BAB III

METODOLOGI DAN PERANCANGAN APLIKASI

3.1 Metodologi Penelitian

Metode yang digunakan dalam penelitian dijabarkan dengan Tabel 3.1 berikut.

Tabel 3.1 Metode Penelitian

Kegiatan	Minggu ke-													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Telaah literatur														
Analisis kebutuhan														
Desain sistem														
Pemrograman sistem														
Testing dan *debug*														
Uji Coba dan evaluasi														
Konsultasi dan penulisan														

1. Telaah Literatur

   Pada tahapan telaah literatur, informasi akan dikumpulkan dari berbagai macam referensi seperti data, buku, dan jurnal-jurnal dari penelitian yang sesuai dengan penelitian yang akan dilakukan. Telaah literatur akan dilakukan dari awal penelitaan hingga akhir untuk terus menyesuaikan penelitian agar tidak menyimpang dari teori dan tujuan awal penelitian.

2. Analisis Kebutuhan

   Analisis akan dilakukan terhadap metode, algoritma, data, dan kebutuhan sistem apa saja yang diperlukan untuk melakukan penelitian ini. Analisis

terhadap algoritma akan dilakukan untuk menyesuaikan pengimplementasian algoritma terhadap program agar memiliki keluaran yang sesuai.

3. Desain Sistem

   Berdasarkan pengumpulan bahan-bahan yang diperlukan dalam perancangan sistem serta tetap mengacu pada teori yang sesuai maka pada tahapan ini sistem mulai didesain agar memiliki gambaran mengenai sistem yang akan dibangun, fungsionalitas, rancangan antar muka pada program akan didesain pada tahapan ini.

4. Pemrograman Sistem

   Berdasarkan rancangan maka pada tahapan ini akan dilakukan implementasi sesuai dengan rancangan dan mengacu pada teori. Implementasi akan direalisasikan dengan melakukan proses pengerjaan program (*coding*) sesuai dengan spesifikasi dan kegunaan yang sudah dirancang pada tahap sebelumnya termasuk tampilan fungsionalitas sudah dikerjakan pada langkah ini.

5. Testing dan Debug

   Sistem yang sudah dibangun akan dilakukan pengujian pada tahap ini untuk memastikan program berjalan sesuai kebutuhan dan fungsionalitas yang sudah dirancang dengan baik serta tidak adanya kegagalan program.

6. Uji Coba dan Evaluasi

   Uji coba dilakukan dengan menggunakan dokumen asli dibandingkan dengan dokumen uji yang berasal dari dokumen asli yang sudah dilakukan perubahan pada teks operasinya secara acak seperti menghapus, memindahkan, dan memasukkan kata lain. akan dilakukan evaluasi tingkat akurasi dilakukan dari keluaran program dengan menggunakan Rumus 2.3, 2.4, dan 2.5.

7. Konsultasi dan Penulisan

   Penulisan laporan dan konsultasi akan dilakukan dari awal tahap penelitian hingga akhir untuk mendokumentasikan segala hasil penelitian serta memperoleh kesimpulan dan saran bagi peneliti berikutnya yang ingin melanjutkan penelitian. Konsultasi dilakukan dengan dosen pembimbing sebagai pihak yang lebih berpengalaman untuk menuntun melakukan penelitian.

3.2 Perancangan Aplikasi

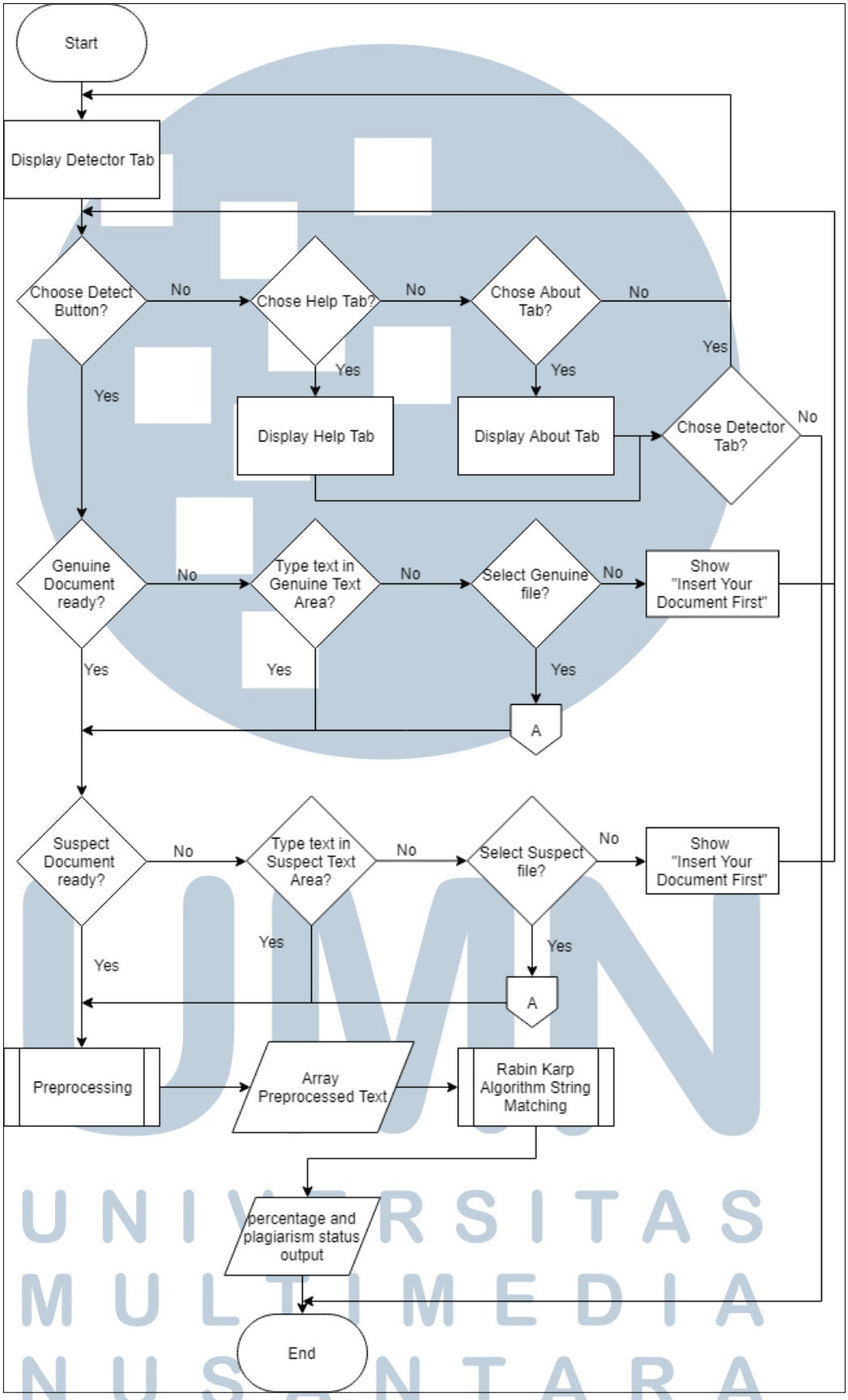
Pada tahapan perancangan aplikasi, perancangan berupa *flowchart* dan desain antarmuka.

