hal yang berbeda dalam menentukan *spam* dan sulit untuk melacak penyebabnya sehingga sebagian besar organisasi nirlaba tidak melihat bagaimana hal itu memengaruhi mereka atau apa yang dapat mereka lakukan (Nonprofit Business Advisor, 2016: 1). *Email* dikirimkan berdasarkan reputasi IP dari mana *email* dikirim. Jika *Internet Service Provider* (ISP) mengetahui bahwa *email* yang dikirim ditandai sebagai *spam* maka sistem merutekan *email* ke *folder spam*, atau lebih buruk lagi sepenuhnya memblokir pengirim (Nonprofit Business Advisor, 2016: 2).

Banyak *provider email* yang tersedia, setiap *provider email* mempunyai sistem penyaringan *email* yang berbeda-beda. Pada sistem panyaringan *email* yang *provider* terapkan dimungkinkan masih terdapat kesalahan dalam mengkategorikan *email spam*, hal tersebut pernah terjadi pada *provider email* Zoho terhadap *provider email* Outlook pada tahun 2018. *Email* yang terkirim dari Zoho dikategorikan sebagai *spam* oleh sistem penyaringan *email* Outlook meskipun pengguna Outlook tidak memasukan *provider* Zoho ke dalam *blacklist* di pengaturan penyaringan *email*. Pihak Zoho mengidentifikasi permasalahan tersebut apakah permasalahan tersebut berasal dari sistem Zoho atau dari Outlook. Permasalahan yang terjadi disebabkan oleh *Exchange Online Protection* (EOP) *server* Outlook yang salah

mengidentifikasi *email* yang dikirim dari Zoho sebagai *spam* karena EOP Outlook telah memblokir rentang IP milik Zoho 135.84.80.192/26 dan 136.143.188.0/23 (Ramya, 2019).

Alamat *email* terdiri dari dua bagian yang dipisahkan tanda *axon* (@). Bagian kiri dari tanda @ disebut nama pengguna yang menunjukan identitas pemilik *email*. Bagian sebelah kanan dari tanda @ disebut nama *domain* yang menunjukan identitas *mail server*. Nama *domain* dapat dirubah sesuai kebutuhan, biasanya menyesuaikan nama perusahaan atau organisasi. Sebuah nama *domain* memiliki bagian yang disebut *Top Level Domain* (TLD). Terdapat dua jenis TLD yaitu *Generic Top Level Domain* (gTLD) dan *Country Code Top Level Domain* (ccTLD), gTLD digunakan untuk berbagai keperluan umum sedangkan ccTLD digunakan khusus untuk negara tertentu (Catherine R. Easton, 2012: 275).

Berdasarkan program gTLD baru pada tahun 2012 *Internet Corporation for Assigned Names and Numbers* (ICANN) meluncurkan putaran pertama secara publik sehingga pemohon dapat mengajukan permohonan untuk nama gTLD baru di Net, program tersebut bertujuan untuk meningkatkan persaingan di pasar nama *domain* dan memperluas pilihan konsumen (Simone Vezzani, 2014: 307). Namun penggunaan gTLD yang baru diluncurkan belum mempunyai reputasi atau gTLD bereputasi buruk pada alamat *email* sehingga memiliki permasalahan yaitu *email* yang dikirim akan dianggap sebagai *spam* oleh sistem penyaringan *email* yang dimiliki *provider email*, sehingga *email* yang yang dianggap *spam* akan masuk ke *folder spam*.

Email yang masuk ke *folder spam* tidak akan memunculkan pemberitahuan kepada pengguna sehingga *email* akan terlewatkan untuk dibaca. Di *folder spam* berisi kumpulan *email* yang dikategorikan *spam* yang merupakan hasil dari sistem penyaringan *email provider* dan pengaturan penyaringan *email* oleh pengguna. *Email* di *folder spam* tidak terdapat sebuah tanda yang membedakan *email spam* hasil dari sistem penyaringan *email* yang dimiliki *provider* atau dari pengaturan pengguna. Hal tersebut dapat menyulitkan pengguna dalam memilah *email* yang ada di *folder spam*.

Shukor Bin Abd Razak dan Ahmad Fahrulrazie Bin Mohamad (2013: 347) menyatakan bahwa identifikasi atau penyaringan *email spam* dapat dilakukan berdasarkan informasi yang diperoleh dari *header email*. Hal tersebut menjadi latar belakang penulis ingin melakukan penelitian penyaringan *email spam* berdasarkan informasi dari *header email*.

Informasi atau data yang diperoleh pada *header email* berupa *string* maka penyaringan *email spam* akan menerapkan teknik *matching string*. Ada berbagai macam algoritma yang dapat diterapkan dalam teknik *matching string*, seperti algoritma Rabin Karp, Boyer Moore, Knuth-Morris-Pratt (KMP) dan Aho Corasick.

Sudah banyak penelitian yang meneliti kinerja berbagai algoritma *matching string* yang ada. Parenrengi, *et al*., (2017) menganalisis perbandingan antara algoritma Boyer Moore dan algortima Knuth Morris Pratt, untuk menentukan algoritma yang paling baik digunakan dalam aplikasi Tripelka *Foodshop* Kendari. Parameter yang digunakan untuk membandingkan kedua algoritma tersebut adalah

waktu pencarian dan tingkat keakurasian data yang ditampilkan. Hasil dari penelitian tersebut menunjukkan bahwa algoritma Boyer Moore dan algoritma Knuth Morris Pratt memiliki tingkat keakurasian yang sama tetapi algoritma Boyer Moore adalah algoritma yang waktu pencariannya lebih cepat dibandingkan algoritma Knuth Morris Pratt.

Pranit Chettri dan Chinmoy Kar (2016: 26) meneliti berbagai algoritma pencarian teks yaitu algoritma Knuth-Morris Pratt, algoritma Pencarian Naïve dan algoritma Boyer Moore dengan menyediakan *input* teks berbagai ukuran dan menganalisis perilaku mereka pada *input* variabel. Setelah menganalisis dan melakukan studi pada algoritma, hasilnya menyatakan bahwa algoritma Boyer Moore bekerja dengan sangat baik dan efisien daripada yang lainnya ketika berhadapan dengan set data yang lebih besar.

