Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Implementasi suatu mesin pencarian pada mesin pencarian informasi pada *website* universitas yang pada penelitian ini difokuskan pada http://euis.umn.ac.id ini dilaksanakan dengan metode penelitian sebagai berikut :

1. Studi Literatur

Tahap awal dari pembangunan aplikasi ini yaitu dengan melakukan studi terhadap referensi-referensi mengenai algoritma yang digunakan dalam aplikasi yang dibangun, yaitu algoritma *string searching* Knuth Morris Pratt dan *Cache Based Query Processing* yang meng-*handle* ketika *server* mati dan *user* tidak bisa terhubung dengan *database* pusat. Referensi dapat berupa jurnal-jurnal teknik informatika, buku, maupun *website-website* yang terpercaya dan bisa dipertanggungjawabkan isinya.

2. Pengamatan Mesin Pencarian Yang Ada

Meneliti mesin pencarian informasi mahasiswa, dosen, dan karyawan yang sudah ada, dari segi cara kerja mesin pencarian, alur program, dan hasil yang dikembalikan oleh mesin pencari. Pengamatan juga dilakukan untuk mengetahui bagian atau aspek-aspek yang dapat dikembangkan.

3. Perancangan Aplikasi

Setelah referensi-referensi yang berkaitan dengan algoritma yang akan digunakan telah dipahami, perancangan aplikasi dimulai. Perancangan aplikasi

terbagi menjadi 2 tahap, tahap yang pertama adalah mengimplementasikan algoritma Knuth Morris Pratt untuk melakukan pencarian informasi pada *database* dan menampilkannya. Tahap yang kedua yaitu mengimplementasikan dan membangun memori *cache* yang akan digunakan oleh mesin pencarian. Aplikasi dirancang dengan bahasa pemrograman PHP sebagai bahasa pemrograman yang handal dalam *web programming*.

4. Pembuatan *Prototype* Aplikasi

Pada implementasinya, mesin pencarian ini mengakses *database* dan *localhost* dari http://euis.umn.ac.id untuk implementasi algoritma yang diteliti dalam penelitian ini pada mesin pencarian informasi. Namun sebelum diimplementasikan dengan localhost dan *database* dari http://euis.umn.ac.id, penelitian ini pertama-tama membangun semacam *prototype* dari mesin pencarian yang dibuat pada localhost penulis sendiri dan menggunakan *database* Oracle seperti yang digunakan oleh EUIS. *Prototype* ini dibuat sebagai tahap percobaan implementasi algoritma-algoritma yang digunakan dalam mesin pencarian informasi yang ingin dirancang.

5. Implementasi Aplikasi Pada http://euis.umn.ac.id

Setelah *prototype* berhasil dirancang dan dibuat, mulailah mesin pencarian informasi yang menggunakan algoritma Knuth Morris Pratt dan *Cache Based Query Processing* ini diimplementasikan langsung pada localhost EUIS dari Universitas Multimedia Nusantara dan mengakses langsung *database* dari *server* EUIS UMN.

6. *Testing*

Mesin pencarian yang sudah dibuat diuji apakah dapat berjalan sesuai yang dirancang dan dicek adanya *bug* atau *error* yang terjadi pada penggunaannya secara

nyata. *Bug* dan *error* yang ditemukan akan dievaluasi untuk kepentingan yang lebih baik lagi.

7. Evaluasi

Dari tahap *testing*, mesin pencarian informasi yang menggunakan algoritma Knuth Morris Pratt dan *Cache Based Query Processing* ini dievaluasi kinerjanya. Evaluasi ini berguna untuk menarik kesimpulan atas penelitian yang telah dijalankan dan untuk pengembangan selanjutnya ke arah yang lebih baik.

8. Penulisan Laporan

Setelah aplikasi selesai dievaluasi dan dinyatakan selesai maka laporan dibuat untuk memuat dokumentasi dan penjelasan mengenai perancangan program, metode-metode yang digunakan, proses pembuatan dan kode, dan hasil dari aplikasi yang dikembangkan dalam penelitian ini.

3.2 Perancangan

Aplikasi yang dibuat dalam penelitian ini dirancang sedemikian rupa dengan menggunakan *flowchart* dan struktur tabel sehingga dapat diimplementasikan dengan baik.

3.2.1 Algoritma Knuth Morris Pratt

Algoritma Knuth Morris Pratt pada mesin pencarian ini digunakan sebagai algoritma pencocokan *string* atau *string matching algorithm* yang berfungsi untuk mencari *string* yang sama dengan input atau *query user* dengan data-data yang ada di dalam *database*.

