BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam membangun aplikasi jembatan keledai dengan menggunakan algoritma genetika dan *Markov Chain Model* adalah sebagai berikut:

1. Studi Literatur

   Pada tahap ini dipelajari dengan seksama ilmu-ilmu yang dibutuhkan untuk merancang dan membangun aplikasi jembatan keledai. Jurnal-jurnal, buku-buku dan skripsi-skripsi terkait akan menjadi bahan referensi dalam perancangan dan pembangunan aplikasi.

2. Penyebaran survei

   Penyebaran survei dilakukan untuk mengetahui kebutuhan dan manfaat aplikasi jembatan keledai bagi masyarakat. Penyebaran survei diutamakan kepada anak sekolah dan perguruan tinggi karena pelajaran yang terdapat banyak hafalan ada pada jenjang sekolah dan perguruan tinggi.

3. Perancangan Sistem Aplikasi Jembatan Keledai

   Pada tahap ini dirancang *flowchart*, *data flow diagram*, dan diagram-diagram lain yang dibutuhkan agar pembangunan aplikasi menjadi lebih fokus dan teratur. Selain itu dirancang juga tampilan aplikasi serta fungsi-fungsi yang akan disediakan pada aplikasi jembatan keledai.

4. Pembangunan Aplikasi Jembatan Keledai

Pada tahap ini rancangan yang telah dibuat sebelumnya digunakan untuk pembangunan aplikasi dari awal hingga selesai.

5. Pengujian Aplikasi

   Pada tahap ini aplikasi sudah selesai dibangun tetapi masih banyak *bug and error* yang tidak diketahui. Oleh karena itu dilakukan pengujian aplikasi agar *bug and error* ditemukan dan dapat dibetulkan sehingga aplikasi menjadi semakin baik. Pengujian dilakukan dengan menyebarkan aplikasi tersebut untuk digunakan oleh banyak orang dengan meminta *feedback* jika ada *bug and error* yang ditemukan.

6. Implementasi Program

   Pada tahap ini aplikasi jembatan keledai sudah siap untuk digunakan dan diharapkan sudah tidak ada *bug and error*. Aplikasi jembatan keledai digunakan oleh banyak orang untuk dinilai berapa persen orang puas dan merasa aplikasi yang dibangun bermanfaat.

7. Penulisan Skripsi

   Pada tahap ini, dilakukan penyusunan laporan yang memuat dokumentasi mengenai perancangan, pembuatan, serta hasil dari aplikasi yang telah dibuat dalam suatu karya ilmiah.

3.2 Perancangan Markov Chain Model

Perancangan *Markov Chain Model* dilakukan untuk membentuk sebuah pola kalimat yang bebas tetapi tetap terstruktur, sehingga kalimat yang dibentuk oleh *Markov Chain Model* ini tetap memiliki struktur kalimat yang benar.

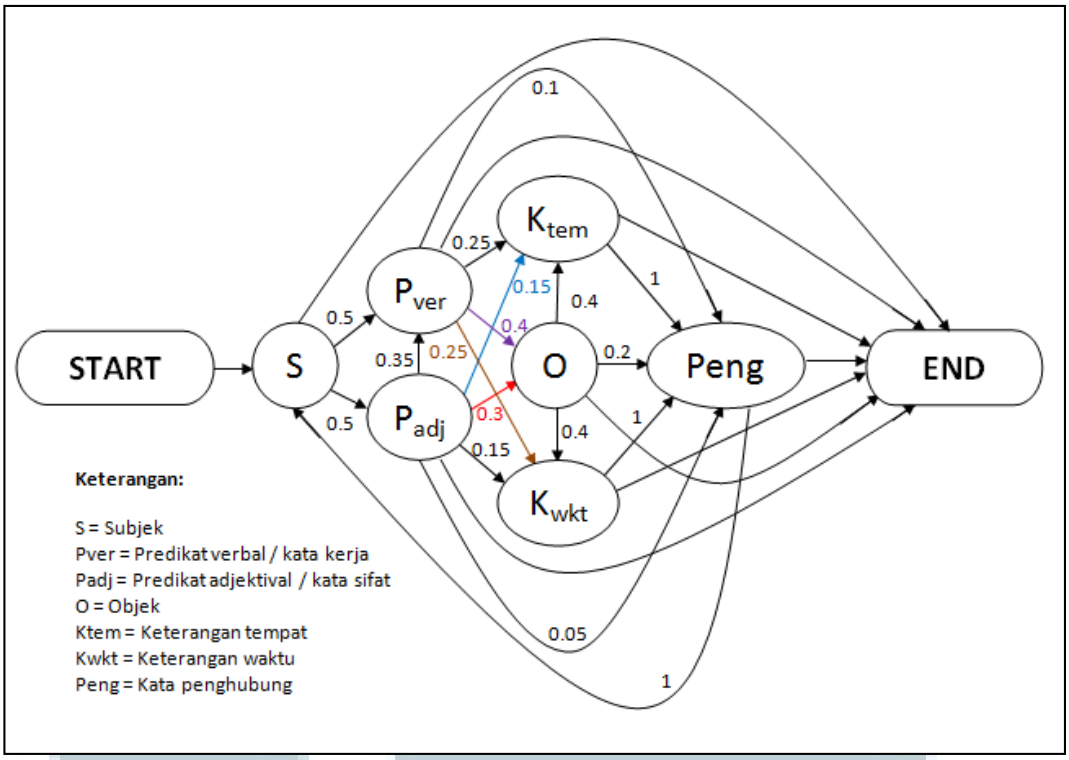
Kalimat dasar dalam bahasa Indonesia memiliki pola kalimat S-P-O-K di mana S adalah subyek, P adalah predikat, O adalah obyek, dan K adalah

keterangan. Keterangan dalam bahasa Indonesia terdapat beberapa jenis, tetapi yang dimasukkan ke dalam *database* aplikasi hanya keterangan waktu dan keterangan tempat saja. Selain itu, predikat dalam bahasa Indonesia terbagi beberapa macam, tetapi yang dimasukkan ke dalam *database* aplikasi hanya predikat verbal dan predikat adjektival. Contoh kalimat dengan pola kalimat S-P-O-K dengan K sebagai keterangan tempat adalah ibu masak nasi di dapur.

Pola kalimat bahasa Indonesia tidak terbatas hanya S-P-O-K saja, tetapi banyak pola-pola kalimat lain yang seringkali diucapkan dalam komunikasi bahasa Indonesia Contoh-contoh kalimat dengan pola lain:

1. Ibu masak di dapur. Kalimat ini memiliki pola S-P-K.
2. Ibu masak nasi. Kalimat ini memiliki pola S-P(kerja)-O.
3. Ibu masak nasi di dapur dan ayah senang. Kalimat ini memiliki pola S-P(kerja)-O-K(tempat)-Penghubung-S-P(sifat).

Karena begitu luasnya pola kalimat yang dapat dibentuk dalam bahasa Indonesia, maka dibuatlah *Markov Chain Model* untuk membentuk pola kalimat secara random tetapi tetap terstruktur. Gambar *Markov Chain Model* dapat dilihat pada gambar 3.1.



Gambar 3.1 *Markov Chain Model* Pola Kalimat

Markov chain pada gambar 3.1 merupakan rancangan pembentukan kalimat yang disesuaikan dengan kebutuhan aplikasi. Kalimat berpola S-P-O-K merupakan kalimat dasar yang sering digunakan, oleh karena itu peluang terbentuknya kalimat berpola S-P-O-K lebih besar dibandingkan pola yang lainnya. Selain itu, tidak adanya peluang menuju ke *state* akhir karena pada setiap *state* dapat menuju ke *state* akhir jika memang kata yang ingin dimasukkan ke dalam kalimat sudah tidak ada.

3.3 Perancangan Aplikasi

Perancangan fungsional aplikasi jembatan keledai ini dilengkapi dengan *flowchart* dan *Data Flow Diagram* (DFD).