3.2.1 Flowchart

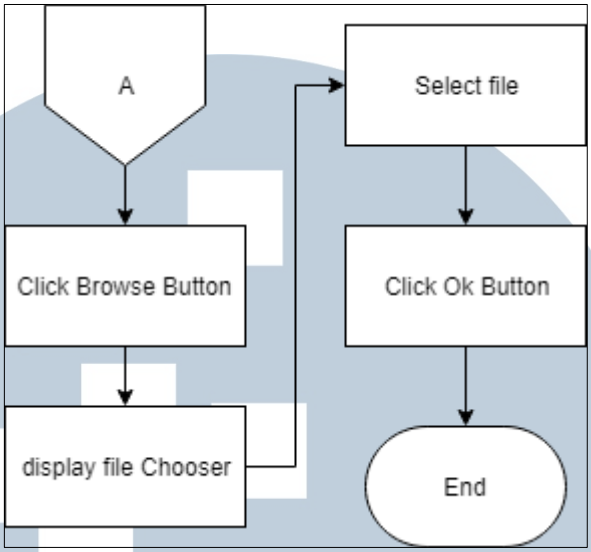
Untuk memberikan gambaran dari tahapan kinerja aplikasi yang dirancang maka menggunakan *flowchart* untuk menjelaskan system.

A. Flowchart Utama Aplikasi

Flowchart utama aplikasi berisi mengenai gambaran umum dari aplikasi. Berikut merupakan *flowchart* dan penjelasan dari alur utama aplikasi ditunjukan pada Gambar 3.1 dan 3.2.



Gambar 3.1 *Flowchart* Utama Aplikasi



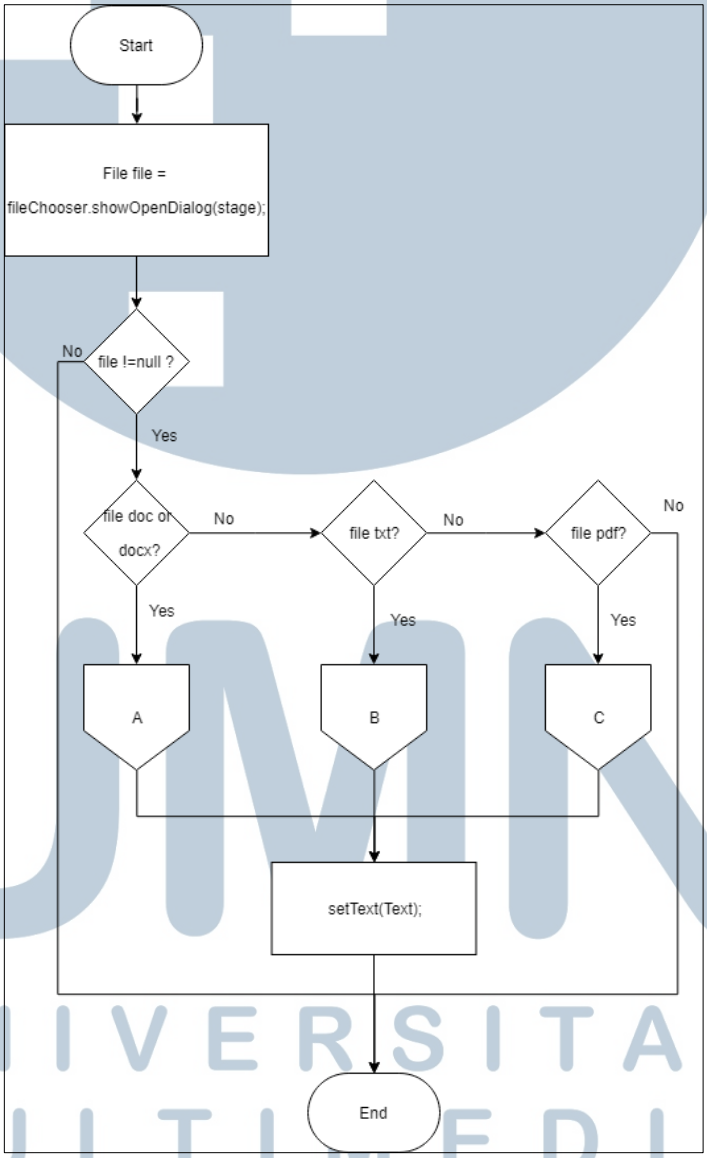
Gambar 3.2 Gambar *Flowchart* Utama Aplikasi (lanjutan)

Gambar 3.1 menunjukkan pada saat pertama kali menjalankan aplikasi. Halaman pertama yang akan muncul adalah Tab Detector. Terdapat tiga *tab* yang dapat dipilih oleh pengguna. Tab Help yang berisi mengenai petunjuk cara menggunakan aplikasi, Tab About yaitu *tab* yang berisi mengenai pembuat dari aplikasi tersebut serta tujuan dari aplikasi tersebut dibuat, dan Tab Detector yaitu *tab* utama yang berfungsi untuk melakukan deteksi plagiarisme.

