BAB III

METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian untuk mengimplementasikan algoritma Rabin-Karp dan *Confix-Stripping* ini adalah sebagai berikut.

1. Studi Literatur

   Pada tahap ini akan dilakukan studi literatur dengan menggunakan berbagai sumber yang ada, baik dari media cetak, jurnal, atau *website*, guna menambah pengetahuan dan pemahaman dalam penelitian, sehingga menghasilkan penelitian yang baik dan *valid*.

2. Perancangan Sistem

   Pada tahap perancangan aplikasi, diawali dengan perancangan sistem atau proses seperti *flowchart*, kemudian dilanjutkan dengan perancangan struktur tabel serta desain *user interface* ataupun fitur-fitur lainnya yang akan digunakan pada aplikasi tersebut.

3. Pengumpulan Data

   Penelitian ini menggunakan data berupa dokumen yang didapatkan melalui *website* yang menyediakan dokumen-dokumen resmi. Selain itu data yang didapatkan akan dimodifikasi sedemikian rupa untuk mendapatkan data tekstual dan disimpan dalam bentuk dokumen berekstensi .pdf, .docx ataupun .txt untuk memenuhi batasan masalah penelitian ini.

4. Implementasi

Aplikasi akan dibuat menggunakan bahasa pemrograman PHP. Selain itu *database engine* yang akan digunakan adalah MySQL. Implementasi akan dilakukan berdasarkan pertimbangan-pertimbangan yang ada pada tahap perancangan.

5. Evaluasi

   Pada tahap ini akan dilakukan evaluasi hasil dari implementasi algoritma Rabin-Karp dan *stemming Confix-Stripping* terhadap performa sistem, baik dari segi waktu maupun tingkat *similarity*.

6. Dokumentasi

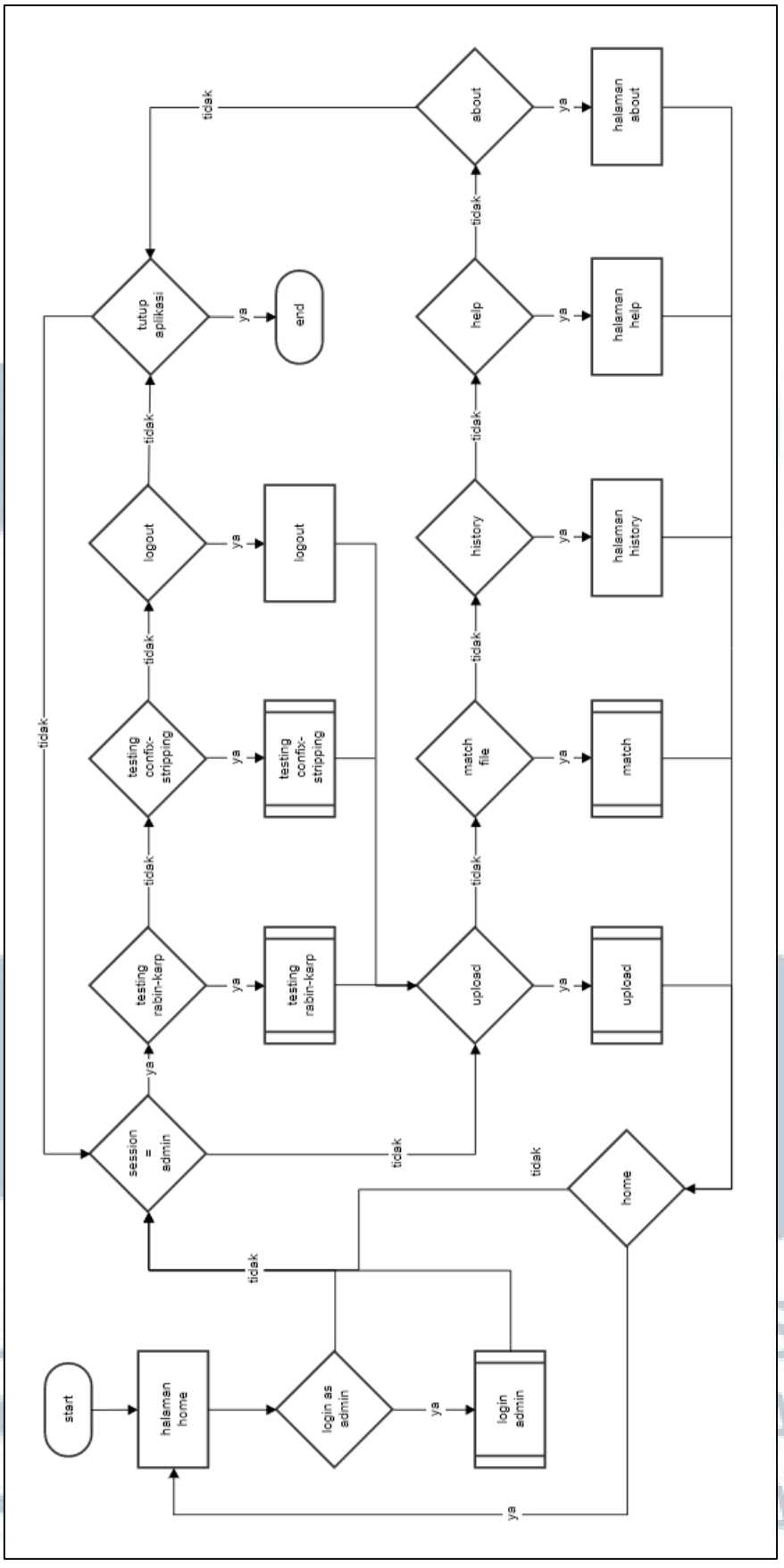
   Penulisan laporan akan dilakukan sebagai bentuk dokumentasi dari seluruh tahap yang dilakukan dalam penelitian.

3.2 Perancangan Aplikasi

Dalam tahap perancangan sistem, perancangan yang dilakukan dalam penelitian ini antara lain *flowchart*, struktur tabel, dan desain antarmuka.

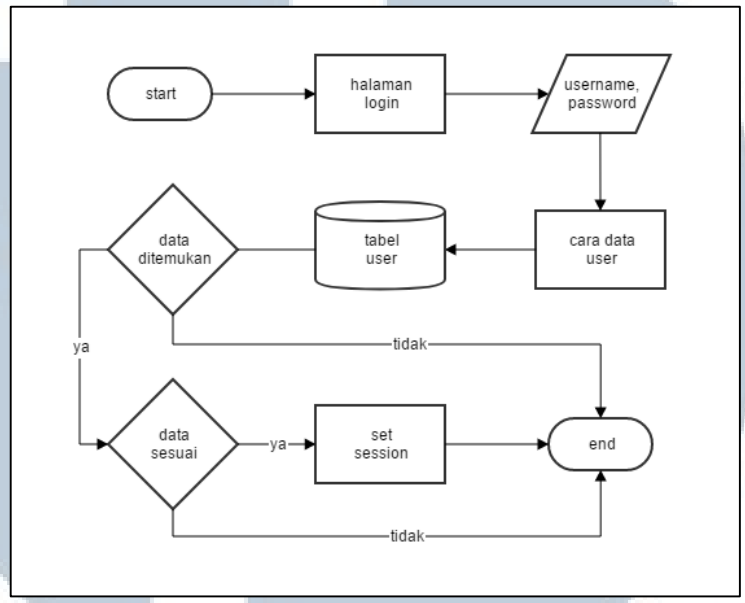
3.2.1 Flowchart

Pada Gambar 3.1 digambarkan *flowchart* sistem secara keseluruhan. Sistem memiliki dua tipe *user* yaitu *guest* dan *admin.* Saat pertama kali masuk ke dalam aplikasi, *user* akan dianggap sebagai *guest* terlebih dahulu dan masuk ke halaman Home yang berfungsi untuk mengunggah dokumen. Selain halaman tersebut, *user guest* juga dapat mengakses halaman Login as Admin, Upload, Match, History dan Help, sedangkan *user admin* dapat mengakses halaman tambahan untuk melakukan *testing* algoritma Rabin-Karp dan *Confix-Stripping*.



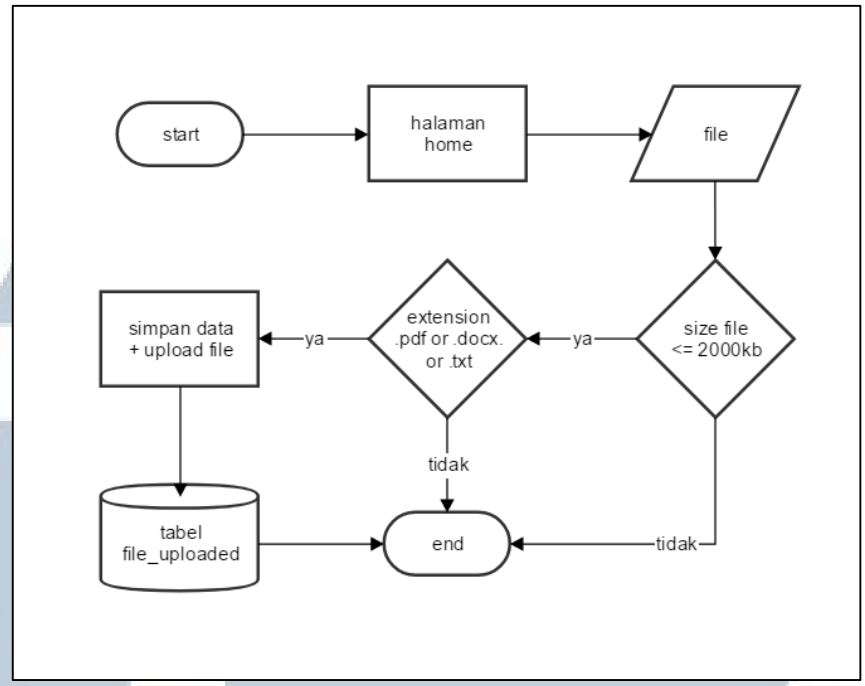
Gambar 3.1 Flowchart Keseluruhan Sistem

Flowchart yang digambarkan pada Gambar 3.1 memiliki lima subproses yang akan dijelaskan lebih rinci didalamnya. Subproses tersebut antara lain *flowchart login admin*, *upload*, *match file*, *testing* Rabin-Karp, dan *testing Confix-Stripping.*



