PENGGUNAAN ALGORITMA *DIJKSTRA* DALAM PENENTUAN RUTE PERJALANAN WISATA DENGAN DUKUNGAN SISTEM INFORMASI GEOGRAFIS BERBASIS WEB

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh:

YOHANA SEKTY MARGIASIH
0310960079 – 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MIPA
UNIVERSITAS BRAWIJAYA
2007

PENERAPAN ALGORITMA *DIJKSTRA* DALAM PENENTUAN RUTE PERJALANAN WISATA DENGAN DUKUNGAN SISTEM INFORMASI GEOGRAFIS BERBASIS WEB

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh:

YOHANA SEKTY MARGIASIH
0310960079 – 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MIPA
UNIVERSITAS BRAWIJAYA
2007


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN TUGAS AKHIR

PENERAPAN ALGORITMA *DIJKSTRA*
DALAM PENENTUAN RUTE PERJALANAN WISATA
DENGAN DUKUNGAN SISTEM INFORMASI GEOGRAFIS
BERBASIS WEB

Oleh:
YOHANA SEKTY MARGIASIH
0310960079-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 20 Juli 2007
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Muh. Arif Rahman, M.Kom	**Yusi Tyroni Mursityo, S.Kom**
NIP. 131 971 481	**NIP. 132 318 423**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas
Brawijaya

Dr. Agus Suryanto, M.Sc
NIP. 132 126 049

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Yohana Sekty Margiasih.
NIM : 0310960079
Jurusan : Matematika
Penulis Tugas Akhir berjudul : Penerapan Algoritma *Dijkstra* Dalam Penentuan Rute Perjalanan Wisata Dengan Dukungan Sistem Informasi Geografis Berbasis Web.

Dengan ini menyatakan bahwa :

1. **Isi dari tugas Akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 20 Juli 2007
Yang menyatakan,

(Yohana Sekty Margiasih)
NIM. 0310960079

PENERAPAN ALGORITMA *DIJKSTRA*
DALAM PENENTUAN RUTE PERJALANAN WISATA
DENGAN DUKUNGAN SISTEM INFORMASI GEOGRAFIS
BERBASIS WEB

ABSTRAK

Kemacetan yang terjadi dalam setiap perjalanan sering mengganggu kegiatan sehari-hari atau dalam perjalanan wisata. Setiap wisatawan menginginkan sampai ke tempat tujuan tepat waktu untuk menghindari ketidaknyamanan karena kemacetan tersebut. Oleh karena itu, dibutuhkan suatu cara untuk menanggulangi kemacetan tersebut agar mencapai suatu tempat wisata dengan waktu yang lebih cepat, yaitu dengan mencari lintasan terpendek dari tempat asal ke tempat tujuan, dimana lintasan terpendek ini memperhitungkan waktu-waktu kemacetan yang sering terjadi.

Tujuan perancangan adalah membuat suatu perangkat lunak yang dapat memberikan informasi mengenai rute jalan dan urutan perjalanan wisata, total waktu dan total jarak antara hotel tempat mereka menginap dengan tempat-tempat wisata yang ada di kota Malang dan juga informasi mengenai hotel dan tempat wisata yang didapat dari peta dimana menggunakan *Geographical Information Systems* (GIS) yang merupakan salah satu solusi untuk mendapatkan informasi geografi tersebut. Perancangan ini dilakukan dengan menggunakan metode *Dijkstra* yang merupakan salah satu algoritma yang berguna untuk mencari solusi lintasan terpendek dari satu titik ke titik lain. Meskipun metode *Dijkstra* hanya mengeluarkan satu nilai output yang merupakan lintasan terpendek, namun dengan adanya variabel kemacetan akan didapatkan rute dengan waktu tersingkat agar masalah kemacetan tersebut dapat teratasi.

Program ini dapat membantu para wisatawan dalam melakukan perjalanan wisatanya, agar perjalanan yang akan dilalui menjadi lebih efektif dan efisien, serta dapat ditentukan pula semakin besar atau kecilnya total jarak tempuh dan total waktu tempuh yang dipengaruhi oleh kemacetan yang terjadi pada setiap lintasan dikarenakan waktu keberangkatan.

THE APPLICATION OF DIJKSTRA ALGORITHM IN TOUR JOURNEY ROUTE APPOINTMENT USING GEOGRAPHICAL INFORMATION SYSTEM BASED ON WEB

ABSTRACT

The traffic jam which often happened during the journey often disturbing in everyday activity or during tour journey. The tourist want to get to the place targeted punctually to avoid uncomfortable of theirs journey. Therefore, one needs an alternative way to overcome the troubles found during the traffic jam in order to be able to reach the targeted place faster. So, they try to find the shortest trajectory which is able to avoid the traffic jam.

The purpose of this project is to make a software which can give information accurately about the route and the sequence of journey, the whole time and distance between their inn and tourist destinations in Malang and also the hotels and tourist destinations found in the map that using Geographical Information System (GIS) usually use as one of the solutions in finding geographical information. The project is performed by using Dijkstra method. Dijkstra is one of the algorithms that is useful to obtain the solution of shortest trajectory from one point to other point. Although the Dijkstra method only brings out one of output values of the shortest trajectory, but with the existence of jam variable they will get the route in shortest time so the jam problems can be solved.