Srinivas, *et al.*, (2018), Algoritma Knuth Morris Pratt bergerak maju hanya dalam urutan *input* dan membutuhkan ruang tambahan. Algoritma Rabin Karp digunakan untuk mengidentifikasi *plagiarisme* dan membutuhkan ruang tambahan untuk pencocokan. Algoritma Boyer Moore sangat cepat dalam urutan yang luas, Boyer Moore menghindari banyak perbandingan yang tidak berguna dengan pola pencarian yang secara mendasar berkaitan dengan konten. Kompleksitas *running case* terbaiknya adalah sub linear. Hasil menunjukkan bahwa Boyer-Moore adalah algoritma terbaik untuk mengatasi masalah pencocokan *string* dalam kasus-kasus biasa, dan Rabin-Karp adalah pilihan yang layak untuk beberapa kasus tertentu, misalnya ketika pola dan teks sangat kecil.

Rasool, *et al*., (2012: 3397) meneliti algoritma pencocokan *string* Boyer Moore, Aho-Corasick dan KMP efisien. Algoritma Boyer Moore cepat dalam hal jumlah *string* yang lebih besar. Tinus Wuruwu dan Rila Mandala (2016: 42) menyatakan bahwa algoritma Boyer Moore lebih cepat daripada algoritma Knuth Morris Pratt dalam waktu akses pencocokan *string*.

Pada penelitian ini akan dibangun sistem penyaringan *email* berbasis *web* yang dapat diakses malalui *browser*. Agar memudahkan dalam pembuatan sistem penyaringan *email* menggunakan *framework.* Ada banyak jenis *framework* antara lain Codeigniter, Symfony, dan Laravel. Laaziri, *et al*., (2019: 870) melakukan pengujian terhadap *framework* Codeigniter, Symfony, dan Laravel menggunakan metode *Qualification and Selection of Opensource Software* (QSOS) yang secara resmi diakui sebagai standar untuk membandingkan *framework*. Hasil penelitian dapat disimpulkan bahwa *framework* Laravel dan Symfony saat ini memenuhi standar dan persyaratan modern. Laravel dan Symfony memiliki fitur yang banyak dibandingkan dengan Codeigniter yang tidak memenuhi persyaratan saat ini. Untuk evaluasi kinerja tiga *framework*: Laravel, Symfony, dan Codeigniter sesuai dengan kriteria evaluasi seperti permintaan per detik, penggunaan memori, waktu respon dan jumlah *file* yang diperlukan, Hasil yang diperoleh menunjukkan bahwa Laravel melampaui *framework* MVC lainnya. Laravel digunakan untuk pengembangan aplikasi skala besar yang cepat.

Berdasarkan penelitian relevan tersebut penulis membandingkaan beberapa algoritma dan menetapkan algoritma Boyer Moore untuk diimplentasikan dalam

penelitian ini, karena data *string* yang diperoleh dari *header* berjumlah besar. Untuk pembuatan sistem menggunakan *framework* Laravel.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat diidentifikasi beberapa permasalahan sebagai berikut:

1. Sistem penyaringan *email* di *provider* dapat terjadi kesalahan dalam mengkategorikan *email* sebagai *spam*.
2. Penggunaan gTLD yang baru diluncurkan belum mempunyai reputasi atau gTLD bereputasi buruk pada alamat *email* memiliki permasalahan yaitu *email* yang dikirim dianggap sebagai *spam* oleh sistem penyaringan *email* yang dimiliki *provider email*.
3. *Email-email* yang berada pada *folder spam* tidak ada tanda yang melatarbelakangi *email* tersebut bisa dianggap sebagai *spam* sehingga pengguna akan sulit memilah.

1.3 Batasan Masalah

Agar tidak melebar dari latar belakang dan permasalahan yang telah dijelaskan sebelumnya maka dalam menyelesaikan penelitian ini penulis membuat batasan masalah sebagai berikut:

1. Penyaringan *email* sebagai *spam* berdasarkan data pada *header email* dengan parameter: *subject*, *email*, *domain*, dan *top level domain*.
2. Sistem penyaringan *email* dirancang menggunakan *framework* Laravel.
3. Menggunakan algoritma Boyer Moore untuk menyaring *email*.

1.4 Perumusan Masalah

Berdasarkan uraian di atas, penelitian ini difokuskan untuk mengimplementasikan algoritma Boyer Moore pada sistem penyaringan *email*, yang permasalahannya diperoleh sebagai berikut:

1. Bagaimana merancang dan membuat sistem penyaringan *email*?
2. Bagaimana menerapkan algoritma Boyer Moore pada sistem penyaringan *email*?
3. Bagaimana hasil pengujian menggunakan pengujian performa algoritma, *white box* dan *black box* pada sistem penyaringan *email*?

1.5 Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Merancang dan membuat sistem penyaringan *email*.
2. Menerapkan algoritma Boyer Moore pada sistem penyaringan *email.*
3. Mengetahui hasil pengujian performa algoritma, *white box* dan *black box* pada sistem penyaringan *email* yang telah dibuat.

1.6 Manfaat Penelitian

Manfaat dari penelitian ini adalah :

1. Terbentuknya sebuah sistem penyaringan *email*.
2. Dapat mengetahui kelebihan dan kekurangan dari algoritma Boyer Moore dalam menyaring *email.*
3. Sebagai bahan studi kasus bagi pembaca dan acuan bagi mahasiswa, terutama bagi yang ingin melakukan penelitian sejenis.

BAB II

KAJIAN PUSTAKA DAN LANDASAN TEORI

2.1 Kajian Pustaka

Penelitian yang relevan dapat diperoleh dengan mencari referensi seperti jurnal, tesis, dan penelitian terdahulu seperti berikut:

Eza Rahmanita (2014: 15) dalam penelitian "Pencarian *String* Mengunakan Algoritma Boyer Moore pada Dokumen" menyatakan proses pencarian merupakan salah satu kegiatan penting dalam pemrosesan data. Proses ini dapat menghabiskan waktu dalam ruang pencarian yang besar sehingga diperlukan suatu teknik pencarian yang efisien. Algoritma Boyer Moore merupakan suatu solusi pencarian yang efisien dapat melakukan perbandingan *pattern* mulai dari kanan ke kiri. Jika terjadi ketidakcocokan *string* dari kanan *pattern* maka ketidakcocokan akan membantu menggerakkan *pattern* tersebut dengan jarak yang lebih jauh. Gerakan melompat ini akan memberikan informasi berapa banyak *pattern* harus digeser untuk mencocokan karakter terakhir yang cocok dengan kemunculan awal *pattern*. Artinya, akan lebih signifikan dalam mengurangi proses perbandingan, jika bisa melompati atau tidak melakukan perbandingan karakter yang diprediksi akan gagal. Algoritma Boyer Moore mempunyai keunggulan dalam waktu menemukan *pattern* yang akan dicari dalam ukuran *file* yang lebih besar. Pada *file* berekstensi .txt dengan *file size* 4.625 *byte* dengan varian *keyword* berbeda. Varian *keyword* yang sedikit dapat ditempuh dengan waktu lebih cepat pada *pattern* 'a' yaitu 0,228 detik dengan banyak *pattern* ditemukan 685. Pada *file* berekstensi .doc dengan *file size*

39.936 *byte*, pada *pattern* ‘yang’ dapat diproses dengan waktu 0,542 detik dengan ditemukannya *pattern* sebanyak 18. Sedangkan pada *file* berekstensi .pdf optimalisasi dari *pattern matching* yang diproses adalah 0,103 detik dengan *file size* 15.804 *byte* dan banyak *pattern* = 1 untuk pencarian *pattern* ‘suatu’.

