BAB III

METODOLOGI DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Metode penelitian yang dilakukan dibagi menjadi empat bagian, yaitu studi literatur, perancangan dan pembuatan aplikasi, *testing* dan *debugging*, dan evaluasi.

a. Studi literatur

Dalam studi literatur, dilakukan pembelajaran mengenai beberapa teori yang akan digunakan dalam pembangunan aplikasi untuk menentukan rute tercepat dengan menggunakan transportasi massal. Teori-teori yang dipelajari adalah *shortest path problem* dan algoritma Floyd-Warshall. Teori-teori yang dipelajari didapat dari jurnal-jurnal ilmiah yang dipublikasikan secara *online*.

b. Perancangan dan pembuatan aplikasi

Setelah tahap studi literatur selesai, dilanjutkan dengan proses perancangan dan pembuatan aplikasi untuk menemukan rute terdekat dengan menggunakan transportasi massal. Aplikasi terdiri dari *client side*, *server side*, dan mempunyai *database*. *Database* yang ada berisi ID tiap halte, jarak dari suatu halte ke halte lainnya, *list* halte yang terhubung dengan halte tersebut, dan apakah ada stasiun kereta yang dekat dengan halte tersebut. Pada *server side*, aplikasi bertugas untuk meng-*update* data di *database* setiap kali terdapat perubahan, misalnya ada halte yang ditutup

maupun baru selesai dibangun dan juga terdapat rute baru dari sebuah halte. *Client side* akan menampilkan *interface* yang akan meminta input berupa halte awal dan halte tujuan, kemudian aplikasi akan mencari rute terpendek untuk sampai ke halte tujuan. Pencarian rute dilakukan dengan memerhatikan beberapa faktor, yaitu panjangnya rute yang dilalui, jumlah halte yang dilewati, dan ada atau tidaknya stasiun kereta di sepanjang rute.

c. Testing dan debugging

Setelah aplikasi selesai dibangun, selanjutnya aplikasi akan di-*debug* terlebih dahulu untuk melihat apakah masih terdapat *error* dan *bug*. Setelah proses *debug* selesai, aplikasi akan diuji secara langsung di lapangan. Pengujian dilakukan sendiri dan juga meminta orang-orang sekitar untuk mencoba menggunakan aplikasi. Setelah selesai melakukan *testing*, *tester* akan diberikan survey untuk diisi.

d. Evaluasi

Setelah aplikasi melalui tahap pengujian dan survey sudah diisi, akan dilakukan evaluasi berdasarkan hasil *testing* dan survey yang ada. Minimal jumlah sampel adalah tiga puluh orang (Sugiyono, 2012). Mula-mula, *tester* akan diminta untuk menguji coba aplikasi yang sudah dibuat. Setelah *tester* selesai mencoba aplikasi, akan diberikan *kuisioner* tentang aplikasi untuk diisi. Isi dari *kuisioner* berupa pertanyaan tentang kemudahan aplikasi untuk digunakan, tingkat akurasi aplikasi, dan apakah aplikasi yang dibuat sesuai dengan kebutuhan *user*. Alat pengukuran menggunakan skala Likert lima tingkat.

3.2 Variabel Penelitian

Variabel yang digunakan saat tahap pengujian dibagi menjadi variabel aplikasi dan variabel algoritma. Variabel aplikasi didasari oleh teori yang dikemukakan oleh Nayebi dkk (2012), yaitu *efficient to use, easier to learn*, dan *user satisfaction*. Sedangkan variabel algoritma diambil berdasarkan faktor yang mempengaruhi perhitungan pencarian rute terdekat, yaitu jumlah halte yang dilewati, jarak antar halte, dan ada atau tidaknya stasiun kereta di sepanjang rute.

3.3 Teknik Pengumpulan Data

Data jalur dan rute didapat dari Google Maps API, sedangkan data-data hasil penelitian didapat dari pengujian aplikasi dan pengisian *kuisioner* oleh responden dengan menggunakan skala Likert lima tingkat sebagai alat pengukuran.

3.4 Teknik Pengambilan Sampel

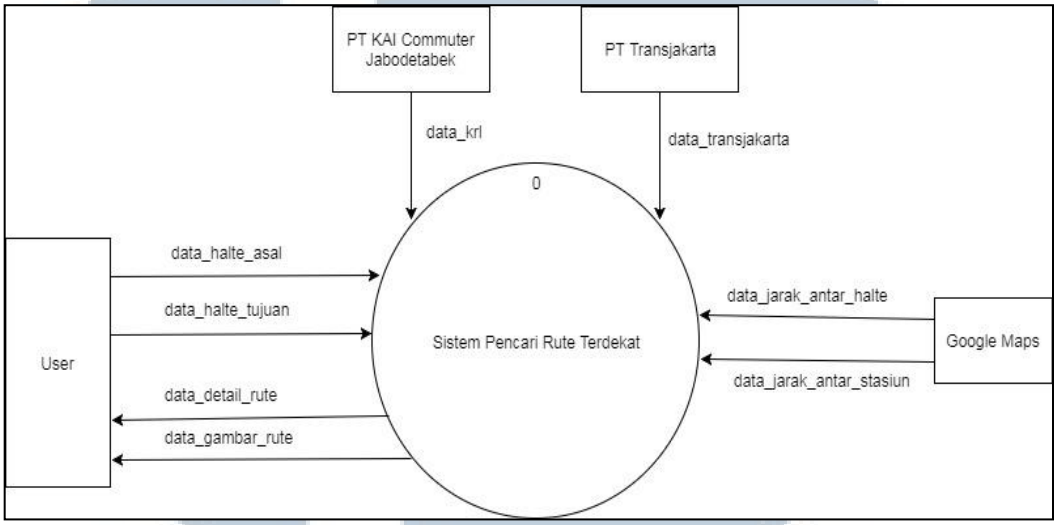
Teknik pengambilan sampel yang digunakan adalah teknik *simple random sampling* dimana sampel dipilih secara acak. Teknik ini dipakai karena tidak ada kriteria khusus dalam pengambilan sampel (siapa saja dapat menjadi sampel). Jumlah minimal sampel yang diambil adalah tiga puluh orang (Sugiyono, 2012).

3.5 Perancangan Aplikasi

Perancangan aplikasi menghasilkan beberapa dokumen yang menjelaskan perpindahan data dalam aplikasi dalam bentuk *Data Flow Diagram* (DFD), alur kerja aplikasi dalam bentuk *flowchart*, dan relasi antar tabel pada *database* dalam bentuk *Database Schema*.

