Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB III

PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Dalam melakukan analisa sistem terhadap aplikasi pencari jalur terpendek dengan metode algortima A* (A-Star) dan algortima *Dijkstra* bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ditemukan ketika pengujian pada aplikasi adalah pengoptimalan pencarian jalur. Pada analisis sistem perbandingan antara metode algortima *A** (*A-Star*) dan algoritma *Dijkstra* meliputi spesifikasi aplikasi, diantaranya perangkat keras (*hardware*), perangkat lunak (*software*). Metode penelitian yang digunakan adalah:

1. Studi Literatur

   Studi literatur dilakukan untuk mempelajari dan memahami teori-teori dasar yang berkenaan denagan penelitian, yaitu membaca jurnal serta buku-buku dan mencari informasi di internet yang berkaitan dengan algoritma *A** (*A-Star*) dan algortima *Dijkstra* serta perbandingannya untuk menentukan optimaisasi pencarian jalur.

2. Perancangan

   Perancangan dilakukan dengan membuat:

   - *flowchart* / diagram alur

     Menggambarkan proses / alur cara kerja algoritma *A** (*A-Star*), algortima *Dijkstra* dan aplikasi jarak terpendek secara garis besar.

   - rancangan tampilan antar muka

Perbandingan Algoritma ..., Nathan Sadeli, FTI UMN, 2014

Pembuatan sketsa aplikasi jarak terpendek pada penelitian ini, menggunakan perangkat lunak *Microsoft Office Visio* 2010.

3. *Coding*

   Melakukan *coding* aplikasi jalur terpendek dengan metode algorima *A** (*A-Star*) dan algoritma *Dijkstra* dengan bahasa pemrograman *Javascript*.

4. Uji coba (*testing*)

   Testing dilakukan dengan melakukan percobaan untuk beberapa jenis karakter, apakah aplikasi dapat memanipulasi, menganalisa, menampilkan informasi jalur terpendek. Kegiatan-kegiatan selama penelitian meliputi

   - mencoba dalam aplikasi yang telah dibangun menggunakan metode algortima *A** (*A-Star*) dengan menggunakan tabel uji coba Tabel 4.1.
   - mencoba dalam aplikasi yang telah dibangun menggunakan metode algortima *Dijkstra* dengan menggunakan tabel uji coba Tabel 4.2.
   - melihat hasil perbandingan antara kedua metode algoritma pencarian jalur terpendek baik secara waktu dan optimasi jalur yang dibentuk.