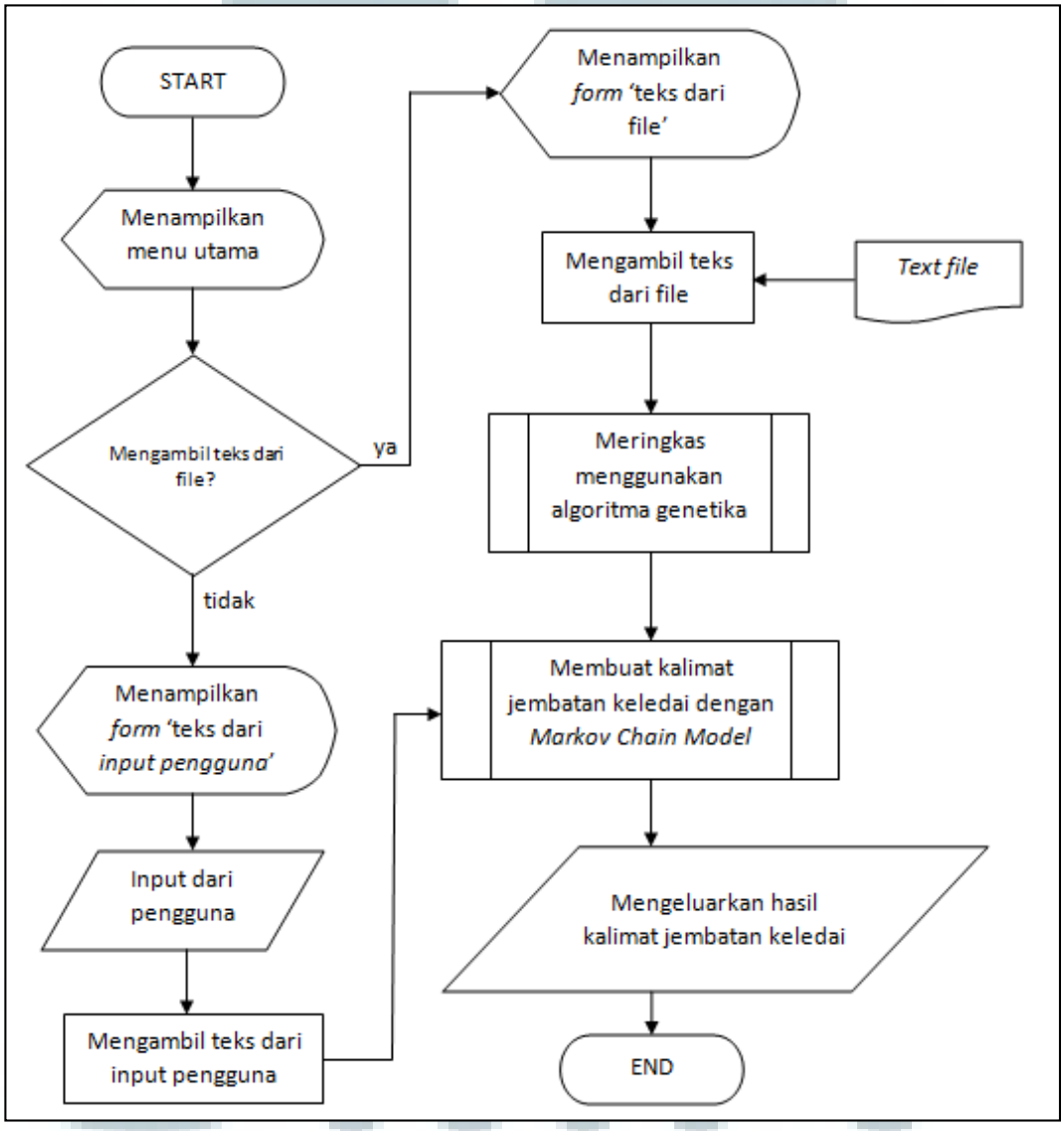
3.3.1 Flowchart

Flowchart merupakan sebuah diagram yang menggambarkan alur proses atau kerja dari aplikasi. *Flowchart* digunakan untuk membantu pembaca lebih mengerti alur proses aplikasi dibandingkan dengan tulisan saja. Pada tahap

perancangan aplikasi ini, terdapat *flowchart* yang menjelaskan alur aplikasi secara umum serta *flowchart* dari algoritma yang digunakan.

A. Flowchart Aplikasi Jembatan Keledai

Pada flowchart ini menjelaskan bagaimana alur aplikasi jembatan keledai mulai dari aplikasi pertama kali dijalankan hingga aplikasi selesai digunakan.



Gambar 3.2 *Flowchart* aplikasi jembatan keledai

B. Flowchart Sub Proses Meringkas Menggunakan Algoritma Genetika

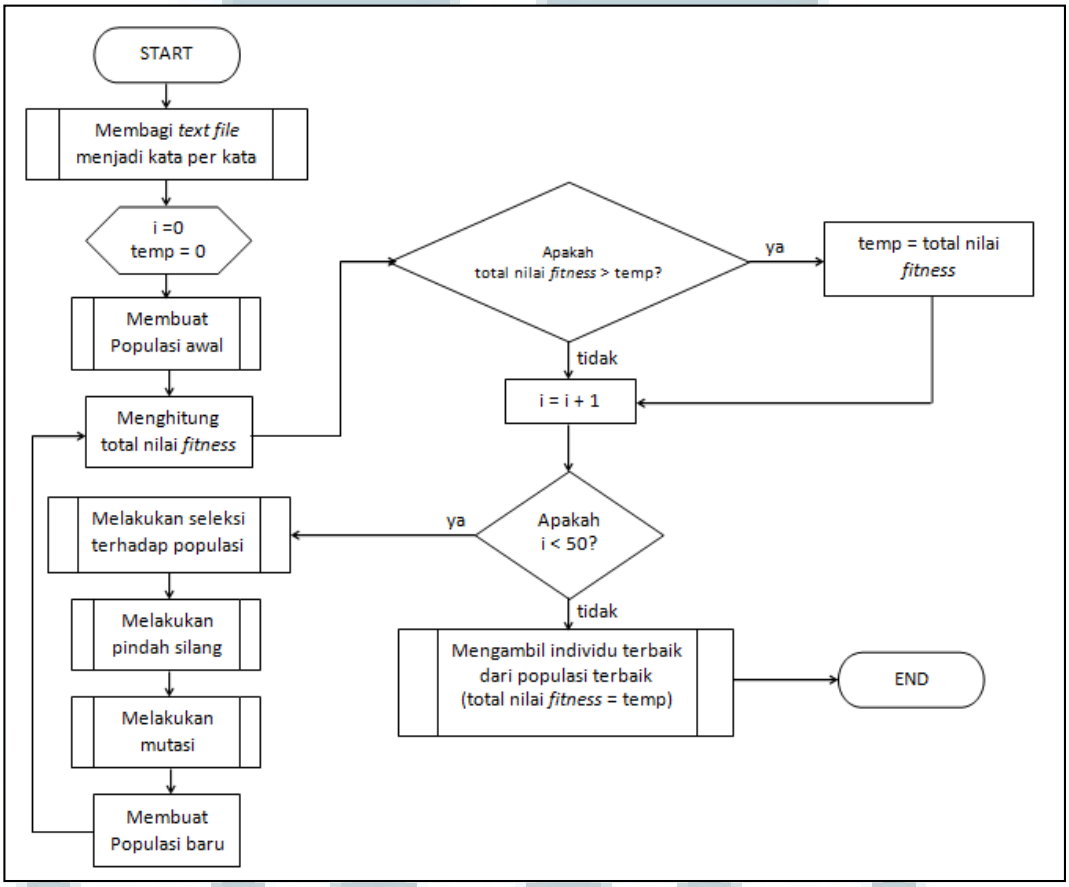
Pada proses meringkas digunakan algoritma genetika. Pada genetika algoritma untuk meringkas bacaan, gen nya adalah id sebuah kata dalam sebuah bacaan, di luar dari kata umum yang terdapat dalam Bahasa Indonesia, seperti tetapi, dan, kemudian, setelah, dan lain-lain. Sebuah individu dibentuk dari sejumlah gen. Karena individu yang merupakan solusi dari peringkasan bacaan, yaitu kata-kata kunci dalam sebuah bacaan, agar tidak terlalu banyak kata kunci yang diperoleh, dan juga tidak terlalu sedikit, maka pada penelitian ini sebuah individu terdiri dari 15 gen. Selain itu, untuk mendapatkan variasi individu yang beragam, tetapi juga mempertimbangkan kecepatan aplikasi dalam melakukan proses algoritma genetika, maka ditetapkan populasi dalam penelitian ini adalah sebanyak 40 individu.

