BAB 3

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Analisis Kebutuhan

   Melakukan analisa kebutuhan awal yang diperlukan untuk mengimplementasikan sistem *spell checker* bahasa Indonesia. Menentukan *software* dan *library* yang dibutuhkan untuk dapat mengimplementasikan model yang digunakan dan membangun aplikasi demonstrasi.

2. Telaah Literatur

   Memahami literatur dan teori yang berkaitan dan diperlukan dalam implementasi sistem *spell checker* bahasa Indonesia, seperti *typography error*, algoritma *Symspell*, model *Bayesian network*, dan optimisasi *dynamic programming*.

3. Pengumpulan Data dan Perancangan Korpus

   Mengumpulkan data berupa kata (unigram) dan pasangan kata (bigram) melalui *web scraping* situs berita dan buku elektronik. Menentukan tempat dan format penyimpanan korpus yang sudah diperkaya data hasil *scraping*.

4. Perancangan Sistem

   Membuat *flowchart* perancangan sistem berdasarkan informasi yang didapat pada telaah literatur. Membuat rancangan antarmuka pengguna untuk keperluan demonstrasi dan uji coba aplikasi.

5. Implementasi

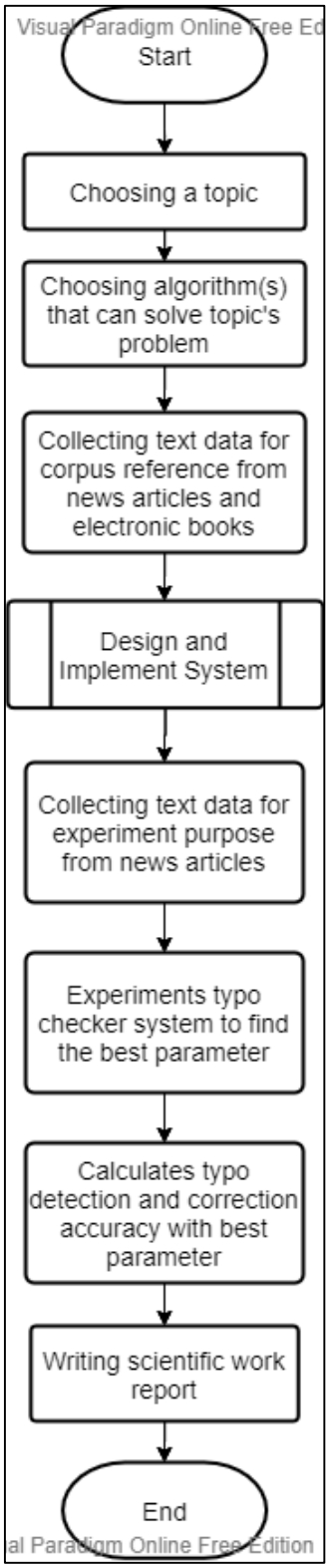
   Mengimplementasikan hasil rancangan sistem yang telah dibuat pada tahap sebelumnya menggunakan bahasa pemrograman Python 3 dan Jupyter Notebook.

6. Uji Coba Aplikasi dan Evaluasi

   Pengujian aplikasi dilakukan dengan menginput teks bacaan yang sudah memenuhi standar KBBI beserta versi bacaan yang sama yang memiliki kesalahan penulisan kata. Teks bacaan yang digunakan merupakan artikel daring yang didapatkan dari situs berita Kompas.com. Performa model *spell checker* dilihat berdasarkan hasil koreksi yang diberikan, akurasi deteksi, dan koreksi yang didapat. Evaluasi dilakukan dengan menguji sistem *spell checker* dalam mengoreksi berbagai jenis *typo* menggunakan berbagai variasi nilai parameter *depth search* dan *suggestions limit*.

7. Penulisan Laporan Karya Ilmiah

   Penulisan laporan sebagai salah satu bentuk dokumentasi penelitian yang dilakukan dari awal hingga penulisan hasil dan kesimpulan. Laporan karya ilmiah ini diharapkan dapat menjadi sarana ilmu pengetahuan dan referensi bagi penelitian selanjutnya.