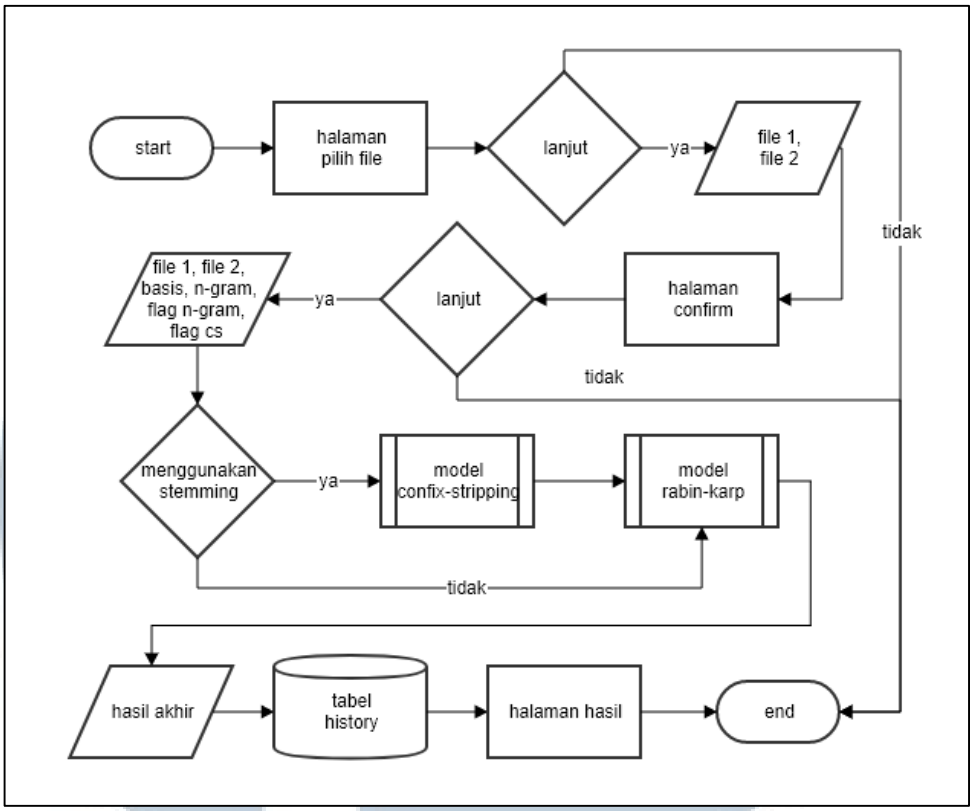
Gambar 3.2 Flowchart *Login* Admin

Dalam *flowchart* proses *login*, sistem melakukan pengecekan terhadap *username* dan *password*. *User* akan melakukan *input* data *username* dan *password* pada halaman Login. Jika data *username* dan *password* ditemukan di dalam tabel *user* dan sesuai, maka proses *login* berhasil dan sistem akan melakukan *set session* sebagai *admin* terhadap *user* tersebut. Namun apabila data *username* maupun *password* yang di-*input* tidak ditemukan di dalam tabel *user*, proses *login* gagal dan sistem tidak akan melakukan *set session* sebagai *admin* terhadap *user* tersebut, *user* akan dianggap sebagai *guest* oleh sistem dan tidak dapat mengakses menu tambahan yang dimiliki oleh *user* dengan *session admin.*



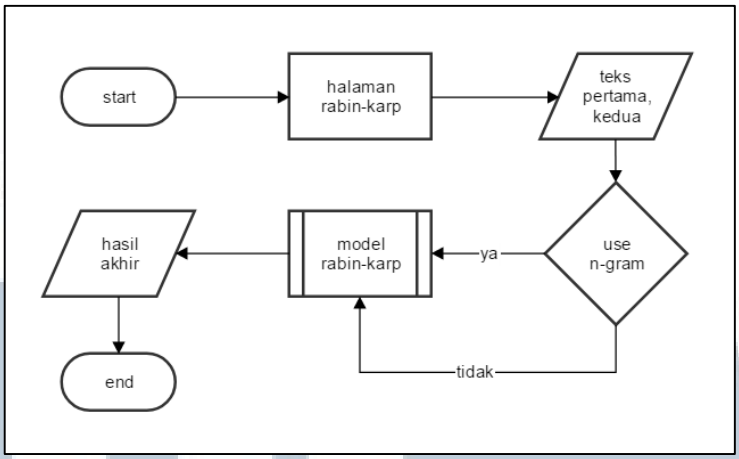
Gambar 3.3 Flowchart *Upload*

Dalam *flowchart* proses *upload*, sistem akan melakukan pengecekan terhadap dokumen yang di-*upload*. *User* akan melakukan *upload* dokumen pada halaman Home, jika dokumen tidak memiliki ukuran lebih besar dari 2000 kb dan memiliki ekstensi dengan jenis .pdf, .docx, atau .txt, proses *upload* berhasil. Kemudian sistem akan menyimpan data nama, ukuran dan ekstensi dokumen dalam *database* pada tabel *file_uploaded* serta meng-*upload* dokumen ke dalam *folder* yang sudah ditentukan. Apabila sistem menemukan bahwa dokumen yang di-*upload* berukuran lebih besar dari 2000 kb atau tidak berekstensi .pdf, .docx, atau .txt, proses *upload* gagal. Sistem tidak akan menyimpan data dokumen ke dalam *database* serta tidak melakukan *upload* dokumen ke dalam *folder* yang sudah ditentukan.



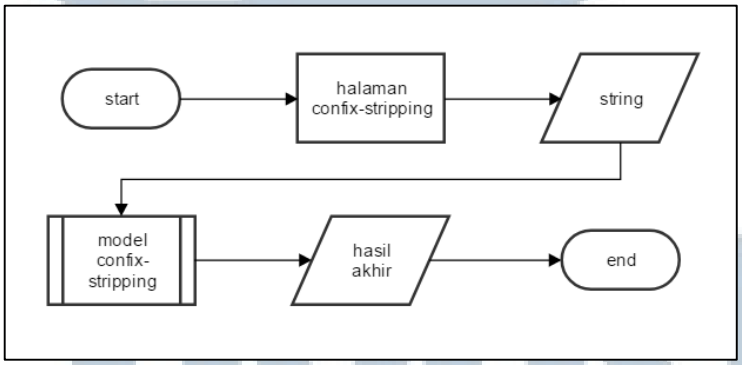
Gambar 3.4 Flowchart *Match File*

Dalam *flowchart* proses *match file*, sistem akan melakukan perbandingan kemiripan terhadap dua dokumen. *User* akan memilih dokumen utama dan pembanding pada halaman Match. Jika *user* tidak menekan tombol Go To The Next Step proses berhenti, jika ya proses dilanjutkan ke halaman Confirm untuk melakukan pengaturan nilai basis, atau gram, serta pemilihan proses tambahan seperti *parsing* N-Gram atau *stemming Confix-Stripping*. Proses perhitungan Rabin-Karp dilakukan dalam model Rabin-Karp, sedangkan *stemming* pada model *Confix-Stripping*. Setelah proses perhitungan selesai, hasil akhir akan disimpan ke dalam tabel *history*, kemudian persentase kemiripan juga ditampilkan pada halaman Result. Jika *user* menekan tombol Cancel pada halaman Confirm, maka proses membandingkan dokumen dibatalkan.



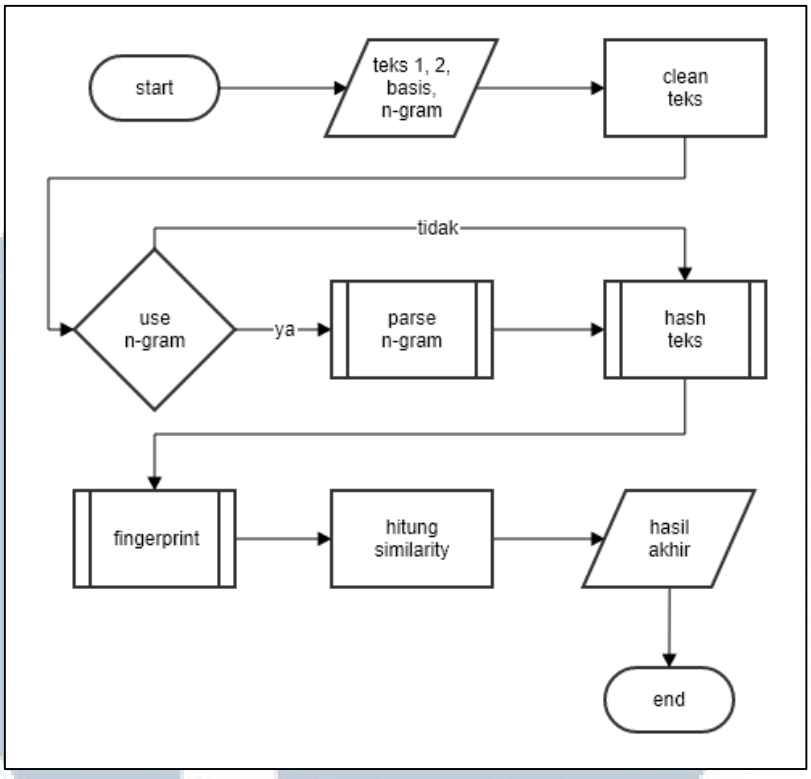
Gambar 3.5 Flowchart *Testing* Rabin-Karp

Dalam *flowchart* proses *testing* Rabin-Karp, sistem akan melakukan perbandingan kemiripan terhadap dua teks. *User* hanya perlu memasukkan teks pertama dan kedua, dan memilih menggunakan N-Gram atau tidak. Setelah itu perhitungan akan dilakukan pada subproses model Rabin-Karp.



