

# ANALISIS PERBANDINGAN DAN IMPLEMENTASI STRING MATCHING DAN SQL QUERY PADA SISTEM INFORMASI PERSEDIAAN OBAT BERBASIS WEB APOTEK ERHA FARMA

<sup>1</sup>Fahrizal Firmansyah, <sup>2</sup>Fauziah, <sup>3</sup>Nur Hayati

<sup>1</sup>Fakultas Teknologi Komunikasi Informatika Universitas Nasional  
Jl. Sawo Manila, Pejaten Ps. Minggu Jakarta 12520

<sup>1</sup>fahrizalfsyh@gmail.com, <sup>2</sup>fauziah@civitas.unas.ac.id, <sup>3</sup>nurhayati@civitas.unas.ac.id

## Abstrak

*String matching merupakan metode yang melakukan pencocokan pattern dengan teks. Pada string matching terdapat beberapa algoritma, seperti algoritma Brute Force dan algoritma Boyer-Moore. Algoritma Brute Force melakukan pencocokan karakter yang dimulai dari pattern sebelah kiri, sedangkan pada algoritma Boyer-Moore pencocokan karakter dimulai dari pattern sebelah kanan. Selain menggunakan metode tersebut, terdapat juga SQL Query yang merupakan rangkaian dari beberapa perintah dasar SQL untuk melakukan pencarian. Ketiga metode ini memiliki cara kerja yang berbeda. Karena itulah, penelitian ini akan melakukan analisis perbandingan kinerja terhadap ketiga metode tersebut, di mana studi kasus yang dilakukan berfokus pada pencarian obat di Apotek Erha Farma. Mengingat mekanisme pencatatan persediaan obat di apotek tersebut masih menerapkan sistem konvensional, muncul kendala di mana para petugas apotek mengalami kesulitan dalam melakukan proses pencarian obat. Dengan demikian, sangat dibutuhkan sistem informasi persediaan obat yang menerapkan metode pencarian tercepat guna mendukung petugas apotek dalam meningkatkan kegiatan operasionalnya serta memudahkan proses pencarian obat sesuai dengan kebutuhan pelanggan. Hasil penelitian ini menunjukkan bahwa algoritma Boyer Moore lebih cepat dalam melakukan proses pencarian dengan rata-rata total running time 0.265 ms, dibandingkan dengan SQL Query dan algoritma Brute Force dengan hasil running time 0.271 ms dan 0.278 ms.*

**Kata Kunci:** Pencarian, String Matching, Boyer Moore, SQL Query, Apotek.

## Abstract

*String matching is a method that matches patterns with text. In string matching, several algorithms exist, such as the Brute Force algorithm and the Boyer-Moore algorithm. The Brute Force algorithm performs character matching starting from the left pattern, while character matching starts from the right pattern in the Boyer-Moore algorithm. In addition to using these methods, there is also SQL Query, a series of several basic SQL commands for performing searches. These three methods have different ways of working. For this reason, this study will conduct a comparative analysis of the performance of the three methods. The case study focuses on the search for drugs at Erha Farma Pharmacy. Since the mechanism for recording drug supplies in pharmacies still implements a conventional system, obstacles arise where pharmacy staff has difficulty carrying out the drug search process. Thus, a drug supply information system is needed that applies the fastest search method to support pharmacy staff in improving their operational activities and facilitating the drug search process according to customer needs. This study showed that Boyer Moore's algorithm was faster in carrying out the search process with an average total running time of 0.265 ms, compared to SQL Query and the Brute Force algorithm with running time results of 0.271 ms and 0.278 ms.*

**Keywords:** Searching, String Matching, Boyer Moore, SQL Query, Pharmacy.

## PENDAHULUAN

*String matching* adalah pendekatan yang berfungsi dalam menemukan kecocokan antar dua *string* berbeda, yaitu *string pattern* dan *string text*. *String pattern* dan *string text* terdiri dari kumpulan karakter yang disebut dengan kata alfabet. Metode *string matching* pada bahasa Indonesia biasanya dikenal dengan metode pencocokan *string* [1]. Terdapat berbagai macam algoritma dalam pendekatan ini, yakni algoritma Boyer-Moore dan Brute Force.

Algoritma Brute Force merupakan sebuah algoritma pencocokan *string* di mana *pattern* dan teksnya dicocokkan antara 0 dan  $n-m$  guna menemui lokasi *pattern* di dalam teks. Proses pencocokan *pattern* algoritma ini dalam teks tersebut dimulai dari karakter awal sampai pada karakter paling akhir. Apabila saat proses pencocokan karakter awal tidak ditemukan kecocokan, maka akan dilakukan proses pemindahan *pattern* ke karakter selanjutnya pada teks yang ada di sebelah kanan. Kegiatan ini akan terus berulang pada karakter berikutnya hingga ditemukan kesamaan karakter *pattern* dalam sebuah teks [2]. Penelitian sebelumnya yang telah menerapkan algoritma Brute Force mendapatkan hasil analisis dan perhitungan yang sangat baik dalam upaya memecahkan masalah [3].