Algoritma Knuth Morris Pratt yang akan diimplementasikan pada mesin pencarian informasi ini disimpan ke dalam suatu php file sendiri yang bernama

kmp.php. Isi **kmp.php** ini berupa suatu PHP *class* yang bernama KMP yang memiliki 2 fungsi yaitu fungsi KMPSearch dan preKMP dimana fungsi KMPSearch merupakan fungsi utama yang akan mencari kecocokan kata yang dicari dengan data yang ada pada *database* dan fungsi preKMP merupakan fungsi pendukung agar KMPSearch dapat berjalan dengan baik.

Cara kerja fungsi KMPSearch ini yaitu menerima 2 parameter yang berupa *string*, parameter pertama merupakan *string* yang dicari, berikutnya disebut *pattern*/pola, dan parameter kedua merupakan teks yang ada pada *database* yang ingin dicari kecocokannya. Kedua parameter yang diterima ini sebagai *string* lalu diubah menjadi *array of characters*. Setelah *string* diubah menjadi suatu *array*, fungsi preKMP dijalankan untuk mendapatkan suatu tabel lompatan yang akan digunakan sebagai acuan pergerakan kursor pencocokan *string*.

Fungsi preKMP akan mengembalikan suatu nilai berupa *array*. Setelah nilai dari fungsi preKMP didapatkan, maka isi fungsi KMPSearch dilanjutkan. Iterasi pencarian pola pada teks yang ada dilakukan selama jumlah iterasi belum melebihi jumlah *array* pada teks yang bila sudah melebihi artinya pencarian sudah mencapai batas akhir teks yang ada. Fungsi KMPSearch akan mengembalikan suatu nilai boolean apakah suatu pola ditemukan pada teks yang ada. Bila ya, maka variabel boolean tersebut akan bernilai *true*, bila tidak maka akan bernilai *false*

Flowchart cara kerja algoritma ini dijelaskan lebih detail pada gambar 3.3 yang menjabarkan *step by step* yang dilakukan algoritma Knuth Morris Pratt dalam melakukan pencocokan *string*.

3.2.2 Cache Based Query Processing

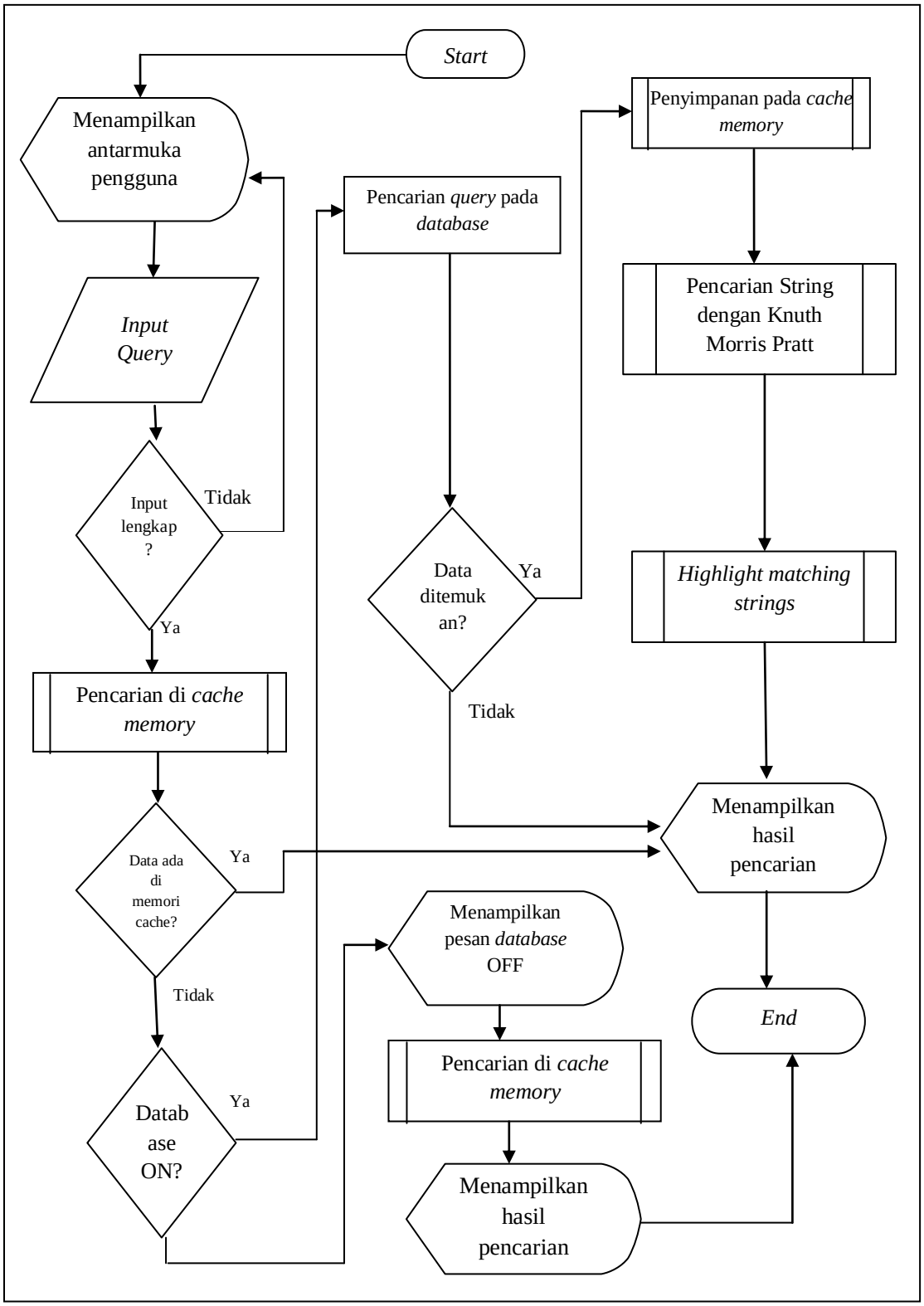
Cache Based Query Processing merupakan algoritma yang melakukan pencariannya bukan pada *database* namun pada suatu *result cache* yang telah dibuat. Mesin pencarian yang dibuat ini mengimplementasikan metode *Cache Based Query Processing* saat *database* sedang dalam kondisi mati/*down* sehingga tidak bisa diakses oleh pengguna.