Gambar 3.1 Flowchart prosedur penelitian

Gambar 3.1 merupakan gambar *flowchart* prosedur penelitian yang dimulai dari pemilihan topik dan identifikasi masalah, pemilihan algoritma yang mampu menyelesaikan masalah yang ditemukan, pengumpulan data teks berita untuk dijadikan referensi dalam pembentukan korpus, mendesain dan mengimplementasi sistem *spell checker*, melakukan eksperimen parameter sistme *spell checker*, dan penulisan laporan karya ilmiah.

3.2 Perancangan Korpus

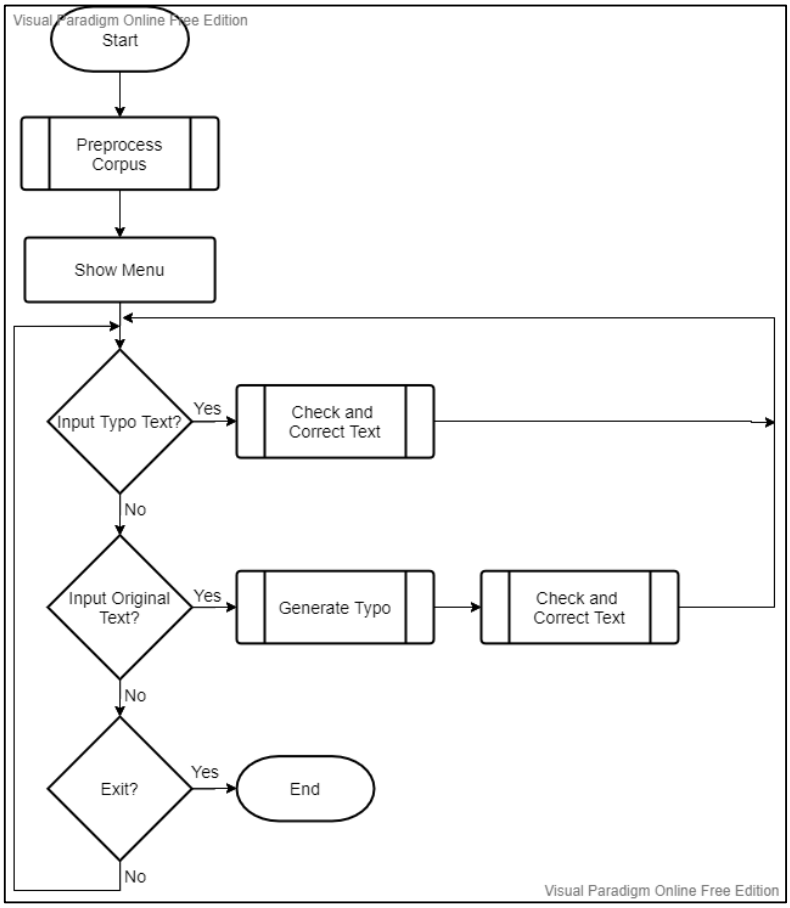
Daftar kata yang digunakan untuk membentuk korpus dikumpulkan melalui *web scraping* 12.750 artikel pada situs berita Kompas.com, 23.000 artikel pada situs berita Republika.co.id, dan 50 buku elektronik dengan format ekstensi *pdf*. Kategori artikel dan buku elektronik yang dikumpulkan tidak dibatasi. *Web scraping* diimplementasi dengan bahasa pemrograman Python dengan bantuan *library requests* dan *BeautifulSoup*.

Korpus dirancang dalam bentuk 1-gram (unigram) dan 2-gram (bigram) yang disimpan dalam bentuk *csv* (*Comma-Separated Values*). Unigram dibentuk dengan menyimpan seluruh kata pada hasil *web scraping* yang sah berdasarkan KBBI daring beserta frekuensi kata tersebut. Jumlah kata yang terkumpulkan pada unigram adalah sebanyak 33.283 kata. Bigram dibentuk dengan menyimpan seluruh pasangan dua kata yang bersebelahan pada hasil *web scraping* di mana kedua kata tersebut terdapat pada unigram yang telah dikumpulkan sebelumnya. Jumlah pasangan kata yang terkumpulkan pada bigram adalah sebanyak 445.328 pasangan kata.