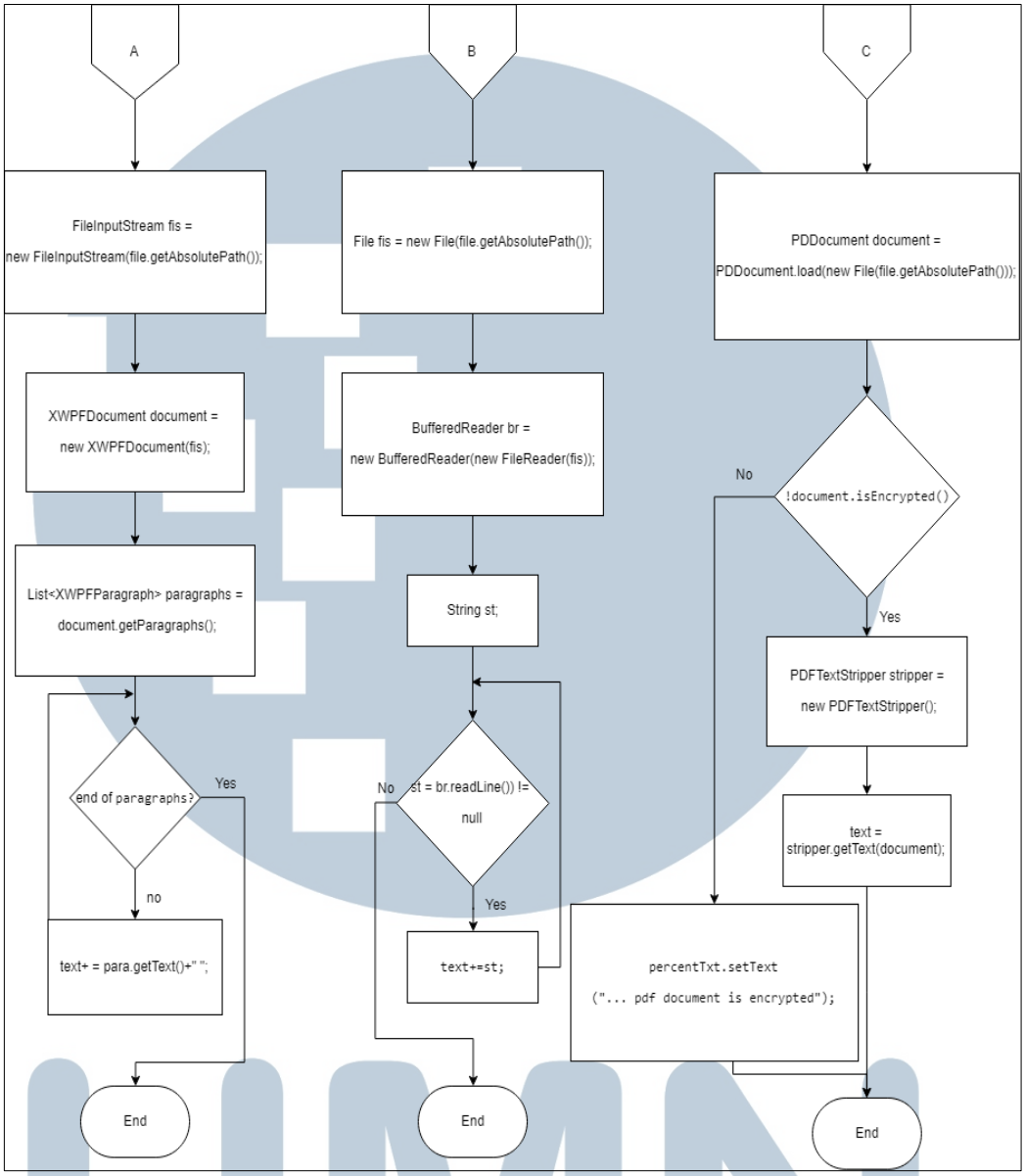
Pada Tab Detector pengguna diminta untuk memasukkan dua buah *file*, berupa satu *file* orisinal dan satu *file* yang akan dibandingkan dengan *file* pertama. Pengguna dapat memasukkan dokumen teks secara manual ke dalam *text area* yang tersedia pada aplikasi atau dengan menekan Tombol Browse pada aplikasi untuk memilih *file* seperti pada Gambar 3.2. Tipe *file* yang dapat digunakan berekstensi .pdf, .txt, .doc, dan .docx. Jika hanya terdapat satu *file* saja maka aplikasi tidak akan berjalan dan meminta hingga *file* orisinal dan pembanding sudah tersedia. Setelah semua *file* tersedia maka program dapat melakukan kalkulasi dari kesamaan kedua dokumen. Dimulai dari melakukan tahap *preprocessing*. *Array* hasil *preprocessing*

ini kemudian masuk ke dalam algoritma Rabin-Karp melalui proses *hashing* kemudian dilakukan pencocokan nilai dari kedua dokumen hasil *hash* tersebut. Hasil keluaran dari algoritma Rabin-Karp berupa jumlah dari nilai *hash* yang sama dari kedua dokumen. Hasil akhir program berupa persentase kesamaan serta status dari kedua dokumen tersebut.

B. Flowchart Select File



Gambar 3.3 *Flowchart Select File*



Gambar 3.4 *Flowchart Select File* (lanjutan)

Gambar 3.3 merupakan *flowchart* program saat pengguna telah selesai memilih *file*. Program akan memeriksa apakah *file* tersebut kosong atau tidak. Jika *file* tersebut memiliki isi maka akan dipastikan isinya, apakah berupa ekstensi *file* .doc atau .docx, .txt atau .pdf. Gambar 3.4 merupakan lanjutan dari Gambar 3.3 yang menunjukkan cara pengaksesan ketiga tipe ekstensi *file* tersebut.