Hal lain yang diperlukan dalam algoritma genetika adalah nilai *fitness*. Berdasarkan rumus 2.1 (total nilai), rumus dari nilai *fitness* adalah jumlah dari rumus 2.2 (nilai judul) dan rumus 2.3 (nilai jumlah) dikalikan bobot masing-masing.

$$\text{Nilai } Fitness \text{ Peringkas Teks} = 10 * \text{Nilai Judul} + 9 * \text{Nilai Jumlah} \quad \text{.....Rumus 3.1}$$

Setelah menentukan rumus nilai *fitness*, maka proses algoritma genetika dapat dimulai, yaitu penghitungan nilai *fitness*, seleksi, pindah silang, mutasi, dan pembentukan populasi baru. *Flowchart* dari genetika algoritma dapat dilihat pada gambar 3.4. Probabilitas untuk pindah silang (*Pc)* adalah 0,8, sedangkan probabilitas untuk mutasi (*Pm*) adalah 0,1. Nilai probabilitas ditetapkan demikian karena mengikuti nilai probabilitas pada umumnya pada penelitian-penelitian lain.

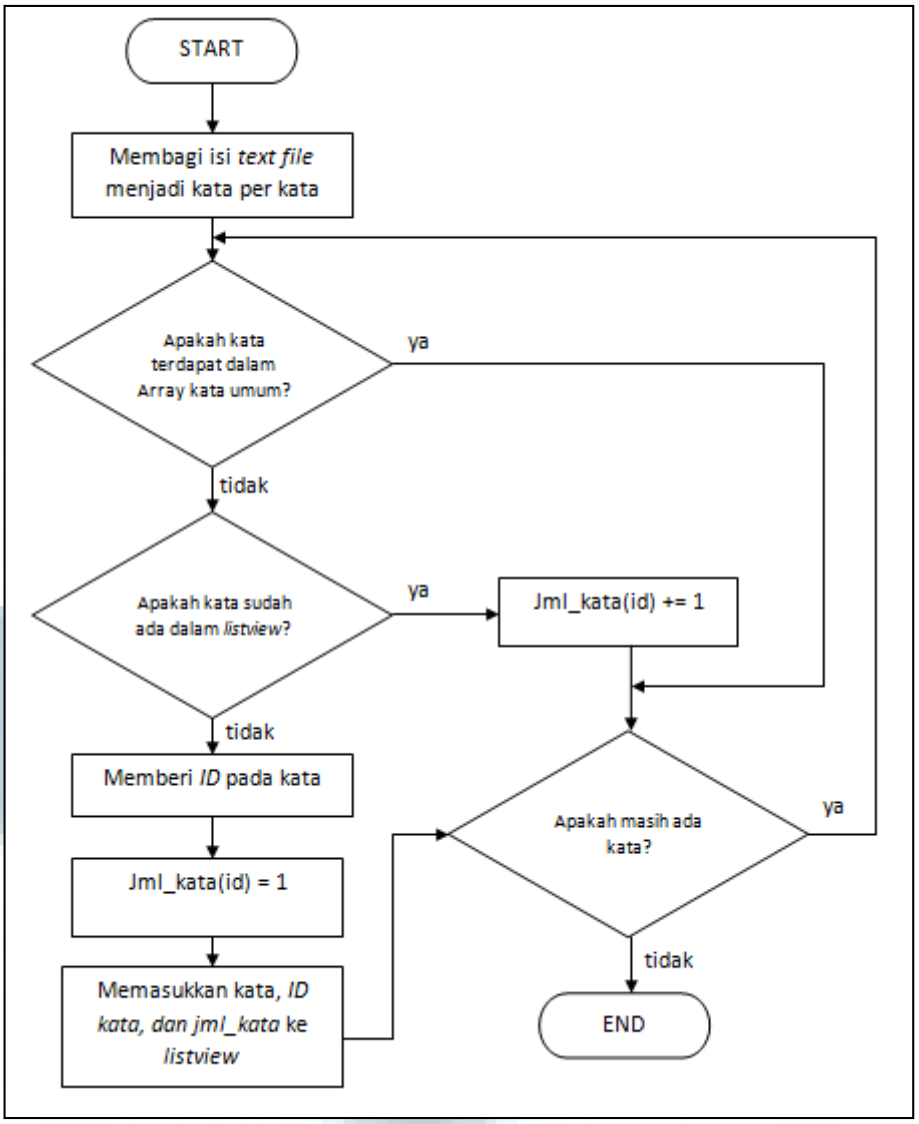
Proses algoritma genetika pada penelitian ini dilakukan sebanyak 50 kali seleksi. Setelah dilakukan beberapa uji coba, nilai *fitness value* relatif tidak semakin meningkat lagi pada tahap antara seleksi ke-40 hingga seterusnya. Oleh karena itu untuk menjaga kecepatan aplikasi ini dalam menjalankan algoritma genetika, maka ditetapkan sebanyak 50 kali dilakukan proses seleksi.