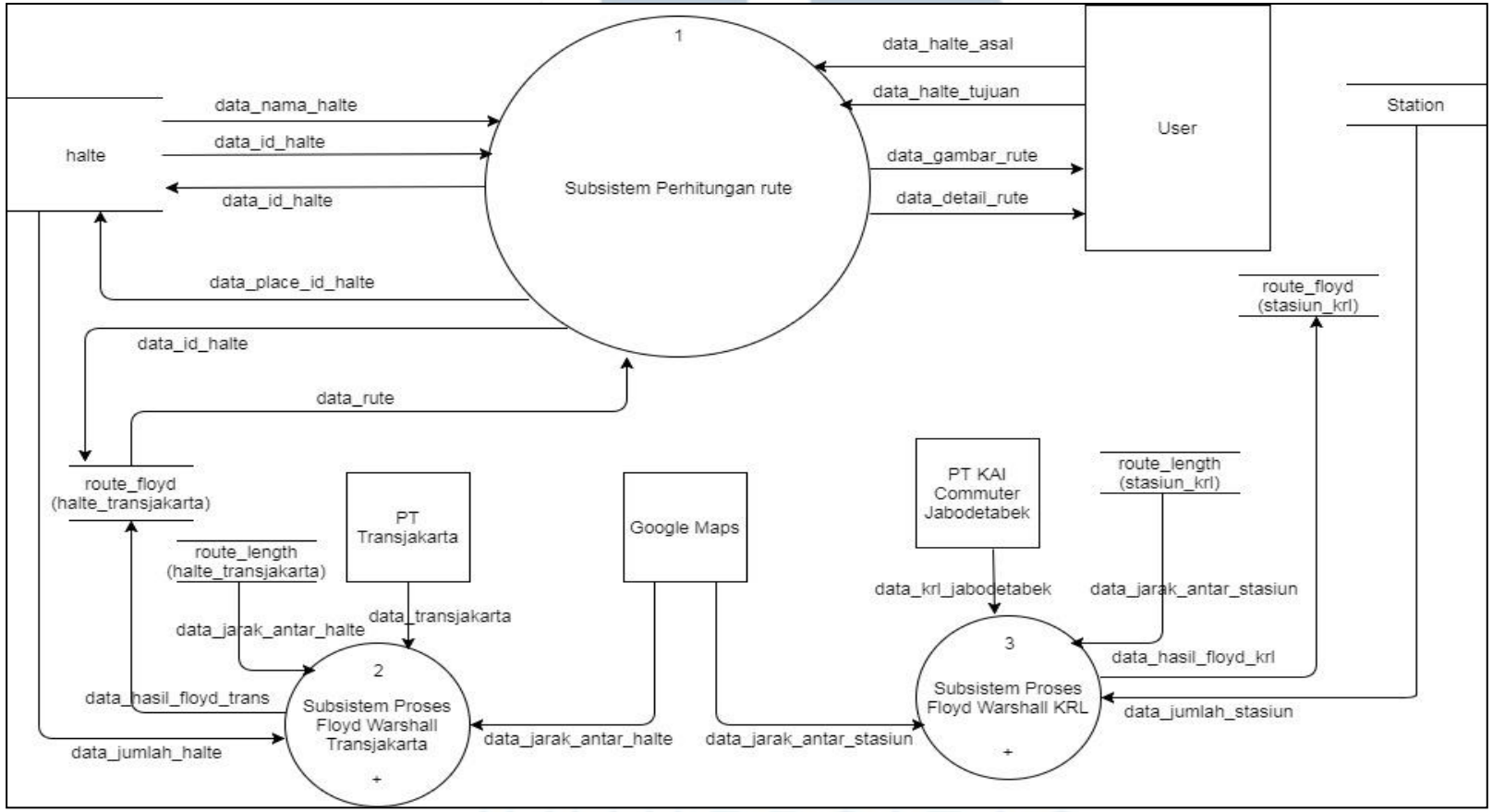
3.5.1 Data Flow Diagram

Pada subbab ini akan dijelaskan tentang alur perpindahan data pada aplikasi dalam bentuk *data flow diagram*. Alur perpindahan data secara garis besar digambarkan dalam bentuk *context diagram* pada Gambar 3.1.



Gambar 3.1 *Context Diagram* Aplikasi

Dalam *context diagram* aplikasi ini terdapat satu proses utama yaitu sistem pencari rute terdekat serta empat entitas, yaitu *user*, PT KAI Commuter Jabodetabek, PT Transjakarta, dan Google Maps. *User* memberikan dua input ke dalam sistem, yaitu data halte asal dan data halte yang ingin dituju. Selain itu, *user* juga menerima dua input dari sistem, yaitu data gambar rute dan data detail rute. Sistem juga memiliki komunikasi dengan tiga entitas lain untuk mendapatkan data-data yang dibutuhkan. Sistem mendapatkan data mengenai KRL dari PT KAI Commuter Jabodetabek, data mengenai transJakarta dari PT TransJakarta, dan data jarak antar halte dan antar stasiun dari Google Maps. Data jarak yang didapat dari google maps selanjutnya digunakan untuk proses mencari rute terdekat.



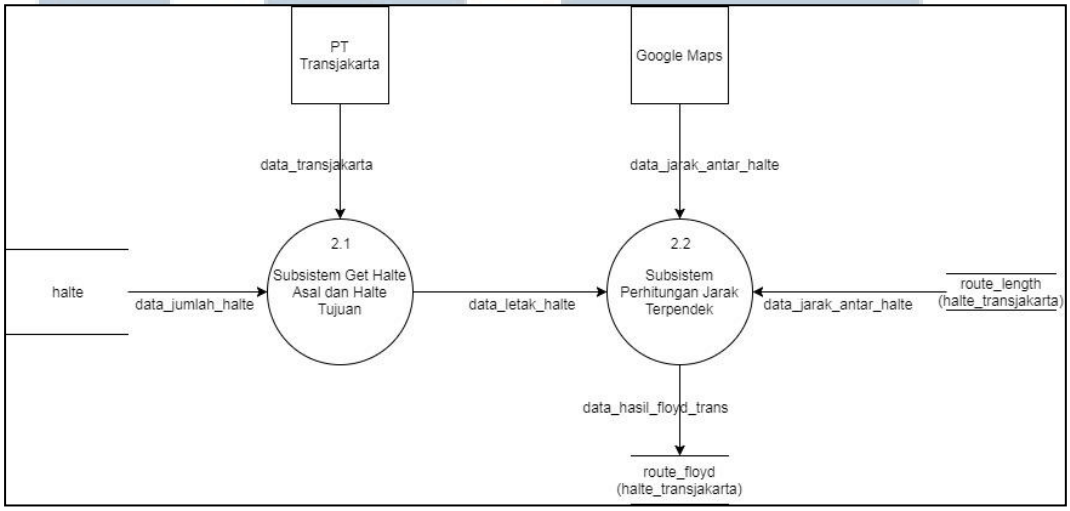
Gambar 3.2 *Data Flow Diagram* Level 1 Aplikasi

Gambar 3.2 menampilkan DFD level 1 aplikasi yang merupakan turunan dari *context diagram* aplikasi. Sistem pencari rute terdekat diturunkan menjadi subsistem perhitungan rute, subsistem proses Floyd-warshall TransJakarta, dan subsistem proses Floyd-Warshall KRL. Subsistem proses Floyd-Warshall TransJakarta berfungsi untuk mengeksekusi algoritma Floyd-Warshall pada setiap pasangan halte yang ada. Subsistem proses Floyd-Warshall KRL berfungsi untuk mengeksekusi algoritma Floyd-Warshall pada setiap pasangan stasiun yang ada. Subsistem perhitungan rute berfungsi untuk menghitung jarak halte asal dan halte tujuan dan mencari rute terdekat yang mungkin berdasarkan data yang didapat dari proses floyd_warshall yang ada.