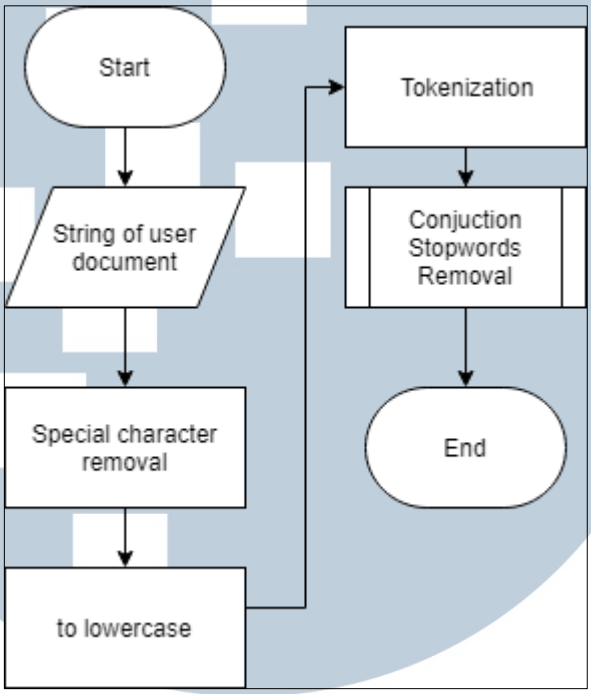
Tipe *file* pertama yang diakses yaitu berekstensi .doc atau .docx. Program akan membuka dokumen sesuai dengan alamat direktori *file* yang sudah dipilih oleh pengguna melalui *filechooser*. Program menggunakan *library* tambahan *Apache POI,* yaitu antarmuka pemrograman aplikasi pada java untuk melakukan pengaksesan pada *file* berupa dokumen *MS-Office.* Sehingga *file* yang tadi dibuka dapat diakses dengan memanggil metode *XWPFDocument*. Dokumen tersebut dimasukkan ke dalam *looping* agar dapat dipindahkan ke dalam variable bertipe *string*.

Tipe *file* kedua yang dapat diakses adalah berupa .txt. Pengaksesan *file* berekstensi .txt tidak memerlukan *library* tambahan. Alamat direktori *file* diambil dari *filechooser* untuk kemudian diakses menggunakan *BufferReader*. Dokumen dimasukkan ke dalam *looping* untuk dilakukan pemindahan kalimat ke dalam variable bertipe *string*.

Tipe *file* ketiga yang dapat diakses berekstensi .pdf. Pengaksesan dokumen bertipe pdf memerlukan *library* tambahan *PDFBox,* yaitu antarmuka pemrograman aplikasi dari *Apache* untuk melakukan operasi pada *file* berekstensi .pdf. Alamat direktori *file* diambil dari *filechooser* ketika pengguna menekan tombol Browse. *File* dapat diakses dengan memanggil metode *PDDocument*. Setelah diakses pdf dipastikan tidak ada proteksi. Jika pdf memiliki akses terproteksi maka aplikasi akan masang teks "… pdf document is encrypted", "…" akan terisi sesuai dengan *field* dokumen, berasal dari *field genuine* atau *suspected.* Jika dokumen tidak terproteksi maka isi dari pdf akan diakses menggunakan PDFTextStripper dan dimasukkan ke dalam variable bertipe *string*. Program akan menampilkan *file* hasil

pada *TextArea* sesuai dengan *field* dokumen yaitu *genuine* atau *suspected* untuk dapat diteruskan ke proses selanjutnya yaitu proses *preprocessing.*

C. Flowchart Preprocessing

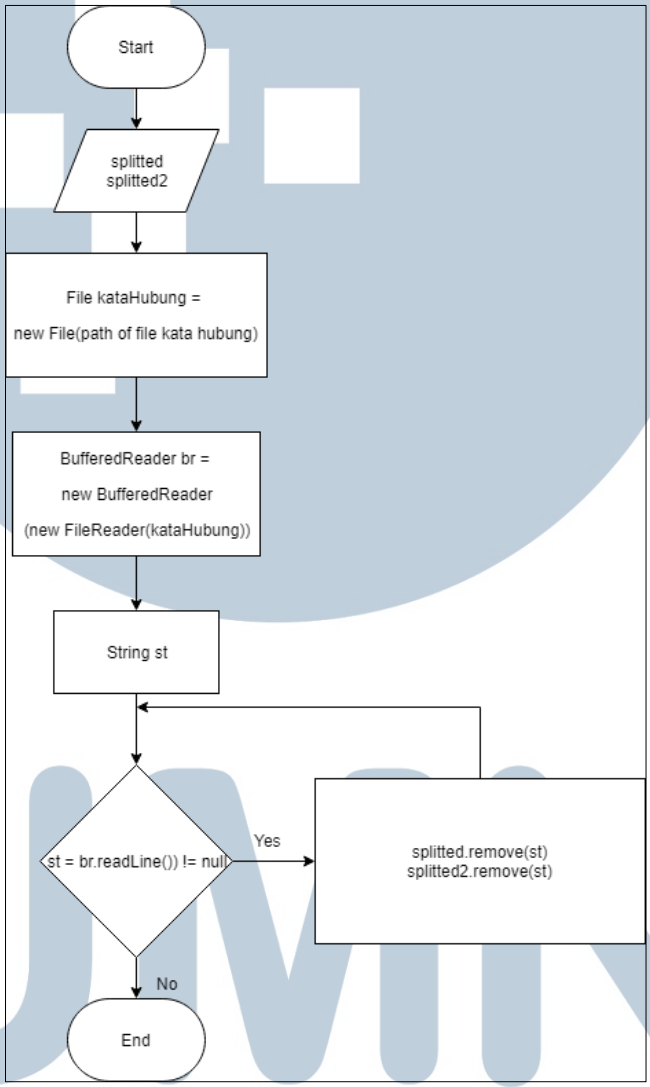


Gambar 3.5 *Flowchart Preprocessing*

Gambar 3.5 merupakan flowchart dari alur *preprocessing* yang dilakukan terhadap *file* pengguna. Program menerima masukkan dari pengguna berupa string. Proses pertama dalam *preprocessing* yaitu string dari pengguna tersebut akan dilakukan penyaringan karakter unik sehingga hanya menyisakan kata-kata berupa alfabetik dan numerik, kemudian proses kedua adalah string diubah menjadi huruf kecil. Proses ketiga program melakukan tokenisasi yaitu proses memisahkan string panjang menjadi kata perkata yang dibagi-bagi ke dalam *array - array*. Proses yang terakhir adalah program menerima dokumen berupa kata hubung yang sudah disediakan berupa *file* .txt berisi kata hubung untuk digunakan dalam proses penghilangan kata hubung terhadap data yang sudah dilakukan tokenisasi. Hasil

akhirnya akan mengeluarkan *array* yang berisi kata-kata yang sudah terpisah dengan menggunakan huruf kecil serta terbebas dari kata hubung dan karakter unik dimasukkan ke dalam array *splitted* dan *splitted2*.