This program can help the tourist in their tour journey, so one's journey will more effective and efficient. Besides, they will able to determine the estimation of whole distance and time influenced by jam that happened in every trajectory because of the leaving time.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya, Tugas Akhir yang berjudul "**Penerapan Algoritma *Dijkstra* Dalam Penentuan Rute Perjalanan Wisata Dengan Dukungan Sistem Informasi Geografis Berbasis Web**" ini dapat diselesaikan. Tugas Akhir ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya yang baik dan suci dengan rahmat yang berkah-Nya menyelamatkan kita pada hari akhirat.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Muh. Arif Rahman, M.Kom selaku Pembimbing Utama. Terima kasih atas semua saran, bantuan, kritikan, waktu, dorongan semangat dan bimbingannya.
2. Yusi Tyroni M, S.Kom selaku Pembimbing Kedua. Terima kasih atas semua saran, bantuan, kritikan, waktu, dorongan semangat dan bimbingannya.
3. Drs. Achmad Ridok, M.Kom selaku Penasihat Akademik.
4. Wayan F. Mahmudy, S.Si, MT selaku Ketua Program Studi Ilmu Komputer Unibraw Malang.
5. Bapak Yoseph Maryono dan Ibu Subekti Sri Sugati, Supriyatun, Maria Susanti dan Thomas Astrio yang senantiasa berdoa dan memberi dukungan serta semangat.
6. Sahabat terdekatku Doni Rohmadi. Terima kasih atas semangat dan segala dukungan yang diberikan.
7. Sahabat-sahabat, Ruly, Santi, Ratna, yani. Terima kasih atas semangat yang diberikan.
8. Teman-teman satu kosan, Kholis, desti, risa, yanti, upin, putri, gandi, ayu, kholisina, sophie, anjar, mbak endah,

mbak nung, mbak nila, mbak yusi, mbak yuli, mbak vero, mbak rahma. Terima kasih atas segala bantuannya.

9. Semua teman-teman Ilmu Komputer angkatan 2003. Terima kasih atas semangat dan doanya.
10. Pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, Juli 2007

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa waktu terakhir ini informasi mengenai geografi semakin dibutuhkan oleh banyak pihak, misalnya untuk mengetahui jarak antara satu daerah dengan daerah lain, memberikan alternatif jalan dari satu daerah ke daerah lain, memberi informasi seputar daerah yang diinginkan, menemukan daerah yang memiliki sumber daya alam yang dicari, menemukan lokasi kecelakaan dengan cepat, mencari tempat perlindungan yang terdekat, dan masih banyak lagi informasi yang bisa didapatkan dari informasi mengenai geografi tersebut. Informasi-informasi seperti ini tentu saja sangat berguna bagi perusahaan-perusahaan yang bergerak di bidang ekspedisi, *travelling*, dan masih banyak perusahaan-perusahaan yang bergerak di bidang lain yang membutuhkan informasi tersebut.

Seiring dengan meningkatnya kebutuhan manusia tersebut, para wisatawan pun juga membutuhkan informasi yang akurat untuk mendapatkan rute terpendek maupun suatu urutan perjalanan wisata, yaitu dari satu tempat asal dimulainya perjalanan ke tempat-tempat wisata yang diinginkannya sebelum melakukan perjalanan wisata, agar waktu yang dibutuhkan menjadi lebih efisien.

Permasalahan penentuan rute dan urutan perjalanan wisata tersebut memungkinkan suatu solusi yang paling efektif untuk diterapkan dalam persoalan yang akan dihadapi oleh para wisatawan, yaitu kemacetan yang sering terjadi selama perjalanan. Kemacetan tersebut mengganggu perjalanan wisata mereka dan membuat ketidaknyamanan dalam perjalanan. Dibutuhkan suatu cara untuk menanggulangi gangguan tersebut, yaitu dengan cara mencari lintasan terpendek dari tempat wisata asal ke tempat wisata tujuan dengan mempertimbangkan waktu di mana kemacetan sering terjadi, serta bagaimana urutan tempat wisata yang akan dikunjungi jika wisatawan ingin mengunjungi beberapa tempat wisata dengan batasan waktu singgah pada setiap tempat wisata. Total waktu yang dibutuhkan dalam melakukan perjalanan ke tempat wisata yang akan dikunjungi wisatawan akan dapat diprediksi.

Telah terdapat banyak penelitian mengenai pencarian rute terpendek dengan berbagai macam metode dan pendekatan, seperti

penggunaan metode algoritma *Dijkstra* (Yulia dan Jeffrey, 2002), metode algoritma *Dijkstra* dan *Bellman-Ford* (Gilang, dkk., 2005), algoritma G*reedy* (Irvan, dkk., 2005), (Taufiq, dkk., 2006).

Berdasarkan penelitian tersebut dapat diketahui bahwa telah dikembangkan beberapa algoritma untuk memecahkan permasalahan penemuan solusi pencarian rute terpendek, salah satunya yang terkenal adalah algoritma *Dijkstra*. Penerapan strategi algoritma *Dijkstra* pada pencarian lintasan terpendek dapat menghasilkan nilai yang optimum karena dapat mengacu lebih dari satu aspek/bobot. Hasil analisis berdasarkan bobot-bobot yang berbeda, menunjukkan bahwa semakin banyak bobot yang diberikan, maka semakin akurat pula data yang dihasilkan, sehingga, menghasilkan waktu yang efisien (Ariyanto, dkk., 2006), di mana bobot yang terlibat dalam tugas akhir ini adalah jarak lintasan dan waktu terjadinya kemacetan.

Dari adanya permasalahan di atas, maka dalam tugas akhir ini akan dibangun suatu alat bantu sistem informasi pencarian rute terpendek dan urutan perjalanan tempat-tempat wisata berdasarkan jarak lintasan yang dipengaruhi waktu terjadinya kemacetan, yang informasinya terdapat pada peta yang disediakan oleh sistem. Sehingga, para wisatawan dapat memperoleh informasi yang jelas mengenai rute perjalanan yang akan ditempuh dan informasi mengenai tempat wisata yang terdapat pada peta tersebut.