Terdapat enam buah tempat penyimpanan dalam bentuk tabel *database*. Tabel station menyimpan data masing-masing stasiun KRL. Tabel route_length KRL menyimpan jarak dari satu stasiun ke stasiun lainnya yang saling berhubungan. Tabel route_floyd KRL menyimpan hasil proses Floyd-Warshall KRL berupa jarak terpendek antar stasiun. Tabel halte menyimpan data masing-masing halte TransJakarta yang digunakan dalam proses perhitungan rute. Tabel route_length TransJakarta menyimpan jarak dari satu halte ke halte lainnya yang saling berhubungan. Tabel route_floyd Transjakarta menyimpan hasil proses Floyd-Warshall TransJakarta berupa jarak terpendek antar halte.

Gambar 3.3 menampilkan DFD level 2 yang merupakan turunan dari subsistem proses Floyd-Warshall transJakarta. Subsistem proses Floyd-Warshall transJakarta diturunkan menjadi dua subsistem, yaitu subsistem get halte asal dan halte tujuan serta subsistem perhitungan jarak terpendek. Subsistem get halte asal dan halte tujuan berfungsi untuk mendapatkan data nama , *place id*, dan koordinat

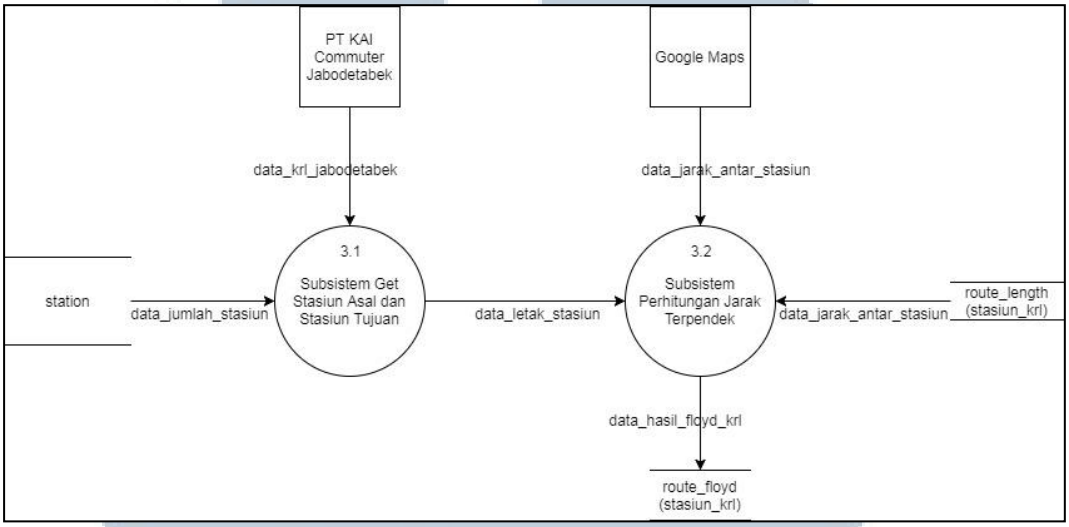
lintang dan bujur masing-masing halte. Data-data yang didapat kemudian dikirimkan ke subsistem perhitungan jarak terpendek. Subsistem perhitungan jarak terpendek berfungsi untuk melakukan perhitungan jarak terpendek yang mungkin dari setiap pasangan halte yang ada dengan menggunakan algoritma Floyd-Warshall. Hasil dari jarak terpendek masing-masing pasangan halte dimasukkan ke dalam tabel route_floyd.



Gambar 3.3 *Data Flow Diagram* Level 2 Proses Floyd-Warshall Transjakarta

Gambar 3.4 menampilkan DFD level 2 yang merupakan turunan dari subsistem proses Floyd-Warshall KRL. Subsistem proses Floyd-Warshall KRL diturunkan menjadi dua subsistem, yaitu subsistem get stasiun asal dan stasiun tujuan serta subsistem perhitungan jarak terpendek. Subsistem get stasiun asal dan stasiun tujuan berfungsi untuk mendapatkan data nama, *place id*, dan koordinat lintang dan bujur masing-masing stasiun. Data-data yang didapat kemudian dikirimkan ke subsistem perhitungan jarak terpendek. Subsistem perhitungan jarak terpendek berfungsi untuk melakukan perhitungan jarak terpendek yang mungkin dari setiap pasangan stasiun yang ada dengan menggunakan algoritma