D. Flowchart Conjuction Stopwords Removal

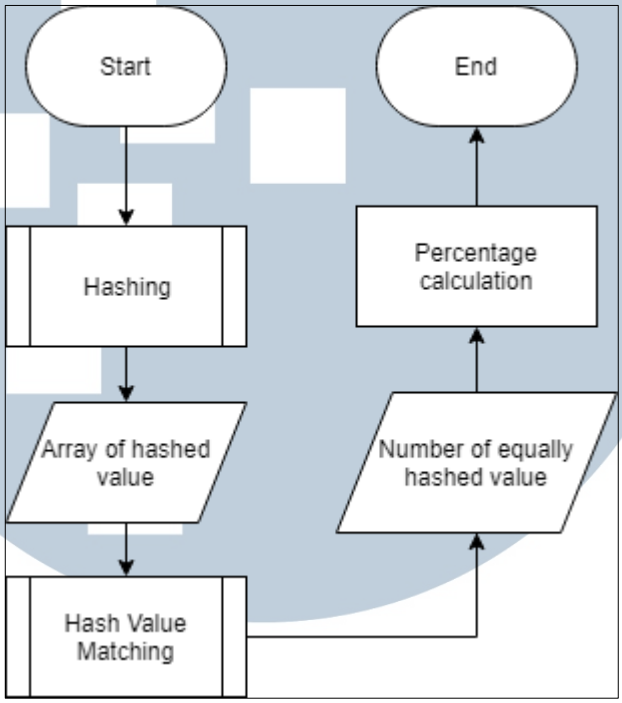


Gambar 3.6 *Flowchart Conjuction Stopwords Removal*

Gambar 3.6 menunjukan proses dari penghilangan kata hubung. *Array* dokumen pengguna yang sudah dilakukan preprocessing yaitu *splitted* dan *splitted2* hasil dari proses *lowercase*, *tokenization* akan dilakukan perbandingan dengan kata hubung. Kata hubung diambil menggunakan *file* reader. Jika terdapat kata hubung dalam

array dokumen pengguna maka akan dilakukan penghapusan. Keluarannya menghasilkan *Array splitted* dan *splitted2* yang terbebas dari kata hubung sehingga memudahkan dalam proses pencocokan *hash.*