Baranwal, *et al*., (2018: 37) dalam penelitian “*Spam Filtration using Boyer Moore Algorithm and Naïve Method*” mengembangkan *classifier* Naïve Bayes klasik dengan teknik pembelajaran *non*-mesin dan algoritma pencocokan pola. Setelah menganalisis kinerja model yang diusulkan, dapat disimpulkan bahwa algoritma Boyer Moore meningkatkan kinerja dalam bentuk akurasi proses klasifikasi.

Shukor Bin Abd Razak dan Ahmad Fahrulrazie Bin Mohamad (2013: 353) dalam penelitian “*Identifcation of Spam Email Based on Information from Email Header*” menyatakan bahwa ada banyak teknik deteksi *spam* yang telah diusulkan dan diimplementasikan. *Spammer* dapat menggunakan teknik-teknik canggih untuk menghindari sistem penyaringan. Tidak cukup untuk mendeteksi berbagai jenis pesan *spam* dengan melihat informasi di badan *email*. Berdasarkan informasi yang ada di *header* menjadi pendekatan yang populer di kalangan peneliti untuk mengidentifikasi fitur potensial yang dapat digunakan untuk mendeteksi perilaku *spam*.

Rahim, *et al*., (2017: 4) dalam penelitian “*Visual Approach of Searching Process using Boyer-Moore Algorithm*” menyatakan bahwa proses visualisasi dari algoritma Boyer Moore memudahkan peneliti atau pelajar untuk mempelajari bagaimana cara kerja algoritma Boyer Moore, dan dalam pengembangan aplikasi

akan lebih mudah untuk membuat fungsi pencarian kata dan dapat diimplementasikan ke dalam banyak proses pencarian.

Fince Tinus Wuruwu dan Rila Mandala (2016: 42) dalam penelitian "Perbandingan Algoritma Knuth Morris Pratt dan Boyer Moore dalam Pencocokan *String* pada Aplikasi Kamus Bahasa Nias" menyimpulkan pada perbandingan algoritma Knuth Morris Pratt dan Boyer Moore yang lebih cepat dalam waktu akses pencocokan *string* adalah algoritma Boyer.

Pranit Chettri dan Chinmoy Kar (2016: 26) dalam penelitian "*Comparative Study between Various Pattern Matching Algorithms*" meneliti berbagai algoritma pencarian teks yaitu algoritma Knuth-Morris Pratt, algoritma Pencarian Naïve dan algoritma Boyer Moore dengan menyediakan *input* teks berbagai ukuran dan menganalisis perilaku mereka pada *input* variabel. Setelah menganalisis dan melakukan studi pada algoritma, hasilnya menyatakan bahwa algoritma Boyer Moore bekerja dengan sangat baik dan efisien daripada yang lainnya ketika berhadapan dengan set data yang lebih besar.

Srinivas, *et al*., (2018), dalam penelitian "*Study of String Matching Algorithm*" menyimpulkan algoritma Knuth Morris Pratt bergerak maju hanya dalam urutan *input* dan membutuhkan ruang tambahan. Algoritma Rabin Karp digunakan untuk mengidentifikasi plagiarisme dan membutuhkan ruang tambahan untuk pencocokan. Algoritma Boyer Moore sangat cepat dalam urutan yang luas, Boyer Moore menghindari banyak perbandingan yang tidak berguna dengan pola pencarian yang secara mendasar berkaitan dengan konten; kompleksitas *running case* terbaiknya adalah *sub linear*. Hasil menunjukkan bahwa Boyer Moore adalah

algoritma terbaik untuk mengatasi masalah pencocokan *string* dalam kasus-kasus biasa, dan Rabin Karp adalah pilihan yang layak untuk beberapa kasus tertentu, misalnya ketika pola dan teks sangat kecil.

Vina Sagita dan Maria Irmina Prasetiyowati (2013: 31) dalam penelitian "Studi Perbandingan Implementasi Algoritma Boyer Moore, Turbo Boyer Moore, dan *Tuned* Boyer-Moore dalam Pencarian *String*" menyatakan bahwa algoritma Boyer Moore memiliki waktu pencarian tercepat dari tiga varian Boyer Moore yang telah ditentukan. Algoritma Turbo Boyer Moore merupakan algoritma tercepat kedua dan yang paling lambat adalah *Tuned* Boyer Moore.

Berdasarkan hasil uji performa dari penelitian yang relevan menetapkan algoritma Boyer Moore untuk diimplentasikan pada sistem penyaringan *email* di dalam penelitian ini. Kecepatan pencocokan adalah faktor yang utama dikarenakan data *string* pada *header* berjumlah besar.

2.2 Dasar Teori

2.2.1 Algoritma

Lamhot Sitorus (2015: 2) algoritma berasal dari kata *al-kwarizmi* yang terdapat di buku Abu Ja'far Muhammad Ibnu Musa Al-Kwarizmi, seorang ahli Matematika dari Persia dengan judul bukunya "Aljabar wal Muqabala". Menurut Donald E. Knuth, algoritma yang baik memiliki kriteria sebagai berikut:

a. *Input*

   Suatu algoritma harus memiliki 0 (nol) atau lebih masukan (*input*). Artinya, suatu algoritma itu dimungkinkan tidak memiliki masukan secara langsung dari pengguna tetapi dapat juga memiliki beberapa masukan. Algoritma yang tidak

memiliki masukan secara langsung dari pengguna, maka semua data dapat diinisialisasikan atau dibangkitkan dalam algoritma.

b. *Output*

Suatu algoritma harus memiliki satu atau lebih *output*. Suatu algoritma yang tidak memiliki keluaran (*output*) adalah suatu algoritma yang sia-sia, yang tidak perlu dilakukan. Algoritma dibuat untuk tujuan menghasilkan suatu yang diinginkan, yaitu berupa hasil keluaran.