Pada penelitian ini, mesin pencarian merupakan suatu mesin yang mencari suatu data yang *exact* sesuai dengan yang diketikkan oleh pengguna. Misalnya ketika pengguna mengetikkan pencarian "andre rusli", maka pengguna ingin mencari informasi dari data yang mengandung *string* "andre rusli" tersebut. Oleh karena itu, saat kondisi *database* sedang menyala/aktif, setiap hasil *query* yang didapat dari *database* akan disimpan ke dalam suatu memori *cache*. Memori *cache* menampung maksimal 1000 data didalamnya, ketika data ke 1001 atau lebih ingin disimpan, maka data yang paling lama akan dibuang dan data yang baru dimasukkan.

Dengan kata lain, memori *cache* menyimpan 1000 hasil pencarian terbaru yang didapatkan oleh mesin pencari. Langkah-langkah yang dilakukan oleh metode *cache based query processing* ini dijelaskan lebih detail dalam bentuk *flowchart* pada gambar 3.4.

3.2.3 Flowchart Sistem

Berikut merupakan *flowchart* dari alur jalannya mesin pencarian yang dibuat pada penelitian ini. *Flowchart* ini menjelaskan proses kerja mesin pencarian baik saat *database* nyala maupun ketika sedang mati.



Gambar 3.1 *Flowchart* Mesin Pencarian

Pertama-tama, tampilan antarmuka pengguna menampilkan halaman mesin pencarian kepada *user*. Lalu *user* diminta untuk memasukkan *query*/*input* yang ingin dicari hasilnya, dan juga memasukkan tipe data yang ingin dicari (mahasiswa atau dosen/karyawan). Bila input belum lengkap, maka *user* diminta untuk mengisi ulang data secara lengkap dan sistem menampilkan pesan kesalahan yang terjadi. Bila sudah lengkap, maka sistem akan mengecek *database* apakah sedang ON atau OFF.