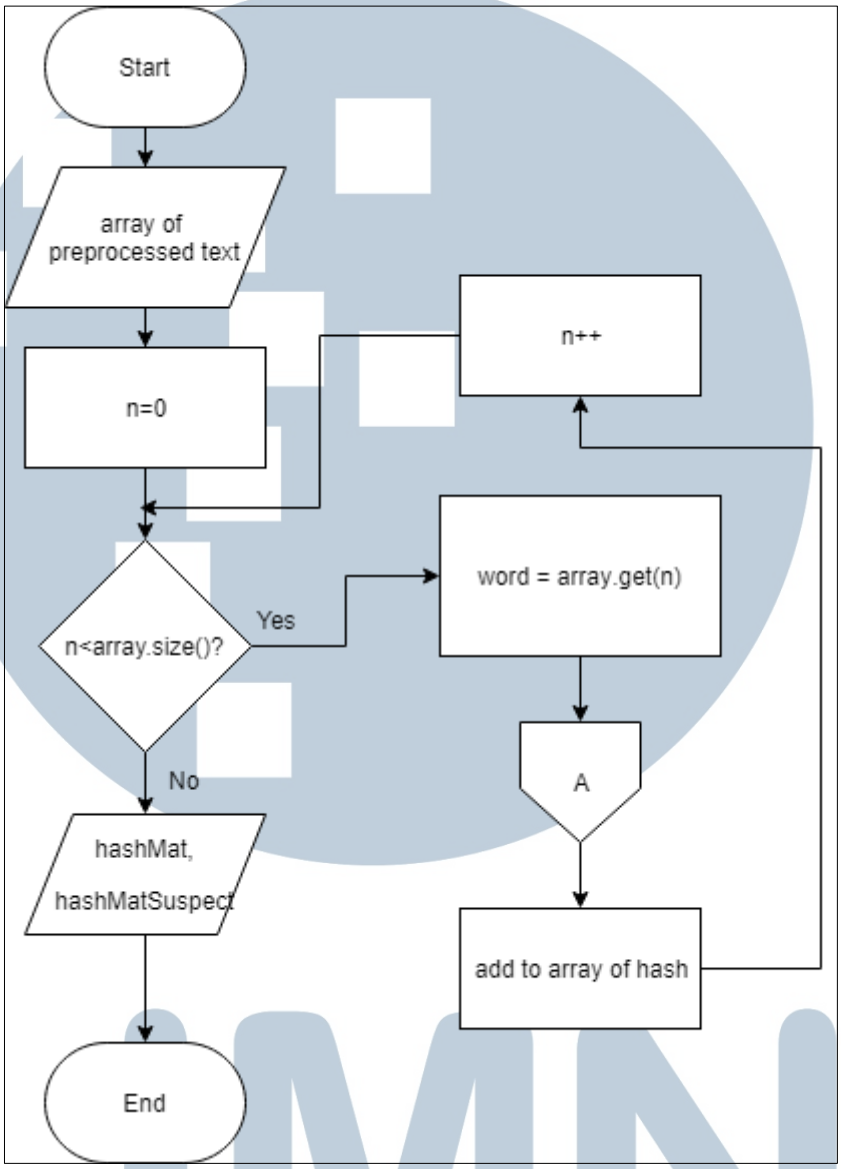
E. Rabin-Karp Algorithm String Matching



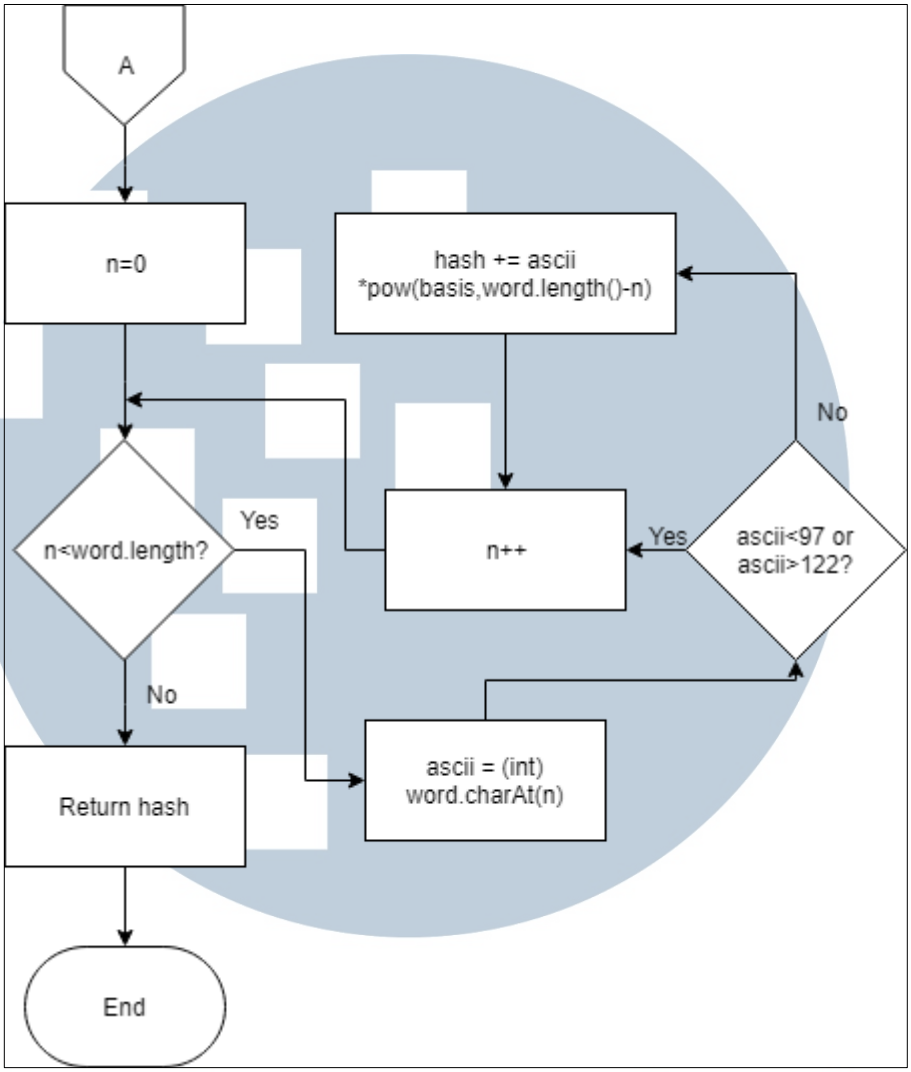
Gambar 3.7 *Flowchart Rabin-Karp Algorithm String Matching*

Gambar 3.7 merupakan flowchart dari alur Rabin-Karp dalam melakukan pencocokan string. Isi dari *array – array* yang sudah melalui proses *preprocessing* sebelumnya kemudian satu persatu dilakukan *hashing*. Proses *hashing* ini mengubah kata-kata dalam *array* menjadi nilai *hash* yang unik dan berbeda dari satu kata dengan kata lainnya. Selanjutnya dilakukan proses *matching* untuk menghitung jumlah kesamaan nilai *hash*. Jumlah kesamaan *hash* digunakan untuk melakukan perhitungan persentase kesamaan dari kedua dokumen oleh program. Hasil akhir dari program adalah menampilkan persentase hasil perhitungan serta status plagiat.

F. Flowchart Hashing



Gambar 3.8 *Flowchart Hashing*

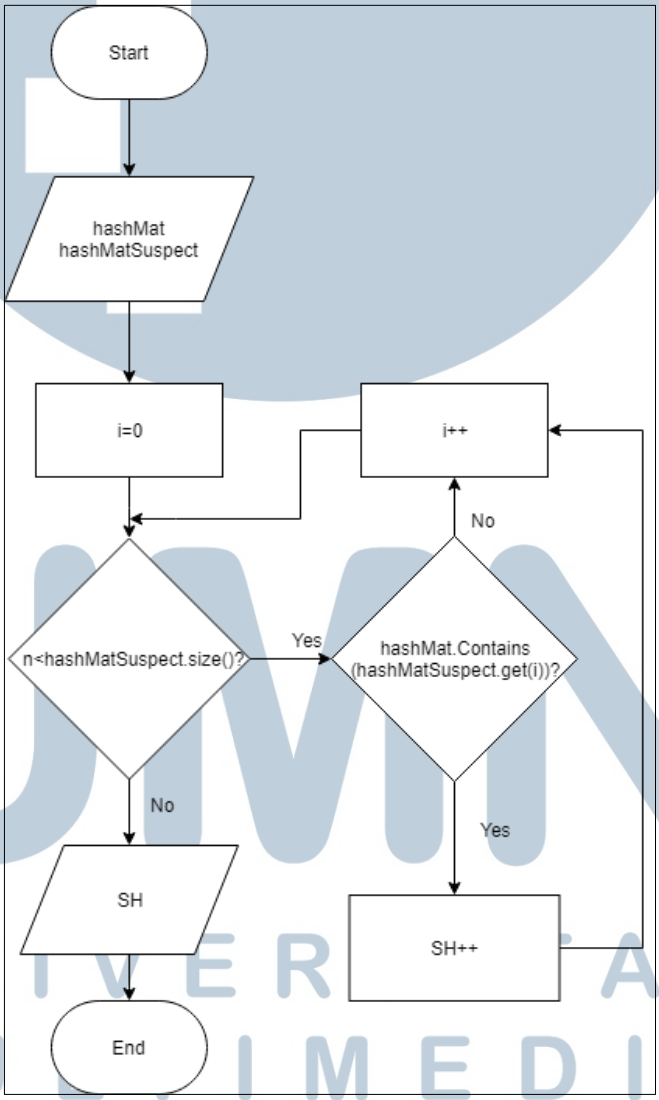


Gambar 3.9 *Flowchart Hashing* (lanjutan)