Algoritma Boyer-Moore yang diciptakan oleh J.S Moore dan R.M Boyer merupakan satu dari algoritma pencarian yang

sangat berguna untuk mencari *string* pada teks. Algoritma ini menjalankan pencocokan diawali pada bagian karakter paling kanan lalu ke arah karakter bagian kiri, namun penggeseran jendela masih dilakukan dari bagian kiri lalu ke arah bagian kanan. Apabila terdapat sebuah kecocokkan antara karakter teks dengan karakter *pattern* terdahulu, maka akan dibandingkan dengan pengurangan indeks pada teks dan *pattern* dengan jumlah yang sama [4]. Penelitian terdahulu yang telah menerapkan algoritma Boyer-Moore ini menyatakan bahwa algoritma tersebut dapat dikatakan efektif berdasarkan hasil pengujian dengan nilai akurasi sebesar 99.41% [5].

Selain algoritma Brute Force dan Boyer-Moore, terdapat pula pencarian yang menerapkan metode SQL Query di mana tersusun atas beberapa perintah dasar SQL demi menemukan hasil pencarian yang sesuai dengan kebutuhan. Penelitian terdahulu yang menerapkan pencarian menggunakan SQL Query ini menyatakan bahwa dalam segi *running time*, SQL Query dinilai lebih efisien dibandingkan dengan algoritma pembandingnya [6].

Setiap algoritma memiliki kemampuan menerapkan metode pencarian yang berbeda dalam memenuhi kebutuhan aplikasi sistem informasi pencarian obat pada apotek. Dengan adanya perbedaan proses kinerja dari ketiga metode pencarian ini, maka sangat penting untuk melakukan upaya perbandingan kinerja atau performa dari ketiga metode tersebut demi menemukan

metode yang paling tepat untuk diterapkan ke dalam sistem pencarian obat di Apotek Erha Farma. Sistem pencatatan persediaan obat di Apotek Erha Farma yang masih dilakukan secara konvensional seringkali memunculkan berbagai kendala, salah satunya kesulitan para petugas dalam melakukan pencarian obat sesuai dengan permintaan pelanggan. Karena itu, diperlukanlah sistem informasi persediaan obat di Apotek Erha Farma yang terkomputerisasi oleh metode pencarian dengan *running time* tercepat demi memenuhi kebutuhan apotek tersebut.

Adapun tujuan penelitian ini adalah untuk menciptakan suatu aplikasi sistem informasi persediaan obat yang menerapkan metode pencarian tercepat guna mendukung petugas apotek dalam meningkatkan kegiatan operasionalnya serta memudahkan proses pencarian obat sesuai dengan kebutuhan pelanggan. Demi meminimalisir melebarnya permasalahan dalam penelitian, maka penulis memberi beberapa batasan masalah yang diterapkan pada sistem yang dibangun, di antaranya perancangan aplikasi pada penelitian ini hanya untuk perangkat berbasis web, aplikasi hanya mendukung transaksi penjualan obat, aplikasi hanya dapat dipakai dalam pencatatan persediaan obat, aplikasi hanya dapat dipakai oleh kasir untuk melakukan transaksi, mengelola data obat, mengelola pemasok, dan melihat riwayat transaksi. Sementara itu, untuk admin dapat melakukan semua aktivitas kasir serta dapat mengelola kasir. Dengan demikian, upaya

pembangunan sistem informasi persediaan obat ini diharapkan dapat mendukung kegiatan operasional apotek sejalan dengan penggunaan komputer pada perusahaan swasta yang juga turut berkembang, termasuk dalam bidang obat-obatan [7].

## METODE PENELITIAN

### *String Matching*

*String matching* atau biasa dikenal dengan algoritma yang mencocokkan teks dengan *pattern*. Teks adalah suatu *string* yang panjangnya diinisialkan dengan “*n*”. *Pattern* adalah suatu *string* yang panjangnya diinisialkan dengan “*m*”, yang dimana biasanya panjang karakter pada *pattern* lebih pendek dari panjang karakter pada teks. *String matching* adalah suatu alur untuk pencarian di mana pada setiap saat munculnya suatu *query* kemudian dihadapkan pada *pattern* yang digunakan untuk *string* yang lebih panjang. *String matching* merumuskan metodenya sebagai berikut [8].

$$x = x [0 \dots m - 1] \quad (1)$$

$$y = y \{0 \dots n - 1\} \quad (2)$$

Keterangan :

*x* = *pattern*

*y* = teks

*m* = panjang *pattern*

*n* = panjang teks

## Perancangan Penelitian

Pada Gambar 1 di bawah ini menjelaskan skema penelitian yang dilakukan dalam upaya membangun sistem informasi persediaan obat yang memenuhi kebutuhan di Apotek Erha Farma. Tahapan penelitian dimulai dengan pengumpulan data obat pada Apotek Erha Farma, lalu dilakukan perancangan sistem persediaan obat untuk apotek tersebut. Selanjutnya dilakukan implementasi dan pengujian *string matching* serta SQL Query dengan membandingkan serta menemukan algoritma terbaik dari tiga metode pencarian yakni Brute Force, Boyer-Moore, dan juga SQL Query untuk diterapkan ke dalam sistem informasi persediaan obat di Apotek Erha Farma.