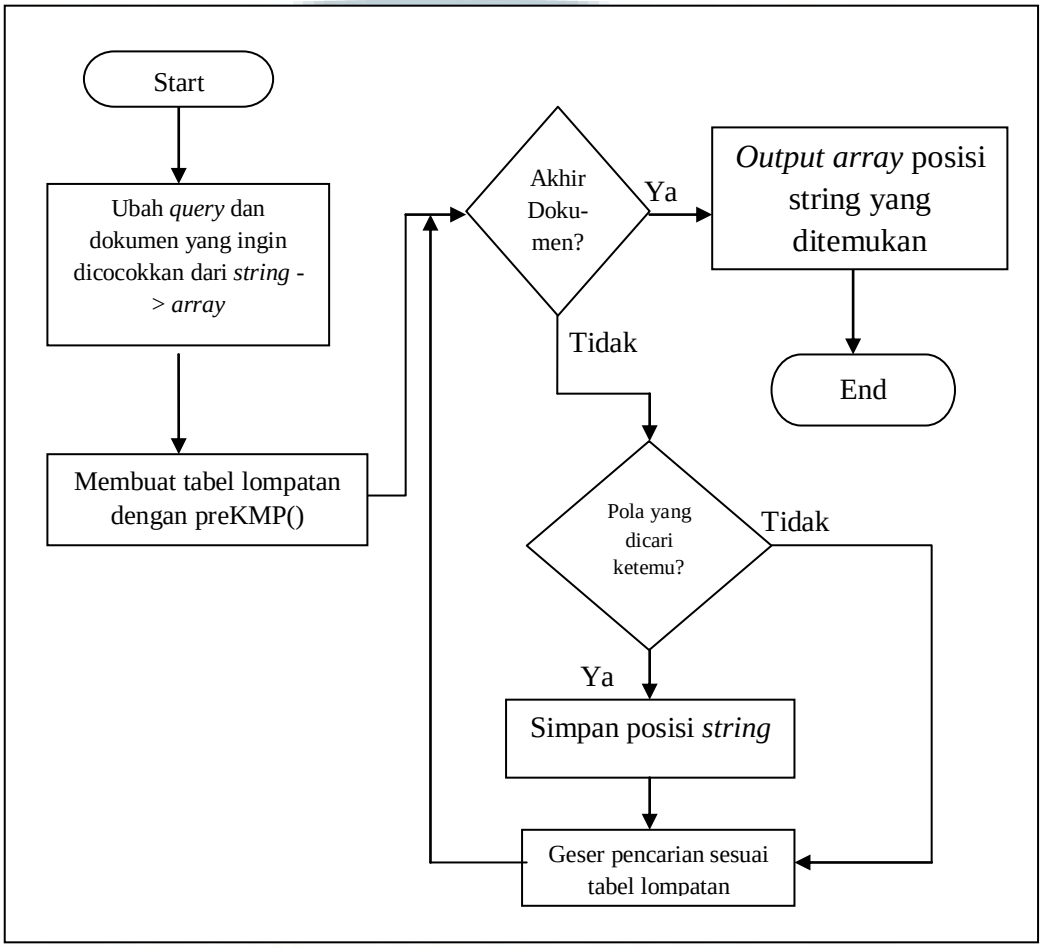
Saat kondisi *database* ON, maka sistem akan melakukan pencarian di *database* langsung dengan menggunakan *query database*. Bila hasil ditemukan maka hasil pencarian akan disimpan ke dalam memori *cache* dan lalu dari hasil yang ditemukan, dengan algoritma Knuth Morris Pratt akan dicari string yang mengalami kecocokan lalu dihighlight dengan warna yang berbeda, lalu setelah itu, hasil akan ditampilkan ke *user*, bila tidak ada maka akan ditampilkan pesan bahwa data yang dicari tidak ada.

Bila *database* atau *backend server* sedang tidak bisa diakses karena satu alasan atau yang lainnya, maka sistem akan melakukan pencarian pada *result cache* yang telah dibuat. Dan akan mengembalikan hasil yang tersimpan pada *cache* dan juga pemberitahuan bahwa saat ini kondisi *database* sedang OFF sehingga data diambil dari *result cache.*

3.2.4 Flowchart Pencarian String Dengan Knuth Morris Pratt

Setelah hasil didapatkan dari *database* sesuai yang di-*input* oleh pengguna dan hasil yang ditemukan dimasukkan ke dalam memori *cache*, mesin pencari lalu mencari *string* yang cocok pada data-data tersebut dengan menggunakan algoritma Knuth Morris Pratt yang setelah dicocokkan dengan *input user* lalu diberi warna

berbeda untuk mengindikasikan kesesuaian hasil yang didapat. Gambar 3.2 menjelaskan bagaimana algoritma Knuth Morris Pratt berjalan pada mesin pencari yang dikembangkan.



Gambar 3.2 *Flowchart* Pencarian Knuth Morris Pratt Pada Memori *Cache*

Pada proses ini, sistem membaca seluruh hasil pencarian yang didapat dari *database*. Setelah semua data dibaca, maka pencarian dimulai baris per baris. *String* yang dicari dan *string* pada dokumen yang ada diubah menjadi bentuk *array of characters.* Lalu dibuat tabel lompatan untuk *string* yang dicari sebagai tahap awal

algoritma Knuth-Morris-Pratt. Lalu *query* tersebut dicocokkan dengan data yang ada pada dokumen yang diambil dari *database*.

Setelah dicocokkan, dicek apakah pencarian sudah mencapai akhir dokumen, bila tidak, maka pencarian dilanjutkan. *Query* yang diinput pengguna akan dicocokkan karakter per karakter dengan hasil pencarian yang ada, bila karakter yang serupa ditemukan, maka pergeseran pencarian bergeser sesuai tabel lompatan, bila belum ditemukan maka pencarian berlangsung normal dan bergeser satu karakter ke kanan. Bila sudah mencapai akhir dokumen, *matching strings* yang ditemukan pada hasil pencarian tersebut di-*highlight* warna yang berbeda dengan teks normal, untuk mengindikasikan kesesuaian pencarian yang dilakukan.