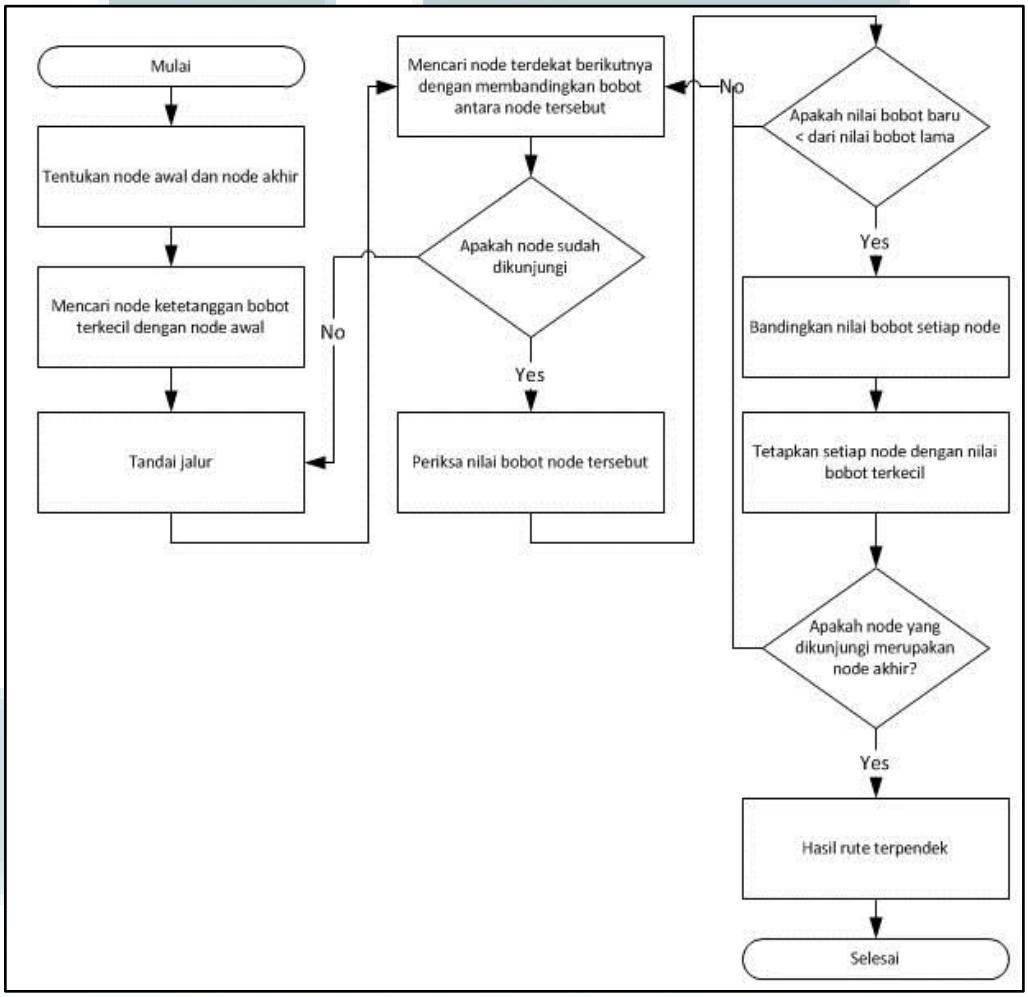
3.2 Perancangan

Perancangan dilakukan dengan melakukan pembuatan *Flowchart* dan rancang tampilan antar muka.

3.2.1 Flowchart

Gambar 3.1 menggambarkan *flowchart* untuk penggunaan aplikasi pencari jalur terpendek secara garis besar. Pengguna memulai aplikasi dan pada layar ditampilkan simpul yang diimplementasikan sebagai sutau lokasi digambarkan kotak-kotak, lalu pengguna meng-klik salah satu kotak sebagai penentu lokasi