Berdasarkan latar belakang yang telah dipaparkan, maka penulis menggambil judul pada tugas akhir ini dengan **"Penerapan Algoritma *Dijkstra* dalam Penentuan Rute Perjalanan Wisata dengan dukungan Sistem Informasi Geografis Berbasis web"**

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam tugas akhir ini perumusan masalahnya adalah:

1. Bagaimana para wisatawan merencanakan suatu perjalanan dari sebuah penginapan (hotel) ke tempat-tempat wisata dengan jarak lintasan dan waktu yang efektif dan efisien.
2. Berapa waktu dan jarak yang dibutuhkan wisatawan dalam melakukan perjalanan wisata dalam satu kali perjalanan dengan masukan beberapa tempat wisata tujuan.

1.3 Batasan Masalah

Dari permasalahan di atas, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Diasumsikan bahwa wisatawan memulai perjalanannya di hotel, karena sistem yang akan dibuat sementara ini ditujukan sebagai fasilitas yang diberikan hotel untuk para wisatawan yang menginap.
2. Jalan yang dilalui adalah jalan dua arah.

1.4 Tujuan Penelitian

Adapun tujuan yang ingin didapat dari penelitian masalah ini adalah:

1.4.1 Bagi Penulis

Menyediakan suatu aplikasi (*software*) yang berguna untuk menentukan jalan terpendek menuju suatu tempat di mana terdapat suatu variabel lain yang mempengaruhi serta urutan, jarak dan waktu perjalanan wisata yang dibutuhkan.

1.4.2 Bagi Pengguna (Wisatawan)

Menentukan perencanaan suatu rute terpendek yang dipengaruhi variabel kemacetan ke tempat wisata yang akan dikunjungi dan urutan perjalanan wisata.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini yaitu:

1. Membantu para wisatawan atau siapapun yang membutuhkan informasi dalam mencari jalan terpendek menuju suatu tempat wisata yang ingin dikunjungi, di mana terdapat suatu variabel kemacetan yang dapat mempengaruhi perjalanan tersebut
2. Serta dapat mengetahui total waktu, total jarak dan urutan perjalanan dari sistem tempat wisata mana dulu yang akan dilalui dengan memberikan masukan beberapa tempat wisata.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep pencarian rute tependek dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat alat bantu sistem informasi pencarian rute tempat – tempat wisata terdekat.
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Tugas akhir ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan algoritma pencarian jalur terpendek, graf dan sistem informasi yang digunakan untuk menampilkan informasi peta .
3. BAB III METODOLOGI DAN PERANCANGAN SISTEM
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam pembuatan aplikasi pencarian rute terpendek menggunakan algoritma *Dijkstra* berbasis web.
4. BAB IV IMPLEMENTASI DAN UJI COBA SISTEM
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa sistem perangkat lunak yang dibangun, yaitu

apakah hasil dari pencarian rute terpendek menggunakan graf menghasilkan jalur jalan yang informatif dan waktu efisien.

5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1 Konsep Sistem Informasi geografis (SIG)

GIS atau *Geographical Information System* adalah suatu sistem dalam komputer yang digunakan untuk menyimpan dan memanipulasi informasi tentang geografi. GIS didesain untuk koleksi, penyimpanan, dan analisa terhadap suatu objek serta suatu fenomena yang terjadi di mana letak geografi mempunyai suatu karakteristik yang penting atau yang dapat dianalisa. Data dari GIS dapat berasal dari peta, data yang berbentuk tabel, atau daftar nama dan alamat. GIS digunakan untuk menghasilkan informasi yang diperlukan oleh *user* atau klien. Klien dapat berupa perseorangan maupun beberapa orang atau ada kemungkinan anggota dari perusahaan umum, pemerintah ataupun industri khusus. Informasi dalam GIS dapat ditampilkan dalam dua bentuk dasar, yaitu peta dan tabel. Bentuk yang dipilih tergantung dari kebutuhan. Contohnya adalah untuk menunjukkan manakah tanah yang dapat digunakan atau tidak dapat digunakan, bentuk yang sesuai adalah bentuk peta. Sedangkan untuk menunjukkan informasi tentang berapa banyak sumber daya yang dapat dimanfaatkan dalam satu daerah tertentu, bentuk yang sesuai adalah bentuk tabel. Beberapa manfaat dari GIS adalah dapat mengetahui jarak antara satu daerah dengan daerah lain, memberikan alternatif jalan dari satu daerah ke daerah lain, memberi informasi seputar daerah yang diinginkan, menemukan daerah yang memiliki sumber daya alam yang dicari, menemukan lokasi kecelakaan dengan cepat, mencari tempat perlindungan yang terdekat, dan mencari daerah di mana suatu sumber daya alam yang diinginkan berada, dan masih banyak lagi informasi yang dapat diperoleh dengan menggunakan bantuan GIS tersebut (puslit.petra.ac.id).

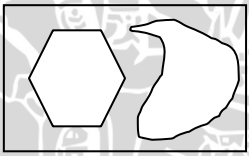
Kemampuan dasar dari SIG adalah mengintegrasikan berbagai operasi basisdata seperti *query*, menganalisisnya dan menyimpan serta menampilkannya dalam bentuk pemetaan berdasarkan letak geografisnya. Inilah yang membedakan SIG dengan sistem informasi lainnya (Aziz dan Pujiono, 2006).

2.1.1 Model Data SIG

Data yang ada di dalam SIG dikelompokkan menjadi 2 jenis data, yaitu data spasial dan data atribut (aspasial). Data spasial adalah data mengenai tata ruang (menyangkut titik koordinat). Data spasial terbagi atas 2 representasi *entity* spasial yang dalam penyimpanannya terbagi atas 4 macam tipe layer penyimpanan. Representasi *entity* yang dimaksud yaitu model *entity* data *raster* dan model *entity* data *vector*. Model *entity* data *raster* adalah model data yang berupa *image*. Model data *raster* akan disimpan dalam bentuk *grid*, dimana setiap *grid* mewakili data tertentu. Model data *vector* adalah model data yang didefinisikan dalam suatu bentuk garis, poligon, titik dan sejenisnya.

