PENGGUNAAN ALGORITMA DIJKSTRA

DALAM PENCARIAN RUTE TERCEPAT DAN RUTE TERPENDEK

(Studi Kasus Pada Jalan Raya antara Wilayah Blok M dan Kota)

IMRON FAUZI

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UIN SYARIF HIDAYATULLAH

JAKARTA

2011

PENGGUNAAN ALGORITMA DIJKSTRA

DALAM PENCARIAN RUTE TERCEPAT DAN RUTE TERPENDEK

(Studi Kasus Pada Jalan Raya antara Wilayah Blok M dan Kota)

Skripsi

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar

Sarjana Komputer

Fakultas Sains dan Teknologi

Universitas Islam Negeri Syarif Hidayatullah Jakarta

Oleh

Imron Fauzi

106091002883

Menyetujui,

Pembimbing 1	Pembimbing 2
Khodijah Hulliyah, M.Si	Hendra Bayu Suseno, M.Kom
NIP 19730402 200112 2 001	NIP 19821211 200912 1 003

Mengetahui,

Ketua Program Studi Teknik Informatika

Yusuf Durrachman, MSc, M.IT

NIP 19710522 200604 1 002

PENGESAHAN UJIAN

Skripsi yang berjudul "Penggunaan Algoritma Dijkstra Dalam Pencarian Rute Tercepat dan Rute Terpendek (Studi Kasus: Pada Jalan Raya antara Wilayah Blok M dan Kota)", telah diuji dan dinyatakan lulus dalam Sidang Munaqosah Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta, pada hari Jumat tanggal 10 Juni 2011. Skripsi ini telah diterima sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Teknik Informatika

Jakarta, Juni 2011

Tim Penguji,

Penguji I	Penguji II
Fitri Mintarsih, M.Kom	**Imam M Shofi, MT**
NIP 19721223 200710 2 004	**NIP 19720205 200801 1 010**

Tim Pembimbing

Pembimbing I	Pembimbing II
Khodijah Hulliyah, M.Sc	**Hendra Bayu Suseno, M.Kom**
NIP 19730402 200112 2 001	**NIP 19821211 200912 1 003**

Mengetahui,

Dekan	Ketua Program Studi
Fakultas Sains dan Teknologi	Teknik Informatika
DR. Syopiansyah Jaya Putra, M.Sis	**Yusuf Durrachman, MIT, M.Sc**
NIP 19680117 200112 1 001	**NIP 19710522 200604 1 002**

PERNYATAAN

DENGAN INI SAYA MENYATAKAN BAHWA SKRIPSI INI BENAR-BENAR HASIL KARYA SENDIRI YANG BELUM PERNAH DIAJUKAN SEBAGAI SKRIPSI ATAU KARYA ILMIAH PADA PERGURUAN TINGGI ATAU LEMBAGA MANAPUN

Tangerang Selatan, 27 Maret 2011

Imron Fauzi
106091002883

ABSTRAK

Imron Fauzi – 106091002883, Optimalisasi Penggunaan Algoritma Dijkstra dalam Pencarian Rute Tercepat dan Terpendek pada Jalan Raya di Jakarta, dibimbing oleh **Khodijah Hulliyah, M.Si** dan **Hendra Bayu Suseno, M.Kom**.

Kemacetan di Jakarta sudah menjadi pemandangan sehari-hari. Banyak langkah-langkah yang telah dilakukan oleh pemerintah untuk mengatasi kemacetan tersebut, seperti pembangunan *flyover* dan *underpass*, pengoperasian jalur *busway*, pemberlakuan jam *tree in one* dan sebagainya. Akan tetapi kemacetan tetap saja masih sering terjadi sampai saat ini. Oleh karena itu diperlukan peran aktif dari pengguna jalan sendiri untuk dapat mengatasi kemacetan tersebut. Salah satu cara yang paling efektif yaitu dengan mencari rute alternatif yang dapat dilalui. Sebelumnya telah dilakukan penelitian mengenai hal ini, namun belum bisa menjawab persoalan diatas karena kebanyakan penelitian tersebut hanya menggunakan parameter jarak tempuh. Oleh karena itu penulis mencoba membuat sebuah sistem yang menggunakan algoritma Dijkstra yang dapat menemukan jalur tercepat dan terpendek dengan menyertakan faktor kecepatan dan waktu tempuh perjalanan, ruang lingkup yang luas dan bersifat *online*. Adapun penggunaan algoritma Dijkstra karena algoritma ini dipastikan menemukan solusi terbaik dan memiliki kompleksitas yang lebih sedikit jika dibandingkan dengan algoritma sejenis seperti algoritma Bellman Ford dan Floyd Warshall. Pada pengembangan sistem ini penulis menggunakan metode *spiral model* dan kode program dibuat dengan menggunakan *framework* Code Igniter (CI). Sistem ini memberikan keluaran berupa jalur tercepat dan terpendek dari tempat asal menuju tempat tujuan yang diinputkan oleh pengguna. Jalur tercepat dan terpendek tersebut dilengkapi dengan total jarak tempuh, waktu tempuh serta kecepatan rata-rata.

Kata kunci : Algoritma Dijkstra, Rute Tercepat, Spiral Model

Jumlah Halaman : 150 halaman

Jumlah Daftar Pustaka : 30 sumber

KATA PENGANTAR

Bismillahirrahmanirrahiim……

Puji syukur kehadirat Allah SWT, atas rahmat dan hidayah Nya penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya, sholawat serta salam kita limpahkan kepada Nabi Muhammad SAW.

Skripsi ini penulis buat sebagai syarat kelulusan dalam menempuh pendidikan jenjang Strata-1 (S1) di Universitas Islam Negeri Syarif Hidayatullah Jakarta. Selain itu, penulis juga berharap penelitian ini dapat dipergunakan dengan baik oleh semua pihak yang membutuhkan dengan sebaik-baiknya untuk memberikan manfaat bagi kemajuan bangsa dan negara.

Pada kesempatan ini, penulis mengucapkan terima kasih kepada pihak-pihak yang telah membantu penulis menyelesaikan skripsi ini, yaitu:

1. Bapak Dr. Syopiansyah Jaya Putra, M.Sis selaku Dekan Fakultas Sains dan Teknologi.
2. Bapak Yusuf Durrachman, M.Sc, MIT selaku Ketua Program Studi Teknik Informatika.
3. Ibu Khodijah Hulliyah, M.Si selaku Pembimbing 1 dan Bapak Hendra Bayu Suseno, M.Kom selaku Pembimbing 2 yang telah membimbing penulis dalam menyelesaikan skripsi ini.
4. Bapak Husni Teja Sukmana, Ph.D selaku direktur PUSKOM UIN Syarif Hidayatullah Jakarta yang telah memberikan waktu kepada penulis untuk

menyelesaikan skripsi ini.

5. Bapak Agung Trianto H, S.SiT selaku Staf Seksi Manajemen Lalu Lintas Dinas Perhubungan Provinsi DKI Jakarta yang telah meluangkan waktunya untuk membantu penulis melengkapi data yang dibutuhkan dalam penelitian ini.
6. Ibu Fitri Mintarsih, M.Kom dan Bapak Imam M Shofi, MT sebagai penguji yang telah memberikan kritik dan saran sebagai perbaikan pada skripsi ini.
7. Dosen-dosen Fakultas Sains dan Teknologi yang telah mengajarkan kepada penulis berbagai macam ilmu yang dapat penulis terapkan dalam skripsi ini.
8. Kedua orang tua penulis, Bapak Maemun dan Ibu Warsiyah serta kakak dan adik penulis yang telah memberikan dukungan baik moril maupun materil sehingga penulis dapat menyelesaikan skripsi ini.
9. Kekasihku tercinta Imas Maslahah, yang selalu setia memberikan dukungan kepada penulis untuk menyelesaikan skripsi ini.
10. Semua teman-teman dan kerabat penulis baik di jurusan Teknik Informatika maupun di PUSKOM UIN Syarif Hidayatullah serta semua pihak yang telah membantu penulis menyelesaikan skripsi ini.

Penulis menyadari masih terdapat banyak kekurangan dalam penelitian ini baik penulisan maupun aplikasinya sendiri. Oleh karena itu penulis mengharapkan saran dan kritik yang dapat membangun skripsi ini lebih baik lagi.

Tangerang Selatan, Maret 2011

Penulis

DAFTAR ISI

Halaman

Halaman Judul i

Persetujuan Pembimbing ii

Halaman Pengesahan iii

Halaman Pernyataan iv

Abstrak v

Kata Pengantar vi

Daftar Isi viii

Daftar Gambar xiii

Daftar Tabel xviii

Daftar Simbol xxi

Daftar Lampiran xxiv

BAB I PENDAHULUAN

1.1. Latar Belakang 1

1.2. Perumusan Masalah 3

1.3. Pembatasan Masalah 3

1.4. Tujuan Penelitian dan Manfaat Penelitian 4

1.5. Metodologi Penelitian 5

1.6. Sistematika Penulisan 7

BAB II LANDASAN TEORI

2.1. Graf 8

2.1.1. Definisi Graf 8

2.1.2. Jenis-jenis Graf 9

2.2. Algoritma 10

2.2.1. Definisi Algoritma 10

2.2.2. Komponen Algoritma 11

2.2.3. Kode Semu (Pseudo Code) 12

2.3. Struktur Data 15

2.3.1. Tree (Pohon) 15

2.3.2. Stack (Tumpukan) 16

2.3.3. Queue 16

2.4. Algoritma Traversal Pada Graf 17

2.4.1. Algoritma Pencarian Melebar (BFS) 17

2.4.2. Algoritma Depth First Search (DFS) 20

2.5. Algoritma Shortest Path 23

2.5.1. Algoritma Dijkstra 23

2.5.2. Algoritma Bellman Ford 30

2.5.3. Algoritma Floyd Warshall 32

2.5.4. Perbandingan Algoritma Shortest Path 35

2.6. Transportasi 36

2.6.1. Definisi Sistem Transportasi 36

2.6.2. Pemodelan Jaringan Transportasi 37

2.6.3. Pemilihan Rute Terpendek Pada Jaringan Jalan 37
2.7. Metodologi Penelitian 38
2.7.1. Metode Pengumpulan Data 38
2.7.2. Metode Pengembangan Sistem 39
2.8. Unified Modeling Language (UML) 42
2.8.1. Use Case Diagram 43
2.8.2. Class Diagram 45
2.8.3. Activity Diagram 46
2.8.4. Sequence Diagram 48
2.8.5. Deployment Diagram 50
2.9. PHP 52
2.9.1. Definisi PHP 52
2.9.2. Penggunaan PHP 52
2.10. Database MySQL 53
2.10.1. Definisi Database MySQL 53
2.10.2. Penggunaan Database MySQL 54
2.11. Pemrograman Model View Control (MVC) 55
2.11.1. Definisi Pemrograman Model View Control (MVC) 55
2.11.2. Model 56
2.11.3. View 57
2.11.4. Controller 57
2.12. Frame Work 57
2.12.1. Definisi Frame Work 57

2.12.2. Frame Work Code Igniter 57

BAB III METODOLOGI PENELITIAN

3.1. Metode Pengumpulan Data 59

3.1.1. Observasi 59

3.1.2. Studi Pustaka 59

3.1.3. Wawancara 60

3.1.3. Kuisioner 60

3.2. Metode Pengembangan Sistem 60

3.2.1. Spiral Model 60

3.3. Kerangka Berpikir 64

BAB IV ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem 66

4.2. Perancangan Sistem 84

4.2.1. Modeling 84

4.2.2. Construction 122

4.2.3. Deployment 147

BAB V KESIMPULAN DAN SARAN

5.1. Kesimpulan 148

5.2. Saran 148

DAFTAR PUSTAKA 150

LAMPIRAN – LAMPIRAN

DAFTAR GAMBAR

Gambar 2.1 Sebuah Graf Sederhana 9

Gambar 2.2 Graf Tidak Berarah 9

Gambar 2.3 Graf Berarah 10

Gambar 2.4 Struktur dan Jenis Algoritma 11

Gambar 2.5 Tree dan Komponennya 15

Gambar 2.6 Stack 16

Gambar 2.7 Queue 17

Gambar 2.8 Urutan Penelusuran Node pada Algoritma BFS 18

Gambar 2.9 Cara Kerja Algoritma BFS 20

Gambar 2.10 Urutan Penelusuran Algoritma DFS 21

Gambar 2.11 Cara Kerja Algoritma DFS 22

Gambar 2.12 Cara Kerja Algoritma Dijkstra 30

Gambar 2.13 Cara Kerja Algoritma Bellman Ford 32

Gambar 2.14 Cara Kerja Algoritma Floyd Warshall 34

Gambar 2.15 Spiral Model 40

Gambar 2.16 Contoh Use Case Diagram 44

Gambar 2.17 Contoh Class Diagram 46

Gambar 2.18 Contoh Activity Diagram 48

Gambar 2.19 Contoh Sequence Diagram 50

Gambar 2.20 Contoh Deployment Diagram 51

Gambar 2.21 Konsep Model View Control (MVC) 56

Gambar 3.1 Kerangka Berpikir 65

Gambar 4.1 Pemodelan Graf 69

Gambar 4.2 Graf Sederhana Sebagai Sampel 73

Gambar 4.3 Kondisi Graf Saat Node A4 dan C7 Ditemukan 75

Gambar 4.4 Kondisi Graf Saat Node A3 dan C6 Ditemukan 77

Gambar 4.5 Kondisi Graf Saat Node C3 Ditemukan 78

Gambar 4.6 Kondisi Graf Saat Node C4 Ditemukan 80

Gambar 4.7 Kondisi Graf Saat Node C7 Dikunjungi 81

Gambar 4.8 Kondisi Graf Saat Akhir Pencarian 83

Gambar 4.9 Use Case Diagram User dan Login Admin 85

Gambar 4.10 Use Case Node 86

Gambar 4.11 Use Case Path 86

Gambar 4.12 Use Case Wilayah 87

Gambar 4.13 Use Case Jalan 87

Gambar 4.14 Class Diagram 88

Gambar 4.15 Activity Diagram Menu Home 90

Gambar 4.16 Activity Diagram Menu Login Admin 91

Gambar 4.17 Activity Diagram Menu Data Node 92

Gambar 4.18 Activity Diagram Menu Data Path 93

Gambar 4.19 Activity Diagram Menu Data Wilayah 94

Gambar 4.20 Activity Diagram Menu Data Jalan 95

Gambar 4.21 Activity Diagram Menu Laporan 96

Gambar 4.22 Activity Diagram Menu Grafik 96

Gambar 4.23 Activity Diagram Menu Pengaturan Data 97

Gambar 4.24 Activity Diagram Menu Ubah Password 98

Gambar 4.25 Activity Diagram Menu Panduan 98

Gambar 4.26 Activity Diagram Menu Daftar Persimpangan 99

Gambar 4.27 Sequence Diagram User 101

Gambar 4.28 Sequence Diagram Admin 102

Gambar 4.29 Sequence Diagram Node Index 103

Gambar 4.30 Sequence Diagram Insert Data Node 103

Gambar 4.31 Sequence Diagram Update Data Node 104

Gambar 4.32 Sequence Diagram Delete Data Node 104

Gambar 4.33 Sequence Diagram Search Data Node 105

Gambar 4.34 Sequence Diagram Path Index 105

Gambar 4.35 Sequence Diagram Insert Data Path 106

Gambar 4.36 Sequence Diagram Update Data Path 106

Gambar 4.37 Sequence Diagram Delete Data Path 107

Gambar 4.38 Sequence Diagram Search Data Path 107

Gambar 4.39 Sequence Diagram Wilayah Index 108

Gambar 4.40 Sequence Diagram Insert Data Wilayah 108

Gambar 4.41 Sequence Diagram Update Data Wilayah 109

Gambar 4.42 Sequence Diagram Delete Data Wilayah 109

Gambar 4.43 Sequence Diagram Search Data Wilayah 110

Gambar 4.44 Sequence Diagram Jalan Index 110

Gambar 4.45 Sequence Diagram Insert Data Jalan 111

Gambar 4.46 Sequence Diagram Update Data Jalan 111

Gambar 4.47 Sequence Diagram Delete Data Jalan 112

Gambar 4.48 Sequence Diagram Search Data Jalan 112

Gambar 4.49 Sequence Diagram Laporan 113

Gambar 4.50 Sequence Diagram Grafik 113

Gambar 4.51 Sequence Diagram Panduan 113

Gambar 4.52 Sequence Diagram Daftar Persimpangan 114

Gambar 4.53 Deployment Diagram Sistem 115

Gambar 4.54 Relasi Tabel-Tabel Yang Digunakan pada Sistem 120

Gambar 4.55 Perancangan Layout Halaman User 121

Gambar 4.56 Perancangan Layout Halaman Login Admin 121

Gambar 4.57 Perancangan Layout Halaman Admin 122

Gambar 4.58 Struktur Fungsi Algoritma Dijkstra 124

DAFTAR TABEL

Tabel 2.1 Perbandingan Beberapa Algoritma Shortest Path 35

Tabel 2.2 Simbol pada Use Case Diagram 43

Tabel 2.3 Simbol pada Class Diagram 45

Tabel 2.4 Simbol pada Activity Diagram 47

Tabel 2.5 Simbol pada Sequence Diagram 49

Tabel 2.6 Simbol pada Deployment Diagram..... 51

Tabel 3.1 Penjadwalan Pengembangan Sistem 61

Tabel 4.1 Daftar Node pada Pemodelan Graf 70

Tabel 4.2 Kondisi Node pada Saat Awal Pencarian 74

Tabel 4.3 Kondisi Node pada Saat Node A4 dan C7 Ditemukan 76

Tabel 4.4 Kondisi Node pada Saat Node A3 dan C6 Ditemukan 77

Tabel 4.5 Kondisi Node pada Saat Node C3 Ditemukan 79

Tabel 4.6 Kondisi Node pada Saat Node C4 Ditemukan 80

Tabel 4.7 Kondisi Node pada Saat Node C7 Ditemukan 81

Tabel 4.8 Kondisi Node pada Akhir Pencarian 83

Tabel 4.9 Tabel Node 115

Tabel 4.10 Tabel Path 116

Tabel 4.11 Tabel Wilayah 117

Tabel 4.12 Tabel Jalan 117

Tabel 4.13 Tabel Pengukuran 118

Tabel 4.14 Tabel Recordpath 119

Tabel 4.15 Tabel Path 119

Tabel 4.16 Tabel User 120

Tabel 4.17 Kelas-Kelas yang Terdapat pada Sistem 125

Tabel 4.18 Tabel Testing Halaman Home 126

Tabel 4. 19 Tabel Testing Halaman Simulasi 128

Tabel 4.20 Tabel Testing Halaman Login Admin 130

Tabel 4.21 Tabel Testing Halaman Data Node 131

Tabel 4.22 Tabel Testing Halaman Data Path 133

Tabel 4.23 Tabel Testing Halaman Data Wilayah 135

Tabel 4.24 Tabel Testing Halaman Data Jalan 137

Tabel 4.25 Tabel Testing Halaman Laporan 138

Tabel 4.26 Tabel Testing Halaman Ubah Password 139

Tabel 4.27 Tabel Testing Halaman Grafik 140

Tabel 4.28 Tabel Testing Halaman Daftar Persimpangan 140

Tabel 4.29 Tabel Testing Halaman Panduan 141

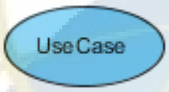
Tabel 4.30 Tabel Testing Halaman Pengaturan Data 142

Tabel 4.31 Tabel Testing Kemungkinan Algoritma Dijkstra 143

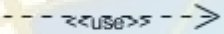
Tabel 4.32 Spesifikasi Software untuk *Running* Sistem 147

DAFTAR SIMBOL

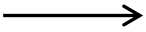
A. Use Case Diagram

Simbol	Nama Simbol	Kegunaan
	Aktor	Sebagai subjek yang berinteraksi atau menggunakan system
	Use Case	Sebagai kegiatan yang dapat dilakukan oleh pengguna pada system
	Asosiasi	Sebagai pengubung antara aktor dan use case yang dilakukan
	Include	Sebagai penghubung antara use case yang membutuhkan use case yang lain
	Sistem	Sebagai cakupan wilayah system

B. Class Diagram

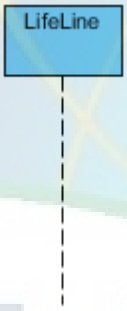
Simbol	Nama Simbol	Kegunaan
	*Class*	Sebagai kelas yang digunakan pada system
	*Generalization*	Menunjukan hubugan *inheritance* antar kelas
	Usage	Menunjukan hubungan penggunaan suatu kelas dengan kelas yang lain

C. Activity Diagram

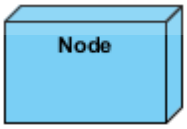
Simbol	Nama Simbol	Kegunaan
	*Inisial node*	Awal aktifitas
	*Final node*	Akhir aktifitas
	*Action*	Sebagai aktifitas yang dilakukan oleh system
	*Control Flow*	Sebagai penghubung urutan aktifitas
	*Decision*	Merupakan aktifitas pengecekan kondisi
	*Exception handler*	Menunjukan kondisi pengecualian

		apabila suatu *action* tidak dapat dilakukan

D. Sequence Diagram

Simbol	Nama Simbol	Kegunaan
	*Actor*	Sebagai subjek yang menggunakan system
	*Life line*	Bagian dari sistem yang melakukan aktifitas pemrosesan data
	*Message*	Alur data yang diproses oleh system

E. Deployment Diagram

Simbol	Nama Simbol	Kegunaan
	*Node*	Komponen atau perangkat yang digunakan pada proses deployment system
	Asosiation	Sebagai penghubung aktifitas yang dilakukan antar komponen

DAFTAR LAMPIRAN

Lampiran 1 Interface Hasil Pengujian Sistem L-1

Lampiran 2 Data Observasi L-2

Lampiran 3 Wawancara L-3

Lampiran 4 Kuisioner L-4

Lampiran 5 Source Code Sistem L-5

Lampiran 6 Surat Keterangan L-6

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kemacetan di Jakarta selalu terjadi setiap hari kerja. Jumlah kepadatan penduduk yang besar dan aktifitas yang beragam membuat kota ini tidak pernah sepi dari lalu lintas manusia. Mobilitas yang tinggi tersebut sering terkendala dengan minimnya sarana dan prasarana jalan yang sangat terbatas. Oleh karena itu diperlukan sebuah solusi yang tepat yang dapat memperlancar mobilitas tersebut. Salah satu solusi yaitu dengan menambah sarana dan prasarana jalan yang saat ini sangat terbatas. Akan tetapi hal ini sulit dilakukan selain karena tidak adanya lahan untuk perluasan jalan, tetapi juga karena hal ini membutuhkan dana yang sangat besar. Kita bisa ambil contoh banyak proyek semacam itu yang sampai saat ini masih terbengkalai.

Solusi lain yang mungkin dilakukan adalah dengan menggunakan kecermatan kita sendiri memilih dan memanfaatkan sarana yang tersedia dengan semaksimal mungkin. Hal ini akan sangat membantu kita untuk menghemat waktu yang diperlukan dalam menempuh perjalanan dari satu tempat ke tempat lain yang ada di Jakarta.

Sebelumnya telah dilakukan penelitian mengenai hal ini, salah satu penelitian tersebut yaitu penelitian tugas akhir yang dilakukan oleh Yadi Rusyad Nurdin yang berjudul "***Implementasi Algoritma Dijkstra Dalam Menentukan Rute Terpendek Pada Dua Titik Lokasi Wilayah Menteng***

Raya". Namun penelitian tersebut memiliki beberapa kekurangan diantaranya:

1. Penelitian yang dilakukan hanya sebatas menemukan jalur terpendek berdasarkan jarak, sedangkan pada permasalahan kemacetan yang terjadi saat ini sistem tersebut tidak dapat diterapkan.
2. Ruang lingkup penelitian sangat kecil, sehingga tidak bisa menjawab persoalan masyarakat yang cenderung lebih luas. Contohnya seperti masyarakat yang bekerja pada wilayah perkantoran di pusat kota Jakarta yang ingin mencari rute tercepat menuju ke kantor mereka atau rute tercepat dari suatu kantor ke kantor yang lainnya.
3. Titik lokasi yang dijadikan input bersifat tetap, tidak bisa dilakukan penambahan, pengurangan dan perubahan data oleh sistem. Begitu juga mengenai jarak yang menghubungkan antara satu titik ke titik yang lain.
4. Sistem yang dihasilkan bersifat sistem desktop sehingga tidak bisa diakses secara *on line*.

Oleh karena itu penulis mencoba membuat sebuah sistem yang dapat menemukan jalur tercepat dan terpendek dengan menyertakan faktor kecepatan dan waktu tempuh perjalanan, ruang lingkup yang lebih luas, bersifat *on line* serta titik lokasi yang dijadikan input dapat dilakukan perubahan. Sehingga sistem yang dihasilkan lebih bermanfaat bagi pengguna serta dapat diakses kapanpun dan dimanapun. Dalam hal ini, penulis menggunakan judul penelitian **Penggunaan Algoritma Dijkstra**

Untuk Pencarian Jalur Tercepat dan Jalur Terpendek (Studi Kasus Pada Jalan Raya antara Wilayah Blok M dan Kota). Dimana pengguna hanya menginput lokasi asal dan lokasi tujuan. Adapun penggunaan algoritma Dijkstra dalam penelitian ini dilakukan karena algoritma ini sangat cocok dan efisien dalam mencari path dengan bobot terkecil pada sebuah graph.

1.2. Perumusan Masalah

Beberapa permasalahan yang dapat dirumuskan pada penelitian ini diantaranya:

1. Bagaimana cara menemukan jalur tercepat dan jalur terpendek dari satu lokasi ke lokasi lain yang ada di Jakarta khususnya antara wilayah Blok M dan Kota?
2. Faktor-faktor apa saja yang harus diperhitungkan dalam mencari jalur tersebut?

1.3. Batasan Masalah

Batasan masalah yang dilakukan pada penelitian ini yaitu:

1. Penelitian ini hanya dilakukan di wilayah Jakarta (khususnya antara wilayah Blok M dan Kota).
2. Objek pada penelitian ini hanya pada jalan raya, yaitu jalan umum yang dapat dilalui oleh kendaraan yang berukuran besar seperti Bus.
3. Penelitian ini hanya sebatas menemukan jalur tercepat yang dapat digunakan oleh pengguna.
4. Data kecepatan dan waktu tempuh pada penelitian ini menggunakan

acuan kendaraan roda empat yang didapatkan dari pihak Dishub Provinsi DKI Jakarta.

1.4. Tujuan dan Manfaat Penelitian

1.4.1. Tujuan Penelitian

Penelitian ini penulis lakukan dengan beberapa tujuan. Adapun tujuan tersebut sebagai berikut.

1. Membantu pengguna (masyarakat) yang ingin menemukan jalur tercepat dari satu lokasi ke lokasi lainnya yang ada di Jakarta khususnya antara wilayah Blok M dan Kota.
2. Menemukan faktor-faktor yang mempengaruhi waktu tempuh perjalanan di Jakarta.

1.4.2. Manfaat Penelitian

1.4.2.1. Bagi Mahasiswa

Beberapa manfaat yang diperoleh penulis dalam penelitian ini yaitu:

1. Penulis dapat menerapkan ilmu yang didapatkan dalam perkuliahan ke dalam sebuah permasalahan yang ada di masyarakat.
2. Penulis dapat mempelajari secara mendalam mengenai algoritma Dijkstra dalam melakukan pencarian terhadap jalur tercepat.

1.4.2.2. Bagi Masyarakat

Adapun beberapa manfaat bagi masyarakat (pengguna) yaitu:

1. Mempermudah masyarakat dalam menemukan jalur terbaik untuk menempuh perjalanan dari suatu tempat ke tempat lain yang ada di Jakarta khususnya antara wilayah Blok M dan Kota.
2. Membantu masyarakat untuk mengetahui jalur alternatif yang dapat digunakan, ketika jalur yang biasa mereka lalui tidak efektif dan memakan waktu yang lama.

1.5. Metodologi Penelitian

Metodologi penelitian yang penulis gunakan pada penelitian ini yaitu sebagai berikut.

1.5.1. Metode Pengumpulan Data

Berikut ini metode pengumpulan data yang penulis lakukan yaitu:

1. Observasi ke kantor Dinas Perhubungan Provinsi DKI Jakarta dan ke wilayah penelitian secara langsung.
2. Wawancara terhadap perwakilan pegawai Dinas Perhubungan Provinsi DKI Jakarta.
3. Studi pustaka, yaitu mengumpulkan data melalui buku, jurnal dan artikel yang berkaitan dengan penelitian yang penulis

lakukan.

1.5.2. Metode Pengembangan Sistem

Metode pengembangan sistem yang penulis gunakan pada penelitian ini yaitu metode S*piral Model*. Metode ini terdiri dari beberapa tahapan yaitu:

1. Tahap *communication*, pada tahap ini penulis melakukan komunikasi secara langsung dengan pihak Dinas Perhubungan Provinsi DKI Jakarta.
2. Tahap *planning*, pada tahap ini penulis melakukan estimasi kebutuhan sistem, penjadwalan pembangunan sistem dan analisis resiko.
3. Tahap *modeling*, pada tahap ini penulis melakukan pemodelan terhadap sistem yang akan dibangun.
4. Tahap *construction*, tahap ini terdiri dari tahap *coding* dan tahap *testing*. Pada tahap *coding* penulis membangun kode program untuk digunakan pada sistem, sementara itu pada tahap testing penulis melakukan pengujian terhadap sistem yang telah dibangun dengan menggunakan metode *black box testing*.
5. Tahap *deployment*, pada tahap ini penulis melakukan penerapan atau *running* sistem yang telah penulis bangun pada tahap *construction*.

1.6. Sistematika Penulisan

BAB I PENDAHULUAN

Berisi mengenai latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi mengenai uraian teori yang digunakan pada penelitian ini. Uraian teori tersebut yaitu mengenai graf, algoritma, struktur data, algoritma traversal pada graf, algoritma shortest path, transportasi, metode penelitian, pemodelan UML, PHP, database MySQL, pemrograman *Model View Control (MVC)* serta *frame work.*

BAB III METODOLOGI PENELITIAN

Berisi mengenai metode-metode yang digunakan pada penelitian ini yaitu metode pengumpulan data dan metode pengembangan sistem.

BAB IV ANALISIS DAN PERANCANGAN SISTEM

Berisi mengenai analisis masalah yang diteliti serta penerapan dari metode pengembangan sistem yang digunakan.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan dari penelitian yang telah dilakukan, serta saran untuk pengembangan penelitian selanjutnya.

BAB II

LANDASAN TEORI

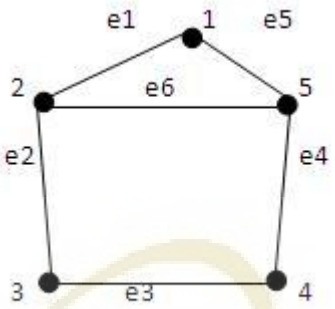
2.1. Graf

2.1.1. Definisi Graf

Teori graf merupakan pokok bahasan yang memiliki banyak terapan sampai saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan dengan objek-objek tersebut.

Secara matematis graf didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi G = (V, E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (vertex atau node) dan E adalah himpunan sisi (edge) yang menghubungkan sepasang simpul (Munir, 2005).

Simpul (vertex) pada graf dapat dinyatakan dengan huruf, bilangan atau gabungan keduanya. Sedangkan sisi-sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e1, e2, e3 dan seterusnya. Dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v, maka e dapat dituliskan sebagai e = (u, v).



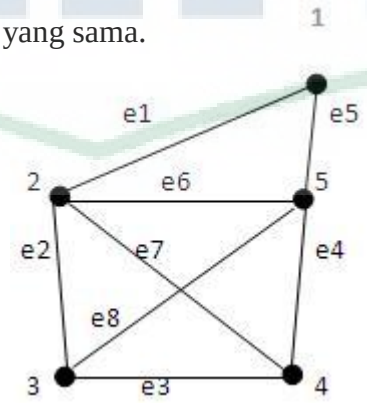
Gambar 2.1 Sebuah Graf Sederhana

2.1.2. Jenis-Jenis Graph

Klasifikasi pada graf cukup luas, klasifikasi tersebut bergantung pada faktor-faktor yang membedakannya. Berdasarkan orientasi arah pada sisinya, maka secara umum graf dibedakan atas dua jenis sebagai berikut :

1. Graf tidak berarah (*undirected graph*)

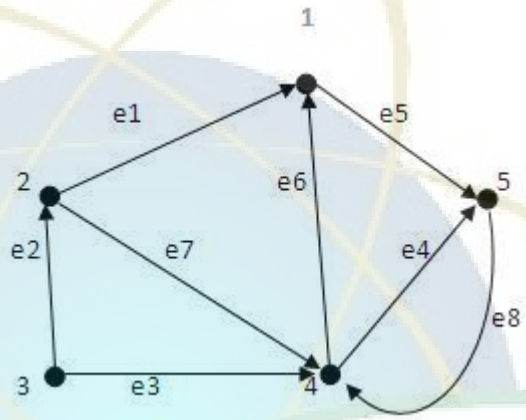
   Graf yang sisinya tidak memiliki orientasi arah disebut graft tidak berarah. Pada graf tidak berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, (u, v) = (v, u) adalah sisi yang sama.



Gambar 2.2 Graf Tidak Berarah

2. Graf berarah (*directed graph*)

   Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah, pada graf berarah (u, v) dan (v, u) menyatakan dua buah sisi yang berbeda. Dengan kata lain dapat ditulis (u, v) ≠ (v, u).



Gambar 2.3 Graf Berarah

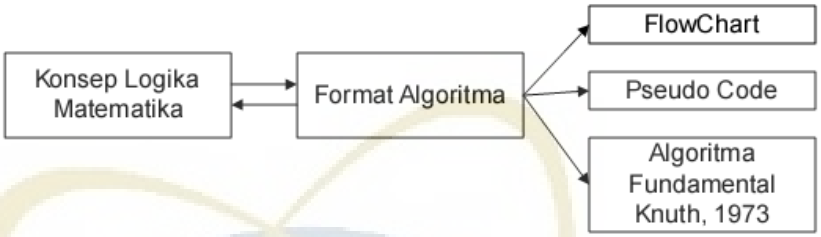
Adapun graf yang memiliki nilai bobot pada setiap sisinya disebut graf berbobot (*weighted graph*).

2.2. Algoritma

2.2.1. Definisi Algoritma

Kata algoritma (algorithm) berasal dari kata algorism yang diambil dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu Musa Al-Khuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi algorism)(Munir, 2005). Adapun pengertian algoritma adalah kumpulan instruksi atau perintah yang dibuat secara jelas dan sistematis berdasarkan urutan yang logis untuk penyelesaian suatu masalah.

Knuth (1973) menyarankan algoritma fundamental sebagaimana yang dituliskan pada Nurhayati (2010).



Gambar 2.4 Struktur dan Jenis Algoritma

2.2.2. Komponen Algoritma

Knuth (1973) sebagaimana yang dituliskan pada Nurhayati (2010) menyatakan 5 komponen utama dalam algoritma yaitu finiteness, definitenesss, input, output dan effectiveness. Sehingga dalam merancang sebuah algoritma ada tiga komponen yang harus ada yaitu:

1. **Komponen masukan (input)**

   Komponen ini biasanya terdiri dari pemilihan variabel, jenis variabel, tipe variabel, konstanta dan paramenter (dalam fungsi).

2. **Komponen keluaran (output)**

   Komponen ini merupakan tujuan dari perancangan algoritma dan program. Permasalahan yang diselesaikan dalam algoritma dan program harus ditampilkan dalam komponen keluaran. Karakteristik keluaran yang baik adalah keluaran yang benar menjawab permasalahan dan tampilan (*interface*) yang ramah.

3. **Komponen proses (*processing*)**

   Komponen ini merupakan bagian utama dan terpenting dalam merancang sebuah algoritma. Dalam bagian ini terdapat logika masalah, logika algoritma (sintaksis dan semantik), rumusan, metode (rekursi, perbandingan, penggabungan, pengurangan dan lain-lain).

Adapun jika dilihat dari segi kondisinya, kompoenen algoritma terdiri dari dua kondisi yaitu:

1. ***Pre condition***

   Pre condition adalah kondisi suatu program ketika algoritma siap dijalankan (sebelum dilaksanakan algoritma). Dengan kata lain *pre condition* merupakan kondisi awal dimana algoritma akan dijalankan. *Pre condition* dinyatakan dengan mendefinisikan input dari sebuah algoritma.

2. ***Post condition***

   Post condition merupakan kondisi setelah suatu algoritma selesai dijalankan. *Post condition* dinyatakan dengan mendefinisikan hasil (output) dari suatu algoritma.

2.2.3. Kode Semu (*Pseudo Code*)

Kode semu atau *pseudo code* dalam sebuah algoritma sangat sangat diperlukan untuk memberikan gambaran eksekusi

algoritma secara lebih jelas dengan menyajikan kode program secar mendasar. Dengan kata lain kode semu atau *pseudo code* merupakan gambaran pokok dari algoritma yang disajikan dalam bentuk kode program. Dengan demikian algoritma akan lebih mudah dipahami programmer terutama dari sisi kode program yang akan dijalankan.

Pada kode semu harus dijelaskan kondisi awal (*pre condition*) dan kondisi akhir (*post condition*) serta input dan output dari algoritma. Adapun beberapa komponen pada kode semu (*pseudocode*) dari sebuah algoritma diantaranya sebagai berikut.

1. *Pre condition* dan *post condtion*
2. Input dan output algoritma
3. Pendefinisian variabel serta fungsi yang akan digunakan
4. Komentar pada beberapa bagian kode program untuk mempermudah pemahaman pada kode semu.

Berikut ini contoh kode semu (*pseudocode*) dari algoritma pencarian faktorial dari bilangan antara 1 dan 20.

Algoritma Faktorial

```
< pre condition >      input berupa bilangan bulat antara 1 sampai
                       20
< post contion >       output berupa nilai faktorial dari bilangan
                       input
Begin
        i = 1;         hasil_kali = 0;
        read n         /* baca input bilangan n
        if (n > 1 and n < 20) then
                hasil_kali = n;
                do
                       hasil_kali = hasil kali * (n-1);
                       n = n – 1;
                while(n > 0)
        endif
        return hasil_kali
End.
```

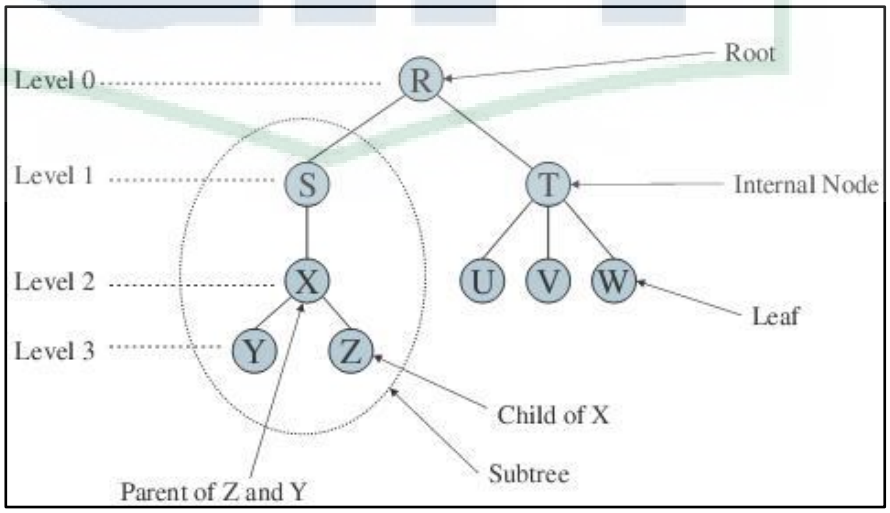
2.3. Struktur Data

2.3.1. Pohon (*Tree*)

Tree atau pohon, termasuk struktur non liniear, tree merupakan bagian dari graph (Sjukani, 2007). Dalam hal ini tree digunakan untuk mrepresentasikan graf.

Berikut ini beberapa komponen dari *tree*, yaitu:

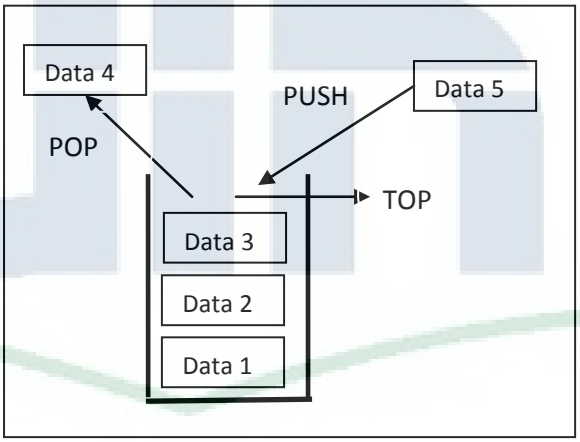
- *Root* atau Akar, merupakan node tertinggi atau node awal dari *tree*.
- Node, merupakan simpul atau elemen pembentuk *tree*
- Level, merupakan tingkatan pada setiap simpul atau node.
- *Parent*, node yang berada diatas node lain secara langsung
- *Child*, node yang berada dibawah node lain secara langsung
- *Sibling*, node yang berada pada level yang sama
- *Leaf*, sebuah node yang tidak memiliki child. Adapun node yang memiliki child disebut internal node.



Gambar 2.5 *Tree* dan Komponennya

2.3.2. Tumpukan (*Stack*)

Stack merupakan bentuk khusus dari suatu struktur data, dimana node yang ditambahkan dan diambil hanya pada kepalanya saja (Kristanto, 2003). Dengan demikian stack merupakan sebuah kumpulan data yang berbentuk suatu tumpukan dimana elemennya hanya dapat ditambahkan dan dikurangi hanya pada bagian atas. Cara kerja *stack* menggunakan prinsip *Last In First Out* (*LIFO*), dimana data yang terakhir masuk merupakan data yang akan keluar pertama. Terdapat dua operasi yang dapat digunakan pada *stack*, yaitu *push* dan *pop*. Operasi push digunakan untuk menambahkan elemen pada *stack*. Sedangkan operasi *pop* digunakan untuk mengambil elemen dari *stack*. Berikut ini contoh sebuah *stack*.



Gambar 2.6 Stack

2.3.3. Antrian (*Queue*)

Queue atau antrian adalah suatu kumpulan data dimana penambahan elemennya hanya bisa dilakukan pada satu ujung yang disebut dengan sisi belakang atau *rear* dan pengambilan elemennya

dilakukan pada ujung yang lain atau *front* (Sjukani, 2007). Cara kerja queue menggunakan prinsip *First In First Out* (*FIFO*) yaitu elemen yang pertama kali masuk maka akan menjadi elemen yang pertama keluar.

Ada dua operasi pada *queue* yaitu *enqueue* dan *dequeue*. Operasi *enqueue* digunakan untuk menambahkan elemen pada ujung *queue* (*rear*). Sedangkan operasi *dequeue* digunakan untuk mengambil elemen yang terdapat pada bagian depan *queue*. Berikut ini contoh *queue* beserta operasi *enqueue* dan *dequeue*.

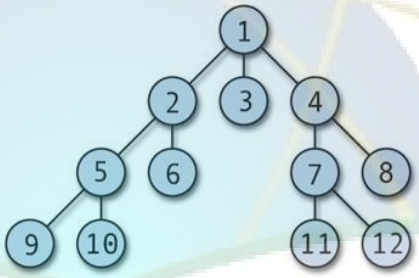
Gambar 2.7 *Queue*

2.4. Algoritma Traversal pada Graf

2.4.1. *Breadth First Search* (BFS)

Breadth First Search adalah penelusuran graph yang arah penelusurannya mendahulukan ke arah ‘lebar’ graph tersebut (Sjukani, 2007). Algoritma pencarian melebar (BFS) melakukan penelusuran setiap simpul pada graf dengan dimulai dari sebuah simpul awal (start), kemudian dilanjutkan dengan menelusuri simpul akar dari simpul awal, lalu dilanjutkan dengan menelusuri

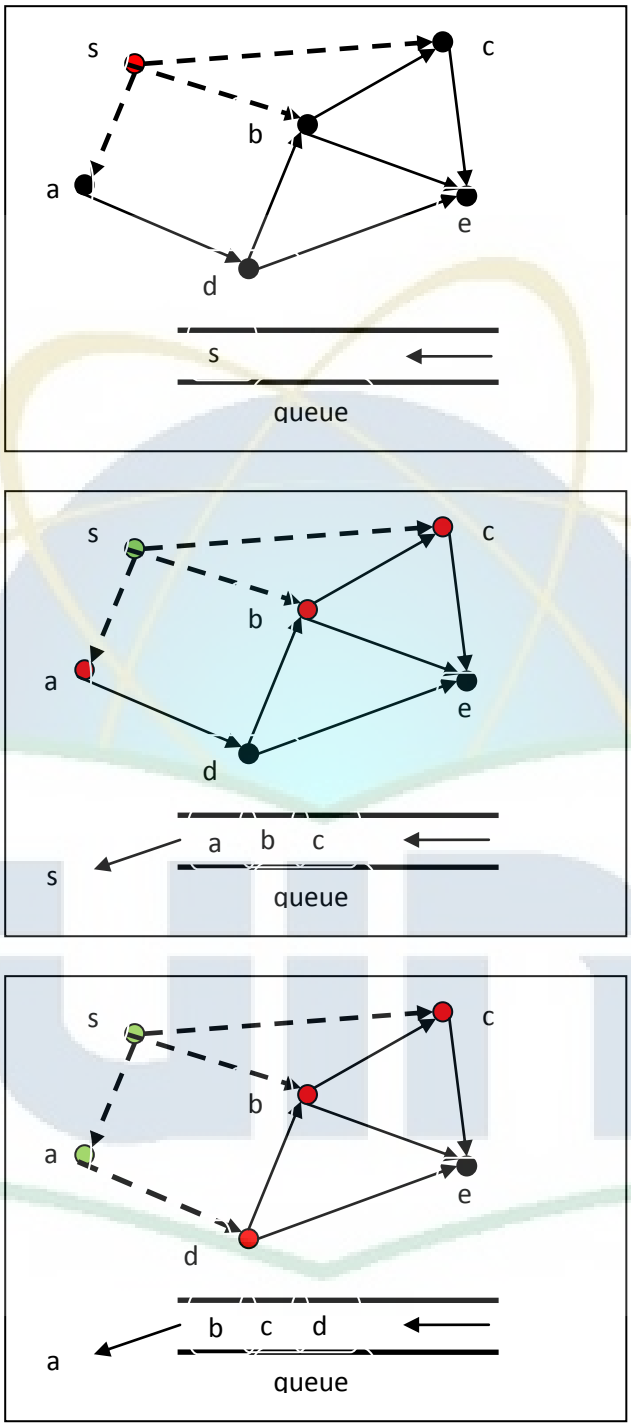
simpul-simpul yang bertetangga (satu level) dengan simpul akar tersebut. Setelah itu, kemudian akan menelusuri simpul pada level berikutnya, begitu seterusnya. Jadi dengan kata lain algoritma BFS menelusuri simpul-simpul berdasarkan urutan level. Simpul pada level lebih tinggi akan lebih dahulu ditelusuri daripada simpul yang berda pada level dibawahnya. Berikut ini gambaran umum dari urutan simpul-simpul yang ditelusuri algoritma BFS.

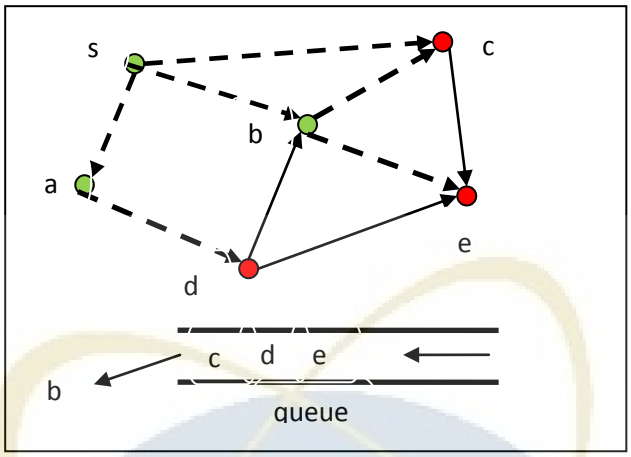


Gambar 2.8 Urutan Penelusuran Algoritma BFS

Cara kerja algoritma BFS menggunakan prinsip antrian (*queue*) yaitu *first in first out* (FIFO). Algoritma ini memasukan setiap simpul yang ditemukannya ke dalam *queue*. Selain itu pada algoritma BFS terdapat dua status yang diberikan kepada setiap simpul yaitu status dikunjungi dan status ditemukan.

Suatu simpul berada pada status ditemukan jika simpul tersebut berada pada antrian, sedangkan simpul yang telah keluar dari antrian berstatus dikunjungi. Adapun simpul yang belum pernah masuk antrian berarti berstatus belum dikunjungi dan belum ditemukan. Berikut ini gambaran cara kerja algoritma BFS.


s
c
b
a
e
d
s
queue
s
c
b
a
e
d
a b c
s
queue
s
c
b
a
e
d
b c d
a
queue



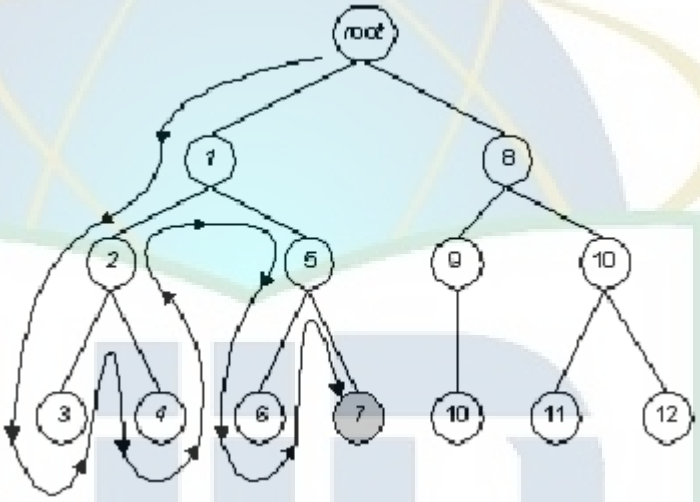
Gambar 2.9 Cara kerja algoritma BFS

Kompleksitas waktu dari algoritma dapat dinyatakan dengan notasi big O. Adapun kompleksitas dari algoritma BFS adalah O (jumlah simpul + jumlah sisi). Hal ini menunjukan bahwa kompleksitas waktu dari algoritma BFS sangat besar terutama jika digunakan untuk menyelesaikan masalah yang besar, akan tetapi algoritma BFS dipastikan mendapatkan solusi terbaik. Dengan demikian algoritma BFS harus digunakan pada pemecahan masalah atau kasus yang tepat. Beberapa kasus yang menggunakan algoritma BFS diantaranya seperti aplikasi pencarian rute terpendek, persoalan maximum flow, permainan game minesweeper dan lain-lain.