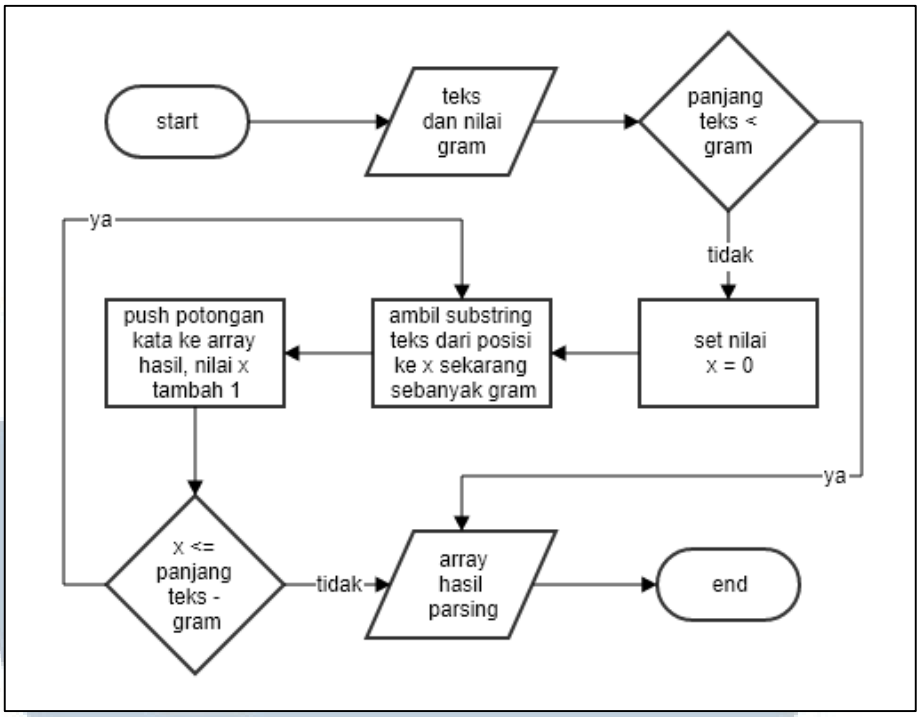
Gambar 3.6 Flowchart *Testing Confix-Stripping*

Dalam *flowchart* proses *testing Confix-Stripping*, sistem akan melakukan pencarian kata dasar. *User* hanya perlu memasukkan sebuah kata atau kalimat, kemudian proses pencarian kata dasar dilakukan pada subproses model *Confix-Stripping*. Proses pada Gambar 3.5 dan Gambar 3.6 merupakan fitur tambahan yang digunakan *admin* untuk melakukan *testing* dan *debugging* pada algoritma yang digunakan dalam penelitian.



Gambar 3.7 Flowchart Model Rabin-Karp

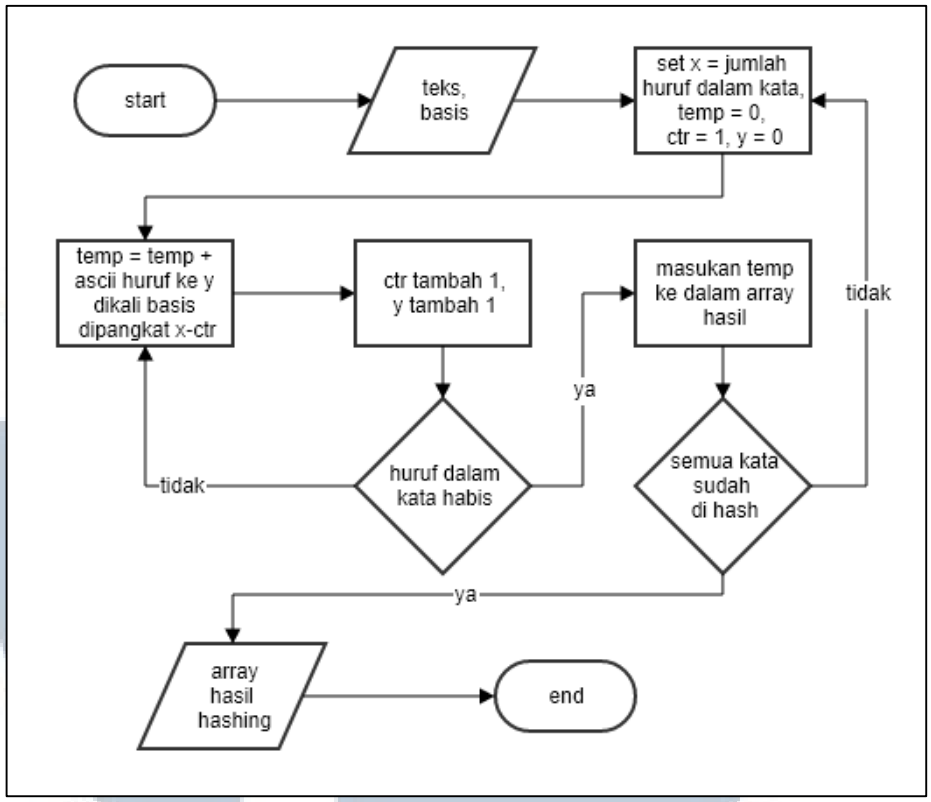
Gambar 3.7 merupakan *flowchart* model algoritma *string matching* Rabin-Karp yang digunakan untuk menghitung tingkat kemiripan pada dua dokumen teks. Model tersebut menerima input teks pertama, teks kedua, nilai basis, dan nilai N-Gram. Kemudian proses dilanjutkan dengan membersihkan teks dari delimiter-delimiter tertentu dan pengubahan huruf dalam teks ke huruf non kapital. Jika parameter yang diterima menggunakan N-Gram maka proses dilanjutkan dengan *parsing* N-Gram terlebih dahulu, kemudian dilanjutkan dengan melakukan *hashing* tiap kata dalam teks. Setelah kedua teks melalui proses tersebut, dilanjutkan dengan cek *fingerprint* kedua teks, dan perhitungan presentase kemiripan dengan menggunakan rumus *Dice's Similarity Coefficient*. Pada Gambar 3.7 terdapat subproses yang akan dijelaskan lebih rinci pada *flowchart parse* N-Gram, *hash* teks, dan *fingerprint*.



Gambar 3.8 Flowchart *Parse* N-Gram

Gambar 3.8 merupakan proses *parsing* yang dilakukan untuk mendapatkan potongan-potongan teks sebesar nilai gram yang diinginkan. Apabila panjang teks lebih kecil dari nilai gramnya, maka kata tersebut sudah tidak bisa dipotong sebesar nilai gram dan langsung dijadikan sebagai hasil, namun apabila panjang teks lebih besar dari nilai gramnya, maka proses *parsing* akan dilakukan dengan melakukan iterasi sebanyak panjang teks dikurang nilai gramnya.

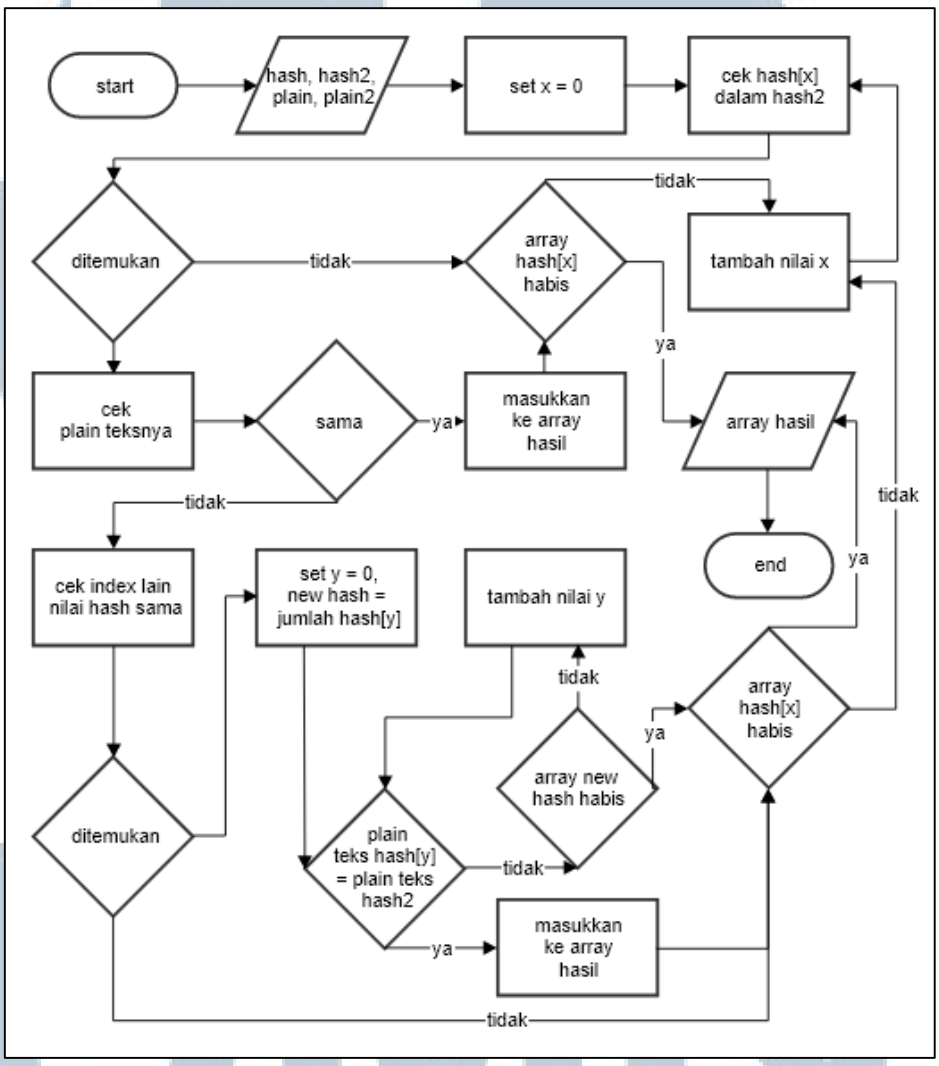
Sistem akan mengambil potongan-potongan teks sebesar nilai gram yang diinginkan dengan posisi pemotongan yang terus bertambah setiap kali proses iterasi dilakukan. Setiap kali hasil pemotongan didapatkan, potongan tersebut akan dimasukkan ke dalam *array* hasil.