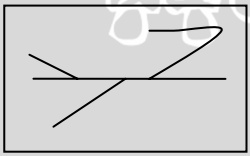
Layer penyimpanan dan pengolahan data spasial yang digunakan dalam SIG adalah sebagai berikut (Aziz dan Pujiono, 2006):

a. *Bounday* (poligon)
   Tipe data ini digunakan untuk mengolah data yang berbentuk luasan seperti yang terlihat pada gambar berikut. Contoh penggunaan poligon misalnya untuk menggambarkan gedung, persil, dan komplek bangunan.



Gambar 2.1 Poligon

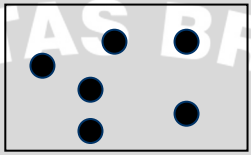
b. *Line* (garis)
   Tipe data *line* digunakan untuk pengolahan data yang berbentuk garis, seperti pada gambar di bawah ini. Berkas garis yang dimaksud adalah kenampakan geografis pada permukaan bumi, seperti jalan, sungai, jaringan, kabel dan sebagainya.



Gambar 2.2 Garis

c. *Point* (titik)

Tipe *point* digunakan untuk pengolahan data titik dan simbol untuk mewakili data pada posisi tersebut yang berisi tentang informasi titik-titik posisi, seperti terlihat pada gambar berikut. Contoh penggunaannya misalnya untuk melambangkan posisi hidran, posisi tempat sampah dan posisi ibukota suatu daerah pemerintahan.



Gambar 2.3 Titik

d. *Image* (gambar)

Tipe *Image* digunakan untuk memberikan informasi yang bersifat presentasi grafis. Contoh penggunaannya misalnya untuk legenda, skala, dan nama obyek.

2.1.2 *ToWKT Extension*

Bagian dari proyek GeoServer, ToWKT adalah suatu ekstensi dari *software* ArcView GIS yang digunakan untuk menyiapkan data untuk pemetaan *online* menggunakan *software* GeoClient.

GeoClient adalah suatu grafik "*client*" untuk data geografis yang diberikan oleh GeoServer, GeoServerLite, atau operasi *server* lainnya pada fitur *server* standar web Open GIS Consortium.

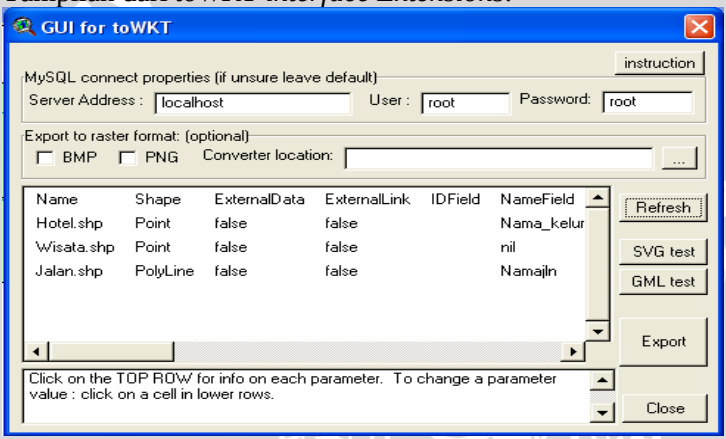
Misi dari proyek GeoServer adalah untuk memberi kemudahan dalam kemampuan pertukaran data geografis yang lebih baik dengan mengimplementasikan standar OpenGIS dan mengurangi kesalahan dalam memasukkan data geografis oleh pengguna.

ToWKT *eksport* data ke dalam database MySQL dalam format "well knwn text" yang didefinisikan dalam fitur sederhana OpenGIS untuk spesifikasi SQL. ToWKT juga dapat eksport data ke dalam dua format lainnya, yaitu Scaleable Vector Graphics (SVG) dan Geography Markup Language (GML).

Cara penggunaan dari ekstensi ini, antara lain:

- *Copy file* toWKT.avx ke dalam direktori *extension* ArcView 3.x (umumnya pada direktori ext32 pada *install path* dari ArcView 3.x)

- Pada ArcView, *buat project* baru (atau *project* yang sudah ada) dan buka ekstensi dari *File* > *Extensions* > *toWKT interface extension*. Ini akan menambah tombol berlian warna biru pada *View objects.*
- Menyiapkan *View* untuk *export*:
  - Buat *View*, atau buka yang sudah ada.
  - Pastikan *map units view* sudah diatur (pada *View* > *Properties Menu*).
  - Atur skala minimum atau maximum untuk beberapa *themes* yang akan ditampilkan (pada *Theme* > *Properties Menu*).
  - Aktifkan *theme* yang akan di *export.*
- Tampilan dari *toWKT interface Extensions*:



Gambar 2.4 *toWKT Interface Extensions*

- Klik tombol *export*. Kemudian akan diminta untuk memasukkan direktori penyimpanan hasil *export*, dan proses selesai.

(geoserver.sourceforge.net)

2.2 *Database* (Basis Data)

2.2.1 Pengertian *database*

Basis data adalah kumpulan data-data (file) *non redundant* yang saling terkait satu sama lainnya (dinyatakan oleh atribut-atribut kunci dan tabel-tabelnya/struktur data dan relasi-relasi) di dalam usaha membentuk bangunan informasi yang penting (*enterprice*) (prahasta, 2001).

Basis data juga dapat diartikan sebagai sekumpulan data store (bisa dalam jumlah yang sangat besar) yang tersimpan dalam *magnetic disk, optical disk, magnetic drum*, atau media penyimpanan sekunder lainnya (Al-Bahra, 2004).

2.2.2 *Structured Query Language* (SQL)

SQL merupakan bahasa *query* standar yang digunakan untuk mengakses basis data *relational*. Standardisasi internasional terhadap SQL pertama kali dilakukan oleh ANSI (*American National Standards Institution*), melalui publikasi *Database Language SQL* (ANSI X3.136-1986). Saat ini ANSI dan ISO (*International Standard Organization*) merupakan dua organisasi yang membuat standardisasi terhadap SQL.