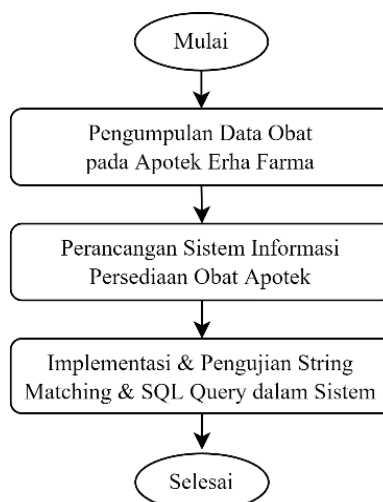
## Pengumpulan Data

Pengumpulan data obat pada penelitian ini dilakukan dengan cara melakukan proses

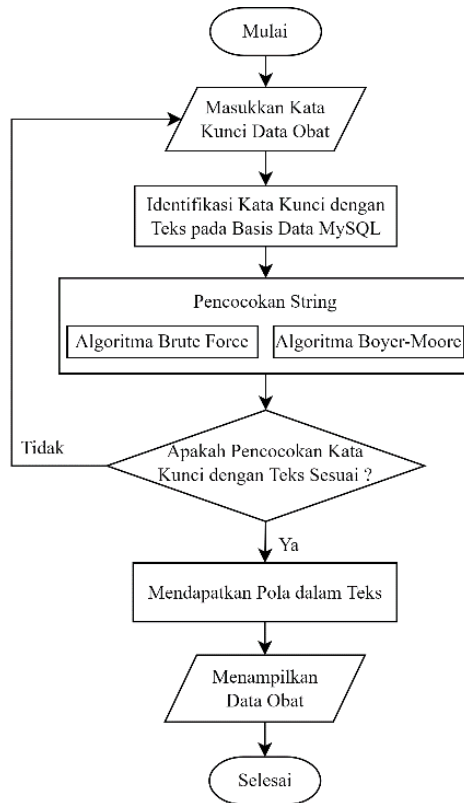
wawancara dengan petugas Apotek Erha Farma. Data tersebut nantinya digunakan untuk pengujian pada algoritma Boyer-Moore, Brute Force, dan SQL query dalam penelitian ini.

## Algoritma Brute Force, Boyer-Moore dan SQL Query

Sesudah proses pengumpulan data dilakukan, maka akan dilangsungkan perbandingan tiga metode pencarian, yaitu algoritma Brute Force, Boyer-Moore dan SQL Query terlebih dahulu demi menemukan metode mana yang terbaik guna diterapkan ke dalam sistem informasi persediaan obat yang dibangun. *Flowchart* sistem dari kedua metode *string matching* ini diilustrasikan pada Gambar 2.



Gambar 1. Skema penelitian



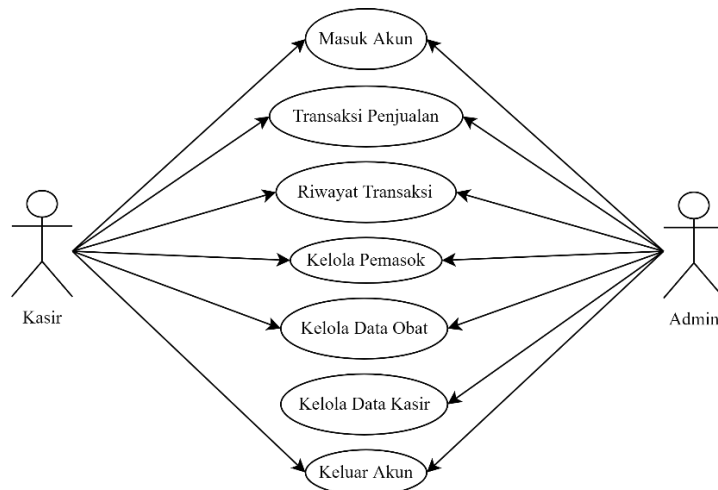
Gambar 2. Flowchart Brute Force dan Boyer-Moore

Pada Gambar 2 disajikan alur algoritma Brute Force dan Boyer-Moore terhadap proses pencarian teks. Proses pertama dimulai dengan *user* memasukkan kata kunci yang diinginkan untuk mencari data ke dalam *database*. Selanjutnya, dilakukan identifikasi kata kunci tersebut terhadap data pada *database*. Proses pencocokan *string* dilakukan dari bagian kanan ke bagian kiri untuk algoritma Boyer-Moore, sementara pada algoritma Brute Force proses pencocokan *string* dilakukan dari bagian kiri ke bagian kanan. Proses-proses tersebut akan terus dilakukan secara

berulang hingga semua data dalam *database* ditemukan pola atau kesamaan terhadap kata kunci dan teks yang dilakukan pencarian. Setelah ditemukan pola atau kesamaan dari kata kunci dan teks yang diidentifikasi, maka sistem akan menampilkan data obat yang dicari.

### Perancangan Sistem

*Use Case Diagram* adalah suatu ilustrasi *graphical* dari semua atau beberapa *actor*, *use case*, dan interaksi yang mengenalkan suatu sistem[11], ditunjukkan pada gambar 4 berikut ini.