Gambar 3.9 Flowchart *Hash* Teks

Gambar 3.9 merupakan *flowchart* proses *hashing* teks yang merupakan bagian dari proses algoritma Rabin-Karp. Pada proses tersebut akan dilakukan iterasi sebanyak jumlah kata dalam teks, baik yang telah melalui pemotongan dengan N-Gram terlebih dahulu ataupun tidak agar seluruh kata dalam teks dapat melalui proses *hashing*. Setiap kali iterasi bagian pertama tersebut dilakukan, akan dilakukan lagi iterasi didalamnya sebanyak jumlah huruf pada kata tersebut untuk menggeser posisi huruf pada kata tersebut dan akan dilakukan perhitungan didalamnya. Huruf tersebut akan diubah ke dalam bentuk ASCII-nya dan dikalikan dengan basis yang telah dipangkatkan berdasarkan nilai variabel pangkatnya. Nilai dari variabel pangkat akan dimulai dari jumlah huruf pada kata tersebut dikurang 1. Setiap kali iterasi bagian kedua dilakukan, maka nilai variabel pangkat yang sebelumnya akan dikurang 1, sehingga akan terbentuk perhitungan dengan nilai

pangkat yang menurun. Seluruh hasil perhitungan huruf yang telah dilakukan pada kata tersebut akan dijumlah, dan hasil penjumlahan tersebut merupakan nilai *hashing* pada kata tersebut.

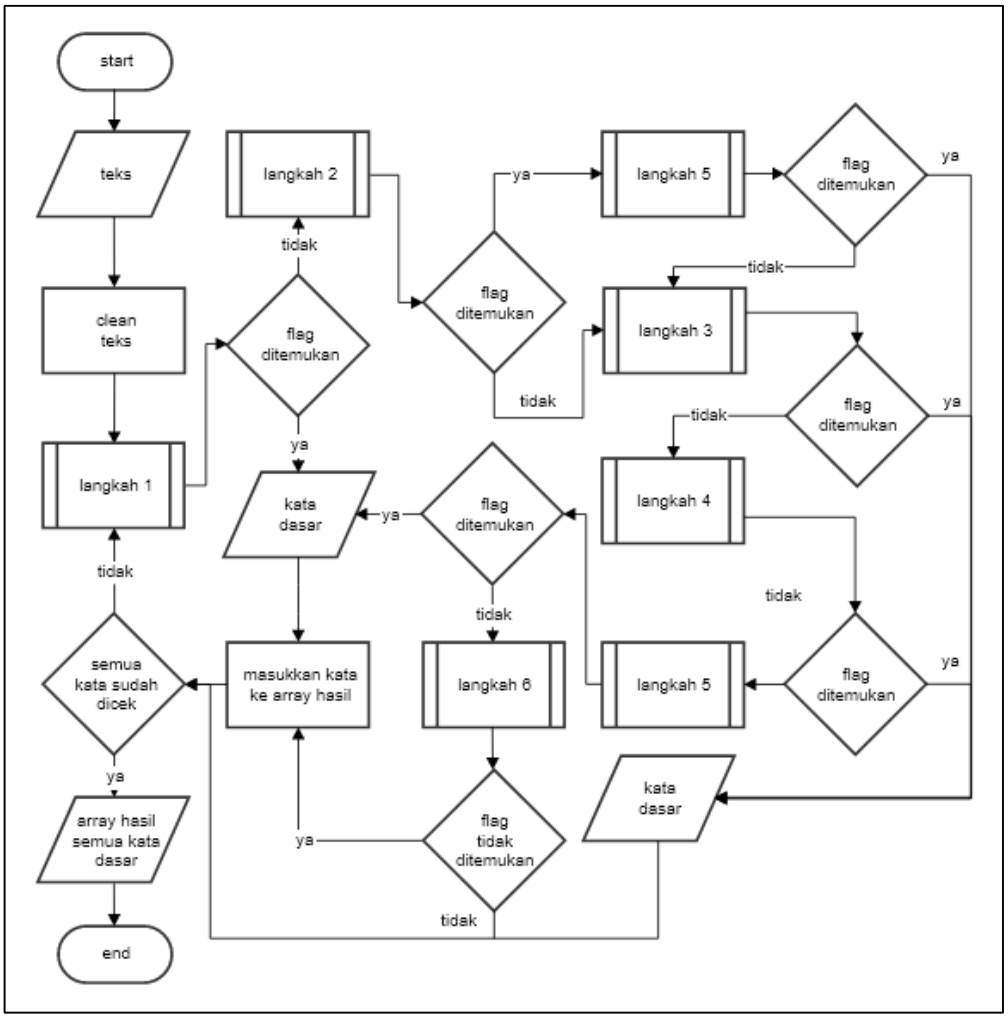


Gambar 3.10 Flowchart *Fingerprint*

Gambar 3.10 merupakan *flowchart* proses *fingerprint* dalam algoritma Rabin-Karp. Pada proses tersebut akan dilakukan iterasi sebanyak jumlah teks pada dokumen utama yang bertindak sebagai *key*. Setiap kali iterasi dilakukan sistem akan melakukan pengecekan nilai *hash* dokumen utama terhadap dokumen pembanding. Apabila ditemukan, maka nilai *hash* dicek apakah nilainya sama atau tidak. Jika nilainya tidak sama maka proses iterasi dilanjutkan dengan pengecekan

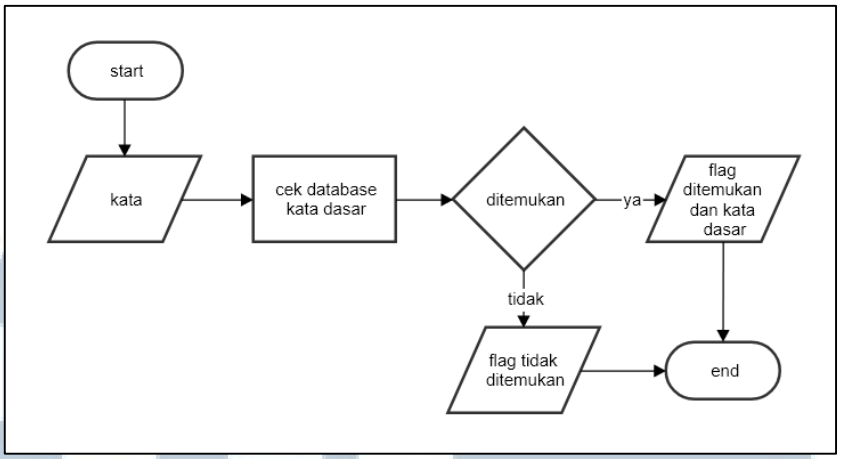
indeks nilai *hash* berikutnya, namun jika nilainya *hash*-nya sama, maka pengecekan berikutnya dilakukan pada *plain* teks dari kedua nilai *hash* tersebut. Apabila *plain* teks dari kedua nilai *hash* tersebut sama, maka nilai *hash* tersebut akan dimasukkan ke dalam *array* hasil dan proses iterasi berikutnya akan dilanjutkan selama nilai *hash* pada dokumen utama belum habis, namun jika *plain* teks dari kedua nilai *hash* tersebut tidak sama, sistem akan mencoba mencari kembali indeks lain pada dokumen pembanding yang memiliki nilai *hash* sama. Apabila terdapat indeks-indeks lain dengan nilai *hash* sama, maka proses iterasi bagian kedua akan dilakukan. Jika tidak terdapat indeks lain dengan nilai *hash* sama, maka proses iterasi bagian pertama akan dilanjutkan selama indeks *array* hasil *hash* dokumen utama belum habis.

Pada iterasi bagian kedua, sistem akan kembali melakukan pengecekan *plain* teks pada setiap nilai *hash* lain yang ditemukan. Apabila *plain* teksnya sama, maka niilai *hash* akan dimasukkan ke dalam *array* hasil, namun jika *plain* teksnya tidak ada yang sama, maka proses iterasi bagian pertama akan dilanjutkan dengan indeks dokumen utama yang berikutnya.



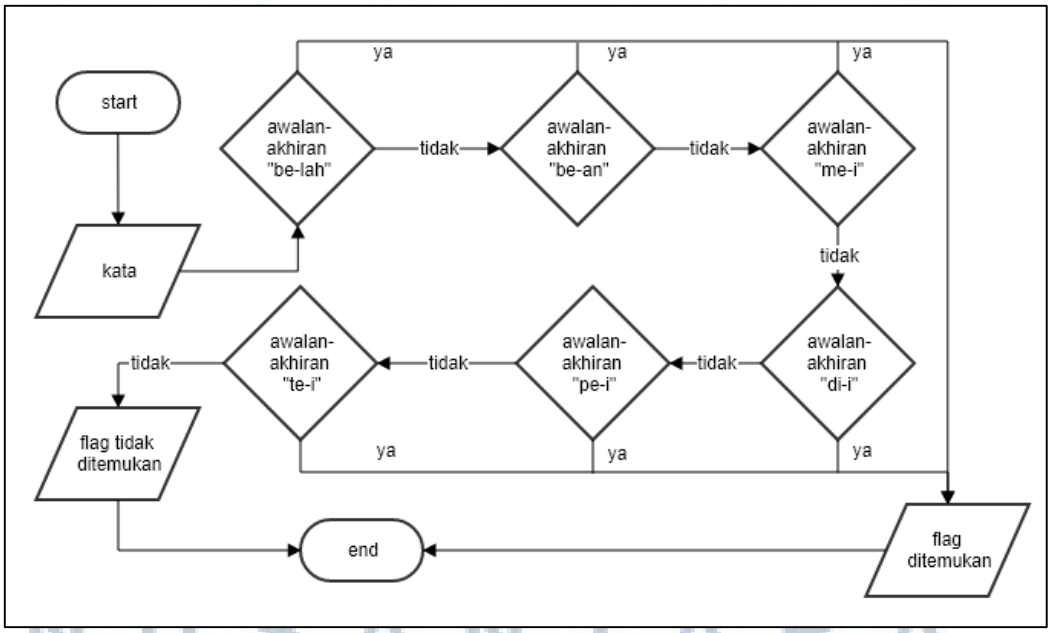
Gambar 3.11 Flowchart Model Algoritma *Confix-Stripping*

Gambar 3.11 merupakan *flowchart* model algoritma *Confix-Stripping* yang digunakan untuk melakukan *stemming* tiap kata pada sebuah teks. *Flowchart* tersebut menggunakan aturan sesuai dengan algoritma *Confix-Stripping* yang dibahas pada Bab II dalam laporan ini. Pada *flowchart* tersebut terdapat enam subproses yang masih akan dijelaskan lebih terperinci untuk menjelaskan langkah pertama sampai dengan langkah keenam pada algoritma *Confix-Stripping*.