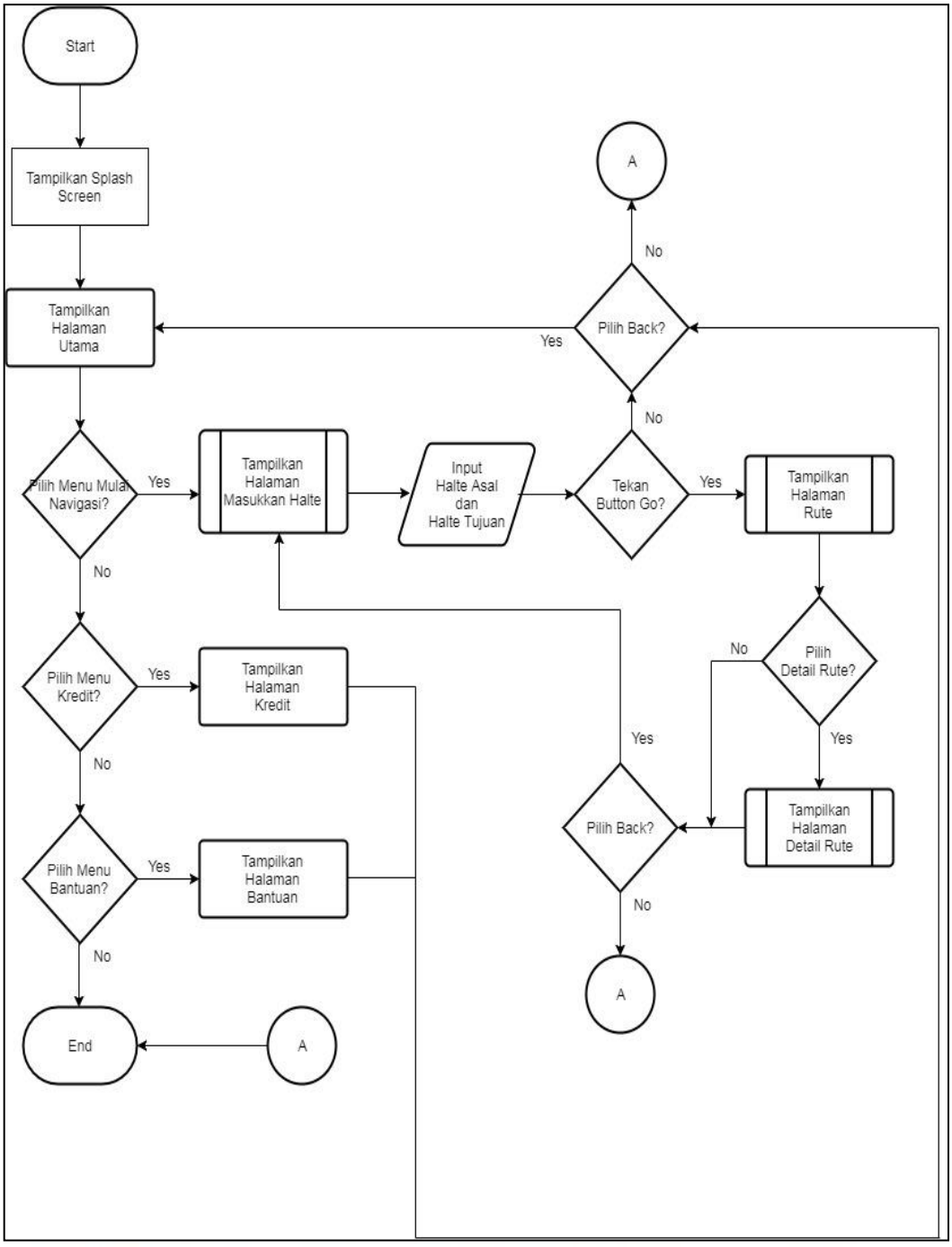
Floyd-Warshall. Hasil dari jarak terpendek masing-masing pasangan stasiun dimasukkan ke dalam tabel route_floyd.



Gambar 3.4 *Data Flow Diagram* Level 2 Proses Floyd-Warshall KRL

3.5.2 Flowchart

Pada subbab ini akan dijelaskan tentang alur kerja aplikasi dalam bentuk *flowchart*.



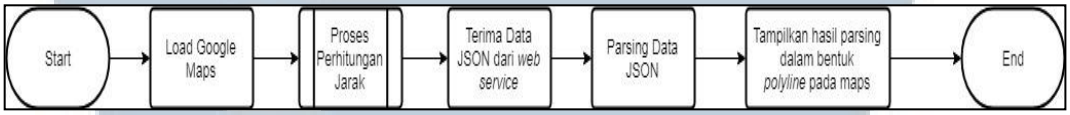
Gambar 3.5 *Flowchart* Aplikasi

A. Flowchart Aplikasi

Gambar 3.5 menggambarkan *flowchart* keseluruhan sistem aplikasi pencari rute terdekat. Proses diawali dengan menampilkan *splash screen* logo UMN, kemudian masuk ke halaman utama. Pada halaman utama terdapat empat menu yang dapat *user* pilih, yaitu menu mulai navigasi, menu kredit, menu bantuan, dan menu keluar. Menu mulai navigasi akan membuka halaman masukkan halte. Menu kredit berisi penjelasan singkat mengenai aplikasi serta referensi-referensi yang ada. Menu bantuan berisi simbol-simbol yang digunakan dalam aplikasi dan instruksi bagaimana cara menggunakan aplikasi beserta dengan *screenshot*. Menu keluar akan menutup aplikasi. Pada halaman masukkan halte terdapat dua buah *textbox* untuk mengisi halte asal dan halte tujuan, tombol *back* di bagian atas layar untuk kembali ke halaman utama, dan tombol *go* dengan simbol panah hijau pada bagian bawah layar untuk melakukan perhitungan rute. Jika *user* menekan tombol *go*, maka halaman rute akan terbuka. Halaman rute berisi peta yang didalamnya terdapat *marker* pada halte asal dan halte tujuan serta garis *polyline* untuk menandakan rute yang ditempuh. Selain itu juga terdapat tombol *back* di bagian atas layar dan tombol detail rute di bagian bawah layar. Jika *user* menekan tombol detail rute, halaman detail rute akan terbuka. Halaman detail rute berisi alamat lengkap halte asal dan halte tujuan, jarak antar halte, halte-halte apa saja yang dilewati selama perjalanan, dan tombol *back* yang terdapat di bagian atas layar. *Flowchart* dari halaman masukkan halte ditunjukkan pada Gambar 3.6.

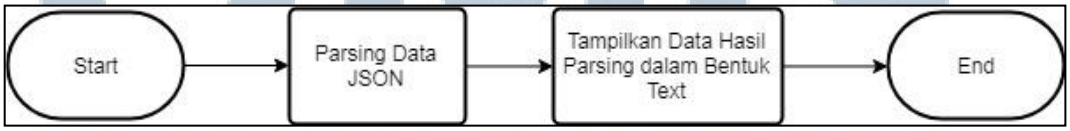
Gambar 3.6 *Flowchart* Halaman Masukkan Halte

Saat *user* mengisi nama halte asal dan halte tujuan lalu menekan tombol *Go*, aplikasi akan meng-*generate* data JSON berdasarkan nama halte asal dan halte tujuan yang telah diisi oleh *user*, kemudian mengirim data tersebut ke *web service* untuk diproses dan ditampilkan pada halaman rute. *Flowchart* untuk halaman rute ditunjukkan pada Gambar 3.7.



Gambar 3.7 *Flowchart* Halaman Rute

Setelah Google Maps berhasil di-*load*, Data yang telah dikirim ke *web service* akan diproses untuk menghitung jarak terpendek dari rute dan dikirim ke aplikasi dalam bentuk JSON. Data tersebut kemudian akan di-*parsing* dan ditampilkan hasilnya pada Google Maps dalam bentuk *polyline* untuk menandai rute yang dilewati. Jika tombol detail rute pada halaman ini ditekan, *user* akan berpindah ke halaman detail rute. *Flowchart* untuk halaman detail rute ditunjukkan pada Gambar 3.8.