Gambar 3. Use Case Diagram Kasir dan Admin

Gambar 3. Di atas menjelaskan fitur-fitur kasir yang bisa diakses, seperti melakukan *login* dengan akun yang telah dibuat oleh admin untuk masuk kedalam sistem. Setelah masuk, kasir akan langsung dapat melakukan proses transaksi penjualan obat, melihat riwayat transaksi, mengelola data obat, mengelola data pemasok, dan yang terakhir dapat melakukan *logout* pada sistem. Sedangkan fitur-fitur pada Admin yang dapat dijalankan, yaitu melakukan *login* dengan akun yang telah dibuat untuk masuk kedalam sistem, setelah masuk akan langsung dapat melakukan proses transaksi penjualan obat, selanjutnya dapat melihat riwayat transaksi, mengelola data obat, mengelola data kasir, dan yang terakhir dapat melakukan *logout* pada sistem.

Secara sistematis, berikut ini adalah tahapan-tahapan algoritma Brute Force dalam proses pencocokkan *string* [9]:

1. Brute Force memulai pencocokkan *pattern* di awalan teks.

2. Mulai karakter bagian kiri mengarah pada karakter bagian kanan, lalu terus mencocokkan karakter pada *pattern* dengan karakter pada teks, hingga satu dari persyaratan di bawah ini terwujud:

- a. Karakter pada *pattern* dan karakter pada teks saat diperbandingkan *mismatch*.
- b. Seluruh karakter pada *pattern* sama dengan karakter pada teks. Selanjutnya, algoritma akan menginformasikan posisi keberadaan teks.

3. Langkah selanjutnya yaitu melakukan pergeseran *pattern* sebanyak satu kali ke kanan, kemudian akan meneruskan tahapan yang kedua hingga *pattern* berada di bagian paling akhir teks.

Berikut ini merupakan contoh penggunaan Brute Force dalam pencarian *pattern* "MENTIN" dengan teks "AUGMENTIN":

Tabel 1. Langkah I Penggunaan Algoritma Brute Force

Langkah I							
Teks	A	U	G	M	E	N	T I N
Pattern	M	E	N	T	I	N	

Tabel 2. Langkah II Penggunaan Algoritma Brute Force

Langkah II							
Teks	A	U	G	M	E	N	T I N
Pattern		M	E	N	T	I	N

Tabel 3. Langkah III Penggunaan Algoritma Brute Force

Langkah III							
Teks	A	U	G	M	E	N	T I N
Pattern			M	E	N	T	I N

Tabel 4. Langkah IV Penggunaan Algoritma Brute Force

Langkah IV							
Teks	A	U	G	M	E	N	T I N
Pattern				M	E	N	T I N

Pada Tabel 1 disajikan tahapan pertama mencocokkan *pattern* dengan teks “AUGMEN”. Dapat dilihat bahwa M dan A *mismatch*, sehingga *pattern* akan melakukan pergeseran ke kanan sebanyak 1 kali.

Pada Tabel 2 disajikan tahapan kedua mencocokkan *pattern* dengan teks “UGMENT”. Dapat dilihat bahwa M dan U *mismatch*, sehingga *pattern* akan melakukan pergeseran ke kanan sebanyak 1 kali.

Pada Tabel 3 disajikan tahapan ketiga mencocokkan *pattern* dengan teks “GMENTI“, dapat dilihat bahwa M dengan G *mismatch*, sehingga *pattern* akan melakukan pergeseran ke kanan sebanyak 1 kali.

Pada Tabel 4 disajikan tahapan keempat mencocokkan *pattern* dengan teks “MENTIN“. Dapat dilihat bahwa seluruh

karakter yang ada di *pattern* cocok dengan seluruh karakter yang ada di teks, sehingga penggeseran *pattern* berhenti.

Secara sistematis, berikut adalah tahapan-tahapan Boyer-Moore dalam proses pencocokkan *string* [10] :

1. Boyer-Moore memulai pencocokkan *pattern* dari awalan teks.
2. Mulai karakter bagian kanan mengarah pada karakter bagian kiri, lalu terus mencocokkan karakter pada *pattern* dengan karakter pada teks, hingga satu dari persyaratan di bawah ini terwujud:
  - a. Karakter pada *pattern* dan karakter pada teks saat diperbandingkan *mismatch*.
  - b. Seluruh karakter pada *pattern* sama dengan karakter pada teks.

Selanjutnya algoritma akan menginformasikan posisi keberadaan teks.

3. Algoritma selanjutnya melakukan pergeseran *pattern* dengan memaksimalkan *value* pada geseran *good-suffix* dan juga geseran *bad-character*, kemudian akan meneruskan tahapan yang dilanjutkan dengan tahapan yang kedua hingga *pattern* sampai di bagian paling akhir teks.

Berikut ini merupakan contoh penggunaan Boyer Moore dalam pencarian *pattern* “TIN” dengan teks “FLADYSTIN”.