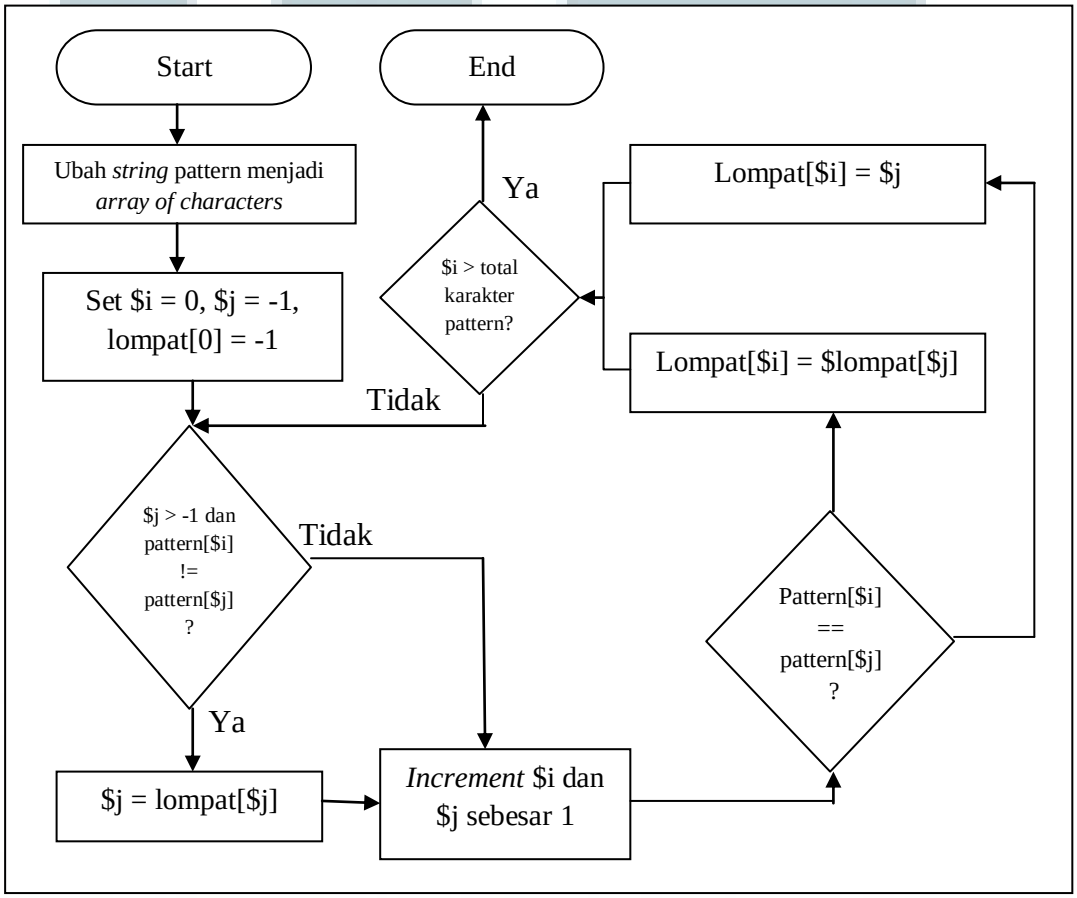
3.2.5 Flowchart Pembuatan Tabel Lompatan preKMP()

Tabel lompatan dibuat untuk menentukan pergeseran iterasi pencocokan *string* pada input pengguna dan *string* pada *database* yang ada. Fungsi preKMP() yang bertujuan untuk membentuk tabel lompatan tersebut membutuhkan satu parameter yaitu *pattern* atau input user yang ingin dicari kecocokannya dengan dokumen yang ada pada *database*.

Pertama-tama *pattern* dipecah menjadi *array of characters*, setelah itu selama iterasi belum melebihi jumlah karakter yang ada pada *pattern* suatu *array* yang menyimpan daftar posisi lompatan/pergeseran pencarian dibentuk. Berikut program/*code* dari proses yang diterapkan pada fungsi preKMP() :

```
function preKMP($pattern) {
   $i = 0;
   $j = $lompat[0] = -1;
   while($i<count($pattern))
   {
```

