BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah sebagai berikut.

a. Telaah Literatur

Dalam studi literatur, pembelajaran mengenai teori-teori yang berhubungan dengan penelitian dilakukan. Teori-teori tersebut antara lain adalah *text mining*, ALBERT, *voice chatbot* Jacob, SQuAD, akurasi, *F-score*, dan *user time*.

b. Analisis dan Perancangan

Pada tahap analisis, dilakukan analisis terhadap metode, data, proses, dan spesifikasi yang digunakan pada *web service*. Proses analisis kebutuhan *web service* diamati dari aplikasi Jacob. Dimulai dari bahasa yang digunakan, fitur yang dimilik, dan lain-lain. Untuk hasil analisisnya dibahas pada Subbab 3.2.

Setelah mengumpulkan literatur dan analisis kebutuhan, perancangan *web service* dibuat diagram *web service model* dan *flowchart* untuk mendeskripsikan proses dari *web service* yang dikembangkan. Hasil perancangan diagram dapat dilihat pada Subbab 3.3.

c. Implementasi

Proses implementasi metode terdiri dari 2 proses yaitu *document retrieval* dan *machine reading comprehension.* Bahasa pemrograman Python digunakan untuk membuat sebuah *web service.* Pada proses *document retrieval, web service* akan menerima *keyword* berupa pertanyaan dari *user* dan akan mendapatkan kumpulan dokumen dari Google Custom Search API. Dokumen akan diolah dengan

bantuan *library* Python Beautiful Soup. Setelah mendapatkan dokumen proses berikutnya adalah *machine reading comprehension* untuk mencari jawaban di dokumen dengan mengimplementasikan ALBERT. Proses *training* ALBERT dilakukan menggunakan *dataset* SQuAD v2 dengan bantuan *library* Python Transformers. ALBERT akan memberikan *output* berupa jawaban yang paling sesuai dengan pertanyaan dari dokumen yang diterima.

d. Pengujian dan Evaluasi

Pengujian hasil implementasi dilakukan setelah implementasi metode selesai untuk mengetahui apakah masih terdapat kesalahan. Dalam penelitian ini pengujian dilakukan dengan pendekatan *whitebox testing* untuk mengetahui apakah implementasi ALBERT pada *web service* sudah berjalan dengan benar sesuai dengan perancangan dan dibagi menjadi 2 yaitu, pengujian mode Explore dan pengujian mode Explore More. Evaluasi dilakukan dengan melihat nilai akurasi, *F-Score*, dan *user time* dari *web service* yang telah dibuat. Evaluasi tambahan juga dilakukan dengan melihat nilai akurasi dan *F-score* dengan *dataset* SQuAD dari model ALBERT yang dipakai.

e. Penulisan Laporan

Proses dan hasil akhir dari penelitian akan dicatat dalam bentuk laporan sebagai dokumentasi dari penelitian yang telah dilakukan.

3.2 Analisis Kebutuhan

Hasil analisis terhadap aplikasi *voice chatbot* Jacob adalah sebagai berikut.