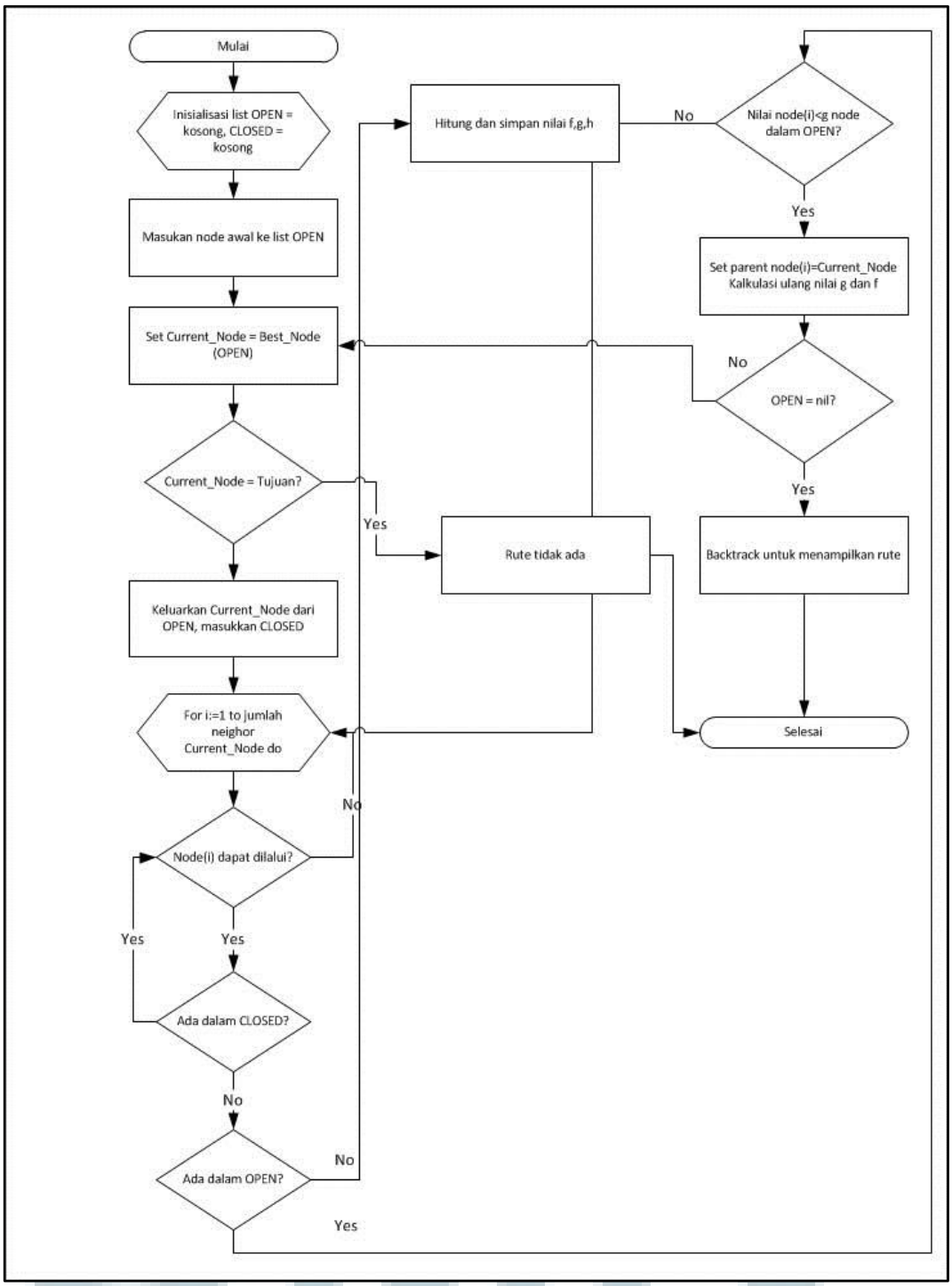
akhir, kemudian aplikasi akan menyimpan dimemori simpul awal dan akhir untuk diproses. Aplikasi akan melakukan perhitungan untuk mencari simpul ketetanggan yang memiliki bobot terkecil dari simpul awal, kemudian tandai simpul untuk menandai jalur. Dan seterusnya membandingkan apakah simpul sudah dikunjungi atau belum, kemudian aplikasi akan membandingkan bobot lama dan nilai bobot baru terhadap setiap simpul ketetanggan dan aplikasi menampilkan jalur terpendek.