Kemampuan SQL tidak terbatas hanya untuk *query* (memperoleh data), tetapi mencangkup kemampuan lain seperti berikut ini:

1. Pendefinisian struktur data
2. Pengubahan data
3. Pengaturan sekuritas, dll.

Sekalipun SQL dapat digunakan untuk mengakses basis data, namun SQL tidak menyediakan hal-hal seperti di bawah ini:

1. Pernyataan penguji kondisi (semacam *IF* pada COBOL).
2. Pernyataan pengulangan/*iterasi* (semacam *repeat* pada pascal).

Kelompok pernyataan SQL yang digunakan dalam tugas akhir ini, antara lain:

1. DDL (*Data Definition Language*).

   DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atibut-atribut basis data, tabel, atribut(kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Yang termasuk dalam kelompok DDL ini adalah *CREATE* (membuat), *ALTER* (memodifikasi) dan *DROP* (menghapus).
2. DML (*Data Manipulation Language*).

   DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan data. Perintah yang termasuk dalam kategori DML dapat dilihat pada tabel 2.1.

Tabel 2.1 *Data Manipulation Language*

No	Perintah	Keterangan
1	SELECT	Memilih data
2	INSERT	Menambah data
3	DELETE	Menghapus data
4	UPDATE	Mengubah data

Contoh DML diatas, dimana SQL dapat memberikan bahasa *query* tingkat tinggi dengan struktur sederhana dan kosakata dan gramatika yang sederhana pula, seperti di bawah ini:

Selection $A_1, A_2, ..., A_n$

From $T_1, T_2, ..., T_n$

Where P (2.1)

Dimana:

$A_1, A_2, ..., A_n$ himpunan dari setiap atribut yang akan ditampilkan.

$T_1, T_2, ..., T_n$ himpunan dari semua tabel yang diperlukan dalam *query*.

P kriteria yang diinginkan tentang informasi yang dicari. (Al-Bahra, 2004)

Sebagai contoh adalah terdapat tabel *Products*, dengan fieldnya adalah *prod_name*.

Untuk memanggil kolom tunggal yang disebut *prod_name* dari dalam tabel *Products*. Nama kolom yang diinginkan ditentukan di sebelah kanan setelah *keyword* SELECT, dan *keyword* FROM menentukan nama tabel dari mana data diambil.

Input

SELECT *prod_name*
FROM *Products*

Output

Prod_name

Fish bean bag toy
Bird bean bag toy
Rabbit bean bag toy

8 inch teddy bear
12 inch teddy bear
18 inch teddy bear
Rageddy Ann
(Forta, 2000)

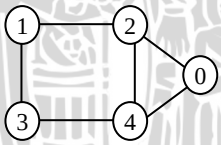
2.3 Graf

Graf adalah suatu diagram yang memuat informasi tertentu jika diinterpretasikan secara tepat (Jong, 2002). Suatu graf G terdiri dari 2 himpunan yang berhingga, yaitu himpunan titik-titik tidak kosong(V(G)) dan himpunan garis-garis (E(G)). Setiap garis berhubungan dengan satu atau dua titik. Dua titik dikatakan berhubungan (*adjacent*) jika ada garis yang menghubungkan keduanya.

2.3.1 Jenis – jenis Graf

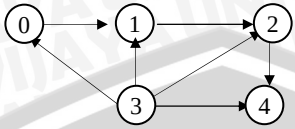
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak berarah (*undirected graph*)
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak berarah. Pada graf tak berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, $(v_j, v_k) = (v_{k,} v_j)$ adalah sisi yang sama.



Gambar 2.5 Graf tak berarah

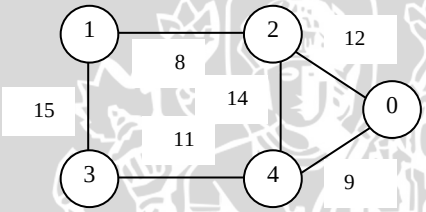
2. Graf berarah (*directed graph* atau *digraph*)
   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Pada graf berarah, (v_j, v_k) dan (v_k, v_j) menyatakan dua buah busur yang berbeda, dengan kata lain $(v_j, v_k) \neq (v_k, v_j)$. Untuk busur (v_j, v_k), simpul v_j dinamakan simpul asal (*initial vertex*) dan simpul v_k dinamakan simpul teminal (*terminal vertex*).



Gambar 2.6 Graf berarah

2.3.2 Graf Berbobot (*weighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat menyatakan jarak antara dua buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya (Munir, 2003).



Gambar 2.7 Graf berbobot

2.3.3 Representasi Graf

- **Representasi Adjacency-matrix**

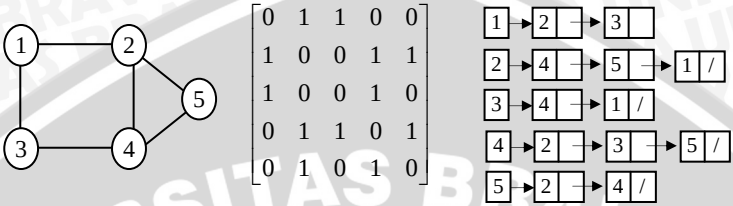
  Misalkan G adalah graf tak berarah dengan titik-titik $v_1,v_2,..v_n$(n berhingga). *Adjacency* matriks yang sesuai dengan graf G adalah matriks A = (a_{ij}) dengan a_{ij} = jumlah garis yang menghubungkan titik v_i dengan v_j, dimana i, j=1,2,..,n

$$ai, j = \begin{cases} 1 & \text{jika } (i, j) \in E \\ 0 & \text{jika tidak} \end{cases} \qquad (2.2)$$

- **Representasi Adjacency-list**

  Sebuah graf G terdiri dari sebuah *array Adj list* sebanyak jumlah titik, satu untuk setiap titik dalam V. Untuk setiap u∈V, *adjacency list* Adj[u] berisi *pointer-pointer* ke semua titik v sedemikian hingga ada sebuah *edge*(*u*,*v*) ∈ E. Adj[u] terdiri dari

semua titik terhubung ke u di G. Titik di masing-masing *adjacency list* disimpan sesuai urutan abjad.