c. *Finiteness*

Setiap pekerjaan yang dikerjakan pasti berhenti. Demikian juga algoritma harus dapat dijamin akan berhenti setelah melakukan sejumlah langkah proses.

d. *Definiteness*

Tidak menimbulkan makna ganda (*ambiguous*). Setiap baris aksi/ pernyataan dalam suatu algoritma harus pasti, artinya tidak menimbukan penafsiran lain bagi setiap pembaca algoritma, sehingga memberikan *output* yang sesuai dengan yang diharapan oleh pengguna.

e. *Effectiveness*

Langkah-langkah algoritma dikerjakan dalam waktu yang wajar. Suatu algoritma tidak terdapat suatu aksi yang tidak perlu dilakukan. Setiap aksi akan memerlukan waktu eksekusi, padahal aksi tersebut jelas tidak berpengaruh atau tidak ada gunanya. Misalnya aksi $X \leftarrow X + 0$. Aksi ini jelas tidak ada pengaruh dan tidak ada gunanya karena $X + 0$ akan menghasilkan bilangan X juga yang berarti tidak berguna. Jadi tidak perlu dilakukan karena sia-sia.

Lamhot Sitorus (2015: 5) secara umum, algoritma terdiri dari sekuensial (*sequential*), tes kondisi atau percabangan (*branching*) dan perulangan (*looping*).

1. Algoritma Sekuensial (*Sequential*)

   Dalam kehidupan sehari-hari, algoritma yang sering dilakukan adalah algoritma sekuensial. Algoritma sekuensial adalah langkah-langkah atau aksi-aksi yang dilakukan secara berurutan sesuai dengan urutan penulisnya. Jika urutan penulisannya diubah, maka kemungkinan akan memberikan hasil akhir yang berbeda pula. Misalkan algoritma memiliki 4 baris aksi, yaitu t1, t2, t3 dan t4, maka semua aksi akan dilaksanakan secara berurutan mulai aksi t1 hingga t4. Setelah selesai mengerjakan aksi t1, maka aksi t2 akan dilaksanakan. Setelah selesai mengerjakan aksi t2, maka aksi t3 akan dilaksanakan. Setelah selesai mengerjakan aksi t3, maka aksi t4 akan dilaksanakan. Setelah selesai mengerjakan aksi t4, baru algoritma berhenti karena aksi t4 merupakan aksi terakhir.

2. Algoritma Percabangan (*Branching*)

   Ada kalanya suatu kegiatan tertentu akan dilaksanakan atau tidak akan dilakukan karena tergantung situasi atau kondisi tertentu. Demikian juga dalam algoritma ada kalanya satu atau beberapa aksi akan dikerjakan atau tidak akan dikerjakan tergantung kondisi tertentu, akan dikerjakan jika kondisi tertentu dipenuhi. Percabangan ini hanya mengerjakan satu aksi dari dua atau lebih pilihan yang diberikan.

3. Algoritma Perulangan (*Looping*)

   Banyak kegiatan sehari-hari yang harus dilakukan secara berulang-ulang, misalnya makan berjalan, naik tangga dan lain-lain. Dalam membuat algoritma juga mengenal perulangan, yaitu melaksanakan satu atau beberapa aksi secara berulang-ulang sesuai dengan kebutuhan atau kondisi. Salah satu kelebihan komputer dibandingkan dengan manusia adalah melaksanakan pekerjaan secara berulang-ulang tanpa mengenal istilah lelah atau bosan. Manusia dalam melaksanakan pekerjaan yang sama secara berulang-ulang akan mengalami kelelahan dan bosan yang cenderung akan melaksanakan kesalahan.

2.2.2 Algoritma Boyer Moore

Algoritma Boyer Moore diperkenalkan oleh Bob Boyer dan J.S. Moore pada tahun 1977, pada algoritma Boyer Moore pencocokan kata dimulai dari karakter terakhir kata kunci menuju karakter awalnya, jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu, hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci (Eza Rahmanita, 2014: 17) .

Rahim, *et al.*, (2017: 2) langkah-langkah pencarian *pattern* pada teks menggunakan algoritma Boyer Moore sebagai berikut:

a. Pertama, membutuhkan tabel yang mempunyai nilai *Match Heuristic* (MH), dan *Occurrence Heuristic* (OH) untuk menentukan jumlah perpindahan yang akan dilakukan pada suatu *pattern* jika ada yang tidak sesuai pada karakter dalam proses pencocokan karakter terhadap karakter teks.

b. Jika dalam proses perbandingan ada ketidakcocokan antara karakter pada *pattern* dengan karakter pada teks dan karakter pada teks yang tidak cocok terdapat pada karakter *pattern* maka pergeseran dilakukan dengan menggunakan nilai terbesar dari nilai MH dan OH.
c. Jika terjadi ketidakcocokan namun karakter yang tidak cocok pada teks tidak terdapat pada *patte*rn maka *pattern* bergeser sebanyak panjang *pattern*.
d. Jika karakter dalam teks cocok dengan karakter pada *pattern* maka karakter mencocokan karakter sebelah kiri yang belum dicocokan sampai karakter *pattern* yang paling kiri.
e. Jika semua karakter memiliki kecocokan maka *pattern* telah ditemukan pada teks.

Contoh cara kerja algoritma Boyer Moore ini adalah sebagai berikut:

a. Deklarasi teks dan *pattern*.

Teks: ABDBCADDABC

Pattern: ABC

Tabel 2.1 Nilai MH dan OH

Index	0	1	2	
Karakter	A	B	C	*
MH	2	1	0	3
OH	1			

Pada tabel 2.1 dapat dijelaskan bahwa hanya karakter unik pada *pattern* yang diambil pada tabel, *index* dimulai dari 0, nilai OH yaitu 1 dan nilai MH di

setiap karakter pada *pattern* diperoleh dengan cara panjang *pattern* dikurangi *index* dan dikurangi 1. Nilai MH dan OH merupakan jumlah perpindahan atau pergeseran *pattern* ke kanan, jika karakter pada teks yang tidak cocok terdapat pada *pattern* maka nilai MH dan OH dibandingkan dan diambil nilai terbesarnya. Untuk karakter yang tidak ada pada *pattern* maka nilai MH sama dengan panjang *pattern*.

b. Langkah pertama

Index	0	1	2	3	4	5	6	7	8	9	10	11
Teks	A	B	D	B	A	C	D	B	D	A	B	C
Pattern	A	B	C									

Karakter paling kanan dari *pattern* yaitu "C" tidak cocok dengan karakter yang sejajar pada teks yaitu "D". karakter “D” tidak terdapat pada *pattern* maka *pattern* berpindah sebanyak panjang *pattern*. Panjang *pattern* diketahui yaitu 3 maka *pattern* bergeser ke kanan sebanyak 3 karakter, dari *index* 2 menuju *index* 5.