3.3 Perancangan Sistem

Sistem dirancang menggunakan *flowchart* yang terdiri dari, *flowchart* utama sistem *spell checker*, *flowchart* Preprocess Corpus, *flowchart* Check and Correct Text, dan *flowchart* Generate Typo.

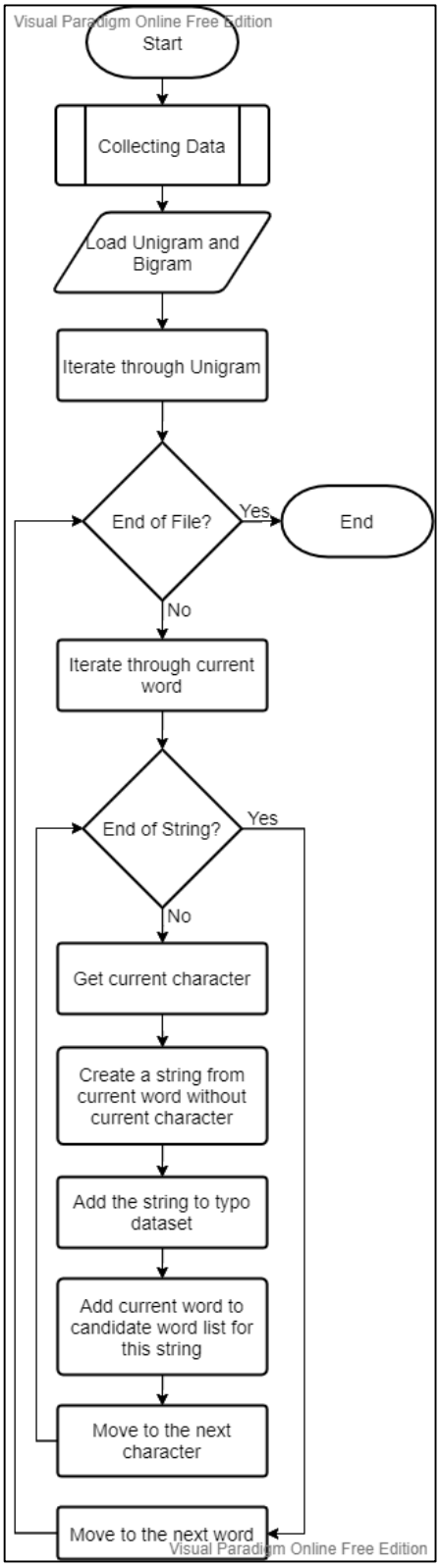
3.3.1 Flowchart Utama



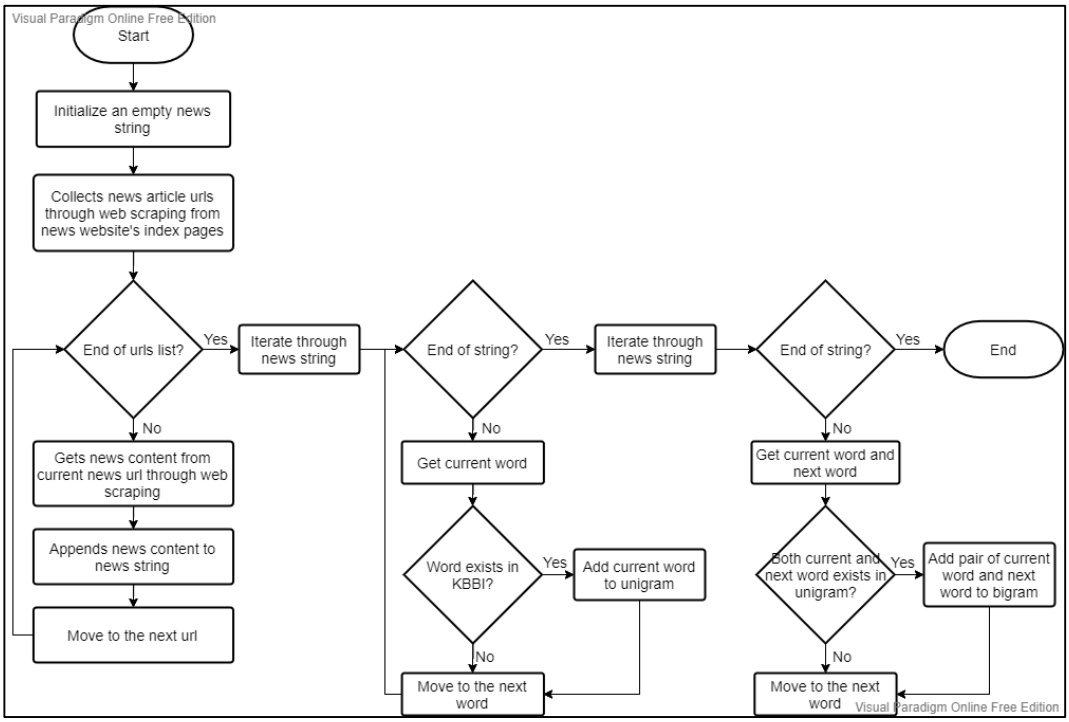
Gambar 3.2 Flowchart utama sistem *spell checker*

Gambar 3.2 merupakan gambar *flowchart* utama alur kerja sistem *spell checker*. Alur kerja sistem diawali dengan melakukan praproses korpus sebelum halaman utama aplikasi ditampilkan. Fitur utama pada sistem *spell checker* adalah pengecekan dan pengoreksian teks *typo* dan *typo generator* dengan parameter yang dapat dikonfigurasi kapanpun.