Gambar 3.3 *Flowchart* sub proses meringkas menggunakan genetika algoritma

C. Flowchart Sub Proses Membagi Text File Menjadi Kata per Kata

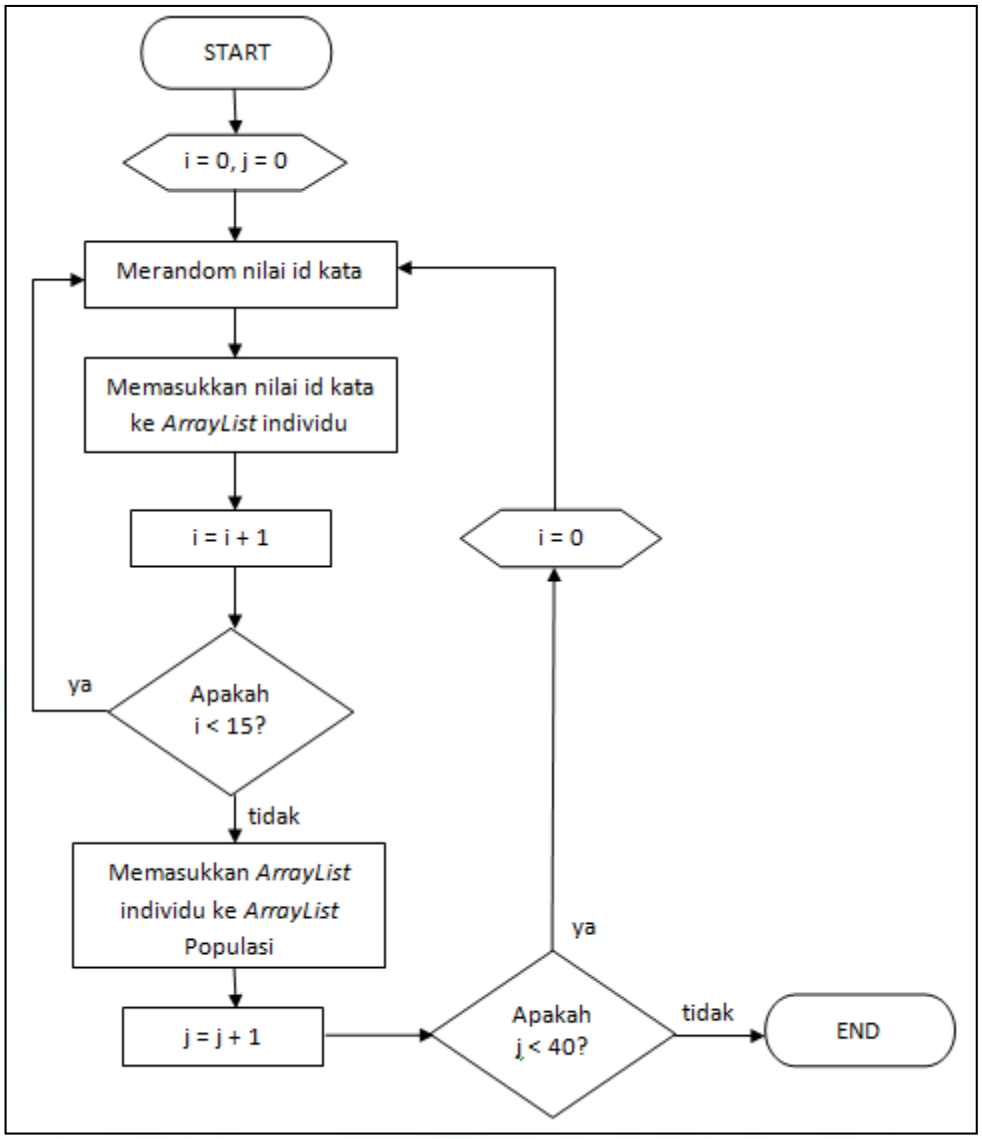
Flowchart ini menjelaskan proses pemotongan sebuah *text file* menjadi kata-kata tunggal. Kata-kata yang dipilih di luar kata-kata umum Bahasa Indonesia.



Gambar 3.4 *Flowchart* sub proses membagi *text file* menjadi kata per kata

D. Flowchart Sub Proses Membuat Populasi Awal

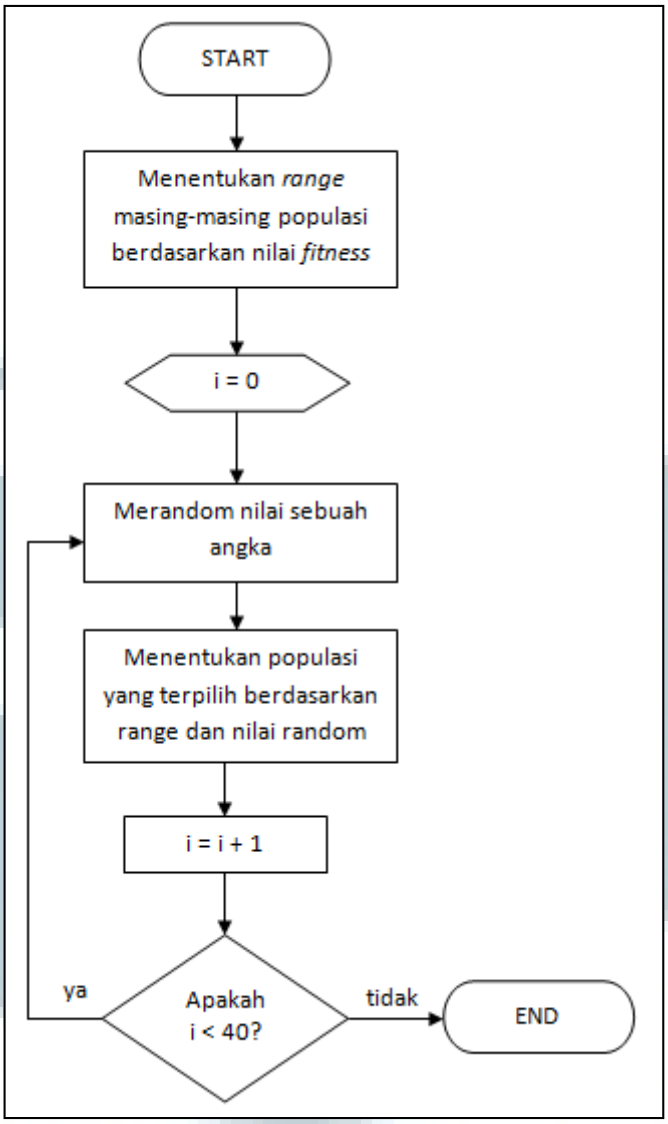
Flowchart ini menjelaskan salah satu proses pada algoritma genetika, yaitu membuat populasi awal. Pada gambar 3.5 dijelaskan bagaimana alur membuat populasi awal untuk meringkas sebuah *text file*. Populasi pada laporan ini adalah kumpulan dari 15 id kata yang dipilih secara acak.



Gambar 3.5 *Flowchart* sub proses membuat populasi awal

E. Flowchart Sub Proses Melakukan Seleksi terhadap Populasi

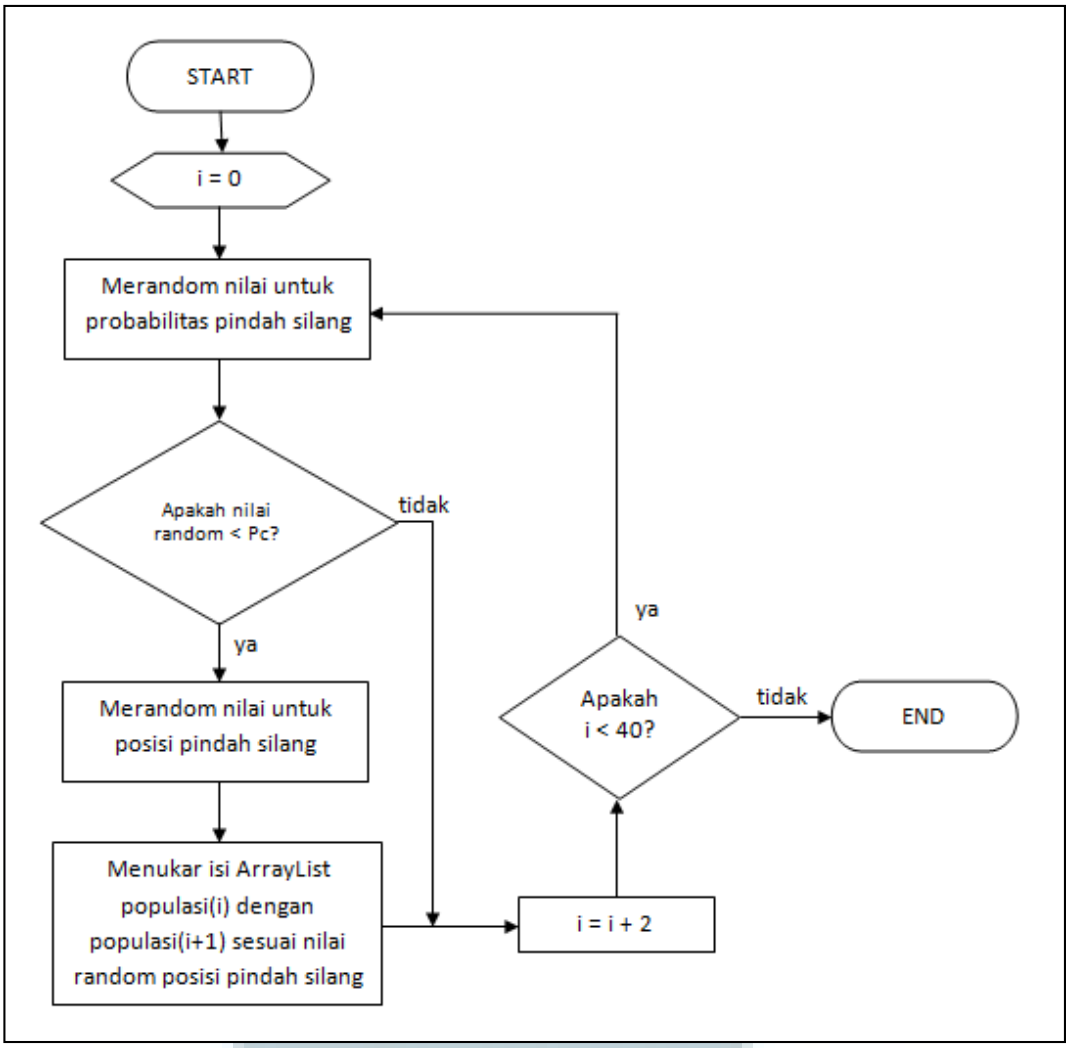
Flowchart ini menjelaskan proses seleksi terhadap populasi pada algoritma genetika. Proses seleksi ini dilakukan dengan sistem *Roullete Wheel*, dimana setiap populasi memiliki *range* tertentu sesuai dengan nilai *fitness*nya, dan setiap kali sebuah angka *random* ditentukan, maka dicari angka *random* tersebut berada di *range* populasi mana, dan populasi tersebut akan terpilih menjadi induk untuk dilakukan proses-proses berikutnya.