Gambar 2.8 Representasi Graf tak berarah dalam *Adjancency-matrix* dan *Adjacency-list*

Gambar 2.9 Representasi Graf berarah dalam *Adjancency-matrix* dan *Adjacency-list*

Keterangan untuk gambar 2.8 dan 2.9:

Angka 1	:	Menunjukkan ada hubungan antara satu node dengan node yang lain.
Angka 0	:	Menunjukkan tidak ada hubungan antara node yang satu dengan node yang lain.

Dapat dilihat pada persamaan 2.2.

2.4 Algoritma *Shortest Path*

Merupakan algoritma yang digunakan untuk mencari jalur terpendek di antara 2 titik. Dalam permasalahan *shortest path*, diberikan sebuah graf berarah dan berbobot. G=(V,E), dengan fungsi bobot $w : E \rightarrow \mathbf{R}$ yang memetakan garis ke bilangan real (nilai dari bobot). Bobot dari path $p = v_0,v_1,\ldots..,v_k$ adalah jumlah dari bobot garis yang terpilih.

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \quad (2.3)$$

Didefinisikan bobot jalur terpendek antara u dan v dengan :

$$d(u,v) = \begin{cases} \min\{w(p) : u \to v\}, & \text{jika terdapat jalur antara u dan v} \\ \infty, & \text{jika tidak} \end{cases} \quad (2.4)$$

Terdapat beberapa algoritma untuk pencarian jalur terpendek, antara lain: *Dijkstra, Floyd Washall, Bellman Ford.*

- **Algoritma *Dijkstra***

Algoritma yang ditemukan oleh Dijkstra untuk mencari path terpendek merupakan algoritma yang lebih efisien dibanding algoritma *Warshall*, meskipun implementasinya juga lebih sukar (Jong, 2002).

Misalkan G adalah graf berarah berlabel dengan titik – titik V(G) = {v1, v2, v3..., vn} dan path terpendek yang dicari adalah dari v1 ke vn. Algoritma Dijkstra dimulai dai titik v1, dalam iterasinya, algoritma akan mencari satu titik yang jumlah bobotnya dari titik 1 terkecil. Titik – titik yang terpilih dipisahkan, dan titik –titik tersebut tidak diperhatikan lagi dalam iterasi berikutnya.

Misalkan:

V(G) = {v1, v2,...., vn}

L = Himpunan titik- titik $\in$ V(G) yang sudah terpilih dalam alur path terpendek.

D(j) = Jumlah bobot path terkecil dari v1 ke vj

W(i,j) = Bobot garis dari titik vi ke titik vj

W*(1,j) = jumlah bobot path terkecil dari vi ke vj

Secara formal, algoritma *Dijkstra* untuk mencari path tependek adalah sebagai berikut:

1. L = {}
   V = $\{ v_2, v_3, ..., v_n \}$
2. untuk i = 2,...., n, lakukan
   D(i) = w(1,i)
3. selama $v_n \notin L$ lakukan:
   a. pilih titik $v_k \in V - L$ dengan D(k) terkecil

L = L $\cup$ { v_k }

b. untuk setiap $v_j \in V - L$ lakukan:

jika D(j) > D(k) + W(k,j) maka ganti
D(j) = > D(k) + W(k,j)

4. untuk setiap $v_j \in V, W^*(1, j) = D(j)$ (2.5)

Menurut algoritma di atas, path terpendek dari titik v1 ke vn adalah melalui titik – titik di dalam L secara berurutan, dan jumlah bobot path terkecilnya adalah D(n).

2.5 *Flowchart*

Flowchart digunakan karena dapat membantu (mempermudah) programmer dalam mendesain program, sebagai spesifikasi program, sebagai alat verifikasi dan sekaligus untuk dokumentasi program. Dalam proses desain, *flowchart* dapat membantu memecahkan persoalan yang cukup kompleks kedalam serangkaian instruksi Dalam proses verifikasi, *flowchart* lebih mudah diperiksa oleh seorang *quality control* (QC) dari pada langsung memeriksa *source code* (instruksi-instruksi) program, atau *flowchart* dapat mempermudah pekerjaan QC tersebut dalam pemeriksaan kualitas program.

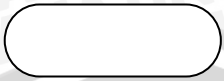
Flowchart dapat digunakan juga sebagai dokumen spesifikasi proses dalam pembuatan Data Flow Diagram (www.cic.ac.id).

Pengertian *flowchart* adalah diagram yang menunjukkan alur kontrol eksekusi program. Sebuah flowchart digambarkan dalam bentuk simbol – simbol tertentu. Dengan menampilkan ilustrasi menggunakan simbol – simbol tersebut, penggunaan flowchart memudahkan orang untuk mempelajari alur logika sebuah program.

Adapun simbol – simbol yang umumdigunakan dalam pembuatan flowchart adalah sebagai berikut:

- *Terminator*

  Digunakan untuk mengawali dan mengakhiri alur flowchart. Untuk mengawali , dituliskan *start* didalamnya dan *end* pada akhir program. Sedangkan untuk mengawali dan mengakhiri sebuah modul atau suatu bagian dari program, digunakan istilah *enter* dan *return*. Gambar 2.10 adalah simbol untuk sebuah terminator.