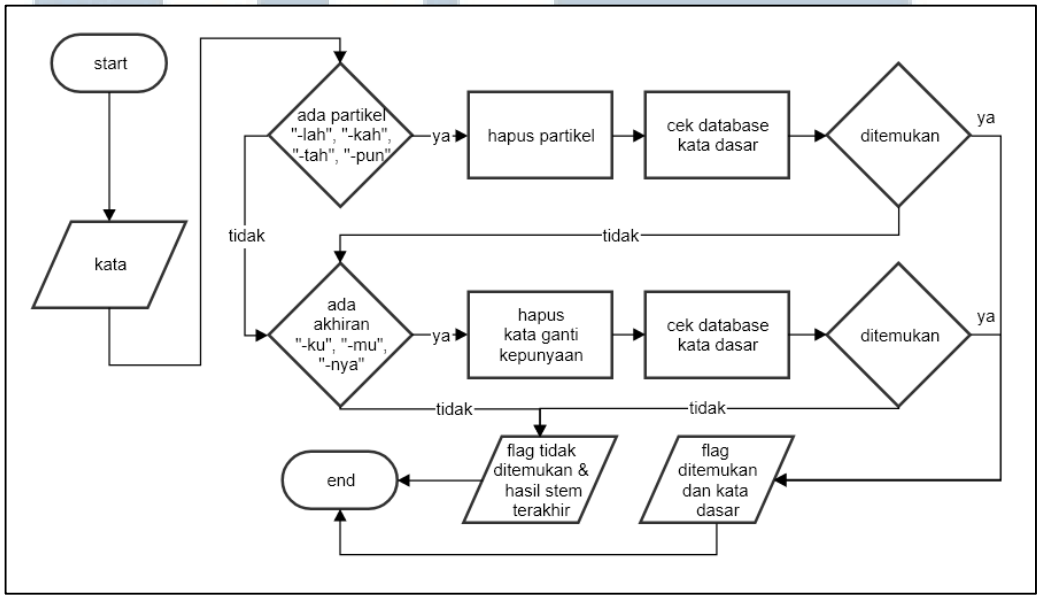
Gambar 3.12 Flowchart Langkah 1

Gambar 3.12 merupakan *flowchart* langkah kedua pada model algoritma *Confix-Stripping* yang berfungsi untuk melakukan pengecekan pertama terhadap kata awal ke dalam *database* kata dasar. Apabila pada langkah ini kata tersebut ditemukan di dalam *database*, maka *flag* ditemukan akan diberikan, dan langkah *stemming* selanjutnya tidak perlu dilakukan, namun apabila kata yang dicek tidak ada di dalam *database* maka *flag* tidak ditemukan akan diberikan dan langkah *stemming* selanjunya akan dilakukan.



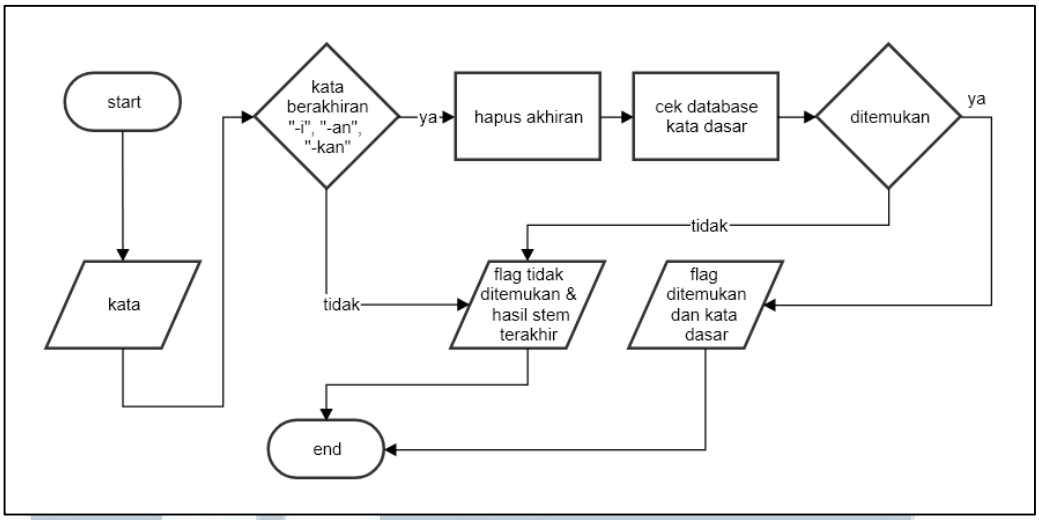
Gambar 3.13 Flowchart Langkah 2

Gambar 3.13 merupakan *flowchart* langkah kedua pada model algoritma *Confix-Stripping* yang berfungsi untuk melakukan pengecekan pada awalan dan akhiran sebuah kata. Apabila kata tersebut memiliki awalan dan akhiran “be-lah”, “be-an”, “me-i”, “di-i”, “pe-i”, dan “te-i” maka *flag* ditemukan akan diberikan, namun jika tidak terdapat awalan dan akhiran pada kata tersebut, maka *flag* tidak ditemukan akan diberikan



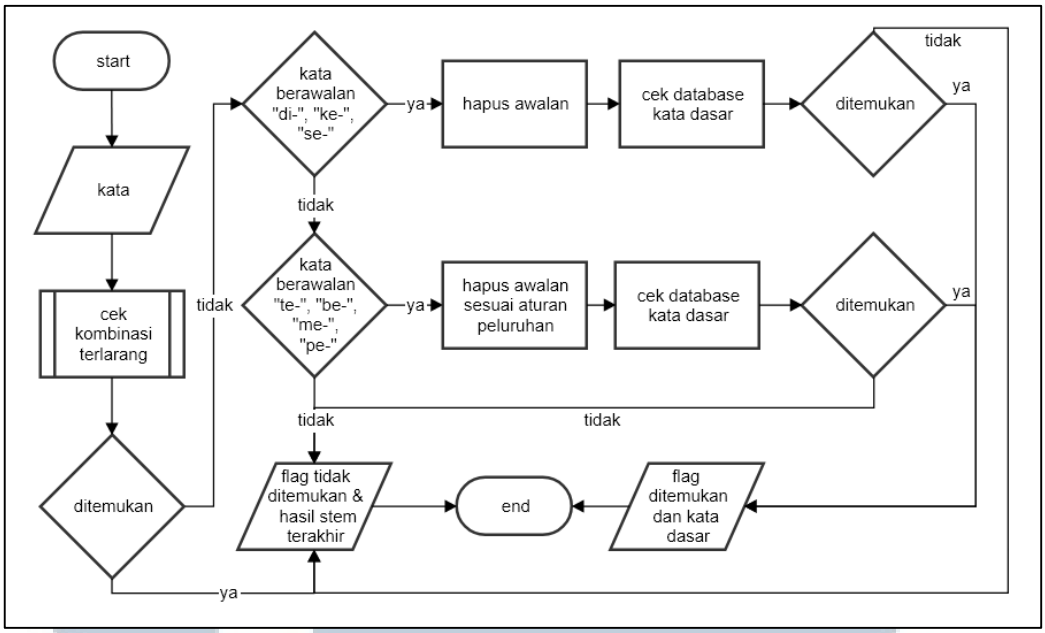
Gambar 3.14 Flowchart Langkah 3

Gambar 3.14 merupakan *flowchart* langkah ketiga pada model algoritma *Confix-Stripping* yang berfungsi untuk melakukan pengecekan pada akhiran sebuah kata. Apabila kata tersebut memiliki akhiran berupa partikel ataupun kata ganti kepunyaan, maka akhiran akan dihapus dari kata tersebut. Setelah penghilangan dua jenis akhiran tersebut dilakukan, maka kata hasil *stemming* akan dicek ke dalam *database* kamus kata dasar. Jika kata yang dicari ditemukan di dalam *database* kata dasar, maka kata tersebut dianggap sebagai kata dasar dan *flag* ditemukan akan diberikan. Apabila kata yang dicari tidak ditemukan di dalam *database* kata dasar, maka proses *stemming* belum berhasil dan *flag* tidak ditemukan akan diberikan.