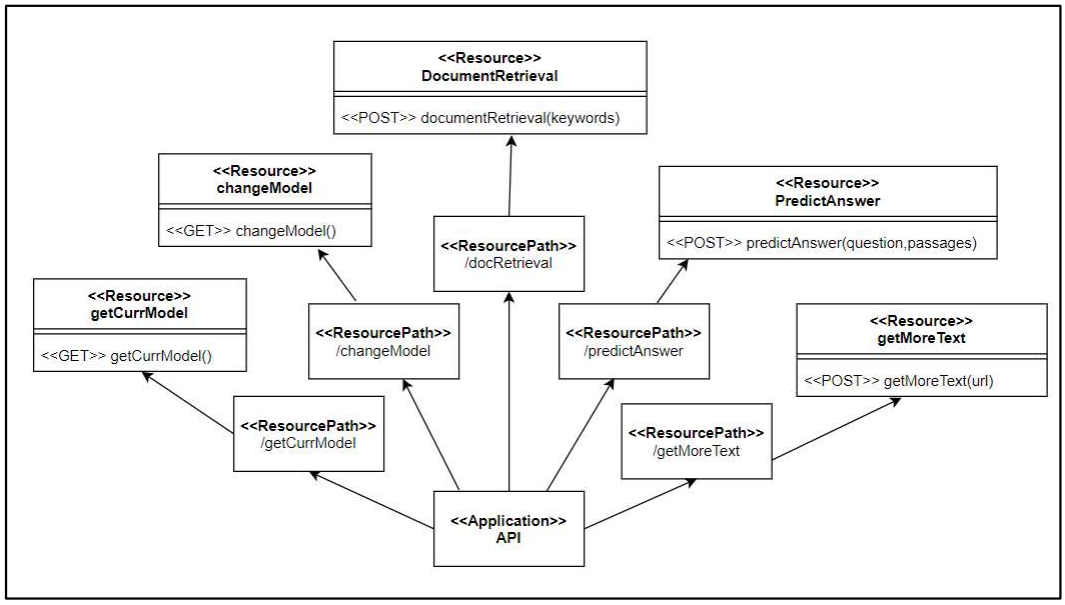
1. Bahasa yang digunakan pada *voice chatbot* Jacob adalah Bahasa Inggris, sehingga dokumen yang digunakan untuk *text mining* berbasis Bahasa Inggris.
2. *Voice chatbot* Jacob dibuat menggunakan *framework* Laravel, sehingga *web service* dirancang dan dikembangkan menggunakan bahasa pemrograman Python dengan *framework* Flask.
3. Model ALBERT untuk prediksi jawaban menggunakan *pre-trained model* dari *library* Transformers pada bahasa pemrograman Python.
4. Pencarian informasi mengenai pertanyaan *user* di internet dilakukan dengan bantuan Google Custom Search Engine API.
5. Melakukan *update* terhadap *entity* question_word pada Wit.ai yang digunakan *voice chatbot* Jacob sehingga memenuhi unsur 5W+1H.
6. Menambahkan *entity* wit/datetime pada Wit.ai yang digunakan *voice chatbot* Jacob untuk membantu deteksi unsur waktu pada pertanyaan *user*.
7. Menggunakan Bootstrap Toggle untuk pengembangan *interface voice chatbot* Jacob.

3.3 Perancangan Web Service

Dari hasil analisis di atas, maka dirancanglah sebuah *web service* dengan fungsionalitas yang ditunjukkan dengan diagram-diagram, yaitu *web service model*, *flowchart* sistem, dan *flowchart web service.* Tampilan *interface* untuk integrasi *web service* ke *voice chatbot* Jacob juga akan dirancang.

3.3.1 Web Service Model

Model *web service* yang dibuat dalam penelitian ini dapat dilihat pada Gambar 3.1. *Web service* yang dibangun memiliki 6 *resource,* yaitu Get Curr Model untuk mengetahui model apa yang sedang digunakan *web service*, Change Model untuk mengganti model yang digunakan *web service*, Document Retrieval untuk mendapatkan hasil pencarian *keywords* menggunakan Google Custom Search API, Get More Text untuk mendapatkan isi teks dari sebuah halaman *website,* dan Predict Answer untuk memprediksi jawaban atas sebuah pertanyaan dari kumpulan kalimat. Get Curr Model dan Change Model merupakan *resource* yang digunakan untuk membantuk proses pengujian dan evaluasi pada penelitian ini. Data *request* dan *response* pada *web service* berbentuk JSON.