2.4.2. *Depth First Search* (DFS)

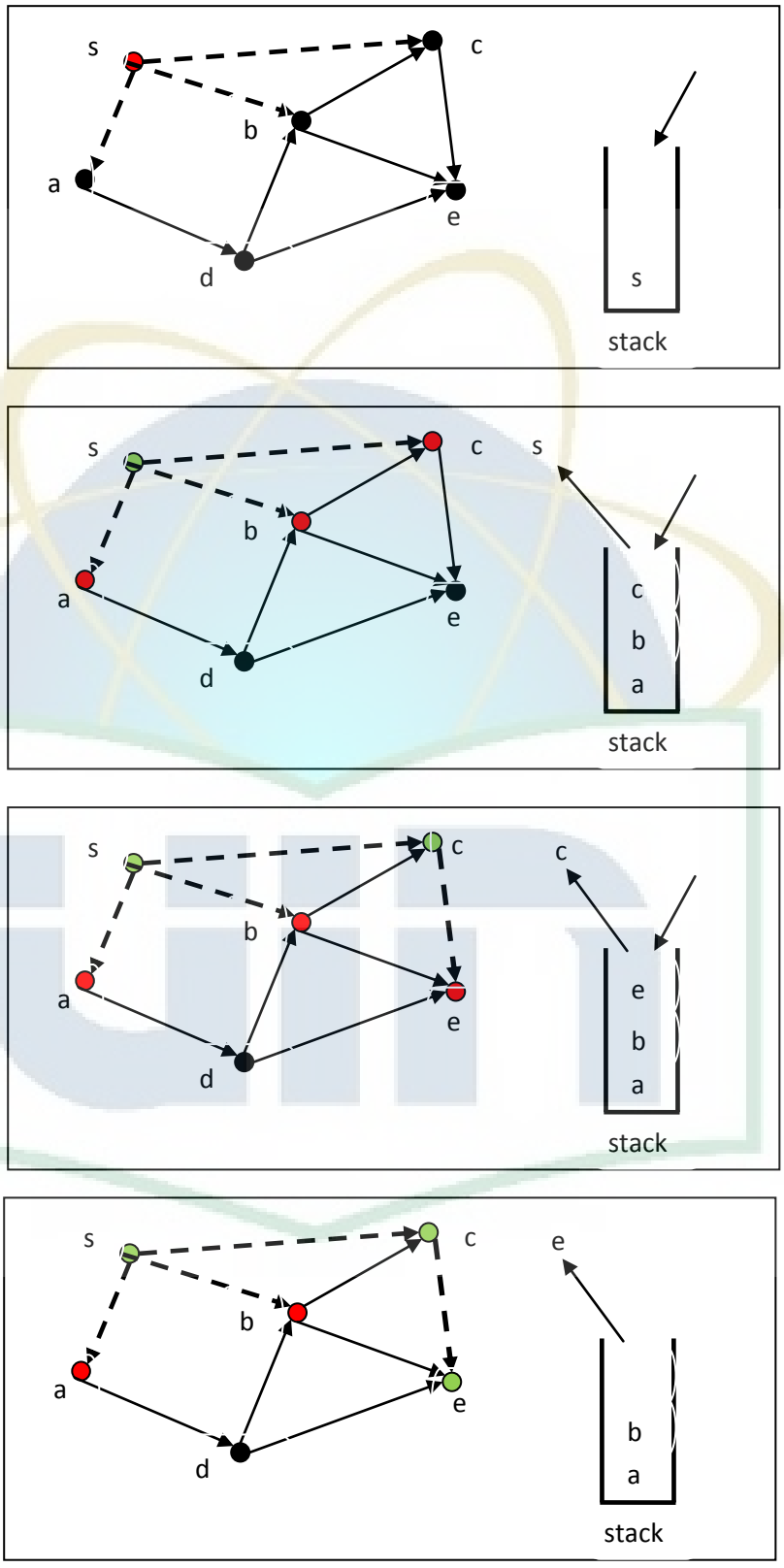
Depth First Search adalah penelusuran graph yang arah penelusurannya mendahulukan ke arah kedalaman graf tersebut (Sjukani, 2007). Algoritma pencarian mendalam (DFS) melakukan

pencarian solusi dengan cara menelusuri setiap akar pertama yang ditemukan. Penelusuran dimulai melalui simpul awal (start) kemudian berlanjut ke simpul pertama yang berada dibawahnya atau simpul pertama yang berada tepat satu level dari simpul awal. Kemudian begitu seterusnya sampai semua simpul berhasil ditelusuri. Berikut ini gambaran penelusuran yang dilakukan algoritma DFS pada sebuah grapf.



Gambar 2.10 Urutan Penelusuran Algoritma DFS

Cara kerja algoritma DFS menggunakan prinsip tumpukan (*stack*) yaitu *Last In First Out* (LIFO). Algoritma ini memasukan simpul yang telah ditelusuri kedalam sebuah stack. Dimana simpul yang terakhir masuk kedalam stack akan keluar pertama kali. Sama seperti algoritma BFS, algoritma DFS juga memiliki dua jenis status yang diberikan kepada setiap node yang terdapat pada graf. Berikut ini penjelasan cara kerja algoritma DFS



Gambar 2.11 Cara kerja algoritma DFS

Algoritma DFS memiliki kompleksitas ruang yang lebih rendah dari algoritma BFS, karena hanya menyimpan simpul-simpul pada suatu sub pohon. Akan tetapi kompleksitas waktu algoritma DFS sama dengan kompleksitas waktu algoritma BFS yang dinyatakan dengan notasi big O, yaitu O(jumlah simpul + jumlah sisi). Algoritma DFS dapat menemukan solusi terbaik karena dapat menjangkau semua simpul yang terdalam sekalipun. Adapun beberapa aplikasi yang menggunakan algoritma DFS diantarnya aplikasi pencarian komponen terhubung dalam graf serta aplikasi pengurutan topologi pada graf.

2.5. Algoritma *Shortest Path*

2.5.1. Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma yang paling sering digunakan dalam pencarian rute terpendek, sederhana penggunaannya dengan menggunakan simpul-simpul sederhana pada jaringan jalan yang tidak rumit (Chamero, 2006). Adapun nama algoritma Dikstra sendiri berasal dari penemunya yaitu Edsger Dikstra.

Dalam mencari solusi, algoritma Dijkstra menggunakan prinsip greedy, yaitu mencari solusi optimum pada setiap langkah yang dilalui, dengan tujuan untuk mendapatkan solusi optimum pada langkah selanjutnya yang akan mengarah pada solusi terbaik. Hal ini membuat kompleksitas waktu algoritma Dijkstra menjadi

cukup besar, yaitu sebesar O (V * Log (v + e)), dimana v dan e adalah simpul dan sisi pada graf yang digunakan.

Input dari algoritma Dijkstra berupa sebuah graf berbobot G(e , v), sedangkan outputnya berapa rute terpendek dari simpul awal (start) ke masing-masing simpul yang ada pada graf. Dengan demikian algoritma Dijkstra dapat menemukan solusi terbaik.

Cara kerja algoritma Dijkstra hampir sama dengan cara kerja algoritma BFS yaitu dengan menggunakan prinsip antrian (queue), akan tetapi antrian yang digunakan algoritma Dijkstra adalah antrian berprioritas (priority queue). Jadi hanya simpul yang memiliki prioritas tertinggi yang akan ditelusuri. Dalam menentukan simpul yang berprioritas, algoritma ini membandingkan setiap nilai (bobot) dari simpul yang berada pada satu level. Selanjutnya nilai (bobot) dari setiap simpul tersebut disimpan untuk dibandingkan dengan nilai yang akan ditemukan dari rute yang baru ditemukan kemudian, begitu seterusnya sampai ditemukan simpul yang di cari.

Berikut ini *pseudocode* algoritma Dijkstra dalam mencari rute terpendek pada sebuah graf.

Pseudocode algoritma Dijkstra (Edmons, 2008):

```
⟨pre−cond⟩: G is a weighted (directed or undirected) graph and s is one of
            its nodes.
⟨post−cond⟩: π specifies a shortest weighted path from s to each node of G
             and d specifies their lengths.
begin
    :
    notHandled = priority queue containing all nodes. Priorities given by d(v).
    loop
        ⟨loop−invariant⟩: See above.
        exit when notHandled = ∅
        let u be a node from notHandled with smallest d(u)
        for each v connected to u
            foundPathLength = d(u) + w(u,v)
            if d(v) > foundPathLength then
                d(v) = foundPathLength
                (update the notHandled priority queue)
                π(v) = u
            end if
        end for
        move u from notHandled to handled
    end loop
    return ⟨d, π⟩
end algorithm
```

Pada *pseudocode* Dijkstra tersebut terdapat tiga elemen utama yang menggambarkan kondisi status dari setiap simpul yang sedang ditelusuri. Adapun tiga kondisi tersebut yaitu:

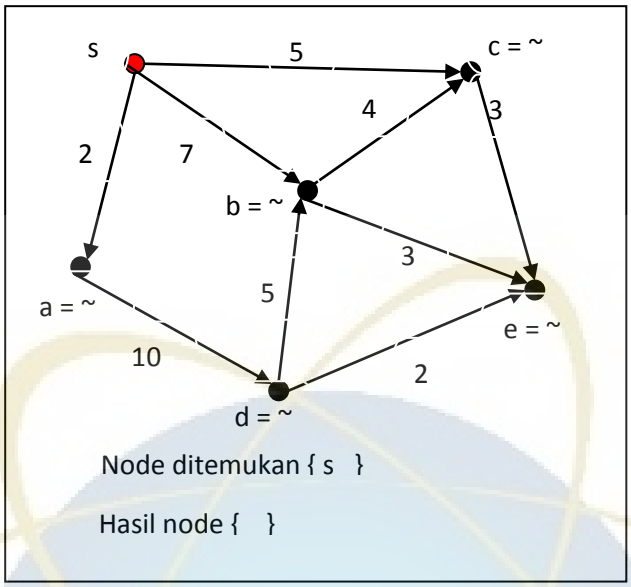
1. Kondisi node yang belum ditemukan dan belum dikunjungi
2. Kondisi node yang sudah ditemukan tetapi belum dikunjungi
3. Kondisi node yang telah ditemukan dan sudah dikunjungi

Dalam hal ini node yang dikunjungi merupakan node yang terpendek dari setiap tahap algoritma Dijkstra. Jadi jalur atau rute yang dibentuk oleh algoritma Dijkstra tersusun dari node yang telah ditemukan dan telah dikunjungi. Adapun langkah-langkah

dari algoritma Dijkstra sesuai *pseudocode* diatas yaitu:

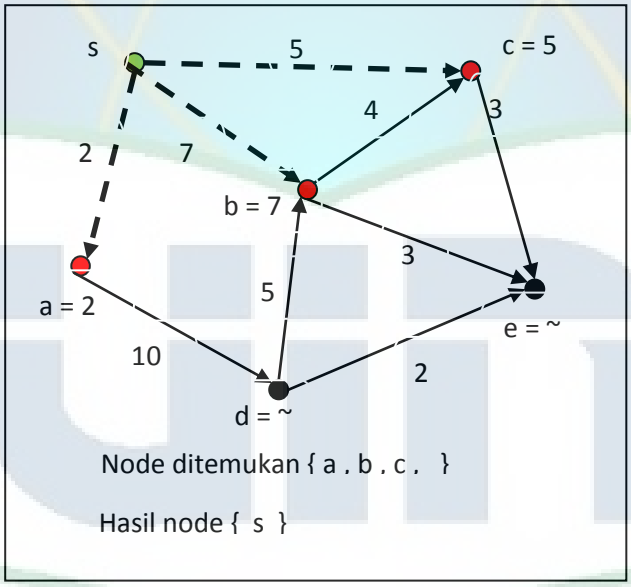
1. Langkah pertama yaitu menetapkan node awal sebagai status ditemukan (*found*) dan kemudian dikunjungi atau ditangani (*handled*)
2. Langkah kedua yaitu dilakukan pencarian terhadap setiap node yang dapat dicapai secara langsung dari node yang sedang dikunjungi
3. Langkah ketiga yaitu:
   - Apabila node yang didapatkan pada langkah kedua belum pernah ditemukan, maka rubah statusnya menjadi ditemukan
   - Apabila node yang didapatkan sudah pernah ditemukan maka lakukan *update* pada bobotnya, ambil bobot yang lebih kecil
4. Langkah keempat yaitu dilakukan pencarian terhadap node yang memiliki bobot paling kecil dari semua node yang berada pada status ditemukan kemudian mengunjunginya.
5. Lakukan *looping* secara berurutan pada langkah kedua, ketiga dan keempat sampai semua node ditemukan.

Berikut ini langkah-langkah algoritma Dijkstra dalam mencari rute terpendek pada sebuah graf.



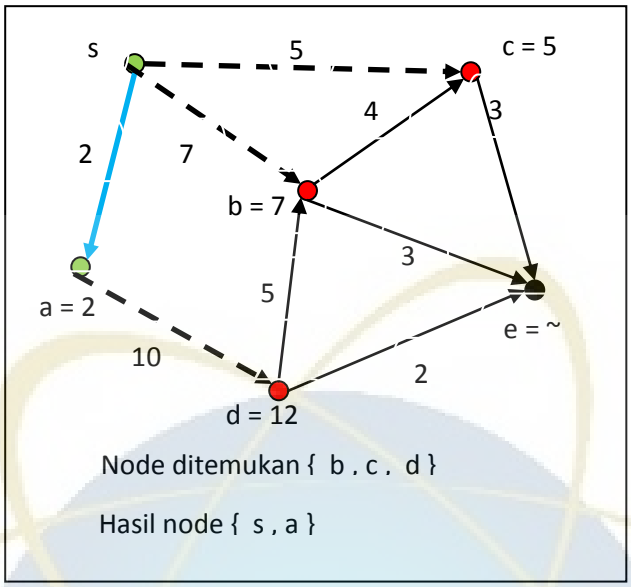
Node ditemukan { s }

Hasil node { }



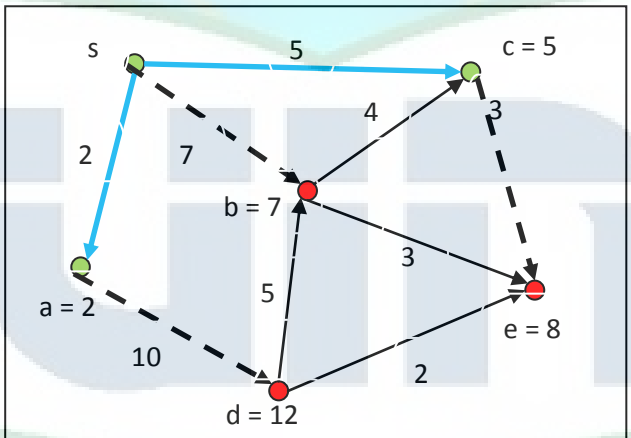
Node ditemukan { a , b , c , }

Hasil node { s }



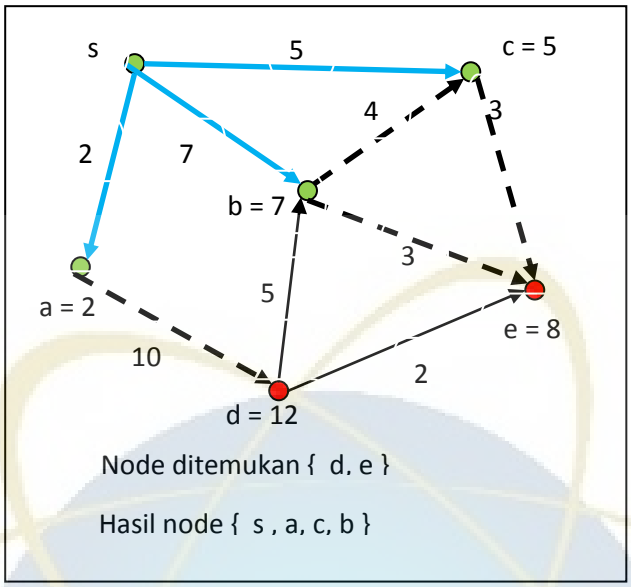
Node ditemukan { b , c , d }

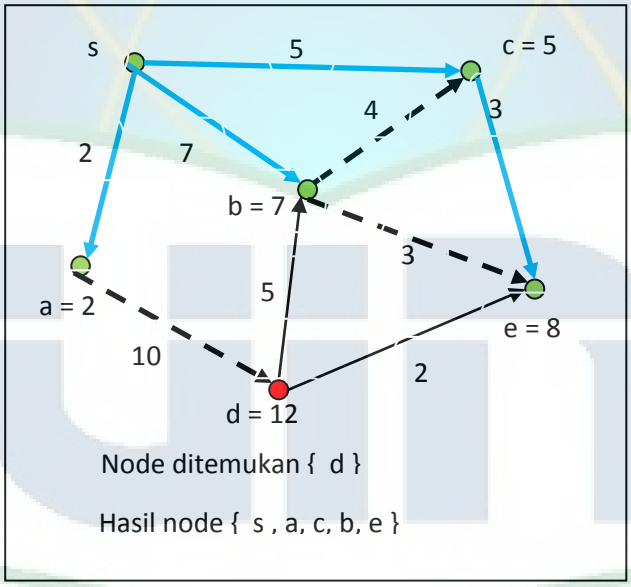
Hasil node { s , a }

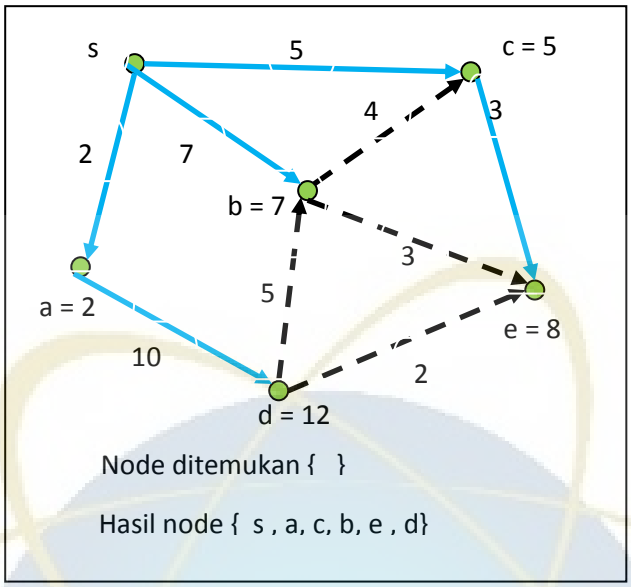


Node ditemukan { b , d, e }

Hasil node { s , a, c}


s
5
c = 5
4
3
2
7
b = 7
3
5
a = 2
e = 8
10
2
d = 12
Node ditemukan { d, e }
Hasil node { s , a, c, b }


s
5
c = 5
4
3
2
7
b = 7
3
5
a = 2
e = 8
10
2
d = 12
Node ditemukan { d }
Hasil node { s , a, c, b, e }



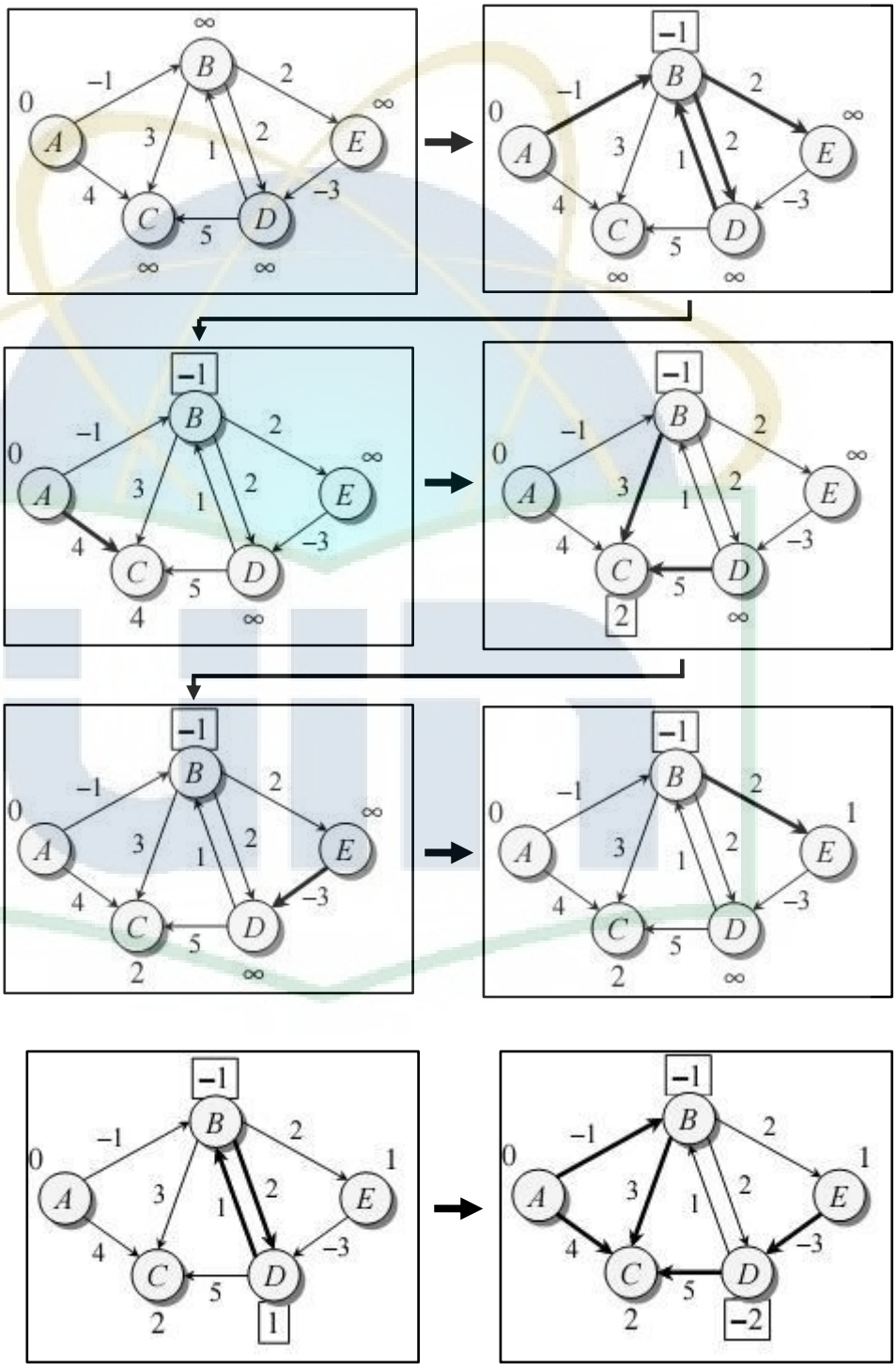
Gambar 2.12 Cara kerja algoritma dijsktra

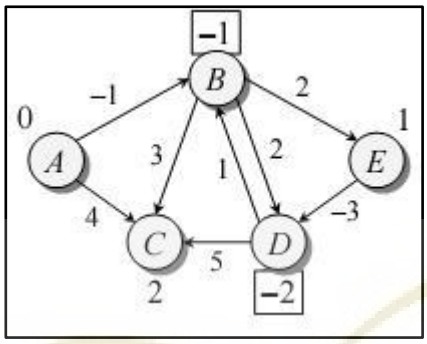
2.5.2. Algoritma Bellman Ford

Algoritma Bellman-Ford adalah algoritma untuk menyelesaikan permasalahan lintasan terpendek dengan sumber tunggal (Purwanto, 2008). Algoritma Bellman-Ford dikembangkan oleh Richard Bellman dan Lester Ford. Algoritma ini sangat mirip dengan algoritma Dijkstra, akan tetapi algoritma ini mampu menghitung path yang memilki bobot negatif.

Cara kerja algoritma ini dalam mencari jarak terpendek ke suatu node tujuan yaitu dengan meghitung setiap kemungkinan node yang mengarah ke node tujuan tersebut. Dalam hal ini berarti algoritma ini melakukan iterasi pada setiap langkahnya sebanyak n-1, dimana n adalah jumlah node yang terdapat pada graf. Dengan demikian kompleksitas algoritma ini cukup tinggi. Adapun

kompleksitas waktu dari algoritma bellman ford dapa dinyatakan dengan notasi big O yaitu sebesar O (V.E). Berikut ini pseudocode dan gambaran contoh langkah-langkah algoritma bellman ford.





Gambar 2.13 Cara kerja algoritma Bellman Ford

2.5.3. Algoritma Floyd Warshall

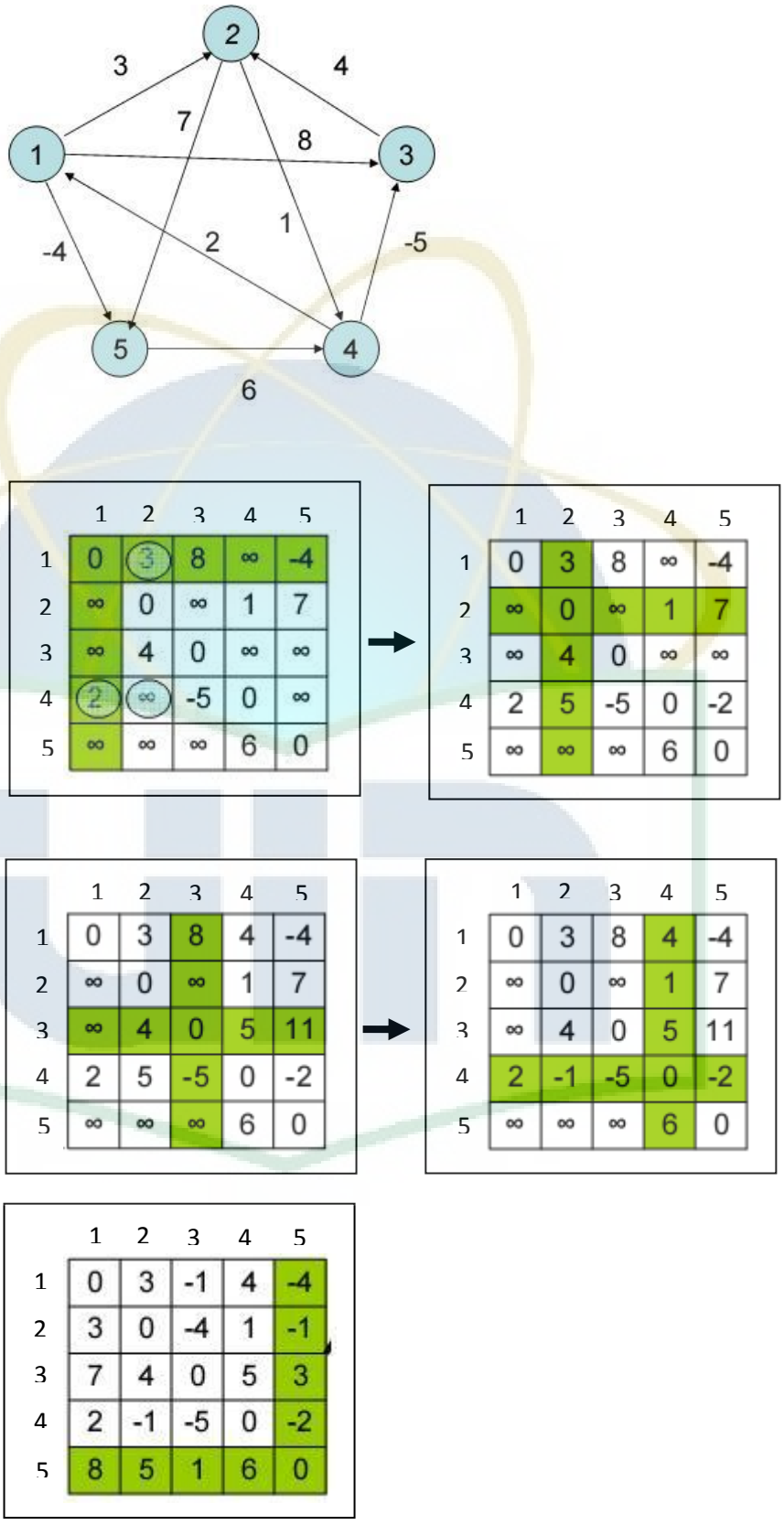
Algoritma Floyd Warshall tidak sekedar mencari lintasan terpendek antara dua buah simpul tertentu, tetapi langsung membuat tabel lintasan terpendek antar simpul (Sjukani, 2007). Algoritma Floyd Warshall merupakan salah satu varian pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Hal yang membedakan pencarian solusi menggunakan pemrograman dinamis dengan algoritma greedy adalah bahwa keputusan yang diambil pada tiap tahap pada algoritma greedy hanya berdasarkan pada informasi yang terbatas sehingga nilai optimum yang diperoleh pada saat itu tidak mencakup nilai optimum keseluruhan. Jadi pada algoritma greedy, kita tidak memikirkan konsekuensi yang terjadi seandainya kita memilih

suatu keputusan pada suatu tahap.

Dalam beberapa kasus algoritma greedy gagal memberikan solusi terbaik karena kelemahan yang dimilikinya tersebut. Disinilah peran pemrograman dinamis yang mencoba untuk memberikan solusi yang memiliki pemikiran terhadap konsekuensi yang ditimbulkan dari pengambilan keputusan pada suatu tahap. Pemrograman dinamis mampu mngurangi pengenumerasian keputusan yang tidak mengarah ke solusi. Prinsip yang dipegang oleh pemrograman dinamis adalah prinsip optimalitas, yaitu jika solusi total optimal, maka bagian solusi sampai suatu tahap (misalnya ke-i) juga optimal.

Algoritma Floyd Warshall merupakan salah satu jenis algoritma *all pair shortest path*, yaitu mencari rute terpendek untuk semua pasangan node yang ada pada sebuah graf. Input dari algoritma ini berupa graf berbobot dan berarah. Seperti algoritma Bellman Ford, algoritma ini juga dapat menghitung sisi yang berbobot negatif. Cara kerja algoritma floyd warshall dapat digambarkan dengan menggunakan matriks. Berikut ini contoh matrik yang digunakan untuk menggambarkan cara kerja algoritma Floyd warshall



Gambar 2.14 Cara kerja algoritma *Floyd Warshall*

2.5.4. Perbandingan Algoritma Dijkstra, Bellman Ford dan Floyd Warshall

Pada dasarnya setiap algoritma memiliki kelebihan dan kekurangan masing-masing sesuai dengan persoalan yang ditangani. Dengan demikian, jenis permasalahan yang ditangani akan sangat berpengaruh pada pemilihan algoritma yang digunakan. Berikut ini perbandingan algoritma Dijkstra, Bellman Ford dan Floyd Warshall.

Tabel 2.1 Perbandingan Beberapa Algoritma Shortest Path

Komponen Perbandingan	**Dijkstra**	**Bellman Ford**	**Flod Warshall**
Jenis Algoritma	Single Source Shortest	Single Source Sorthest Path	All Pair Shortest Path
Prinsip cara kerja yang digunakan	Priority Queue	Priority Queue	Matriks
Jenis pemrograman	Greedy	Dinamis	Dinamis
Kompleksitas Waktu	O (V * Log (v + e))	O (V.E)	$O(V^3)$
Nilai bobot simpul	Positif	Positif, negative	Positif, negatif

Adapun objek permasalahan pada penelitian ini yaitu sebuah graf berarah dan berbobot yang tidak memiliki sisi negatif dan tidak memiliki gelang atau *loop*. *Loop* dalam hal ini merupakan path yang memiliki titik awal dan titik akhir yang sama. Oleh karena itu penulis menggunakan algoritma Dijkstra dengan pertimbangan sebagai berikut.

1. Algoritma Dijkstra memiliki kompleksitas waktu yang lebih kecil, sehingga lebih efektif.
2. Graf yang digunakan sebagai input sistem merupakan graf yang tidak memiliki bobot negatif, sehingga algoritma Dijkstra lebih efektif karena tidak memerlukan pencarian terhadap sisi yang bernilai negatif.
3. Dalam kasus terburuk, algoritma Dijkstra tetap menemukan solusi terbaik dengan kompleksitas yang lebih sedikit daripada algoritma Bellman Ford dan Floyd Warshall.

2.6. Transportasi

2.6.1. Definisi Sistem Transportasi

Sistem adalah sekelompok unsur yang erat berhubungan satu dengan lainnya yang berfungsi bersama-sama untuk mencapai tujuan tertentu (Mulyadi, 2001).

Transportasi adalah kegiatan pemindahan barang atau penumpang dari suatu tempat ke tempat lain. Dengan demikian sistem transportasi adalah himpunan komponen-komponen yang

saling berkaitan dalam pemindahan barang atau penumpang tersebut. Adapun komponen transportasi tersebut diantaranya berupa kendaraan, jaringan jalan, serta komponen lainnya yang merupakan bagian dari sistem transportasi.

2.6.2. Pemodelan Jaringan Transportasi

Jaringan transportasi dapat dicerminkan dalam beberapa tingkat pengelompokan yang berbeda pada suatu pemodelan. Simpul dapat mencerminkan persimpangan, sedangkan ruas jalan dapat mencerminkan jalan yang menghubungkan antara dua persimpangan. Beberapa komponen ruas jalan yang perlu diketahui seperti panjang jalan, lebar jalan serta kecepatan dan waktu tempuh yang dapat dilalui pada jalan tersebut.

2.6.3. Pemilihan Rute Terpendek Pada Jaringan Jalan

Lintasan terpendek dapat dicari dengan memodelkan jaringan jalan ke sebuah graf. Graf yang digunakan adalah graf berbobot, yaitu graf yang setiap sisinya memiliki nilai atau bobot. Dalam hal ini bobot tersebut berupa jarak dan waktu tempuh dari suatu persimpangan ke persimpangan lain.

Ada beberapa macam persoalan lintasan terpendek yaitu:

a. Lintasan terpendek antara dua simpul tertentu (a pair shortest path)
b. Lintasan terpendek antara semua pasangan simpul (all pair shortest path)

c. Lintasan terpendek dari sebuah simpul tertentu ke semua simpul (single source shortest path)

d. Lintasan terpendek antara dua simpul yang melalui beberapa simpul tertentu (intermediate shortest path)

2.7. Metodologi Penelitian

2.7.1. Metode Pengumpulan Data

2.7.1.1. Observasi

Observasi adalah penginderaan secara khusus dengan penuh perhatian terhadap suatu obyek (Ali, 2007).

2.7.1.2. Studi Pustaka

Studi pustaka adalah suatu karangan ilmiah yang berisi pendapat berbagai pakar mengenai suatu masalah, yang kemudian ditelaah dan dibandingkan dan ditarik kesimpulannya (A.G Haryanto, 2000).

2.7.1.3. Wawancara

Wawancara adalah cara-cara atau kemampuan melakukan Tanya jawab dengan pihak lain untuk memperoleh keterangan atau pendapat tentang suatu hal yang diperlukan (Barata, 2003).

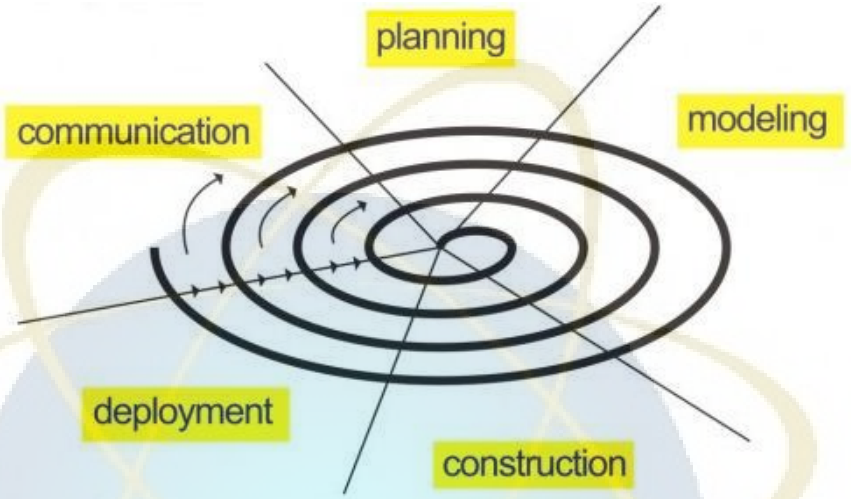
2.7.1.4. Kuisioner

Kuisioner dengan skala Likert adalah instrument yang umumnya digunakan untuk meminta responden agar memberikan respon terhadap beberapa *statement* dengan menunjukan apakah dia setuju, tidak setuju atau tidak tahu pada tiap-tiap *statement* (Sumanto, 1995).

2.7.2. Metode Pengembangan Sistem

Metode pengembangan sistem yang penulis gunakan dalam penelitian ini yaitu metode spiral model. Penulis menggunakan metode ini dikarenakan penelitian ini membutuhkan tahap tahap pengmbangan sistem yang selalu berlanjut. Dengan kata lain pengembangan tidak bersifat statis tetapi dinamis dengan melakukan pengembangan prototype yang dilakukan secara berulang untuk mencapai sisten yang sesungguhnya. Adapun metode ini pertama kali diusulkan oleh Barry Boehm. Spiral model merupakan metode pengembangan sistem evolusioner yang menyatukan sifat iterasi dari prototyping dengan kontrol dan aspek sistematis dari model sekuensial atau waterfall model (Pressman, 2010).

Metode pengembangan spiral model, memiliki beberapa tahapan yaitu communication, modeling, coding dan deployment.



Gambar 2.15 Spiral Model

2.7.2.1. *Communication*

Tahap ini merupakan tahap komunikasi atau interaksi antara pengembang dengan customer. Tahap ini membahas kebutuhan dari costumer, dengan demikian maka akan diperoleh definisi dan spesifikasi sistem yang dibutuhkan

2.7.2.2. *Planning*

Pada tahap *planning* terdapat tiga bagian yaitu:

a. *Estimation*

Estimation merupakan proses penetapan perkiraan semua komponen proyek, seperti perkiraan waktu, biaya dan sumberdaya yang dibutukan oleh sistem.

b. *Scheduling*

Scheduling merupakan tahap penjadwalan dari seluruh kegiatan pengembangan sistem.

c. *Risk Analysis*

Risk analysis merupakan proses analisa terhadap kemungkinan resiko yang terjadi pada proses pengembangan sistem dilakukan.

2.7.2.3. *Modeling*

Tahap *modeling* merupakan tahap dilakukannya pemodelan sistem. Pada sistem yang beroientasi objek, pemodelan biasanya menggunakan pemodelan *Unified Modeling Language* (UML).

2.7.2.4. *Construction*

a. Coding

Tahap *coding* meliputi serangkaian pengkodean yang dilakukan untuk membuat *prototype* dan bagian-bagian program lainnya yang pada akhirnya akan diintegrasikan menjadi satu kesatuan yang membentuk sebuah sistem yang utuh.

b. Testing

Tahap *testing* merupakan serangkaian pengujian terhadap sistem secara menyeluruh. Tahap ini biasanya menggunakan metode pengujian *black box testing*

ataupun *white box testing. Black box testing* adalah uji coba yang mefokuskan pada keperluan fungsional dari *software*. Sementara itu *white box testing* merupakan cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak.

2.7.2.5. *Deployment*

Tahap *deployment* yaitu tahap implementasi sistem. Tahap ini meliputi kegiatan saat komponen-komponen dalam sistem di-*deploy.* Selanjutnya sistem dapat *running* dan bisa mulai digunakan.

2.8. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) tidak mendefinisikan proses standar tetapi dimaksudkan untuk menjadi berguna dengan proses pengembangan berulang. Hal ini dimaksudkan untuk mendukung proses pengembangan dengan *object oriented*. UML menyimpan informasi tentang struktur statis dan perilaku dinamis suatu sistem. Perilaku dinamis mendefinisikan sejarah obyek dari waktu ke waktu dan komunikasi antara objek-objek untuk mencapai tujuan (Rumbaugh, *et al*., 2006: 3).

Beberapa diagram yang terdapat pada pemodelan UML diantaranya *use case diagram, class diagram, activity diagram, sequence diagram* dan *deployment diagram.*

2.8.1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Dalam pembuatan *Use Case Diagram*, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem.

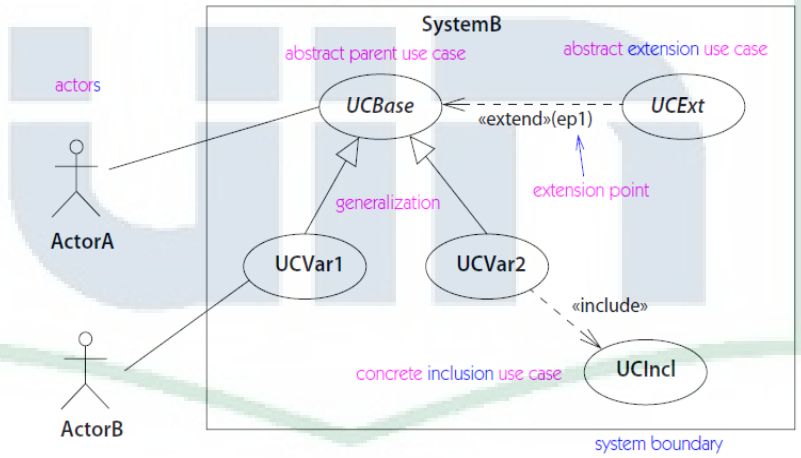
Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain (Dharwiyanti & Wahono, 2003: 4).

Berikut ini beberapa -simbol yang digunakan pada *use case diagram* diantaranya terdapat pad tabel berikut.

Tabel 2.2 Simbol Pada *Use Case Diagram*

Simbol	Nama Simbol	Kegunaan
	Aktor	Sebagai subjek yang berinteraksi atau menggunakan system
	Use Case	Sebagai kegiatan yang dapat dilakukan oleh pengguna pada system

	Asosiasi	Sebagai pengubung antara aktor dan use case yang dilakukan
<<Include>>	Include	Sebagai penghubung antara use case yang membutuhkan use case yang lain
Sistem	Sistem	Sebagai cakupan wilayah system



Gambar 2.16 Contoh *Use Case Diagram*
(Sumber: Rumbaugh, *et al*., 2006: 695)

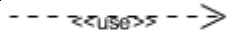
2.8.2. *Class Diagram*

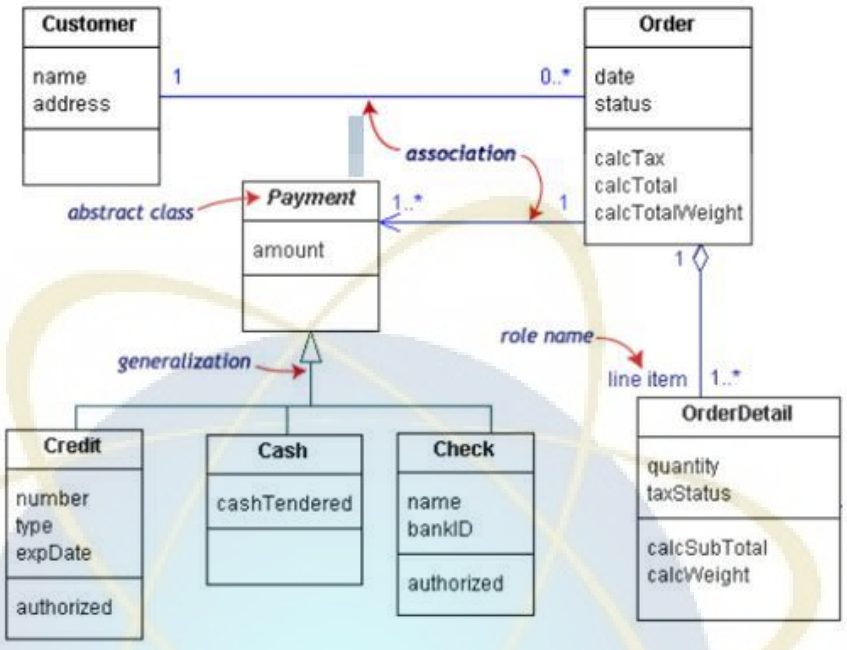
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package*, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Berikut ini beberapa symbol yang digunakan pada *class diagram*.

Tabel 2.3 Simbol Pada *Class Diagram*

Simbol	Nama Simbol	Kegunaan
	*Class*	Sebagai kelas yang digunakan pada system
	*Generalization*	Menunjukan hubugan *inheritance* antar kelas
	Usage	Menunjukan hubungan penggunaan suatu kelas dengan kelas yang lain



Gambar 2.17 Contoh *Class Diagram*
(Sumber: Dharwiyanti & Wahono, 2003: 6)

2.8.3. *Activity Diagram*

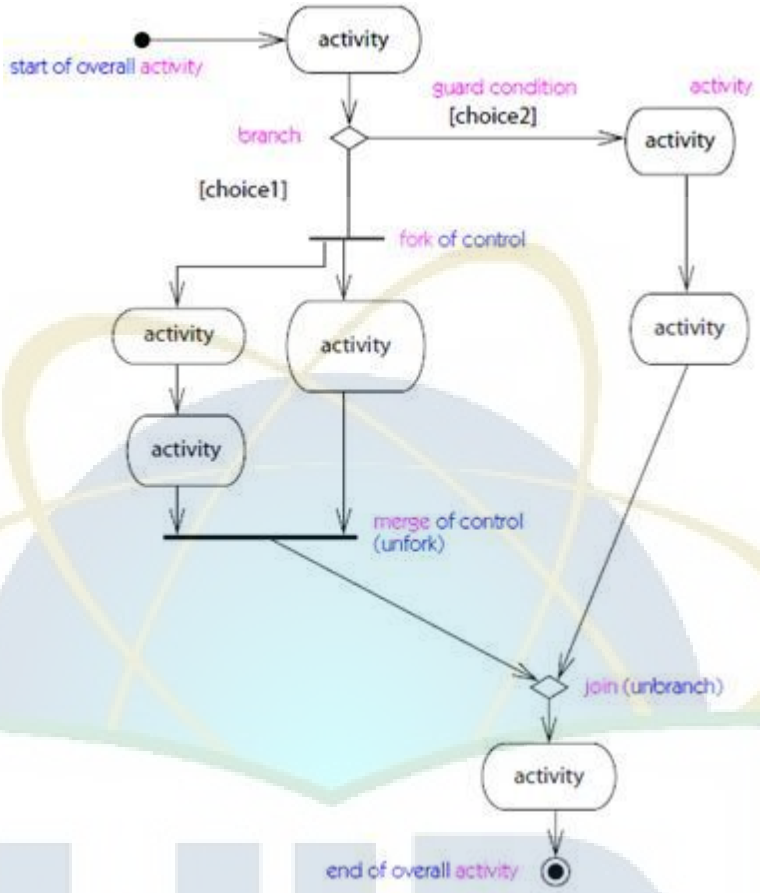
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas (Dharwiyanti & Wahono, 2003: 7).

Berikut ini beberapa simbol yang digunakan pada *activity diagram.*

Tabel 2.4 Simbol Pada *Activity Diagram*

Simbol	Nama Simbol	Kegunaan
	*Inisial node*	Awal aktifitas
	*Final node*	Akhir aktifitas
	*Action*	Sebagai aktifitas yang dilakukan oleh system
	*Control Flow*	Sebagai penghubung urutan aktifitas
	*Decision*	Merupakan aktifitas pengecekan kondisi
	*Exception handler*	Menunjukan kondisi pengecualian apabila suatu *action* tidak dapat dilakukan



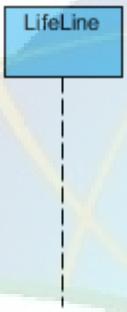
Gambar 2.18 Contoh *Activity Diagram*
(Sumber: Rumbaugh, *et al*., 2006: 699)

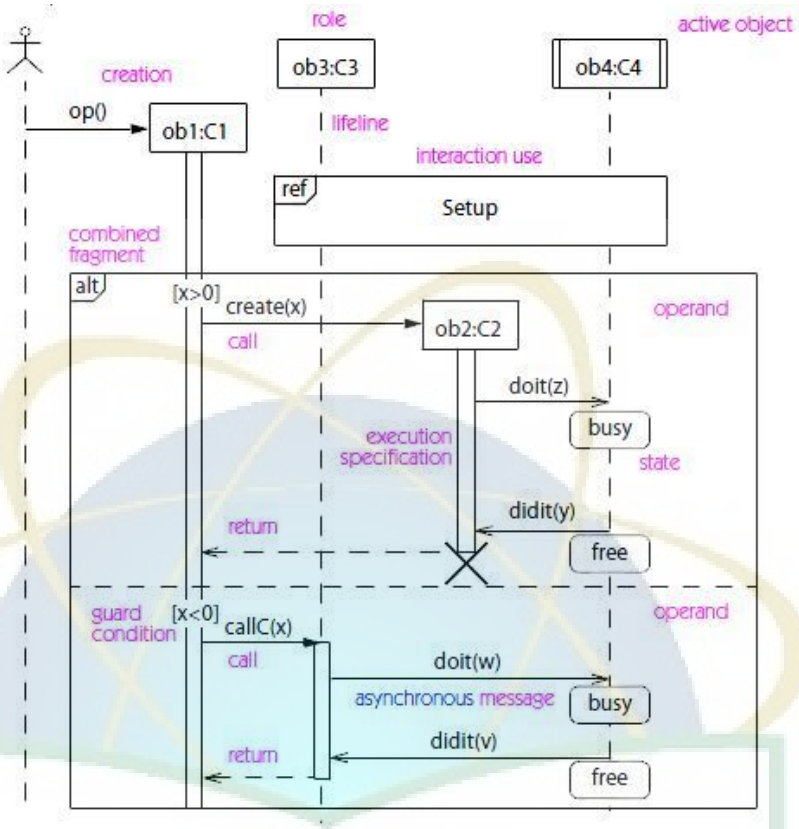
2.8.4. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai tanggapan dari sebuah *event* untuk menghasilkan *output* tertentu.

Berikut ini beberapa simbol yang digunakan pada *sequence diagram.*

Tabel 2.5 Tabel Simbol Pada *Sequence Diagram*

Simbol	Nama Simbol	Kegunaan
	*Actor*	Sebagai subjek yang menggunakan system
	*Life line*	Bagian dari sistem yang melakukan aktifitas pemrosesan data
	Message	Alur data yang diproses oleh system



Gambar 2.19 Contoh *Sequence Diagram*
(Sumber: Rumbaugh, *et al*., 2006: 700)

2.8.5. *Deployment Diagram*

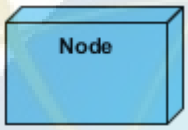
Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal.