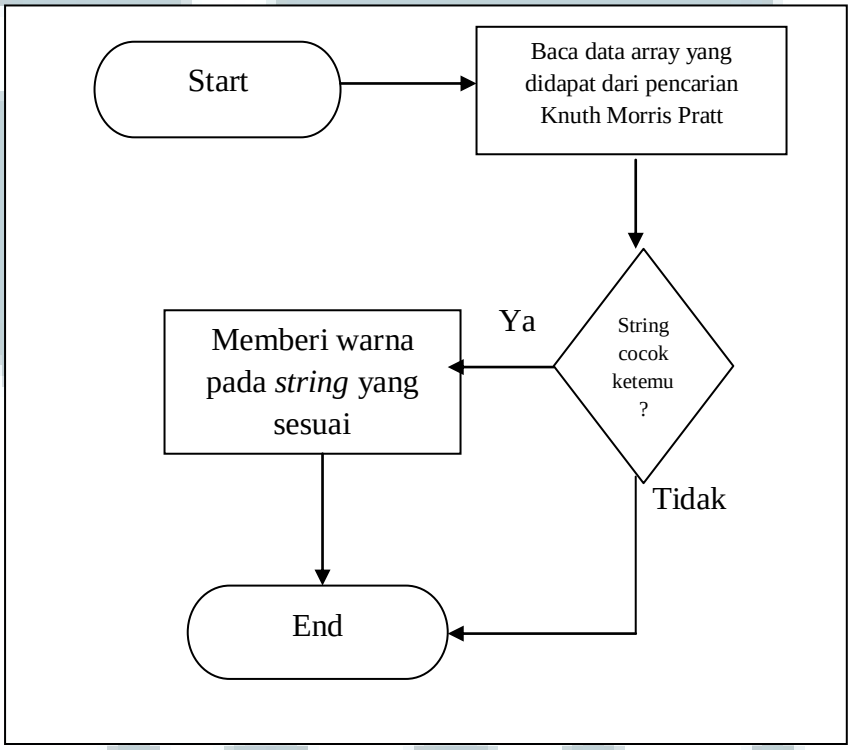
```
            while($j>-1 && $pattern[$i]!=$pattern[$j])
            {
                    $j = $lompat[$j];
             }
             $i++;
            $j++;
            if($pattern[$i]==$pattern[$j])
            {
                    $lompat[$i]=$lompat[$j];
             }
            Else
            {
                    $lompat[$i]=$j;
             }
        }
    return $lompat; }
```



Gambar 3.3 *Flowchart* Pembuatan Tabel Lompatan

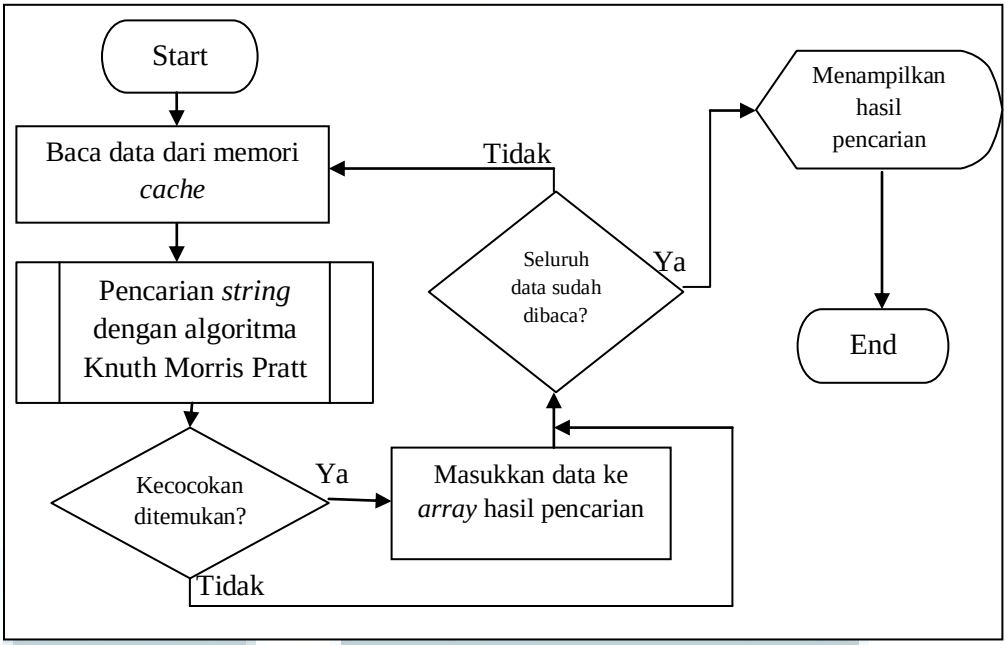
3.2.6 Flowchart Highlight Matching Strings

Output yang dikembalikan oleh proses pencarian dengan algoritma Knuth Morris Pratt yaitu *array* yang menunjukkan posisi *string* yang cocok dengan *input* yang dicari pada dokumen yang ada diterima dan dibaca datanya. Setelah itu berdasarkan posisi yang didapat, sebelum ditampilkan *string* hasil pencarian diberi warna berbeda untuk *pattern* yang sesuai dengan yang dicari oleh pengguna.