Gambar 3.1 *Web Service* Model

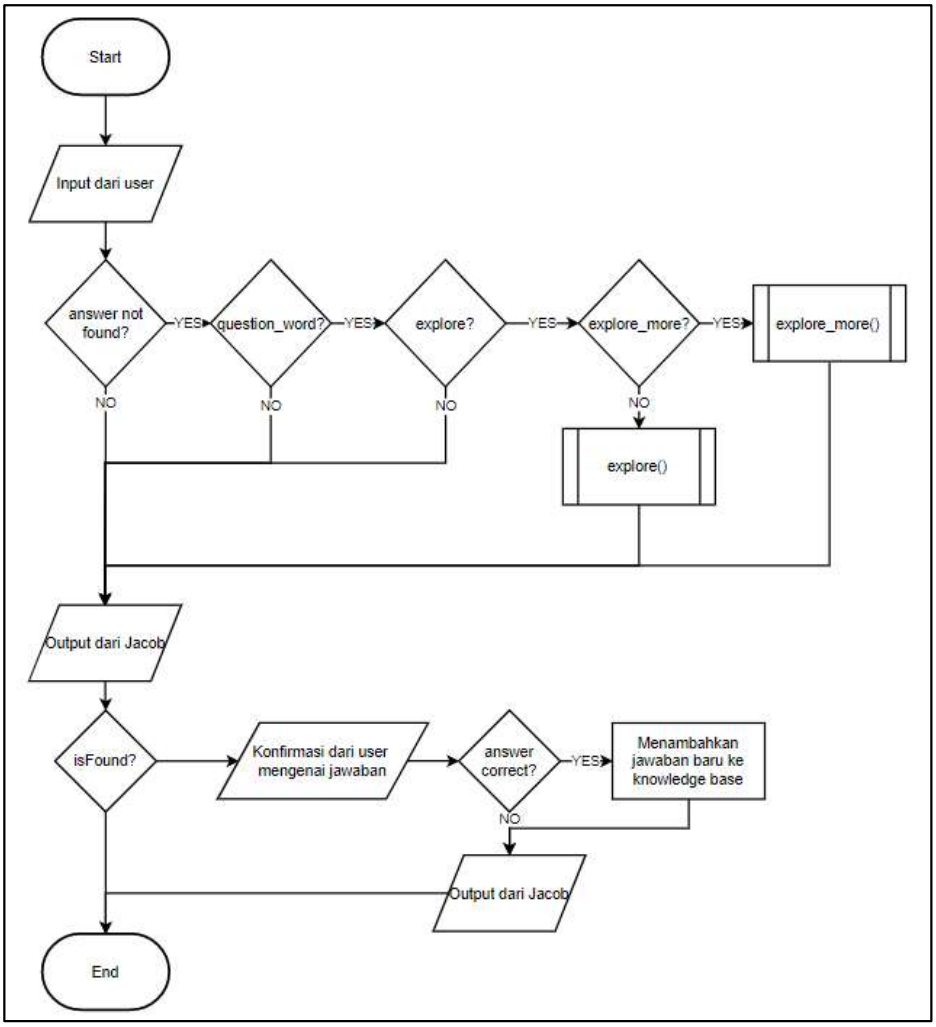
3.3.2 Flowchart Sistem

Flowchart sistem digunakan untuk menggambarkan alur dari hasil integrasi *web service* ke dalam aplikasi *voice chatbot* Jacob. *Flowchart* sistem terdiri dari *flowchart* utama, *explore*, dan *explore more*.

A. Flowchart Utama

Flowchart utama menggambarkan alur saat *voice chatbot* Jacob menerima pertanyaan dari *user* yang jawabannya tidak ada di *database*. Gambar 3.1 menunjukkan *flowchart* utama. Ketika Jacob menerima pertanyaan yang tidak diketahui jawabannya, Jacob akan mengecek apakah "pertanyaan" tersebut benar-benar sebuah pertanyaan. Pengecekan dapat menggunakan *entity* question_word yang didapat dari wit.ai. Jika terdapat *entity* question_word dari "pertanyaan" tersebut maka Jacob akan melihat nilai dari 2 *button* yang akan dirancang di subbab 3.3.4, yaitu *button* toggle-explore dan *button* toggle-deeper. Untuk memakai *web service* yang dibuat di penelitian ini, toggle-explore harus bernilai *true*, sehingga Jacob akan melakukan *text mining* jawaban dari pertanyaan *user* di internet yang ditunjukkan pada Gambar 3.3. Toggle-deeper digunakan untuk menentukan apakah proses *text mining* perlu memakai data yang lebih banyak atau tidak. Alur saat toggle-deeper bernilai *true* dapat dilihat pada Gambar 3.4. Setelah melihat 2 nilai *button*, Jacob akan mengeluarkan output suara berdasarkan ada atau tidaknya jawaban yang didapat. Jika jawaban ditemukan dari proses explore atau explore more, maka variabel isFound akan bernilai *true* menandakan jawaban dari pertanyaan yang sebelumnya tidak ada di *database* ditemukan dan dapat dimasukkan ke *database*. Untuk melakukan *insert* ke *database*, Jacob akan menerima input mengenai konfirmasi dari *user* untuk menentukan apakah jawaban