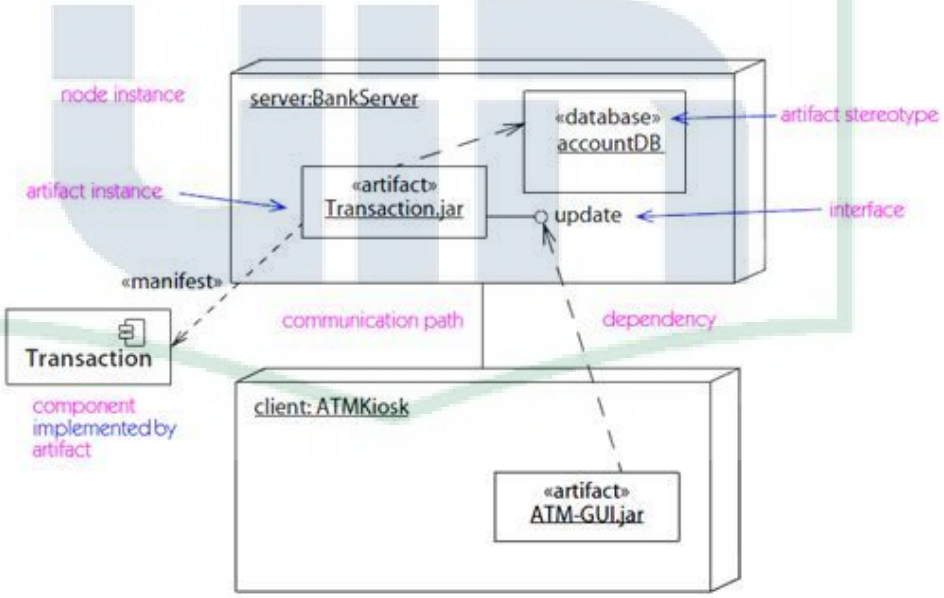
Sebuah *node* adalah *server*, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP)

dan *requirement* dapat juga didefinisikan dalam diagram ini (Dharwiyanti & Wahono, 2003: 10).

Berikut ini beberapa symbol yang digunakan pada *deployment diagram.*

Tabel 2.6 Simbol Pada *Deployment Diagram*

Simbol	Nama Simbol	Kegunaan
	*Node*	Komponen atau perangkat yang digunakan pada proses deployment system
	Asosiation	Sebagai penghubung aktifitas yang dilakukan antar komponen



Gambar 2.20 Contoh *Deployment Diagram*
(Sumber: Rumbaugh, *et al*., 2006: 701)

2.9. PHP

2.9.1. Definisi PHP

PHP singkatan dari PHP *Hypertext Prepocessor*, sementara itu kata PHP sendiri merupakan singkatan dari *Personal Home Page* (Kasiman, 2006). PHP digunakan sebagai bahasa script *server-side* dalam pembuatan aplikasi Web yang disisipkan pada dokumen HTML. Bahasa *script server-side* adalah bahasa pemrograman web yang bekerja pada server. Jadi fungsi-fungsi atau prosedur yang ada pada bahasa tersebut hanya dapat dijalankan di server.

2.9.2. Penggunaan PHP

Penggunaan PHP memungkinkan aplikasi dapat dibuat dinamis sehingga dapat mempermudah pengelolaan (*maintenance*) dari aplikasi tersebut.

PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis. PHP difokuskan pada pembuatan *script server-side*. PHP bisa melakukan semua pekerjaan yang dapat dilakukan oleh CGI(), seperti mengumpulkan data dari form, menghasilkan isi halaman aplikasi Web yang dinamis dan kemampuan mengirim serta menerima *cookies, session* dan informasi lainnya.

PHP dapat digunakan pada semua sistem operasi, antara lain Linux, Unix (termasuk variannya HP-UX, Solaris dan

OpenBSD), Microsoft Windows, Mac OS X, RISC OS. PHP juga mendukung banyak web server, seperti Apache, *Microsoft Internet Information Serve*r (MIIS), *Personal Web Server* (PWS) dan lain-lain. PHP tidak terbatas pada hasil keluaran HTML (*HyperText Markup Language*). PHP juga memiliki kemampuan untuk mengolah keluaran gambar, file PDF dan *movies* Flash. PHP juga dapat menghasilkan teks seperti XHTML dan XML lainnya. Selain itu salah satu fitur utama yang dimiliki PHP adalah kemampuannya yang dapat mendukung banyak database, seperti dBase, MySQL, ODBC, Oracle, PostgreSQL dan FrontBase.

2.10. Database MySQL

2.10.1. Definisi Database MySQL

Database atau sering disebut basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Adapun perangkat lunak yang digunakan untuk mengelola dan memanggil basis data tersebut dinamakan dengan sistem manajemen basis data (*database management system*, DBMS). Sementara itu MySQL merupakan salah satu perangkat lunak DBMS. MySQL adalah sebuah server database SQL *multiuser* dan *multi-thread*. SQL sendiri adalah bahasa standar yang digunakan untuk mengakses server database. SQL merupakan singkatan dari Structured Query

Language. Semenjak tahun 70-an bahasa ini telah dikembangkan oleh IBM. Dengan menggunakan SQL, proses akses database menjadi lebih mudah dipahami.

2.10.2. Penggunaan Database MySQL

MySQL merupakan perangkat lunak yang bersifat *open source* sehingga gratis digunakan oleh siapa saja. Software MySQL dapat di download www.mysql.com MySQL umumnya digunakan bersamaan dengan PHP untuk membuat aplikasi web yang dinamis dan powerful. Saat ini SQL merupakan salah satu bahasa database yang paling popular di dunia. Adapun berikut ini beberapa keunggulan dari MySQL:

a. MySQL merupakan program yang bersifat *multi-thread*, sehingga dapat digunakan pada server yang memiliki lebih dari satu CPU.
b. MySQL didukung program-program umum seperti C, C++, Java, PHP dan lain-lain.
c. MySQL bekerja pada berbagai platform system operasi seperti windows, linux, mac OS dan sebagainya.
d. MySQL memiliki jenis kolom yang cukup banyak sehingga memudahkan konfigurasi sistem database.
e. MySQL memiliki sistem keamanan yang cukup baik, yaitu dengan adanya verifikasi host.

2.11. Pemrograman Model View Control (MVC)

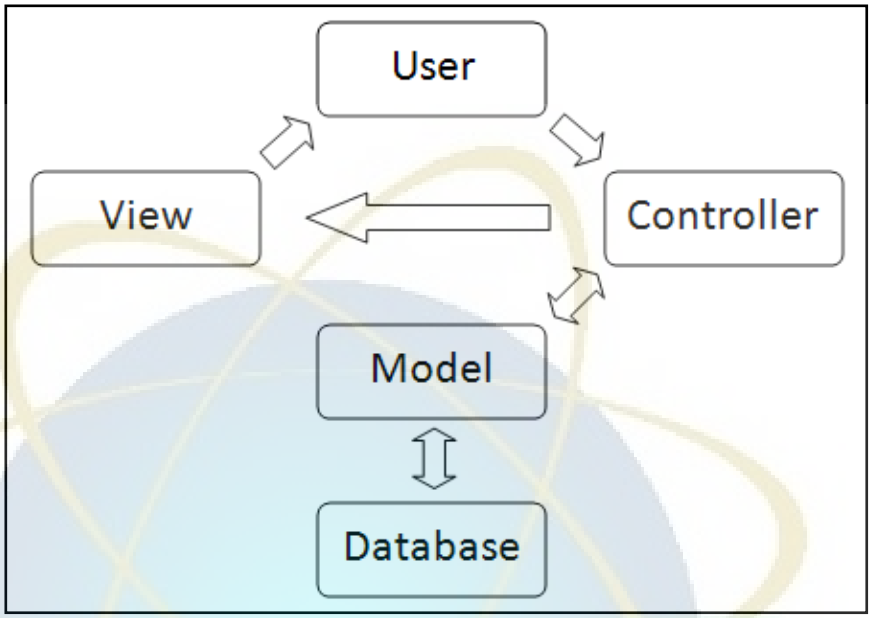
2.11.1. Definisi Pemrograman Model View Control (MVC)

MVC adalah sebuah pola pemrograman yang bertujuan memisahkan logika bisnis, logika data, dan logika tampilan (*interface*), atau secara sederhana memisahkan antara proses, data, dan tampilan. MVC mengatur arsitektur sebuah aplikasi. Umumnya aplikasi yang dibangun dengan konsep MVC adalah aplikasi yang cukup besar, karena salah satu keuntungan dari MVC itu adalah kemudahan *maintenance,* dan pengembangan aplikasi tersebut (Wardana, 2010: 52).

Konsep MVC biasanya digunakan pada framework. *Framework* dapat diartikan sebagai koleksi atau kumpulan potongan-potongan program yang disusun atau diorganisasikan sedemikian rupa, sehingga dapat digunakan untuk membantu membuat aplikasi utuh tanpa harus membuat semua kodenya dari awal (Basuki, 2010: 3). Setiap bahasa pemrograman web memiliki berbagai jenis framework masing-masing. Contoh framework untuk ASP diantaranya ada ASP.NET dan lain-lain. Sementara framework untuk PHP diantaranya CodeIgniter (CI), cake PHP serta Zend framework.

Dalam konsep MVC dikenal tiga komponen pembangun, di mana ada interaksi yang terjadi di antara mereka yaitu model, view

dan controller.



Gambar 2.21 Konsep MVC

2.11.2. Model

Model adalah sebuah layer pada MVC yang merepresentasikan data yang digunakan oleh aplikasi sesuai dengan proses bisnis yang terjadi pada data tersebut, dengan memilahnya menjadi beberapa bagian terpisah kembali, seperti penampungan data, persistence dan proses manipulasi. Model melakukan penempatan detail data dan operasinya pada area yang ditentukan (Model) sehingga tidak tersebar pada keseluruhan lingkup aplikasi. Selain itu model dapat digunakan pada lebih dari satu aplikasi, hal ini karena model terpisah dari elemen *logic* dan *interface* program.

2.11.3. View

View adalah sebuah layer pada MVC yang mengandung keseluruhan detai dari implementasi user interface dengan melibatkan komponen grafis yang menyediakan representasi proses internal aplikasi dan meuntun alur interaksi user terhadap aplikasi

2.11.4. Controller

Controller adalah sebuah layer pada MVC yang menyediakan detai alur program dan transisi layer, dan bertanggung jawab akan penampungan event yang dibuat oleh user dari view dan melakukan update terhadap komponen model menggunakan data dari user.

2.12. Frame Work

2.12.1. Definisi *Frame Work*

Framework dapat diartikan sebagai koleksi atau kumpulan potongan-potongan program yang disusun atau diorganisasikan sedemikian rupa, sehingga dapat digunakan untuk membantu membuat aplikasi utuh tanpa harus membuat semua kodenya dari awal (Basuki, 2010: 3)

2.12.2. *Frame Work* Code Igniter

Frame work Code Igniter atau biasa disebut CI merupakan salah satu framework PHP yang menggunakan konsep *Model View Control* (MVC). Code Igniter pada awalnya ditulis oleh Rick Ellis,

pendiri dan CEO EllisLab.com, perusahaan yang mengembangkan Code Igniter. Saat ini, Code Igniter dikembangkan oleh komunitas dan disebarkan ke seluruh dunia dengan lisensi bebas

Struktur pemrograman pada *frame work* CI menggunakan kelas-kelas. Sebuah kelas memiliki tiga bagian yaitu kelas pada *controller*, kelas pada *model* dan kelas pada *view*. Di dalam *frame work* CI, terdapat dua kelas utama yaitu kelas *controller* dan kelas *model*. Kelas utama tersebut merupakan kelas yang berperan sebagai penghubung untuk dapat mengakses fungsi-fungsi dan komponen-komponen yang terdapat pada *frame work* CI. Adapun kelas-kelas yang dibuat pada bagian *controller* harus didefinisikan sebagai *child* dari kelas controller, begitu juga kelas-kelas yang dibuat pada *model* harus didefinisikan sebagai *child* dari kelas *model.*

BAB III

METODOLOGI PENELITIAN

3.1. Metode Pengumpulan Data

3.1.1. Observasi

Observasi penulis lakukan ke kantor Dinas Perhubungan Provinsi DKI Jakarta untuk mendapatkan data mengenai waktu tempuh dan jarak antara dua persimpangan pada jalan raya di Jakarta khususnya antara wilayah Blok M dan Kota. Penulis juga melakukan observasi secara langsung ke beberapa persimpangan untuk melengkapi data yang diperoleh dari pihak Dinas Perhubungan Provinsi DKI Jakarta. Adapun data yang penulis dapatkan dari kedua hasil observasi tersebut ada pada bagian lampiran 2.

3.1.2. Studi Pustaka

Pada penelitian ini penulis menggunakan metode studi pustaka dengan tujuan untuk mendapatkan referensi yang berbobot dari masalah yang diteliti. Adapun bahan pustaka yang digunakan penulis yaitu berupa buku-buku referensi, literatur dan jurnal ilmiah.

3.1.3. Wawancara

Wawancara yang penulis lakukan yaitu terhadap Bapak Agung, pegawai Dinas Perhubungan DKI Jakarta. Wawancara dilakukan ditempat unit Manajemen dan Rekayasa Lalu Lintas (MRLL) Dinas Perhubungan DKI Jakarta. Isi wawancara terdapat pada lampiran 3. Tujuan wawancara yaitu untuk mendapatkan data mengenai faktor-faktor apa saja yang dapat mempengaruhi waktu tempuh dari suatu persimpangan ke persimpangan yang lainnya.

3.1.4. Kuisioner

Penulis melakukan dua kali kuisioner, pada setiap kuisioner penulis mengambil sampel sebanyak dua puluh responden. Kuisioner pertama dilakukan dengan tujuan untuk mendapatkan data sikap responden terhadap masalah kemacetan di Jakarta serta perlu tidaknya dibuat sebuah sistem untuk menangani kemacetan tersebut. Sedangkan kuisioner kedua dilakukan dengan tujuan untuk mendapatkan data sikap responden terhadap aplikasi yang telah dibangun. Adapun untuk rincian pertanyaan dan hasil kuisioner terdapat pada bagian lampiran 4.

3.2. Metode Pengembangan Sistem

3.2.1. *Spiral Model*

3.2.1.1. *Communication*

Dalam hal ini penulis berkomunikasi secara langsung dengan pihak Dishub Provinsi DKI Jakarta yang

diwakili oleh Bapak Agung pada bagian Manajemen dan Rekayasa Lalu Lintas. Komunikasi yang penulis lakukan diantaranya yaitu mengenai data waktu tempuh antar persimpangan, faktor-faktor waktu tempuh perjalanan serta pembangunan rancangan dan pengembangan aplikasi.

3.2.1.2. *Planning*

a. *Estimation*

Pada pembuatan aplikasi ini, penulis menetapkan estimasi waktu selama 4 (empat) bulan. Dimulai dari bulan Juni dan berakhir pada bulan September tahun 2010. Sedangkan estimasi biaya sebesar Rp. 200.000. Biaya tersebut digunakan untuk biaya transportasi selama penelitian. Adapun untuk biaya software, penulis menggunakan program yang bersifat gratis.

b. *Scheduling*

Scheduling atau penjadwalan pada pembuatan aplikasi ini yaitu:

Tabel 3.1 Penjadwalan Pengembangan Sistem

Kegiatan	Waktu
Komunikasi dengan pihak Dishub serta melakukan perencanaan system	Juni 2010

Perancangan model dan desain system	Juli 2010
Pembuata prototype kode program dan penyatuan semua modul sistem	Agustus - September 2010
Proses deployment	September 2010
Komunikasi dengan pihak Dishub mengenai system yang telah dibuat,	September 2010

c. Risk Analysis

Risk analysis yang penulis lakukan terutama terhadap kemungkinan resiko yang akan terjadi. Adapun beberapa resiko pada pengembangan aplikasi yang penulis bangun diantaranya yaitu adanya ketidaksesuaian suatu data dengan data yang lain, serta adanya data yang berulang. Untuk menangani kendala tersebut penulis melakukan komunikasi langsung dengan pihak Dishub yang diwakili oleh bapak Agung.

3.2.1.3. *Modeling*

Dalam hal ini penulis melakukan pemodelan sistem yang mencakup pemodelan database dan pemodelan interface. Pada pemodelan sistem penulis menggunakan *Unified Modelling Language* (UML) sementara itu pada pemodelan database penulis menggunakan bagan relasi

antar tabel sedangkan pada pemodelan *interface* penulis menggunakan *layout* halaman user, halaman login admin dan halaman admin. Dalam pemodelan interface penulis membuat perancangan *interface* berbasis web, hal ini selain karena sistem ini berbasis web tetapi juga dikarenakan sisten ini digunakan oleh masyarakat umum. Dengan demikian *interface* yang dibuat harus mudah dipahami oleh user.

Adapun dalam menggunakan UML penulis menggunakan beberapa diagram yang yaitu *use case diagram, class diagram, activity diagram, sequence diagram* dan *deployment diagram.* Adapun pada pemodelan database penulis menggunakan diagram relasi antar tabel.

3.2.1.4. Construction

a. **Coding**

Pada tahap ini penulis melakukan pembuatan prototype algoritma Dijkstra yang selanjutnya akan diintegrasikan dengan bagian-bagian dari sistem yang lainnya dengan menggunakan konsep pemrograman Model View Control (MVC) yang terdapat pada frame work Code Igniter (CI) Selanjutnya penulis mengembangkan prototype tersebut dan mengintegrasikan

dengan bagian-bagian sistem lainnya yang juga dibuat secara bersamaan.

b. **Testing**

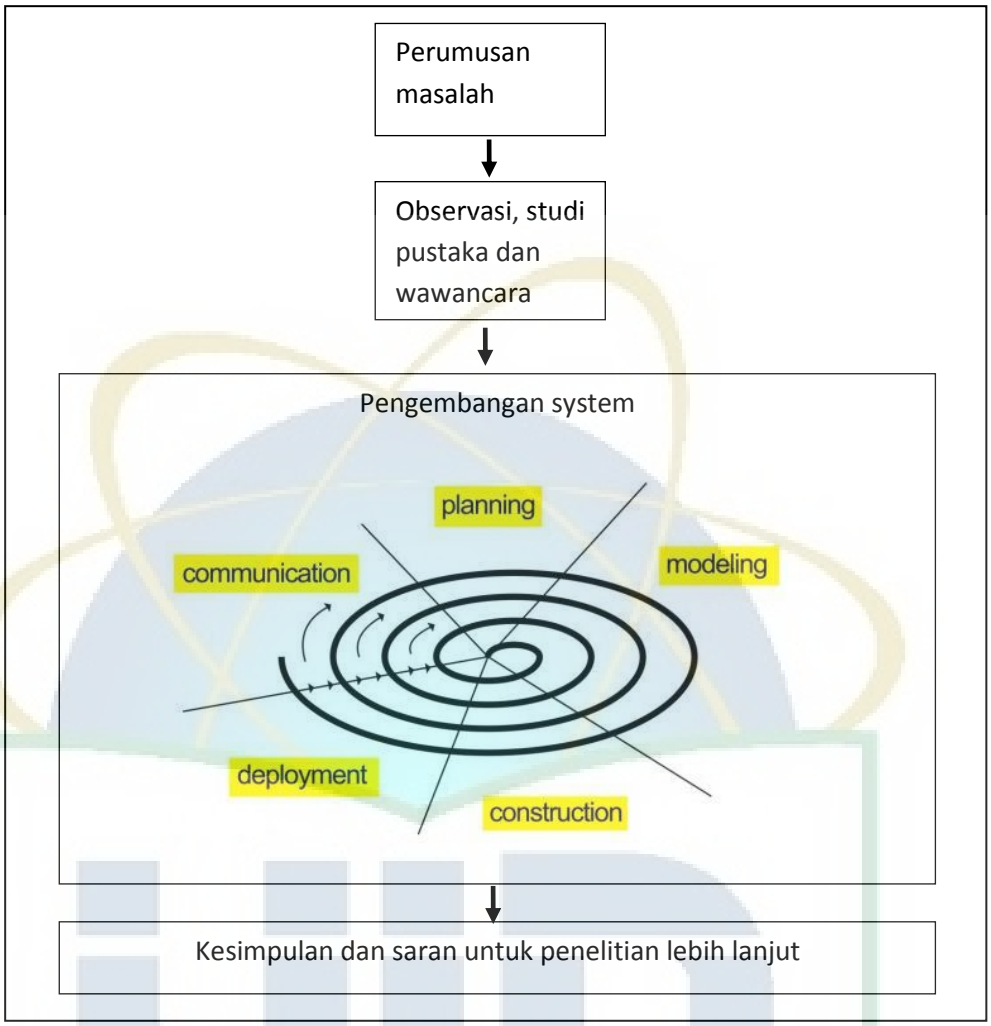
Pada tahap ini dilakukan pengujian dari coding yang telah dibangun. Hasil dari testing ini selanjutnya akan menjadi bahan evaluasi sistem.

3.2.1.5. Deployment

Tahap deployment yaitu tahap implementasi system. Tahap ini meliputi kegiatan saat komponen-komponen dalam system di-*deploy.* Selanjutnya system dapat running dan bisa mulai digunakan.

3.3. Kerangka Berpikir

Dalam penelitian ini penulis melakukan tahapan-tahapan kegiatan dengan mengikuti kerangka berpikir sebagai berikut.



Gambar 3.1 Kerangka Berpikir

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem

Berdasarkan hasil observasi dan wawancara yang penulis lakukan pada Dinas Perhubungan Provinsi DKI Jakarta. Terdapat berbagai macam faktor yang dapat menyebabkan kemacetan sehingga mempengaruhi waktu tempuh perjalanan. Secara garis besar, faktor-faktor tersebut diantaranya yaitu fluktuasi arus dan hambatan samping. Fluktuasi arus adalah penumpukan suatu arus kendaraan yang menuju pada suatu tempat dengan menggunakan pilihan jalur yang sama sehingga menimbulkan kepadatan lalu-lintas dan selanjutnya menyebabkan kemacetan. Fluktuasi arus ini biasanya terjadi pada jam berangkat kerja dan pulang kerja. Sedangkan hambatan samping diantaranya berupa parkir di tepi jalan, penyebrang jalan, pedagang kaki lima dan angkutan umum yang berhenti di tepi jalan untuk mencari penumpang. Data penyebab kemacetan secara lebih rinci terdapat pada lampiran 2, adapun berikut ini penjelasan dari faktor-faktor penyebab kemacetan tersebut sebagai berikut:

- Lalu lintas padat, lalu lintas padat merupakan akibat dari fluktuasi arus. Lalu lintas yang padat dapat mengurangi laju kendaraan sehingga terjadi antrian kendaraan.
- Parkir di tepi jalan, parkir di tepi jalan dapat mempersempit ruas jalan yang dilalui kendaraan lain, sehingga mengurangi laju kendaraan yang

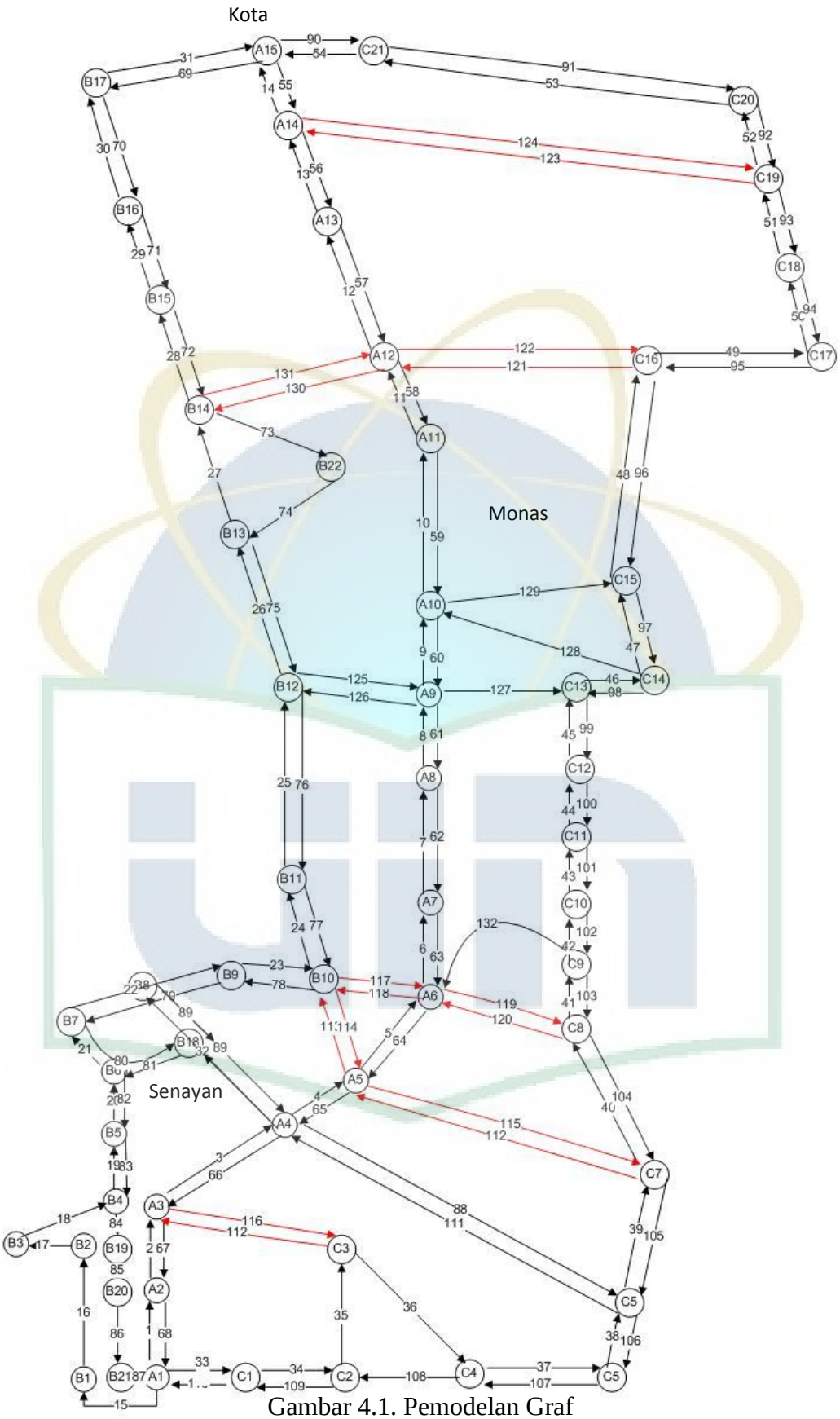
melewati daerah tersebut.

- Penyebrang jalan, penyebrang jalan dapat menyebabkan antrian kendaraan ketika lalu lintas ramai, karena akan mengurangi laju kendaraan yang melewati jalur tersebut.
- Pedagang kaki lima, pedagang kaki lima dapat mempersempit ruas jalan, ditambah dengan pembeli yang berlalu-lalang sehingga tentu saja akan mengurangi laju kendaraan dan menyebabkan antrian kendaraan.
- Angkutan umum yang berhenti di tepi jalan untuk mencari penumpang, hal ini dapat mengakibatkan kendaraan yang berada dibelakang nya ikut terhenti sehingga sehingga dapat menyebabkan antrian kendaraan.

Selain data kemacetan, penulis juga mendapatkan data kecepatan dan waktu tempuh kendaraan serta data volume lalu lintas dari pihak Dishub Provinsi DKI Jakarta. Data tersebut didapatkan dari hasil penelitian yang dilakukan pihak Dishub setiap tahunnya. Penelitian tersebut bertujuan untuk memantau arus lalu-lintas yang terjadi di Jakarta. Dalam hal ini penulis hanya menggunakan data kecepatan dan waktu tempuh kendaraan, dikarenakan data mengenai volume lalu-lintas tidak mencukupi untuk digunakan pada penelitian ini. Pada data kecepatan dan waktu tempuh kendaraan, kecepatan dan waktu tempuh kendaraan dihitung antara dua titik persimpangan. Selain itu, pihak Dishub membagi menjadi enam bagian waktu pengukuran yaitu:

- Pagi sebelum *three in one* yaitu pukul 00.00 – 06.59 WIB
- Pagi saat *three in one* yaitu pukul 07.00 – 10.00 WIB
- Pagi setelah *three in one* yaitu pukul 10.01 – 11.59 WIB
- Sore sebelum *three in One* yaitu pukul 12.00 – 15.59 WIB
- Sore saat *three in one* yaitu pukul 16.00 – 20.00 WIB
- Sore setelah *three ini one* yaitu pukul 20.01 – 23.59 WIB

Dari data kecepatan dan waktu tempuh tersebut, selanjutnya penulis membuat sebuah pemodelan graf yang dapat merepresentasikan setiap persimpangan dan jaringan jalan yang berada pada wilayah penelitian. Dalam graf tersebut node merupakan persimpangan, sementara simpul merupakan jalur yang menghubungkan anatara dua persimpangan. Berikut ini pemodelan graf yang penulis maksud.



Gambar 4.1. Pemodelan Graf

Pada Graf tersebut, nama persimpangan diwakili oleh nama node sedangkan nama jalur diwakili dengan penomoran pada tiap simpul. Adapun pemberian huruf A, B dan C pada nama node dimaksudkan untuk memudahkan dalam mengelompokan letak persimpangan. Adapun pengelompokan tersebut sebagai berikut.

1. Persimpangan yang berada pada jalur tengah atau jalur utama antara Blok M dan Kota dilambangkan dengan nama node dengan huruf A.
2. Persimpangan yang berada sebelah barat dari jalur utama antara blok M dan Kota dilambangkan dengan nama node dengan huruf B.
3. Persimpangan yang berada pada sebelah timur dari jalur utama antara Blok M dan Kota dilambangkan dengan nama node dengan huruf C.

Berikut ini daftar node yang terdapat pada graf diatas.

Tabel 4.1. Daftar Node Pada Pemodelan Graf

No	Node	Nama Persimpangan
1	A1	Simpang CSW
2	A2	Raden Patah 2
3	A3	Bundaran Senayan
4	A4	Fly Over Semanggi
5	A5	Fly Over Sudirman
6	A6	Dukuh atas
7	A7	Bundaran HI
8	A8	Wahid Hasyim-thamrin
9	A9	Kebon Sirih-thamrin
10	A10	Patung Arjuna
11	A11	Merdeka Utara
12	A12	Harmoni
13	A13	Zainul Arifin-gajah mada
14	A14	Mangga Besar-gajah mada

15	A15	U Turn Lindeteves
16	B1	Bulungan
17	B2	Pakubuwono 6 Barat
18	B3	Hang Lekir
19	B4	Senayan City
20	B5	Pintu Senayan
21	B6	Asia Afrika
22	B7	Gelora
23	B8	Penjernihan
24	B9	Bendungan Hilir
25	B10	Fly Over Karet
26	B11	Kebon Kacang
27	B12	Fly Over Cideng
28	B13	Tanah Abang 2
29	B14	Caringin
30	B15	Hasyim Asyari
31	B16	Zainul Arifin
32	B17	Tubagus Angke
33	B18	Flyover Taman Ria
34	B19	Pakubuwono 6
35	B20	Hang Tuah 7
36	B21	Kyai Maja
37	B22	Jl Kesehatan
38	B23	Flover Slipi
39	C1	Trunojoyo
40	C2	Gunawarman
41	C3	Senopati
42	C4	Tendean
43	C5	Mampang
44	C6	Fly Over Kuningan
45	C7	Under Pass Casablanca
46	C8	Fly Over Sultan Agung
47	C9	Imam Bonjol
48	C10	Sultan Syahrir
49	C11	Sam Ratulangi
50	C12	Wahid Hasyim
51	C13	Kebon Sirih
52	C14	Tugu Tani
53	C15	Merdeka Tenggara
54	C16	Juanda
55	C17	Gunung Sahari
56	C18	Samanhudi
57	C19	Mangga Besar
58	C20	Jayakarta

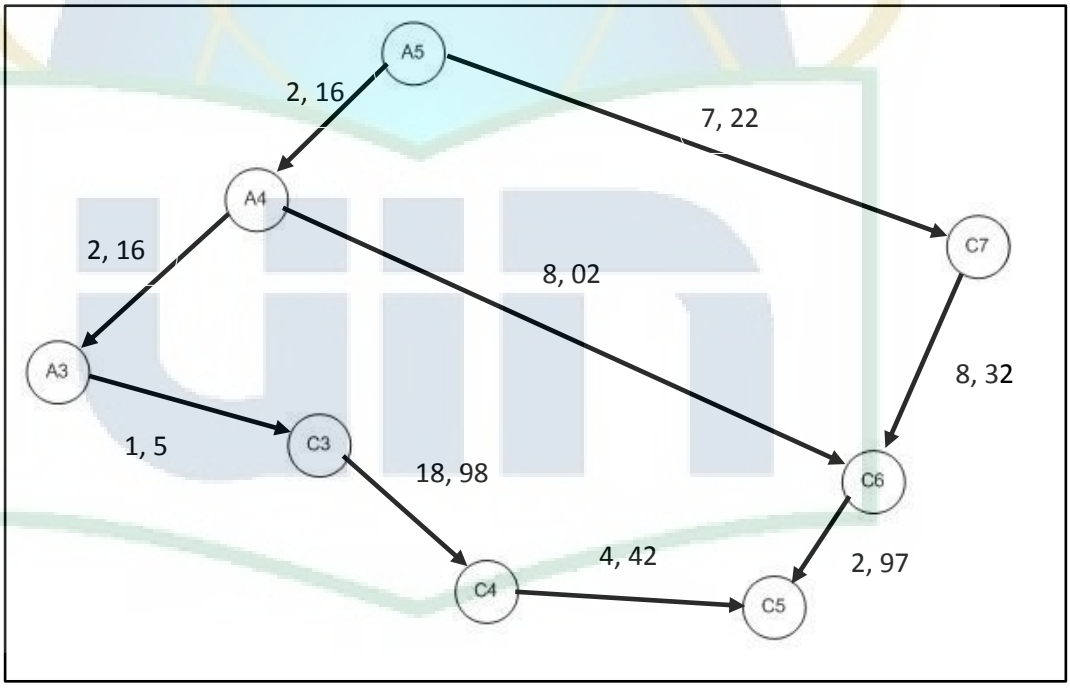
59	C21	Mangga Dua
60	C22	Kathedral
61	C23	Merdeka Timur
62	C24	Cut Mutia

Dari uraian pemodelan graf diatas, dapat diambil sebuah contoh kasus misalkan pada pukul 11.00 WIB, seseorang yang hendak menempuh perjalanan dari wilayah Sudirman ke Wilayah Mampang tepatnya dari persimpangan Flyover Sudirman ke persimpangan Mampang (pada graf dinotasikan dari node A5 ke node C5). Seseorang tersebut menginginkan jalur tercepat yang dapat dilalui. Maka kita dapat mengambil sampel tiga kemungkinan jalur yang dapat dilalui, yaitu:

1. Jalur pertama yaitu dari persimpangan Flyover Sudirman ke persimpangan Underpass Casablanca kemudian ke persimpangan Flyover Kuningan kemudian ke persimpangan Mampang. Pada graf jalur tersebut dinotasikan A5 => C7 => C6 => C5.
2. Jalur kedua yaitu dari persimpangan Flyover Sudirman ke persimpangan Semanggi kemudian ke persimpangan Flyover Kuningan kemudian ke persimpangan Mampang. Pada graf jalur tersebut dinotasikan A5 => A4 => C6 => C5.
3. Jalur ketiga yaitu dari persimpangan Flyover Sudirman ke persimpangan Semanggi kemudian ke persimpangan Bundaran Senayan kemudian ke persimpangan Senopati kemudian ke persimpangan Tendean kemudian ke persimpangan Mampang. Jalur

tersebut pada graf dinotasikan dengan A5 => A4 => C3 => C4 => C5.

Dari ketiga sampel jalur tersebut diperoleh sampel node-node yang dapat dilalui dari persimpangan Sudirman ke persimpangan Mampang yaitu node A5, A4, C3, C4, C6, C7 dan C5. Dari sampel node-node tersebut dapat dibuat graf sederhana yang hanya mencakup node-node dan jalur yang mengarah ke node C5. Berikut ini graf yang dimaksud yang dilengkapi dengan data waktu tempuh setiap jalur pada pukul 11.00 WIB (dalam satuan menit).



Gambar 4.2 Graf Sederhana Sebagai Sampel

Dari graf sampel tersebut selanjutnya sistem akan melakukan perhitungan dengan algoritma Dijkstra. Berikut ini uraian langkah-langkah pada algoritma Dijkstra untuk menangani kasus tersebut.

1. Input berupa node awal, node tujuan dan graf. Dalam hal ini node awal

yaitu node A5, node tujuan yaitu node C5 sementara input graf berupa graf sederhana yang terdiri dari node A5, A4, A3, C3, C4, C5, C6 dan C7 beserta path yang memiliki bobot waktu tempuh.

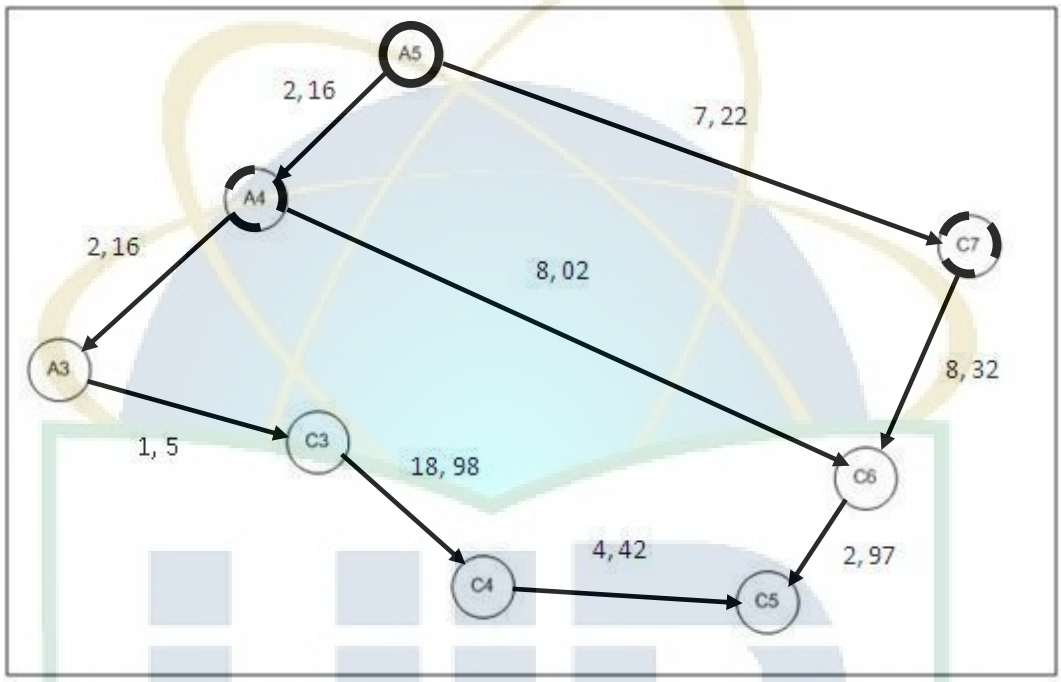
2. **Langkah pertama yaitu menetapkan node awal sebagai status ditemukan (*found*) dan kemudian dikunjungi**. Dalam hal ini berarti menetapkan node A5 berstatus ditemukan dan kemudian berstatus dikunjungi. Sementara itu semua node lainnya berstatus belum ditemukan dan belum dikunjungi.

Tabel 4.2 Kondisi Node Pada Saat Awal Pencarian

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Belum ditemukan	~	-
A4	Belum ditemukan	~	-
C3	Belum ditemukan	~	-
C4	Belum ditemukan	~	-
C6	Belum ditemukan	~	-
C7	Belum ditemukan	~	-
C5 (tujuan)	Belum ditemukan	~	-

3. **Langkah kedua yaitu dilakukan pencarian terhadap setiap node yang dapat dicapai secara langsung dari node yang sedang dikunjungi**. Karena saat ini node yang sedang dikunjungi yaitu node A5. Maka dilakukan pencarian terhadap setiap node yang dapat dicapai secara langsung dari node A5. Dalam hal ini ditemukan dua node yaitu node A4 dan node C7 dengan memiliki akumulasi bobot saat ini A4 =

2,16 dan C7 = 7,22. Adapun bobot akumulasi adalah total bobot yang dapat dicapai dari node awal ke node tersebut melalui rute yang sedang ditelusuri.



Gambar 4.3 Kondisi Graf Saat Node A4 dan C7 Ditemukan

4. **Langkah ketiga yaitu:**
   - **Apabila node yang didapatkan pada langkah kedua belum pernah ditemukan, maka rubah statusnya menjadi ditemukan**
   - **Apabila node yang didapatkan sudah pernah ditemukan maka lakukan *update* pada bobotnya, ambil bobot yang lebih kecil**.

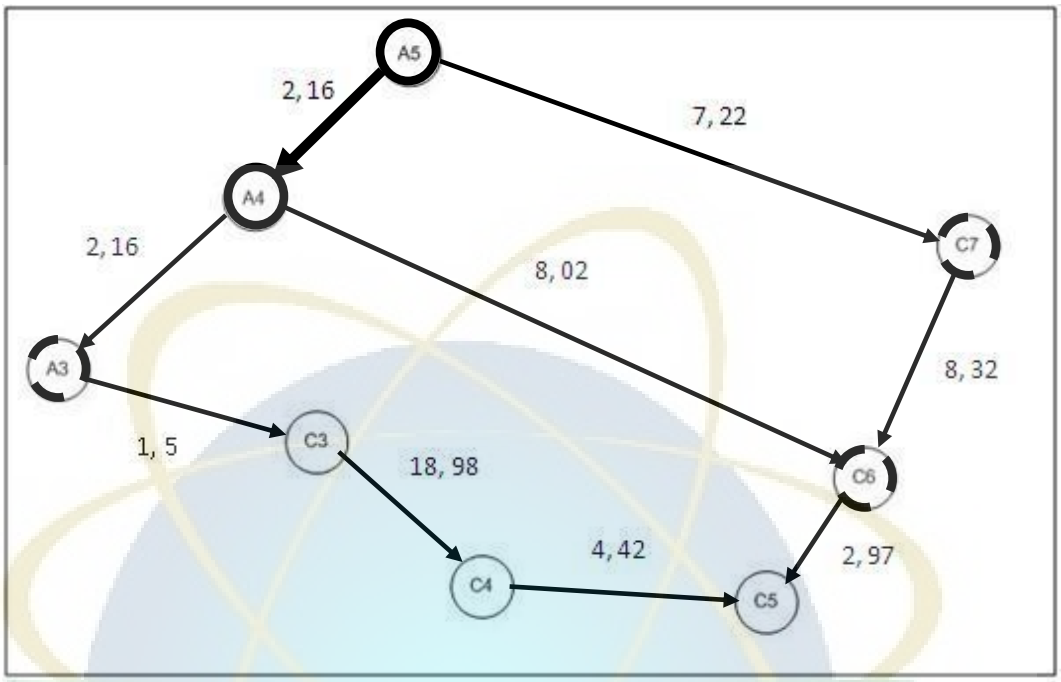
Pada saat ini node yang didapatkan adalah node A4 dan node C7 dimana node tersebut belum pernah ditemukan sebelumnya, maka

kedua node tersebut berubah statusnya menjadi ditemukan.

Tabel 4.3 Kondisi Node Pada Saat Node A4 dan C7 Ditemukan

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Belum ditemukan	~	-
A4	Ditemukan	2, 16	A5 => A4
C3	Belum ditemukan	~	-
C4	Belum ditemukan	~	-
C6	Belum ditemukan	~	-
C7	Ditemukan	7, 22	A5 => C7
C5 (tujuan)	Belum ditemukan	~	-

5. **Langkah keempat yaitu dilakukan pencarian terhadap node yang memiliki bobot paling kecil dari semua node yang berada pada status ditemukan kemudian mengunjunginya**. Maka untuk saat ini pencarian tersebut akan menghasilkan node A4 sebagai node dengan bobot paling kecil. Kemudian selanjutnya sistem mengunjungi node A4 tersebut. Dengan demikian node A4 berubah statusnya menjadi dikunjungi.
6. Selanjutnya dijalankan kembali langkah kedua yaitu dilakukan pencarian node yang dapat dicapai secara langsung dari node A4. Maka sistem akan menemukan node A3 dan C6 dengan bobot akumulasi node A3 = 4, 32 sedangkan node C6 = 10, 18. Selanjutnya dijalankan kembali langkah ketiga, maka pada node A3 dan node C6 berubah statusnya menjadi ditemukan.

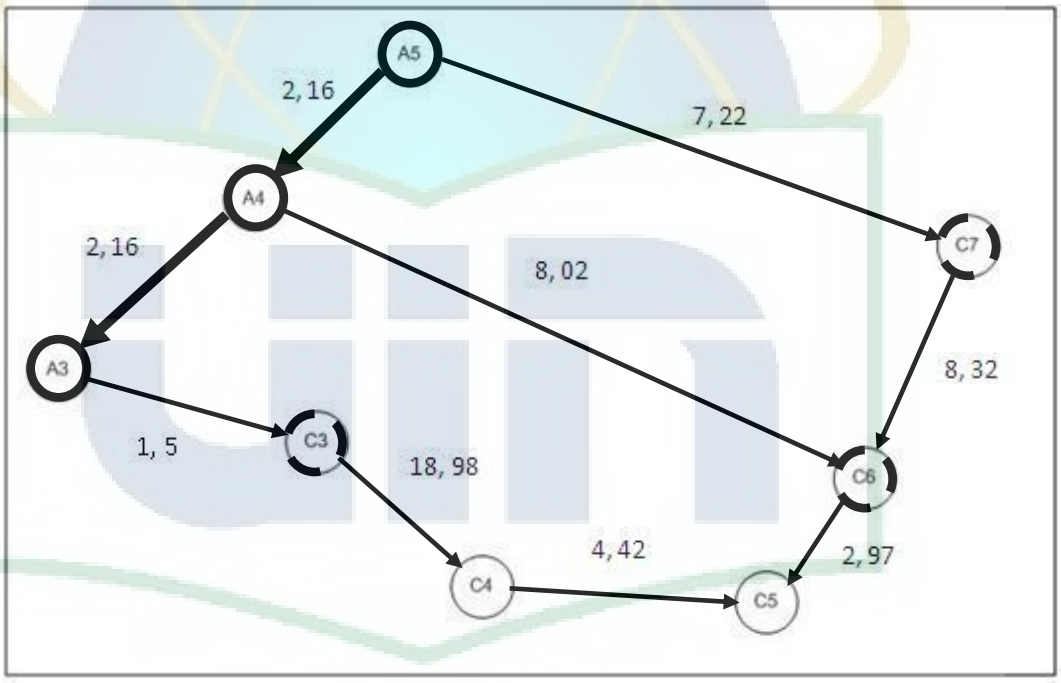


Gambar 4.4 Kondisi Graf Saat Node A3 dan C6 Ditemukan

Tabel 4.4 Kondisi Node Pada Saat Node A3 dan C6 Ditemukan

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Ditemukan	4, 32	A5 => A4 => A3
A4	Dikunjungi	2, 16	A5 => A4
C3	Belum ditemukan	~	-
C4	Belum ditemukan	~	-
C6	Ditemukan	10, 18	A5 => A4 => C6
C7	Ditemukan	7, 22	A5 => C7
C5 (tujuan)	Belum ditemukan	~	-

7. Selanjutnya dijalankan kembali langkah keempat, dalam hal ini node yang berada pada status ditemukan yaitu node C7, A3, dan C6. Maka akan dihasilkan node A3 dengan bobot terkecil dan sistem akan mengunjungi node A3.
8. Selanjutnya pada node A3 dilakukan kembali langkah kedua, maka akan ditemukan node C3 dengan bobot akumulasi 5, 82. Setelah itu dijalankan langkah ketiga, maka node C3 berubah statusnya menjadi ditemukan.