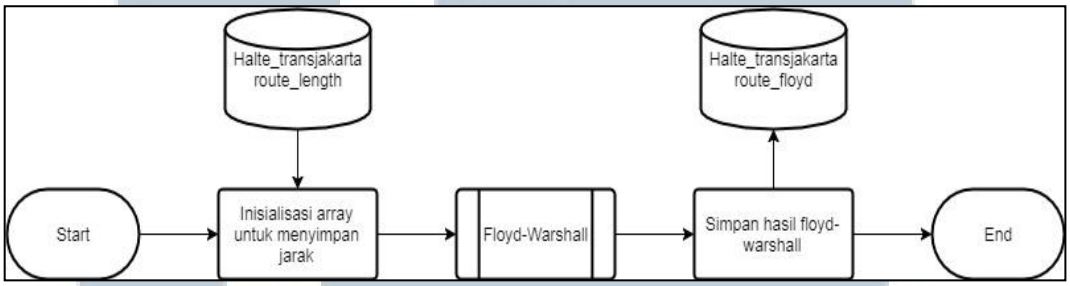


Gambar 3.8 *Flowchart* Halaman Detail Rute

Data dari *web service* yang telah dikirimkan akan di-*parsing* dan hasil dari *parsing* tersebut ditampilkan pada *scrollview* yang terdapat di halaman ini dalam bentuk teks.

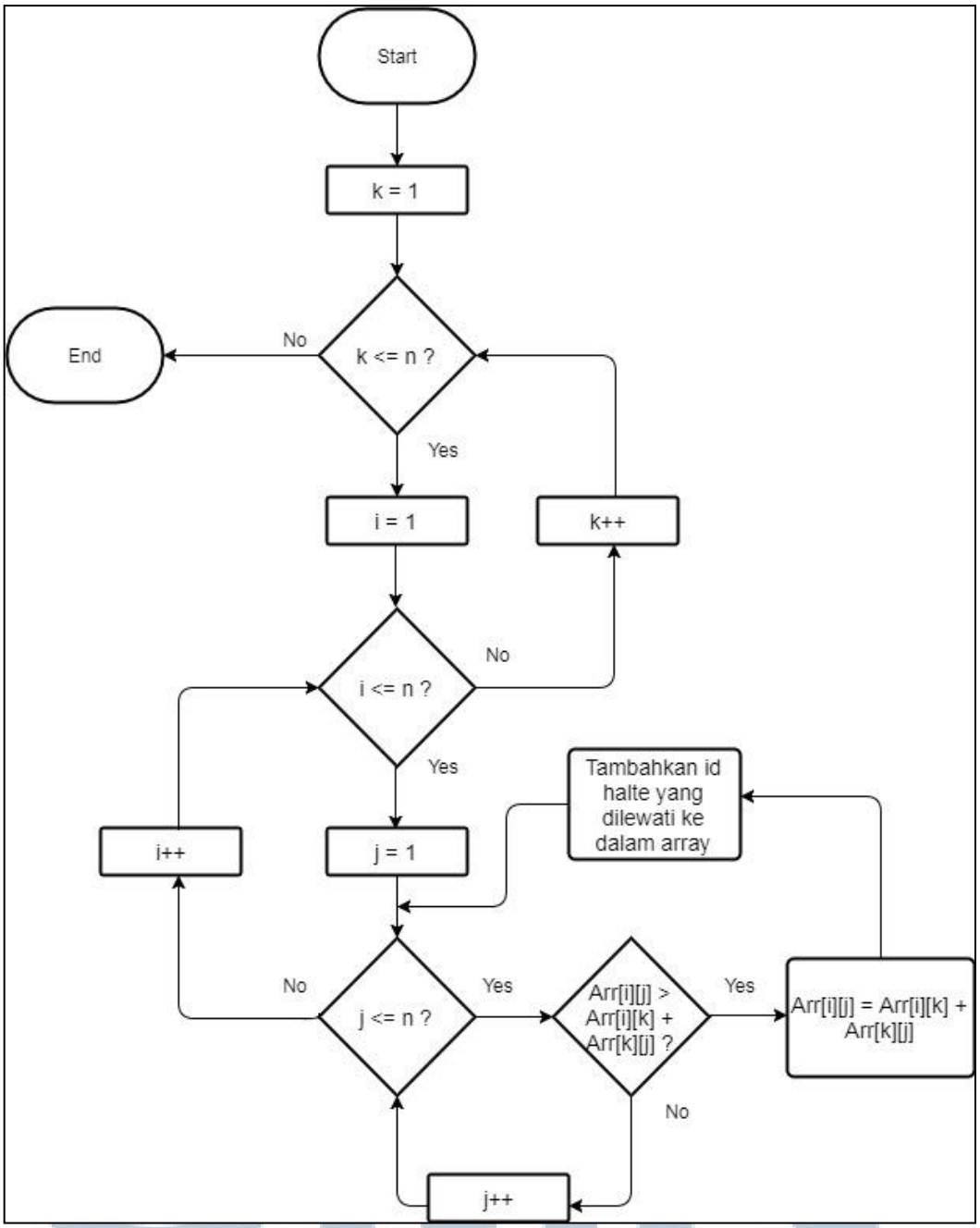
B. Flowchart *Web Service*

Pada pembuatan aplikasi ini digunakan dua buah *web service*, yaitu *web service* untuk membuat tabel route_floyd pada *database* dan *web service* untuk melakukan proses perhitungan jarak. Alur kerja dari *web service* untuk membuat tabel route_floyd ditunjukkan pada Gambar 3.9.



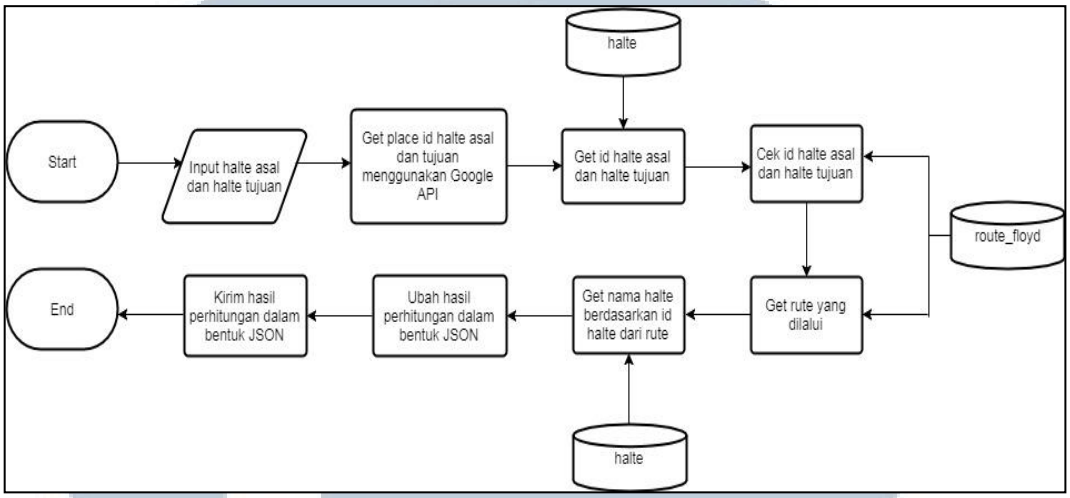
Gambar 3.9 *Flowchart* Pembuatan Tabel route_floyd

Pertama-tama, lakukan inisialisasi pada *array* yang berfungsi menyimpan jarak antar halte. Masukkan data-data dari tabel route_length ke dalam *array* tersebut. Setelah itu lakukan proses Floyd-warshall pada data-data yang terdapat di dalam *array*. Hasil dari proses Floyd-warshall kemudian disimpan ke dalam tabel route_floyd pada *database* halte_transjakarta. Alur kerja algoritma Floyd-Warshall pada ditunjukkan pada Gambar 3.10.