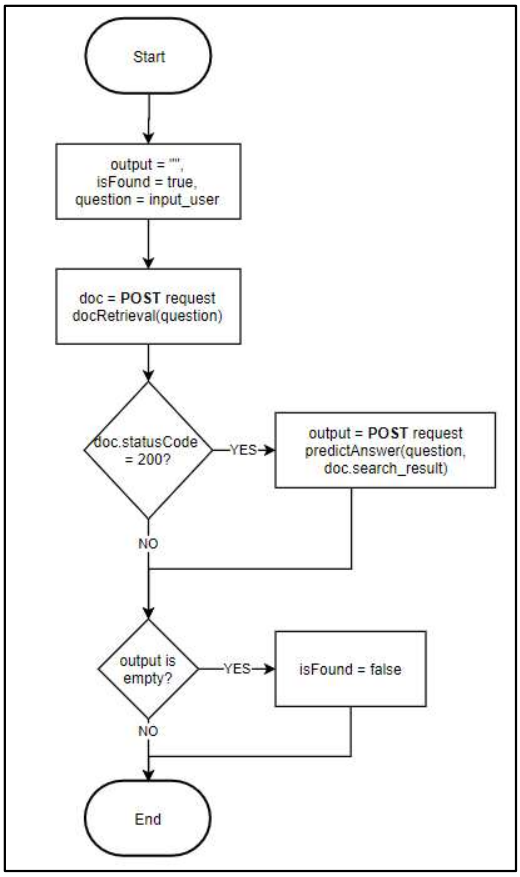
tersebut benar atau setidaknya masuk akal. Proses *insert* akan dilakukan jika input dari *user* memiliki *intent* “yes”.



Gambar 3.2 *Flowchart* Utama

B. Flowchart Explore

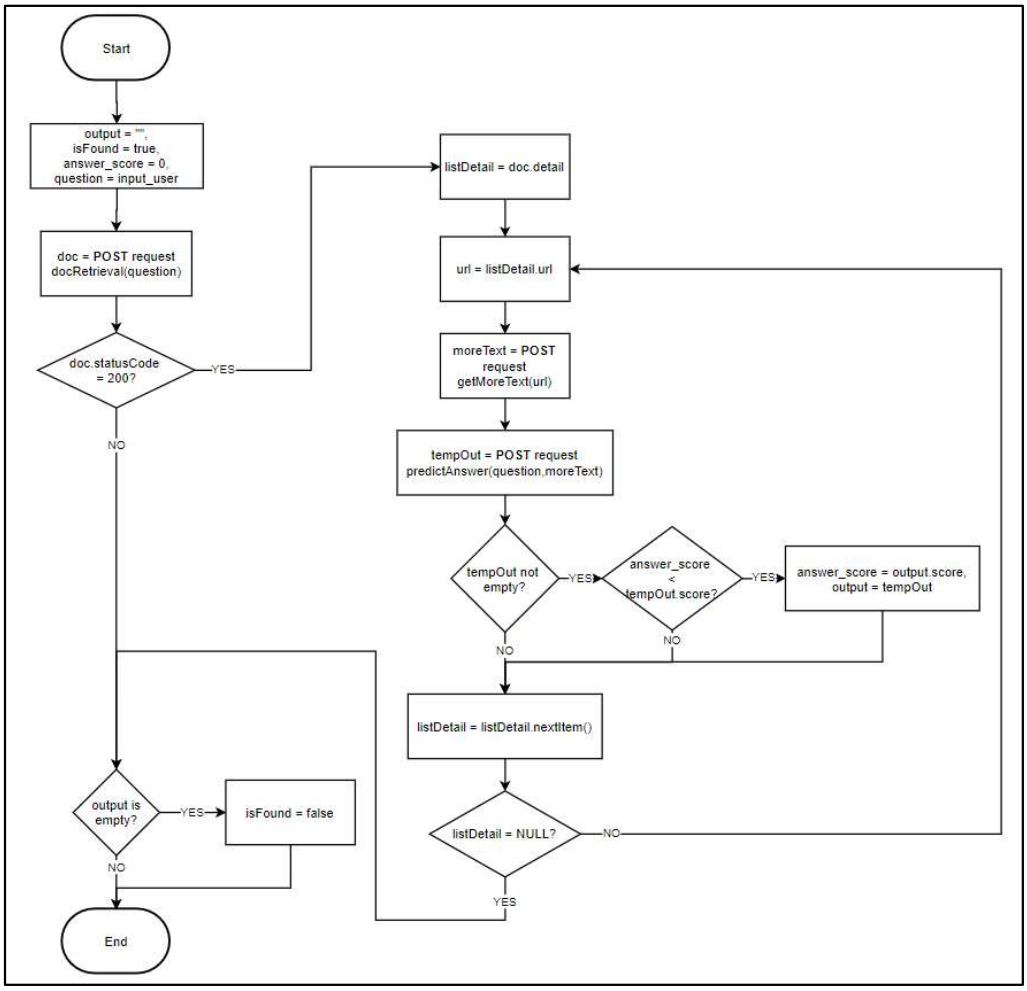
Flowchart Explore pada Gambar 3.3 menggambarkan alur ketika *button* toggle-explore bernilai *true.* Jacob akan mengirim *request post* Document Retrieval ke *web service* untuk mendapatkan teks hasil pencarian terkait pertanyaan *user*. Jika *request* berhasil, maka Jacob akan mengirim *request post* Predict Answer ke *web service* untuk mendapatkan jawaban dari pertanyaan *user* yang dianggap ada pada teks hasil pencarian.