Pada Tabel 5 disajikan tahapan pertama mencocokkan *pattern* dengan teks “FLA”. Dapat dilihat bahwa N dan A *mismatch*, kemudian karakter A dengan karakter pada *pattern* “TIN” tidak ada yang

sama, sehingga *pattern* akan melakukan pergeseran sejumlah karakter *pattern* yang ada, yaitu sebanyak tiga langkah.

Pada Tabel 6 disajikan tahapan pertama mencocokkan *pattern* dengan teks “DYS”. Dapat dilihat bahwa N dan S *mismatch*, kemudian karakter S dengan karakter dipattern “TIN” tidak ada yang sama, sehingga *pattern* akan melakukan pergeseran sejumlah karakter *pattern* yang ada, yaitu sebanyak tiga langkah.

Pada Tabel 7 disajikan tahapan ketiga mencocokkan *pattern* dengan teks “TIN”. Dapat dilihat bahwa seluruh karakter yang ada pada *pattern* cocok dengan seluruh karakter yang ada di teks, sehingga penggeseran *pattern* berhenti.

Tabel 5. Langkah I Penggunaan Algoritma Boyer Moore

Langkah I									
Teks	F	L	A	D	Y	S	T	I	N
Pattern	T	I	N						

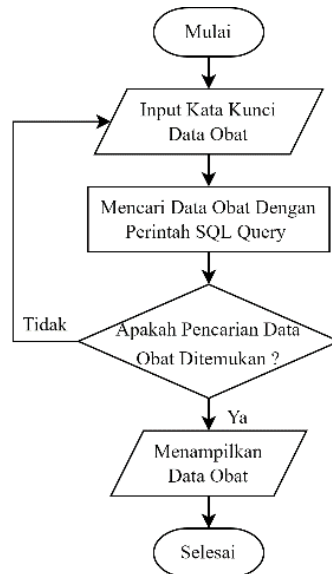
Tabel 6. Langkah II Penggunaan Algoritma Boyer Moore

Langkah I									
Teks	F	L	A	D	Y	S	T	I	N
Pattern				T	I	N			

Tabel 7. Langkah III Penggunaan Algoritma Boyer Moore

Langkah I									
Teks	F	L	A	D	Y	S	T	I	N
Pattern							T	I	N





Gambar 4. *Flowchart* SQL Query

SQL Query adalah rangkaian dari beberapa perintah dasar SQL agar mendapatkan hasil pencarian yang sesuai dengan yang ingin dicari [6]. *Flowchart* sistem dari SQL Query ini diilustrasikan pada Gambar 4.

Pada Gambar 4 dijelaskan alur SQL Query pada proses pencarian data obat. Proses pertama yaitu *user* memasukkan kata kunci yang diinginkan untuk dicarikan ke dalam *database*, Tahap pencarian ini dimulai dengan melakukan pengecekan *query* terhadap data yang sesuai dengan *database* menggunakan perintah dasar SQL. Setelah data obat ditemukan, maka sistem akan menampilkan data obat yang dicari.

## HASIL DAN PEMBAHASAN

### Hasil Pengujian Pencarian Nama Obat dengan Algoritma Brute Force

Pengujian ini dilaksanakan menggunakan algoritma Brute Force dengan sepuluh kata kunci. Dan dari tiap-tiap kata kunci yang ingin diuji akan melaksanakan *searching* nama obat yang berada di *database*. Lalu hasil pengujian digambarkan dalam Tabel 8 berikut ini.

Dari tabel 8 dijelaskan percobaan sebanyak sepuluh kali dalam mencari *pattern* pada teks (nama obat), lalu keseluruhan hasil rata-rata *running time* dalam pencarian obat diperoleh sebesar 0.278 ms.

Tabel 8. Hasil pengujian dengan algoritma Brute Force

Pengujian	Pattern										Total
	az	ui	bes	eco	asma	ycin	oxsan	xillin	zovirax	primperan	
1	0.032	0.026	0.025	0.036	0.027	0.035	0.036	0.016	0.028	0.013	0.274

Tabel 8. Hasil pengujian dengan algoritma Brute Force

Pengujian	Pattern										Total
	az	ui	bes	eco	asma	ycin	oxsan	xillin	zovirax	primperan	
2	0.025	0.023	0.029	0.032	0.034	0.032	0.034	0.040	0.040	0.028	0.317
3	0.013	0.012	0.030	0.028	0.015	0.033	0.030	0.033	0.033	0.031	0.258
4	0.037	0.025	0.015	0.036	0.025	0.023	0.026	0.027	0.034	0.035	0.283
5	0.025	0.021	0.038	0.013	0.036	0.034	0.026	0.028	0.040	0.036	0.297
6	0.038	0.029	0.033	0.038	0.026	0.024	0.031	0.033	0.035	0.024	0.311
7	0.029	0.030	0.026	0.017	0.038	0.034	0.032	0.039	0.041	0.038	0.324
8	0.024	0.025	0.010	0.029	0.024	0.028	0.029	0.025	0.022	0.032	0.248
9	0.018	0.023	0.033	0.009	0.021	0.026	0.022	0.031	0.023	0.025	0.231
10	0.030	0.010	0.032	0.027	0.024	0.021	0.026	0.023	0.026	0.025	0.244
Total Running Time (ms)											2787
Rata-rata Total Running Time (ms)											0.278