Gambar 3.10 *Flowchart* Subproses Floyd-Warshall halte_transjakarta

Alur kerja dari *web service* untuk melakukan proses perhitungan jarak ditunjukkan pada Gambar 3.11.

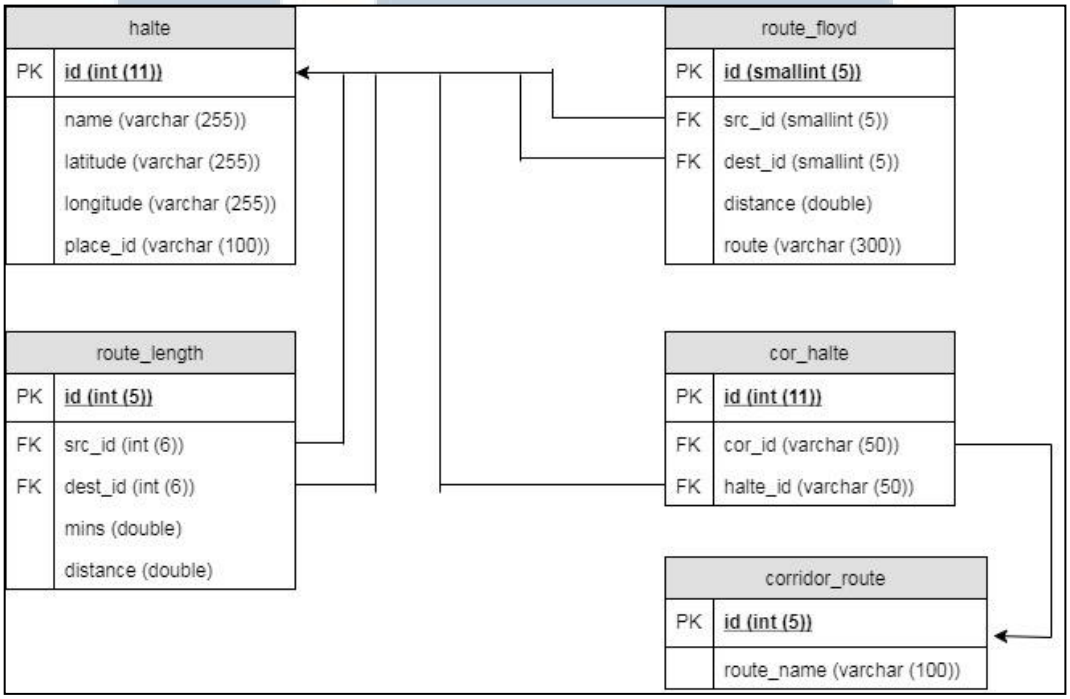


Gambar 3.11 *Flowchart* Proses Perhitungan Jarak

Pertama-tama *web service* menerima input berupa nama halte asal dan halte tujuan dari *user*. Kemudian *web service* mendapatkan *place id* dari halte asal dan halte tujuan dari *Google Maps API*. Setelah mendapatkan *place id*, *web service* mengambil id halte asal dan halte tujuan berdasarkan *place id* yang didapat dari tabel halte yang ada di *database*. Kemudian dilakukan *query database* untuk mencari id halte yang sesuai dengan id halte asal dan halte tujuan rute yang dilalui, serta jarak rute dari tabel route_floyd. Setelah itu, dilakukan *query database* untuk mencari nama halte dari rute karena rute yang didapat dari tabel route_floyd masih dalam bentuk id halte. Pencarian nama halte mengambil data dari tabel halte. Setelah selesai, hasil yang didapat akan diubah ke dalam bentuk JSON dan dikirim ke aplikasi.

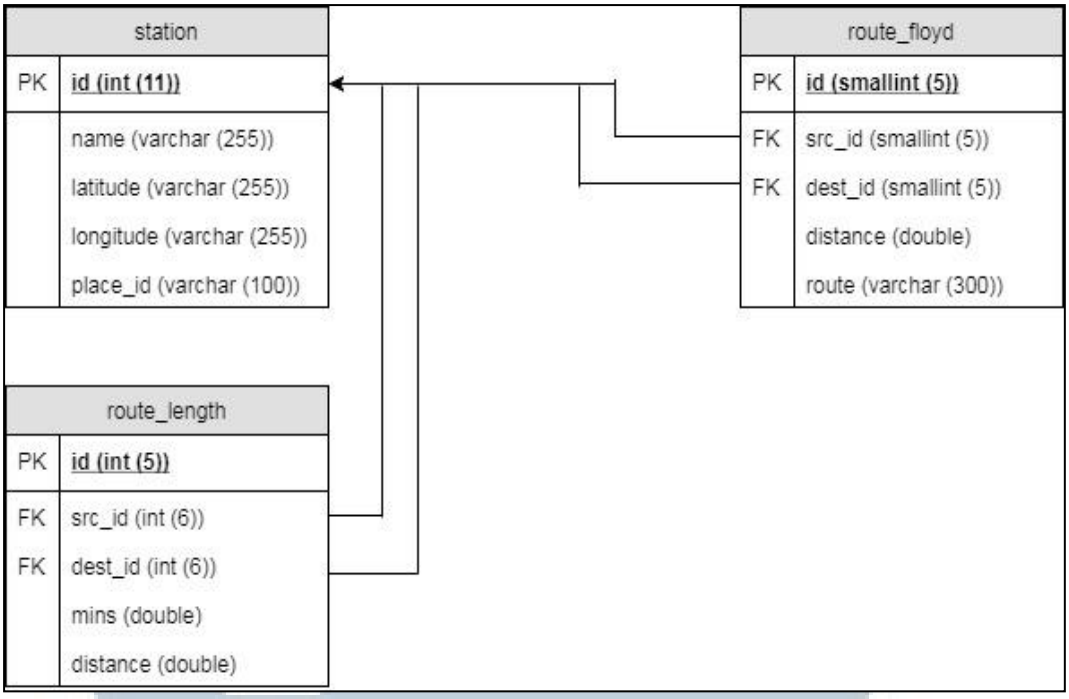
3.5.3 Database Schema

Database yang digunakan pada penelitian ini diberi nama halte_transjakarta dan stasiun_krl. Skema dari *database* halte_transjakarta ditunjukkan pada Gambar 3.12.



Gambar 3.12 *Database Schema* halte_transjakarta

Skema dari *database* stasiun_krl ditunjukkan pada Gambar 3.13.