Gambar 3.3 *Flowchart* Explore

C. Flowchart Explore More

Flowchart Explore More pada Gambar 3.4 menggambarkan alur ketika kedua *button* toggle-explore dan toggle-deeper bernilai *true*. Di awal alur, sama dengan *flowchart* Explore, Jacob melakukan *request post* Document Retrieval ke *web service*. Perbedaannya terletak di elemen yang digunakan dari *response* yang didapat ketika *request post* Document Retrieval, yaitu elemen detail. Jacob akan mencari jawaban dengan mengambil teks yang didapat dari *request post* Get More Text dengan nilai *url* di detail. Proses tersebut akan dilakukan sebanyak jumlah elemen yang ada di detail. Jacob akan mengambil jawaban dengan skor tertinggi (probabilitas tertinggi).



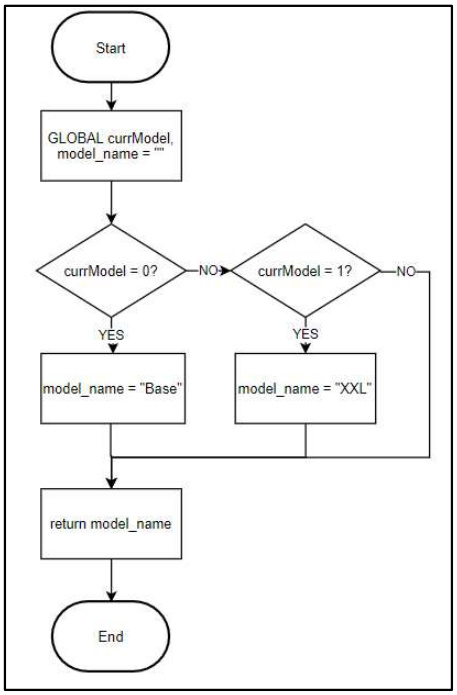
Gambar 3.4 *Flowchart* Explore More

3.3.3 Flowchart Web Service

Flowchart web service digunakan untuk menggambarkan alur *web service* yang dibuat di penelitian ini. *Flowchart web service* terdiri dari *flowchart* Get Curr Model, Change Model, Document Retrieval, Get More Text, dan Predict Answer .

A. Flowchart Get Curr Model

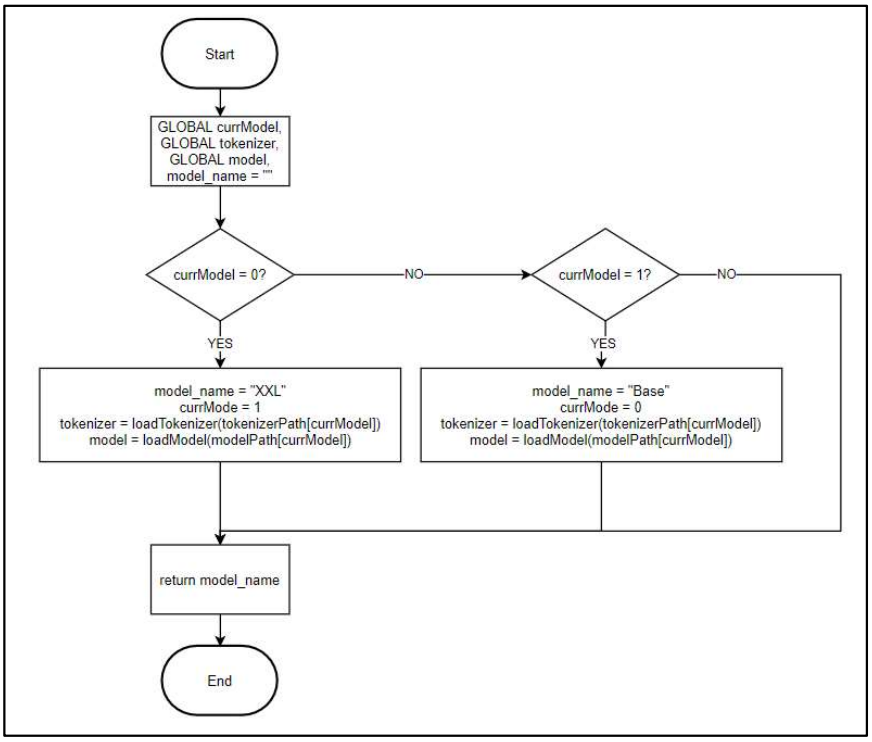
Flowchart Get Curr Model pada Gambar 3.5 menggambarkan alur ketika *resource* Get Curr Model diakses. *Response* dari *resource* berupa nama dari model yang sedang dipakai *web service*, yaitu Base atau XXLarge. Variabel global currModel digunakan untuk membantu pengecekan model yang sedang dipakai.