3.3.2 Flowchart Preprocess Corpus



Gambar 3.3 Flowchart *preprocess corpus*



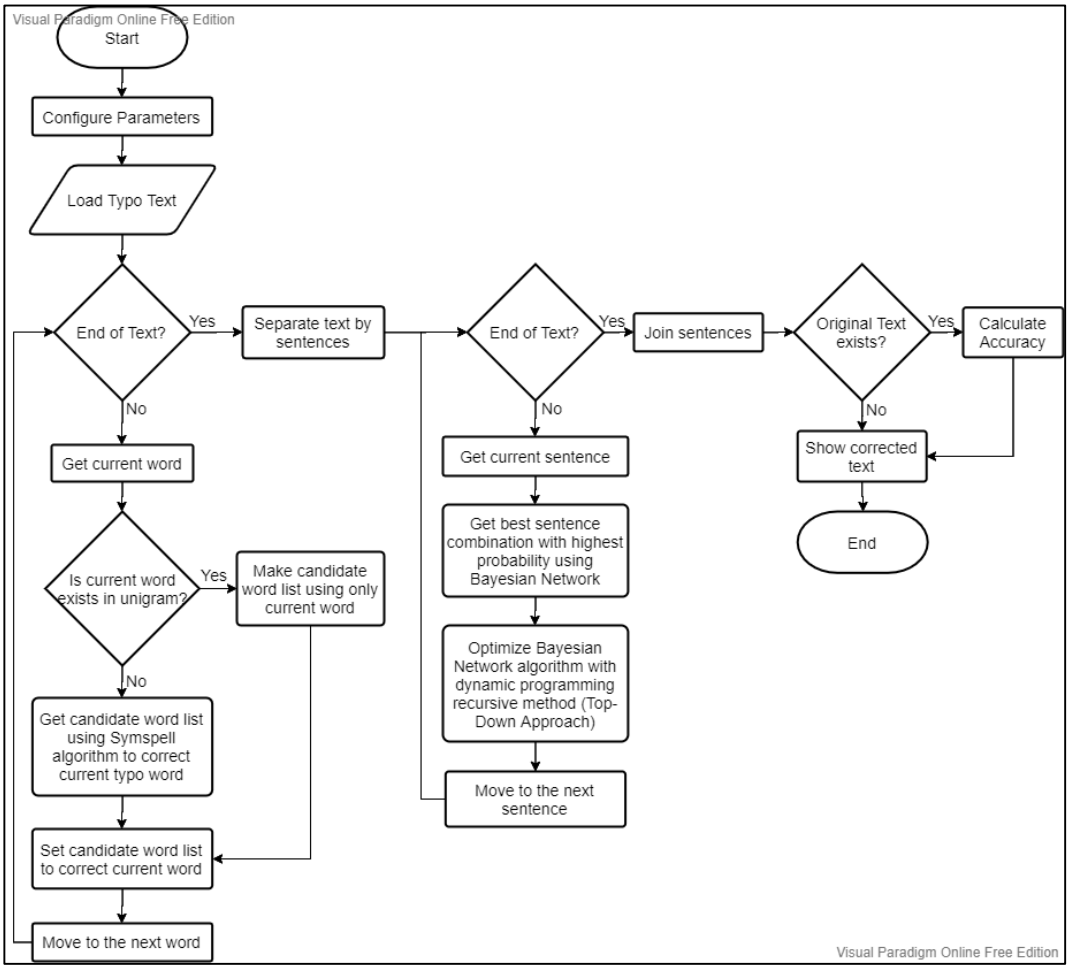
Gambar 3.4 Flowchart pengumpulan data

Gambar 3.3 merupakan gambar *flowchart* praproses korpus yang terdiri dari proses pengumpulan data, proses impor daftar unigram kata dan bigram kata, dan implementasi algoritma *Symspell* untuk membangun dataset *typo*. Algoritma *Symspell* yang dilakukan pada korpus hanya sampai pada iterasi pertama untuk memprioritaskan kandidat kata pengganti dengan jarak *edit* 1 (satu) atau 2 (dua) dari kata *typo* yang dihadapi dan untuk meminimalisir waktu komputasi pada praproses korpus.

Gambar 3.4 merupakan gambar *flowchart* pengumpulan data berupa teks bacaan dari situs berita dan buku elektronik untuk dijadikan referensi pada pembentukan korpus (unigram dan bigram). Pembentukan unigram dilakukan dengan pengecekan ketersediaan kata pada KBBI daring. Jika suatu kata tersedia pada KBBI, kata tersebut beserta frekuensi kemunculannya akan dimasukkan ke

dalam unigram. Pembentukan bigram dilakukan dengan pengecekan ketersediaan kedua kata pada unigram untuk setiap pasangan kata yang ditemukan. Jika kedua kata tersebut tersedia pada unigram, pasangan kata tersebut beserta frekuensi kemunculannya akan dimasukkan ke dalam bigram.