c. Langkah kedua

Index	0	1	2	3	4	5	6	7	8	9	10	11
Teks	A	B	D	B	A	C	D	B	D	A	B	C
Pattern				A	B	C						

Karakter paling kanan dari *pattern* yaitu "C" cocok dengan karakter yang sejajar pada teks yaitu "C". Jika terjadi kecocokan maka langkah selanjutnya mencocokan karakter sebelah kiri dari karakter *pattern* yang sudah cocok. Karakter yang selanjutnya dicocokan yaitu karakter “B” terhadap karakter yang sejajar pada teks yaitu “A”. ternyata hasilnya terjadi ketidakcocokan, karakter “A” yang terdapat pada *pattern* memiliki nilai MH yaitu 2. Jika karakter teks yang tidak cocok terdapat

pada *pattern* maka dilakukan perbandingan nilai MH dan OH. Diambil nilai yang paling besar diantara keduanya. Hasilnya diperoleh nilai terbesar yaitu 2 maka *pattern* bergeser ke kanan sebanyak 2 karakter, dari *index* 5 menuju *index* 7.

d. Langkah ketiga

Index	0	1	2	3	4	5	6	7	8	9	10	11
Teks	A	B	D	B	A	C	D	B	D	A	B	C
Pattern						A	B	C				

Karakter paling kanan dari *pattern* yaitu "C" tidak cocok dengan karakter yang sejajar pada teks yaitu "B". karakter “B” terdapat pada *pattern*. Jika karakter teks yang tidak cocok terdapat pada *pattern* maka dilakukan perbandingan nilai MH dan OH. Karakter “B” yang terdapat pada *pattern* memiliki nilai MH yaitu 1. Diambil nilai yang paling besar diantara keduanya. Hasilnya diperoleh nilai terbesar yaitu 1 maka *pattern* bergeser ke kanan sebanyak 1 karakter, dari *index* 7 menuju *index* 8.

e. Langkah keempat

Index	0	1	2	3	4	5	6	7	8	9	10	11
Teks	A	B	D	B	A	C	D	B	D	A	B	C
Pattern							A	B	C			

Karakter paling kanan dari *pattern* yaitu "C" tidak cocok dengan karakter yang sejajar pada teks yaitu "D". karakter “D” tidak terdapat pada *pattern* maka *pattern* berpindah sebanyak panjang *pattern*. Panjang *pattern* diketahui yaitu 3 maka *pattern* bergeser ke kanan sebanyak 3 karakter, dari *index* 8 menuju *index* 11.

f. Langkah kelima

Index	0	1	2	3	4	5	6	7	8	9	10	11
Teks	A	B	D	B	A	C	D	B	D	A	B	C
Pattern										A	B	C

Karakter paling kanan dari *pattern* yaitu "C" cocok dengan karakter yang sejajar pada teks yaitu "C". Jika terjadi kecocokan maka langkah selanjutnya mencocokan karakter sebelah kiri dari karakter *pattern* yang sudah cocok. Karakter yang selanjutnya dicocokan yaitu karakter pada *pattern* yaitu “B” terhadap karakter yang sejajar pada teks yaitu “B”. Ternyata hasilnya terjadi kecocokan. Jika terjadi kecocokan maka langkah selanjutnya mencocokan karakter sebelah kiri dari karakter *pattern* yang sudah cocok. Karakter yang selanjutnya dicocokan karakter pada *pattern* yaitu “A” terhadap karakter yang sejajar pada teks yaitu “A”. Ternyata hasilnya terjadi kecocokan. Semua karakter *pattern* sudah dicocokan terhadap karakter yang sejajar pada teks dan hasilnya semua karakter pada *pattern* cocok. Jadi dapat diambil kesimpulan bahwa *pattern* ditemukan pada teks.

2.2.3 *Email*

Electronic mail atau biasa disingkat sebagai *email*, merupakan sebuah metode untuk mengirimkan pesan dalam bentuk digital, pesan ini biasanya dikirimkan melalui medium internet, sebuah pesan elektronis tediri dari isi, alamat pengirim, dan alamat-alamat yang dituju (Edy Victor Haryanto, 2012: 127).

Sistem *email* beroperasi di atas jaringan berbasis pada model *store* and *forward*. Sistem ini mengaplikasikan sebuah sistem *server email* yang menerima, meneruskan, mengirimkan, serta menyimpan pesan-pesan pengguna. Pengguna

hanya perlu menghubungkan PC ke dalam jaringan (Edy Victor Haryanto, 2012: 127).

Mengirim sebuah *email* dari alamat *email* yang satu ke alamat *email* yang lain digunakan sebuah *protocol* (aturan), yaitu *Simple Mail Transfer Protocol* (SMTP). SMTP telah menjadi aturan dasar yang disepakati untuk mengirim *email* (Edy Victor Haryanto, 2012: 128).

2.2.4 *Internet Message Access Protocol* (IMAP)

IMAP merupakan metode akses *email*, yaitu dirancang untuk memungkinkan pengguna untuk menyimpan pesan secara permanen di *server*, IMAP menyelesaikan masalah *backup* administrator sistem dan memungkinkan pengguna untuk mengakses semua pesan dari tempat mana pun di dunia (selain pembatasan *firewall*), IMAP juga memiliki implementasi yang lebih luas dari koneksi yang dijamin TLS, membuat IMAP aman untuk meningkatkan kinerja dan memungkinkan pengguna untuk bekerja dengan *mailbox* mereka sementara tidak terhubung ke *server email*, sebagian besar aplikasi *email* dengan dukungan IMAP *caching mailbox* dan pesan yang diunduh di *hard drive* lokal (Haycox, *et al*., 2009: 11).

IMAP mendukung banyak *folder* dan menyimpan informasi status pesan (baik pesan belum dibaca, dibalas, atau dihapus) di *server*. Ini berarti bahwa pengguna yang mengakses penyimpanan pesan mereka dari lokasi yang berbeda, dengan kemungkinan *email client* yang berbeda, akan disajikan dengan tampilan pesan mereka yang konsisten dan terbaru. IMAP juga mendukung pencarian sisi

server, sehingga aplikasi *client* tidak perlu mengunduh semua pesan untuk mencari *email* (Haycox, *et al*., 2009: 11).