Gambar 3.8 merupakan *flowchart* dari proses *hashing* yang terjadi di dalam program. Variabel n merupakan variabel yang digunakan sebagai penanda untuk mengakhiri pengulangan. *Array* string yang telah dilakukan *preprocessing* dilakukan *looping* untuk mengambil kata perkatanya yang kemudian dimasukkan ke dalam fungsi *hash*. Kata–kata yang masuk ke dalam fungsi *hash* ditunjukkan pada Gambar 3.9, masuk ke dalam *loop* untuk dipisahkan per karakter dan ditempatkan pada variabel ascii dengan tujuan untuk mendapatkan nilai ASCII nya, variabel *ascii* akan ditentukan apakah karakter yang terdapat di dalamnya berupa

abjad, jika karakter bukan merupakan abjad maka variabel *n* bertambah satu dan melanjutkan *loop* kembali. Jika karakter berupa abjad maka nilai ascii akan dilanjut untuk perhitungan menggunakan rumus *hash function* Rumus 2.1. Hasil akhir perhitungan ini mengembalikan nilai *hash* unik berbeda antara satu kata dengan kata lainnya. *Hash-hash* unik ini kemudian dimasukkan ke dalam *array hashMat* dan *hashMatSuspect*.

G. Flowchart Hash Value Matching

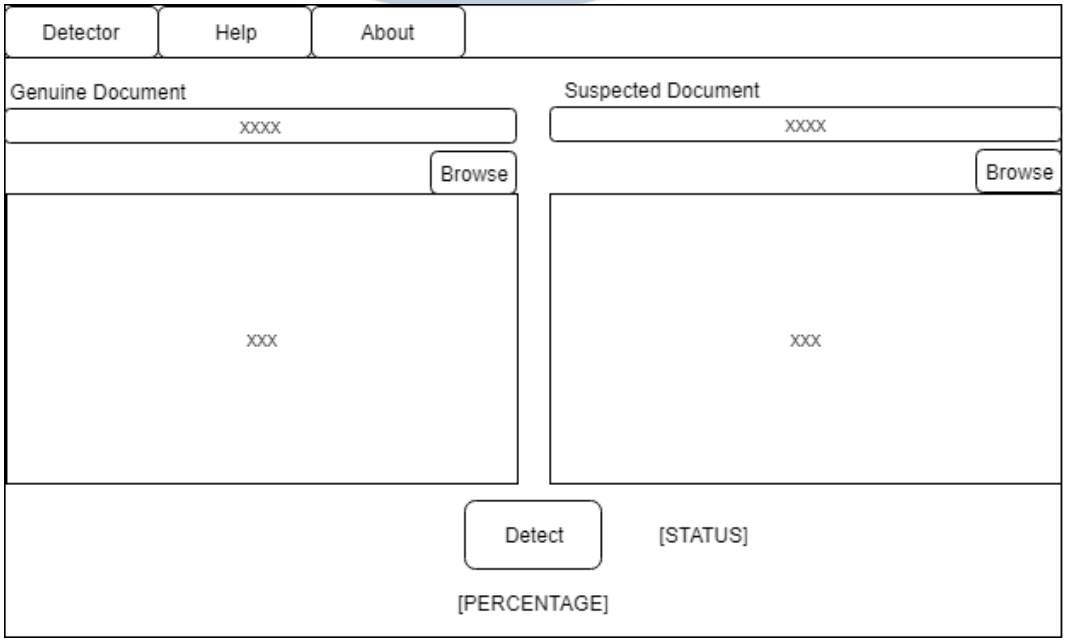


Gambar 3.10 *Flowchart Hash Value Matching*

Gambar 3.10 menunjukan proses pencocokan nilai *hash* untuk menentukan jumlah kesamaan *hash* dokumen yang dimasukkan oleh pengguna. Variabel i merupakan variabel yang digunakan sebagai penanda dari pengulangan. Dokumen yang dimasukkan oleh pengguna akan dicocokan apakah nilai *hash* dalam dokumen yang diduga melakukan plagiat terdapat pada *array* dokumen asli. Jika terdapat nilai *hash* yang sama maka variabel akan melakukan penambahan terhadap jumlah variabel *SH*. Kemudian proses akan mengembalikan nilai jumlah *hash* yang sama dalam variabel *SH*.