Gambar 2.10 Simbol *Terminator*

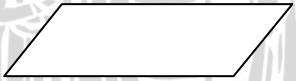
- *Preparation*
  Digunakan untuk penginisialisasikan variabel, *record* atau *field* lainnya yang dipelukan program. Biasanya diletakkan sesudah *terminator start* dan sebelum manjalankan alur program. Berikut gambar simbol *preparation:*

Gambar 2.11 Simbol *Preparation*

- *Process*
  Digunakan untuk menampilkan proses sedehana, seperti penambahan, pembagian dan sebagainya. Gamba simbol untuk *process*:

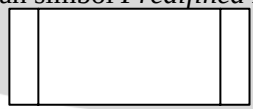
Gambar 2.12 Simbol *Process*

- *Input/Output*
  Digunakan untuk mewakili segala proses yang melibatkan *input* dan *output* data, seperti meminta inputan dari *user*.
  Gambar simbol untuk *input/output*:



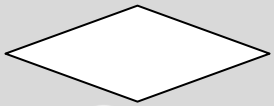
Gambar 2.13 Simbol *Input/output*

- *Predifined Process*
  Digunakan untuk memanggil prosedur yang terpisah dari *flowchart* utama. Nama prosedur ditulis pada simbol ini dan prosedur ini dijabarkan menjadi *flowchart* tersendiri. Gambar 2.14 adalah simbol *Predifined Process*.



Gambar 2.14 Simbol *Predifined Process*

- *Decision*
  Digunakan untuk menggambarkan percabangan (*Branching*) dalam alur program. Percabangan terjadi bila ada dua atau lebih kondisi yang mungkin terjadi. simbol *Decision:*



Gambar 2.15 Simbol *Decision*

- *Connector*
  Digunakan untuk menghubungkan ujung-ujung alur program dai percabangan. Simbol *Connector:*

Gambar 2.16 Simbol *Connector*

- *Arrow*
  Digunakan untuk menggambarkan arah aliran program antara dua simbol. Simbol *Arrow:*

Gambar 2.17 Simbol *Arrow*

2.6 Penelitian yang telah dilakukan

2.6.1 Perencanaan rute perjalanan di jawa timur Dengan dukungan gis menggunakan Metode dijkstra's

Penelitian yang dilakukan oleh Yulia Kartika Gunadi dan Jeffrey Tanuhardja pada tahun 2002, dengan menggunakan dukungan gis dan metode algoritma *Dijkstra* dalam menentukan rute yang mana hanya mengeluarkan satu nilai output yang merupakan lintasan terpendek yang tidak dipengaruhi oleh variabel lain seperti kemacetan yang sering terjadi. Program yang dibuat dalam penelitian ini juga tidak memiliki kemampuan untuk memberikan jalan lain yang dapat dilalui.

2.6.2 Analisis Algoritma Pencarian Rute Terpendek Dengan Algoritma Dijkstra dan Bellman-Ford

Penelitian yang dilakukan oleh Gilang, Agam dan Yahdi pada tahun 2005 adalah pencarian rute terpendek dengan membandingkan

algoritma *Dijkstra* dengan algoritma Bellman-Ford yang juga merupakan algoritma yang cukup banyak dipakai dalam permasalahan pencarian rute terpendek.

Baik algoritma dijkstra maupun bellman-ford sama-sama digunakan untuk mencari lintasan terpendek. Namun, tidak seperti algoritma dijkstra, algoritma bellman-ford dapat digunakan pada graf yang mengandung simpul negatif, selama graf tersebut tidak mengandung kalang negatif yang dapat dicapai dari titik awal. Algoritma dijkstra lebih menguntungkan dari sisi *running time* , namun untuk permasalahan khusus yang mengandung simpul negatif, algoritma bellman-Ford lah yang lebih menguntungkan.

2.6.3 Algoritma *Greedy* untuk Menentukan Lintasan Terpendek

Penelitian yang dilakukan oleh Irvan, Boyke dan Christoforus pada tahun 2005 Menggunakan Algoritma *greedy* untuk mencari lintasan terpendek dapat dirumuskan dengan:

1. Memeriksa semua sisi yang langsung bersisian dengan simpul *a*. Pilih sisi yang bobotnya terkecil. Sisi ini menjadi lintasan terpendek pertama, sebut saja L(1).
2. Tentukan lintasan terpendek kedua dengan cara berikut:
   (i) hitung: $d(i)$ = panjang $L(1)$ + bobot sisi dari simpul akhir $L(1)$ ke simpul i yang lain
   (ii) pilih $d(i)$ yang terkecil kemudian membandingkan $d(i)$ dengan bobot sisi (a, i). Jika bobot sisi (a, i) lebih kecil daripada $d(i)$, maka $L(2) = L(1)$ U (sisi dari simpul akhir $L(i)$ ke simpul i).
3. Dengan cara yang sama, ulangi langkah diatas untuk menentukan lintasan terpendek berikutnya.

2.7 PHP

PHP (atau resminya PHP: *Hypertext Preprocessor*) adalah skrip bersifat *server – side* yang ditambahkan kedalam HTML. PHP sendiri merupakan singkatan dari *Personal Home Page tools*. Skrip ini akan membuat suatu aplikasi dapat diintegrasikan ke dalam HTML sehingga suatu halaman web tidak lagi bersifat statis, namun menjadi bersifat dinamis. Sifat *server – side* berarti pengerjaan skrip

akan dilakukan di *server*, baru kemudian hasilnya dikirimkan ke *browser*,

Keunggulan dari sifatnya yang *server – side* tersebut antara lain:

- Tidak diperlukan kompatibilitas *browser* atau harus menggunakan *browser* tertentu, karena serverlah yang akan mengerjakan skrip PHP. Hasil yang dikirimkan kembali ke *browser* umumnya bersifat text atau gambar saja sehingga pasti dikenal oleh *browser* apapun.
- Dapat memanfaatkan sumber – sumber aplikasi yang dimiliki oleh *server*, misalnya koneksi ke database.
- Kode yang menyusun program tidak perlu diedarkan ke pemakai sehingga kerahasiaan kode dapat dilindungi (skrip tidak dapat "diintip" dengan menggunakan fasilitas *view* HTML *source*).
- PHP bisa juga digunakan untuk mengakses berbagai macam database seperti Access, Oracle, MySQL, dan lain – lain.