2.2.5 *Simple Mail Transfer Protocol* (SMTP)

Simple Mail Transfer Protocol (SMTP) adalah protokol standar untuk pengiriman *email* di internet, SMTP adalah protokol yang cukup sederhana, berbasis teks dan protokol ini menyebutkan satu atau lebih penerima *email* untuk kemudian diverifikasi, jika penerima *email* valid, *email* akan segera dikirim (Edy Victor Haryanto, 2012: 130).

SMTP adalah protokol teks *line-oriented* yang berjalan melalui TCP, yang membuatnya mudah untuk memecahkan kode transkrip SMTP dan untuk memulai sesi SMTP menggunakan klien telnet biasa yang ditemukan di hampir semua komputer (Haycox, *et al*., 2009: 11). Namun, karena terbatas pada kemampuannya untuk antrian pesan di sisi penerima, biasanya akan menggunakan salah satu dari dua protokol ini, yaitu POP3 (*Post Office Protocol* versi 3) dan IMAP (*Internet Message Access Protocol*) yang memungkinkan pengguna menyimpan pesan dalam kotak surat (*mailbox*) *server* dan mengunduh secara berkala dari *server* (Edy Victor Haryanto, 2012: 131).

2.2.6 HTML

HTML merupakan singkatan dari *Hypertext Markup Language*. Singkatan ini terdiri dari 3 komponen kata, yakni: *Hypertex*t, *Markup* dan *Language*. Kata *Hypertext* dari HTML menekankan pengertian *text* yang lebih dari sekedar teks (‘*hyper*’- *text*), maksudnya selain berfungsi sebagai teks biasa, sebuah teks di dalam

HTML juga biasa berfungsi sebagai penghubung ke halaman lain atau dikenal dengan istilah *link* (Andre Pratama, 2015: 1).

Kata kedua dari singkatan HTML adalah *Markup*. *Markup* dapat diterjemahkan sebagai tanda atau penanda (bahasa inggris: *mark*), di dalam HTML terdapat tanda-tanda khusus. Tanda ini diperlukan untuk mengatur format dan membuat struktur halaman *web* (Andre Pratama, 2015: 1).

Bagian terakhir dari HTML adalah *Language*. Istilah *language* jika diterjemahkan berarti: bahasa, HTML tidak menggunakan '*Programming Language*', tetapi hanya '*Language*' saja, hal ini secara tidak langsung menyatakan bahwa HTML bukanlah sebuah bahasa pemrograman. HTML tidak memiliki struktur dasar seperti variabel, kondisi *IF*, *function*, atau *class* seperti layaknya sebuah bahasa pemrograman komputer (Andre Pratama, 2015: 1).

2.2.7 *Hypertext Preprocessor* (PHP)

PHP merupakan singkatan dari PHP: *Hypertext Preprocessor*, singkatan ini disebut singkatan rekursif, yakni permainan kata dimana kepanjangannya juga terdiri dari singkatan PHP itu sendiri, yakni PHP: *Hypertext Preprocessor*. *Hypertext Preprocessor* bisa diterjemahkan sebagai 'pemroses *hypertext*', atau 'pemroses HTML'. Dalam pengertian paling sederhana, PHP adalah bahasa pemrograman *web* yang digunakan untuk men-*generate* atau menghasilkan kode HTML. PHP termasuk Bahasa pemrograman *server side*, karena termasuk bahasa pemrograman berbasis *server*, kita harus menjalankan kode-kode PHP dari sebuah *server*, dengan kata lain, kode PHP tidak bisa dijalankan tanpa *server* (Andre Pratama, 2016:4).

2.2.8 *Model-View-Controller* (MVC)

Yudho Yudhanto dan Helmi Adi Prasetyo (2018: 6) *Model-View-Controller* (MVC) adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*).

a. *Model* mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data, seperti memasukkan data ke basis data, pembaruan data, dan lain-lain.
b. *View* adalah bagian yang mengatur tampilan ke pengguna.
c. *Controller* merupakan bagian yang menjembatani *model* dan *view*. *Controller* berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman *web*.

2.2.9 *Framework* Laravel

Laravel adalah *framework* pengembangan web MVC yang ditulis dalam bahasa pemrograman PHP, Laravel telah dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi baik biaya pengembangan awal dan biaya pemeliharaan berkelanjutan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi, dengan memberikan sintaksis ekspresif yang jelas dan serangkaian fungsionalitas inti yang akan menghemat waktu implementasi. Laravel telah dirancang sebagai *framework* yang minimalis dan fungsionalitas, sehingga lebih mudah untuk memahami basis kode yang lebih kecil dan Laravel mengimplementasikan solusi dengan cara yang bersih, sederhana, dan elegan (Shawn McCool 2012: 3).

Laravel menawarkan modularitas kode, hal ini dicapai melalui kombinasi *driver* dan sistem *bundle*-nya, *driver* memungkinkan pengguna mudah untuk mengubah dan memperluas fungsi *caching*, *session*, *database,* dan *authentication.* Dengan menggunakan *bundle*, pengguna dapat mengemas segala jenis kode untuk digunakan kembali sendiri atau untuk disediakan ke seluruh komunitas Laravel (Shawn McCool 2012: 3).

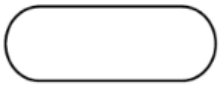
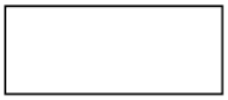
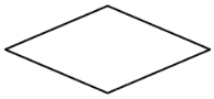
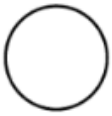
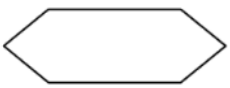
2.2.10 MySQL

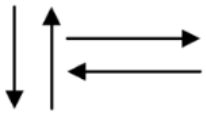
MySQL adalah salah satu aplikasi RDBMS (*Relational Database Management System*) pengertian sederhana RDBMS adalah aplikasi *database* yang menggunakan prinsip relasional, MySQL bersifat gratis dan *open source*, artinya setiap orang boleh menggunakan dan mengembangkan aplikasi ini, namun walaupun gratis, MySQL di-*support* oleh ribuan *programmer* dari seluruh dunia (Andre Pratama, 2017:2).

2.2.11 *Flowchart*

Lamhot Sitorus (2015: 14) *flowchart* berfungsi untuk menggambarkan urutan logika dari suatu prosedur pemecahan masalah supaya sebuah algoritma yang terstruktur mudah dipahami oleh orang lain (khususnya *programmer* yang bertugas mengimplementasikan program), sehingga *flowchart* merupakan langkah-langkah penyelesaian masalah yang dituliskan dalam simbol-simbol tertentu. Simbol-simbol yang digunakan untuk menggambarkan algoritma dalam bentuk diagram alir dan kegunaan dari simbol-simbol yang bersangkutan dapat dilihat pada tabel 2.2.