Gambar 3.4 *Flowchart Highlight Matching Strings*

3.2.7 Flowchart Pencarian Pada Cache Memory

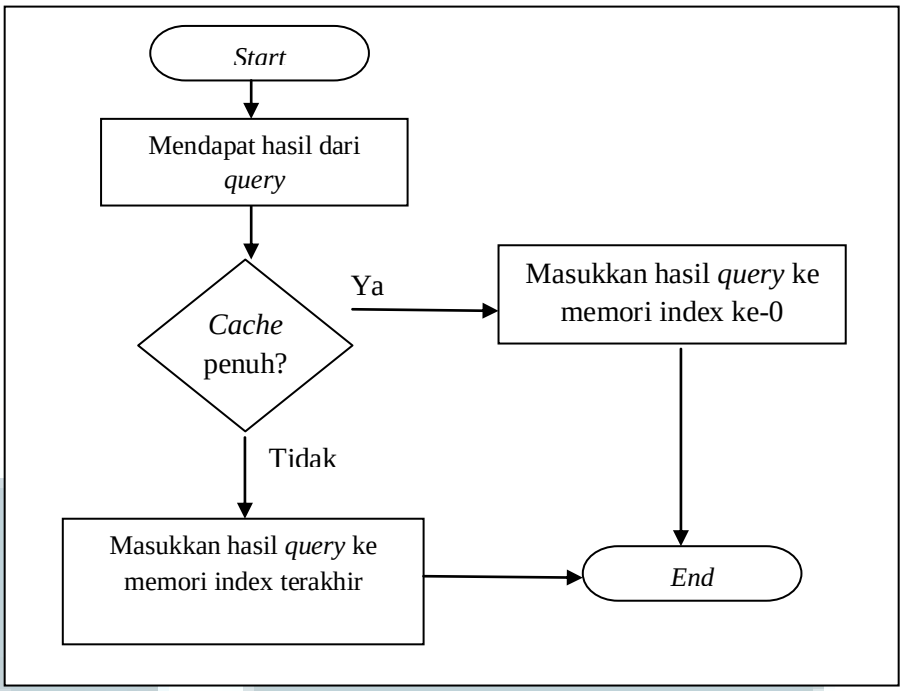


Gambar 3.5 *Flowchart* Pencarian Pada *Cache Memory*

Saat *database* dalam kondisi mati atau *down* sehingga tidak bisa diakses oleh pengguna, maka pencarian dilakukan terhadap data-data yang ada pada memori *cache*. Pencocokan *string* pada memori *cache* dilakukan dengan algoritma Knuth Morris Pratt.

Sistem mengecek apakah sudah akhir dari dokumen apa belum, bila belum bila ditemukan kecocokkan karakter maka pencarian bergeser sesuai tabel lompatan, bila tidak maka bergeser satu karakter ke kanan. Sedangkan bila sudah akhir dari semua dokumen maka lalu sistem akan menampilkan hasil pencarian, bila tidak ditemukan maka akan ditampilkan pesan bahwa data yang dicari tidak ditemukan.

3.2.8 Flowchart Penyimpanan Pada Cache Memory



Gambar 3.6 *Flowchart* Penyimpanan Pada *Cache Memory*

Setiap kali ada data pencarian yang berhasil ditemukan dari pencarian *query* pada *database*, maka hasil pencarian tersebut akan disimpan di suatu memori yang disebut dengan *result cache*. Prosesnya yaitu ketika ada *string* yang berhasil ditemukan, maka hasilnya akan diambil dan disimpan ke dalam memori *cache*. Sebelum disimpan, dicek terlebih dahulu apakah memori *cache* sudah penuh atau belum.

Bila memori *cache* sudah penuh maka data akan disimpan di tempat memori *cache* yang paling awal atau paling lama, yaitu pada index ke-0. Bila memori *cache* belum penuh maka data akan disimpan di index terakhir yang masih kosong.

3.2.9 Struktur Tabel

Pada penelitian ini, tabel yang digunakan merupakan hasil view dari tabel-tabel yang ada di dalam *database* Universitas Multimedia Nusantara. Oleh karena itu, untuk keperluan penelitian ini hanya digunakan 2 tabel yaitu tabel master_mahasiswa yang mengandung data-data mahasiswa dan tabel master_karyawan yang mengandung data-data dosen dan karyawan UMN. Adapun kedua tabel tersebut tidak memiliki keterkaitan satu sama lain sehingga sistem yang dibuat ini tidak memiliki *entity relationship diagram.* Berikut merupakan struktur tabel yang digunakan dalam penelitian ini

A. Tabel Mahasiswa

Nama Tabel : master_mahasiswa.

Fungsi : menyimpan data NIM, Nama, dan Jurusan setiap mahasiswa.

Primary Key : NIM

Foreign Key : (-)

Tabel 3.1 Tabel Mahasiswa

Nama Field	Tipe Data	Keterangan
NIM	Varchar(11)	Nomor Induk Mahasiswa
NAMA	Varchar(50)	Nama Lengkap Mahasiswa
JURUSAN	Varchar(200)	Program Studi

B. Tabel Dosen dan Karyawan

Nama Tabel : master_karyawan.

Fungsi : menyimpan data ID, NIK, Nama, dan Jabatan milik karyawan.