Gambar 3.5 *Flowchart* Get Current Model

B. Flowchart Change Model

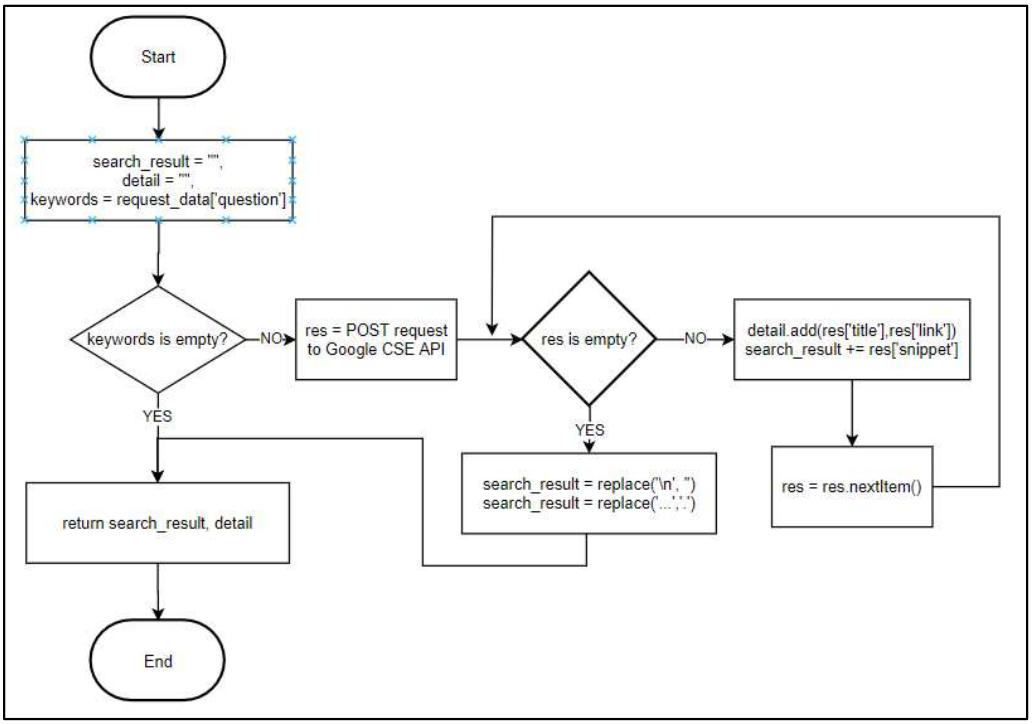
Flowchart Change Model yang ditunjukkan pada Gambar 3.6 menggambarkan alur ketika *resource* Change Model diakses. Variabel global currModel digunakan untuk membantu menentukan model yang sedang dipakai *web service*. Model dan tokenizer pada *web service* akan dimuat ulang sesuai dengan model dari indeks currModel. *Response* dari *resource* berupa nama model yang sedang dipakai, yaitu Base atau XXLarge.