Tabel 2.2 Simbol-Simbol *Flowchart*

No	Simbol	Nama	Fungsi
1		*Terminal*	Menyatakan permulaan atau akhir suatu program
2		*Input* / *Output*	Menyatakan proses *input* atau *output* tanpa tergantung jenis peralatannya
3		*Process*	Menyatakan suatu tindakan (proses) yang dilakukan oleh komputer
4		*Decision*	Menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban: ya / tidak
5		*Connector*	Menyatakan sambungan dari proses ke proses lainnya dalam halaman yang sama
6		*Offline Connector*	Menyatakan sambungan dari proses ke proses lainnya dalam halaman yang berbeda
7		*Predefined Process*	Menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal
8		*Punched Card*	Menyatakan *input* berasal dari kartu atau *output* ditulis ke kartu

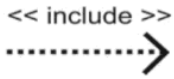
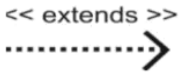
No	Simbol	Nama	Fungsi
9		*Document*	Mencetak keluaran dalam bentuk dokumen (melalui *printer*)
10		*Flow*	Menyatakan jalannya arus suatu proses

2.2.12 Use Case

Uus Rusmawan (2019: 72) *use case* adalah teknik untuk merekam persyaratan fungsional sebuah sistem, *use case* mendiskripsikan interaksi tipikal antara para pengguna sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem digunakan. Dua hal penting dalam *use case* adalah skenario dan aktor. Skenario adalah rangkaian langkah-langkah yang menjabarkan sebuah interaksi antara seorang pengguna dengan sebuah sistem. secara umum simbol-simbol *use case* dapat dilihat pada tabel 2.3.

Tabel 2.3 Simbol – Simbol *Use Case*

Simbol	Nama	Keterangan
	*Actor*	Seseorang atau apa saja yang berhubungan dengan sistem yang sedang dibangun.
	*Use case*	Menggambarkan bagaimana seseorang menggunakan sistem
	Relasi asosiasi	Relasi yang dipakai untuk menunjukkan hubungan antara *actor* dan *use case*

Simbol	Nama	Keterangan
	Relasi *include*	Memungkinkan fungsionalitas yang disediakan oleh *use case* lainnya
	Relasi *extend*	Memungkinkan suatu *use case* secara *optional* menggunakan fungsionalitas yang disediakan oleh *use case* lainnya

2.2.13 ISO 9126

Pressman (2014: 418) standar ISO 9126 dikembangkan dalam usaha identifikasi atribut-atribut kualitas perangkat lunak komputer. Standar ISO 9126 mengidentifikasi 6 atribut-atribut kualitas perangkat lunak, yang dijabarkan sebagai berikut:

a. Fungsionalitas. Derajat tentang bagaimana perangkat lunak memenuhi kebutuhan yang telah ditetapkan sebelumnya dan memiliki sub-sub atribut berikut ini: kecocokan, akurasi, interoperabilitas, kesesuaian dan keamanan.
b. Keandalan. Jumlah waktu penggunaan perangkat lunak yang tersedia dan memiliki sub atribut-sub atribut: kematangan, toleransi kesalahan, kemampuan untuk melakukan pemulihan.
c. Kemudahan penggunaan. Derajat tentang bagaimana kemudahan perangkat lunak digunakan, dimana hal ini seringkali diindikasikan menggunakan sub atribut-sub atribut berikut ini: kemudahan untuk dipahami, kemudahan untuk dipelajari, operabilitas.

d. Efisiensi. Derajat penggunaan sumber daya sistem secara optimal, dimana hal ini diindikasikan oleh sub atribut-su batribut berikut ini: perilaku waktu, perilaku sumber daya.
e. Kemudahan pemeliharaan. Kemudahan yang menentukan tentang bagaimana perbaikan-perbaikan mungkin dilakukan pada suatu perangkat lunak, dimana hal ini diindikasikan menggunakan sub atribut-sub atribut berikut ini: kemampuan untuk dilakukan analisis, kemampuan untuk dilakukan perubahan, hal-hal yang berkaitan dengan stabilitas, serta kemampuan untuk dilakukan pengujian
f. Portabilitas. Kemudahan bagaimana perangkat lunak dapat dipindahkan dari suatu lingkungan operasional ke lingkungan operasional yang lainnya, yang hal ini diidentifikasikan menggunakan sub atribut-sub atribut: kemampuan untuk beradaptasi, kemampuan untuk dipasangkan, kesesuaian, kemampuan untuk digantikan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat diambil kesimpulan bahwa:

1. Sistem penyaringan *email* dibuat menggunakan *framework* laravel yang merupakan salah satu *framework* yang memisahkan antara data (*model*) dari tampilan (*view*) dan cara bagaimana memprosesnya (*controller*). Dalam pengembangan menggunakan metode *waterfall* yang terdiri dari *communication, planning, modelling, construction,* dan *deployment*.
2. Sistem penyaringan *email* dalam mengkategorikan *email* itu sebagai *spam* menggunakan algoritma Boyer Moore yang berdasarkan data *header email* yaitu, subyek, alamat *email*, *domain*, *Top Level Domain* (TLD).
3. Performa algoritma Boyer Moore pada sistem penyaringan *email* dapat dikategorikan cepat dikarenakan proses pencocokan teks dan *pattern* dalam waktu kurang dari satu detik yaitu rata-rata 20,85 μs. Performa algoritma Boyer Moore dapat maksimal atau lebih cepat dalam proses pencocokan jika teks dan *pattern* memiliki jumlah karakter unik yang banyak pada teks atau *pattern* yang panjang. Hasil pengujian *white box* diperoleh nilai *cyclometic complexity* yaitu 5 yang menujukan algoritma Boyer Moore merupakan algoritma yang sederhana dan memiliki resiko rendah. Pengujian *black box* memperhatikan beberapa aspek yaitu *functionality*,

portability, *efficiency*, dan *usability*. Pada aspek *functionality* memperoleh hasil 100% yang berarti semua fungsi berjalan dan berfungsi dengan baik. Pada aspek *portability* memperoleh hasil 100% yang menunjukan sistem dapat di akses dari berbagai macam *browser* dan *device*. Pada aspek *efficiency* sistem sudah baik karena memperoleh rata-rata *grade* PageSpeed A (93%) dan Yslow B (84%), pada rata-rata *page size* 362 KB dengan *load time* rata-rata 3,2 detik. Pada aspek *usability* memperoleh hasil skor SUS yaitu 77,84. Jika berdasarkan cara penilaian SUS maka pada aspek *usability* sistem termasuk pada kategori *acceptable*, memperoleh *grade* C, dan *adjective rating* pada kategori *excellent*.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, diajukan saran penelitian lanjutan sebagai berikut:

1. Gunakan *framework* lain yang arsitekturnya dapat menangani dan meyediakan fitur *real time*.
2. Gunakan algoritma lain yang bukan berjenis *exact string matching* untuk penyaringan *email* berdasarkan subyek.
3. Gunakan arsitektur *microservice*.