Primary Key : ID

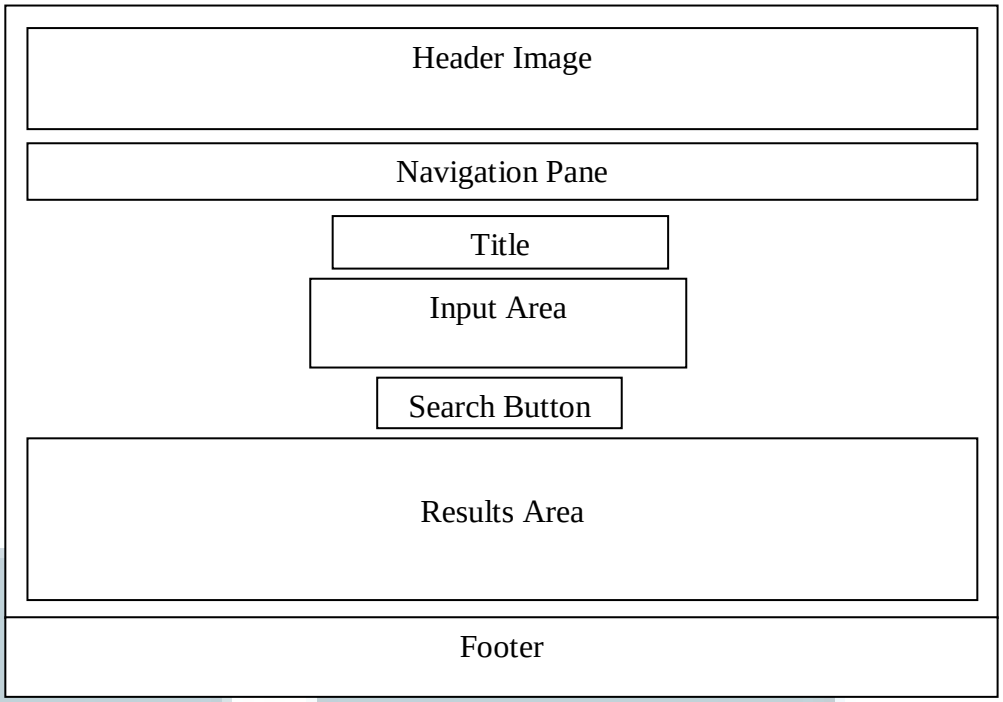
Foreign Key : (-)

Tabel 3.2 Tabel Dosen dan Karyawan

Nama Field	Tipe Data	Keterangan
ID	Integer	ID setiap karyawan
NIK	Varchar(11)	Nomor Induk Karyawan
NAMA	Varchar(50)	Nama Lengkap Karyawan
JABATAN	Varchar(200)	Jabatan Yang Dimiliki

3.2.10 Perancangan Antarmuka Pengguna

Pada halaman mesin pencarian informasi mahasiswa, dosen, dan karyawan pada EUIS milik Universitas Multimedia Nusantara ini terdapat satu halaman antarmuka dimana *user* dapat memasukkan *query* atau input teks yang ingin dicarinya dan di halaman itu juga hasilnya akan ditampilkan. Pada gambar 3.5 ditampilkan rancangan antarmuka pengguna aplikasi.



Gambar 3.5 Rancangan Antarmuka Pengguna

Pada antarmuka halaman pencarian mahasiswa, dosen, dan karyawan yang dibangun ini terdapat 7 area utama yang diletakkan sesuai ketentuan rancangan di atas. Di paling atas terdapat *header image* yaitu merupakan gambar utama EUIS Universitas Multimedia Nusantara yang ada di setiap halaman web EUIS. Di bawah *header image*, terdapat *navigation pane* yang juga ada di setiap halaman *website* EUIS UMN yang berfungsi untuk berpindah ke halaman-halaman lain yang esensial pada http://euis.umn.ac.id.

Di bawah *navigation pane*, ada judul dari halaman ini yang menjelaskan bahwa halaman ini merupakan halaman tersedianya mesin pencarian informasi mahasiswa, dosen, dan karyawan dari UMN. Lalu tepat di bawahnya, terdapat area input yang dimana *user* dapat memasukkan tipe pencarian yang diinginkan. Tipe

tersebut dibagi menjadi dua tipe pencarian yaitu pencarian informasi mahasiswa dan dosen/karyawan. Setelah memilih tipe yang diinginkan, tersedia *textbox* yang dapat diisi oleh *user* untuk memasukkan *query* atau teks yang ingin dicari pada mesin pencarian tersebut.

Bila semua input yang dibutuhkan sudah terisi, *user* hanya perlu menekan tombol pencarian untuk memerintahkan sistem untuk melakukan pencarian di *database* dan mengembalikan apa yang ditemukan maupun bila informasi yang dicari tidak ditemukan oleh sistem. Dan di bagian paling bawah ada *footer* yang menyatakan informasi *website.*