Gambar 3.6 *Flowchart* sub proses melakukan seleksi terhadap populasi

F. Flowchart Sub Proses Melakukan Pindah Silang

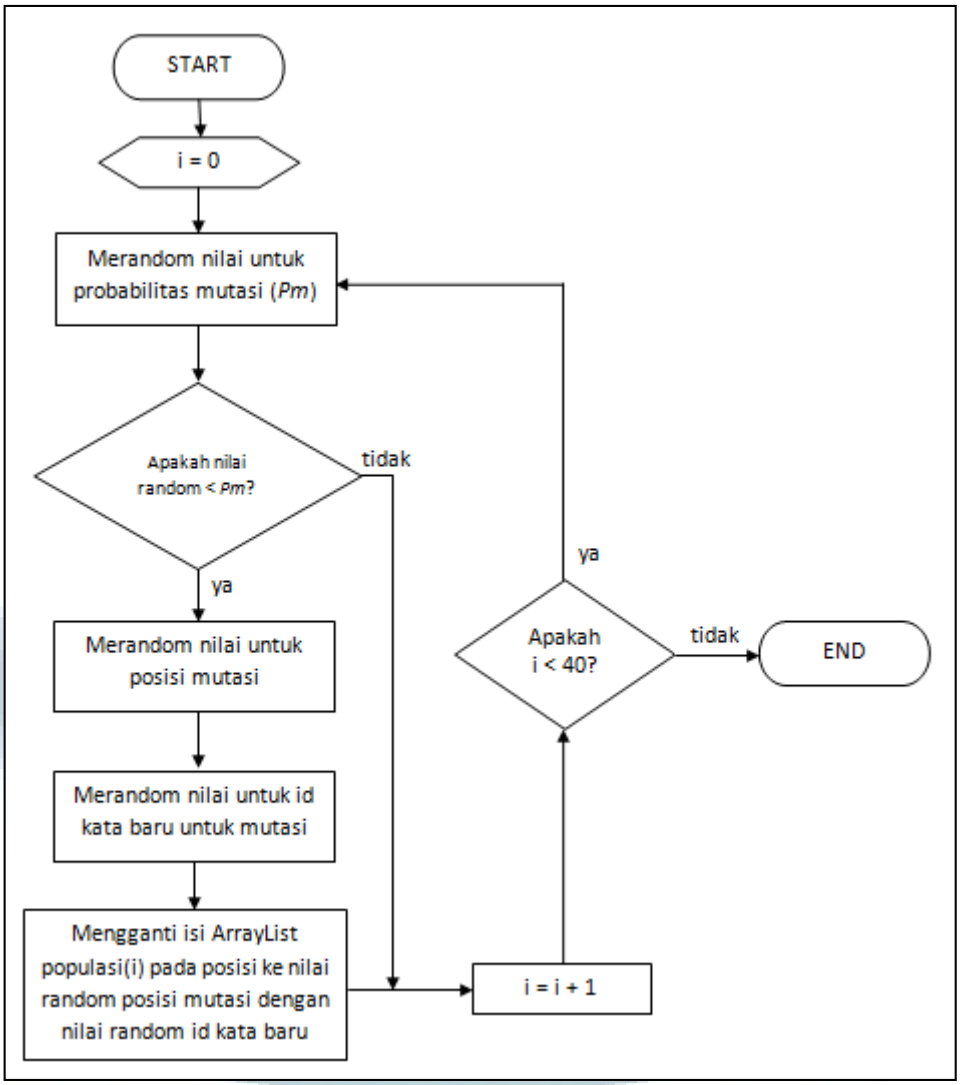
Flowchart ini menjelaskan bagaimana proses pindah silang pada algoritma diterapkan dalam peringkasan teks. Probabilitas pindah silang (Pc) adalah 0.8, jadi jika angka random di bawah 0.8 maka dilakukan pindah silang. Pindah silang dilakukan oleh populasi ke i dengan populasi ke i+1.



Gambar 3.7 *Flowchart* sub proses melakukan pindah silang

G. Flowchart Sub Proses Melakukan Mutasi

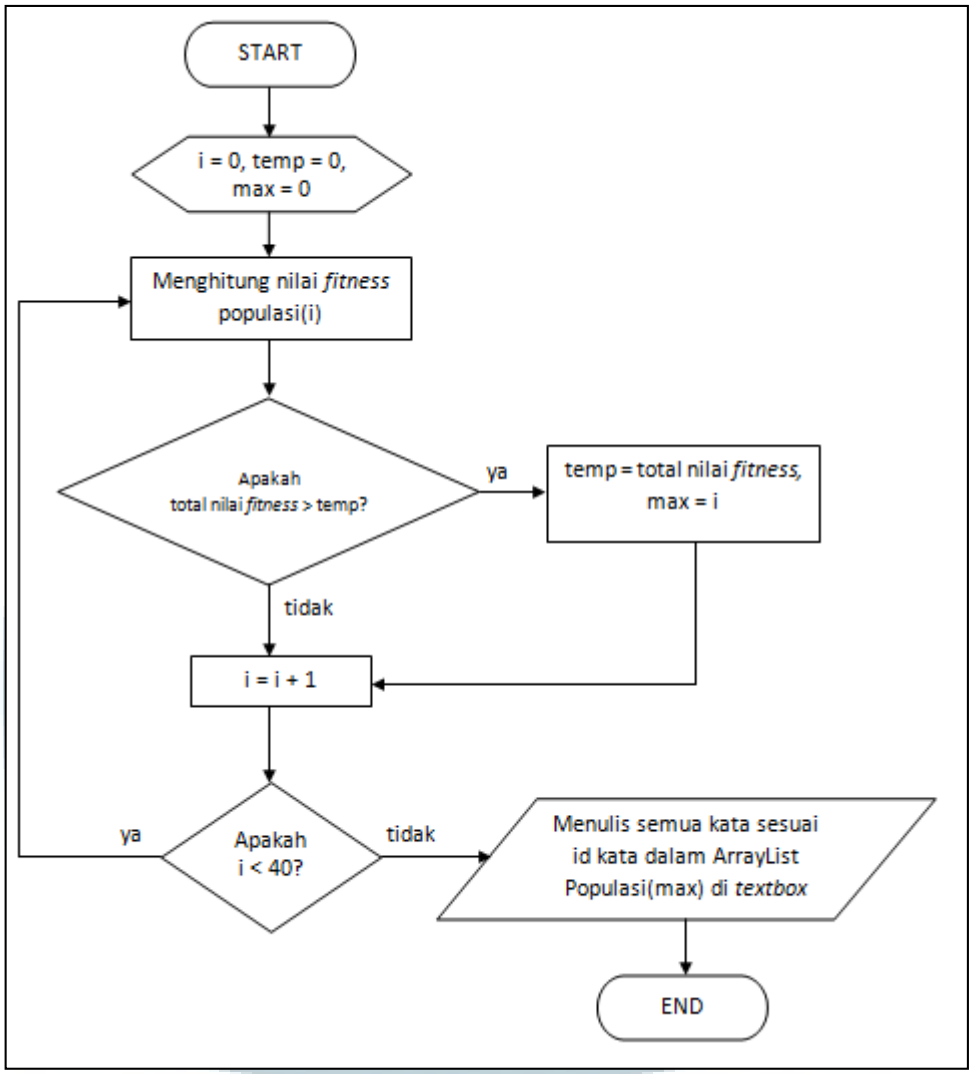
Flowchart ini menjelaskan bagaimana proses mutasi pada algoritma genetika digunakan dalam peringkasan teks. Probabilitas mutasi (*Pm*) pada aplikasi ini adalah 0,1. Jika angka random di bawah 0,1 maka akan dilakukan proses mutasi. Proses mutasi dilakukan pada populasi ke i dengan posisi gen yang terkena mutasi merupakan angka random juga.