DAFTAR PUSTAKA

Ardiansyah, & Ghazali, M. I. (2016). Pengujian Usability User Interface Dan User Experience Aplikasi E-Reader Skripsi Berbasis Hypertext. JITTER (Jurnal Ilmiah Teknik Informasi Terapan), II(3): 213–220.

Baranwal, A., Gaur G., Bhasker A., & Jain R. (2018). Spam Filtration using Boyer Moore Algorithm and Naïve Method, 180(42): 35–38.

Chettri, P., & Kar, C. (2016). Comparative Study between Various Pattern Matching Algorithms. International Journal of Computer Applications, 26–30.

Easton, C. R. (2012). ICANN's core principles and the expansion of generic top-level domain names. International Journal of Law and Information Technology, 20(4), 273–290. https://doi.org/10.1093/ijlit/eas013.

Ependi, U., Kurniawan, T. B., & Panjaitan, F. (2019). System Usability Scale Vs Heuristic Evaluation: a Review. Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer, 10(1), 65–74. https://doi.org/10.24176/simet.v10i1.2725

Haryanto, E. D. (2012). Jaringan Komputer. Yogyakarta: ANDI.

Haycox, I., McDonald, A., Bäck, M., Hildebrandt, R., Koetter, P. B., Rusenko, D., & Taylor. C. (2009). *Linux E-mail Set up, maintain, and secure a small office e-mail server*. Birmingham: Packt Publishing Ltd.

Ismail, F. (2018). Statistika Untuk Penelitian Pendidikan dan Ilmu-Ilmu Sosial. Jakarta: Pranamedia Group.

Laaziri, M., Benmoussa, K., Khoulji, S., & Kerkeb, M. L. (2019). A Comparative study of PHP frameworks performance. Procedia Manufacturing, 32, 864–871. https://doi.org/10.1016/j.promfg.2019.02.295

McCool, S. (2012). *Laravel Starter The defnitive introduction to the Laravel PHP web development framework*. Birmingham: Packt Publishing Ltd.

Nonprofit Business Advisor. (2016). With spam filters catching 7 percent of emails, nonprofits need to curate mailing lists. Nonprofit Business Advisor, 323: 1-3. https://doi.org/10.1002/nba.30216.

Parenrengi, A. M., Saputra, I. A., & Tajudin, L. M. (2017). Analisis Perbandingan Algoritma Boyer Moore Dan Algoritma Knuth Morris Pratt Pada Aplikasi Tripelka Foodshop Kendari Berbasis Android. *semanTIK*. 3(1): 115-126.

Pratama, A. (2015). *HTML Uncover Panduan Belajar HTML Untuk Pemula*. Cetakan Pertama. Padang Panjang: Duniailkom.

_____. (2016). *PHP Uncover Panduan Belajar PHP Untuk Pemula*. Cetakan Pertama. Padang Panjang: Duniailkom.

_____. (2017). *MySQL Uncover Panduan Belajar MySQL Dan MariaDB Untuk Pemula*. Cetakan Pertama. Padang Panjang: Duniailkom.

Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach*. Edisi Delapan. New York: McGraw-Hill Education.

Rahim, R., Ahmar, A. S., Ardyanti, A. P., & Nofriansyah, D. (2017). Visual Approach of Searching Process using Boyer-Moore Algorithm. Journal of Physics: Conference Series, 930(1). https://doi.org/10.1088/1742-6596/930/1/012001.

Rahmanita, E. (2014). Pencarian String Menggunakan Algoritma Boyer Moore Pada Dokumen. *Jurnal Ilmiah Nero* 1(1): 15-26.

Ramya. (2019, Mei). *Emails classified as Spam Office 365 - Reasons and recommendations*. Diakses pada 20 Juni 2019, dari https://help.zoho.com/portal/en/community/topic/emails-classified-as-spam-office-365-reasons-and-recommendations.

Rasool, A., Tiwari, A., Singla, G., & Khare, N. (2012). String Matching Methodologies: A Comparative Analysis. (IJCSIT) *International Journal of Computer Science and Information Technologies*, 3(2), 3394–3397.

Razak, S. A. B., & Mohammad, A. F. B. (2014). Identification of spam email based on information from email header. *International Conference on Intelligent Systems Design and Applications*, ISDA, 347–353. https://doi.org/10.1109/ISDA.2013.6920762

Sachhidanand. (2011). Analysis on Software Cyclomatic Complexity. International Journal of Engineering and Management Research, 1(1), 30-32.

Sagita, V., & Prasetiyowati, M. I. (2013). Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String. Jurnal ULTIMATICS, 5(1), 31–37. https://doi.org/10.31937/ti.v5i1.311

Sharfina, Z., & Santoso, H. B. (2017). An Indonesian adaptation of the System Usability Scale (SUS). 2016 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2016, 145–148. https://doi.org/10.1109/ICACSIS.2016.7872776

Sitorus, L. (2015). *Algoritma Dan Pemrograman*. Cetakan Pertama. Yogyakarta: ANDI.

Srinivas, I. V., Samnani, M. & Shaikh, M. S. (2018). Study of String Matching Algorithm. *IOSR Journal of Computer Engineering*, 32-35.

Sugiyono. (2017). *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Cetakan Ke-26. Bandung: Alfabeta.

Rusmawan, U. (2019). Teknik Penulisan Tugas Akhir dan Skripsi Pemrograman. Jakarta: Elex Media Komputindo.

Vezzani, S. (2014). ICANN's New Generic Top-Level Domain Names Dispute Resolution Procedure Viewed Against the Protection of the Public Interest of the Internet Community: Litigation Regarding Health-Related Strings. *The Law And Practice Of International Courts And Tribunals*, 13, 306-346.

Wuruwu, F. T., & Mandala, R. (2016). Perbandingan Algoritma Knuth Morris Pratt Dan Boyer Moore Dalam Pencocokan String Pada Aplikasi Kamus Bahasa Nias. *Jurnal Ilmiah INFOTEK*, 1(1), 36-43.

Yudhanto, Y., & Prasetyo H. A. (2018). Panduan Mudah Belajar Framework Laravel. Jakarta: Elex Media Komputindo.