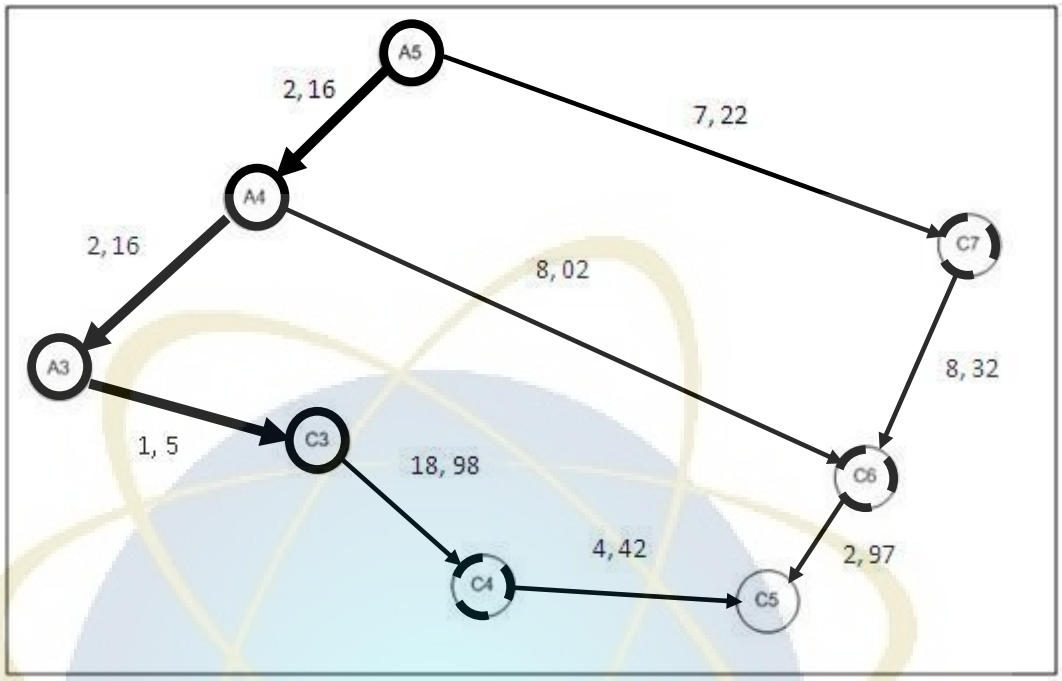


Gambar 4.5 Kondisi Graf Saat Node C3 Ditemukan

Tabel 4.5 Kondisi Node Pada Saat Node C3 Ditemukan

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Dikunjungi	4, 32	A5 => A4 => A3
A4	Dikunjungi	2, 16	A5 => A4
C3	Ditemukan	5, 82	A5 => A4 => A3 => C3
C4	Belum ditemukan	~	-
C6	Ditemukan	10, 18	A5 => A4 => C6
C7	Ditemukan	7, 22	A5 => C7
C5 (tujuan)	Belum ditemukan	~	-

9. Selanjutnya dijalankan langkah keempat, dimana dalam hal ini node yang berada pada status ditemukan yaitu node C7, C6, dan C3. Maka akan dihasilkan node C3 dengan bobot terkecil dan sistem akan mengunjungi node C3.
10. Selanjutnya dilakukan kembali langkah kedua pada node C3 maka akan ditemukan node C4 dengan bobot akumulasi 24, 8. Setelah itu dijalankan langkah ketiga, maka node C4 berubah statusnya menjadi ditemukan



Gambar 4.6 Kondisi Graf Saat Node C4 Ditemukan

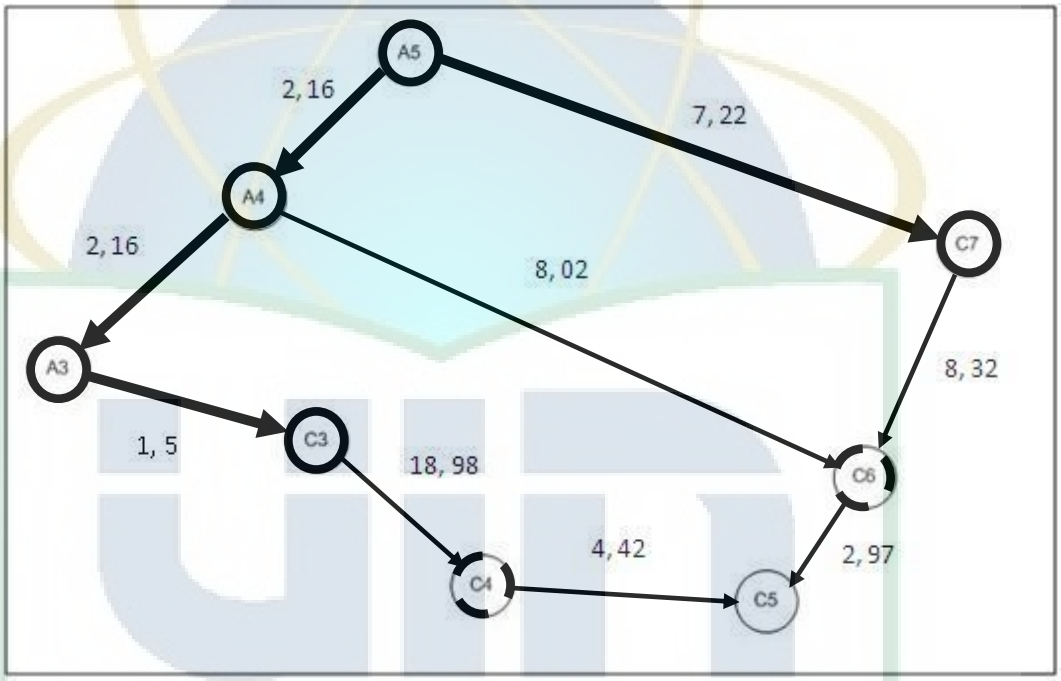
Tabel 4.6 Kondisi Node Pada Saat Node C4 Ditemukan

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Dikunjungi	4, 32	A5 => A4 => A3
A4	Dikunjungi	2, 16	A5 => A4
C3	Dikunjungi	5, 82	A5 => A4 => A3 => C3
C4	Ditemukan	24,8	A5 => A4 => A3 => C3 => C4
C6	Ditemukan	10, 18	A5 => A4 => C6
C7	Ditemukan	7, 22	A5 => C7
C5 (tujuan)	Belum ditemukan	~	-

11. Selanjutnya dijalankan langkah keempat, dimana pada saat ini node yang berstatus ditemukan yaitu node C7, C6 dan C4. Maka akan

dihasilkan node C7 dengan bobot terkecil dan selanjutnya sistem akan mengunjungi node C7.

12. Selanjutnya dijalankan kembali langkah kedua, maka akan ditemukan kembali node C6 dengan bobot akumulasi pada jalur ini sebesar 15,54. Kemudian dilakukan langkah ketiga, maka sistem akan mengambil bobot C6 dengan jalur dari A4.



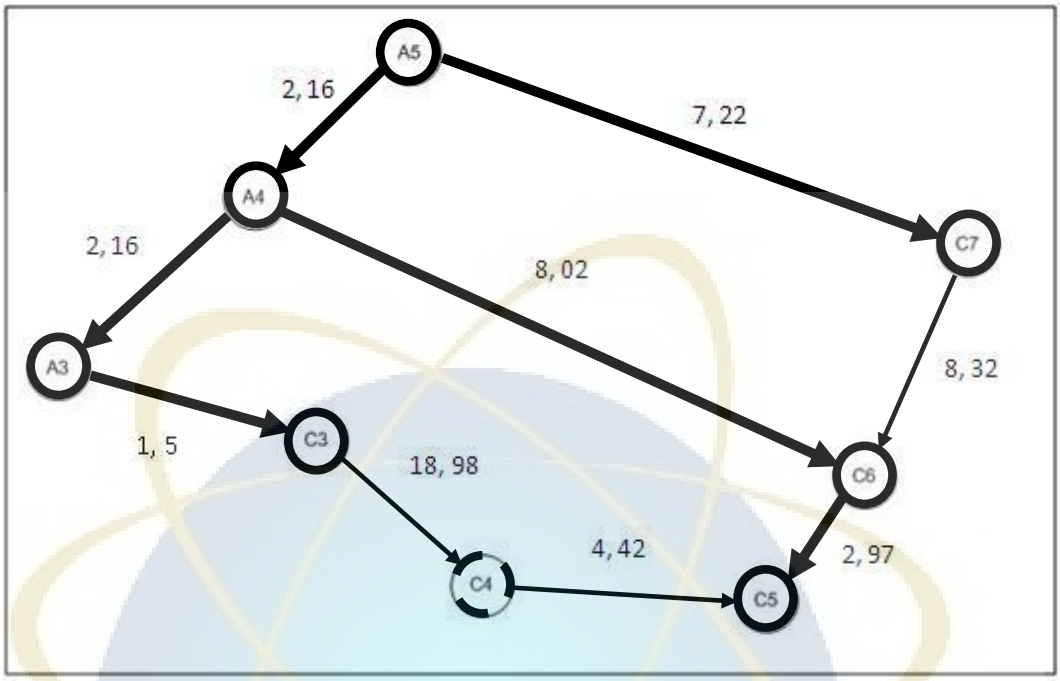
Gambar 4.7 Kondisi Graf Saat Node C7 Dikunjungi

Tabel 4.7 Kondisi Node Pada Saat Node C7 Ditemukan

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Dikunjungi	4, 32	A5 => A4 => A3
A4	Dikunjungi	2, 16	A5 => A4
C3	Dikunjungi	5, 82	A5 => A4 => A3 => C3
C4	Ditemukan	24,8	A5 => A4 => A3 => C3 =>

			C4
C6	Ditemukan	10, 18	A5 => A4 => C6
C7	Dikunjungi	7, 22	A5 => C7
C5 (tujuan)	Belum ditemukan	~	-

13. Selanjutnya dijalankan langkah keempat, dimana pada saat ini node yang berstatus ditemukan yaitu node C4 dan C6. Maka akan dihasilkan node C6 dengan bobot akumulasi terkecil. Selanjutnya sistem akan mengunjungi node C6.
14. Selanjutnya dijalankan kembali langkah kedua, maka akan ditemukan node C5 dengan bobot akumulasi sebesar 13, 15. Kemudian dilakukan kembali langkah ketiga sehingga status node C5 menjadi ditemukan. Dengan demikian maka saat ini node yang berstatus ditemukan yaitu node C4 dengan bobot 24, 8 dan node C5 dengan bobot 13, 15.
15. Selanjutnya dilakukan kembali langkah keempat, sehingga akan dihasilkan node C5 sebagai node dengan bobot akumulasi terkecil. Selanjutnya sistem akan mengunjungi node C5. Karena node C5 sebagai node tujuan, maka pencarian dihentikan. Dengan demikian rute dengan bobot terkecil yang dapat dicapai ke node C5 yaitu A5 => A4 => C6 => C5. Adapun total bobotnya sebesar 13, 15



Gambar 4.8 Kondisi Graf Saat Akhir Pencarian

Tabel 4.8 Kondisi Node Pada Akhir Pencarian

Node	Status	Bobot	Rute
A5 (asal)	Dikunjungi	0	A5
A3	Dikunjungi	4, 32	A5 => A4 => A3
A4	Dikunjungi	2, 16	A5 => A4
C3	Dikunjungi	5, 82	A5 => A4 => A3 => C3
C4	Ditemukan	24,8	A5 => A4 => A3 => C3 => C4
C6	Ditemukan	10, 18	A5 => A4 => C6
C7	Dikunjungi	7, 22	A5 => C7
C5 (tujuan)	Dikunjungi	13, 15	A5 => A4 => C6 => C5

Dari uraian langkah-langkah Algoritma Dijkstra diatas dapat diambil dua kesimpulan yaitu:

1. Dari ketiga kemungkinan jalur yang dapat dilalui dari persimpangan Flyover Sudirman ke persimpangan Mampang, maka didapatkan jalur tercepat adalah jalur kedua yaitu dari persimpangan Flyover Sudirman ke persimpangan Semanggi kemudian ke persimpangan Flyover Kuningan kemudian ke persimpangan Mampang, dengan total waktu tempuh 13, 15 menit.
2. Dalam kasus terburuk Algoritma Dijkstra dipastikan menemukan solusi terbaik. Kasus diatas hampir merupakan kasus terburuk dimana hampir setiap node dikunjungi untuk mencapai node tujuan.

4.2. Perancangan Sistem

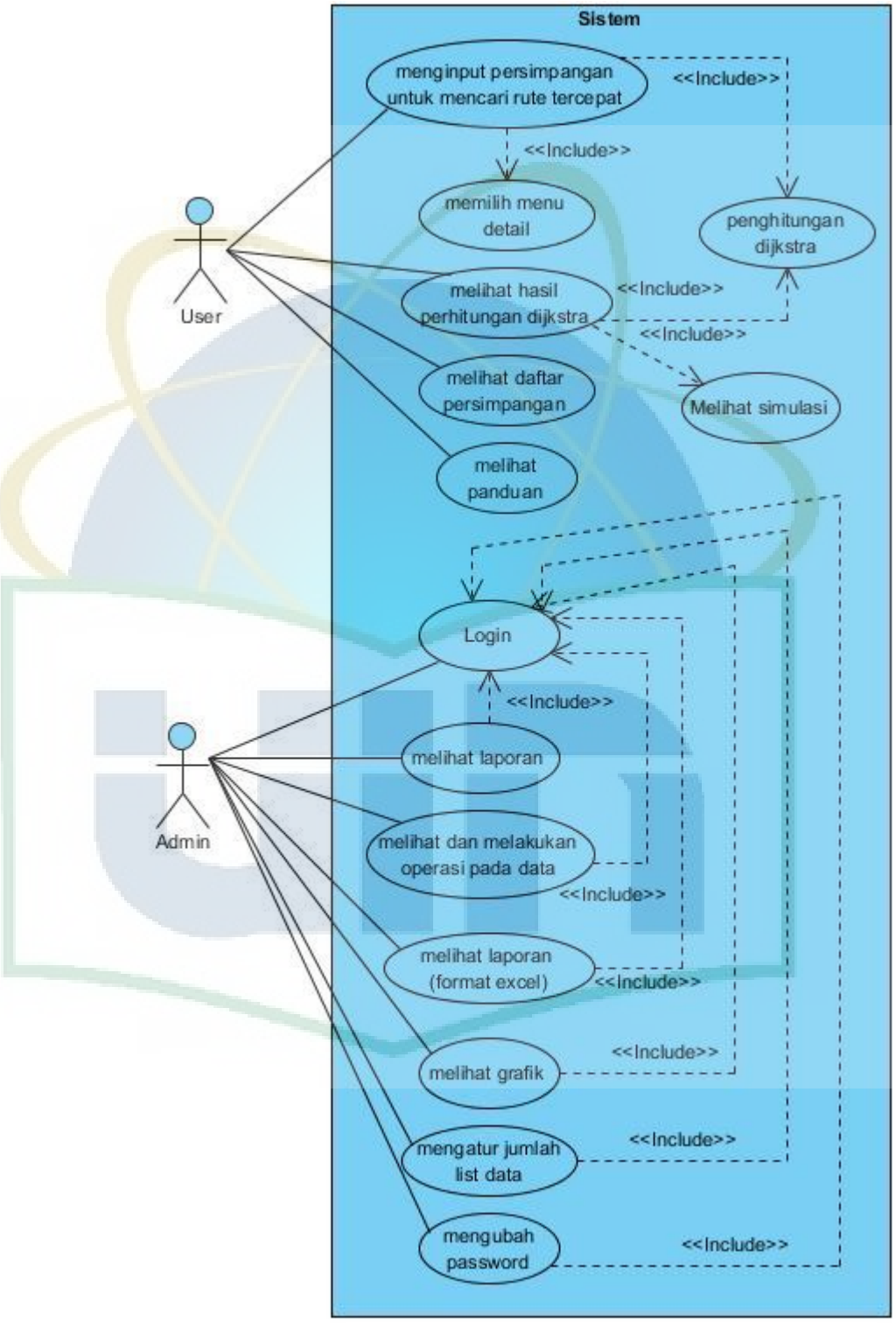
Berikut ini uraian tahap-tahap pengembangan sistem diantaranya *modeling, construction* dan *deployment*. Adapun tahap *communication* dan *planning* telah penulis uraikan pada bab sebelumnya.

4.2.1. *Modeling*

4.2.1.1. *Use Case Diagram*

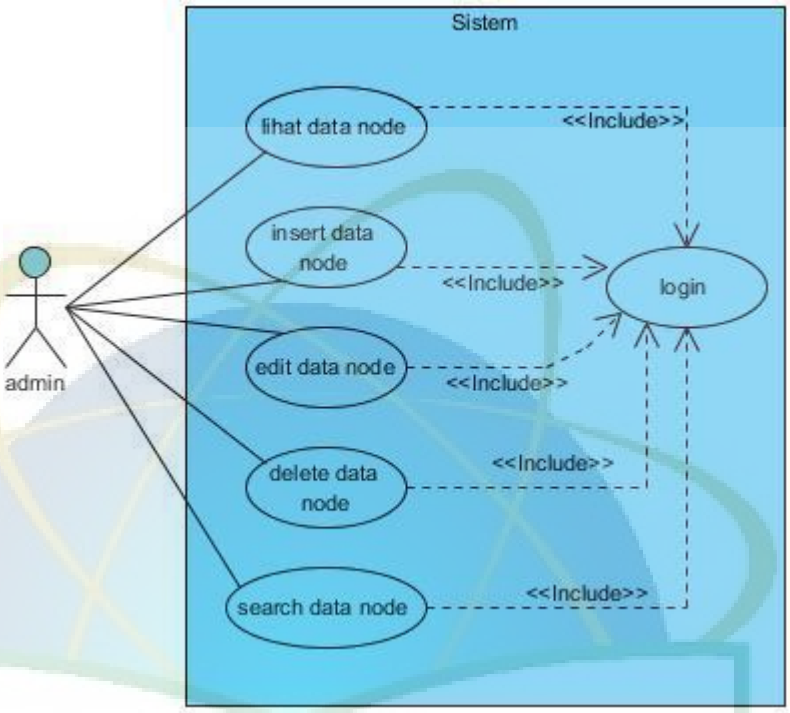
Berikut ini diagram *use case* yang penulis gunakan dalam penelitian ini. Untuk lebih mudah dipahami maka penulis membagi *use case* kedalam beberapa bagian yaitu *use case* user dan admin, *use case* node, *use case* path, *use case* wilayah dan *use case* jalan. Adapun *use case* node, *use case* path, *use case* wilayah dan *use case* jalan merupakan bagian dari *use case* "*melihat dan melakukan operasi pada data*" yang dijelaskan secara rinci.

a. *Use Case* User dan Admin



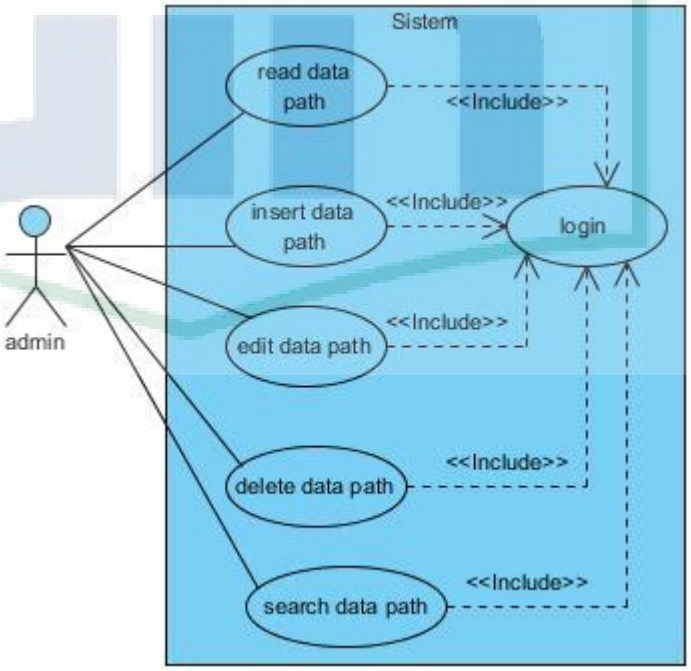
Gambar 4.9 *Use Case Diagram* User dan Login Admi

b. *Use Case* Node



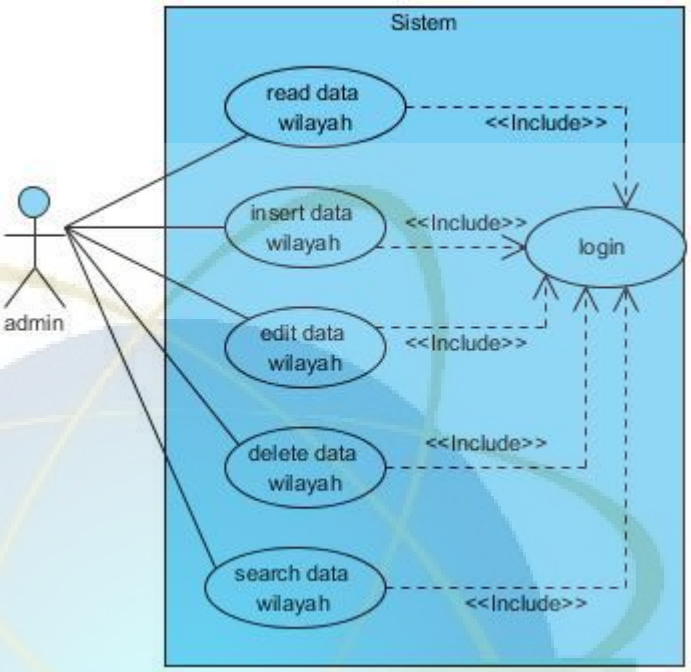
Gambar 4.10 *Use Case* Node

c. *Use Case* Path



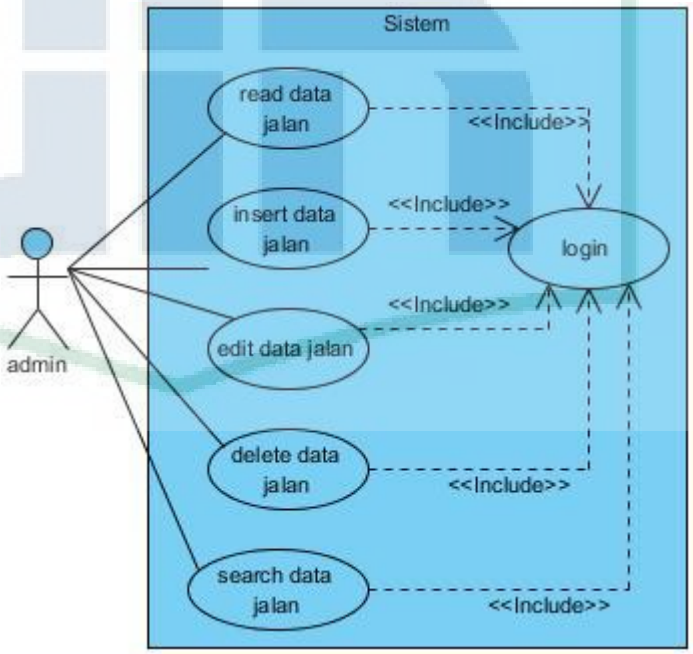
Gambar 4.11 *Use Case* Path

d. *Use Case* Wilayah



Gambar 4.12 *Use Case* Wilayah

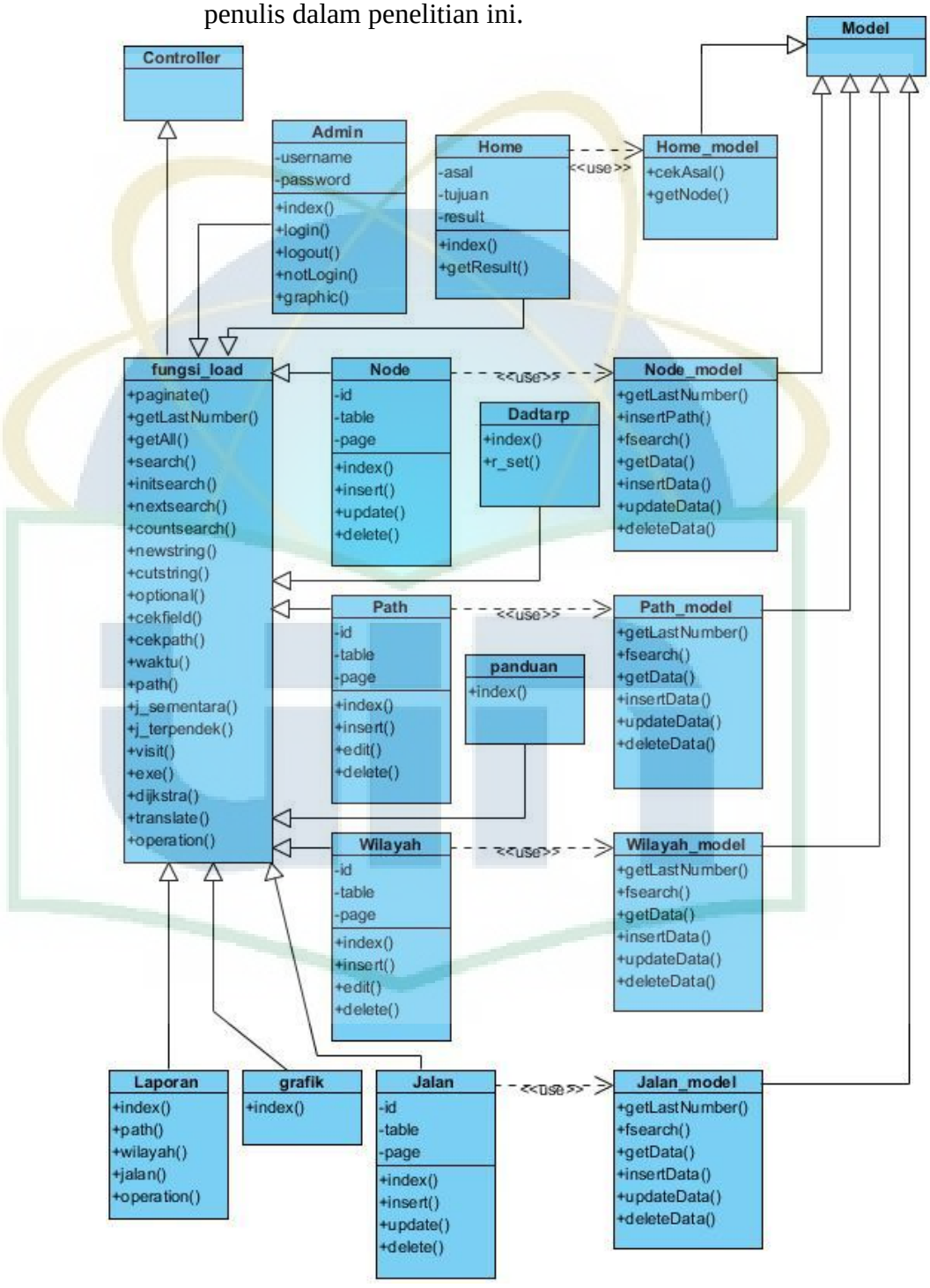
e. *Use Case* Jalan



Gambar 4.13 *Use Case* Jalan

4.2.1.2. *Class Diagram*

Berikut ini class diagram yang digunakan oleh penulis dalam penelitian ini.



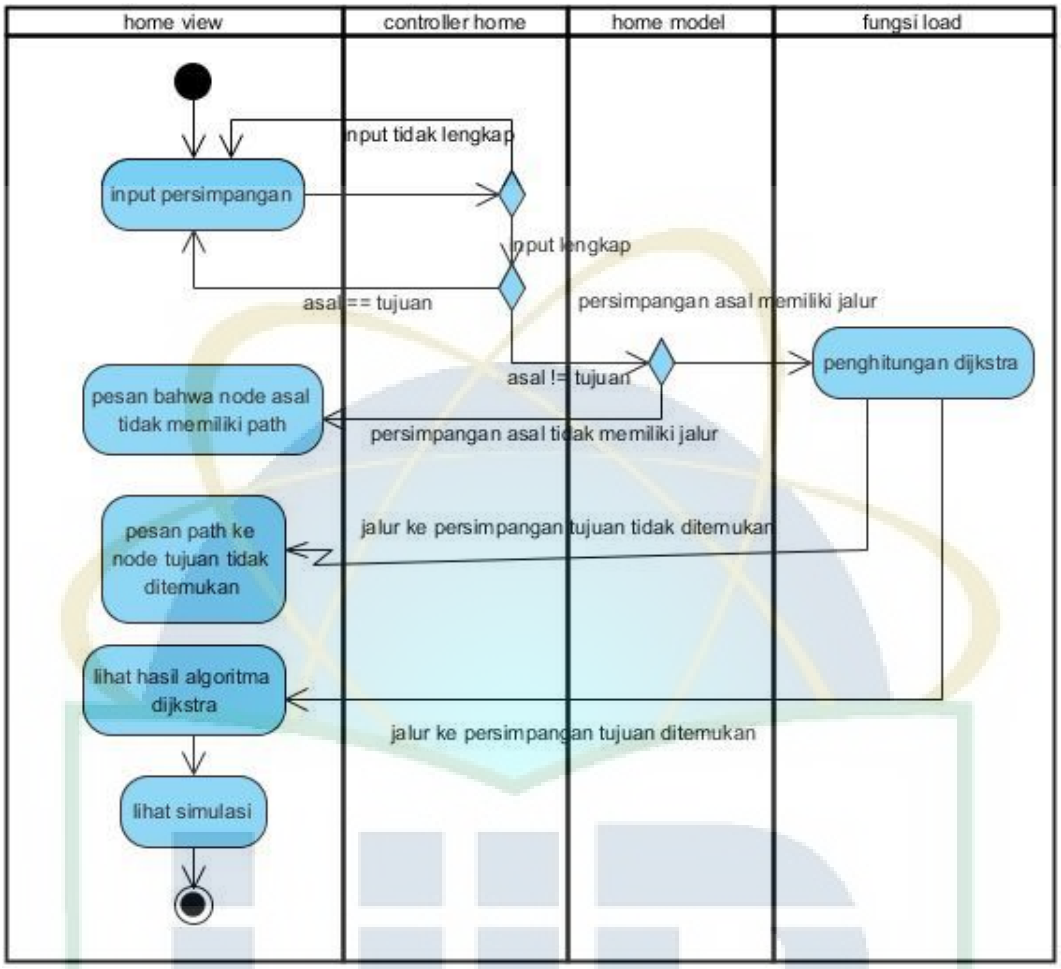
Gambar 4.14 *Class Diagram*

4.2.1.3. *Activity Diagram*

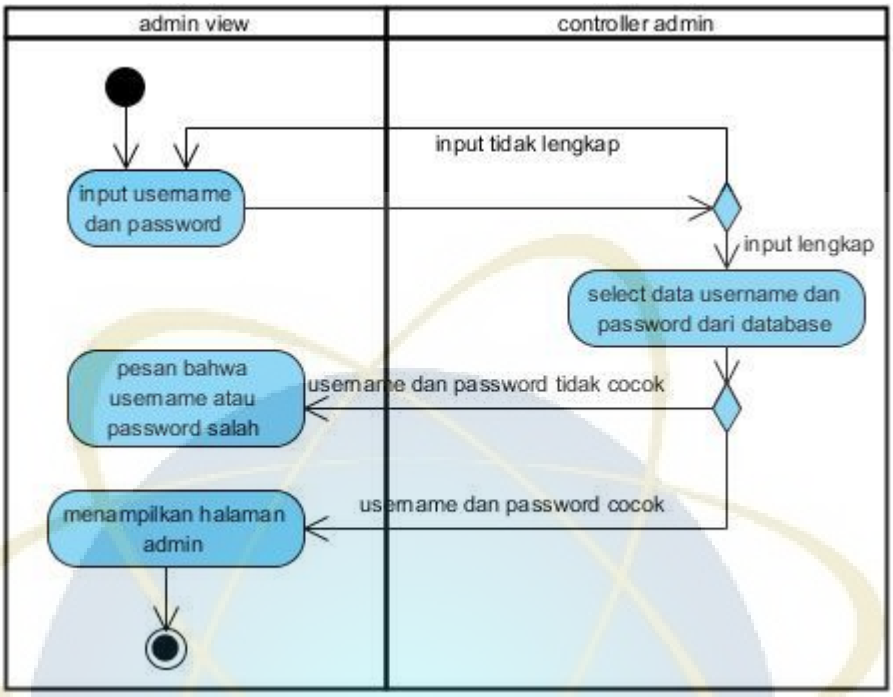
Penulis mengelompokan *activity diagram* berdasarkan menu yang ada pada sistem. *Activity diagram* tersebut yaitu:

a. *Activity diagram* menu home.
b. *Activity diagram* menu login admin.
c. *Activity diagram* menu data node.
d. *Activity diagram* menu data path.
e. *Activity diagram* menu data wilayah.
f. *Activity diagram* menu data jalan.
g. *Activity diagram* menu laporan.
h. *Activity diagram* menu grafik.
i. *Activity diagram* menu pengaturan data.
j. *Activity diagram* menu ubah password.
k. *Activity diagram* menu panduan.
l. *Activity diagram* menu daftar persimpangan.

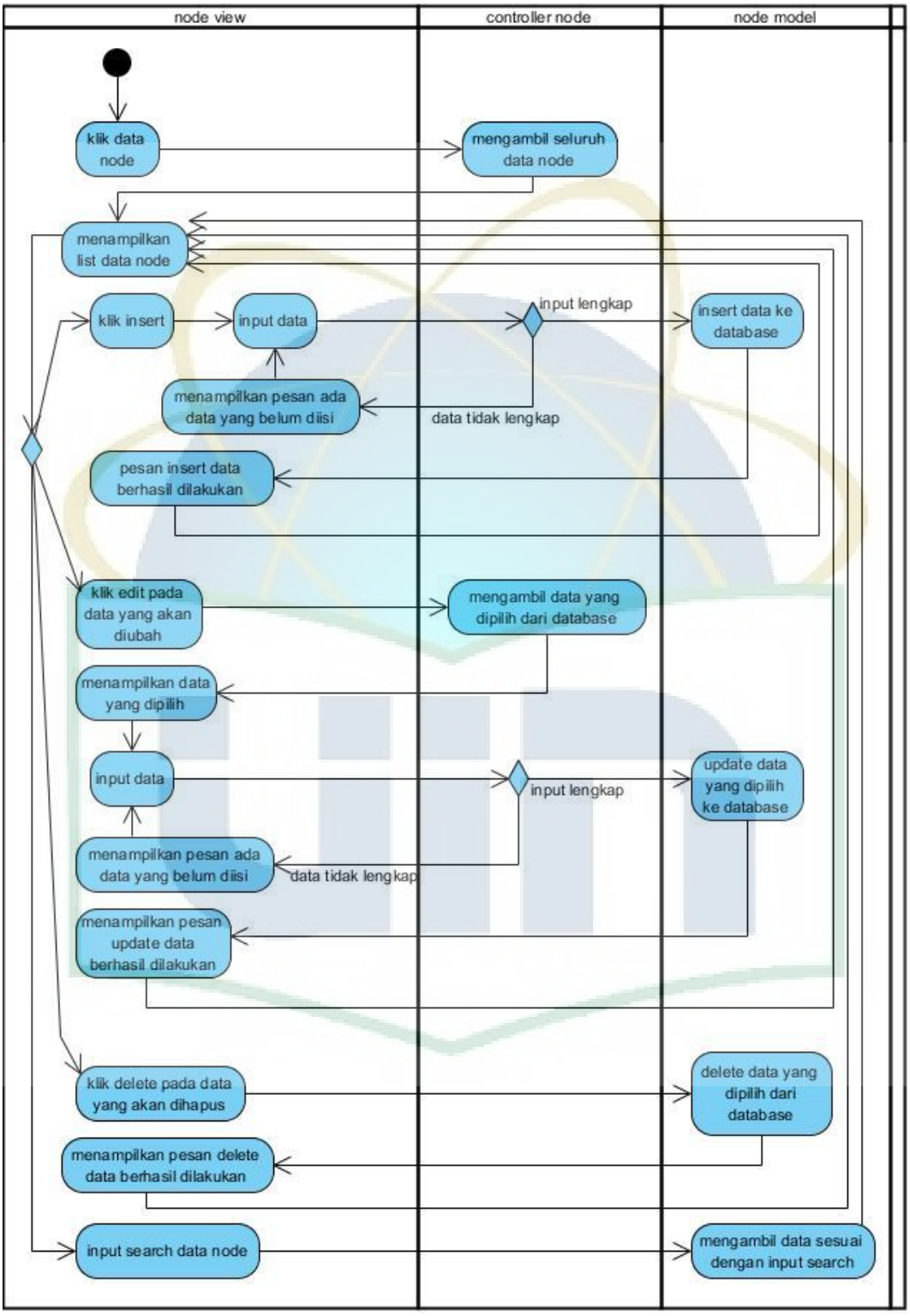
Berikut ini activity diagram yang penulis gunakan dalam penelitian ini.



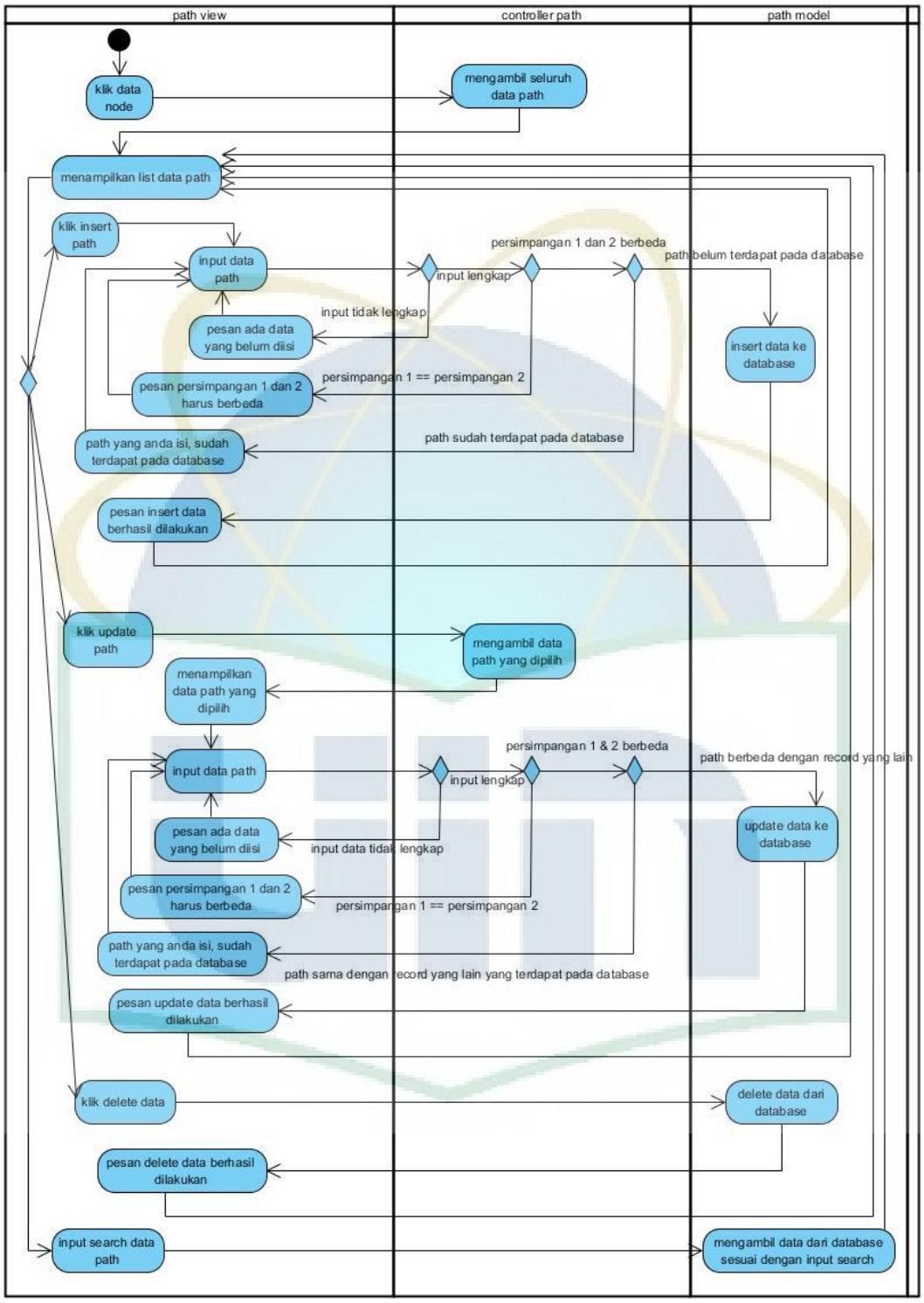
Gambar 4.15 *Activity Diagram* Menu Home



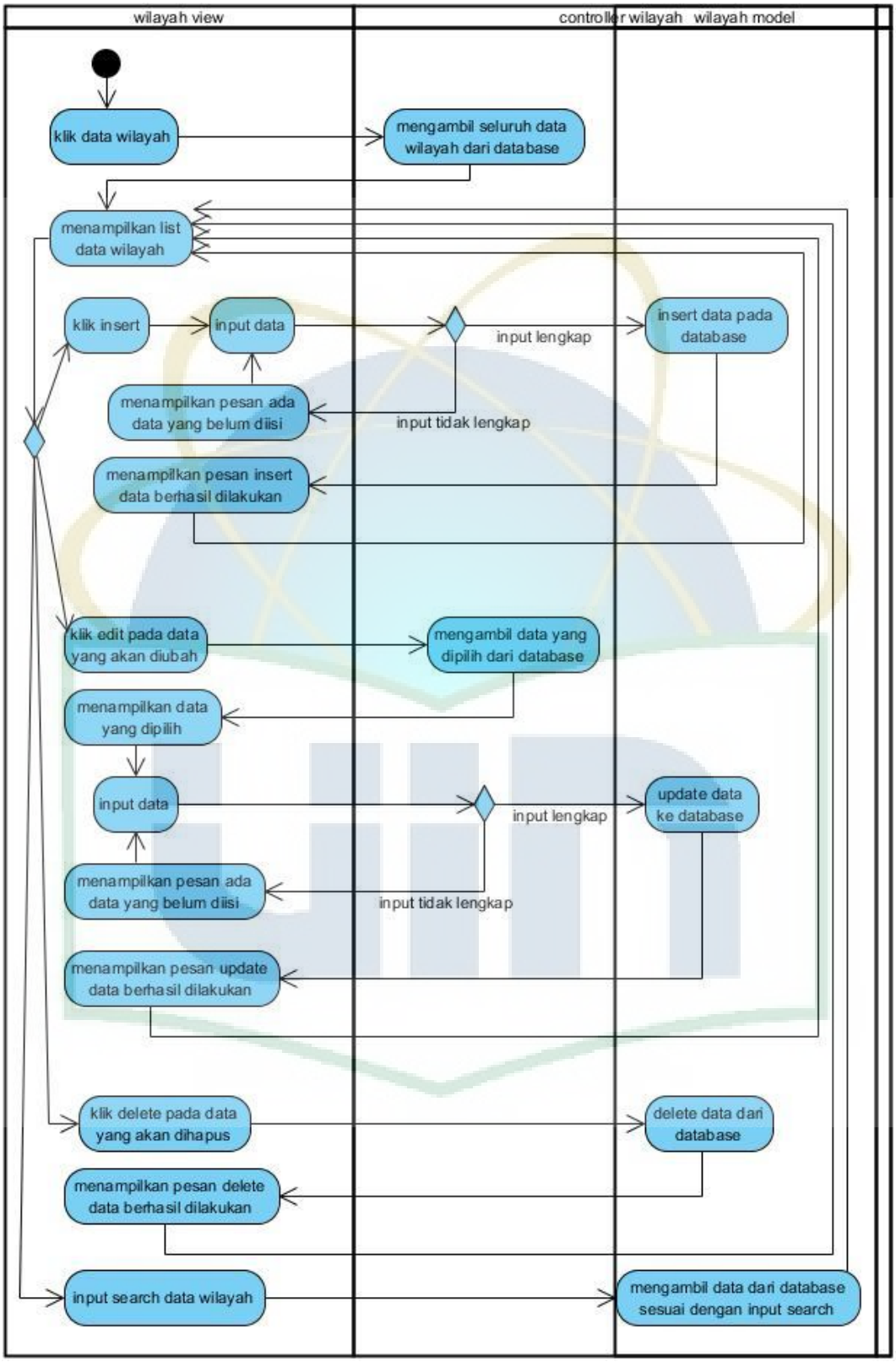
Gambar 4.16 *Activity Diagram* Menu Login Admin



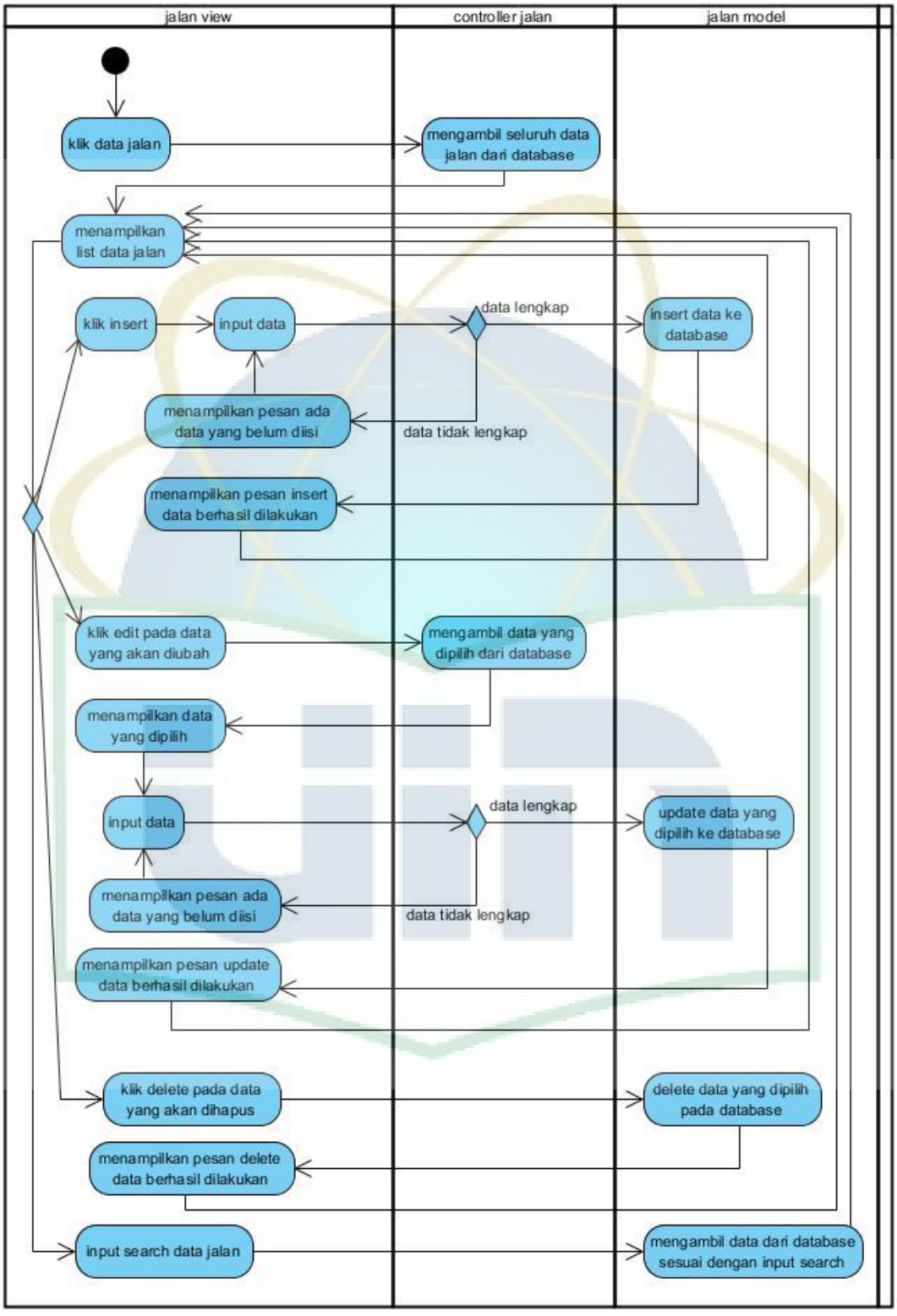
Gambar 4.17 *Activity Diagram* Menu Data Node



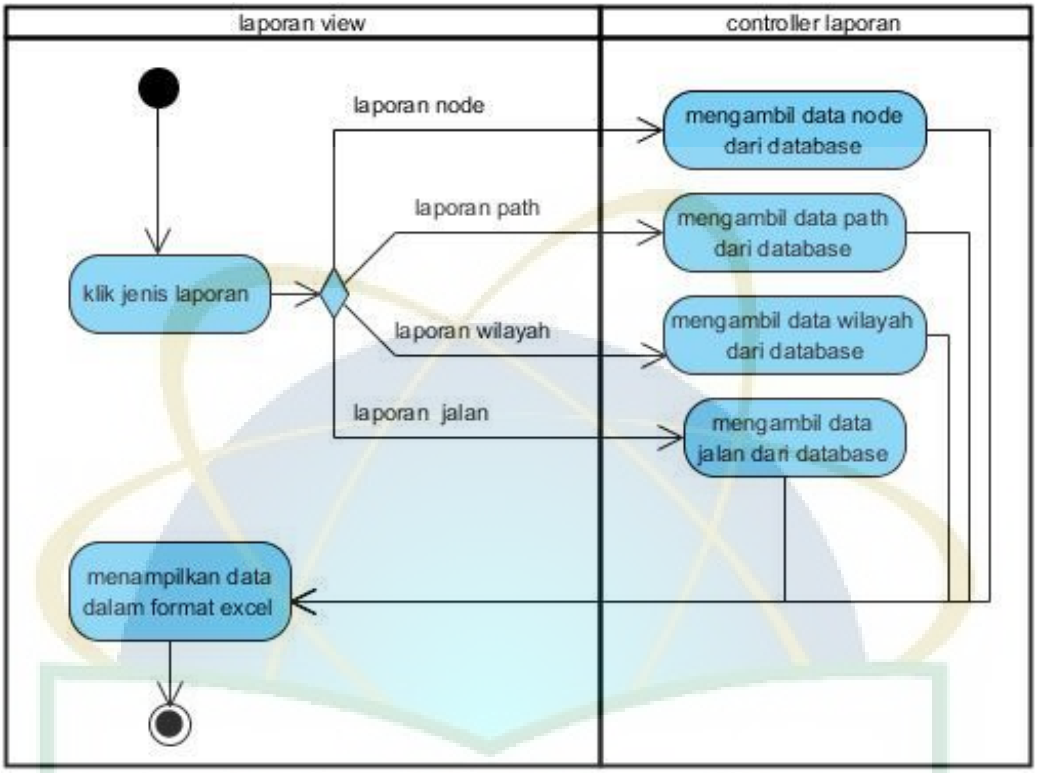
Gambar 4.18 *Activity Diagram* Menu Data Path



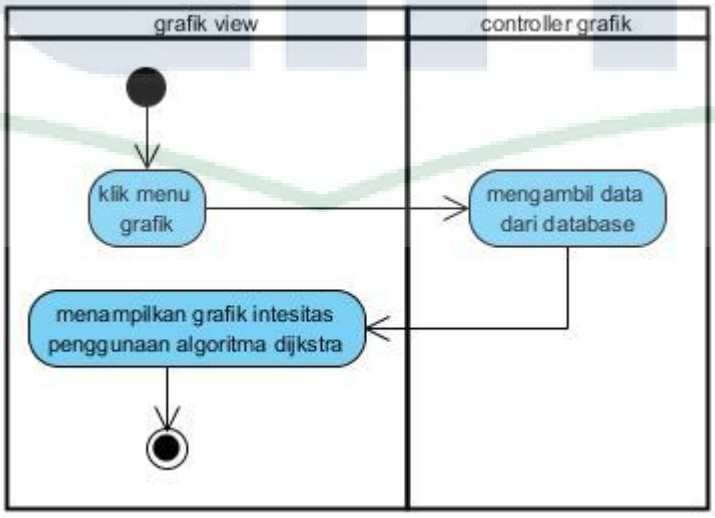
Gambar 4.19 *Activity Diagram* Menu Data Wilayah