Tabel 9. Hasil pengujian dengan algoritma Boyer Moore

Pengujian	Pattern										Total
	az	ui	bes	eco	asma	ycin	oxsan	xillin	zovirax	primperan	
1	0.032	0.017	0.020	0.026	0.036	0.034	0.013	0.015	0.015	0.029	0.237
2	0.028	0.012	0.022	0.027	0.031	0.024	0.028	0.013	0.025	0.037	0.247
3	0.013	0.034	0.028	0.015	0.028	0.024	0.038	0.039	0.027	0.039	0.285
4	0.023	0.038	0.034	0.036	0.030	0.027	0.027	0.029	0.042	0.013	0.299
5	0.037	0.034	0.040	0.030	0.031	0.034	0.015	0.034	0.024	0.027	0.306
6	0.013	0.041	0.030	0.014	0.024	0.035	0.041	0.026	0.034	0.026	0.284
7	0.033	0.012	0.036	0.039	0.030	0.024	0.024	0.015	0.033	0.036	0.282
8	0.031	0.026	0.024	0.030	0.024	0.022	0.019	0.031	0.010	0.022	0.239
9	0.022	0.025	0.021	0.023	0.031	0.030	0.022	0.031	0.026	0.023	0.254
10	0.023	0.028	0.010	0.027	0.024	0.010	0.028	0.019	0.021	0.033	0.223
Total Running Time (ms)											2656
Rata-rata Total Running Time (ms)											0.265

### Hasil Pengujian Pencarian Nama Obat dengan Algoritma Boyer-Moore

Pengujian ini dilaksanakan menggunakan algoritma Boyer-Moore dengan sepuluh kata kunci. Dan dari tiap-tiap kata kunci yang ingin diuji akan melaksanakan *searching* nama obat yang berada di database.

Lalu hasil pengujian digambarkan dalam Tabel 9 berikut ini.

Dari tabel di atas dijelaskan percobaan sebanyak dua puluh kali dalam mencari *pattern* pada teks (nama obat), lalu keseluruhan hasil rata-rata *running time* dalam pencarian obat diperoleh sebesar 0.265 ms.

### Hasil Pengujian Pencarian Nama Obat dengan SQL Query

Pengujian ini dilaksanakan menggunakan SQL Query dengan sepuluh kata kunci. Dan dari tiap-tiap kata kunci yang ingin diuji akan melaksanakan *searching* nama obat yang berada di *database*. Lalu hasil pengujian digambarkan dalam Tabel 10.

Dari tabel 10 dijelaskan percobaan sebanyak dua puluh kali dalam mencari

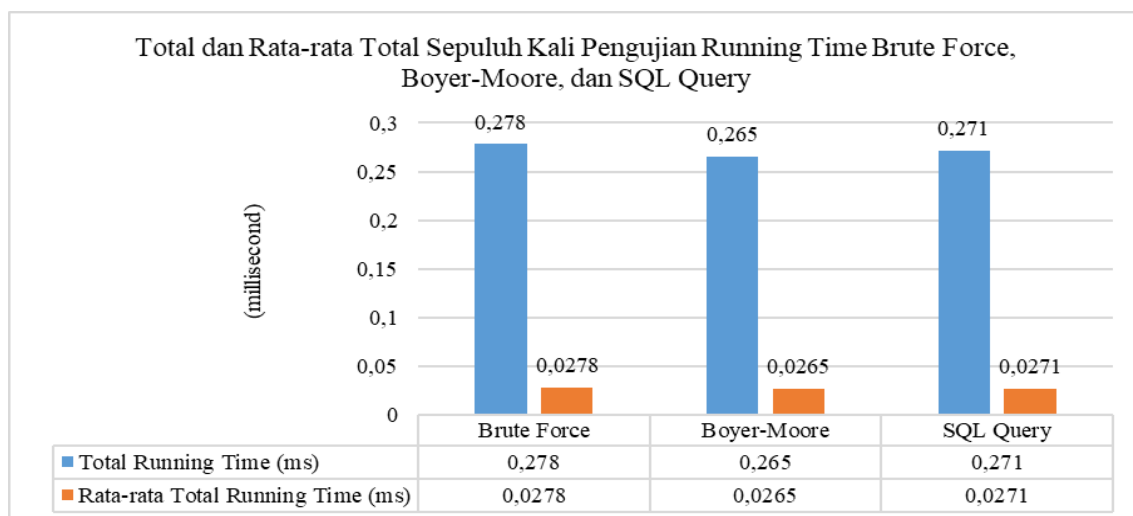
*pattern* pada teks (nama obat), lalu keseluruhan hasil rata-rata *running time* dalam pencarian obat diperoleh sebesar 0.271 ms.

### Analisa Perbandingan Metode String Matching dan SQL Query

Sesudah diuji menggunakan dua metode *string matching* dan SQL Query, maka hasil perbandingannya dapat diilustrasikan dalam Gambar 5.