Gambar 3.8 *Flowchart* sub proses melakukan mutasi

H. Flowchart Sub Proses Mengambil Individu Terbaik dari Populasi Terbaik

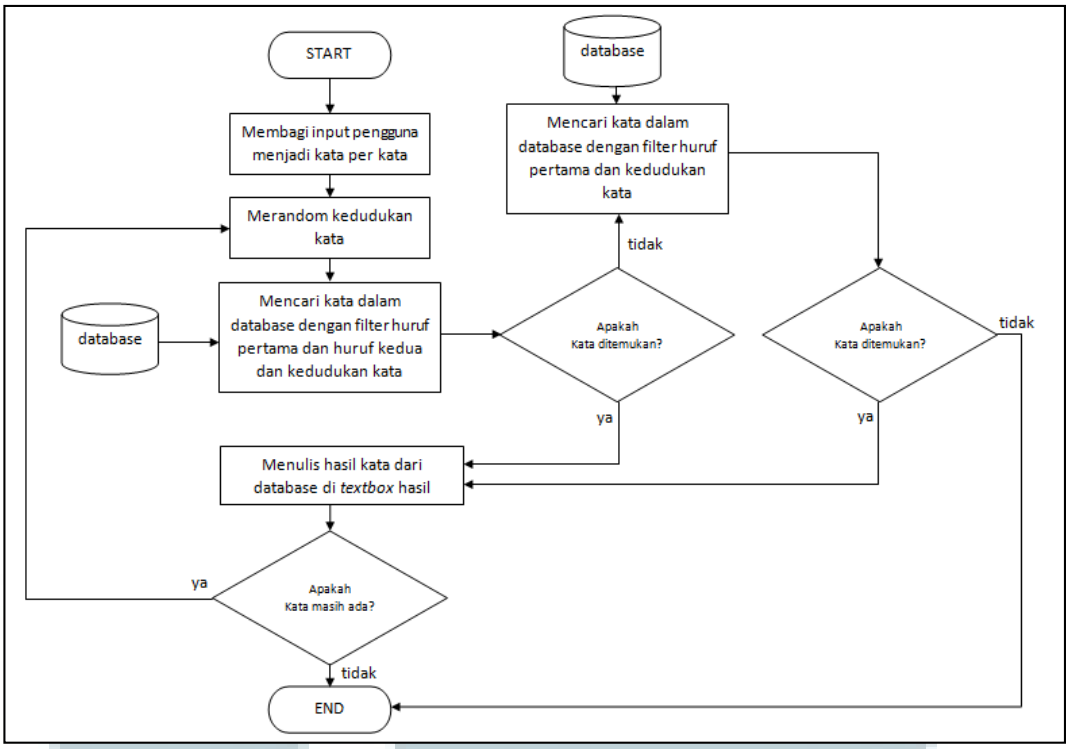
Flowchart ini menjelaskan bagaimana pada akhir proses peringkas kata, sebuah penyelesaian terbaik untuk peringkas kata dihasilkan. Setelah melalui 50 kali proses algoritma genetika, didapatkan sebuah populasi terbaik. Dari populasi terbaik ini, diambil juga individu terbaik yang dapat mewakili solusi dari peringkas kata ini.



Gambar 3.9 *Flowchart* sub proses mengambil individu terbaik dari populasi terbaik

I. Flowchart Sub Proses Membuat Jembatan Keledai Menggunakan Markov Chain Model

Flowchart sub proses meringkas menggunakan *Markov Chain Model* menjelaskan bagaimana sebuah kalimat jembatan keledai dapat dibentuk dengan menggunakan *Markov Chain Model* dengan *state-state* yang telah digambarkan pada gambar 3.2. *Flowchart* dapat dilihat pada gambar 3.10.



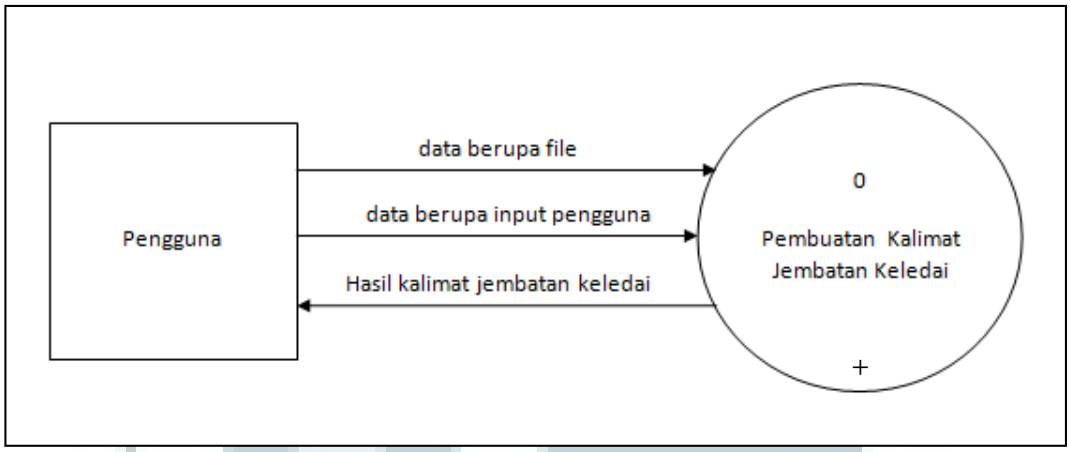
Gambar 3.10 *Flowchart* sub proses membuat jembatan keledai menggunakan *Markov Chain Model*

3.3.2 Data Flow Diagram

Data Flow Diagram (DFD) merupakan representasi grafik dari aliran data pada sebuah sistem informasi. DFD digunakan untuk memvisualisasikan bagaimana sebuah sistem beroperasi.