Gambar 3.15 Flowchart Langkah 4

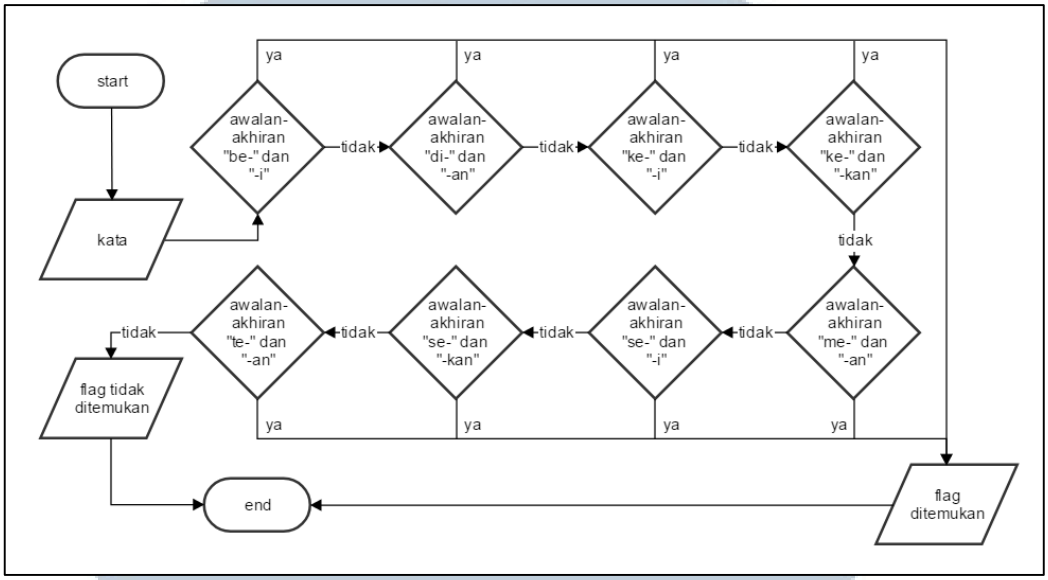
Gambar 3.15 merupakan *flowchart* langkah keempat pada model algoritma *Confix-Stripping* yang berfungsi untuk melakukan pengecekan pada akhiran sebuah kata. Apabila kata tersebut memiliki akhiran “-i”, “-an”, ataupun “-kan”, maka akhiran akan dihapus dari kata tersebut, setelah penghilangan akhiran tersebut dilakukan, maka kata hasil *stemming* akan dicek ke dalam *database* kamus kata dasar. Jika kata yang dicari ditemukan di dalam *database* kata dasar, maka kata tersebut dianggap sebagai kata dasar dan *flag* ditemukan akan diberikan. Apabila akhiran “-i”, “-an”, ataupun “-kan” tidak ditemukan atau kata yang sudah melalui proses *stemming* juga tidak ditemukan di dalam *database* kata dasar, maka proses *stemming* belum berhasil dan *flag* tidak ditemukan akan diberikan.



Gambar 3.16 Flowchart Langkah 5

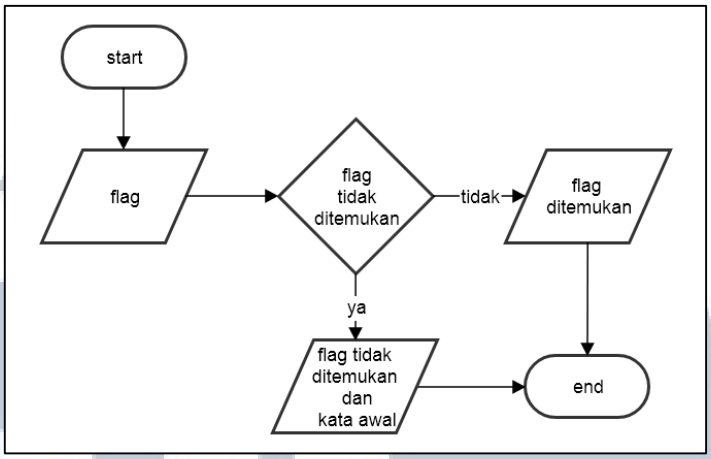
Gambar 3.16 merupakan *flowchart* langkah kelima pada model algoritma *Confix-Stripping* yang berfungsi untuk melakukan pengecekan pada awalan kata. Pada langkah ini terdapat pengecekan kombinasi awalan dan akhiran terlarang pada kata yang akan dijelaskan pada *flowchart* subproses kombinasi terlarang. Apabila kata tersebut tidak termasuk kata dengan kombinasi awalan dan akhiran terlarang, maka proses *stemming* dilanjutkan. Apabila kata tersebut memiliki awalan “di-“, “ke-”, atau “se-”, maka awalan tersebut dapat langsung dihilangkan dan kemudian dilakukan pengecekan hasil *stemming* ke dalam *database* kata dasar. Namun jika kata tersebut memiliki awalan “te-”, “be-”, “me-”, atau “pe-”, maka perlu dilakukan pemotongan awalan kata sesuai aturan peluruhan kata dasar yang tertera pada Tabel 2.3. Setelah proses pemotongan awalan kata dilakukan, maka kata tersebut akan dicek ke dalam *database* kata dasar, Jika kata ditemukan, maka kata tersebut dianggap sebagai kata dasar dan *flag* ditemukan akan diberikan. Apabila kata yang

dicari tidak ditemukan, maka proses *stemming* belum berhasil dan *flag* tidak ditemukan akan diberikan.



Gambar 3.17 Flowchart Kombinasi Terlarang

Gambar 3.17 merupakan *flowchart* yang terdapat dalam langkah kelima dari algoritma *Confix-Stripping*. Proses di atas berfungsi untuk melakukan pengecekan kombinasi-awalan dan akhiran dari sebuah kata yang tidak diperbolehkan berdasarkan aturan yang tertera pada Tabel 2.2. Apabila kata yang dicek mengandung awalan dan akhiran tersebut, maka proses *stemming* akan langsung dihentikan dan kata tersebut akan dianggap sebagai kata dasar, namun apabila kata tersebut tidak mengandung kombinasi awalan dan akhiran yang tidak diperbolehkan, maka proses *stemming* akan tetap dilanjutkan. Pada *flowchart* di atas, sistem akan memberikan *output* berupa *flag* ditemukan atau tidak ditemukan pada kata yang telah melalui pengecekan kombinasi awalan dan akhiran terlarang.



Gambar 3.18 Flowchart Langkah 6

Gambar 3.18 merupakan *flowchart* langkah keenam pada model algoritma *Confix-Stripping*. Langkah ini berfungsi untuk melakukan pengecekan apabila *flag* yang diterima dari proses sebelumnya adalah *flag* tidak ditemukan, atau dengan kata lain seluruh langkah telah gagal melakukan proses *stemming*, sehingga akan diberikan *flag* tidak ditemukan dan kata awal sebelu proses *stemming* dilakukan.

3.2.2 Struktur Tabel

Database engine yang digunakan pada sistem adalah MySQL. Berikut adalah struktur tabel pada aplikasi Matchit.

Nama Tabel : *user*

Fungsi : digunakan untuk menyimpan data *admin*

Tabel 3.1 Struktur Tabel *user*

Nama Kolom	Tipe	Ukuran	Informasi
id	int	11	*Primary Key*
username	varchar	25	
password	text		

Nama Tabel : *file_uploaded*

Fungsi : digunakan untuk menyimpan data dokumen

Tabel 3.2 Struktur Tabel *file_uploaded*

Nama Kolom	Tipe	Ukuran	Informasi
id	int	11	*Primary Key*
name	text		
ext	varchar	10	
size	float		

Nama Tabel : *kata_dasar*

Fungsi : digunakan untuk menyimpan kamus kata dasar.

Tabel 3.3 Struktur Tabel *kata_dasar*

Nama Kolom	Tipe	Ukuran	Informasi
id	int	11	*Primary Key*
katadasar	varchar	70	
tipe_katadasar	varchar	20	

Nama Tabel : *history*

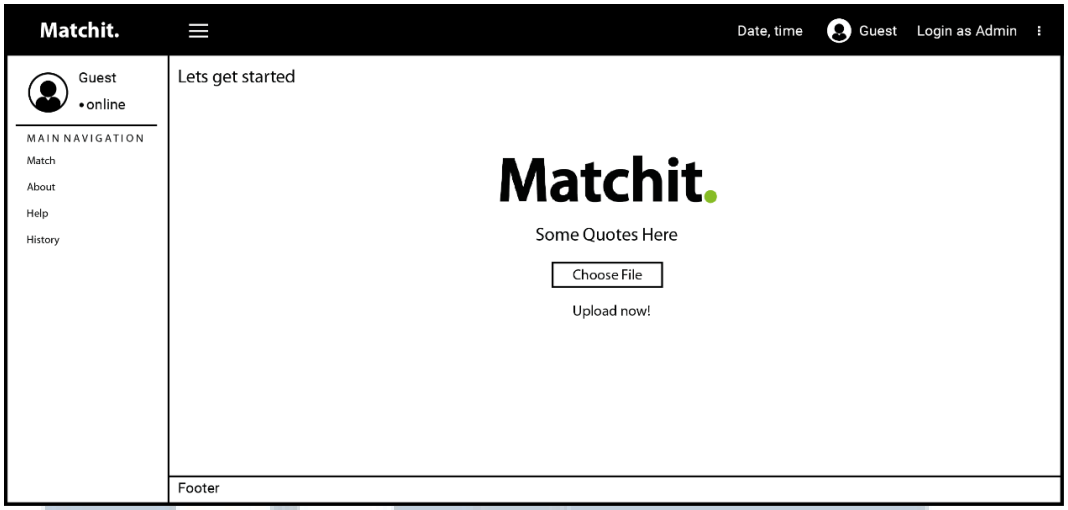
Fungsi : digunakan untuk menyimpan hasil perhitungan kemiripan dokumen

Tabel 3.4 Struktur Tabel *history*

Nama Kolom	Tipe	Ukuran	Informasi
id	int	11	*Primary Key*
file1	text		
file2	text		
word_count	int	11	
word_count2	int	11	
use_ngram	tiny_int	1	
use_stem	tiny_int	1	
cs_time	float		
cs_time2	float		
rk_time	float		
total_time	float		
similarity	float		

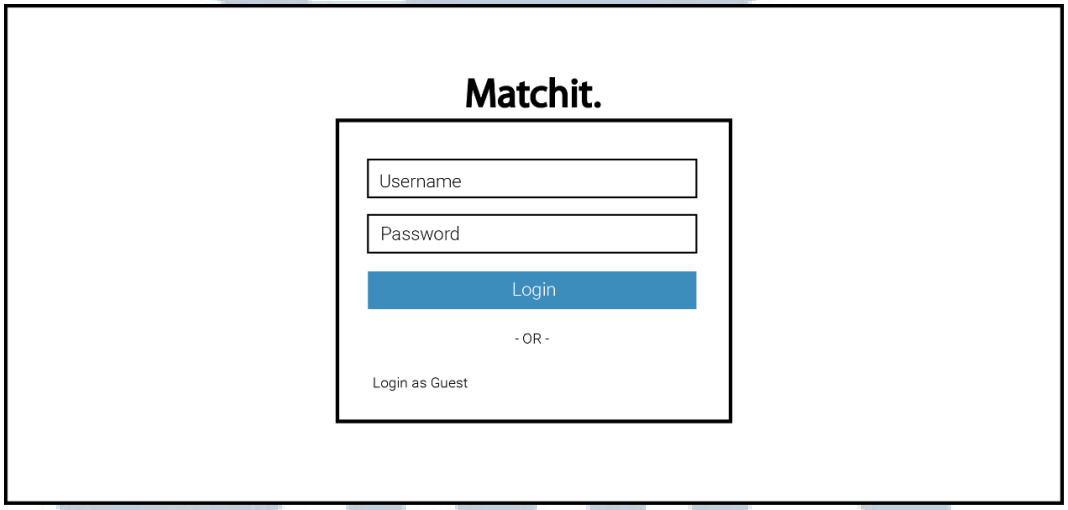
3.2.3 Desain Antarmuka

Berikut ini merupakan rancangan dasar dari *website* yang digunakan untuk penelitian dan pengimplementasian algoritma yang digunakan.