Tabel 10. Hasil pengujian dengan SQL Query

Pengujian	Pattern										Total
	az	ui	bes	eco	asma	ycin	oxsan	xillin	zovirax	primperan	
1	0.013	0.012	0.027	0.027	0.038	0.034	0.014	0.032	0.038	0.024	0.259
2	0.024	0.025	0.029	0.023	0.025	0.011	0.039	0.034	0.015	0.032	0.257
3	0.011	0.027	0.033	0.016	0.035	0.036	0.015	0.014	0.026	0.040	0.253
4	0.031	0.030	0.027	0.024	0.027	0.029	0.036	0.015	0.014	0.037	0.270
5	0.026	0.023	0.034	0.029	0.035	0.024	0.032	0.030	0.037	0.033	0.303
6	0.036	0.023	0.015	0.031	0.040	0.030	0.032	0.022	0.041	0.040	0.310
7	0.031	0.035	0.037	0.012	0.037	0.037	0.025	0.014	0.024	0.032	0.284
8	0.027	0.030	0.028	0.018	0.028	0.022	0.024	0.024	0.034	0.020	0.255
9	0.020	0.026	0.020	0.024	0.034	0.031	0.020	0.027	0.031	0.023	0.256
10	0.025	0.030	0.029	0.030	0.024	0.022	0.029	0.020	0.025	0.030	0.264
Total Running Time (ms)											2711
Rata-rata Total Running Time (ms)											0.271



Gambar 5. Total dan rata-rata *running time* Brute Force, Boyer-Moore, dan SQL Query

Gambar 5 merupakan tampilan hasil *running time* pada pencarian nama obat menggunakan Brute Force, Boyer-Moore, dan SQL query, dari grafik tersebut terilustrasikan perbedaan dalam segi *running time* dari ketiga metode pencarian. Dan dapat dilihat jika algoritma Boyer Moore dari segi *running time* lebih cepat dari metode lainnya dengan menghasilkan total *running time* 0.265 ms, disusul dengan metode tercepat kedua yaitu SQL Query dengan total *running time* 0.271 ms, dan yang terakhir metode yang memiliki *running time* terlambat yaitu Brute Force dengan total *running time* 0.278 ms.

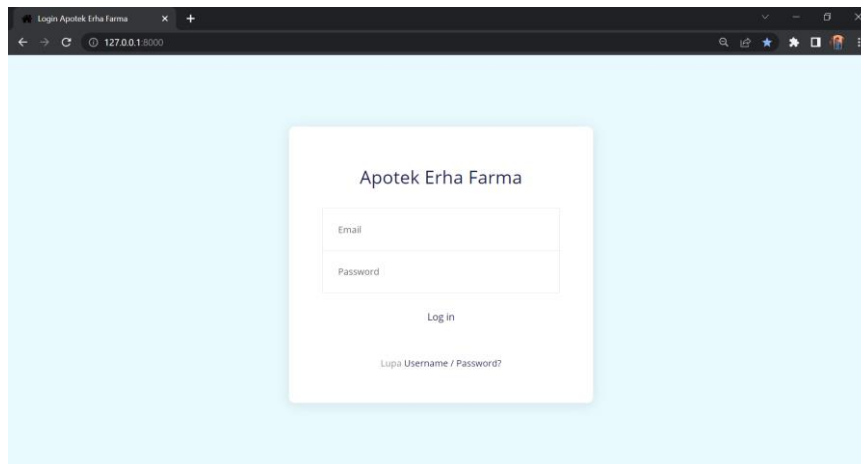
## Implementasi Sistem

### Halaman Login

Pada Gambar 6 merupakan halaman *login user*. Melalui halaman ini, *user* diminta untuk memasukkan *username* dan *password* sebelum mulai menggunakan *website*.

### Halaman Beranda




Pada Gambar 7 merupakan halaman tampilan *dashboard*, halaman ini merupakan halaman utama *user* setelah memasukkan *username* dan *password* yang benar pada halaman *login*.



Gambar 6. Halaman *Login*



Gambar 7. Halaman Beranda

Kode Obat	Foto Obat	Nama Obat	Detail	Jumlah Stok	Tambah Stok
10		AZIWIN 500 MG	<a href="#">Lihat Detail</a>	100	<a href="#">Ubah Stok</a>
11		AZMACON 4 MG	<a href="#">Lihat Detail</a>	103	<a href="#">Ubah Stok</a>
12		AZOPT E D	<a href="#">Lihat Detail</a>	100	<a href="#">Ubah Stok</a>

Gambar 8. Pengujian Metode Pencarian pada Halaman Data Obat

### Halaman Data Obat

Pada Gambar 8 merupakan halaman tampilan data-data obat. Pada halaman ini, kasir dapat melakukan transaksi penjualan, mencari obat, mengedit data obat, menambah stok obat, dan menghapus data obat. Pengujian metode pencarian yang menjadi fokus penelitian dilakukan pada halaman obat ini, di mana masing-masing metode dapat dilakukan pencarian terhadap kata kunci tertentu. Halaman web akan menampilkan total waktu yang diperlukan selama pencarian untuk menemukan metode pencarian mana yang paling efektif dalam memproses pencarian tercepat sesuai dengan kebutuhan pengguna.