3.3.3 Flowchart Check and Correct Text

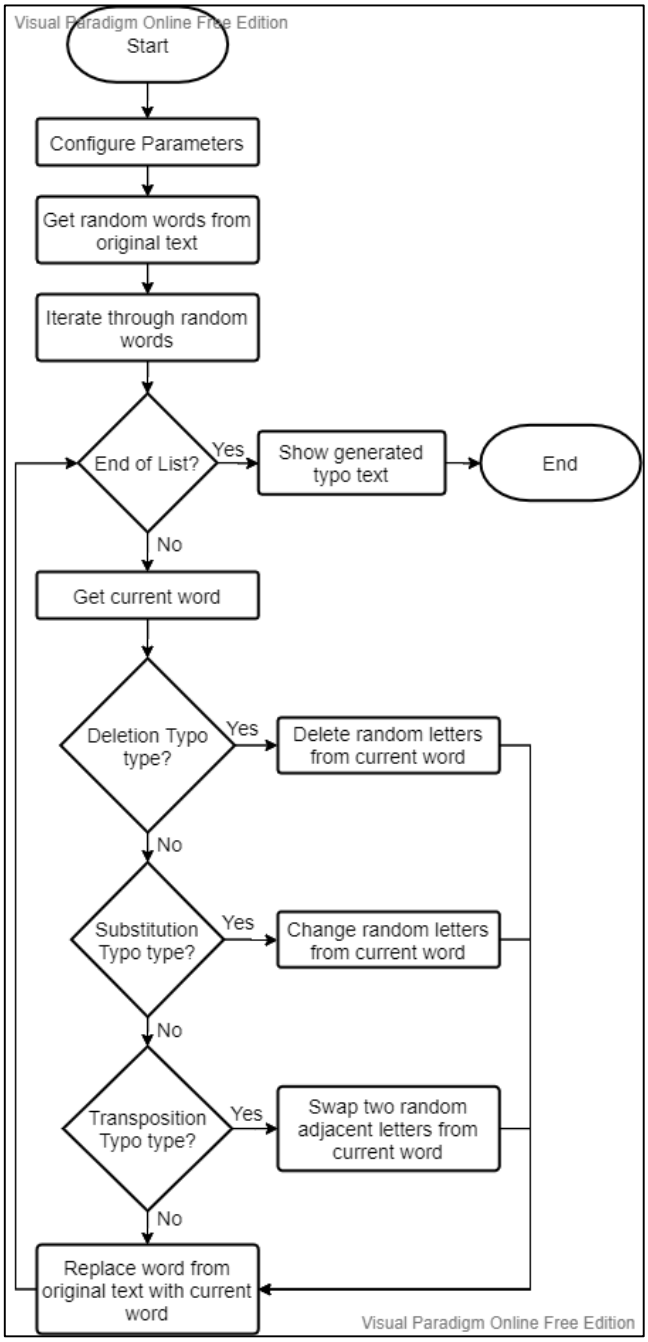


Gambar 3.5 Flowchart *check and correct text*

Gambar 3.5 merupakan gambar *flowchart* proses pengecekan dan pengoreksian teks *typo* oleh sistem dan menghasilkan teks hasil koreksi sebagai output akhir dari proses ini. Pertama, sistem akan menghasilkan daftar kandidat kata

koreksi untuk menggantikan setiap kata pada teks. Jika suatu kata terdeteksi sebagai kata *typo*, daftar kandidat kata untuk mengoreksi kata *typo* tersebut akan dihasilkan oleh algoritma *Symspell*. Jika suatu kata terdeteksi sebagai kata yang sah, daftar kandidat kata hanya akan berisi kata itu sendiri. Pengecekan kata *typo* dilakukan dengan mengecek ketersediaan kata tersebut pada korpus. Pengoreksian teks *typo* akan dilakukan untuk setiap kalimat dalam teks tersebut. Pencarian kalimat terbaik dilakukan dengan algoritma *Bayesian network* yang dioptimisasi dengan metode rekursif *dynamic programming* (*top-down approach*). Jika pengguna menginput teks original yang dapat dijadikan bahan perbandingan oleh sistem, maka sistem juga akan menampilkan akurasi deteksi dan koreksi dari teks hasil koreksi sistem terhadap teks original yang diinput.