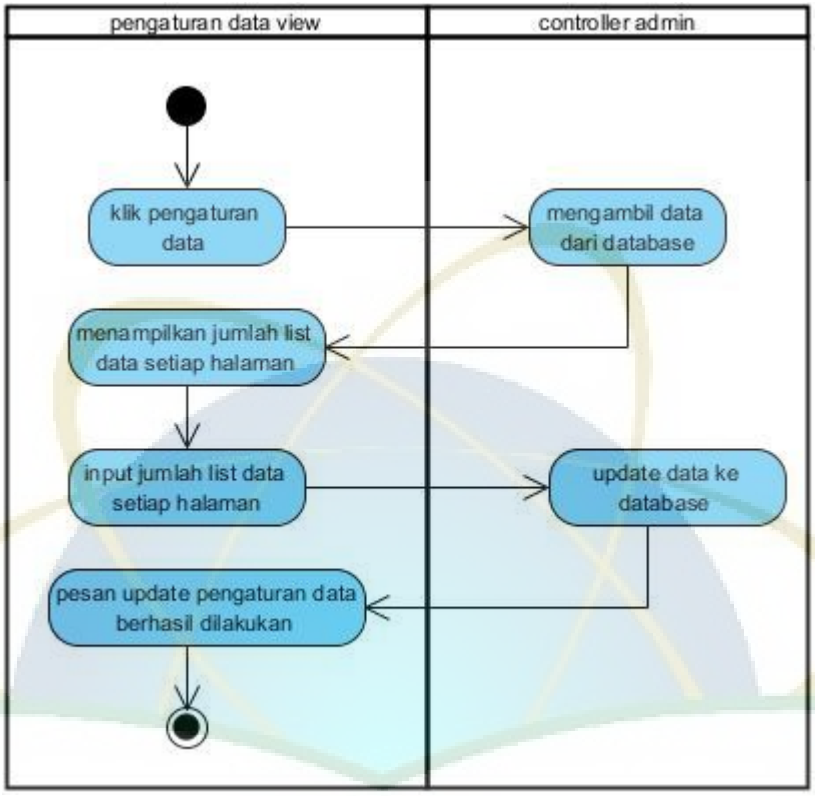
Gambar 4.20 *Activity Diagram* Menu Data Jalan



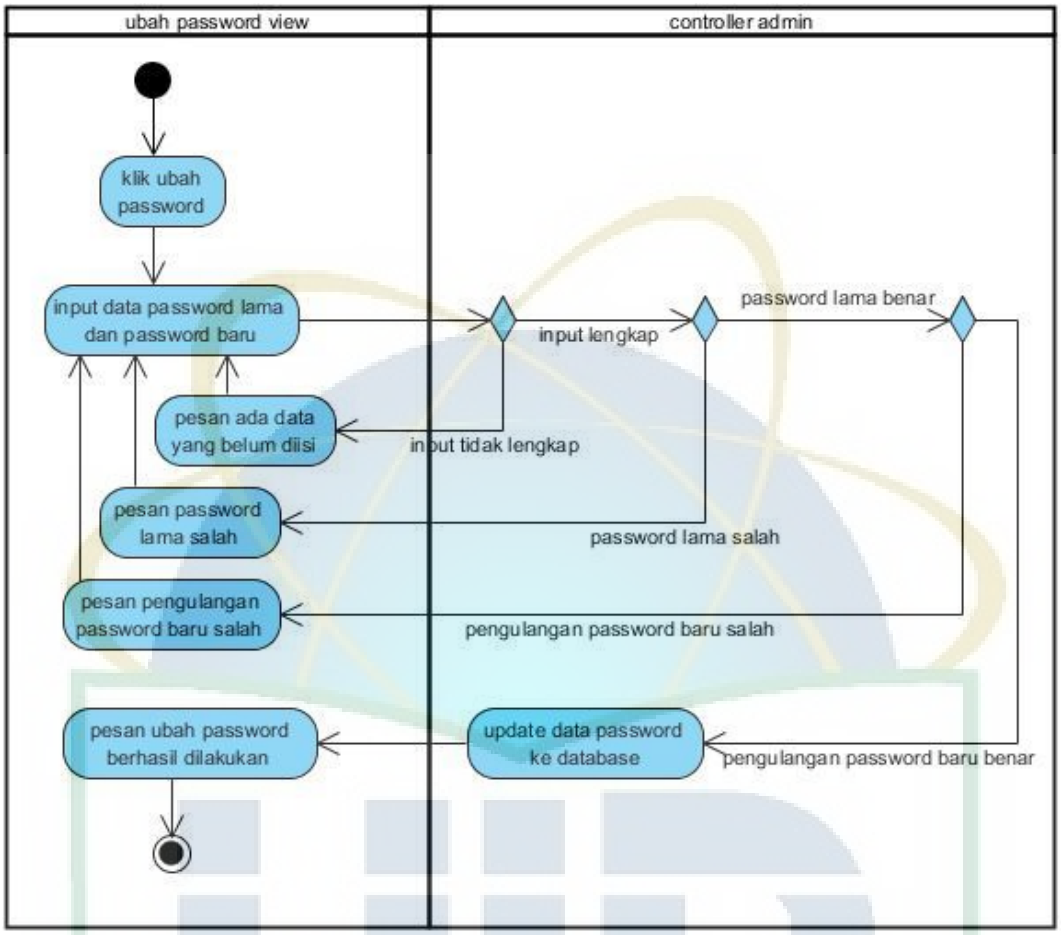
Gambar 4.21 *Activity Diagram* Menu Laporan



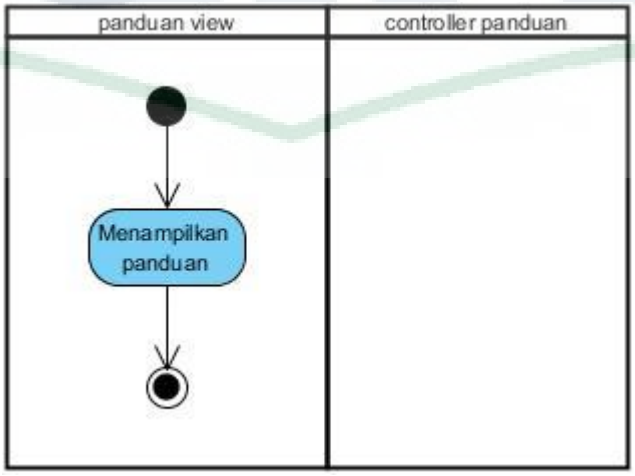
Gambar 4.22 *Activity Diagram* Menu Grafik



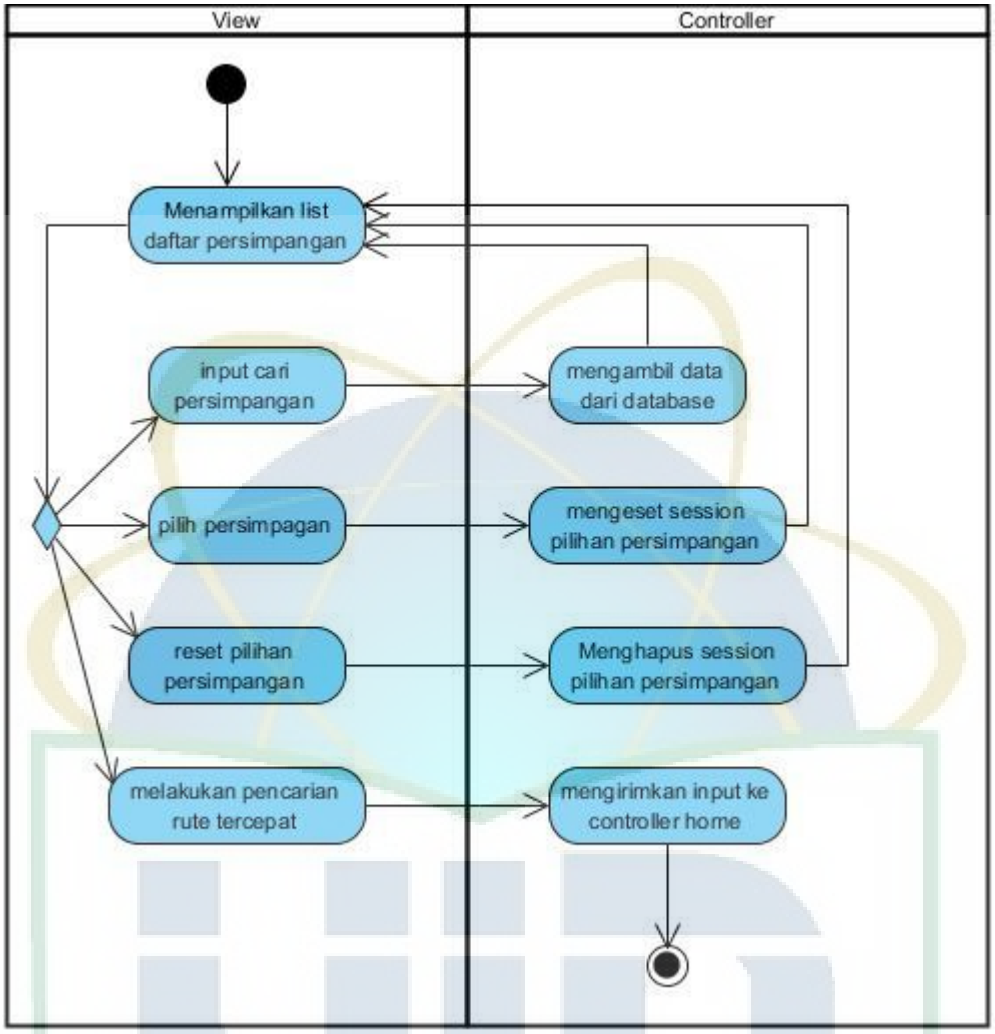
Gambar 4.23 *Activity Diagram* Menu Pengaturan Data



Gambar 4.24 *Activity Diagram* Menu Ubah Password



Gambar 4.25 *Activity Diagram* Menu Panduan



Gambar 4.26 *Activity Diagram* Menu Daftar Persimpangan

4.2.1.4. *Sequence Diagram*

Berikut ini sequence diagram dari setiap event yang terjadi didalam system yaitu:

a. *Sequence diagram* user
b. *Sequence diagram* admin
c. *Sequence diagram* node index
d. *Sequence diagram* insert data node
e. *Sequence diagram update* data node
f. *Sequence diagram delete* data node
g. *Sequence diagram search* data node
h. *Sequence diagram* path *index*
i. *Sequence diagram insert* data path
j. *Sequence diagram update* data path
k. *Sequence diagram delete* data path
l. *Sequence diagram search* data path
m. *Sequence diagram* wilayah index
n. *Sequence diagram insert* data wilayah
o. *Sequence diagram update* data wilayah
p. *Sequence diagram delete* data wilayah
q. *Sequence diagram search* data wilayah
r. *Sequence diagram* jalan index
s. *Sequence diagram insert* data jalan
t. *Sequence diagram update* data jalan

u. *Sequence diagram delete* data jalan

v. *Sequence diagram search* data jalan

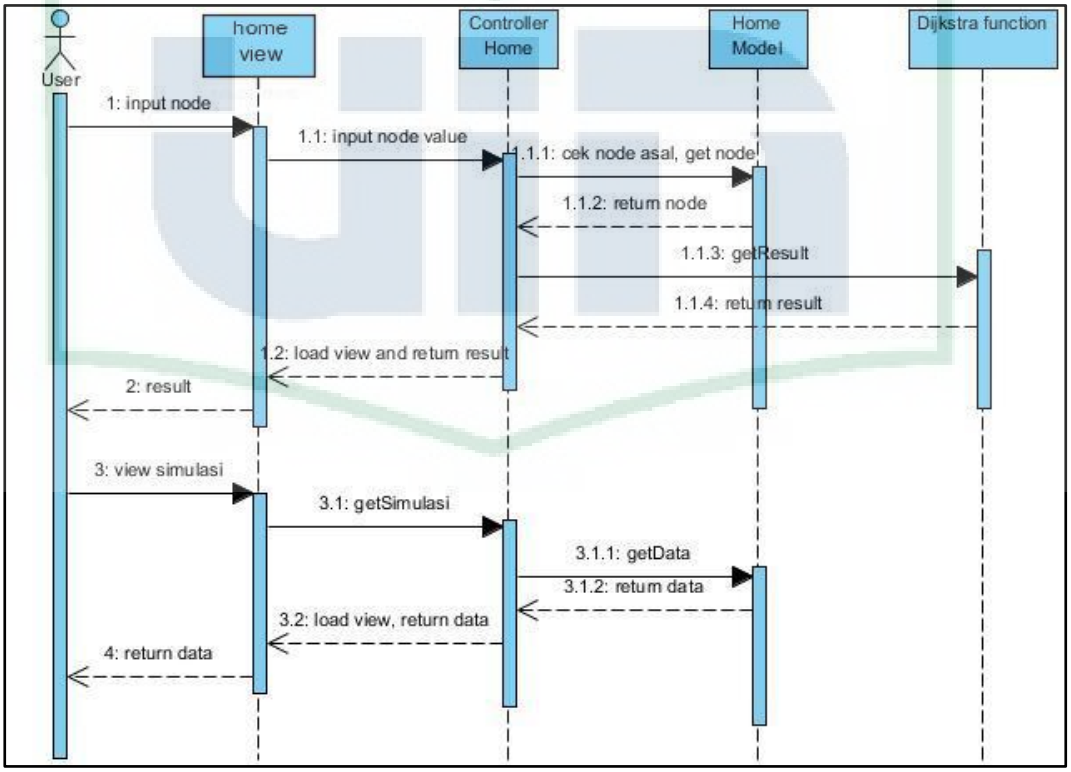
w. *Sequence diagram* laporan

x. *Sequence diagram* grafik

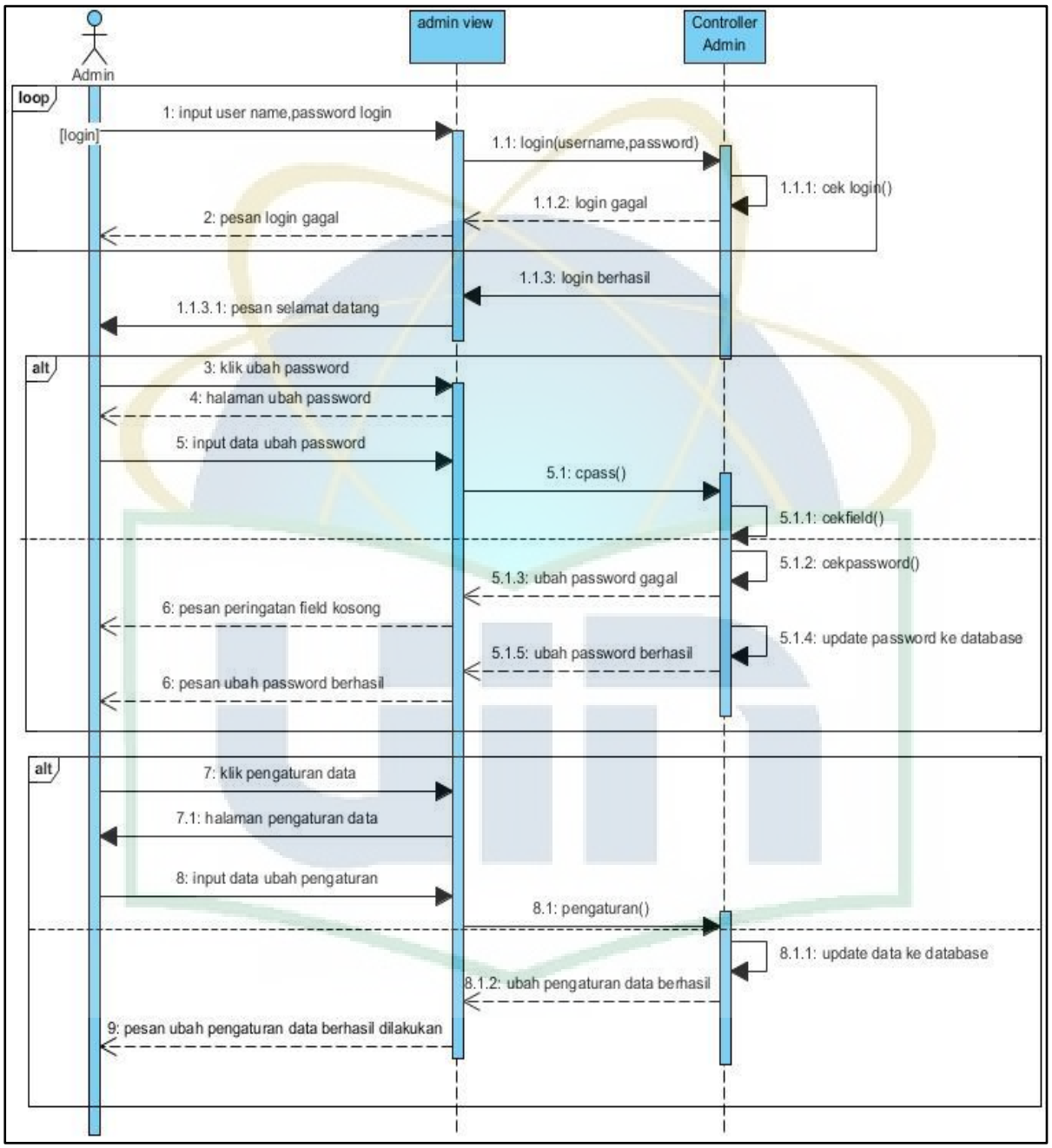
y. *Sequence diagram* panduan

z. *Sequence diagram* daftar persimpangan

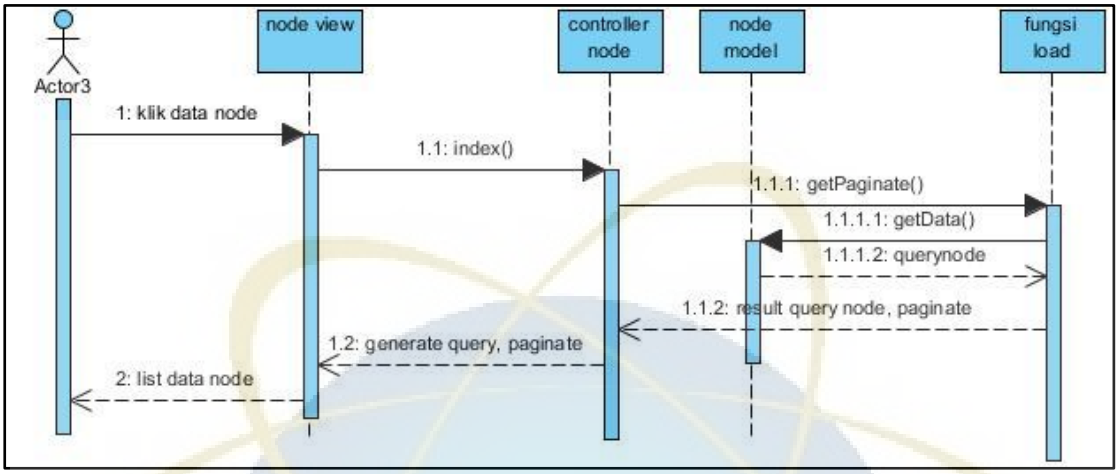
Berikut ini gambaran secara rinci grafik tersebut



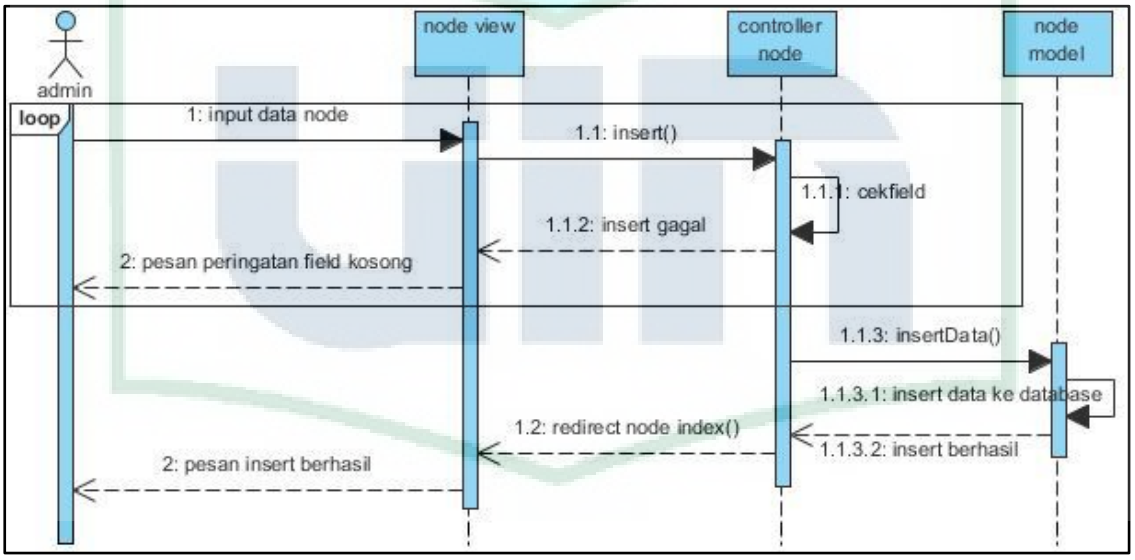
Gambar 4.27 *Sequence Diagram* User



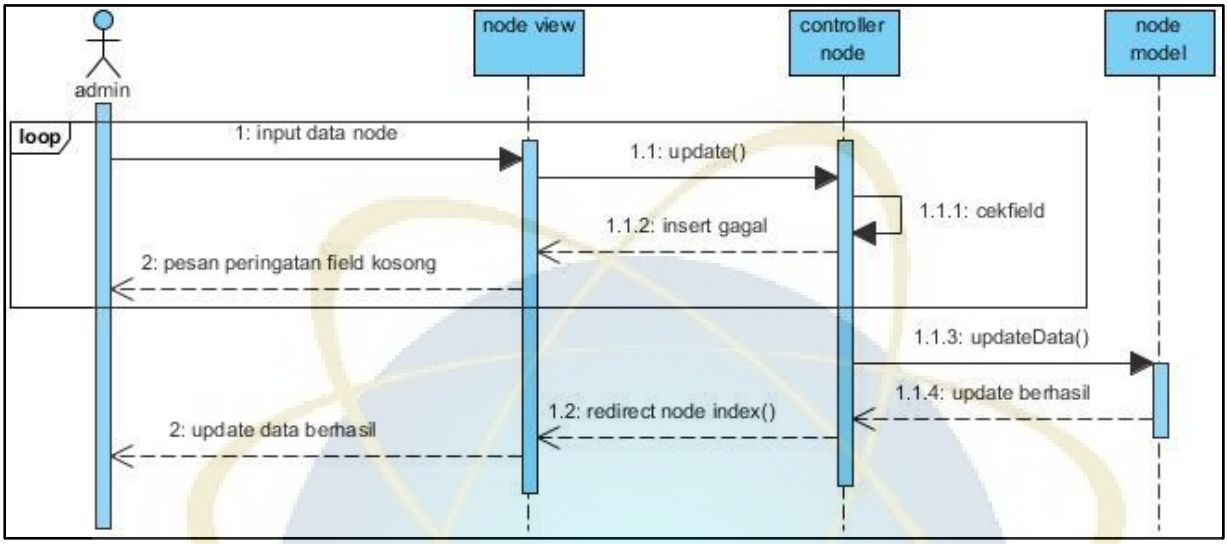
Gambar 4.28 *Sequence Diagram* Admin



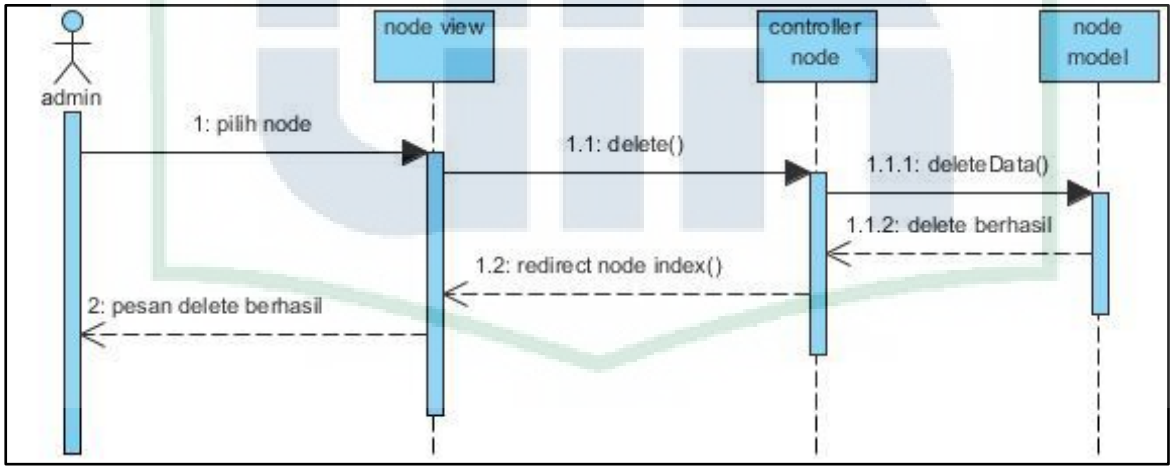
Gambar 4.29 *Sequence Diagram* Node Index



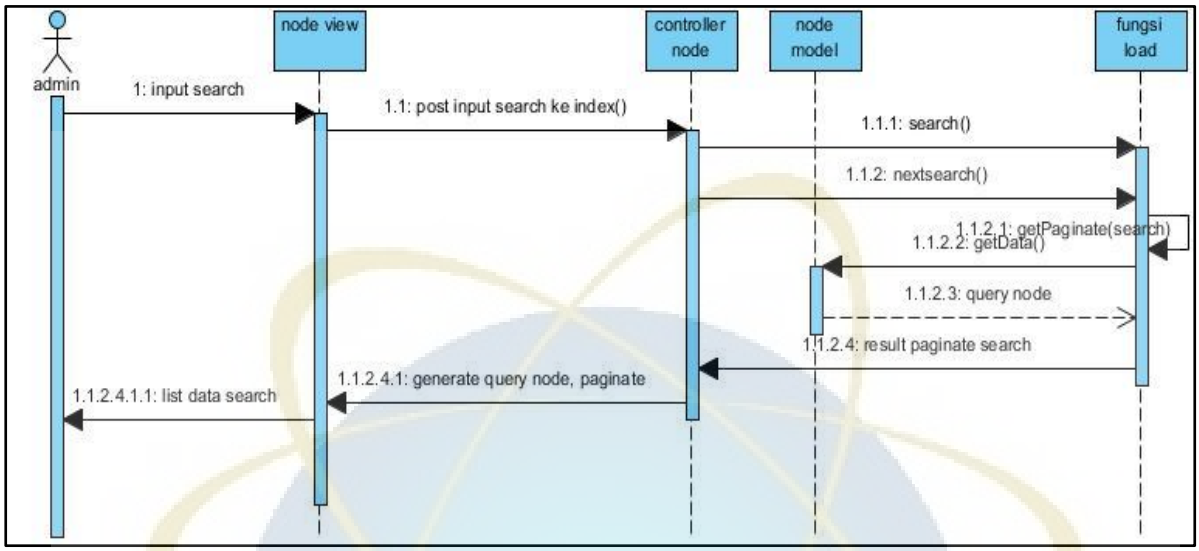
Gambar 4.30 *Sequence Diagram* Insert Data Node



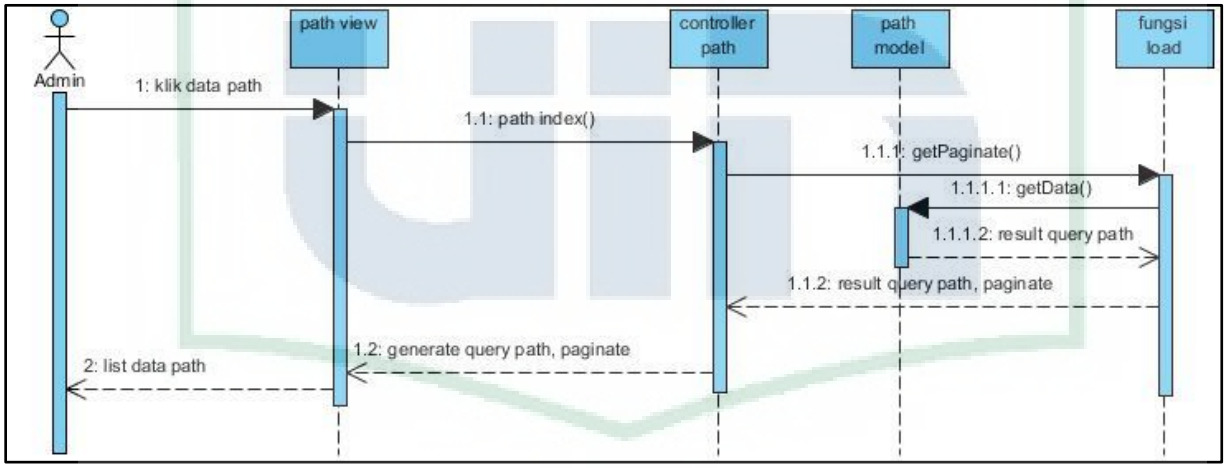
Gambar 4.31 *Sequence Diagram* Update Data Node



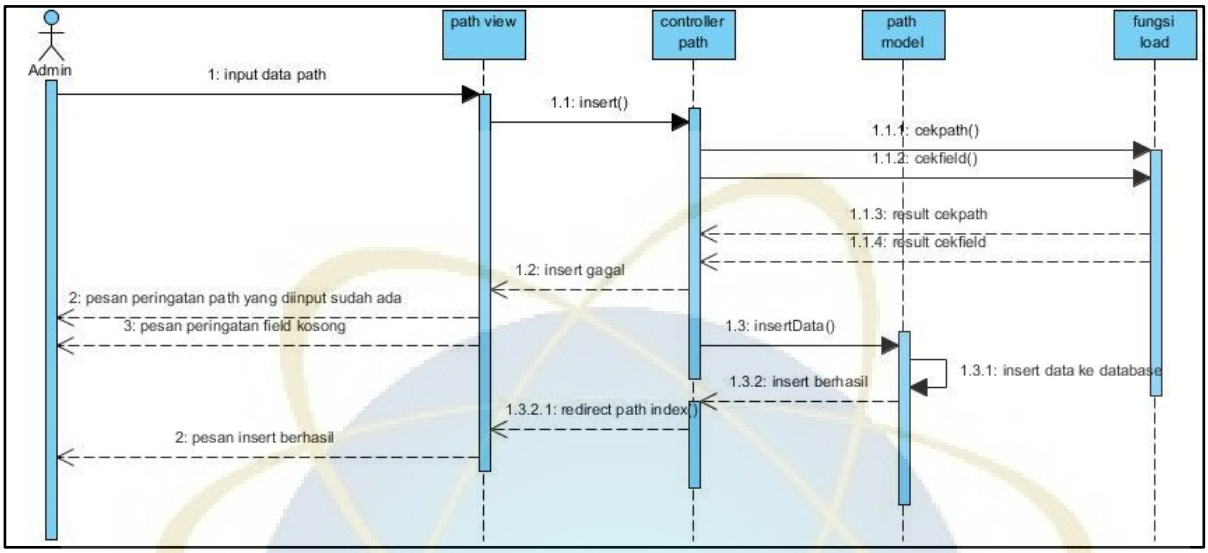
Gambar 4.32 *Sequence Diagram Delete* Data Node



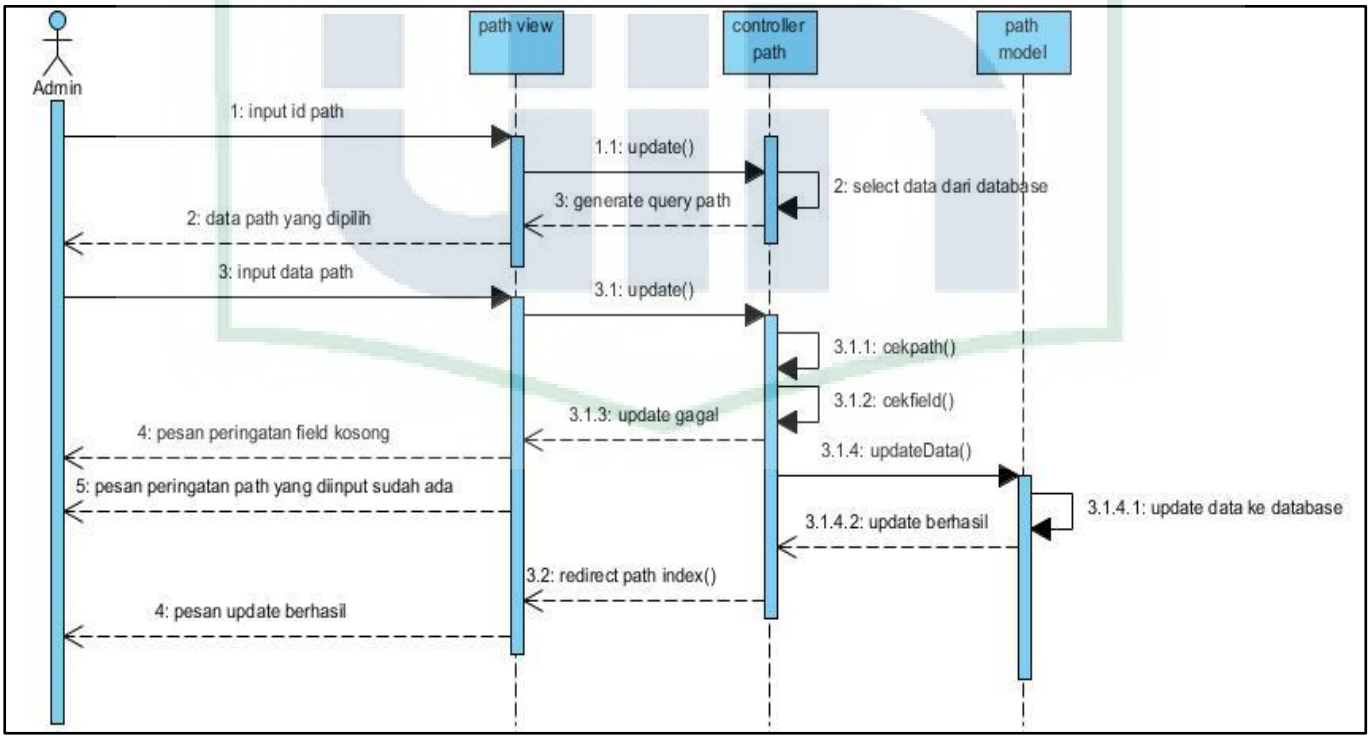
Gambar 4.33 *Sequence Diagram Search* Data Node



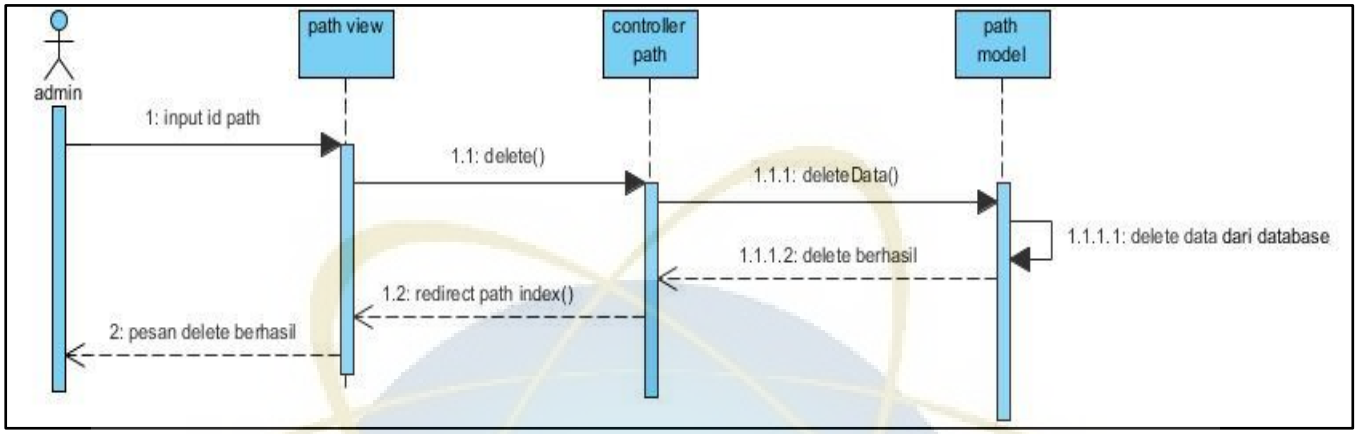
Gambar 4.34 *Sequence Diagram* Path Index



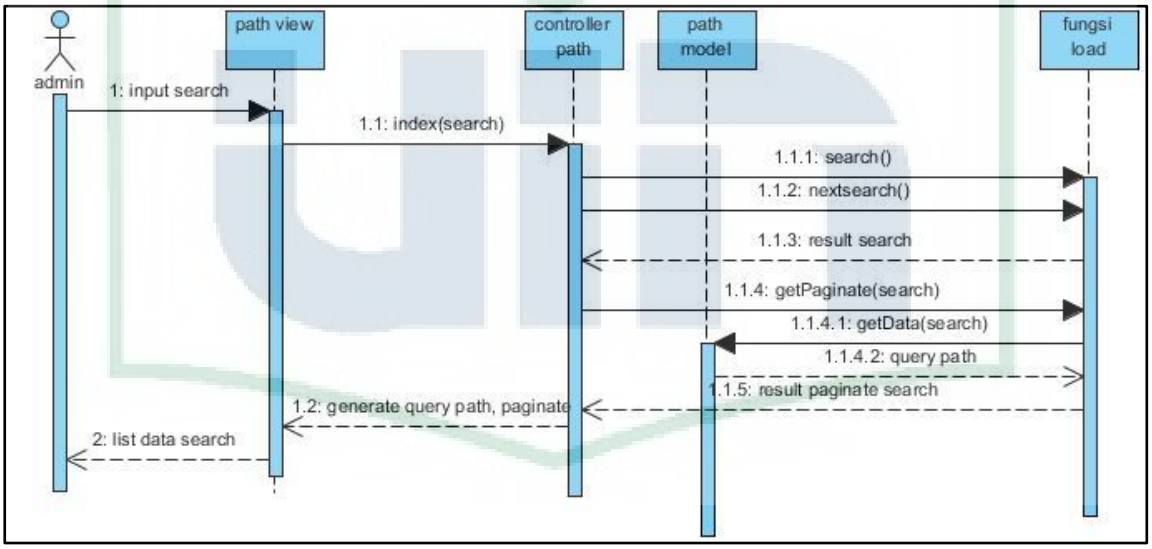
Gambar 4.35 *Sequence Diagram Insert* Data Path



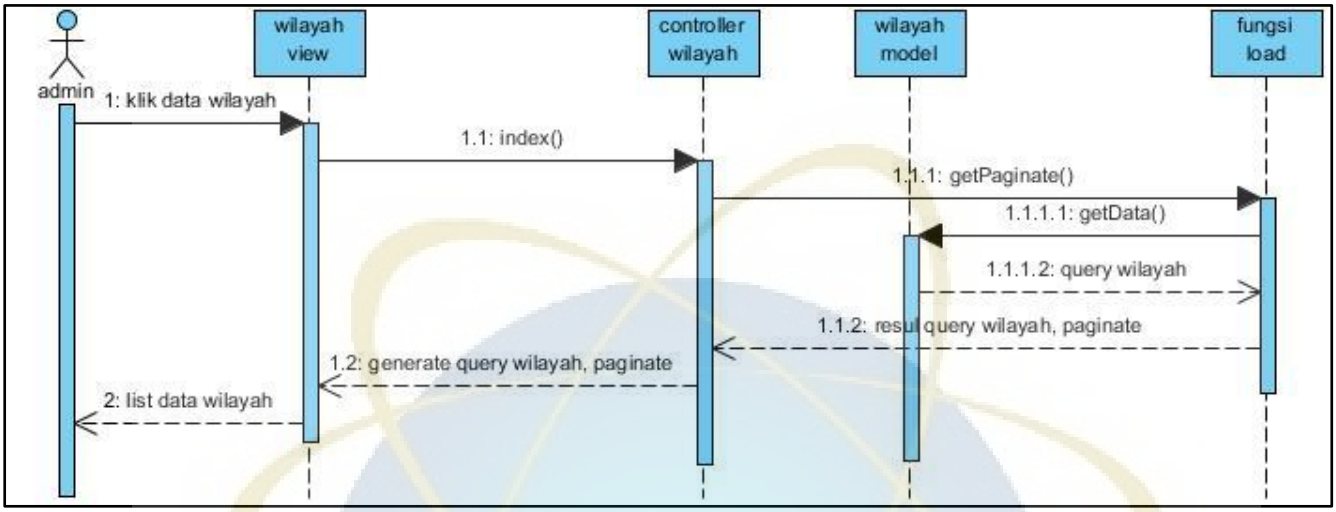
Gambar 4.36 *Sequence Diagram Update* Data Path



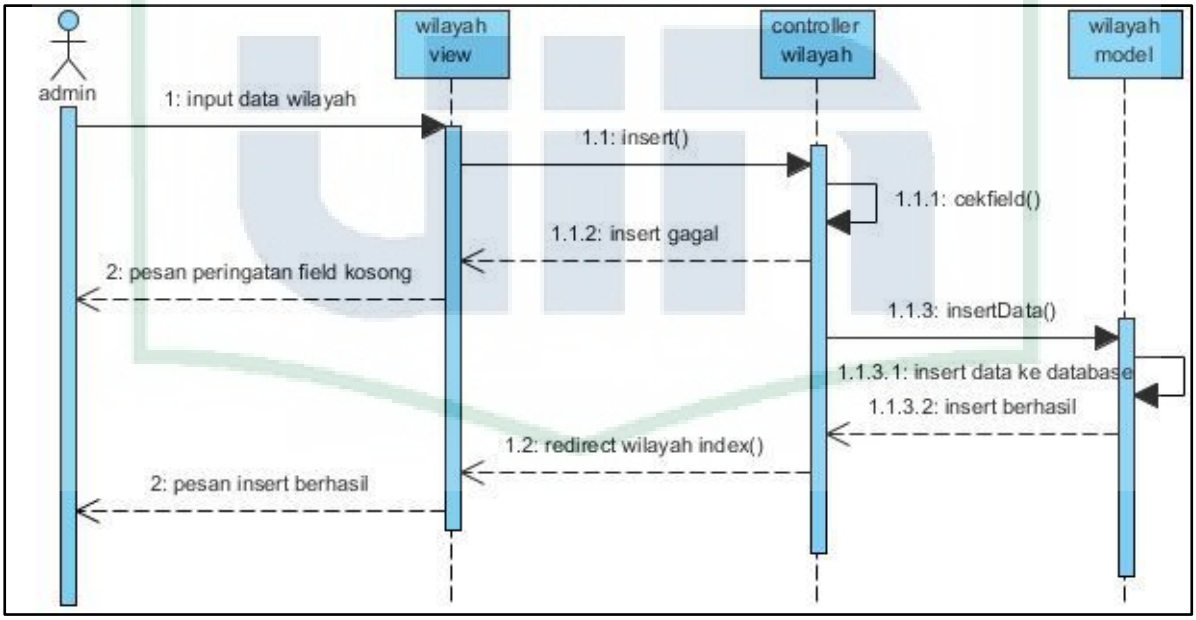
Gambar 4.37 *Sequence Diagram Delete* Data Path



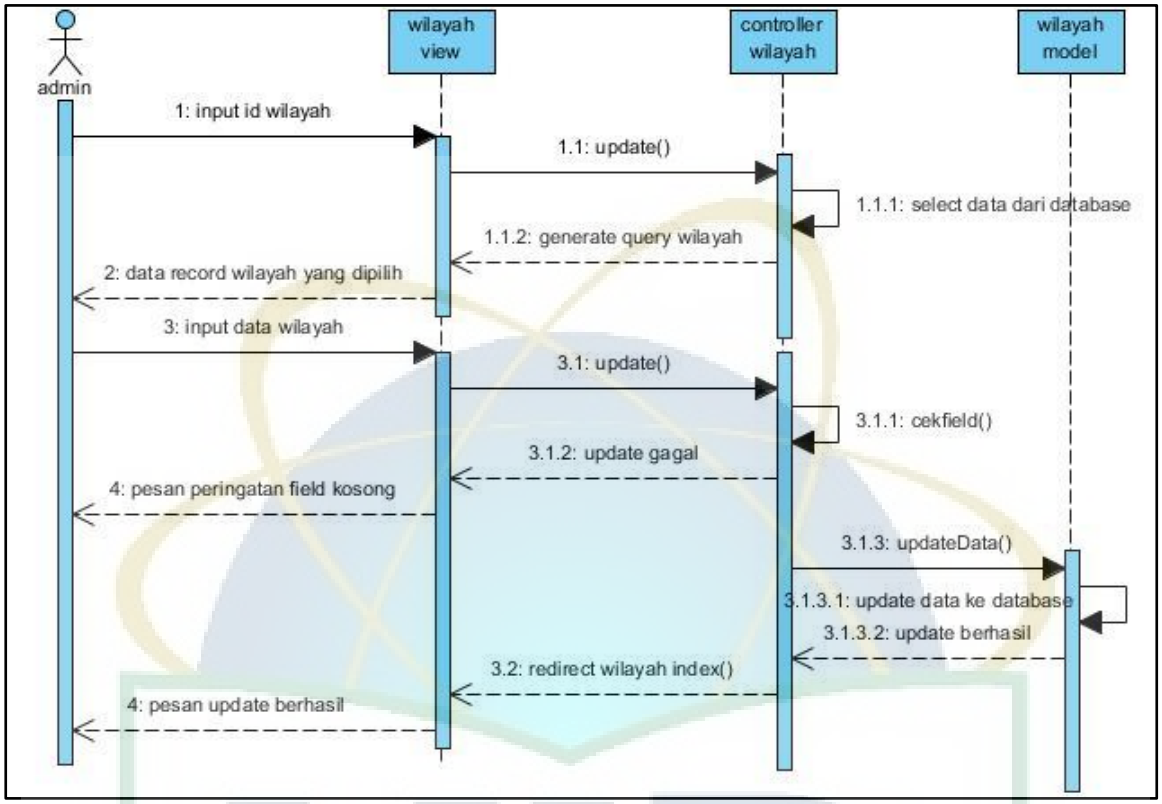
Gambar 4.38 *Sequence Diagram Search* Data Path



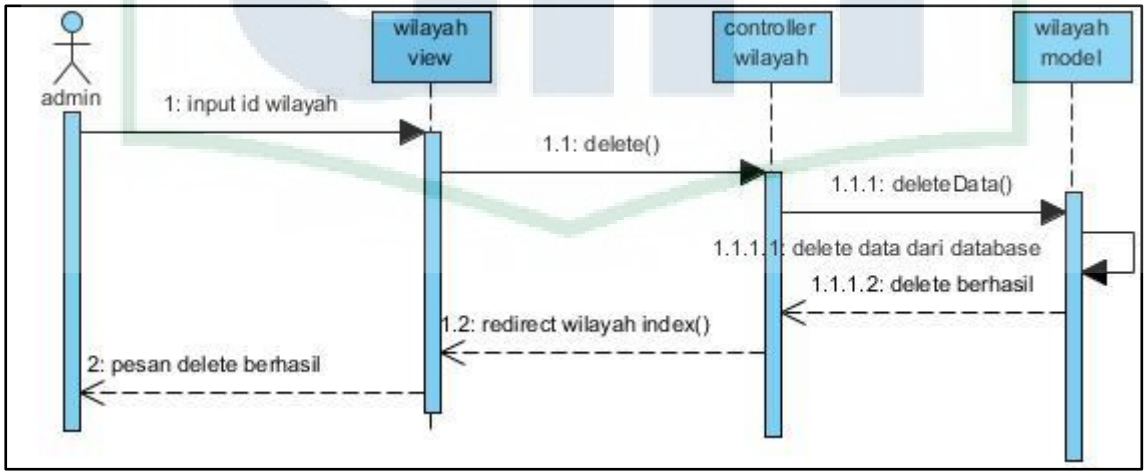
Gambar 4.39 *Sequence Diagram* Wilayah Index



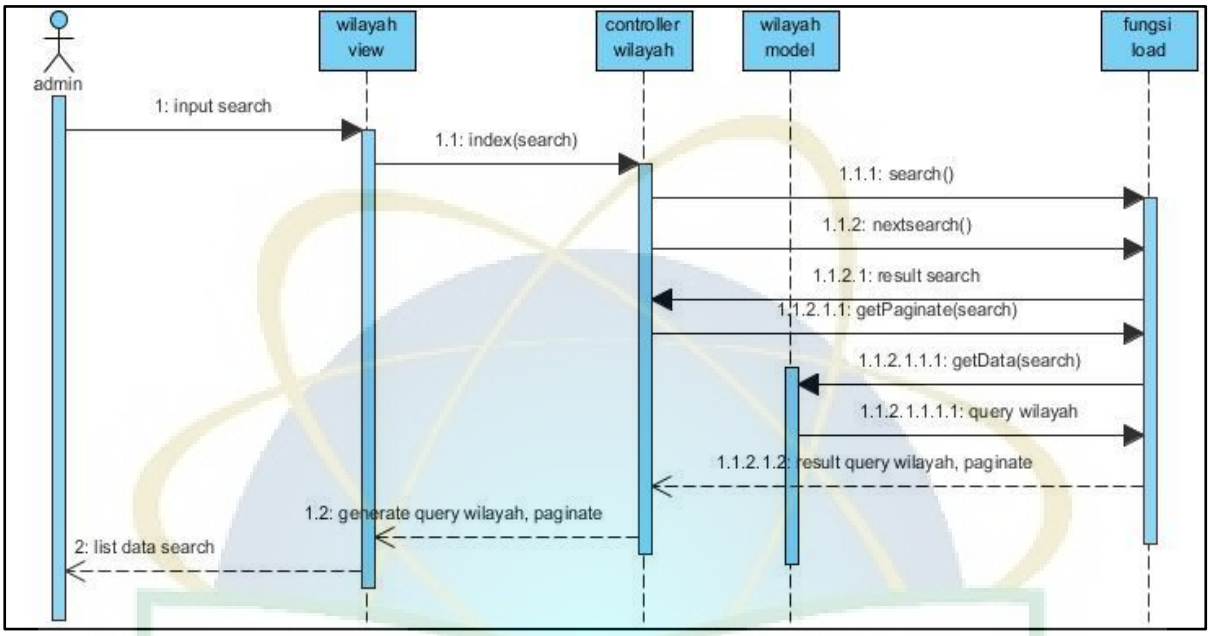
Gambar 4.40 *Sequence Diagram Insert* Data Wilayah