Gambar 3.13 *Database Schema* stasiun_krl

3.5.4 Struktur Tabel

Database yang digunakan dalam penelitian ini adalah MySQL. Struktur tabel dari *database* yang digunakan dalam penelitian ini diuraikan sebagai berikut.

A. Nama *Database*: halte_transjakarta

1. Nama Tabel: halte

Tabel 3.1 Struktur Tabel halte

Nama Field	Tipe	Panjang	Keterangan
id	int	11	id halte (*primary key*)
name	varchar	255	nama halte
latitude	varchar	255	nilai derajat lintang halte
longitude	varchar	255	nilai derajat bujur halte
place_id	varchar	100	place id halte

2. Nama Tabel: route_length

Tabel 3.2 Struktur Tabel route_length

Nama Field	Tipe	Panjang	Keterangan
id	int	5	id rute (*primary key*)

Tabel 3.2 Struktur Tabel route_length (Lanjutan)

Nama Field	Tipe	Panjang	Keterangan
src_id	int	6	id halte asal (*reference key* dari id pada tabel halte)
dest_id	int	6	id halte tujuan (*reference key* dari id pada tabel halte)
mins	double		waktu dari halte asal ke tujuan (dalam menit)
distance	double		jarak antar halte (dalam km)

3. Nama Tabel: route_floyd

Tabel 3.3 Struktur Tabel route_floyd

Nama Field	Tipe	Panjang	Keterangan
id	smallint	5	id rute (*primary key*)
src_id	smallint	5	id halte asal (*reference key* dari id pada tabel halte)
dest_id	smallint	5	id halte tujuan (*reference key* dari id pada tabel halte)
distance	double		jarak rute (dalam km)
route	varchar	300	id-id halte yang dilalui rute

4. Nama Tabel: cor_halte

Tabel 3.4 Struktur Tabel cor_halte

Nama Field	Tipe	Panjang	Keterangan
id	int	11	id data (*primary key*)
cor_id	varchar	50	id koridor
halte_id	varchar	50	id halte (*reference key* dari id pada tabel halte)

5. Nama Tabel: corridor_route

Tabel 3.5 Struktur Tabel corridor_route

Nama Field	Tipe	Panjang	Keterangan
id	int	5	id koridor (*reference key* dari cor_id pada tabel cor_halte)
route_name	varchar	100	nama rute

B. Nama *Database*: stasiun_krl

1. Nama Tabel: station

Tabel 3.6 Struktur Tabel station

Nama Field	Tipe	Panjang	Keterangan
id	int	11	id stasiun (*primary key*)
name	varchar	255	nama stasiun
latitude	varchar	255	nilai derajat lintang stasiun
longitude	varchar	255	nilai derajat bujur stasiun
place_id	varchar	100	place id stasiun

2. Nama Tabel: route_length

Tabel 3.7 Struktur Tabel route_length

Nama Field	Tipe	Panjang	Keterangan
id	int	5	id rute (*primary key*)
src_id	int	6	id stasiun asal (*reference key* dari id pada tabel station)
dest_id	int	6	id stasiun tujuan (*reference key* dari id pada tabel station)
mins	double		waktu dari stasiun asal ke tujuan (dalam menit)
distance	double		jarak antar stasiun (dalam km)

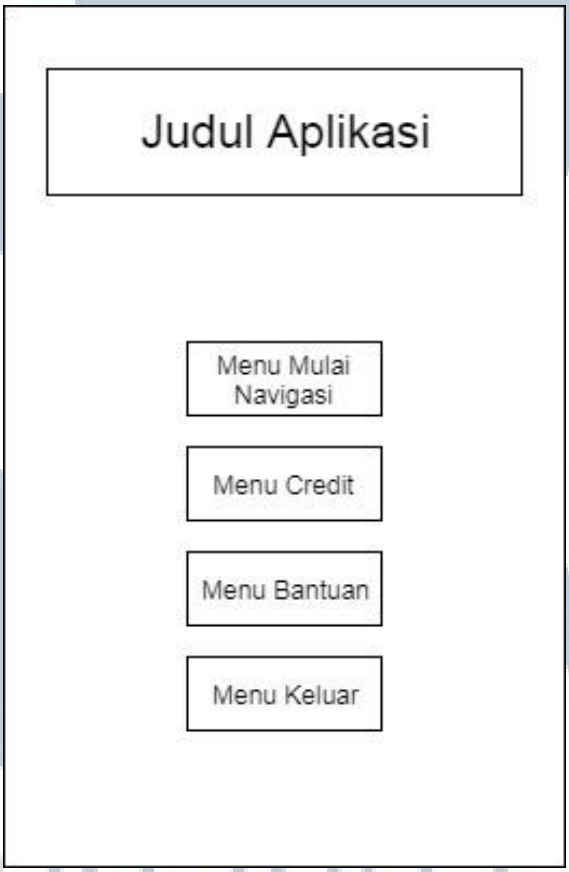
3. Nama Tabel: route_floyd

Tabel 3.8 Struktur Tabel route_floyd

Nama Field	Tipe	Panjang	Keterangan
id	smallint	5	id rute (*primary key*)
src_id	smallint	5	id stasiun asal (*reference key* dari id pada tabel station)
dest_id	smallint	5	id stasiun tujuan (*reference key* dari id pada tabel station)
distance	double		jarak rute (dalam km)
route	varchar	300	id-id stasiun yang dilalui rute

3.5.5 Desain Antarmuka

Aplikasi pencari rute terdekat yang dibuat memiliki rancangan tampilan antarmuka sehingga terdapat gambaran aplikasi yang dibuat setelah proses pembangunan aplikasi selesai dikerjakan. Perancangan tampilan antar muka memiliki enam bagian, sesuai dengan jumlah halaman yang ada pada aplikasi, yaitu halaman utama, halaman masukkan halte, halaman rute anda, halaman detail rute, halaman kredit, dan halaman bantuan. Rancangan halaman utama ditunjukkan pada Gambar 3.14.



Gambar 3.14 Rancangan Tampilan Antarmuka Halaman Utama

Halaman utama berisikan judul, menu mulai navigasi untuk berpindah ke halaman masukkan rute, menu credit untuk berpindah ke halaman kredit, menu

bantuan untuk berpindah ke halaman bantuan, dan menu keluar untuk keluar dari aplikasi. Rancangan halaman masukkan halte ditunjukkan pada Gambar 3.15.