3.2.2 Desain Antarmuka Aplikasi

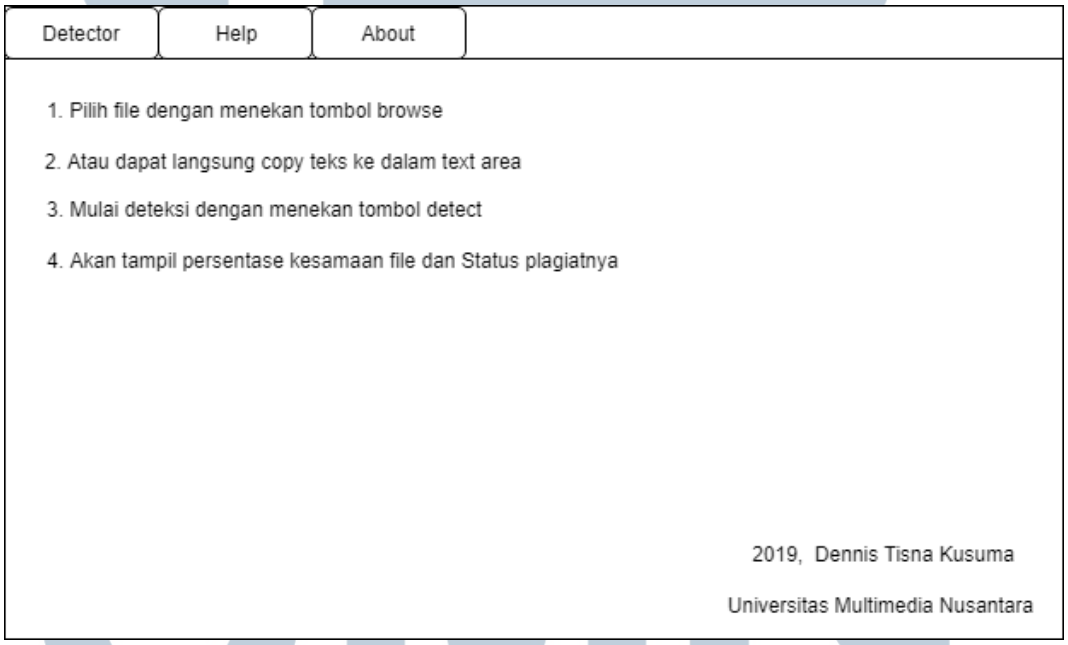
Desain antarmuka ini digunakan sebagai panduan dalam membangun tampilan dari aplikasi. Aplikasi ini terdiri dari 3 *tab* utama. Berikut merupakan desain antarmuka dari aplikasi.



Gambar 3.11 Tab *Detector*

Saat pengguna membuka aplikasi, maka halaman pertama yang muncul adalah seperti pada Gambar 3.11. Bagian paling atas merupakan *tab* - *tab* yang dapat

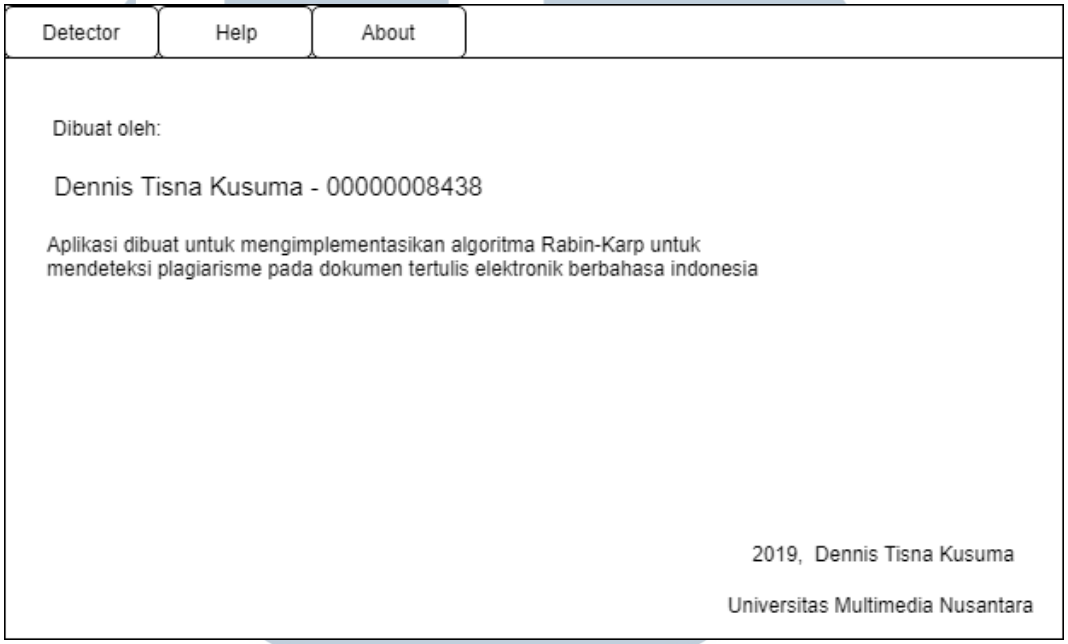
dipilih oleh pengguna. Kemudian di bawah bagian pemilihan tab terdapat bagian untuk memasukkan dokumen. Tombol Browse berguna untuk membuka *filechooser* bagi pengguna untuk melakukan pemilihan *file*. Alamat dari direktori *file* akan tampil pada *textfield* di atas tombol Browse tersebut. Pengguna juga dapat langsung memasukkan teks yang disalin langsung ke dalam *textarea* yang sudah disediakan. Tombol Detect berfungsi untuk mulai melakukan pencocokan kesamaan antara kedua dokumen yang sudah dimasukkan oleh pengguna. Setelah prosesnya berlangsung maka program akan menuliskan tingkat persentase kesamaan dokumen serta status plagiarisme.



Gambar 3.12 Tab *Help*

Gambar 3.12 merupakan desain antarmuka dari Tab Help yang dapat dipilih oleh pengguna. Tab Help berisi mengenai petunjuk cara menggunakan aplikasi. Petunjuk berupa bagaimana cara memilih *file* dengan menekan tombol Browse atau pengguna juga dapat langsung meletakan salinan ke dalam teks area. Pengguna kemudian dapat menekan tombol Detect untuk memulai pendeteksian dan

mendapatkan hasil persentase dan status kesamaannya. Pada bagian pojok kanan bawah merupakan tahun pembuatan, nama penulis, serta universitas tempat penulis menuntut ilmu.



Gambar 3.13 Tab *About*

Gambar 3.13 merupakan gambar dari desain antarmuka Tab About. *Tab* tersebut berisi mengenai nama pembuat aplikasi beserta nomor induk mahasiswa. Tujuan dari aplikasi ini dibuat yaitu untuk mengimplementasikan algoritma Rabin-Karp sebagai pendeteksi plagiarisme pada dokumen tertulis elektronik Bahasa Indonesia. Di pojok kanan bawah terdapat tahun pembuatan aplikasi, nama penulis dan universitas tempat penulis menuntut ilmu.