Gambar 3.1 *Flowchart* Pencarian Jarak Terpendek

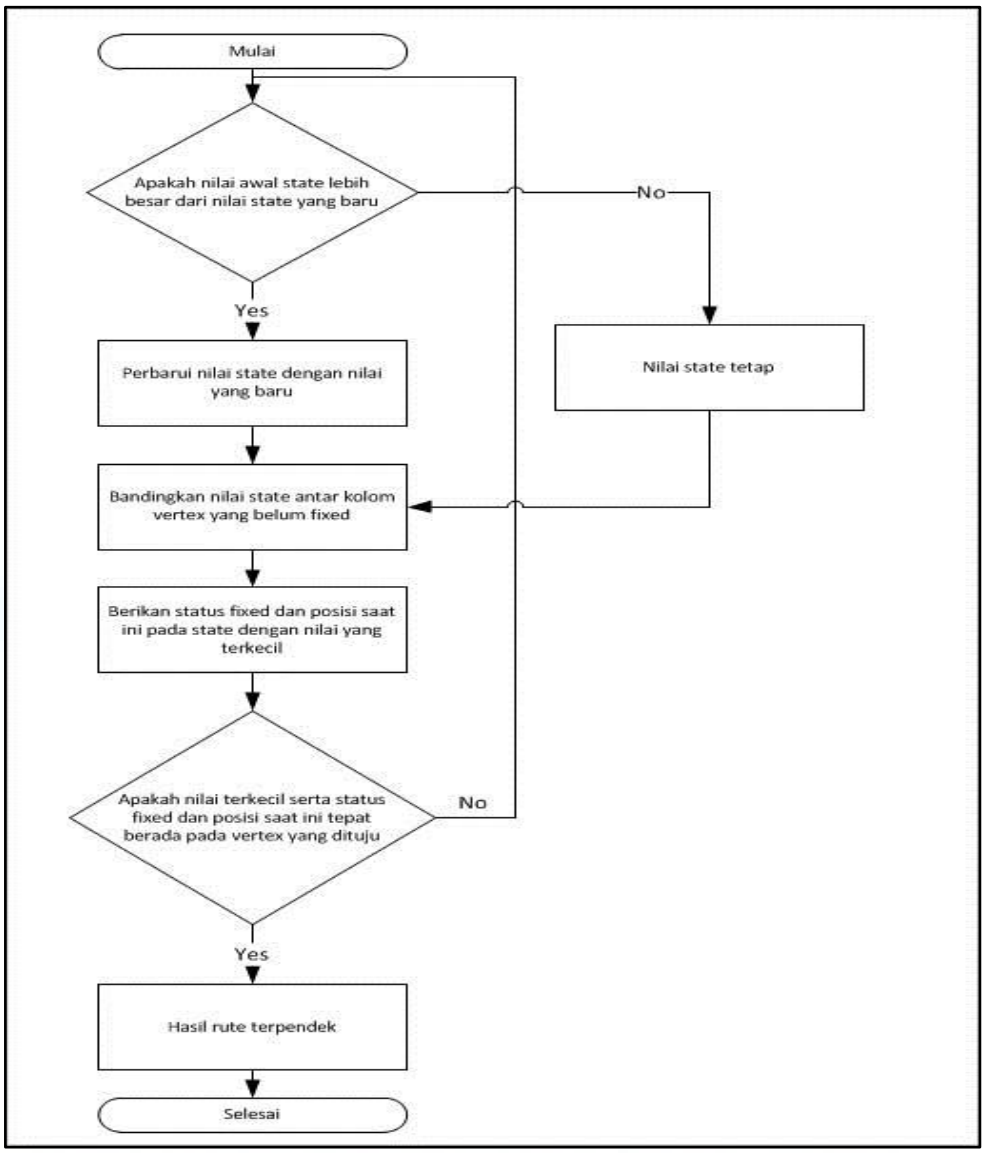
Gambar 3.2 menjelaskan cara kerja algoritma *A** (*A-Star*) bekerja dalam mencari rute paling optimum pada peta. Algoritma ini bekerja dari titik awal ke titik tujuan dengan mencari jalan dengan dengan akumulasi bobot terendah (dilambangkan dengan notasi g) dijumlah dengan perkiraan jalan terendah bobotnya untuk mencapai tujuan (dilambangkan dengan notasi h). Rute yang memiliki node-node dengan nilai f(n) = g(n) + h(n) yang paling kecil akan dikembangkan lebih dulu, karena itu proses pemberian bobot pada node-node sangatlah penting dalam menentukan hasil pencarian.

Proses mencari simpul dengan nilai terkecil ini dilakukan berulang-ulang sampai menemukan simpul tujuan. Apabila simpul yang menjadi titik tujuan telah ditemukan, program akan melakukan *backtrack* ke *parent* dari tiap simpul untuk mendapatkan rangkaian simpul yang membentuk rute yang akan paling optimum yang diinginkan. Penjelasan lebih lengkap tentang algoritma *A** (*A-Star*) beserta *pseudocode* telah dituliskan pada Bab II dalam laporan penelitian ini.