A. Context Diagram

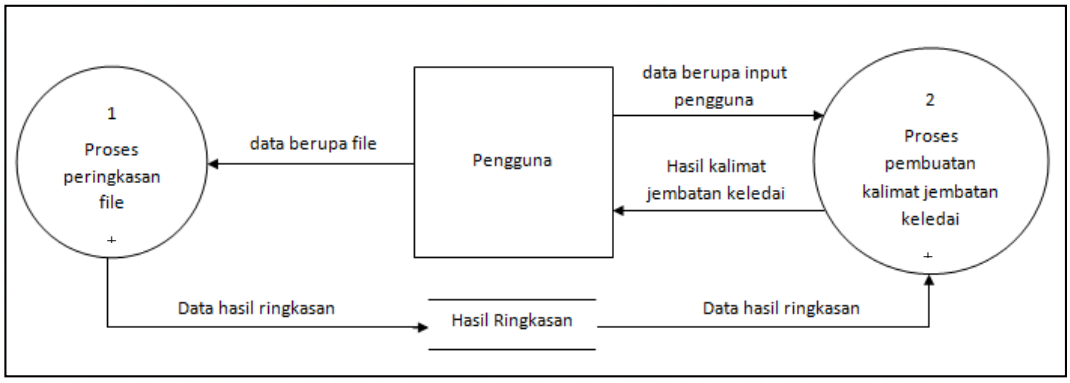
Seperti yang terlihat pada gambar 3.11, terdapat interaksi antara pengguna/*user* dengan aplikasi jembatan keledai. Interaksi yang dilakukan oleh pengguna adalah memasukkan data berupa *text files*, atau dapat berupa *input* langsung dari pengguna. Setelah proses pembuatan kalimat jembatan keledai selesai dilakukan, maka sistem akan menampilkan hasil kalimat jembatan keledai tersebut.



Gambar 3.11 *Context diagram*

B. Diagram Level 1

Diagram level 1 merupakan diagram turunan dari *context diagram* pada gambar 3.11. Pada diagram level 1 ini terdapat dua buah proses, yaitu proses proses peringkasan *file* dan proses pembuatan kalimat jembatan keledai. Proses peringkasan *file* dilakukan kepada data *file* yang dimasukkan oleh pengguna. Setelah diringkas maka didapatkan sebuah data yaitu hasil data ringkasan. Hasil data ringkasan ini kemudian digunakan oleh proses pembuatan kalimat jembatan keledai untuk dibuat kalimat jembatan keledainya. Setelah selesai diproses, maka hasil kalimat jembatan keledai ditampilkan ke pengguna.

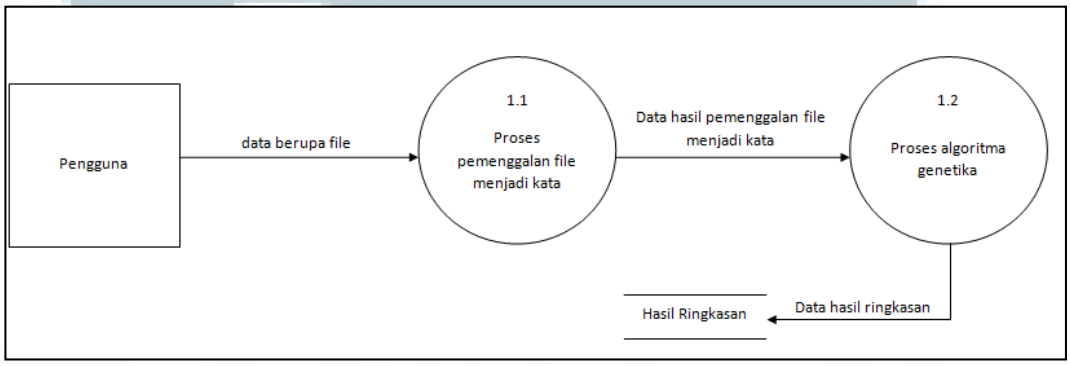


Gambar 3.12 Diagram level 1

C. Diagram Level 2

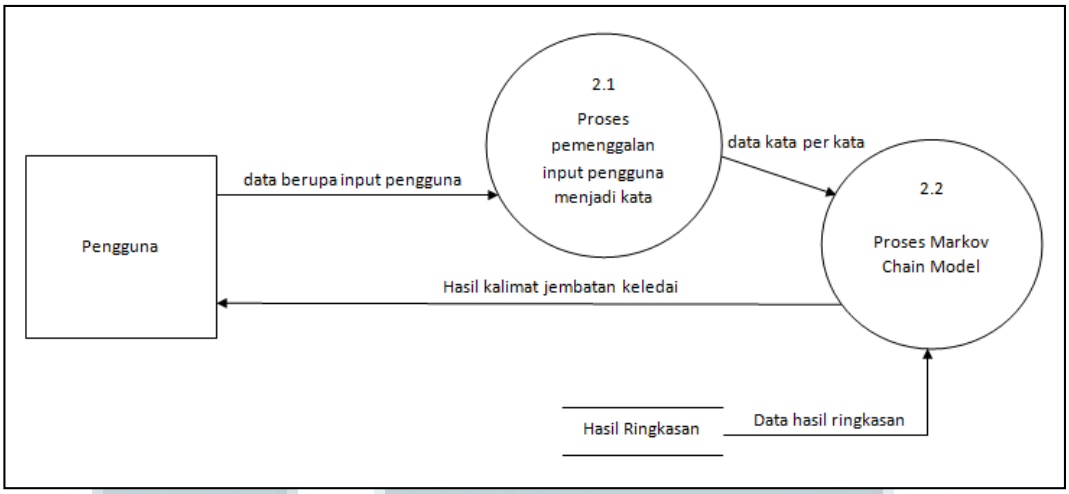
Terdapat dua buah diagram pada level 2 yang berasal dari proses 1 dan proses 2 pada digram level 1. Kedua diagram ini menjelaskan secara lebih rinci proses peringkasan *file* dan proses pembuatan kalimat jembatan keledai.

Pada proses peringkasan *file* secara lebih rinci dijabarkan terdapat proses pemenggalan *file* menjadi kata dan proses algoritma genetika. Data berupa *file* yang dimasukkan oleh pengguna dipecah-pecahkan dahulu menjadi kata-kata tunggal dan sudah disaring agar kata-kata penghubung dan kata umum seperti yang, bahwa, jadi, kemudian, dan lain lain tidak ikut ke dalam proses selanjutnya. Setelah didapat data hasil pemenggalan *file* menjadi kata, dilakukan proses algoritma genetika yang kemudian menghasilkan data hasil ringkasan.



Gambar 3.13 Diagram level 2 proses 1

Pada proses pembuatan jembatan keledai, data hasil ringkasan dari proses algoritma genetika dimasukkan ke dalam proses *Markov Chain Model*. Selain data hasil ringkasan, proses *Markov Chain Model* dapat menerima data langsung dari pengguna. Dari proses *Markov Chain Model* inilah dihasilkan kalimat jembatan keledai.