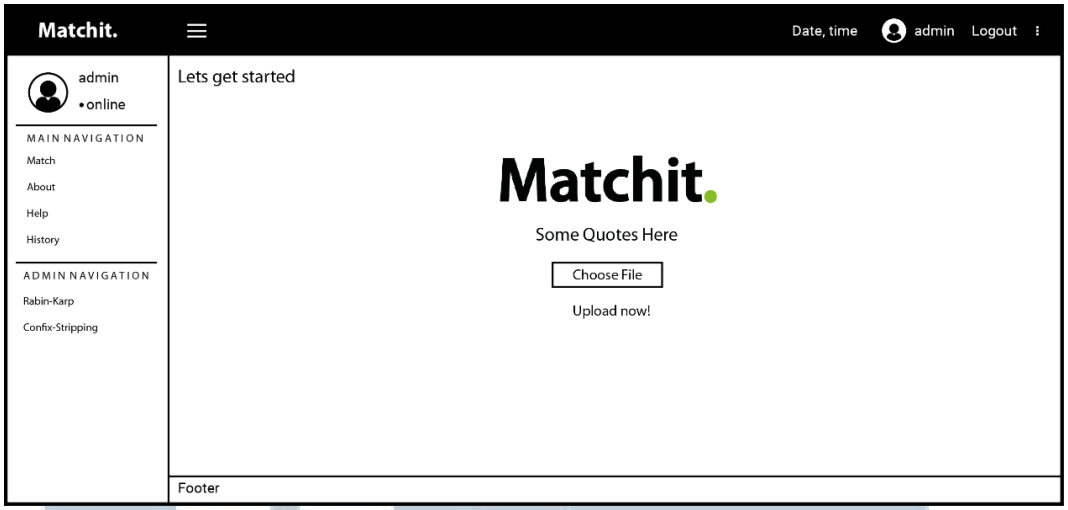
Gambar 3.19 Desain Halaman Home Guest

Saat pertama kali masuk ke dalam aplikasi, *user* akan menemukan halaman Home dengan *session guest*. Halaman ini juga berfungsi untuk mengunggah dokumen ke dalam sistem.



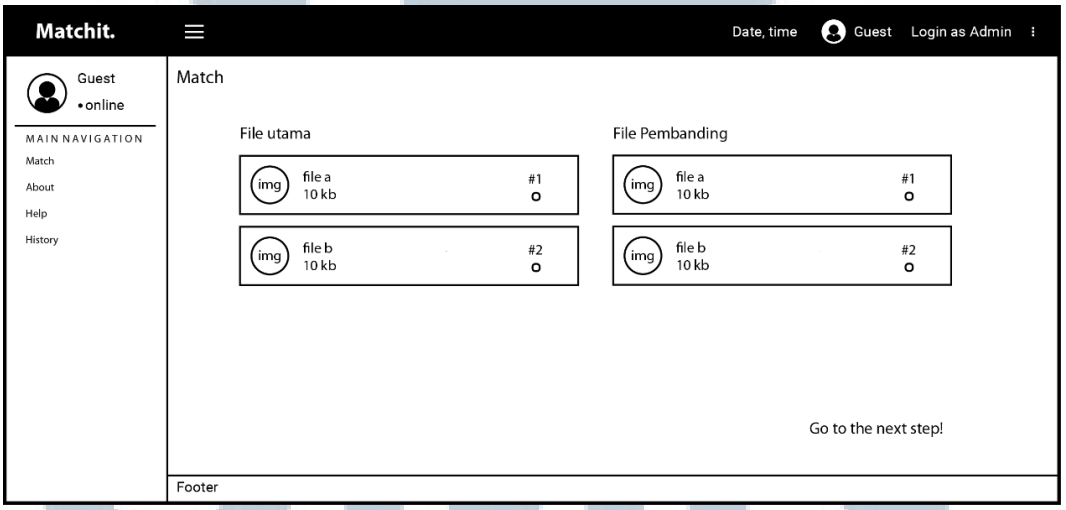
Gambar 3.20 Desain Halaman Login Admin

Jika *user* ingin masuk sebagai admin, *user* dapat melakukan *login* sebagai admin melalui halaman Login as Admin. *User* perlu memasukkan *username* dan *password*.



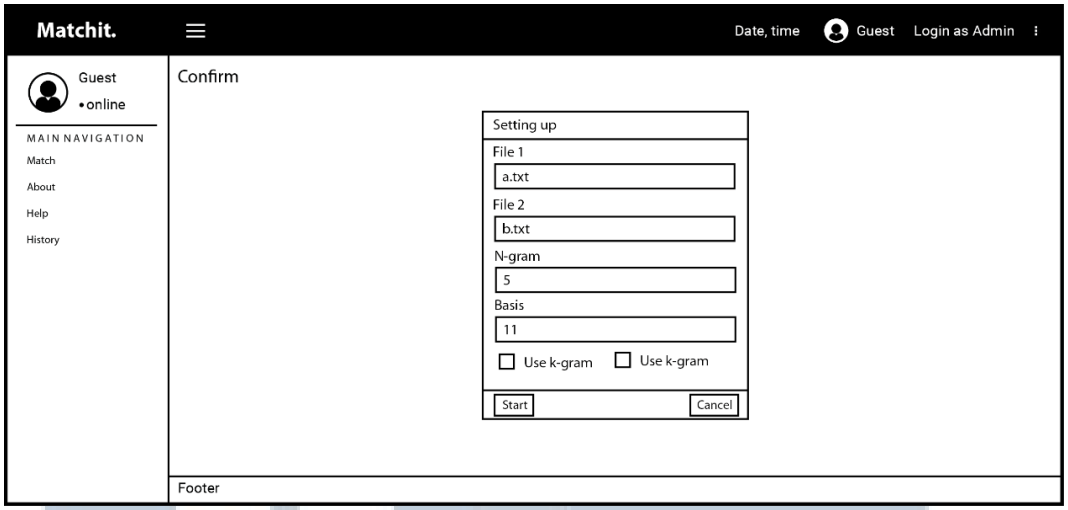
Gambar 3.21 Desain Halaman Home Admin

Gambar 3.21 merupakan desain halaman Home untuk *session* admin apabila *user* yang ingin masuk sebagai admin berhasil melakukan *login*. Pada halaman *website* dengan *session admin* diberikan dua menu tambahan pada bagian navigasi.



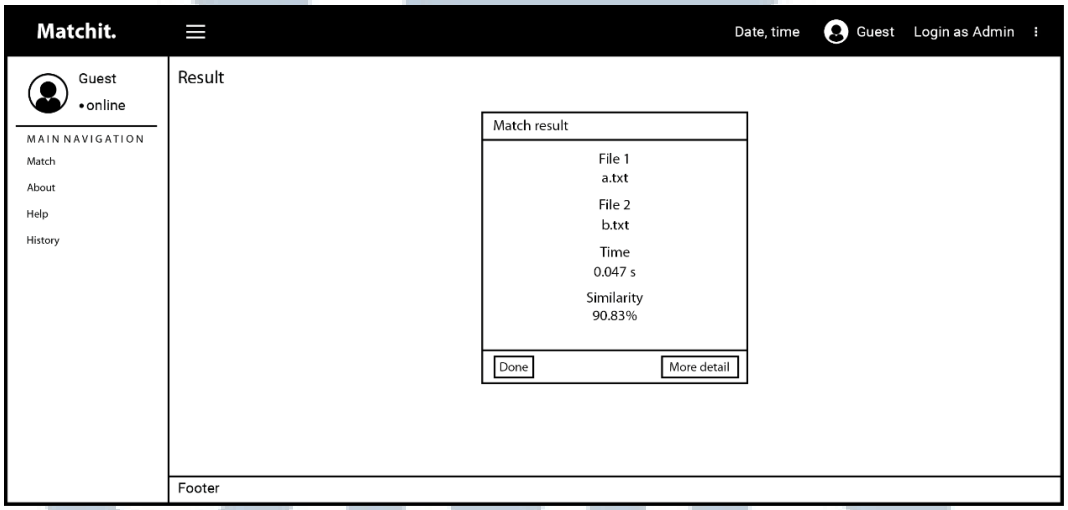
Gambar 3.22 Desain Halaman Match

Saat *user* ingin melakukan cek kemiripan dokumen, *user* terlebih dahulu memilih dokumen utama dan pembandingnya pada halaman Match. Untuk melanjutkan *user* cukup menekan tombol Go To The Next Step



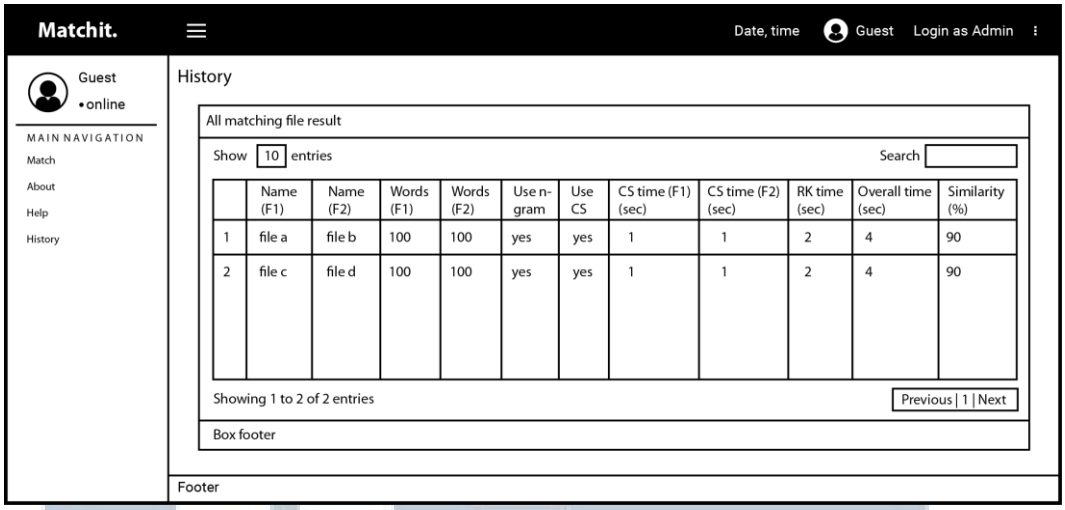
Gambar 3.23 Desain Halaman Confirm

Setelah memilih dokumen, proses dilanjutkan dengan melakukan pengaturan perhitungan kemiripan dokumen yang dapat diatur sesuai dengan keinginan *user* dengan beberapa pilihan yang disediakan.