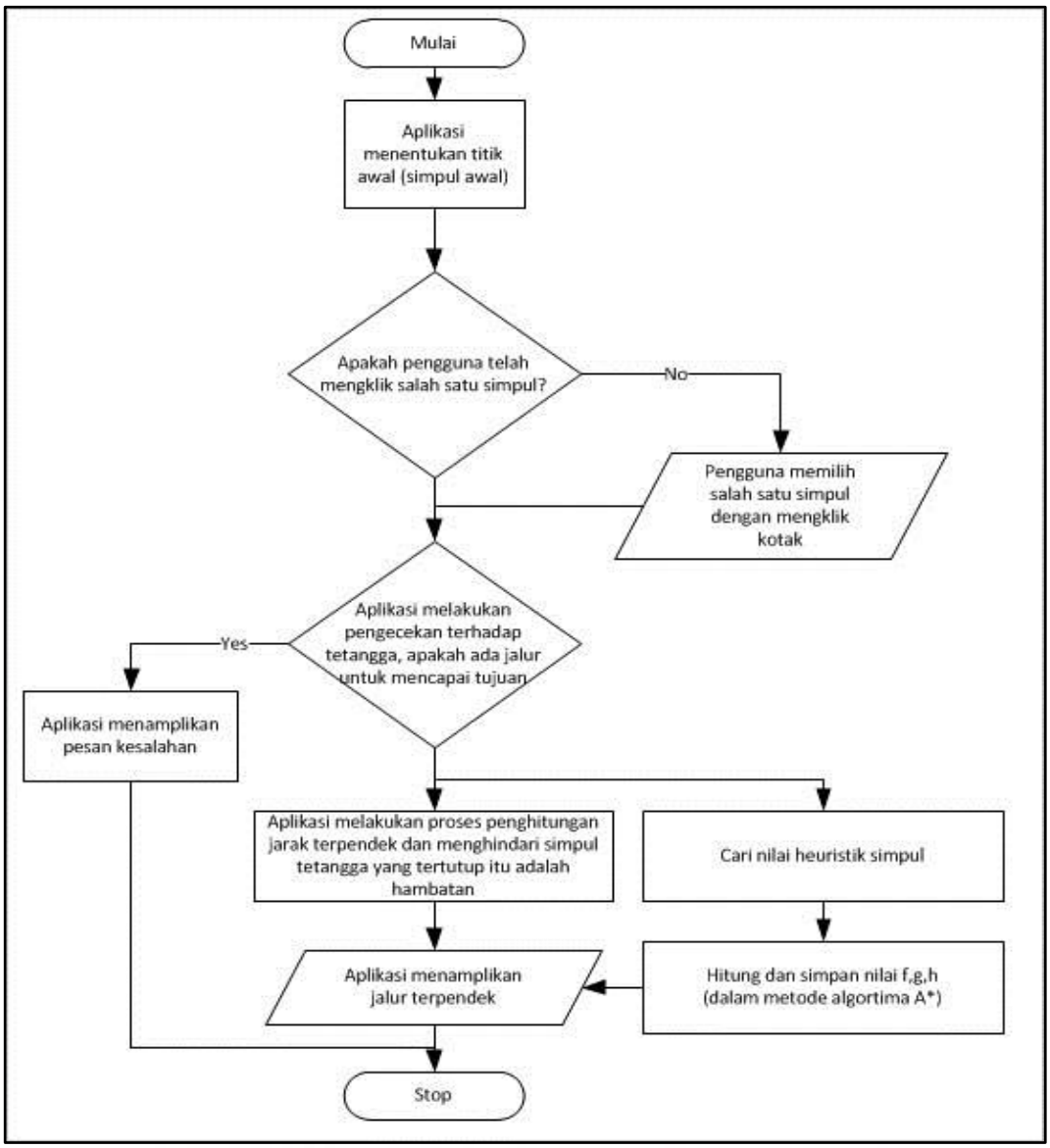
Gambar 3.2 *Flowchart* Algoritma *A** (*A-Star*)

Gambar 3.3 menggambarkan *flowchart* proses penentu jalur terpendek degan algoritma *Dijkstra*, algoritma ini merupakan salah satu yang umum digunakan untuk menyelesaikan masalah pencarian jalur terpendek. Algortima *Dijkstra* pertama kali mulai melakukan pembentukan tabel diperuntukan untuk menyimpan nilai bobot jarak, simpul tertandai sudah pernah dilalui, dan simpul sebelumnya dan simpul saat ini berada. Simpul awal diberi nilai bobot 0, kemudian saat semua simpul yang telah dilewati diberi tanda *false*, berarti algritma *Dijkstra* tidak bisa lagi melawati simpul tersebut. Setelah mengunjungi suatu simpul maka algoritma *Dijkstra* akan melakukan pembaruan data pada tabel yang pada awal dibentuk, yaitu memperbarui data daftar jarak dan daftar simpul sebelumnya berdasrkan simpul mana yang dapat segera dikunjungi dari simpul saat ini, terus diulang hingga semua simpul telah dikunjungi. Kemudian tandai dari simpul awal ke semua simpul sampai simpul akhir yang menandakan jalur terpendek.