### KESIMPULAN

Berdasarkan penelitian yang telah dilakukan dengan melakukan pengujian terhadap tiga metode pencarian dan membandingkan kinerja dari masing-masing metode, dapat dinyatakan bahwa *running time*

algoritma Boyer Moore bekerja lebih cepat dibandingkan kedua metode pencarian lainnya. Meskipun selisih dari masing-masing metode tidak begitu jauh, Boyer Moore memiliki performa yang unggul dengan total *running time* tercepat yaitu 0.265 ms. Lalu, metode tercepat kedua yaitu SQL Query dengan total *running time* 0.271 ms, sementara metode terakhir yang memiliki performa paling lambat yaitu Brute Force dengan total *running time* 0.278 ms. Dengan demikian, algoritma Boyer Moore menjadi metode pencarian yang paling tepat untuk diterapkan pada sistem informasi persediaan obat berbasis web pada Apotek Erha Farma yang masih menjalankan proses bisnisnya secara konvensional. Implementasi metode pencarian yang tepat ke dalam sistem informasi persediaan obat berbasis *website* pada Apotek Erha Farma tentunya akan sangat membantu memudahkan para petugas apotek pada saat mencari data obat yang sesuai dengan keinginan dan kebutuhan pelanggan. Sehingga, aplikasi sistem

informasi persediaan obat yang dibangun diharapkan dapat turut membantu proses kegiatan operasional apotek.

#### DAFTAR PUSTAKA

- [1] E. Mesi and D. Oktarina, "Penerapan Algoritma Horspool Pada Sistem Pendataan Obat Pada Apotek Fajar Mas," *Semin. Nas. Inform.*, hal. 79–85, 2021, [Online]. Available: <http://www.ejournal.pelitaindonesia.ac.id/ojs32/index.php/SENATIKA/article/view/1138>.
- [2] M. R. Azis, I. Fitri, and B. Rahman, "Penggunaan Algoritma Brute Force String Matching Dalam Pencarian Orang Hilang Pada Website Temukandia.Com," *JUPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 6, no. 2, hal. 205–212, 2021, doi: 10.29100/jupi.v6i2.1979.
- [3] A. K. Hadi, E. W. Ardhi, and I. T. Yuniarto, "Desain Konseptual dan Pola Operasi Fasilitas Kesehatan Apung di Wilayah Kepulauan: Studi Kasus Kepulauan Karimunjawa," *J. Tek. ITS*, vol. 10, no. 1, 2021, [Online]. Available: <http://ejurnal.its.ac.id/index.php/teknik/article/view/60207%0Ahttps://ejurnal.its.ac.id/index.php/teknik/article/download/60207/6609>.
- [4] M. Aulia, "Penerapan Algoritma Boyer Moore Untuk Pencarian Data Member Pada PT. Boenk Cosmetic Manufacture Berbasis Desktop," *J. Sist. Komput. dan Inform.*, vol. 1, no. 3, hal. 235, 2020, doi: 10.30865/json.v1i3.2139.
- [5] I. Algoritma, B. Moore, and J. T. Informatika, "SKRIPSI Oleh : ULUNG MUHAMMAD BESTARI," 2021.
- [6] D. Asmarajati, "Analisis Perbandingan Algoritma Tf-Idf Dengan Sql Query Untuk Kasus Pencarian Pada Sistem Informasi Dokumentasi Arsip (Sidokar)," *Device*, vol. 10, no. 1, hal. 1–8, 2020, doi: 10.32699/device.v10i1.1478.
- [7] D. Rusdianto, M. Kom, and A. Nurdesni, "Perancangan Sistem Informasi Persediaan Obat Berbasis Web Pada Apotek Andir Farma," *J. Sist. Informasi, J-SIKA*, vol. 02, no. Mdd, hal. 21–27, 2020.
- [8] G. F. H. Nainggolan, S. Andryana, and A. Gunaryati, "Pencarian Berita Pada Web Portal Menggunakan Algoritma Brute Force String Matching," *JUPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 6, no. 1, hal. 1–10, 2021, doi: 10.29100/jupi.v6i1.1824.
- [9] R. R. Khamdani, P. Astuti, and F. Masykur, "url : <http://studentjournal.umpo.ac.id/index.php/komputek> IMPLEMENTASI ALGORITMA BRUTE FORCE

- PADA PENCARIAN DATA KEPEMILIKAN TANAH,” hal. 101–109, 2021, [Online]. Available: <http://studentjournal.umpo.ac.id/index.php/komputek>.
- [10] C. Irawan and M. R. Pratama, “Perbandingan Algoritma Boyer-Moore dan Brute Force pada Pencarian Kamus Besar Bahasa Indonesia Berbasis Android,” *BIOS J. Teknol. Inf. dan Rekayasa Komput.*, vol. 1, no. 2, hal. 54–60, 2021, doi: 10.37148/bios.v1i2.13.
- [11] E. Hot, E. Sihombing, and I. K. Jaya, “Penerapan Model View Controller ( MVC ) Pada Perancangan Aplikasi Pencarian Pada Sistem Informasi Perpustakaan Berbasis Android Menggunakan Algoritma Boyer Moore ( Studi Kasus : Perpustakaan SMA Methodist 1 Medan ),” vol. 1, no. 2, hal. 52–59, 2021.