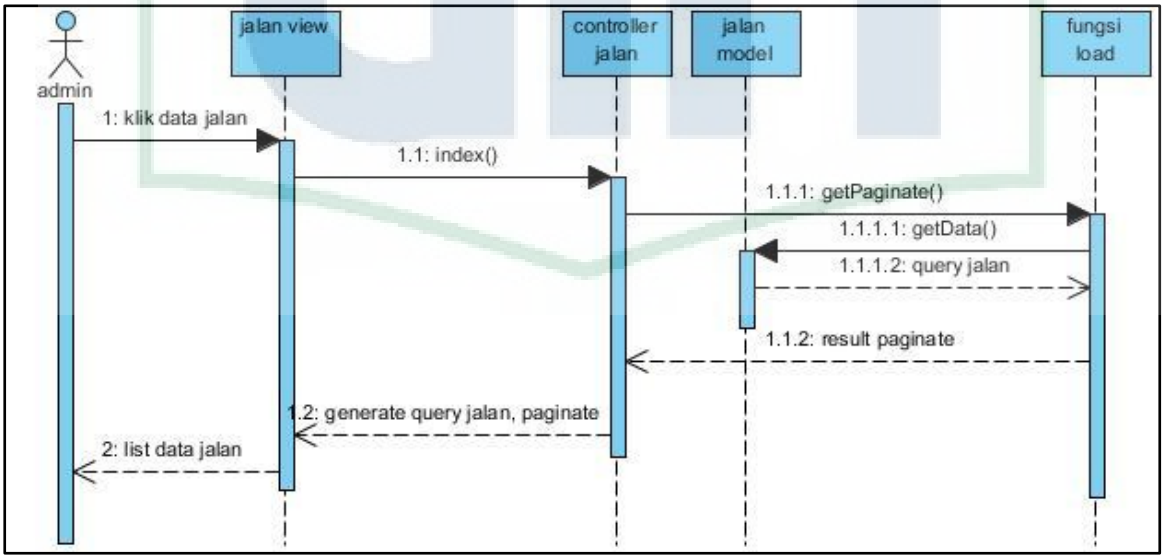
Gambar 4.41 *Sequence Diagram Update* Data Wilayah



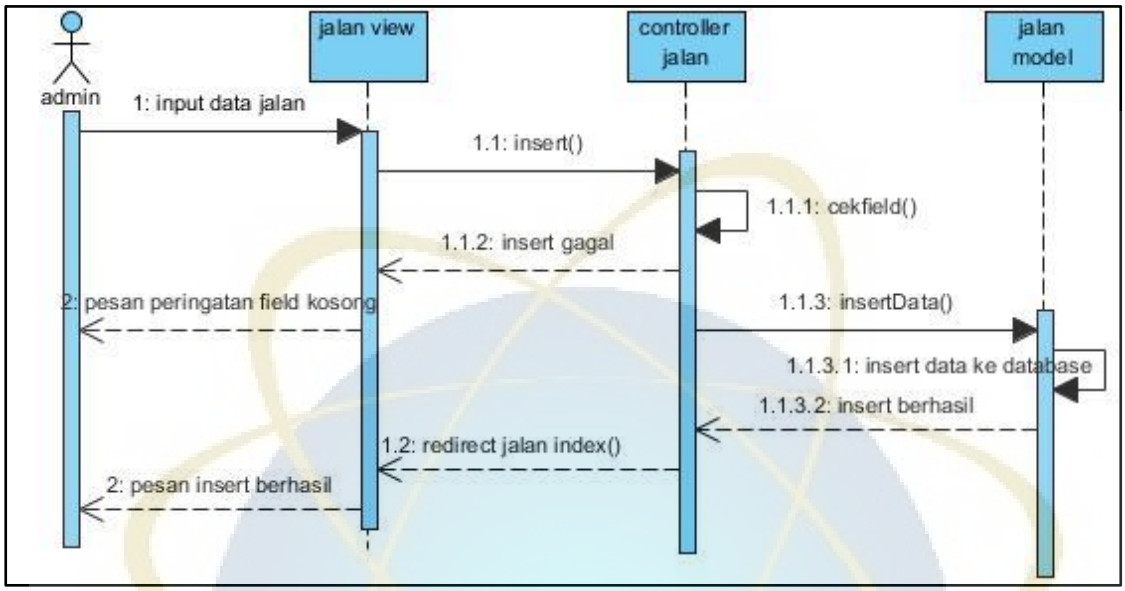
Gambar 4.42 *Sequence Diagram Delete* Data Wilayah



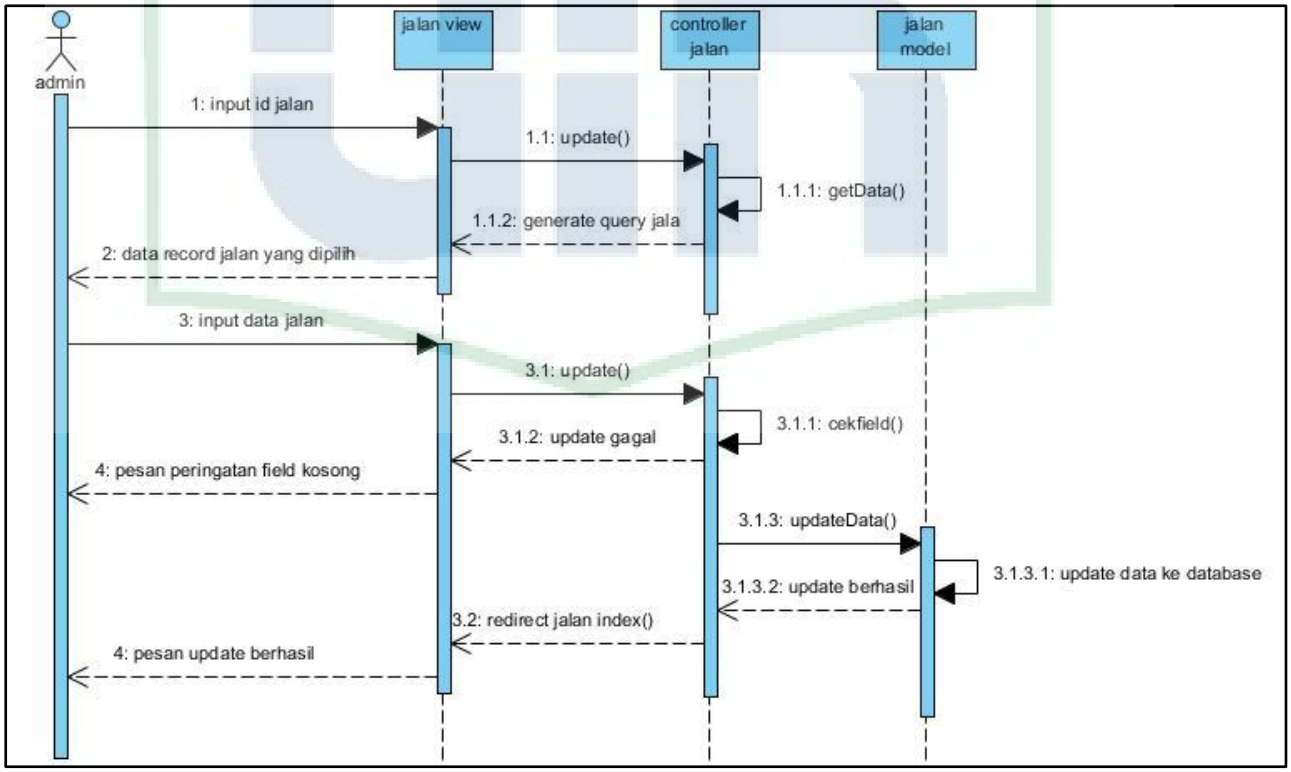
Gambar 4.43 *Sequence Diagram Search* Data Wilayah



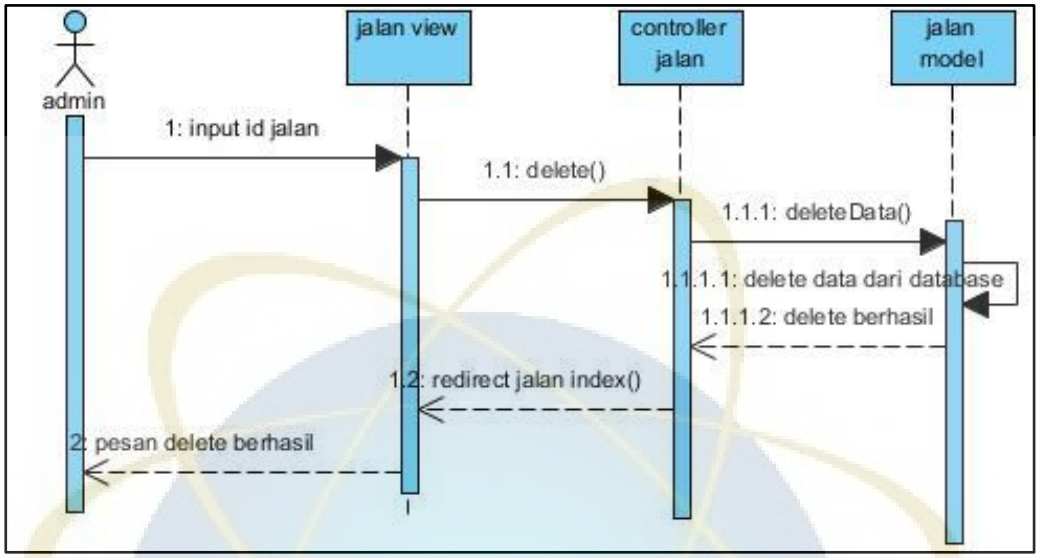
Gambar 4.44 *Sequence Diagram* Jalan Index



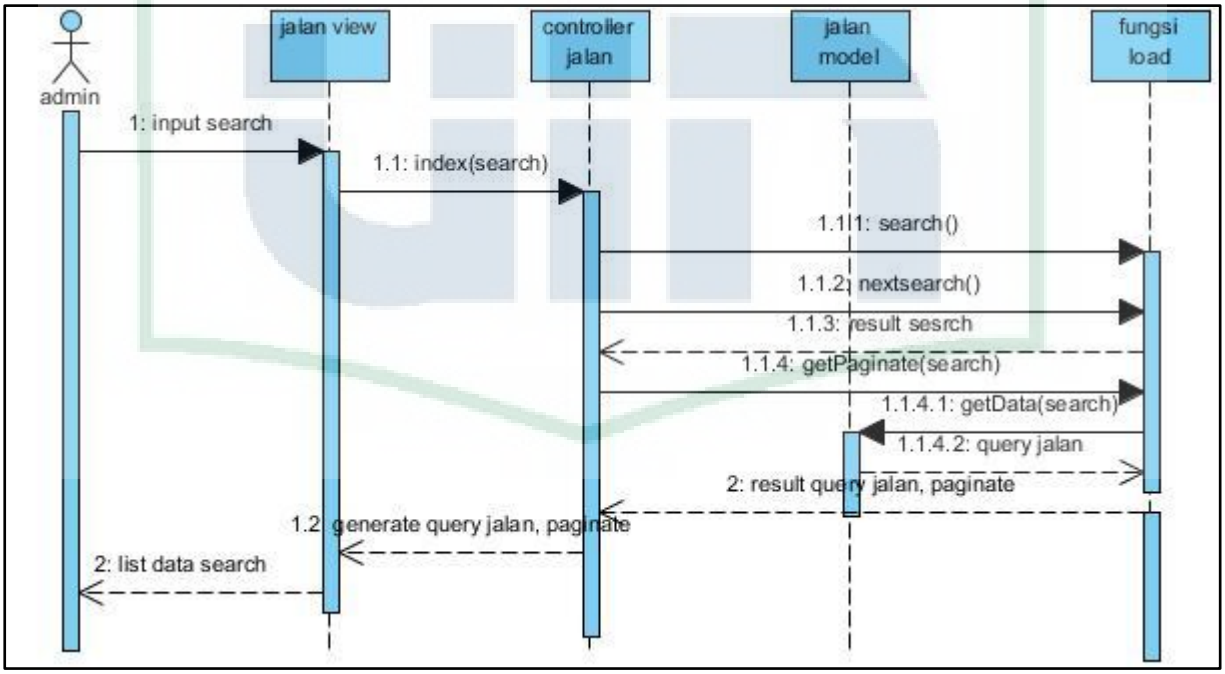
Gambar 4.45 *Sequence Diagram Insert* Data Jalan



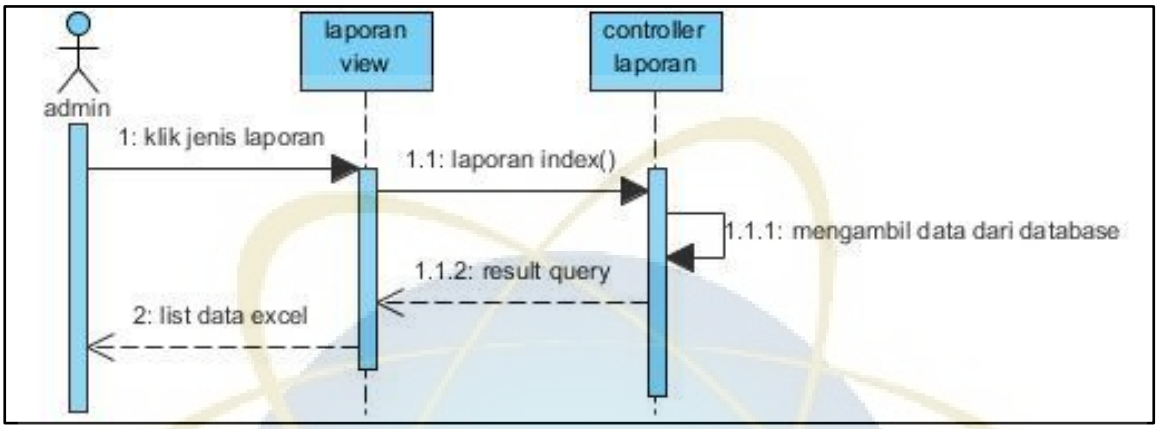
Gambar 4.46 *Sequence Diagram Update* Data Jalan



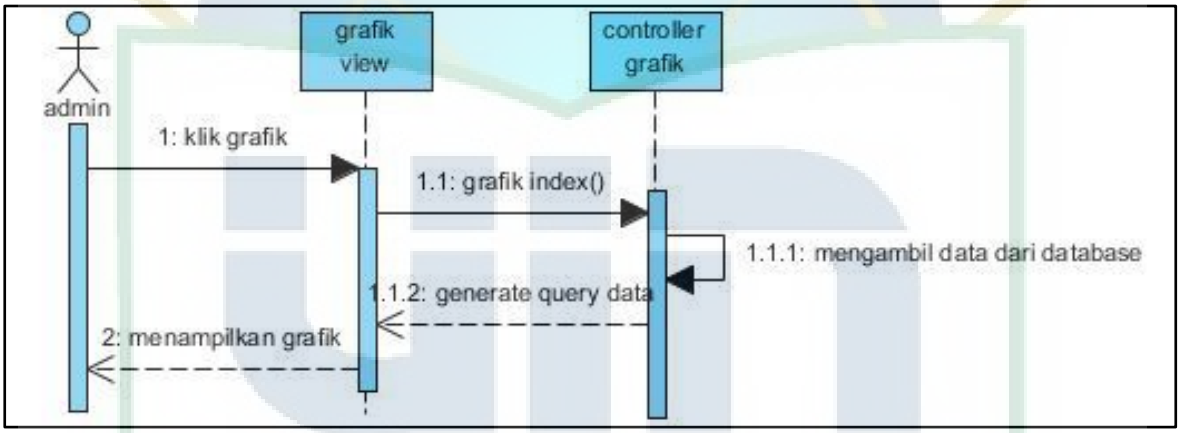
Gambar 4.47 *Sequence Diagram Delete* Data Jalan



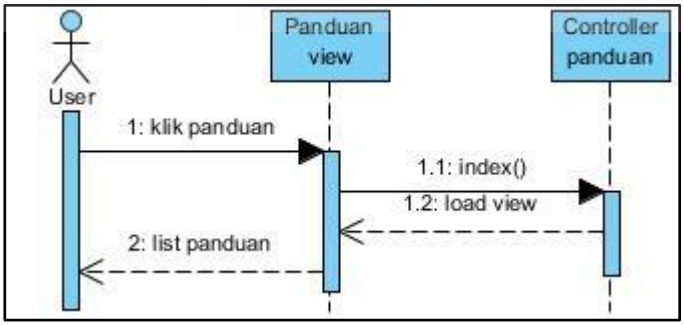
Gambar 4.48 *Sequence Diagram Search* Data Jalan



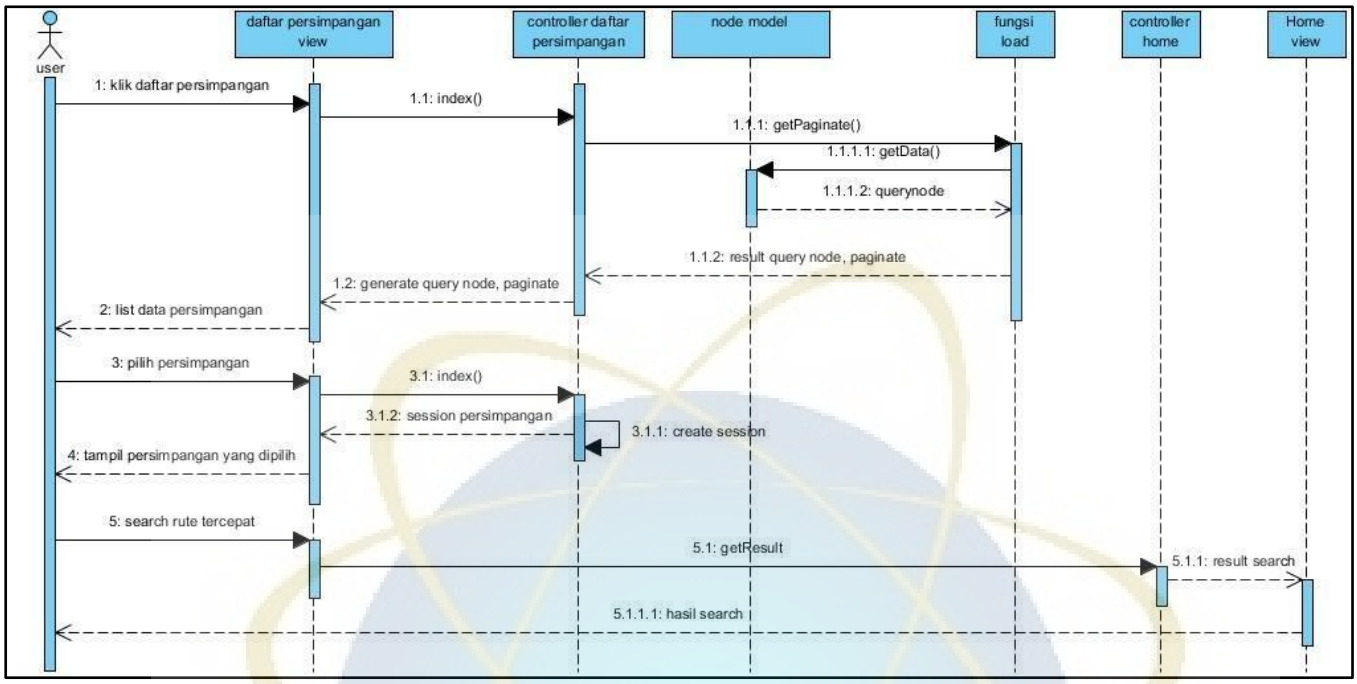
Gambar 4.49 *Sequence Diagram* Laporan



Gambar 4.50 *Sequence Diagram* Grafik



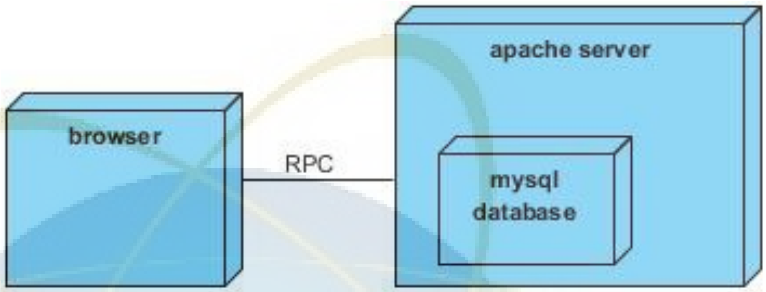
Gambar 4.51 *Sequence Diagram* Panduan



Gambar 4.52 *Sequence Diagram* Daftar Persimpangan

4.2.1.5. *Deploymen Diagram*

Berikut ini deployment diagram yang penulis gunakan pada penelitian ini



Gambar 4.53 Deployment Diagram Sistem

4.2.1.6. Pemodelan Database

Tabel yang penulis gunakan dalam penelitian ini yaitu tabel node, tabel path, tabel pengukuran, tabel wilayah, tabel jalan, tabel map, tabel recordpath dan tabel user. Berikut ini uraian secara rinci dari setiap tabel yang digunakan pada penelitian ini.

1. Tabel Node

   Tabel node merupakan tabel yang digunakan untuk menyimpan semua data titik persimpangan yang terdapat pada sistem. Tabel node berelasi *one to many* dengan tabel wilayah, dimana setiap persimpangan(node) memiliki wilayah.

Tabel 4.9 Tabel Node

Nama field	Tipe (panjang)	Key

Idnode	Int(11)	Primary key
Namanode	Varchar(4)	-
Persimpangan	Varchar(40)	-
Jenispersimpangan	Int(11)	-
Idwilayah	Int(11)	Foreign key

2. Tabel Path

Tabel path berisi data rute dari suatu persimpangan ke persimpangan lainnya. Table path berelasi *many to one* dengan tabel node dan tabel jalan. Dimana setiap path memiliki idnode dan idjalan.

Tabel 4.10 Tabel Path

Nama field	Tipe (panjang)	Key
Idpath	Int(11)	Primary key
Namapath	Varchar(4)	-
Idnode1	Int(11)	Foreign key
Idnode2	Int(11)	Foreign key
Jarak	Int(11)	-
Idjalan	Int(11)	Foreign key

3. Tabel Wilayah

Tabel wilayah berisi data wilayah dari persimpangan-

persimpangan yang ada pada sistem.

Tabel 4.11 Tabel Wilayah

Nama field	Tipe (panjang)	Key
Idwilayah	Int(11)	Primary key
Namawilayah	Varchar(40)	-
Jeniswilayah	Int(11)	-

4. Tabel Jalan

   Tabel jalan berisi data jalan yang dilalui oleh rute (path) yang menghubungkan suatu persimpangan ke persimpangan lainnya.

Tabel 4.12 Tabel Jalan

Nama field	Tipe (panjang)	Key
Idjalan	Int(11)	Primary key
Namajalan	Varchar(40)	-
Jenisjalan	Int(11)	-

5. Tabel Pengukuran

   Tabel pengukuran berisi data pengukuran dari setiap rute (path). Sehingga tabel ini berelasi *many to one* dengan tabel path. Adapun data pengukuran tersebut berupa data kecepatan dan waktu tempuh setiap path dalam waktu tertentu. Sesuai dengan data yang penulis

dapatkan dari Dishub DKI Jakarta, penulis membagi waktu pengukuran beserta kodenya menjadi enam jenis waktu, yaitu:

- Pagi sebelum *three in one* yaitu pukul 00.00 – 06.59 WIB, kode waktu 11
- Pagi saat *three in one* yaitu pukul 07.00 – 10.00 WIB, kode waktu 12
- Pagi setelah *three in one* yaitu pukul 10.01 – 11.59 WIB, kode waktu 13
- Sore sebelum *three in One* yaitu pukul 12.00 – 15.59 WIB, kode waktu 21
- Sore saat *three in one* yaitu pukul 16.00 – 20.00 WIB, kode waktu 22
- Sore setelah *three ini one* yaitu pukul 20.01 – 23.59 WIB, kode waktu 23

Tabel 4.13 Tabel Pengukuran

Nama field	Tipe (panjang)	Key
Idpengukuran	Int(11)	Primary key
Idpath	Int(11)	Foreign key
Waktupengukuran	Int(11)	-
Waktutempuh	Int(11)	-
Kecepatan	Int(11)	-

6. Tabel RecordPath

Tabel ini berisi data intensitas rute yang diinput oleh user kedalam sistem. Tabel ini digunakan untuk menampilkan data pada bagian menu grafik.

Tabel 4.14 Tabel Recordpath

Nama field	Tipe (panjang)	Key
Idrecordpath	Int(11)	Primary key
Idnode1	Int(11)	Foreign key
Idnode2	Int(11)	Foreign key
Akses	Int(11)	-

7. Tabel Map

Tabel map berisi nama map atau gambar yang digunakan untuk menampilkan simulasi hasil pencarian rute tercepat dan terpendek.

Tabel 4.15 Tabel Map

Nama field	Tipe (panjang)	Key
Idmap	Int(11)	Primary key
Idnode1	Int(11)	Foreign key
Idnode2	Int(11)	Foreign key
Namamap	Varchar(5)	-

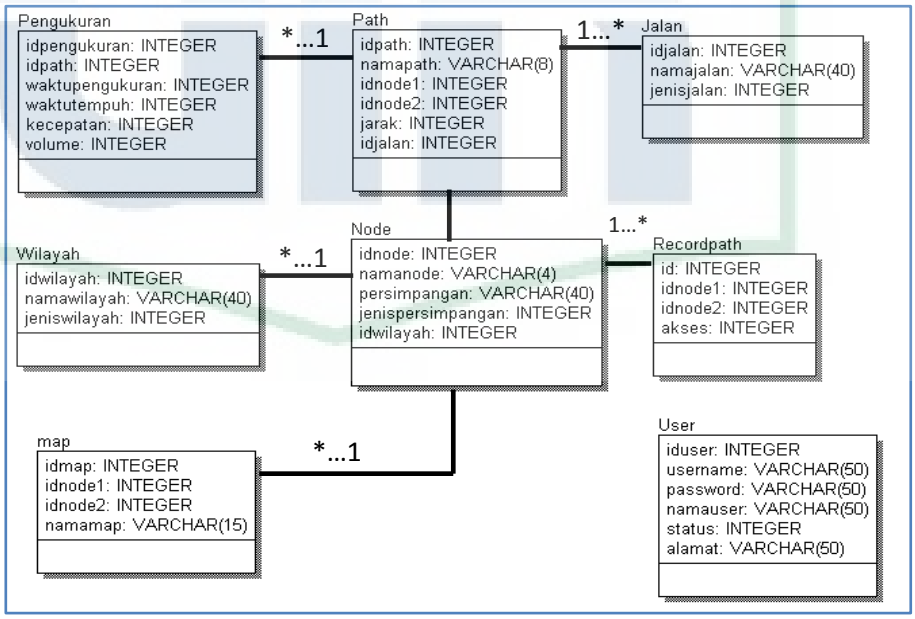
8. Tabel User

Tabel user merupakan tabel yang berisi data pengguna sistem yang dapat melakukan login, dalam hal ini pengguna sistem yang dimaksud yaitu admin.

Tabel. 4.16 Tabel User

Nama field	Tipe (panjang)	Key
Iduser	Int(11)	Primary key
Username	Varchar(45)	-
Password	Varchar(15)	-
Namauser	Int(11)	-

Dari tabel yang telah penulis uraikan, berikut ini relasi dari setiap tabel yang ada pada sistem.

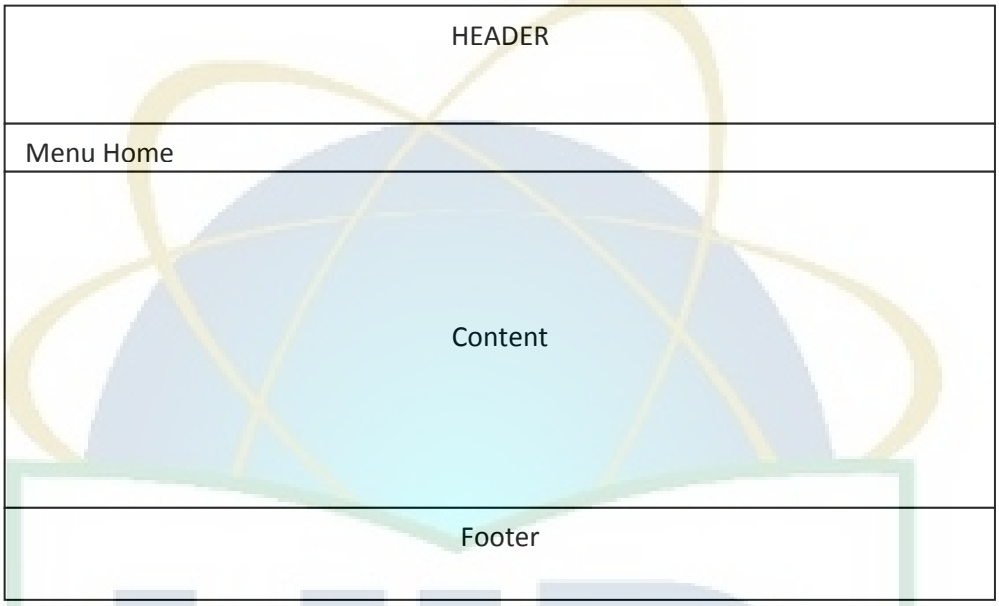


Gambar 4.54 Relasi Tabel-Tabel Yang Digunakan Pada Sistem

4.2.1.7. Pemodelan *Interface*

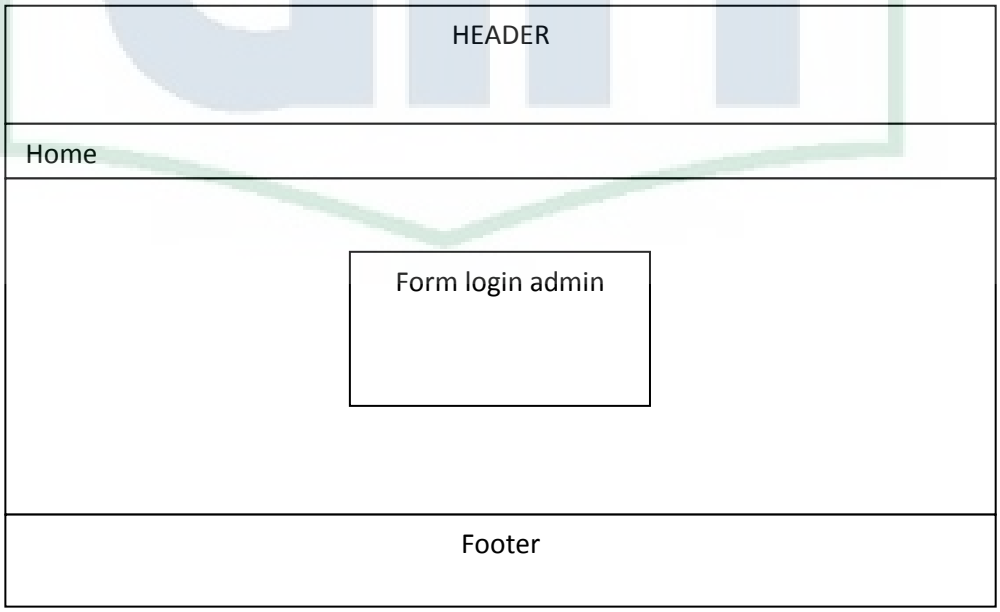
Berikut ini perancangan *layout interface* dalam penelitian ini

1. *Layout* halaman user



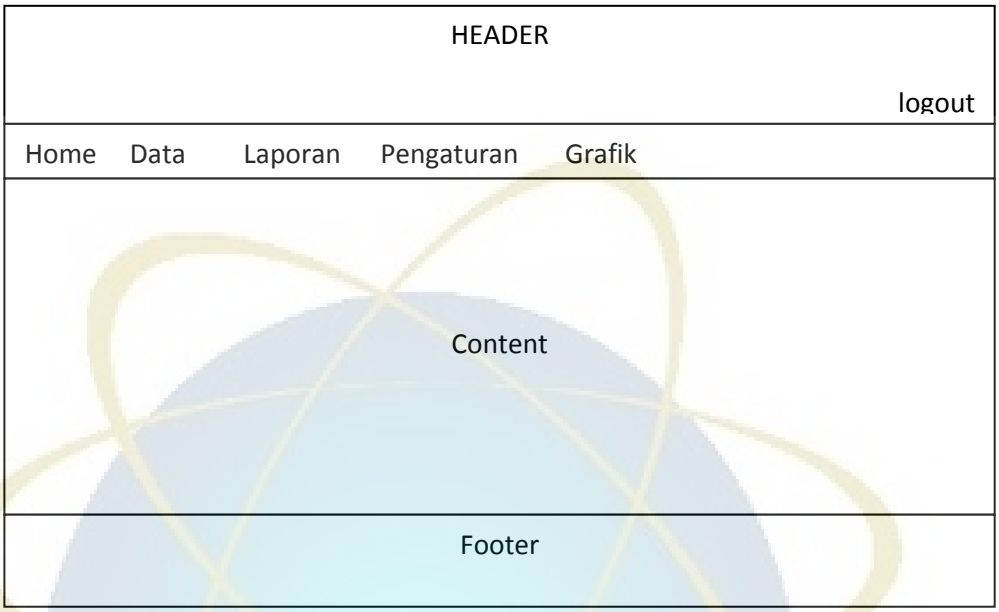
Gambar 4.55 Perancangan *Layout* Halaman User

2. *Layout* halaman login admin



Gambar 4.56 Perancangan *Layout* Halaman Login Admin

3. *Layout* halaman admin



Gambar 4.57 Perancangan *Layout* Halaman Admin

4.2.2. *Construction*

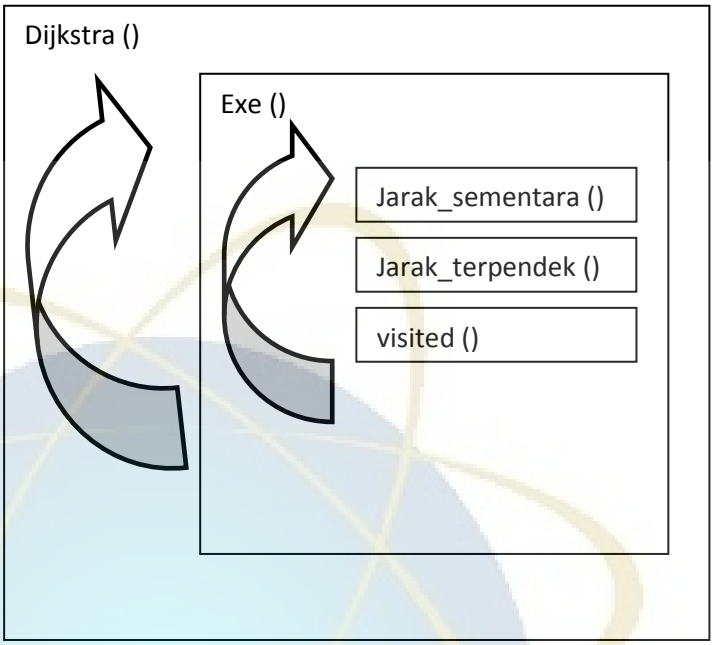
4.2.2.1. Coding

Dalam pembuatan *prototype* algoritma Dijkstra, penulis mengacu pada *pseudocode* algoritma Dijkstra yang sudah dijelaskan pada bab sebelumnya. Dalam hal ini penulis membuat fungsi-fungsi untuk melakukan setiap langkah pada algoritma Dijkstra. Penulis membuat lima buah fungsi yang terdiri dari tiga fungsi utama dan dua fungsi tambahan. Fungsi tambahan digunakan untuk melakukan iterasi pada fungsi utama. Berikut ini fungsi yang penulis gunakan pada untuk menjalankan algoritma Dijkstra.

1. Fungsi jarak sementara
2. Fungsi jarak terpendek
3. Fungsi visited
4. Fungsi exe
5. Fungsi dijkstra

Fungsi yang pertama kali dijalankan adalah fungsi dijkstra, fungsi ini menerima input masukan berupa sebuah graf dan dua buah node yaitu node asal dan node tujuan. Selanjutnya fungsi ini akan melakukan *looping* pada graf sampai node tujuan ditemukan, adapun dalam melakukan *looping* fungsi ini membentuk tiga buah variabel yang akan menjadi parameter untuk fungsi yang lainnya. Variabel tersebut yaitu jarak_sementara, jarak_terpendek dan visited.

Berikut ini gambaran penggunaan fungsi-fungsi tersebut pada jalannya algoritma Dijkstra.



Gambar 4.58 Struktur Fungsi Algoritna Dijkstra

Setelah pembuatan prototype algoritma Dijkstra, tahap selanjutnya yaitu pembuatan *class* dan *fungsi* yang akan digunakan pada sistem. Pembuatan dilakukan pada bagian *model, view* dan *controller* yang terdapat pada *frame work* CI.

Khusus pada bagian controller penulis membuat sebuah kelas general yaitu kelas fungsi load yang berisi fungsi-fungsi global yang digunakan oleh kelas-kelas lain yang terdapat dalam sistem. Kelas fungsi load didefinisikan sebagai child dari controller, sedangkan kelas-kelas yang lainnya didefinisikan sebagai child dari kelas fungsi load. Dengan kata lain kelas fungsi load berfungsi sebagai

perantara antara kelas-kelas yang dibuat dengan kelas controller. Gambaran lebih jelas terdapat pada *class diagram* yang tedapat pada bagian pemodelan UML. Adapun class yang ada pada sistem adalah sebagai berikut

Tabel 4.17 Kelas-Kelas Yang Terdapat Pada Sistem

Nama class	Kegunaan dalam sistem
Home	Sebagai kelas utama untuk user dan sebagai tempat algoritma Dijkstra
Admin	Sebagai kelas login dan kelas pengaturan seperti ubah password dan ubah pengaturan data
Node	Sebagai kelas yang menyediakan data node beserta insert, update, delete dan search
Path	Sebagai kelas yang menyediakan data path beserta insert, update, delete dan search
Wilayah	Sebagai kelas yang menyediakan data wilayah beserta insert, update, delete dan search
Jalan	Sebagai kelas yang menyediakan data jalan beserta insert, update, delete dan search
Laporan	Sebagai kelas untuk menghasilkan output berupa file excel
Grafik	Sebagai kelas untuk menampilkan grafik intensitas penggunaan algoritma Dijkstra oleh user

Daftarp	Sebagai kelas untuk menampilkan list daftar persimpangan yang terdapat pada system
Panduan	Sebagai kelas untuk menampilkan panduan penggunaan sistem

4.2.2.2.Testing

Pada penelitian ini penulis menggunakan metode pengujian *black box testing*. Dalam hal ini penulis melakukan pengujian pada setiap event yang terdapat pada sistem beserta segala kemungkinannya. Adapun yang melakukan pengujian yaitu perwakilan dari Dishub DKI Jakarta dan user pengguna. Berikut ini rincian dari pengujian yang telah dilakukan. Adapun untuk hasil *interface* pengujian terdapat pada lampiran 1.

1. Halaman home

Tabel 4.18 Tabel Testing Halaman Home

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses url sistem (http://localhost/testing/)	Tampil halaman home	OK
Mengklik menu home	Tampil halaman home	OK
Mengklik tempat asal	Tampil semua persimpangan yang ada	OK

	pada database	
Mengklik tombol search tanpa mengisi tempat asal dan tujuan	Redirect ke halaman home	OK
Mengklik tombol search dengan input tempat asal sama dengan tempat tujuan	Redirect halaman home	Ok
Mengklik tombol search dengan input tempat asal berbeda dengan tempat tujuan	Algoritma Dijkstra dijalankan dan menghasilkan output	OK
Kondisi output saat rute untuk tempat tujuan tidak ditemukan	Menampilkan pesan rute dari tempat asal ke tempat tujuan tidak ditemukan	OK
Kondisi output saat rute untuk tempat tujuan ditemukan	Menampilkan rute terpendek dan rute tercepat beserta jarak dan waktu tempuh yang diperlukan dan menampilkan tombol untuk melihat simulasi	OK
Mengklik tombol search	Algoritma Dijkstra	OK

dengan input tempat asal berbeda dengan tempat tujuan dan menyertakan pilihan waktu	dijalankan dengan menggunakan data sesuai waktu yang dipilih dan menghasilkan output	
Mengklik tombol view simulasi yang ada pada hasil output algoritma Dijkstra	Menghasilkan tampilan simulasi sesuai dengan rute yang dihasilkan dari algoritma Dijkstra	OK

2. Halaman simulasi

Tabel 4.19 Tabel Testing Halaman Simulasi

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses url halaman admin secara langsung (http://localhost/testing/css/slide_js/phpslideshow.php)	Redirect ke halaman home	OK
Mengakses halaman simulasi dengan mengklik tombol view simulasi pada halaman	Tampil halaman simulasi	OK

home		
Mengklik link start slide show	Menampilkan simulasi rute terpendek berupa slideshow, dan merubah link start slide show menjadi link stop slide show	OK
Mengklik link stop slide show	Menghentikan slide show dan merubah link stop slide show menjadi start slide show	OK
Mengklik link beginning	Menampilkan gambar rute awal	Ok
Mengklik link Back	Menampilkan rute sebelumnya dari rute yang sedang ditampilkan	OK
Mnegklik link Next	Menampilkan rute selanjutnya dari rute yang sedang ditampilkan	OK
Mengklik tombol home	Menampilkan halaman home	OK
Mengklik tombol view result	Kembali ke halaman home yang menampilkan	OK

	hasil pencarian algoritma dikjstra	

3. Halaman login admin

Tabel 4.20 Tabel Testing Halaman Login Admin

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses url halaman login (http://localhost/testing/index.php/admin)	Tampil halaman login admin	OK
Mengakses semua halaman pada bagian admin tanpa melakukan login	Redirect ke halaman login admin dan menampilkan pesan belum login	OK
Mengklik tombol login dengan isian data tidak lengkap	Menampilkan pesan ada data yang diisi	OK
Mengklik tombol login dengan input username atau password salah	menampilkan pesan bahwa username atau password salah	OK
Mengklik tombol login	Menampilkan pesan	Ok

dengan unername dan password yang benar	selamat datang dan menampilkan menu-menu bagian admin	

4. Halaman data node

Tabel 4.21 Tabel Testing Halaman Data Node

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses halaman data node dengan mengklik menu data node	Tampil halaman data node yang menampilkan seluruh data pada tabel node	OK
Mengklik pada judul Data Node	Tampil form untuk melakukan search	OK
Mengisi form search dan mengklik tombol search	Tampil list data sesuai dengan data yang diinput pada form search	OK
Mengklik link view all	Menampilkan seluruh list data pada tabel node	
Mengklik tombol insert dengan isian data pada form insert yang belum	Menampilkan pesan ada data yang belum diisi	Ok

lengkap		
Mengklik tombol insert dengan data yang sudah diisi secara lengkap	Menginsert data ke database dan menampilkan pesan insert berhasil dilakukan	OK
Mengklik link edit pada data yang dipilih	Menampilkan data yang dipilih pada form update, serta tombol insert berubah menjadi tombol update	OK
Mengklik tombol update dengan isian data yang belum lengkap	Menampilkan pesan ada data yang belum diisi	OK
Mengklik tombol update dengan isian data yang sudah lengkap	Mengupdate data ke database dan menampilkan pesan update berhasil dilakukan	OK
Mengklik link delete pada data yang dipilih	Menghapus data dari database dan menampilkan pesan delete data berhasil dilakukan	OK

5. Halaman data path

Tabel 4.22 Tabel Testing Halaman Data Path

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses halaman data path dengan mengklik menu data path	Tampil halaman data path yang menampilkan seluruh data pada tabel path	OK
Mengklik pada judul Data Path	Tampil form untuk melakukan search	OK
Mengisi form search dan mengklik tombol search	Tampil list data sesuai dengan data yang diinput pada form search	OK
Mengklik link view all	Menampilkan seluruh list data pada tabel path	
Mengklik tombol insert dengan isian data pada form insert yang belum lengkap	Menampilkan pesan ada data yang belum diisi	Ok
Mengklik tombol insert dengan data yang sudah diisi secara lengkap	Menginsert data ke database dan menampilkan pesan insert berhasil dilakukan	OK

Mengklik link edit pada data yang dipilih	Menampilkan data yang dipilih pada form update, serta tombol insert berubah menjadi tombol update	OK
Mengklik tombol update dengan isian data yang belum lengkap	Menampilkan pesan ada data yang belum diisi	OK
Mengklik tombol update dengan isian data yang sudah lengkap	Mengupdate data ke database dan menampilkan pesan update berhasil dilakukan	OK
Mengklik tombol insert atau update dengan input persimpangan 1 sama dengan persimpangan 2	Menampilkan pesan persimpangan 1 dan 2 harus berbeda	OK
Mengklik tombol insert atau update dengan input pasangan persimpangan 1 dan persimpangan 2 sudah	Menampilkan pesan bahwa path dengan persimpangan yang dipilih sudah terdapat pada database	OK

terdapat pada database		
Mengklik link delete pada data yang dipilih	Menghapus data dari database dan menampilkan pesan delete data berhasil dilakukan	OK

6. Halaman data wilayah

Tabel 4.23 Tabel Testing Halaman Data Wilayah

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses halaman data path dengan mengklik menu data wilayah	Tampil halaman data wilayah yang menampilkan seluruh data pada tabel wilayah	OK
Mengklik pada judul Data Wilayah	Tampil form untuk melakukan search	OK
Mengisi form search dan mengklik tombol search	Tampil list data sesuai dengan data yang diinput pada form search	OK
Mengklik link view all	Menampilkan seluruh list data pada tabel wilayah	
Mengklik tombol insert dengan isian data pada	Menampilkan pesan ada data yang belum diisi	Ok

form insert yang belum lengkap		
Mengklik tombol insert dengan data yang sudah diisi secara lengkap	Menginsert data pada database dan menampilkan pesan insert berhasil dilakukan	OK
Mengklik link edit pada data yang dipilih	Menampilkan data yang dipilih pada form update, serta tombol insert berubah menjadi tombol update	OK
Mengklik tombol update dengan isian data yang belum lengkap	Menampilkan pesan ada data yang belum diisi	OK
Mengklik tombol update dengan isian data yang sudah lengkap	Mengupdate data ke database dan menampilkan pesan update berhasil dilakukan	OK
Mengklik link delete pada data yang dipilih	Menghapus data dari database dan menampilkan pesan delete data berhasil dilakukan	OK

7. Halaman data jalan

Tabel 4.24 Tabel Testing Halaman Data Jalan

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengakses halaman data jalan dengan mengklik menu data jalan	Tampil halaman data jalan yang menampilkan seluruh data pada tabel jalan	OK
Mengklik pada judul Data Jalan	Tampil form untuk melakukan search	OK
Mengisi form search dan mengklik tombol search	Tampil list data sesuai dengan data yang diinput pada form search	OK
Mengklik link view all	Menampilkan seluruh list data pada tabel jalan	
Mengklik tombol insert dengan isian data pada form insert yang belum lengkap	Menampilkan pesan ada data yang belum diisi	Ok
Mengklik tombol insert dengan data yang sudah diisi secara lengkap	Menginsert data pada database dan menampilkan pesan insert	OK

	berhasil dilakukan	
Mengklik link edit pada data yang dipilih	Menampilkan data yang dipilih pada form update, serta tombol insert berubah menjadi tombol update	OK
Mengklik tombol update dengan isian data yang belum lengkap	Menampilkan pesan ada data yang belum diisi	OK
Mengklik tombol update dengan isian data yang sudah lengkap	Mengupdate data ke database dan menampilkan pesan update berhasil dilakukan	OK
Mengklik link delete pada data yang dipilih	Menghapus data dari database dan menampilkan pesan delete data berhasil dilakukan	OK

8. Halaman laporan

Tabel 4.25 Tabel Testing Halaman Laporan

Jenis pengujian	Hasil yang diharapkan	Hasil Uji

Mengklik menu laporan node	Menghasilkan output file excel dari data node	OK
Mengklik menu laporan path	Menghasilkan output file excel dari data path	OK
Mengklik menu laporan wilayah	Menghasilkan output file excel dari data wilayah	OK
Mengklik menu laporan jalan	Menghasilkan output file excel dari data jalan	OK

9. Halaman ubah password

Tabel 4.26 Tabel Testing Halaman Ubah Password

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengklik menu ubah password	Menampilkan halaman ubah password	OK
Mengklik tombol save dengan isian data password belum lengkap	Menampilkan pesan ada data yang belum diisi	OK
Mengklik tombol save dengan isian data yang sudah lengkap dengan kondisi password lama	Menampilkan pesan password lama salah	OK

salah		
Mengklik tombol save dengan isian data yang sudah lengkap dengan kondisi pengulangan password baru yang salah	Menampilkan pesan pengulangan password baru salah	OK

10. Halaman grafik

Tabel 4.27 Tabel Testing Halaman Grafik

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengklik menu grafik	Menampilkan grafik intensitas penggunaan algoritma Dijkstra	OK

11. Halaman daftar persimpangan

Tabel 4.28 Tabel Testing Halaman Daftar Persimpangan

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengklik menu daftar persimpangan	Menampilkan daftar persimpangan yang terdapat pada sistem	OK

Melakukan pencarian persimpangan	Menampilkan hasil pencarian persimpangan	OK
Memilih salah satu persimpangan sebagai tempat asal atau tempat tujuan	Persimpangan yang dipilih tampil pada kotak pilihan	OK
Mengklik reset pada kotak pilihan persimpangan	Menghapus persimpangan pada kotak pilihan persimpangan	OK
Mengklik search pada kotak pilihan persimpangan	Menampilkan hasil pencarian rute tercepat dengan redirect ke halaman home	OK

12. Halaman panduan

Tabel 4.29 Tabel Testing Halaman Panduan

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengklik menu panduan	Menampilkan panduan penggunaan sistem	OK

13. Halaman pengaturan data

Tabel 4.30 Tabel Testing Halaman Pengaturan Data

Jenis pengujian	Hasil yang diharapkan	Hasil Uji
Mengklik menu pengaturan data	Menampilkan setting pengaturan jumlah data yang tampil perhalaman data	OK
Mengubah pengaturan dan mengklik save	Menampilkan pengaturan berhasil di ubah	OK

14. Pengujian Jalannya Algoritma Dijkstra terhadap Semua Kemungkinan

Pada sistem, jika algoritma Dijkstra digunakan maka terdapat empat macam kemungkinan berdasarkan input pencarian, yaitu:

- Tempat asal dan tujuan berada pada Graf dan saling terhubung.
- Tempat asal berada pada graf tetapi tempat tujuan berada diluar graf.
- Tempat asal berada diluar graf tetapi tempat tujuan berada didalam graf.
- Tempat asal dan tujuan keduanya berada diluar

graf.