Gambar 3.15 Rancangan Tampilan Antarmuka Halaman Masukkan Halte

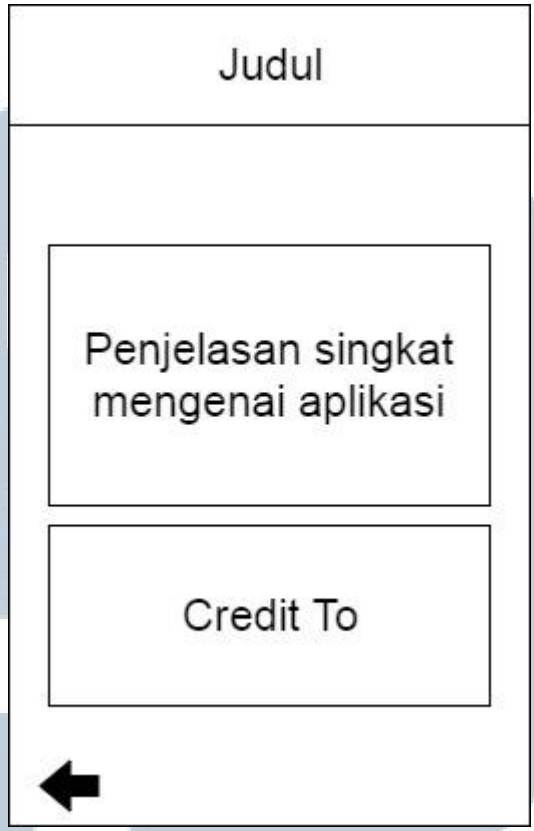
Halaman masukkan halte berisi tombol *back* untuk kembali ke halaman utama, *textbox* halte asal untuk memasukkan nama halte asal, *textbox* halte tujuan untuk memasukkan nama halte tujuan, dan tombol go untuk berpindah ke halaman rute anda. Rancangan tampilan rute anda ditunjukkan pada Gambar 3.16.

Gambar 3.16 Rancangan Tampilan Antarmuka Halaman Rute

Halaman rute anda berisi tombol *back* untuk kembali ke halaman masukkan rute, judul halaman "Peta Rute", *Google Maps* untuk menunjukkan lokasi halte awal, halte tujuan, dan rute yang dilalui, dan tombol detail rute untuk berpindah ke halaman detail rute. Rancangan tampilan antarmuka halaman detail rute ditunjukkan pada Gambar 3.17.

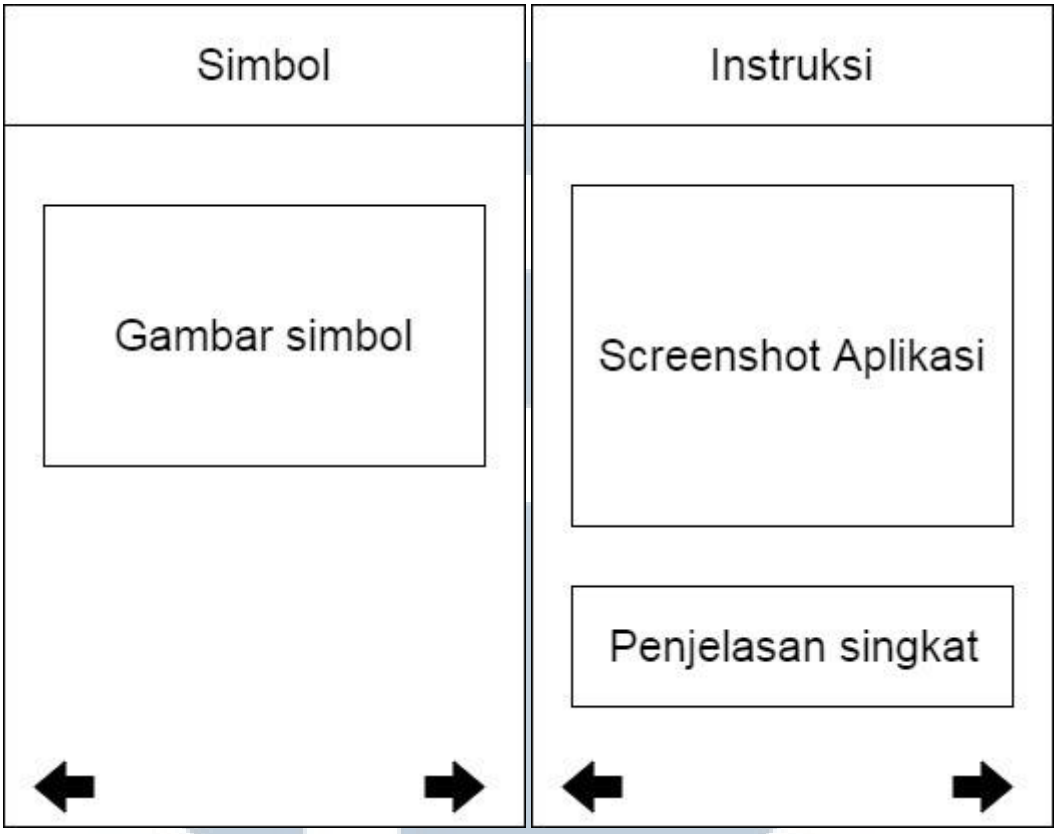
Gambar 3.17 Rancangan Tampilan Antarmuka Halaman Detail Rute

Halaman detail rute berisi tombol *back* untuk kembali ke halaman masukkan halte, judul halaman "Detail Rute", dan *scroll view* yang berisi detail dari rute, yaitu nama halte asal, nama halte tujuan, jarak antar halte, dan halte-halte apa saja yang dilalui selama perjalanan. Rancangan tampilan antarmuka halaman kredit ditunjukkan pada Gambar 3.18.



Gambar 3.18 Rancangan Tampilan Antarmuka Halaman Kredit

Halaman kredit berisi judul aplikasi, penjelasan singkat mengenai aplikasi, bagian *credit to* yang berisi nama *website* dan gambar yang berkaitan dengan isi *website*, dan tombol *back* untuk kembali ke halaman utama. Gambar pada bagian *credit to* dapat diklik untuk membuka *website* yang bersangkutan pada *browser*. Rancangan tampilan antarmuka halaman bantuan ditunjukkan pada Gambar 3.19.



Gambar 3.19 Rancangan Tampilan Antarmuka Halaman Bantuan

Rancangan tampilan antarmuka halaman bantuan dibagi menjadi dua bagian, yaitu halaman simbol dan halaman instruksi. Halaman simbol berisi judul halaman "Simbol", gambar dari simbol-simbol yang dipakai di dalam aplikasi, tombol *back* untuk kembali ke halaman utama, dan tombol *next* untuk berpindah ke halaman instruksi. Halaman instruksi memiliki empat halaman. Halaman pertama berisi judul halaman "Instruksi", *screenshot* dari aplikasi, penjelasan singkat tentang *screenshot*, tombol *back* untuk kembali ke halaman simbol, dan tombol *next* untuk berpindah ke halaman selanjutnya. Halaman kedua dan ketiga memiliki tampilan yang sama dengan halaman pertama, hanya saja tombol *back* berfungsi untuk kembali ke halaman sebelumnya, bukan ke halaman simbol. Halaman keempat memiliki tampilan yang sama dengan halaman kedua dan

ketiga tanpa tombol *next* karena halaman keempat merupakan halaman terakhir dan tidak ada lagi halaman selanjutnya setelah halaman keempat.