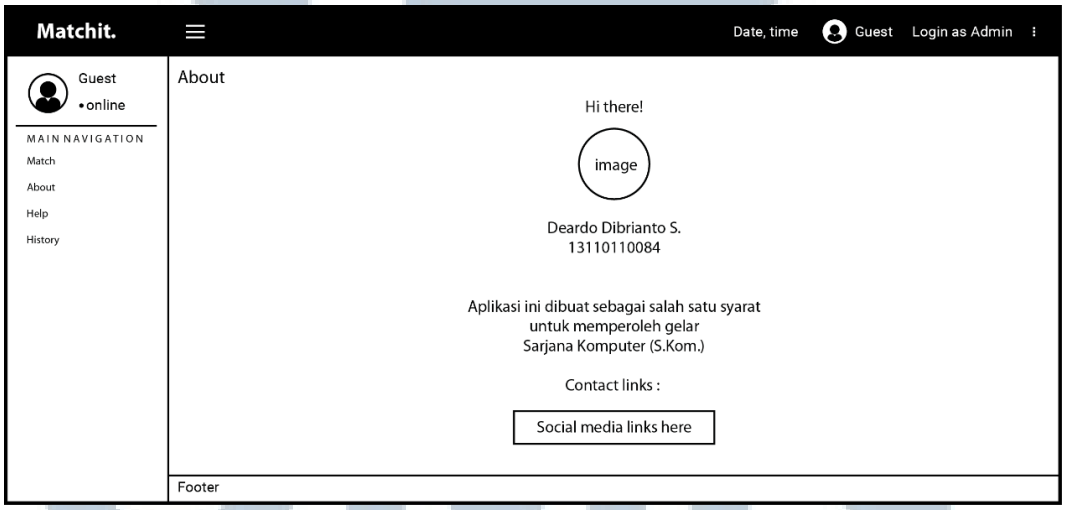
Gambar 3.24 Desain Halaman Result

Gambar 3.24 merupakan desain halaman yang menunjukkan hasil dari perhitungan kemiripan dua dokumen secara singkat. *User* dapat bernavigasi ke halaman History dengan menekan tombol More Detail.



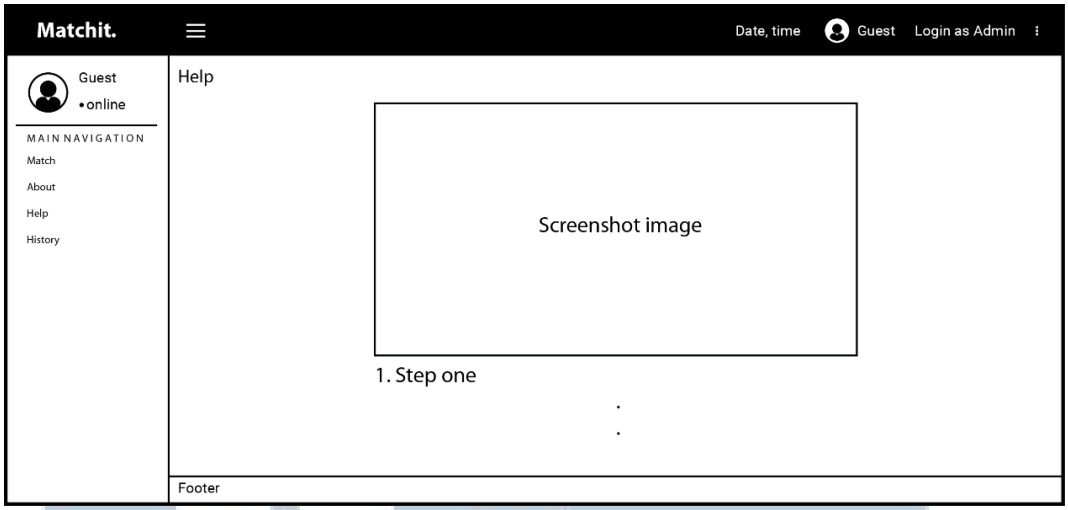
Gambar 3.25 Desain Halaman History

Halaman History berisi informasi hasil perhitungan kemiripan dua dokumen yang pernah dilakukan. Pada halaman tersebut diberikan tabel yang memberikan hasil perhitungan secara terperinci.



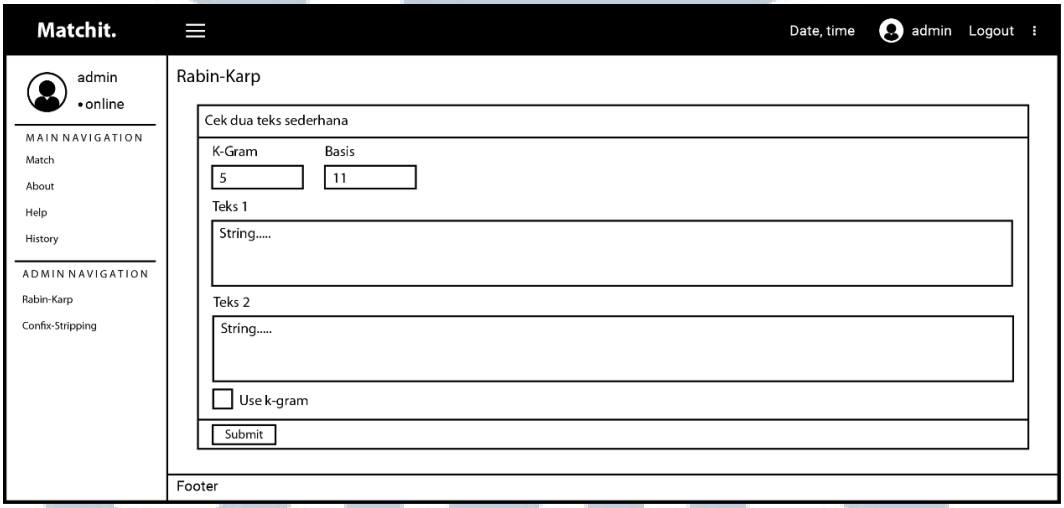
Gambar 3.26 Desain Halaman About

Halaman About berisikan informasi singkat tentang *programmer website* serta maksud dan tujuan dari dibuatnya aplikasi tersebut. Pada halaman tersebut juga terdapat tautan *social media* dari pembuat aplikasi



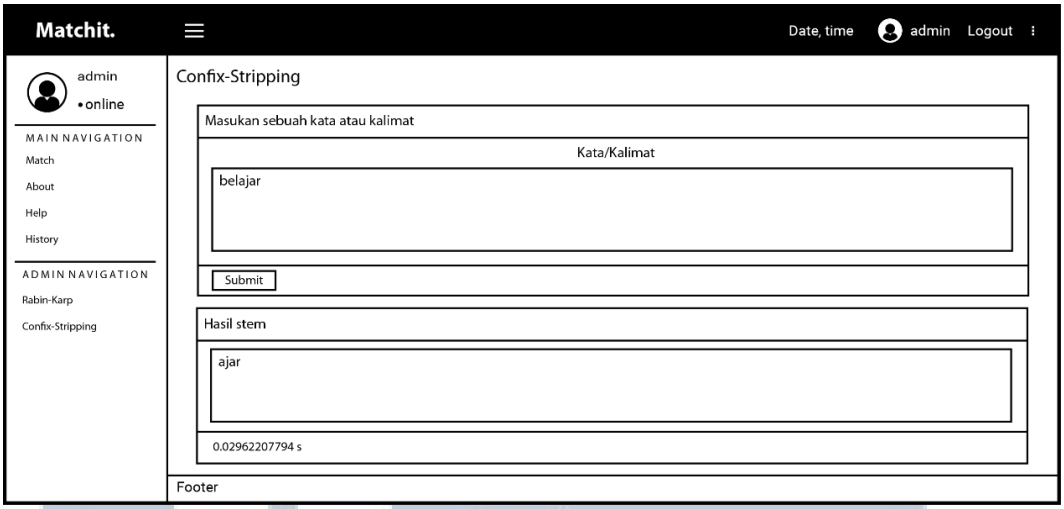
Gambar 3.27 Desain Halaman Help

Halaman Help berisi petunjuk penggunakan *website* Matchit dengan menggunakan contoh *screenshot website* beserta langkah-langkah penggunaannya. Langkah-langkah pada halaman tesrsebut adalah langkah dasar fitur utama aplikasi.



Gambar 3.28 Desain Halaman *Testing* Rabin-Karp

Gambar 3.28 merupakan desain halaman *testing* algoritma Rabin-Karp. Halaman tersebut merupakan fitur tambahan yang dapat digunakan apabila *user* masuk ke dalam *website* dengan menggunakan *session admin*.



Gambar 3.29 Desain Halaman *Testing Confix-Stripping*

Gambar 3.29 merupakan desain halaman *testing* algoritma *Confix-Stripping*. Halaman tersebut juga merupakan fitur tambahan yang dapat digunakan setelah *user* masuk ke dalam *website* dengan menggunakan *session admin*.