Berikut ini hasil pengujian dari keempat kemungkinan tersebut. Adapun hasil pada tampilan aplikasi ada pada bagian lampiran

Tabel 4.31 Tabel Testing Kemungkinan Algoritma Dijksta

Jenis pengujian	Langkah-langkah pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesesuaian hasil uji
Melakukan pencarian dengan kondisi tempat asal dan tujuan berada pada graf dan saling terhubung	1. Persiapkan dua buah node (persimpangan) yang terdapat dalam graf. 2. Pada kotak input lokasi, pilih salah satu dari kedua persimpangan tersebut sebagai tempat asal, kemudian sisanya sebagai tempat	Sistem akan menampilkan rute tercepat dan terpendek dari tempat asal dan tujuan yang telah dipilih.	Sistem menampilkan hasil pencarian rute tercepat dan terpendek dari tempat asal yang telah dipilih.	OK

	tujuan. 3. Klik tombol tombol “search” 4. Lihat hasil pengujian			
Melakukan pencarian dengan kondisi tempat asal berada pada graf, tetapi tempat tujuan berada diluar graf	1. Persiapkan satu buah node (persimpangan) yang berada pada graf dan satu node berada diluar graf. 2. Pada kotak input lokasi, pilih persimpangan yang berada pada graf sebagai tempat asal kemudian pilih persimpangan diluar graf sebagai tempat tujuan. 3. Klik tombol	Sistem akan menampilkan hasil “Maaf jalur yang menghubungkan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan”	Sistem menampilkan hasil “Maaf jalur yang menghubung kan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan”	OK

	"search" 4. Lihat hasil pencarian			
Melakukan pencarian dengan kondisi tempat asal berada diluar graf dan tempat tujuan berada di dalam graf.	1. Persiapkan satu buah node (persimpangan) yang berada diluar graf dan satu node berada didalam graf 2. Pada kotak input lokasi, pilih persimpangan yang berada diluar graf sebagai tempat asal kemudian pilih persimpangan didalam graf sebagai tempat tujuan. 3. Klik tombol	Sistem akan menampilkan hasil "Maaf jalur yang menghubungkan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan"	Sistem akan menampilkan hasil "Maaf jalur yang menghubungkan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan"	OK

	"search" 4. Lihat hasil pencarian			
Melakukan pencarian dengan kondisi tempat asal dan tujuan berada diluar graf	1. Persiapkan dua buah persimpangan yang berada diluar graf 2. Pada kotak input lokasi pilih salah satu sebagai tempat asal dan sisanya sebagai tempat tujuan. 3. Klik tombol "search" 4. Lihat hasil pencarian	Sistem akan menampilkan hasil "Maaf jalur yang menghubungkan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan"	Sistem akan menampilkan hasil "Maaf jalur yang menghubungkan dari **tempat asal** ke **tempat tujuan** tidak berhasil ditemukan"	OK

4.2.3. *Deployment*

Pada tahap *deployment,* penulis menjalankan sistem yang telah dibangun pada *frame work* Code Igniter dengan menggunakan server apache dan database MySQL yang terdapat dalam satu bundel Xampp. Berikut rincian lengkapnya sebagai berikut adapun untuk *interface* hasil running sistem terdapat pada lampiran 1.

Tabel 4.32 Spesifikasi Software untuk *Running* Sistem

Spesifikasi	Nama Software yang Digunakan
Bahasa pemrograman	PHP 5.2.2
Database	My SQL 5.0.41
Framework	CodeIgniter (CI) 1.7.2
Web Server	Xampp 1.3.2 (Win 32)

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan perumusan masalah dan serangkaian penelitian yang telah penulis lakukan. Maka dapat disimpulkan beberapa hal sebagai berikut.

1. Cara menemukan rute tercepat dan rute terpendek dari suatu lokasi ke lokasi yang lain, dalam hal ini berupa suatu persimpangan ke persimpangan yang lain adalah dengan mengunakan algoritma Dijkstra dengan input berupa persimpangan asal dan tujuan serta sebuah graf yang merepresentasikan node sebagai persimpangan dan simpul atau path sebagai jalur yang menghubungkannya.
2. Faktor-faktor yang dapat diperhitungkan dalam mencari jalur terceoat yaitu waktu tempuh dari suatu persimpangan ke persimpangan lainnya. Adapun faktor yang mempengaruhi waktu tempuh tersebut yaitu penumpukan arus lalu lintas dan hambatan samping seperti penyebrang jalan, pedagang kaki lima dan parkir ditepi jalan.

5.2. Saran

Berikut ini beberapa saran yang dapat dipergunakan untuk pengembangan penelitian dalam mencari rute tercepat dan terpendek.

1. Memperluas cakupan wilayah penelitian seperti meliputi kota Jakarta, Bogor, Depok, Tangerang dan Bekasi, sehingga dapat menjangkau kepentingan masyarakat yang lebih luas.
2. Menggunakan data yang *real time,* sehingga sangat akurat dalam

menentukan rute tercepat pada saat sistem digunakan.

DAFTAR PUSTAKA

Ali, Mohammad. 2007. *Ilmu dan Aplikasi Pendidikan*. Grasindo. Jakarta: 359 hlm.

A.G Haryanto, Hartono Ruslijanto, Datu Mulyono. 2000. *Metode Penulisan dan Penyajian Karya Ilmiah: Buku Ajar Untuk Mahasiswa*. Penerbit Buku Kedokteran EGC. Jakarta.

Barata, Atep Adya. 2003. *Dasar-dasar Pelayanan Prima*. Elex Media Komputindo. Jakarta: 299 hlm.

Basuki, A. P. 2010. *Membangun Web Berbasis PHP Dengan Framework CodeIgniter*. Penerbit Lokomedia, Yogyakarta: xiii + 212 hlm.

Connolly, T. M. & C. E. Begg. 2002. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education Inc., London: xlix + 1236 hlm.

Chamero, Juan. 2006. *Dijkstra's Algorithm As a Dynamic Programming strategy*. http://www.intag.org/downloads/ds_006.pdf, 04 November 2010, pk. 14.48 WIB.

Dharwiyanti, S. & R. S. Wahono. 2003. Pengantar Unified Modeling Language (UML): 13 hlm. http://standy-oei.web.ugm.ac.id/ppl/MateriSuplemenUml.pdf, 03 Agustus 2010, pk. 14.45 WIB.

Edmonds, Jeff. 2008. *How to Think About Algorithm.* Cambridge University Press. New York: xi + 439 hlm.

Goodrich, Michael T. & Roberto Tamassia. 2002. *Algorithm Design foundations, Analysis, and Internet Examples*. John Wiley & Sons, Inc. New York.

Heryandi, Andri. 2010. *Struktur Data Stack*. http://if.unikom.ac.id/andri/download/strukdat/STACK.pdf, 03 Januari 2011, pk 14.20 WIB.

Ibrahim, Ali. 2008. *Cara Praktis Membuat Website Dinamis Menggunakan XAMPP*. Neotekno, Yogyakarta: viii + 146 hlm.

Knuth. Donald E. 1973. *The Art of Computer Programming Volume 1*. Addison-Wesley Company, Inc.

Kristanto, Andri. 2003. *Struktur Data dengan C++*. Graha Ilmu, Yogyakarta: 386 hlm.

Kurniawan, Hendra. 2006. *Panduan Praktis Instalasi Email Server Gratis Berbasis Windows Menggunakan HMailServer*. Elex Media Komputindo, Jakarta: xi + 132 hlm.

Mata-Toledo, R. A. & P. K. Cushman. 2007. *Schaum's Outlines Dasar-Dasar Database Relasional*. Penerbit Erlangga, Jakarta: viii + 157 hlm.

Mulyadi. 2001. *Sistem Akuntansi Edisi ke 3*. Salemba Empat, Jakarta.

Munir, Rinaldi. 2005. *Algoritma dan Pemrograman dalam Bahasa Pascal dan C Buku 1*. Informatika, Bandung: x + 390 hlm.

Munir, Rinaldi. 2005. *Buku Teks Ilmu Komputer Matematika Diskrit Edisi Ketiga*. Informatika. Bandung: xii + 547 hlm.

Nader, J. C. 1992. *Prentice Hall's Illustrated Dictionary of Computing*. Prentice Hall Inc., New South Wales: viii + 526 hlm.

Nurhayati, Oky Dwi. 2010. *Dasar Algoritma*. http://eprints.undip.ac.id/18630/1/pertemuan2.pdf, 03 Januari 2011, pk. 15.40 WIB.

Peranginangin, Kasiman. 2006. *Aplikasi Web dengan PHP dan MySQL*. Andi. Yogyakarta.

Pressman, R. S. 2010. *Software Engineering: A Practitioner's Approach*. Mc Graw Hill Companies Inc., New York: xxviii + 895 hlm.

Purwanto, Eko Budi. 2008. *Perancangan dan Analisis Algoritma*. Graha Ilmu, Yogyakarta: x + 296 hlm.

Rahmat C, Antonius. 2010. *Struktur Data*. http://lecturer.ukdw.ac.id/~mahas/dossier/12_strukdat.pdf, 03 Januari 2011, pk 11.33 WIB.

Rumbaugh, J., I. Jacobson & G. Booch. 2006. *The Unified Modeling Language Reference Manual*. Pearson Education Inc., Westford: xiii + 721 hlm.

Sumanto. 1995. *Metodologi Penelitian Sosial dan Pendidikan.* Andi Offset, Yogyakarta.

Wardana. 2010. *Menjadi Master PHP dengan Framework Codeigniter*. Elex Media Komputindo, Jakarta: ix + 249 hlm.

Williams, B. K. & S. C. Sawyer. 2007. Using Information Technology: Pengenalan Praktis Dunia Komputer dan Komunikasi. Penerbit Andi, Yogyakarta: xxii + 586 hlm.

Wismakarma, Komang. 2010. *Panduan Lengkap Menguasai Pemrograman CSS*. Penerbit Lokomedia, Yogyakarta: xiv + 204 hlm.

Nama : Agung Trianto H, S.SiT

Jabatan : Staf Seksi Manajemen Lalu Lintas

Tempat Wawancara : Kantor Dinas Perhubungan Provinsi DKI Jakarta
Jl Taman Jatibaru No.3 Tanah Abang Jakarta Pusat

Hari, Tanggal : Senin 07 Februari 2011

Isi wawancara:

1. Sebenarnya, secara umum faktor-faktor apa saja yang dapat mempengaruhi pemikiran masyarakat dalam menilai waktu tempuh perjalanan di Jakarta?

   Jawaban:

   Fluktuasi arus dan persepsi pengguna jalan dalam menilai kuantitas kepadatan lalu lintas yang hendak dilalui. Adapun fluktuasi arus adalah penumpukan suatu arus kendaraan yang menuju pada suatu tempat dengan menggunakan pilihan jalur yang sama sehingga menimbulkan kemacetan.

2. Hal-hal apa saja yang menyebabkan perubahan data waktu tempuh antara dua persimpangan dari waktu ke waktu?

   Jawaban:

   Hal-hal tersebut biasanya merupakan hambatan samping. Hambatan samping tersebut misalnya parkir ditepi jalan, penyebrang jalan dan pedagang kaki lima. Contohnya di depan mall Ambassador, arus lalu-lintas akan terhambat dengan banyaknya penyeberang jalan di daerah tersebut.

3. Apakah solusi untuk mencari rute tercepat efektif bagi pengguna jalan?

   Tentu saja sangat efektif sekali, kita bisa buktikan sendiri. Kita sering

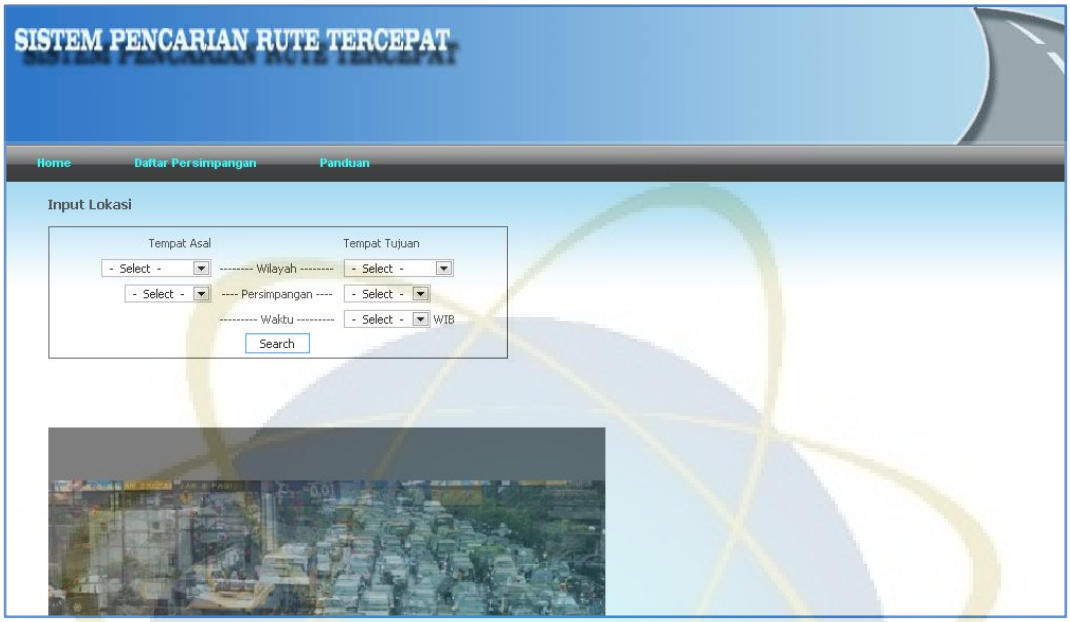
melakukan jalan memutar dan hasilnya ternyata lebih cepat dibandingkan dengan jalan yang biasa dilalui walaupun jaraknya memang lebih jauh. Contohnya dari Pasar Rebo ke kawasan Semanggi, jalur yang biasa dilewati adalah lewat Cililitan, Cawang dan Gatot Subroto. Sedangkan jalur memutar yang bisa dilalui yaitu lewat jalan Tb Simatupang ke arah Cilandak, kemudian ke arah Blok M dan selanjutnya ke Semanggi. Jalur ini lebih cepat walupun jaraknya lebih jauh.

4. Hal-hal apa saja yang telah dilakukan oleh pihak Dishub DKI Jakarta dalam menangani masalah kemacetan sejauh ini?

   Jawaban:

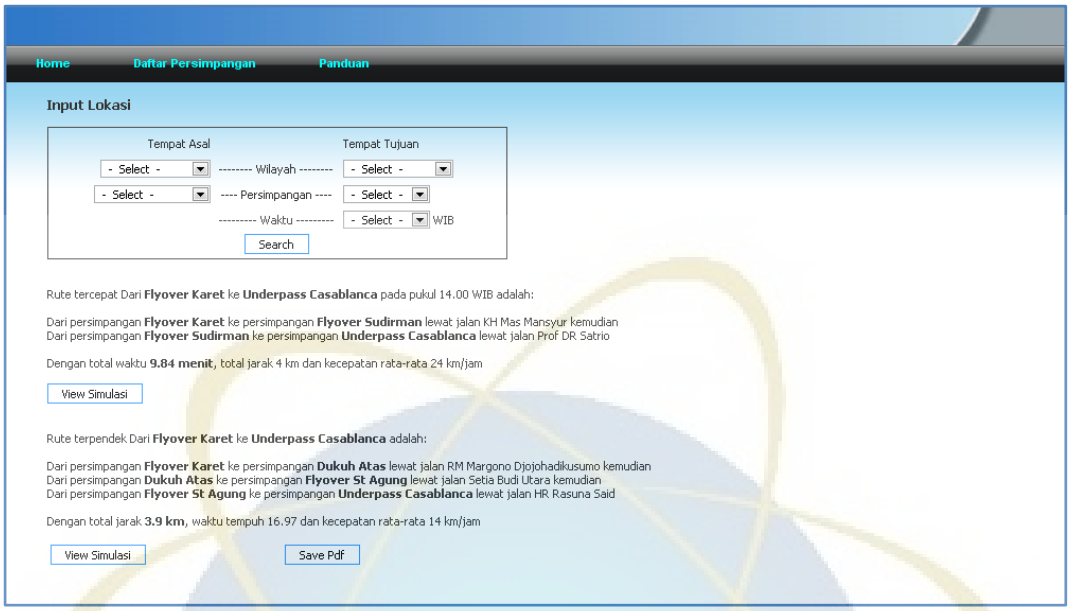
   Banyak sekali, contohnya seperti Pemberlakuan jam *Tree in one*, pengoperasian *busway,* pembangunan fly over dan jalan layang, larangan belok kanan, pemberlakuan jalur satu arah dan lain-lain.

1. Hasil Pengujian pada Halaman Home



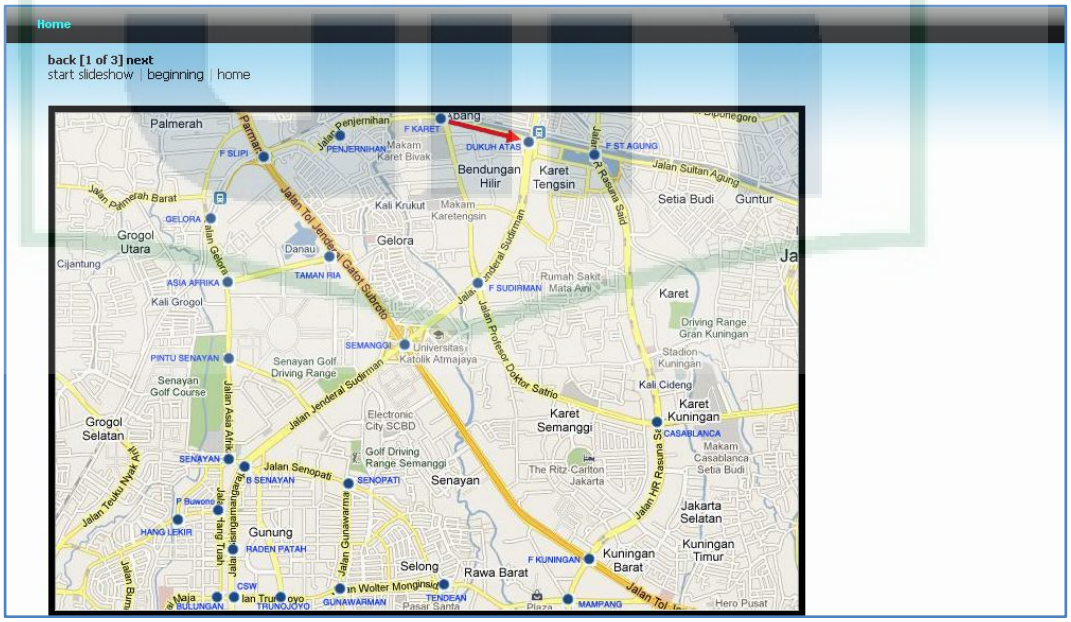
Tampilan halaman home

Tampilan saat wilayah dan persimpangan dipilih



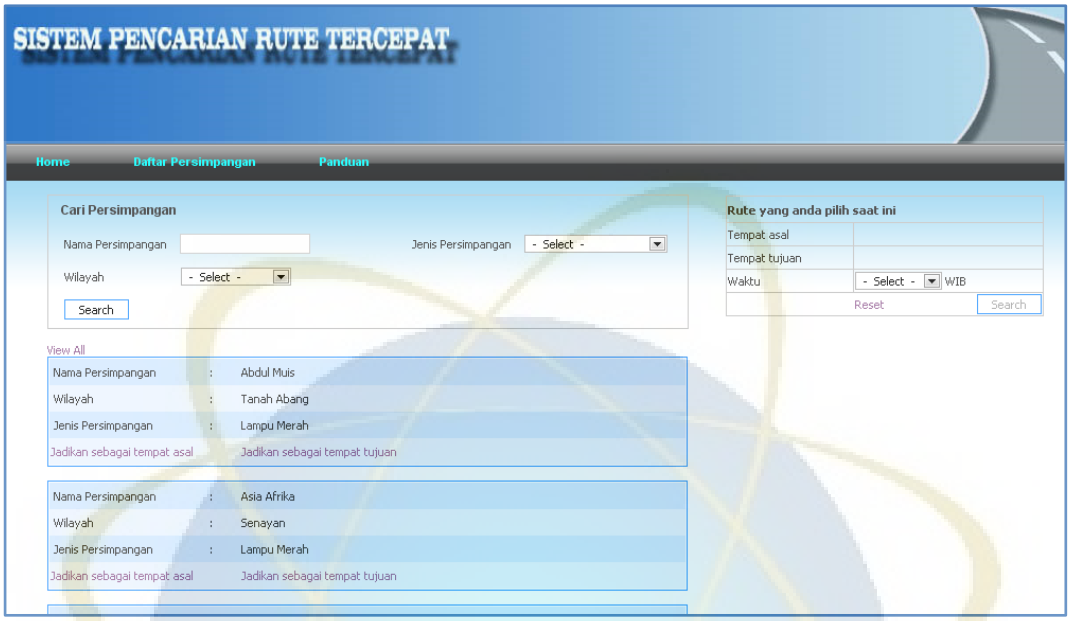
Tampilan hasil pencarian rute tercepat dan rute terpendek

2. Hasil Pengujian pada Halaman Simulasi

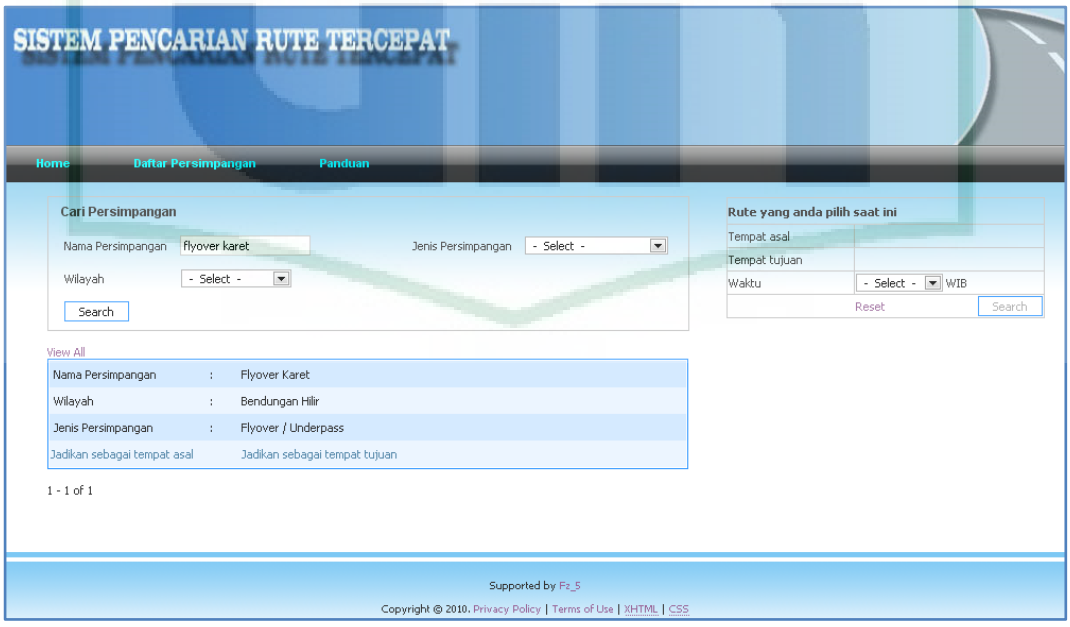


Tampilan simulasi hasil pencarian rute tercepat dan terpendek

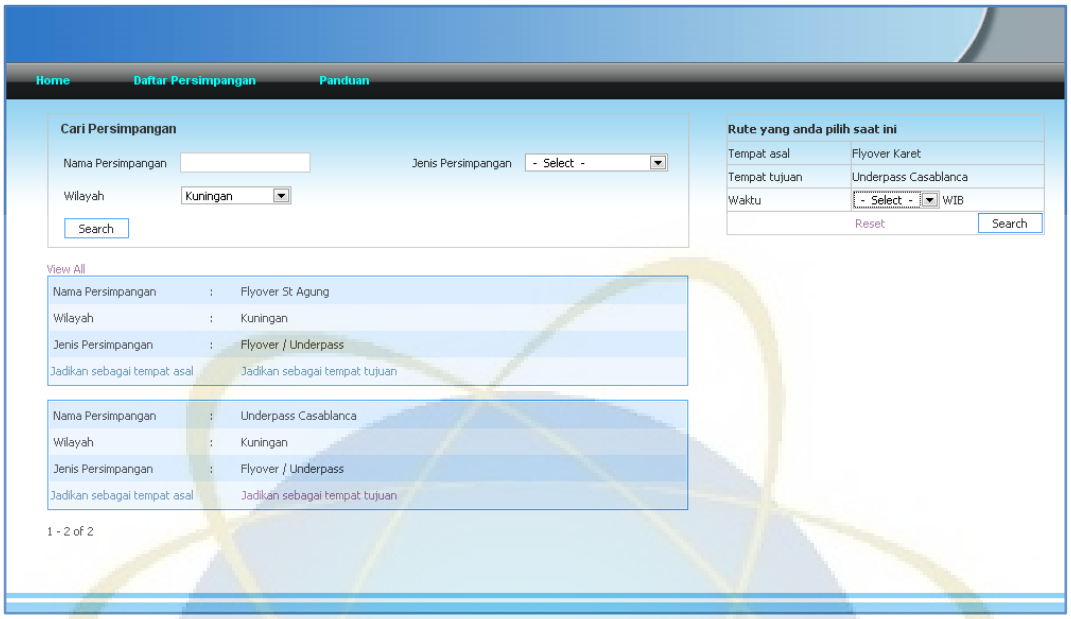
3. Hasil Pengujian pada Halaman Daftar Persimpangan



Tampilan halaman daftar persimpangan

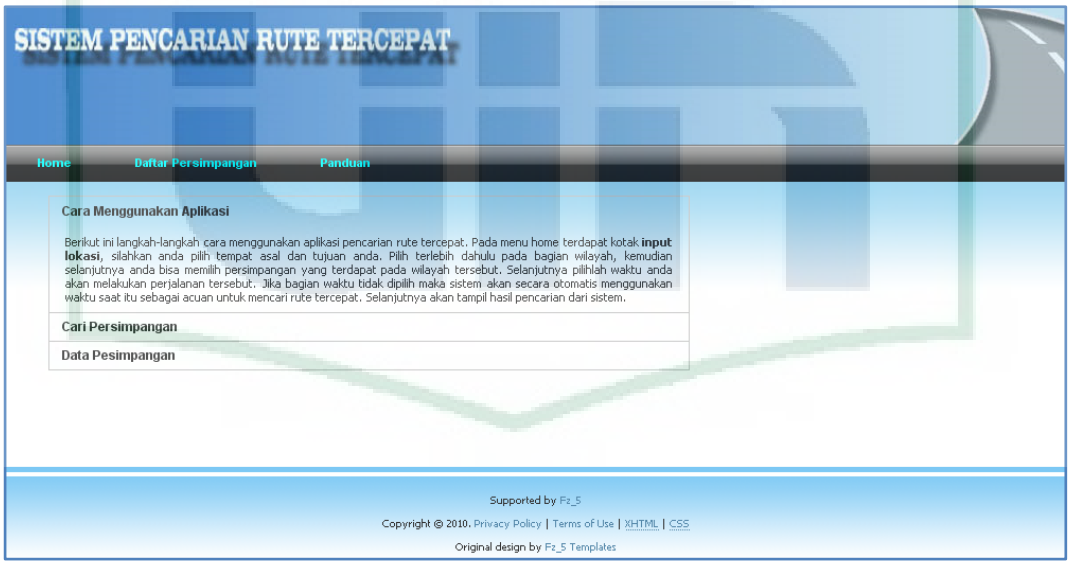


Tampilan saat dilakukan pencarian pada daftar persimpangan



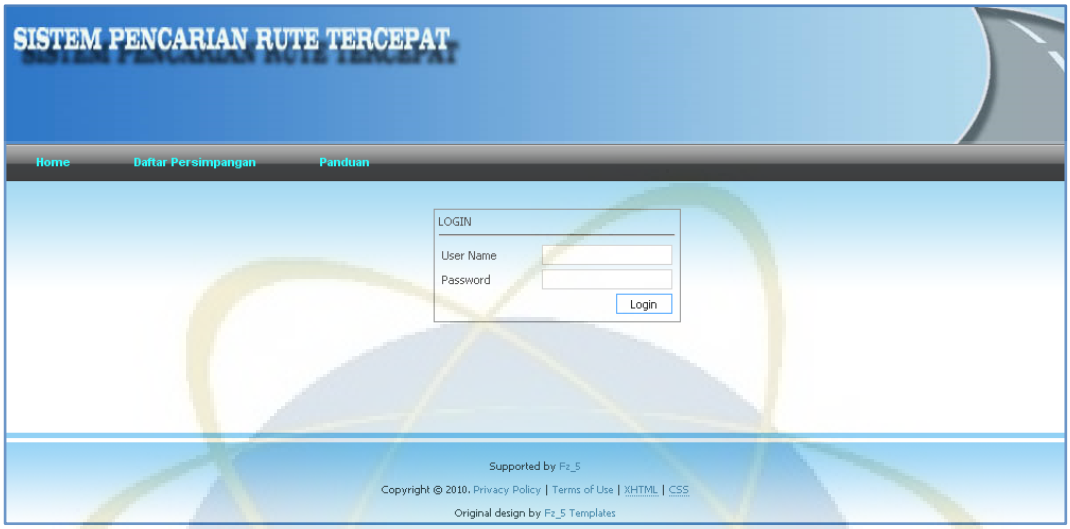
Tampilan saat user telah memilih persimpangan asal dan tujuan

4. Hasil Pengujian pada Halaman Panduan

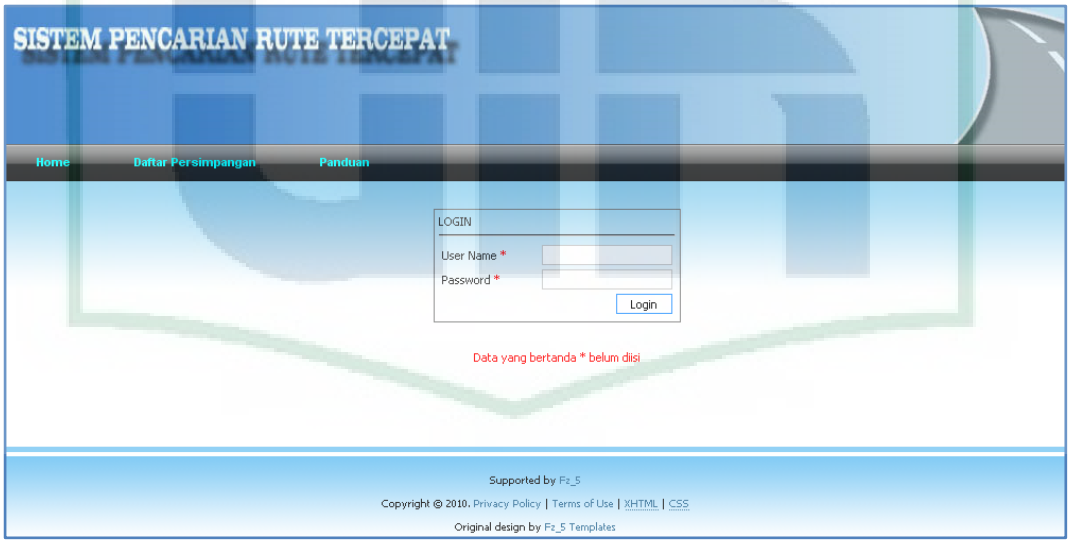


Tampilan pada halaman panduan

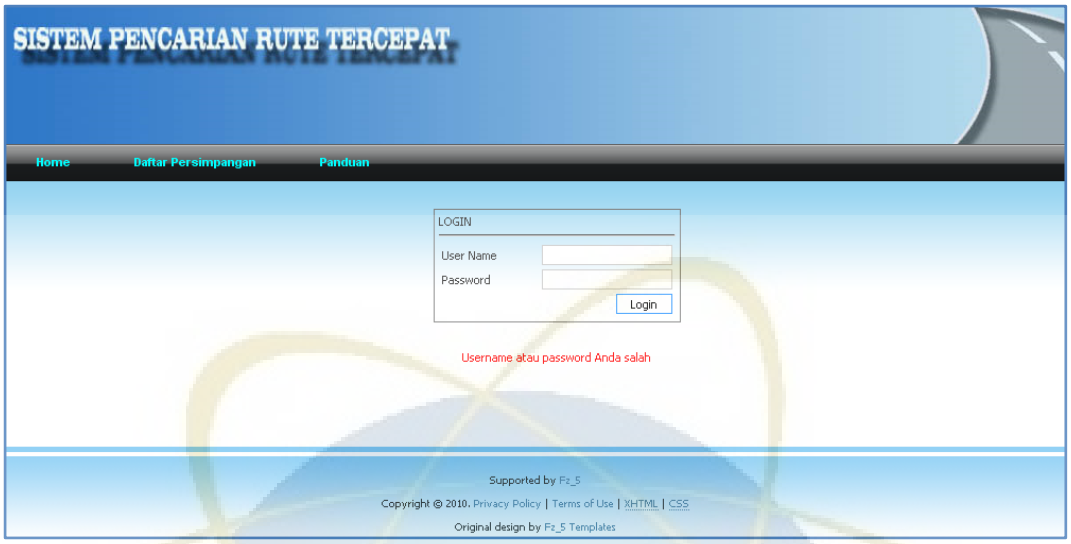
5. Hasil Pengujian pada Halaman Login Admin



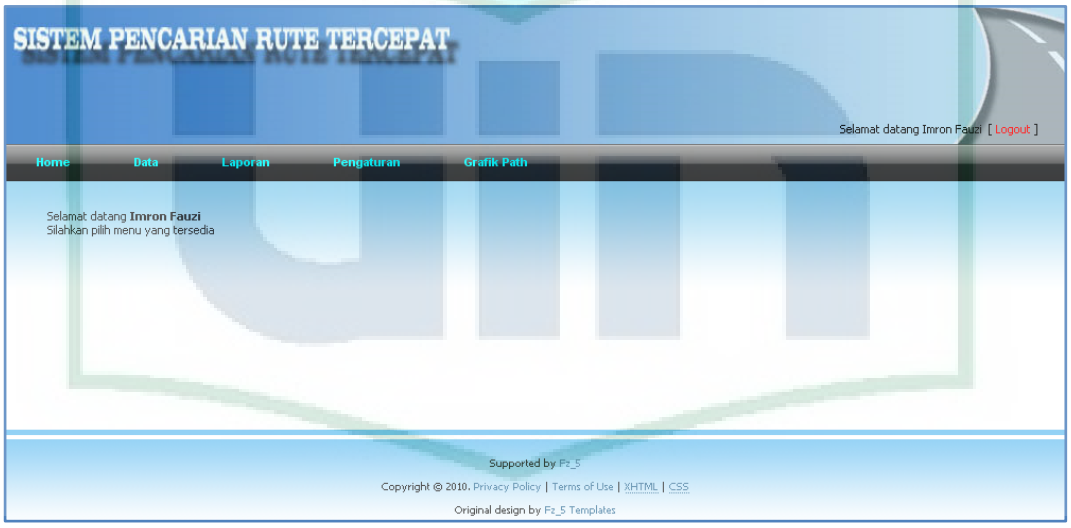
Tampilan hal login admin



Tampilan saat login tanpa mengisi username dan password



Tampilan saat username atau password salah



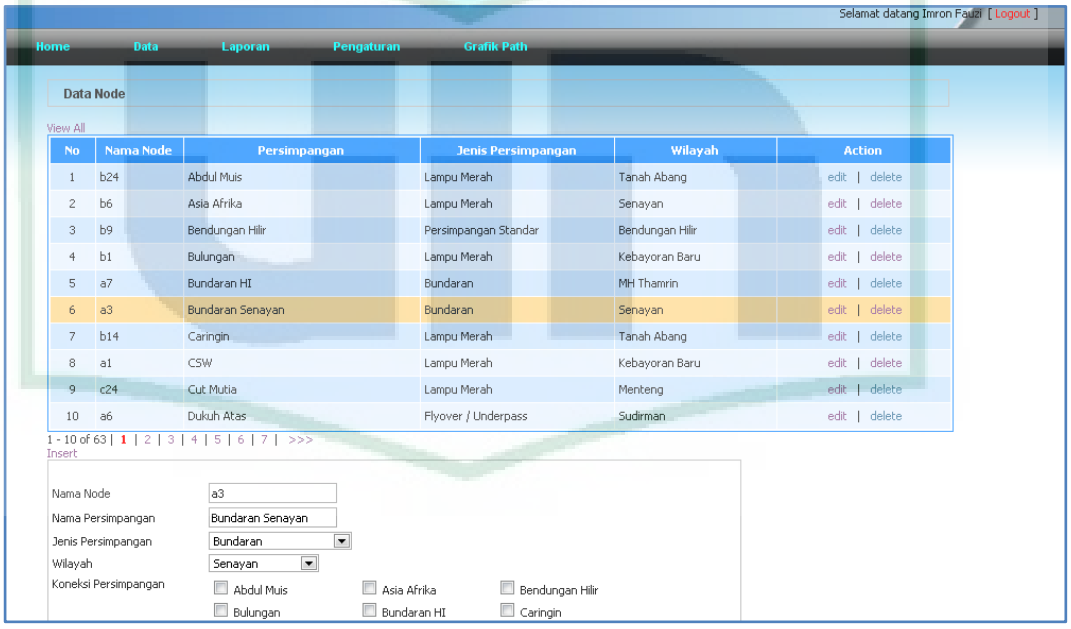
Tampilan saat berhasil login

6. Hasil Pengujian pada Halaman Data Node

Tampilan halaman node

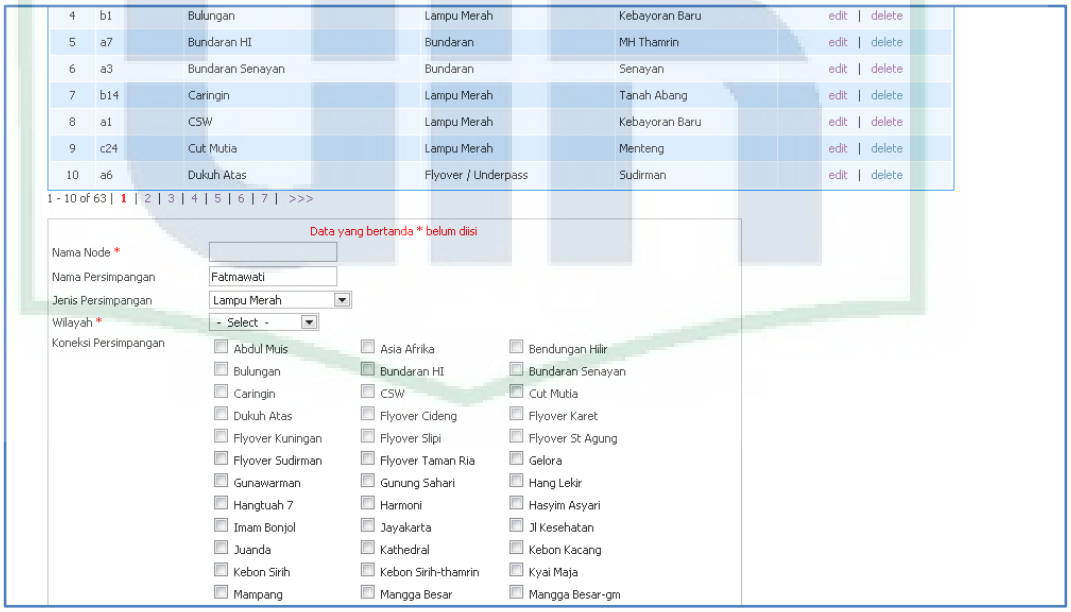
Tampilan saat dilakukan pencarian pada halaman data node

Tampilan hasil pencarian dengan kata kunci “senayan”



Tampilan saat dipilih salah satu node untuk di edit

Tampilan saat update data berhasil dilakukan



Tampilan saat melakukan insert data tanpa melengkapi isian form insert

Tampilan saat insert data berhasil dilakukan

Tampilan saat dilakukan penghapusan pada salah satu data

7. Hasil Pengujian pada Halaman Data Path

Tampilan halaman data path

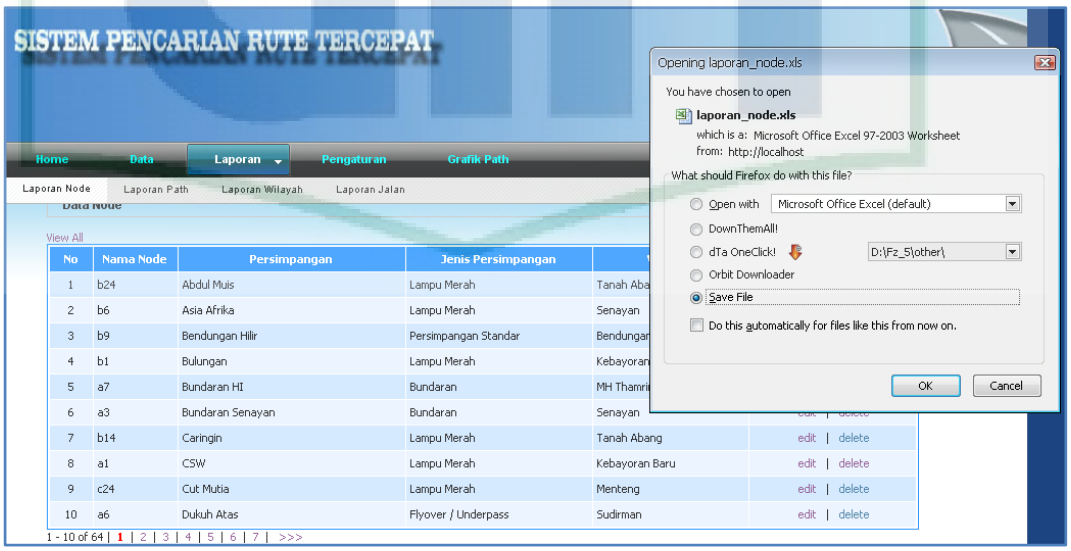
8. Hasil Pengujian pada Halaman Data Wilayah

Tampilan halaman data wilayah

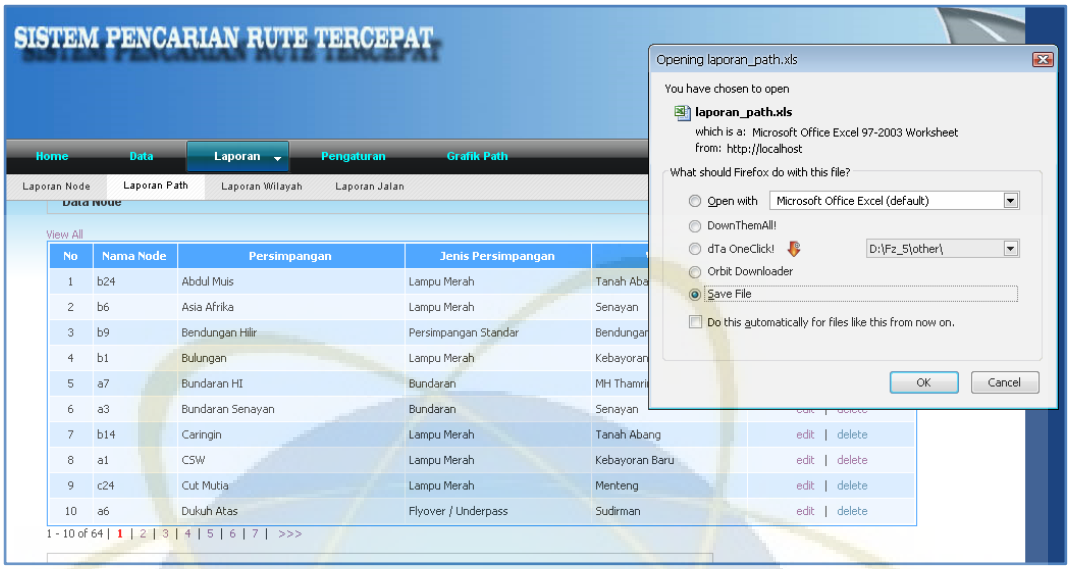
9. Hasil Pengujian pada Halaman Data Jalan

Tampilan halaman data jalan

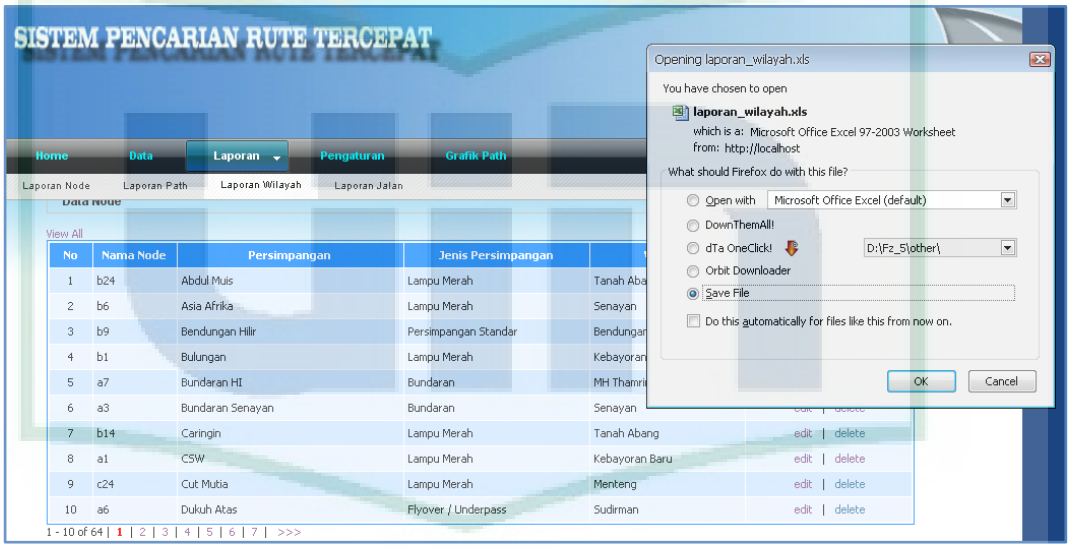
10. Hasil Pengujian pada Halaman Laporan