Gambar 3.3 *Flowchart* Algoritma *Dijkstra*

Gamabr 3.4 adalah *flowchart* dari perencanaan penggunan aplikasi jarak terpendek dengan algoritma *A** (*A-Star*). Pengguna dapat menggunakan aplikasi pada *web browser*, seperti *Google Chrome*, selanjutnya aplikasi akan menentukan titik awal (simpul awal) untuk memulai jalur terpendek yang dihitung. Pengguna meng-klik pada salah satu kotak (pengimplementasian simpul), dan aplikasi akan menampilkan jalur terpendek.



Gambar 3.4 *Flowchart* Aplikasi Pencarian Jalur Terpendek Saat Digunakan

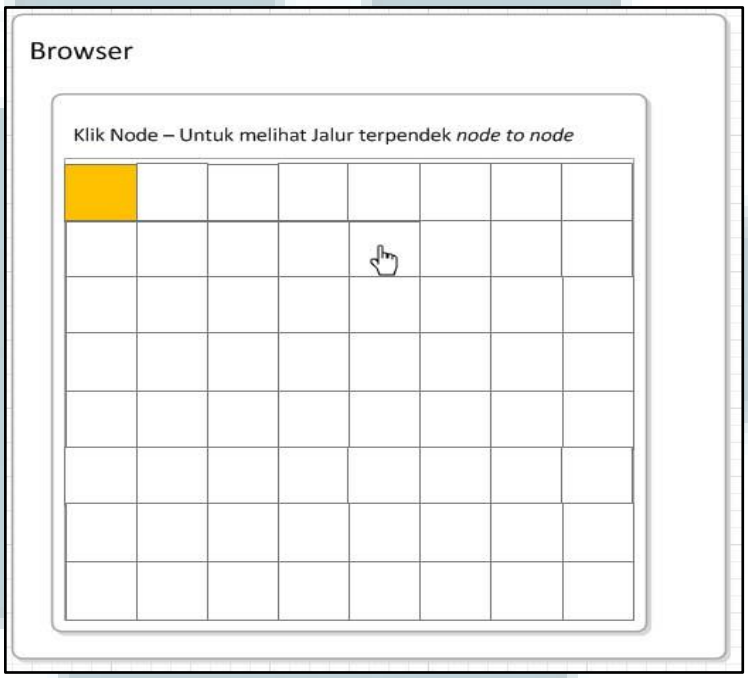
3.2.2 Rancangan Tampilan Antar Muka

Poin A-C merupakan rancangan tampilan antar muka dari aplikasi jarak terpendek yang dibuat:

A. Tampilan Awal Aplikasi

Gambar 3.5 merupakan tampilan awal aplikasi saat pengguna membuka aplikasi jalur terpendek, kemudian pengguna memilih salah satu kota yang

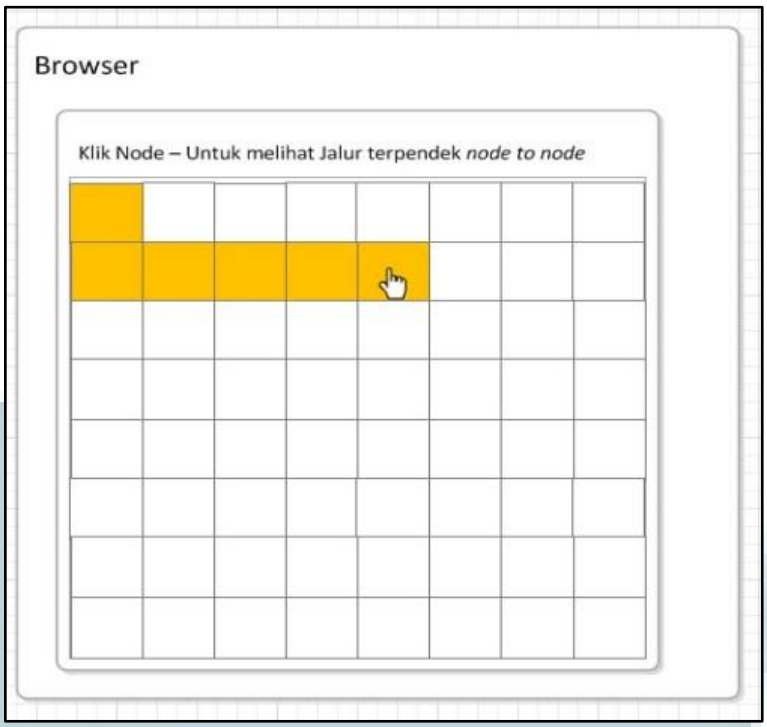
menginterpretasikan sebuah lokasi, dengan cara meng-klik kotak. Setelah pengguna memilih salah satu simpul, maka aplikasi akan melakukan proses dalam algoritma *A** dengan memetakan daerah sekitar simpul awal menuju simpul terdekat dan dipetakan lagi hingga mencapai simpul tujuan.