Gambar 3.14 Diagram level 2 proses 2

3.3.3 Struktur Tabel

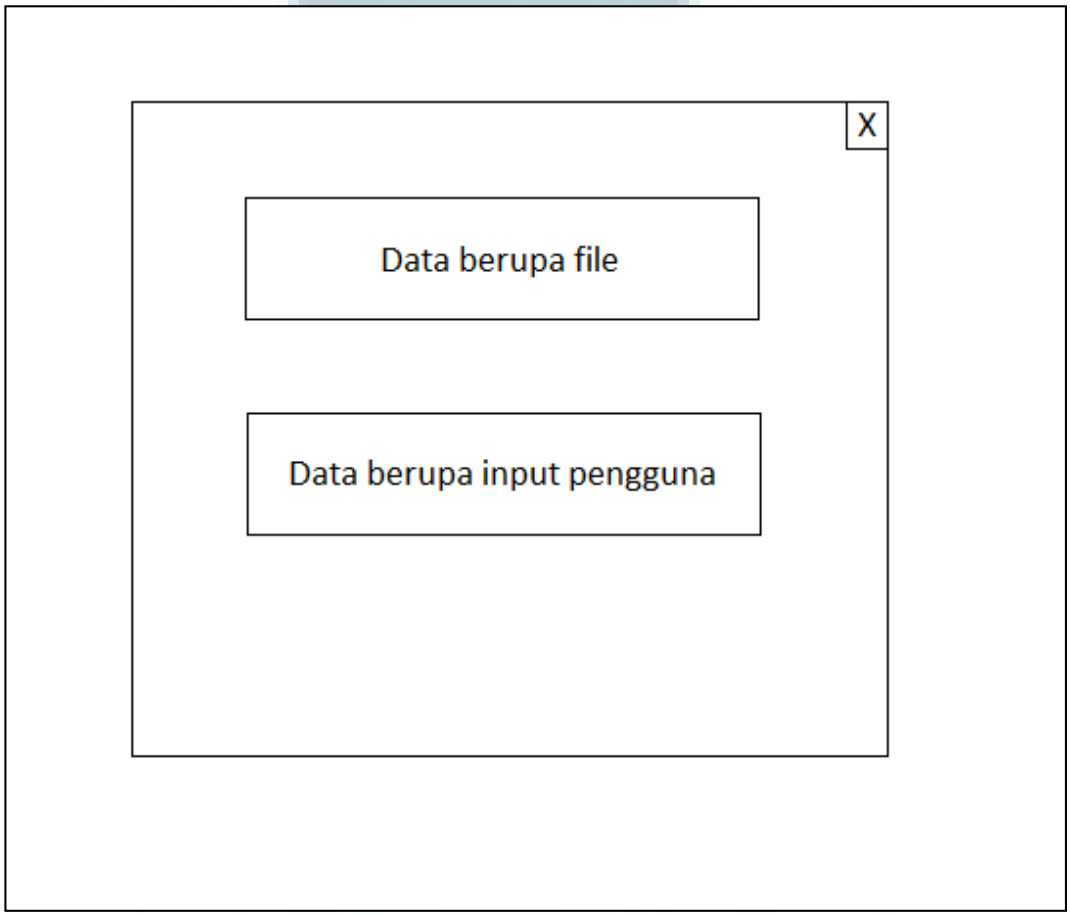
Tidak adanya tabel yang terhubung antara satu sama lain, membuat aplikasi ini tidak memiliki *entity relationship diagram.* Oleh karena itu, hanya dituliskan struktur tabel pada aplikasi ini. Hanya terdapat sebuah tabel dalam aplikasi ini, yaitu tabel kosakata yang digunakan menjadi sumber kata-kata ketika membuat jembatan keledai.

Tabel 3.1 Keterangan tabel Kosakata

Kolom	Tipe Data	Keterangan
id	integer	Id dari kata
kata	varchar(20)	Kata yang akan digunakan untuk membuat kalimat jembatan keledai
kedudukan	varchar(2)	Kedudukan sebuah kata dalam kalimat

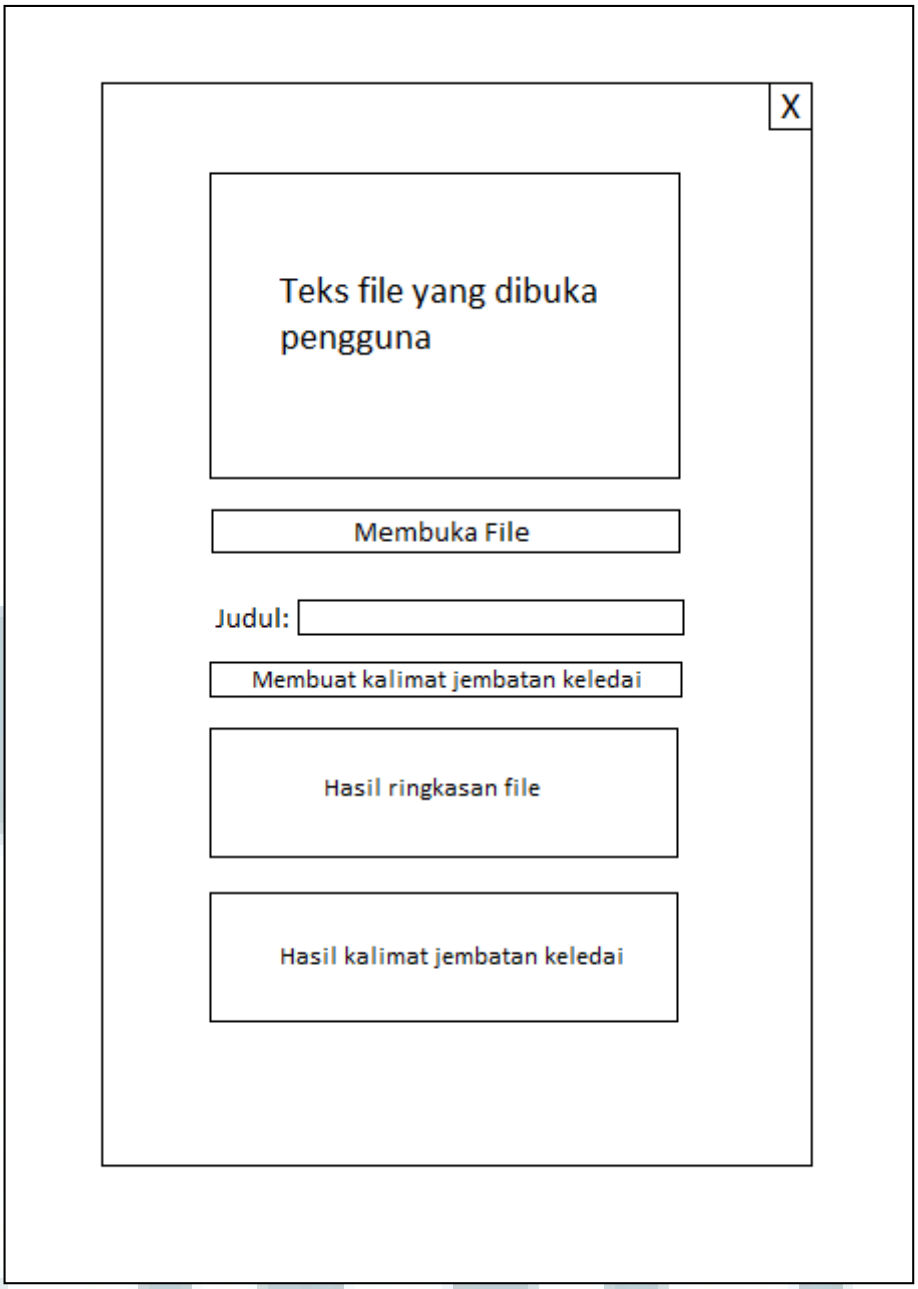
3.3.4 Perancangan Antarmuka Pengguna

Pada perancangan antarmuka pengguna ini, terdapat tiga antarmuka yang digunakan. Antarmuka pertama adalah menu utama dimana pengguna memilih akan memasukkan data dari *file* atau memasukkan *input* sendiri.



Gambar 3.15 Antarmuka menu utama

Antarmuka kedua adalah antarmuka jika pengguna memilih memasukkan data berupa *file*. Pertama *user* harus membuka *file* yang akan diringkas dengan menekan tombol “Membuka File”. Setelah memasukkan *file*, pengguna harus mengetikkan judul teks dari *file* yang dibuka. Setelah itu pengguna menekan tombol “Membuat Kalimat Jembatan Keledai”. Setelah tombol ditekan, hasil ringkasan dan kalimat jembatan keledai akan ditampilkan.



Gambar 3.16 Antarmuka menu data berupa *file*

Antarmuka ketiga adalah antarmuka jika pengguna memilih memasukkan data berupa *input* langsung dari pengguna. Pada antarmuka ini pengguna tinggal memasukkan kata-kata yang ingin dihafalkan. Kata-kata *input* dari pengguna dipisah menggunakan spasi. Setelah itu pengguna menekan tombol "Membuat

Kalimat Jembatan Keledai". Setelah tombol ditekan hasil kalimat jembatan keledai akan ditampilkan.

Gambar 3.17 Antarmuka menu data berupa *input* pengguna