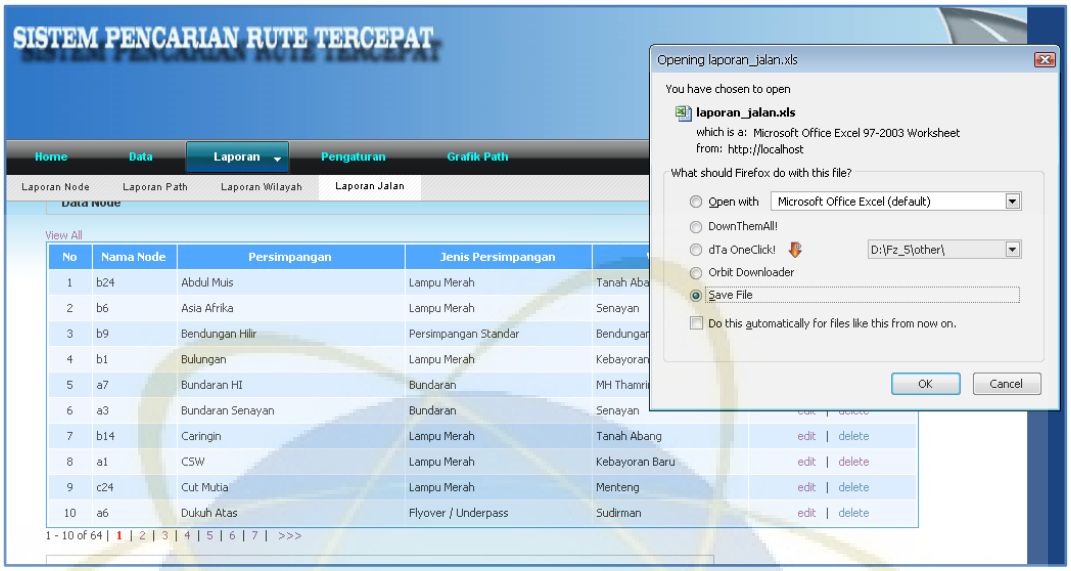
Tampilan saat menu laporan node di klik



Tampilan saat menu laporan path di klik

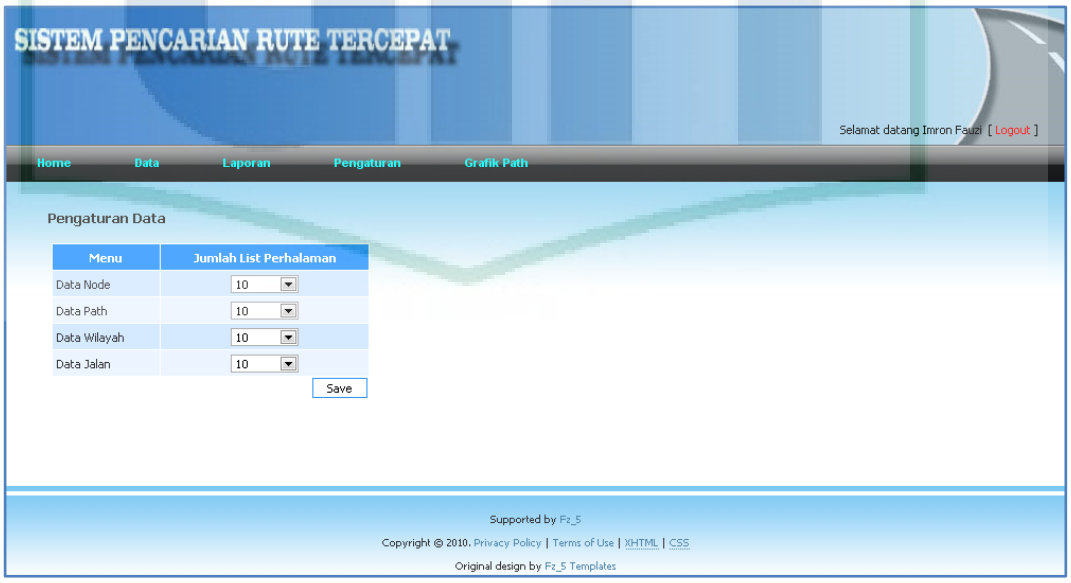


Tampilan saat menu laporan wilayah di klik



Tampilan saat menu laporan jalan di klik

11. Hasil Pengujian pada Halama Setting Data

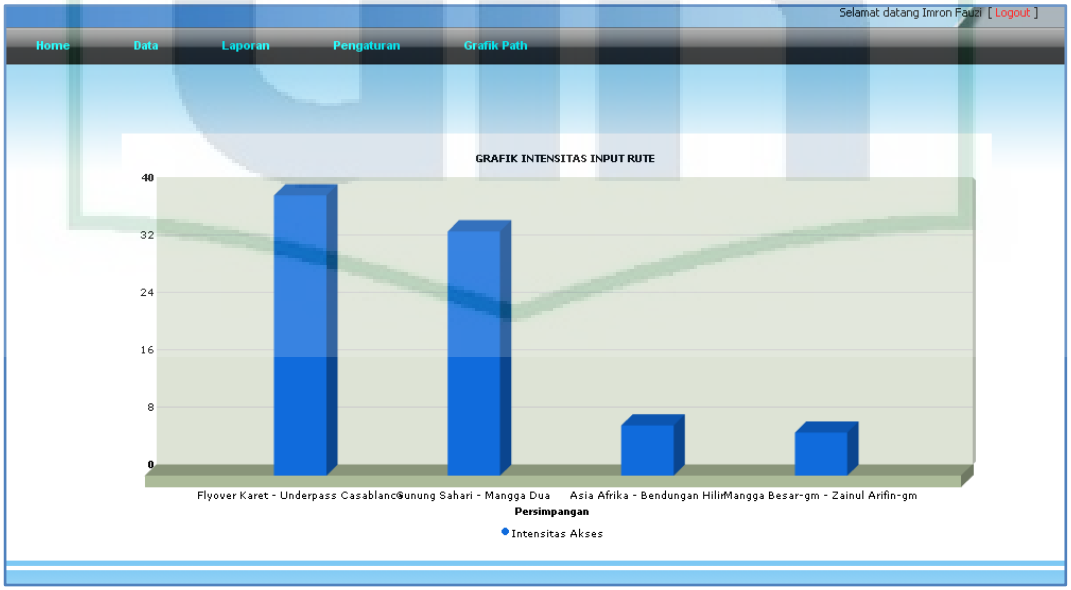


Tampilan pada halaman setting data

12. Hasil Pengujian pada Halaman Ubah Password

Tampilan halaman ubah password

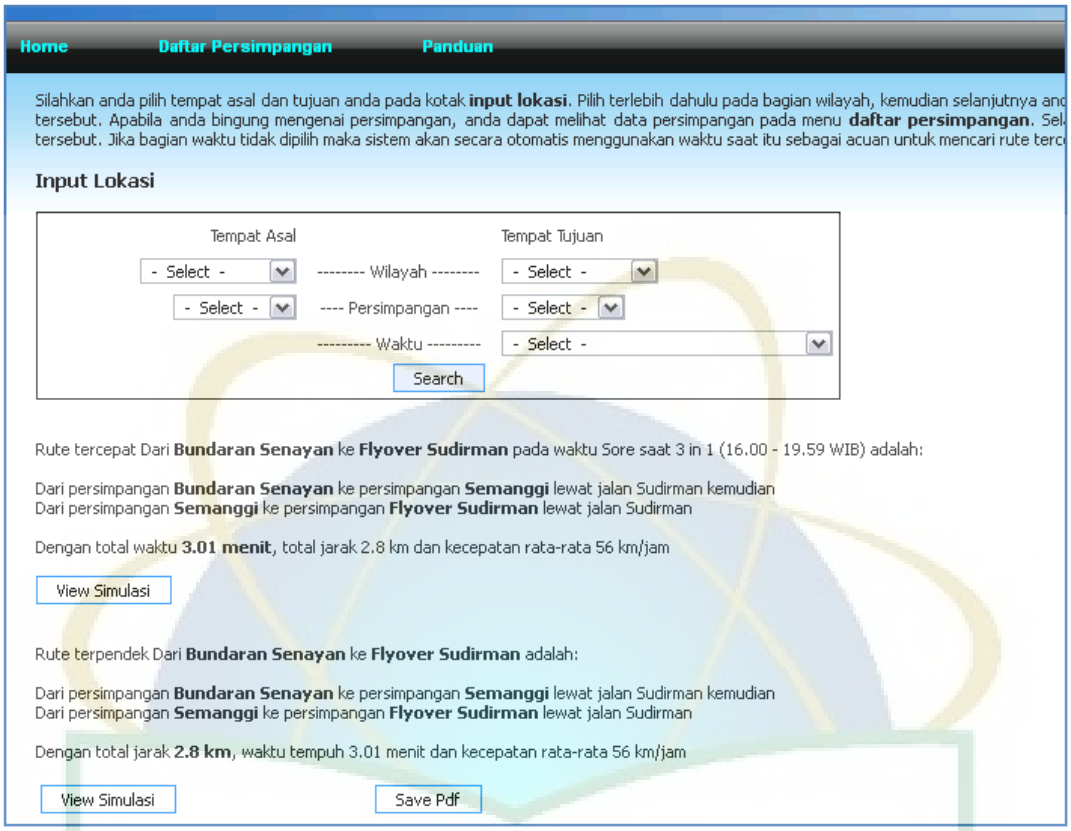
13. Hasil Pengujian pada Halaman Grafik



Tampilan halaman grafik

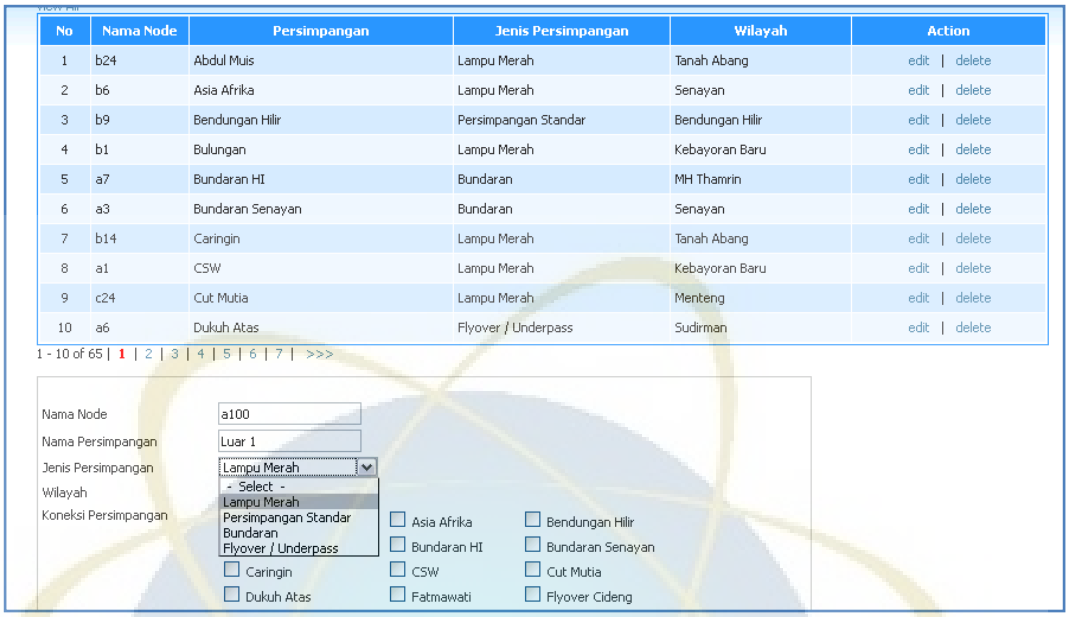
14. Tampilan Hasil Pengujian Algoritma Dijkstra dengan Kondisi Tempat Asal dan Tempat Tujuan Berada pada Graf

Menyiapkan dan memilih dua persimpangan yang terdapat pada graf

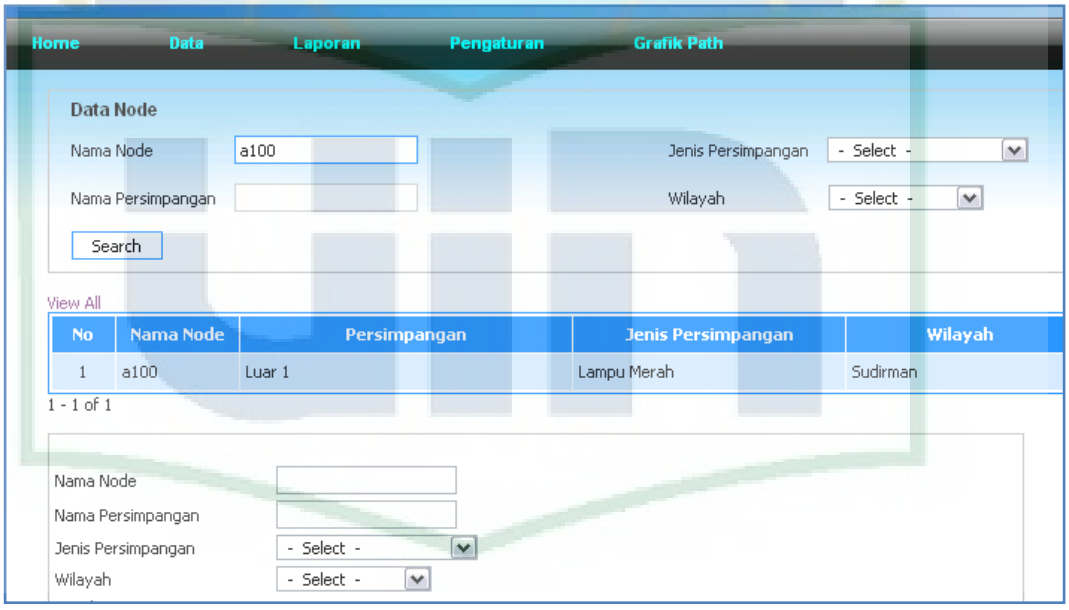


Tampilan hasil pencarian

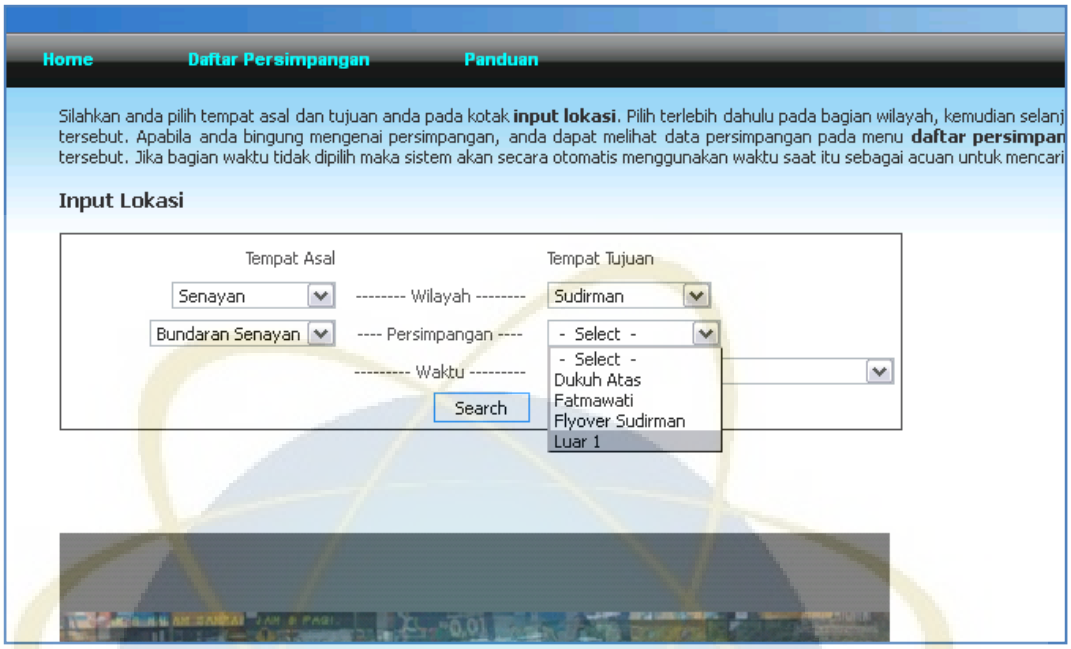
15. Tampilan Hasil Pengujian Algoritma Dijkstra dengan Kondisi Tempat Asal Berada pada Graf tetapi Tempat Tujuan Berada diluar Graf



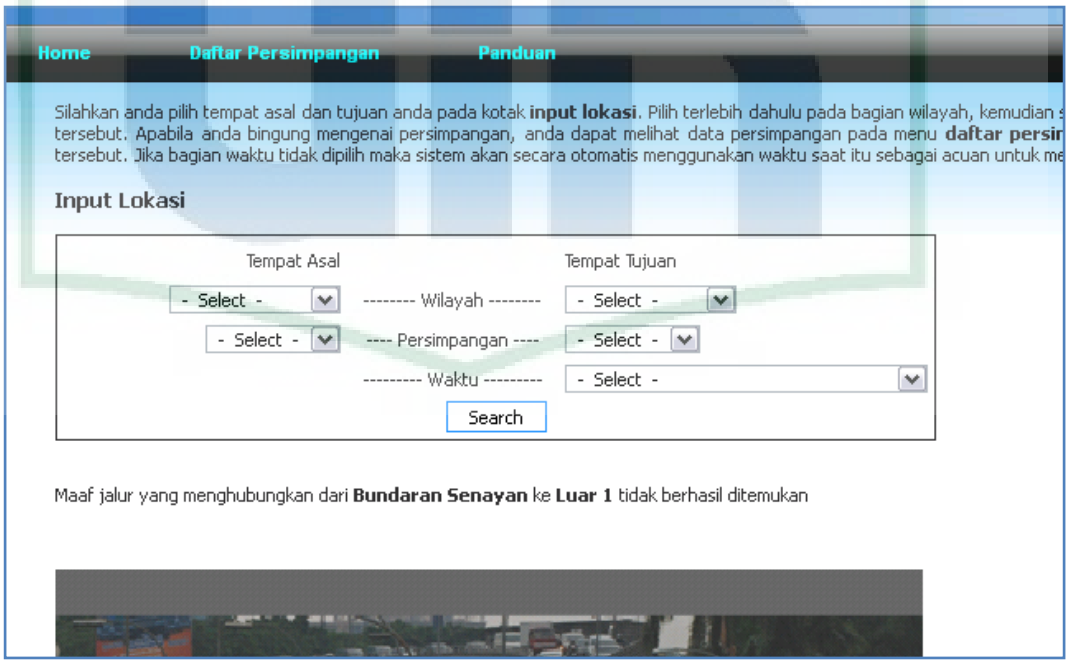
Pembuatan node (persimpangan baru) dengan nama “Luar 1”



Pembuatan node baru berhasil dilakukan

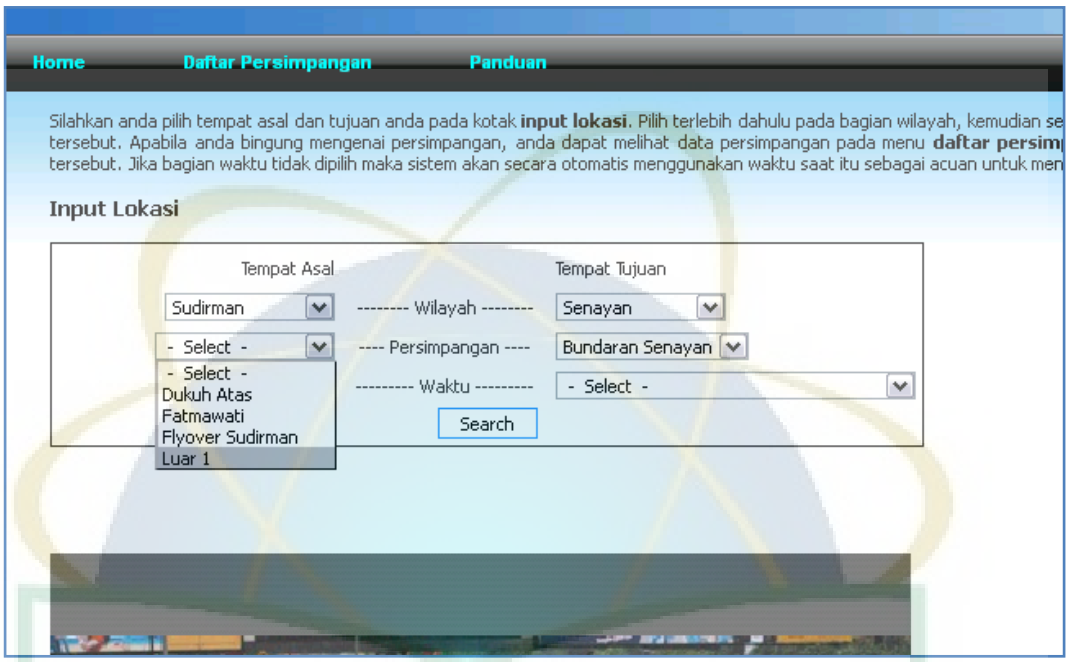


Pemilihan persimpangan “bundaran senayan” sebagai tempat asal dan persimpangan “Luar 1” sebagai tempat tujuan



Hasil pemcarian

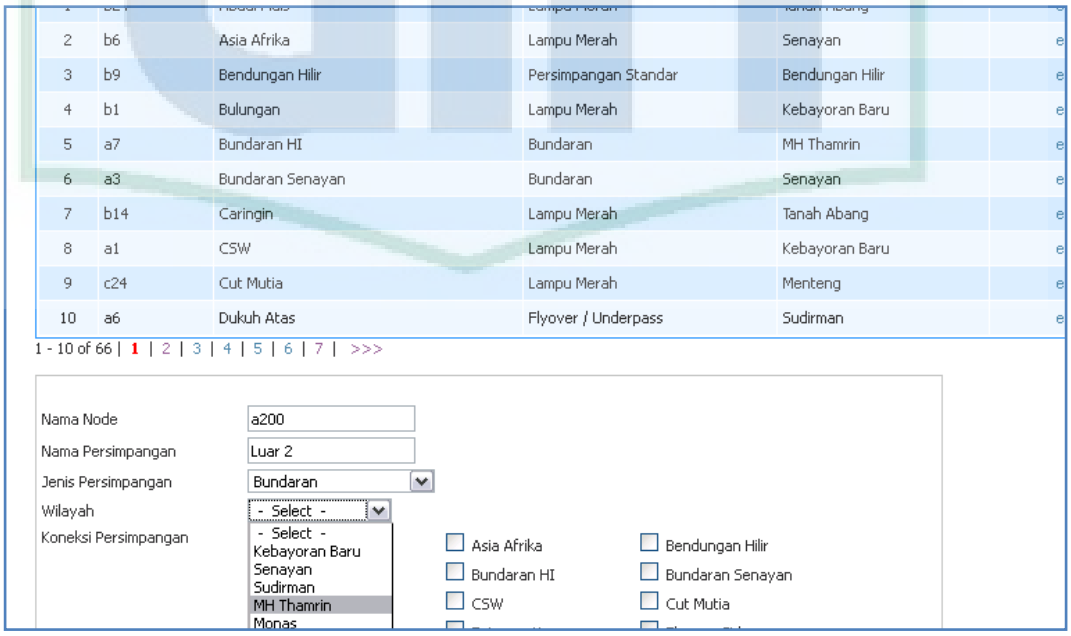
16. Tampilan Hasil Pengujian Algoritma Dijkstra dengan Kondisi Tempat Asal Berada diluar Graf tetapi Tempat Tujuan Berada didalam Graf.



Pemilihan persimpangan “Luar 1” sebagai tempat asal dan persimpangan “Bundaran Senayan” sebagai tempat tujuan

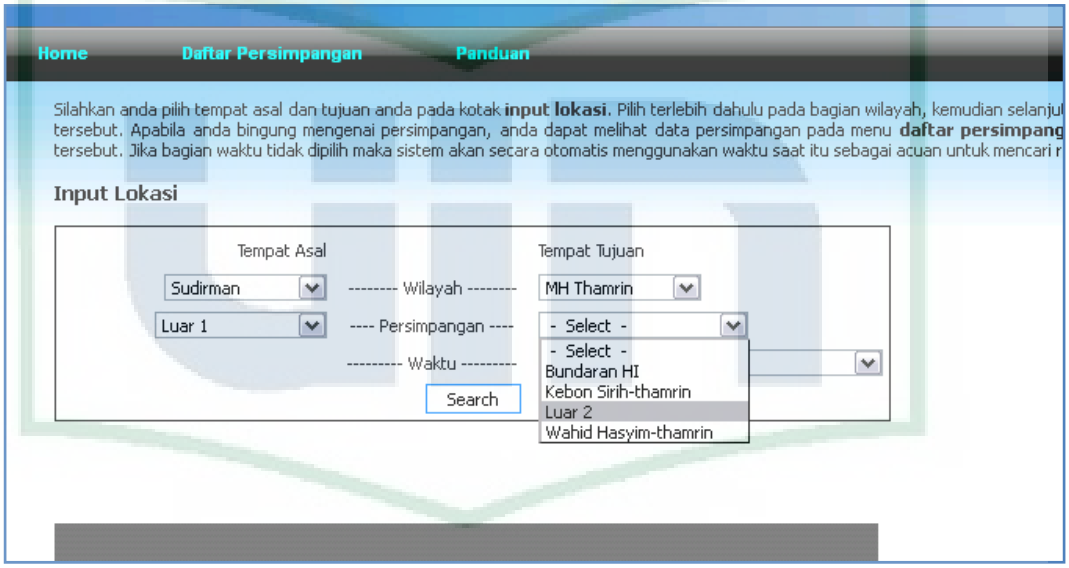
Hasil pencarian

17. Tampilan Hasil Pengujian Algoritma Dijkstra dengan Kondisi Tempat Asal dan Tujuan Berada diluar Graf.

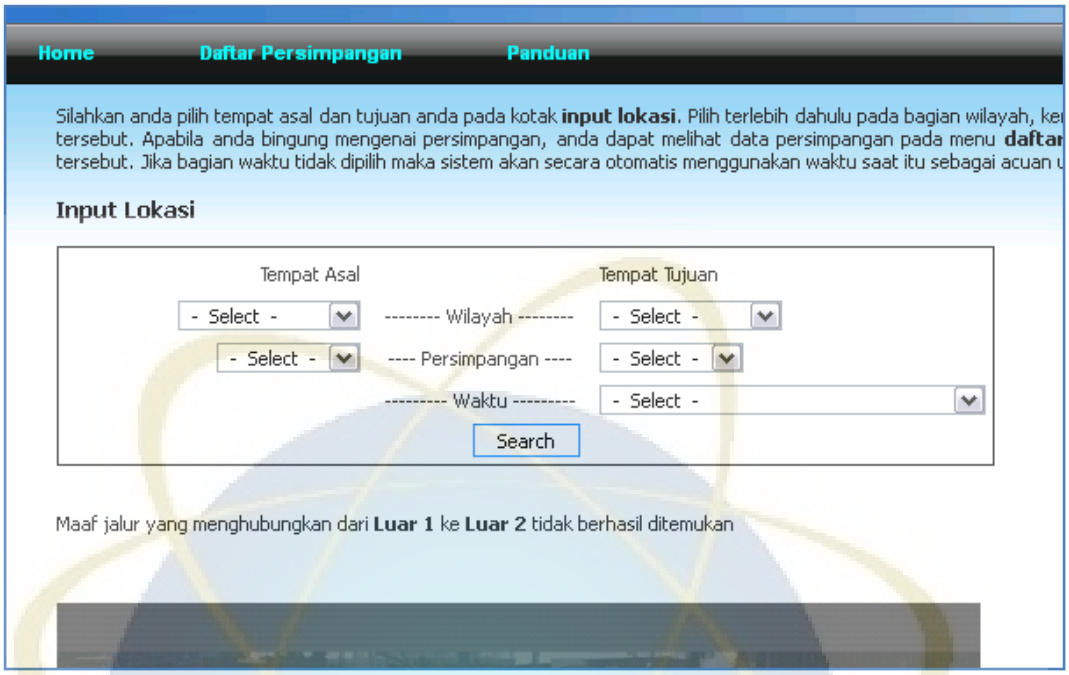


Pembuatan node baru dengan nama persimpangan “Luar 2”

Pembuatan persimpangan ‘Luar 2” berhasil dilakukan



Pemilihan persimpangan “Luar 1” sebagai tempat asal dan persimpangan “Luar 2 ” sebagai tempat tujuan



Hasil pencarian

PENGGUNAAN ALGORITMA DIJKSTRA DALAM PENCARIAN RUTE TERCEPAT DAN TERPENDEK (Studi Kasus Pada Jalan Raya Antara Wilayah Blok M dan Kota)

Imron Fauzi – 106091002883
Teknik Informatika, Universitas Islam Negeri Syarif Hidayatullah Jakarta Dibimbing oleh Khodijah Hulliyah, M.Si dan Hendra Bayu Suseno, M.Kom

ABSTRAK

Kemacetan di Jakarta sudah menjadi pemandangan sehari-hari. Banyak langkah-langkah yang telah dilakukan oleh pemerintah untuk mengatasi kemacetan tersebut, seperti pembangunan *flyover* dan *underpass*, pengoperasian jalur *busway*, pemberlakuan jam *tree in one* dan sebagainya. Akan tetapi kemacetan tetap saja masih sering terjadi sampai saat ini. Oleh karena itu diperlukan peran aktif dari pengguna jalan sendiri untuk dapat mengatasi kemacetan tersebut. Salah satu cara yang paling efektif yaitu dengan mencari rute alternatif yang dapat dilalui. Sebelumnya telah dilakukan penelitian mengenai hal ini, namun belum bisa menjawab persoalan diatas karena kebanyakan penelitian tersebut hanya menggunakan parameter jarak tempuh. Oleh karena itu penulis mencoba membuat sebuah sistem yang menggunakan algoritma Dijkstra yang dapat menemukan jalur tercepat dan terpendek dengan menyertakan faktor kecepatan dan waktu tempuh perjalanan, ruang lingkup yang luas dan bersifat *online*. Adapun penggunaan algoritma Dijkstra karena algoritma ini dipastikan menemukan solusi terbaik dan memiliki kompleksitas yang lebih sedikit jika dibandingkan dengan algoritma sejenis seperti algoritma Bellman Ford dan Floyd Warshall. Pada pengembangan sistem ini penulis menggunakan metode *spiral model* dan kode program dibuat dengan menggunakan *framework* Code Igniter (CI). Sistem ini memberikan keluaran berupa jalur tercepat dan terpendek dari tempat asal menuju tempat tujuan yang diinputkan oleh pengguna. Jalur tercepat dan terpendek tersebut dilengkapi dengan total jarak tempuh, waktu tempuh serta kecepatan rata-rata.

Kata kunci : Algoritma Dijkstra, Rute Tercepat, Spiral Model

Jumlah Halaman : 150 halaman

Jumlah Daftar Pustaka : 30 sumber

I PENDAHULUAN

1.1. Latar Belakang

Kemacetan di Jakarta selalu terjadi setiap hari kerja. Jumlah kepadatan penduduk yang besar dan aktifitas yang beragam membuat kota ini tidak pernah sepi dari lalu lintas manusia. Salah satu solusi yang efektif dilakukan adalah dengan memilih rute tercepat.

Sebelumnya telah dilakukan penelitian mengenai hal ini, salah satu penelitian tersebut yaitu penelitian tugas akhir yang dilakukan oleh Yadi Rusyad Nurdin yang berjudul "***Implementasi Algoritma Dijkstra Dalam Menentukan Rute Terpendek Pada Dua Titik Lokasi Wilayah Menteng Raya***". Namun penelitian tersebut memiliki beberapa kekurangan seperti jalur yang ditemukan hanya sebatas jalur terpendek, titik lokasi yang diinput bersifat tetap tidak bisa dirubah serta aplikasi yang dihasilkan berupa aplikasi desktop.

Oleh karena itu penulis mencoba membuat sebuah sistem yang dapat menemukan jalur tercepat dan terpendek dengan menyertakan faktor kecepatan dan waktu tempuh perjalanan, bersifat *on line* serta titik lokasi yang dijadikan input dapat dilakukan perubahan. Sehingga sistem yang

dihasilkan lebih bermanfaat bagi pengguna serta dapat diakses kapanpun dan dimanapun. Dalam hal ini, penulis menggunakan judul penelitian **Penggunaan Algoritma Dijkstra Untuk Pencarian Jalur Tercepat dan Jalur Terpendek (Studi Kasus Pada Jalan Raya antara Wilayah Blok M dan Kota)**.

1.2. Perumusan Masalah

Beberapa permasalahan yang dapat dirumuskan pada penelitian ini diantaranya:

1. Bagaimana cara menemukan jalur tercepat dan jalur terpendek dari satu lokasi ke lokasi lain yang ada di Jakarta khususnya antara wilayah Blok M dan Kota?
2. Faktor-faktor apa saja yang harus diperhitungkan dalam mencari jalur tersebut?

1.3. Batasan Masalah

Batasan masalah yang dilakukan pada penelitian ini yaitu:

1. Penelitian ini hanya dilakukan di wilayah Jakarta (khususnya antara wilayah Blok M dan Kota).
2. Objek pada penelitian ini hanya pada jalan raya, yaitu jalan umum yang dapat dilalui oleh kendaraan yang berukuran besar seperti Bus.
3. Penelitian ini hanya sebatas menemukan jalur tercepat yang dapat digunakan oleh pengguna.
4. Data kecepatan dan waktu tempuh pada penelitian ini menggunakan acuan kendaraan roda empat yang didapatkan dari pihak Dishub Provinsi DKI Jakarta.

1.4. Tujuan dan Manfaat Penelitian

Berdasarkan uraian latar belakang, maka tujuan pada penelitian ini yaitu untuk membantu masyarakat dalam menemukan rute tercepat dan terpendek pada jalan raya di Jakarta, khususnya antara wilayah blok M dan Kota.

II LANDASAN TEORI

2.1. Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma yang paling sering digunakan dalam pencarian rute terpendek, sederhana penggunaannya dengan menggunakan simpul-simpul sederhana pada jaringan jalan yang tidak rumit (Chamero, 2006). Adapun nama algoritma Dikstra sendiri berasal dari penemunya yaitu Edsger Dikstra.

Dalam mencari solusi, algoritma Dijkstra menggunakan prinsip greedy, yaitu mencari solusi optimum pada setiap langkah yang dilalui, dengan tujuan untuk mendapatkan solusi optimum pada langkah selanjutnya yang akan mengarah pada solusi terbaik. Hal ini membuat kompleksitas waktu algoritma Dijkstra menjadi cukup besar, yaitu sebesar O (V * Log (v + e)), dimana v dan e adalah simpul dan sisi pada graf yang digunakan.

Input dari algoritma Dijkstra berupa sebuah graf berbobot G(e , v), sedangkan outputnya berapa rute terpendek dari simpul awal (start) ke masing-masing simpul yang ada pada graf. Dengan demikian algoritma Dijkstra dapat menemukan solusi terbaik.

Cara kerja algoritma Dijkstra hampir sama dengan cara kerja algoritma BFS yaitu dengan menggunakan prinsip antrian (queue), akan tetapi antrian yang digunakan algoritma Dijkstra adalah antrian berprioritas (priority queue). Jadi hanya simpul yang memiliki prioritas tertinggi yang akan ditelusuri. Dalam menentukan

simpul yang berprioritas, algoritma ini membandingkan setiap nilai (bobot) dari simpul yang berada pada satu level. Selanjutnya nilai (bobot) dari setiap simpul tersebut disimpan untuk dibandingkan dengan nilai yang akan ditemukan dari rute yang baru ditemukan kemudian, begitu seterusnya sampai ditemukan simpul yang di cari.

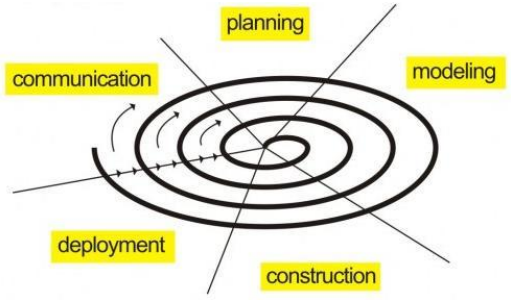
III METODOLOGI PENELITIAN

3.1. Metode Pengumpulan Data

Observasi penulis lakukan ke kantor Dinas Perhubungan Provinsi DKI Jakarta untuk mendapatkan data mengenai waktu tempuh dan jarak antara dua persimpangan pada jalan raya di Jakarta khususnya antara wilayah Blok M dan Kota. Selain itu penulis juga melakukan studi pustaka, wawancara dan kuisioner dengan tujuan untuk mendapatkan data dan referensi yang berbobot dari masalah yang diteliti.

3.2. Metode Pengembangan Sistem Spiral Model

Penulis menggunakan metode ini dikarenakan penelitian ini membutuhkan tahap tahap pengmbangan sistem yang selalu berlanjut. Dengan kata lain pengembangan tidak bersifat statis tetapi dinamis dengan melakukan pengembangan prototype yang dilakukan secara berulang untuk mencapai sisten yang sesungguhnya. Adapun metode ini pertama kali diusulkan oleh Barry Boehm. Spiral model merupakan metode pengembangan sistem evolusioner yang menyatukan sifat iterasi dari prototyping dengan kontrol dan aspek sistematis dari model sekuensial atau waterfall model (Pressman, 2010).



3.1.1 ***Communication***

Dalam hal ini penulis berkomunikasi secara langsung dengan pihak Dishub Provinsi DKI Jakarta yang diwakili oleh Bapak Agung pada bagian Manajemen dan Rekayasa Lalu Lintas. Komunikasi yang penulis lakukan diantaranya yaitu mengenai data waktu tempuh antar persimpangan, faktor-faktor waktu tempuh perjalanan serta pembangunan rancangan dan pengembangan aplikasi

3.2.1 ***Planning***

Pada tahap ini, dilakukakn estimasi biaya, penjadwalan penelitian dan analisis resiko yang kemungkinan terjadi saat sistem digunakan.

3.3.1 ***Modeling***

Pemodelan sistem dilakukan dengan menggunakan *Unified Modeling* Language (UML). Dikarenakan aplikasi yang dibuat berbasis pemrograman berorientasi objek.

3.4.1 ***Construction***

Tahap ini dilakukan pembuatan kode program dan selanjutnya dilakukan testing.

3.5.1 ***Deployment***

Deployment yaitu tahap

running dari aplikasi yang telah dibuat.

IV ANALISIS DAN PERANCANGAN SISTEM

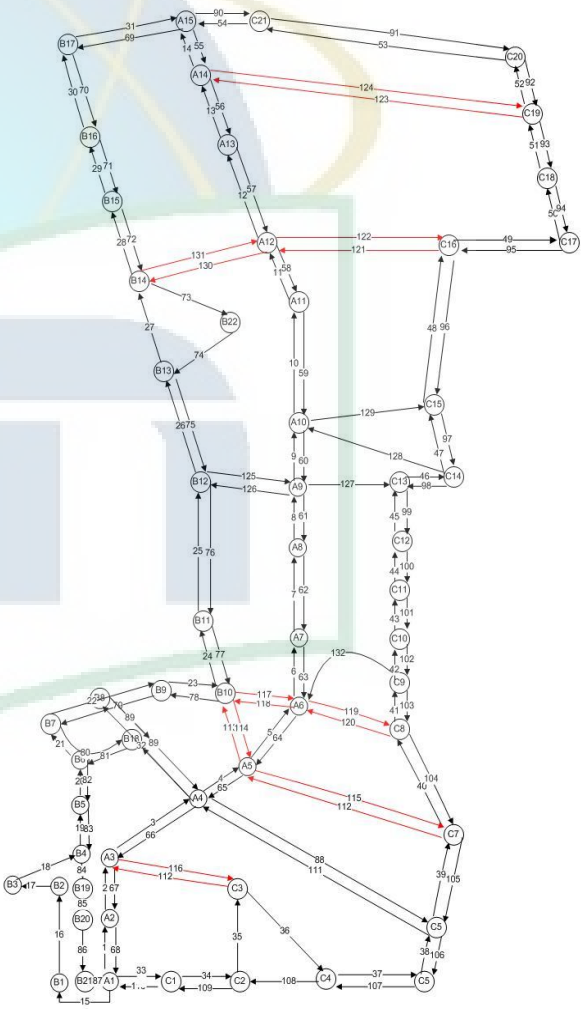
4.1. Analisis Sistem

Berdasarkan hasil observasi dan wawancara yang penulis lakukan pada Dinas Perhubungan Provinsi DKI Jakarta. Terdapat berbagai macam faktor yang dapat menyebabkan kemacetan sehingga mempengaruhi waktu tempuh perjalanan. Secara garis besar, faktor-faktor tersebut diantaranya yaitu fluktuasi arus dan hambatan samping. Fluktuasi arus adalah penumpukan suatu arus kendaraan yang menuju pada suatu tempat dengan menggunakan pilihan jalur yang sama sehingga menimbulkan kepadatan lalu-lintas dan selanjutnya menyebabkan kemacetan. Fluktuasi arus ini biasanya terjadi pada jam berangkat kerja dan pulang kerja. Sedangkan hambatan samping diantaranya berupa parkir di tepi jalan, penyebrang jalan, pedagang kaki lima dan angkutan umum yang berhenti di tepi jalan untuk mencari penumpang.

Pada data kecepatan dan waktu tempuh kendaraan, kecepatan dan waktu tempuh kendaraan dihitung antara dua titik persimpangan. Selain itu, pihak Dishub membagi menjadi enam bagian waktu pengukuran yaitu:

- Pagi sebelum *three in one* yaitu pukul 00.00 – 06.59 WIB
- Pagi saat *three in one* yaitu pukul 07.00 – 10.00 WIB
- Pagi setelah *three in one* yaitu pukul 10.01 – 11.59 WIB
- Sore sebelum *three in One* yaitu pukul 12.00 – 15.59 WIB
- Sore saat *three in one* yaitu pukul 16.00 – 20.00 WIB
- Sore setelah *three ini one* yaitu pukul 20.01 – 23.59 WIB

Dari data kecepatan dan waktu tempuh tersebut, selanjutnya penulis membuat sebuah pemodelan graf yang dapat merepresentasikan setiap persimpangan dan jaringan jalan yang berada pada wilayah penelitian. Dalam graf tersebut node merupakan persimpangan, sementara simpul merupakan jalur yang menghubungkan anatara dua persimpangan. Berikut ini pemodelan graf yang penulis maksud.



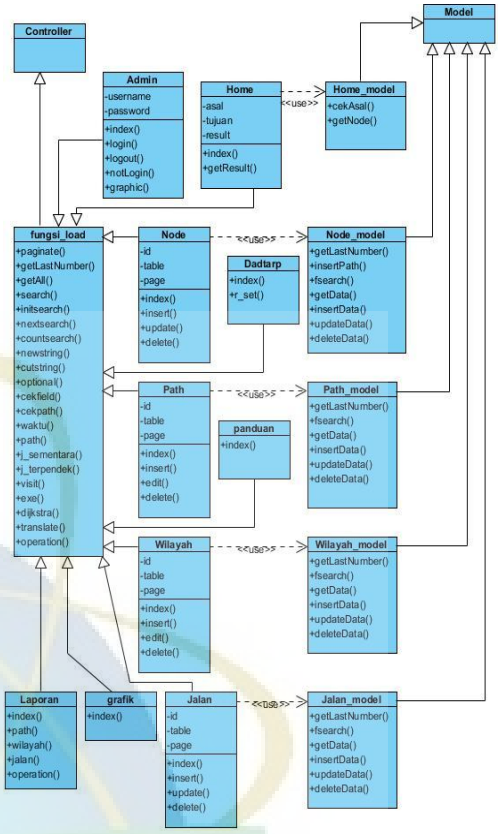
Pada Graf tersebut, nama persimpangan diwakili oleh nama node sedangkan nama jalur diwakili dengan penomoran pada tiap simpul. .

4.2. Perancangan Sistem

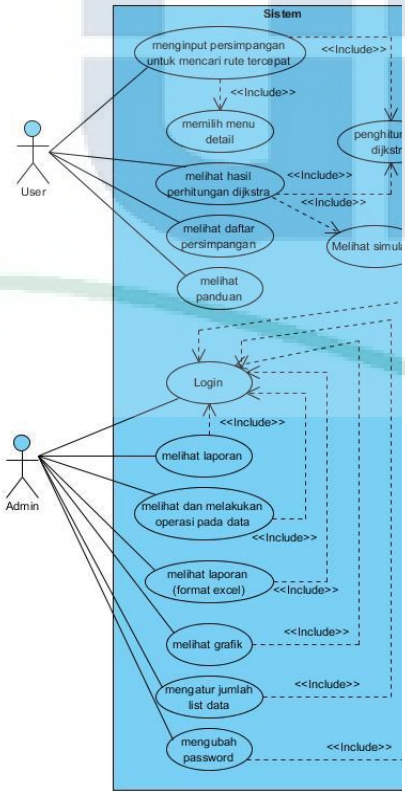
Berikut ini uraian tahap-tahap pengembangan sistem diantaranya *modeling, construction* dan *deployment*. Adapun tahap *communication* dan *planning* telah penulis uraikan pada bab sebelumnya.

4.2.1 *Modeling*

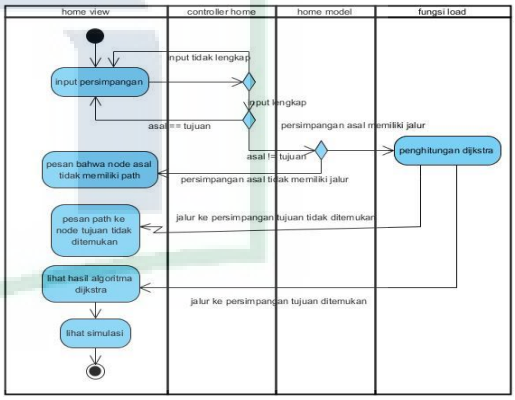
Pada bagian pemodelan, penulis menggunakan UML, yang terdiri dari *use case diagram, class diagram, activity diagram, sequence diagram* dan *deployment diagram.* Berikut ini beberapa diagram yang penulis buat diantaramya.



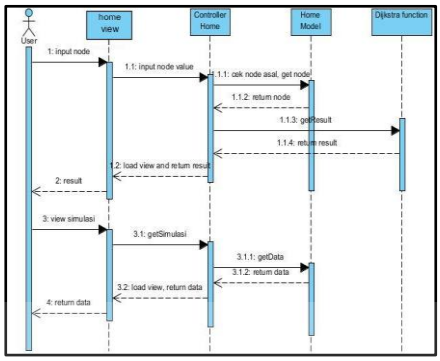
Class diagram sistem



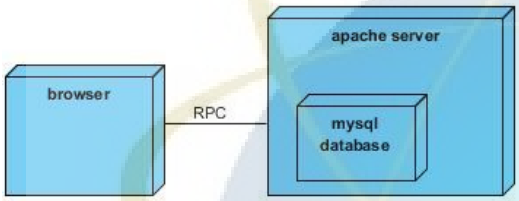
Use Case User dan Admin



Activity diagram pada hlaman home



Sequence diagram halaman home



Deployment diagram sistem

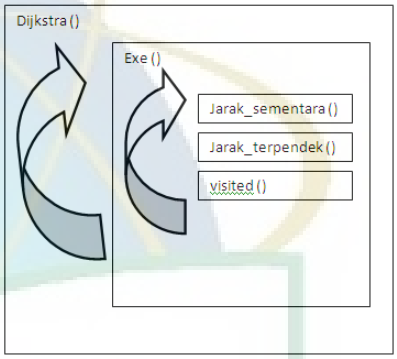
4.3.1 *Construction*

Dalam pembuatan *prototype* algoritma Dijkstra, penulis mengacu pada *pseudocode* algoritma Dijkstra yang sudah dijelaskan pada bab sebelumnya. Dalam hal ini penulis membuat fungsi-fungsi untuk melakukan setiap langkah pada algoritma Dijkstra. Penulis membuat lima buah fungsi yang terdiri dari tiga fungsi utama dan dua fungsi tambahan. Fungsi tambahan digunakan untuk melakukan iterasi pada fungsi utama. Berikut ini fungsi yang penulis gunakan pada untuk menjalankan algoritma Dijkstra.

1. Fungsi jarak sementara
2. Fungsi jarak terpendek
3. Fungsi visited
4. Fungsi exe
5. Fungsi dijkstra

Fungsi yang pertama kali dijalankan adalah fungsi dijkstra, fungsi ini menerima input masukan berupa sebuah graf dan dua buah node yaitu node asal dan node tujuan. Selanjutnya fungsi ini akan melakukan *looping* pada graf sampai node tujuan ditemukan, adapun dalam melakukan *looping* fungsi ini membentuk tiga buah variabel yang akan menjadi parameter untuk fungsi yang lainnya. Variabel tersebut yaitu jarak_sementara, jarak_terpendek dan visited.

Berikut ini gambaran penggunaan fungsi-fungsi tersebut pada jalannya algoritma Dijkstra.



4.4.1 *Deployment*

Pada tahap *deployment*, penulis menjalankan sistem yang telah dibangun pada *frame work* Code Igniter dengan menggunakan server apache dan database MySQL yang terdapat dalam satu bundel Xampp. Berikut rincian lengkapnya sebagai berikut adapun untuk *interface* hasil running sistem terdapat pada **http://rute-tercepat.co.cc**

Tabel 4.32 Spesifikasi Software untuk *Running* Sistem

Spesifikasi	Nama Software yang Digunakan
Bahasa pemrograman	PHP 5.2.2
Database	My SQL 5.0.41
Framework	CodeIgniter (CI) 1.7.2
Web Server	Xampp 1.3.2 (Win 32)

V KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan perumusan masalah dan serangkaian penelitian yang telah penulis lakukan. Maka dapat disimpulkan beberapa hal sebagai berikut.

1. Cara menemukan rute tercepat dan rute terpendek dari suatu lokasi ke lokasi yang lain, dalam hal ini berupa suatu persimpangan ke persimpangan yang lain adalah dengan mengunakan algoritma Dijkstra dengan input berupa persimpangan asal dan tujuan serta sebuah graf yang merepresentasikan node sebagai persimpangan dan simpul atau path sebagai jalur yang menghubungkannya.
2. Faktor-faktor yang dapat diperhitungkan dalam mencari jalur terceoat yaitu waktu tempuh dari suatu persimpangan ke persimpangan lainnya. Adapun faktor yang mempengaruhi waktu tempuh tersebut yaitu penumpukan arus lalu lintas dan hambatan samping seperti penyebrang jalan, pedagang kaki lima dan parkir ditepi jalan.

5.2. Saran

Berikut ini beberapa saran yang dapat dipergunakan untuk pengembangan penelitian dalam mencari rute tercepat dan terpendek.

1. Memperluas cakupan wilayah penelitian seperti meliputi kota Jakarta, Bogor, Depok, Tangerang dan Bekasi, sehingga dapat menjangkau kepentingan masyarakat yang lebih luas.
2. Menggunakan data yang *real time*, sehingga sangat akurat dalam menentukan rute tercepat pada saat sistem digunakan.

DAFTAR PUSTAKA

Ali, Mohammad. 2007. *Ilmu dan Aplikasi Pendidikan*. Grasindo. Jakarta: 359 hlm.

A.G Haryanto, Hartono Ruslijanto, Datu Mulyono. 2000. *Metode Penulisan dan Penyajian Karya Ilmiah: Buku Ajar Untuk Mahasiswa*. Penerbit Buku Kedokteran EGC. Jakarta.

Barata, Atep Adya. 2003. *Dasar-dasar Pelayanan Prima*. Elex Media Komputindo. Jakarta: 299 hlm.

Basuki, A. P. 2010. *Membangun Web Berbasis PHP Dengan Framework CodeIgniter*. Penerbit Lokomedia, Yogyakarta: xiii + 212 hlm.

Connolly, T. M. & C. E. Begg. 2002. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education Inc., London: xlix + 1236 hlm.

Chamero, Juan. 2006. *Dijkstra's Algorithm As a Dynamic Programming strategy*. http://www.intag.org/downloads/ds_006.pdf, 04 November 2010, pk. 14.48 WIB.

Dharwiyanti, S. & R. S. Wahono. 2003. Pengantar Unified Modeling Language (UML): 13 hlm. http://standy-oei.web.ugm.ac.id/ppl/MateriSuplem

enUml.pdf, 03 Agustus 2010, pk. 14.45 WIB.

Edmonds, Jeff. 2008. *How to Think About Algorithm.* Cambridge University Press. New York: xi + 439 hlm.

Goodrich, Michael T. & Roberto Tamassia. 2002. *Algorithm Design foundations, Analysis, and Internet Examples*. John Wiley & Sons, Inc. New York.

Heryandi, Andri. 2010. *Struktur Data Stack.* http://if.unikom.ac.id/andri/download/strukdat/STACK.pdf, 03 Januari 2011, pk 14.20 WIB.

Ibrahim, Ali. 2008. *Cara Praktis Membuat Website Dinamis Menggunakan XAMPP*. Neotekno, Yogyakarta: viii + 146 hlm.

Knuth. Donald E. 1973. *The Art of Computer Programming Volume 1*. Addison-Wesley Company, Inc.

Kristanto, Andri. 2003. *Struktur Data dengan C++*. Graha Ilmu, Yogyakarta: 386 hlm.

Kurniawan, Hendra. 2006. *Panduan Praktis Instalasi Email Server Gratis Berbasis Windows Menggunakan HMailServer*. Elex Media Komputindo, Jakarta: xi + 132 hlm.

Mata-Toledo, R. A. & P. K. Cushman. 2007. *Schaum's Outlines Dasar-Dasar Database Relasional*. Penerbit Erlangga, Jakarta: viii + 157 hlm.

Mulyadi. 2001. *Sistem Akuntansi Edisi ke 3*. Salemba Empat, Jakarta.

Munir, Rinaldi. 2005. *Algoritma dan Pemrograman dalam Bahasa Pascal dan C Buku 1*. Informatika, Bandung: x + 390 hlm.

Munir, Rinaldi. 2005. *Buku Teks Ilmu Komputer Matematika Diskrit Edisi Ketiga*. Informatika. Bandung: xii + 547 hlm.

Nader, J. C. 1992. *Prentice Hall's Illustrated Dictionary of Computing*. Prentice Hall Inc., New South Wales: viii + 526 hlm.

Nurhayati, Oky Dwi. 2010. *Dasar Algoritma*. http://eprints.undip.ac.id/18630/1/pertemuan2.pdf, 03 Januari 2011, pk. 15.40 WIB.

Peranginangin, Kasiman. 2006. *Aplikasi Web dengan PHP dan MySQL*. Andi. Yogyakarta.

Pressman, R. S. 2010. *Software Engineering: A Practitioner's Approach*. Mc Graw Hill Companies Inc., New York: xxviii + 895 hlm.

Purwanto, Eko Budi. 2008. *Perancangan dan Analisis Algoritma*. Graha Ilmu, Yogyakarta: x + 296 hlm.

Rahmat C, Antonius. 2010. *Struktur Data*. http://lecturer.ukdw.ac.id/~mahas/dossier/12_strukdat.pdf, 03 Januari 2011, pk 11.33 WIB.

Rumbaugh, J., I. Jacobson & G. Booch. 2006. *The Unified Modeling Language Reference Manual*. Pearson Education Inc., Westford: xiii + 721 hlm.

Sumanto. 1995. *Metodologi Penelitian Sosial dan Pendidikan*. Andi Offset, Yogyakarta.

Wardana. 2010. *Menjadi Master PHP dengan Framework Codeigniter*. Elex Media Komputindo, Jakarta: ix + 249 hlm.

Williams, B. K. & S. C. Sawyer. 2007. Using Information Technology: Pengenalan Praktis Dunia Komputer dan Komunikasi. Penerbit Andi, Yogyakarta: xxii + 586 hlm.

Wismakarma, Komang. 2010. *Panduan Lengkap Menguasai Pemrograman CSS*. Penerbit Lokomedia, Yogyakarta: xiv + 204 hlm.