Gambar 3.5 Ilustrasi Tampilan Awal Aplikasi

B. Tampilan Aplikasi Jarak Terpendek

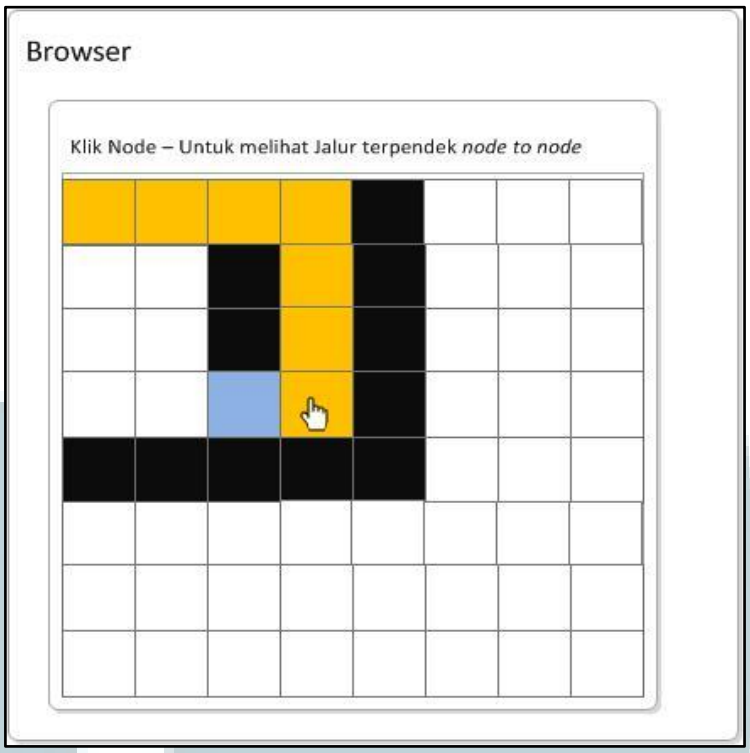
Gamabar 3.6 merupakan tampilan aplikasi setelah pengguna meng-klik salah satu simpul, yang kemudian aplikasi menampilkan jalur yang optimum atau jalur terpendek sesuai dengan algoritma *A**. Kondisi ini dapat berulang-ulang untuk melihat jalur lain, mulai dari simpul terakhir menjadi simpul awal untuk jalur terpendek yang baru.



Gambar 3.6 Ilustrasi Aplikasi Untuk Menetukan Jarak Terpendek

C. Tampilan Aplikasi Yang Memiliki Hambatan

Gambar 3.7 adalah gambar aplikasi ketika digunakan saat ada hambatan. Pada gambar tersebut ada simpul yang berwarna hitam adalah hambatan tidak mungkin dilewati, lalu ada yang warna biru adalah simpul yang memiliki bobot (*weight*) = 5 (lima), pada dasarnya simpul warna putih (warna dasar) memiliki bobot = 1. Terlihat pada gambar bahwa jalur terpendek menyusuri atas lalu belok ke bawah hingga simpul tujuan.



Gambar 3.7 Ilustrasi Yang Terjadi Ketika Aplikasi Memiliki Hambatan