Gambar 3.6 *Flowchart* Change Model

C. Flowchart Document Retrieval

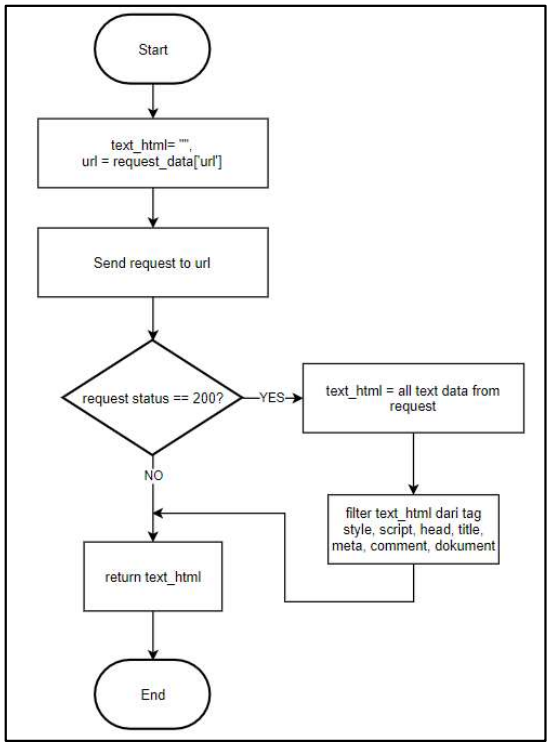
Flowchart Document Retrieval pada Gambar 3.7 menggambarkan alur ketika *resource* Document Retrieval diakses. *Resource* membutuhkan data *keywords* pada *request body* sebagai nilai untuk melakukan pencarian teks di internet dengan memanfaatkan Google Custom Search Engine API. *Response* dari *resource* adalah search_result dan detail. Data search_result merupakan gabungan dari potongan teks (*snippet*) dari hasil pencarian menggunakan Google CSE API. Data detail merupakan kumpulan elemen *title* dan *link* yang didapat dari hasil pencarian menggunakan Google CSE API.



Gambar 3.7 *Flowchart* Document Retrieval

D. Flowchart Get More Text

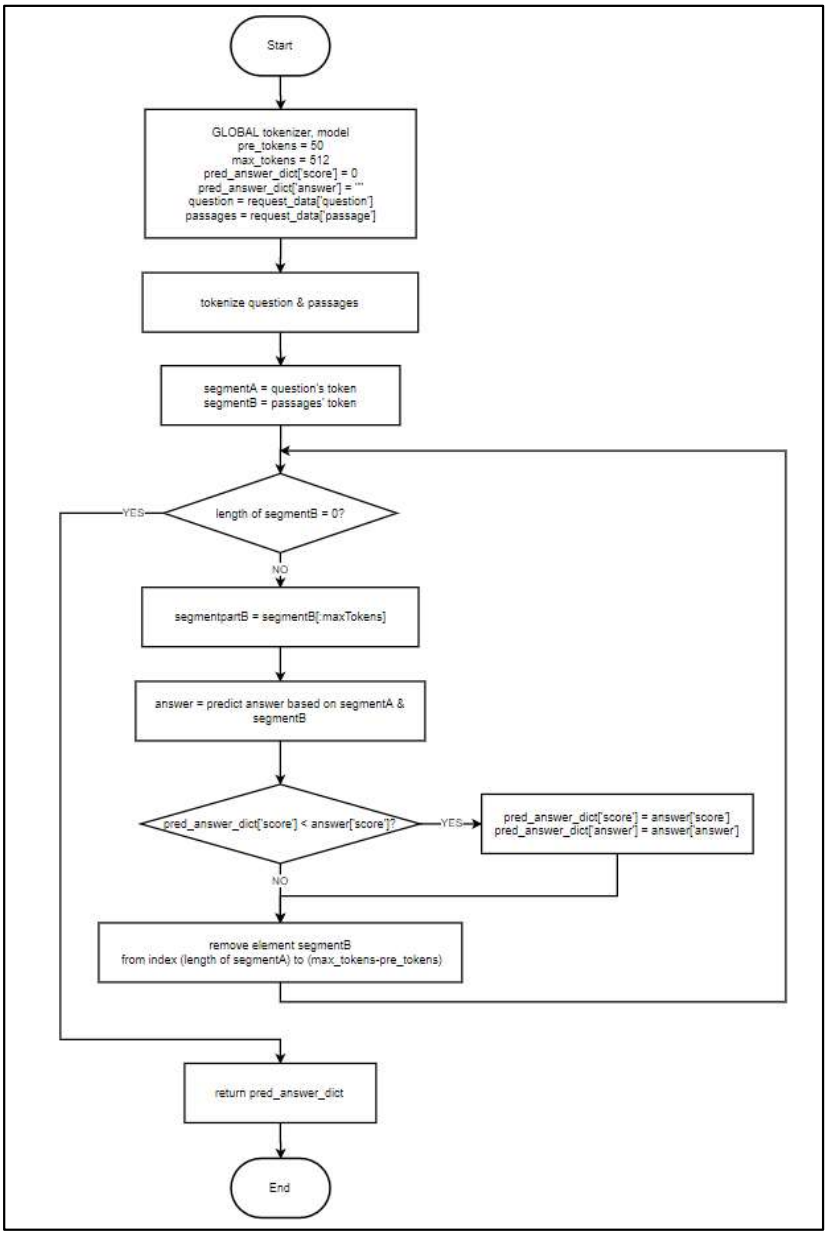
Flowchart Get More Text pada Gambar 3.8 menggambarkan alur ketika *resource* Get More Text diakses. *Resource* membutuhkan data *url* pada *request body* sebagai nilai untuk mendapatkan isi teks dari *url* tersebut. Jika *website* dari *url* memberikan izin untuk melakukan request maka teks pada *website* akan diambil dan kemudian di-*filter* beberapa *tag* yang sudah ditentukan. *Response* dari *resource* adalah teks dari *website* yang sudah di-*filter*.