3.3.4 Flowchart Generate Typo



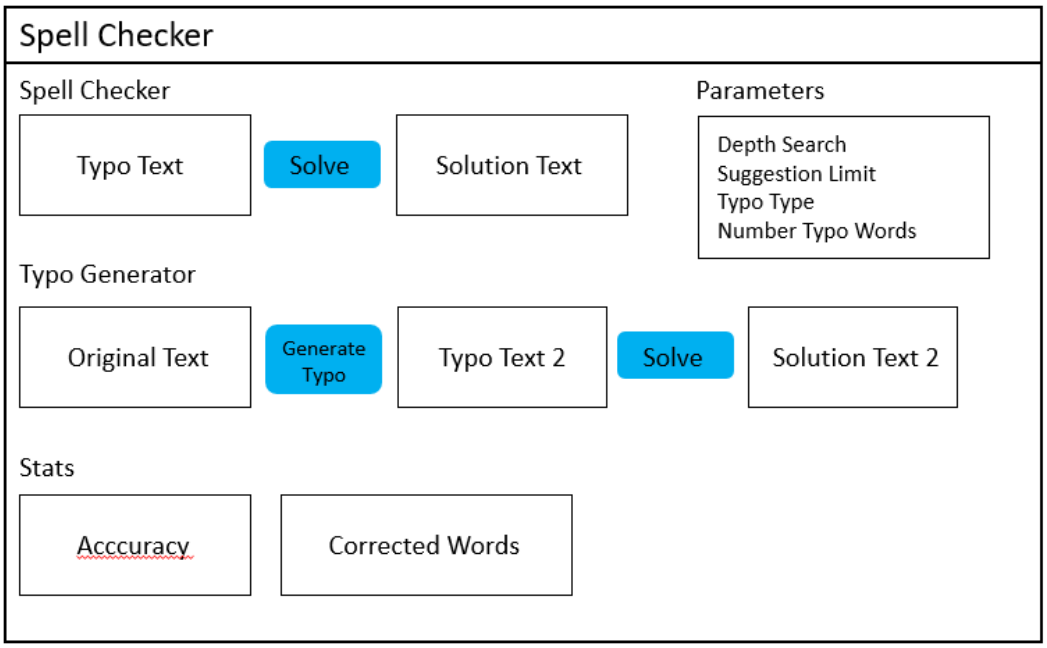
Gambar 3.6 Flowchart *generate typo*

Gambar 3.6 merupakan gambar *flowchart* proses pembuatan teks *typo* berdasarkan teks original yang diinput oleh pengguna. Pengguna dapat mengatur konfigurasi parameter *typo* yang digunakan seperti jumlah kata *typo* yang

diinginkan beserta jenis *typo* yang dihasilkan. Beberapa kata dari teks original akan dipilih secara acak untuk dijadikan kata *typo* berdasarkan parameter yang sudah diatur sebelumnya.

3.4 Perancangan Antarmuka Pengguna

Tampilan aplikasi demonstrasi dirancang menggunakan rancangan antarmuka pengguna yang memungkinkan pengguna untuk berinteraksi dengan sistem *spell checker*. Aplikasi demonstrasi hanya memiliki satu halaman. Gambar 3.7 menunjukkan rancangan antarmuka pengguna.



Gambar 3.7 Rancangan antarmuka pengguna

Praproses korpus dilakukan saat aplikasi *spell checker* dimulai. Pada tahap ini, aplikasi akan memasuki tahap *loading* dan pengguna tidak dapat melakukan interaksi dengan antarmuka pengguna. Setelah tahap *loading* selesai, aplikasi akan menampilkan halaman utama aplikasi yang dapat dilihat pada Gambar 3.7.

Pengguna dapat menggunakan fitur koreksi kata pada aplikasi dengan memasukkan teks bacaan yang ingin dicek pada kotak “Typo Text” pada bagian Spell Checker dan menekan tombol “Solve”. Proses yang dilakukan oleh tombol “Solve” dapat dilihat pada *flowchart* Check and Correct Text. Hasil koreksi teks bacaan akan ditampilkan pada kotak “Solution Text”. Untuk melihat performa dari model *spell checker* yang digunakan, pengguna dapat memasukkan teks bacaan yang tidak memiliki kesalahan penulisan kata pada kolom “Original Text” dan menekan tombol “Generate Typo”. Proses yang dilakukan oleh tombol “Generate Typo” dapat dilihat pada *flowchart* Generate Typo. Tombol “Generate Typo” akan membuat kesalahan pada penulisan beberapa kata pada teks bacaan yang sudah diinput sebelumnya dan hasil teks bacaan *typo* akan ditampilkan pada kotak “Typo Text 2”. Pengguna juga dapat mengatur bagaimana kesalahan penulisan kata akan dibuat oleh sistem dengan mengatur parameter seperti tipe *typo* dan jumlah kata *typo* pada bagian kanan atas halaman. Kemudian, pengguna dapat menekan tombol “Solve” yang berada di samping kotak “Typo Text 2” untuk mendapatkan hasil koreksi dari teks bacaan *typo* yang sudah di-*generate* pada tahap sebelumnya. Informasi mengenai tingkat akurasi dan kata *typo* yang dikoreksi juga dapat dilihat di bagian “Stats” pada bagian bawah halaman.