Gambar 3.8 *Flowchart* Get More Text

E. Flowchart Predict Answer

Flowchart Predict Answer pada Gambar 3.9 menggambarkan alur ketika *resource* Predict Answer diakses. *Resource* membutuhkan data *question* dan *passages* pada *request body*. *Question* merupakan pertanyaan dan *passages* merupakan kumpulan kalimat yang akan digunakan untuk memprediksi jawaban dari pertanyaan. Variabel pre_tokens dan max_tokens digunakan untuk mengatasi masalah yang ada di model. Pre_tokens digunakan sebagai nilai jumlah token sebelumnya yang akan digabungkan dengan *list* token berikutnya. Max_tokens merupakan batas jumlah token yang dapat dimasukkan ke model untuk prediksi. *Response* dari *resource* adalah pred_answer_dict yang memiliki elemen *score* tertinggi dan *answer*.

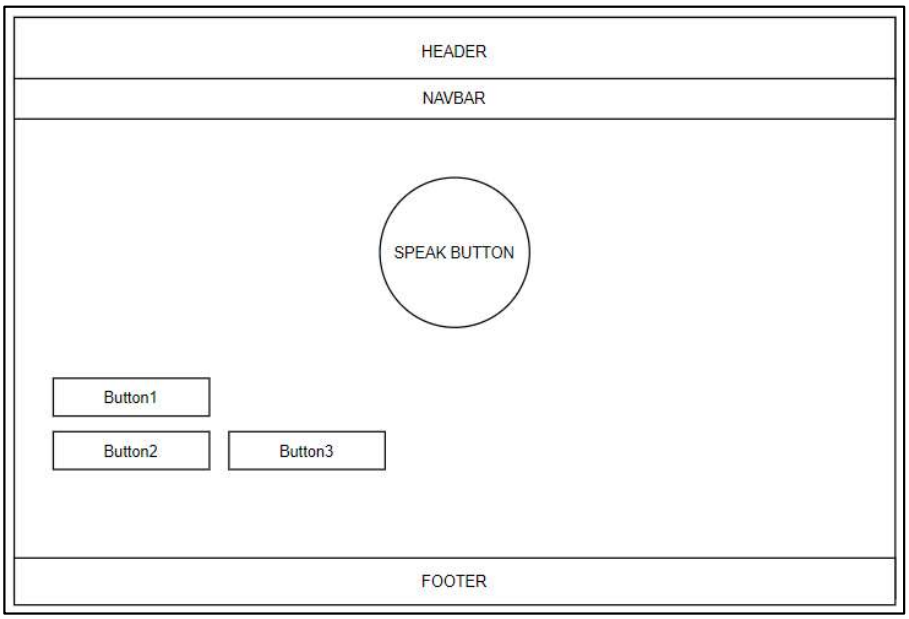


Gambar 3.9 *Flowchart* Predict Answer

3.3.4 Rancangan Interface

Pada subbab ini dijelaskan rancangan *interface* integrasi *web service* ke *voice chatbot* Jacob. Rancangan *interface* yang ditunjukkan pada Gambar 3.10 merupakan pengembangan dari *interface* Jacob yang sudah ada. Pengembangan dilakukan dengan menambahkan 3 *button* di bawah *speak button*. Button1 digunakan sebagai *button* toggle-explore yang sudah dijelaskan sebelumnya.

Button2 memiliki 2 fungsi, yaitu untuk menampilkan model yang dipakai *web service* dan mengakses *resource* Change Model pada *web service*. Button3 digunakan sebagai *button* toggle-deeper yang sudah dijelaskan sebelumnya. Button2 dan Button3 hanya akan muncul apabila Button1 bernilai *true*.



Gambar 3.10 Rancangan *Interface*