(Yahya, 2002).

2.7.1 Struktur PHP

Dengan menggunakan *notepad*, salah satu struktur PHP adalah sebagai berikut:

```
<html><head><title>Skrip PHP</title></head><body>
<?php
      Print ("selamat belajar PHP<br>");
      Print ("Semoga sukses");
?>
</body></html>                                    (2.6)
```

2.7.2 Variabel pada PHP

Variabel berfungsi untuk menyimpan suatu nilai dan nilai yang ada didalamnya dapat diubah sewaktu-waktu.

Dalam membuat suatu nama variabel, nama yang dipilih harus memenuhi aturan pengenal. Pengenal (*identifier*) banyak digunakan dalam program untuk memberi nama variabel, fungsi atau kelas. Aturan yang berlaku untuk pengenal:

1. Karakter yang dapat digunakan yaitu huruf, angka atau garis bawah (_).
2. Karakter yang pertama harus berupa huruf atau garis bawah.

3. Panjang pengenal bisa berapa saja.
4. huruf kecil dan huruf kapital dibedakan.

Variabel pada PHP selalu diawalai dengan tanda $.

2.7.3 Bagaimana dapat menjalankan PHP

Php dapat dijalankan dengan:

- PHPTriad (Apache, PHP, MySQL)
- Sokkit
- IIS dan Modul PHP
- Apache dan Modul PHP
- Konfigurasi Apache pada file httpd.conf
- Konfigurasi PHP pada file php.ini

2.8 MySQL

Pada dasarnya, pengolahan data tidak harus menggunakan sebuah *database*, pengolahan data juga bisa menggunakan *file* teks, tetapi penyimpanan data semacam ini memiliki banyak keterbatasan karena *file* teks tidak memiliki kemampuan untuk mengolah data, misalnya menghitung total nilai, rata-rata dan lain-lain. Pencarian data pun akan sulit dilakukan dengan semakin besarnya ukuran *file* '.txt'. untuk itulah *database* diperlukan. Adanya *database*, program akan lebih mudah mengendalikan akses terhadap data.

MySQL yang berfungsi sebagai database/penyimpanan data, hal ini tentu amat penting agar PHP dapat menjalankan fungsi pengolahan data otomatis seperti dicontohkan di atas.

MySQL merupakan salah satu jenis *database server* yang sangat terkenal. Kepopulerannya disebabkan MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya. Selain itu, ia bersifat *free*(anda tidak perlu membayar untuk menggunakannya) pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware* atau Anda perlu membayar setelah melakukan evaluasi dan memutuskan untuk digunakan dalam keperluan produksi).

MySQL termasuk jenis RDBMS (*Relational Database Management System*). Itulah sebabnya istilah seperti tabel, baris, dan kolom digunakan pada MySQL. Pada MySQL, sebuah database mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom.

Ada dua cara yang dilakukan untuk mengaktifkan MySQL, melalui MS_DOS dan melalui *PHPMyadmin*.

Ada beberapa tipe data yang dikenal dalam MySQL (misalnya dalam membuat tabel), yaitu:

1. Tipe data numerik

   Tipe data ini dibedakan menjadi 2, yaitu tipe *integer* (untuk bilangan bulat) dan tipe data *floating point* (untuk bilangan desimal).
2. Tipe data String

   Pada umumnya berbentuk teks.
3. Tipe data tanggal

   Berupa date, datetime, time, timestamp, year.

(www.sanctaursula-jkt.sch.id)

2.9 Kelajuan Rata-rata

Kelajuan rata-rata didefinisikan sebagai perbandingan jarak total yang ditempuh terhadap waktu total yang dibutuhkan.

$$Kelajuan\ rata-rata = \frac{jaraktotal}{waktutotal} \tag{2.7}$$

Satuan SI kelajuan rata-rata adalah meter per sekon (m/s), dan satuan yang lazim di Amerika adalah feet per sekon (ft/s). Satuan kelajuan yang sehari-hari dikenal di Amerika Serikat adalah mil per jam (mi/jam). Secara internasional, satuan yang lebih umum adalah kilometer per jam (km/jam). Jika anda menempuh 200 km dalam 5 jam, kelajuan rata-rata anda adalah (200 km)/(5 jam)=40 km/jam. Kelajuan rata-rata tidak menceritakan apa-apa tentang rincian perjalanan itu. Anda mungkin berkendaraan dengan kelajuan tetap 40 km/jam selama 5 jam itu, atau Anda mungkin berkendaraan lebih cepat selama sebagian waktu dan lebih lambat selama sisa waktunya, atau Anda mungin telah berhenti untuk satu jam dan kemudian berkendaraan dengan kelajuan yang berubah-ubah selama 4 jam yang lain (Tipler, 1991).

2.10 *Browser* dan *Web Server*

Dalam dunia web, perangkat lunak *client*, yaitu *browser* web mempunyai tugas yang sama yaitu menterjemahkan informasi yang diterima dari *sever* web dan menampilkannya pada layar komputer pengguna. Umumnya *browser* web menerima data dalam bentuk HTML. File HTML sebenarnya adalah file tekt biasa yang selain berisi informasi yang hendak ditampilkan kepada pengguna, juga mempunyai perintah-perintah untuk mengatur tampilan data tersebut. *Browser*lah yang memiliki kuasa penuh dalam menterjemahkan perintah-perintah tadi.

Web server pada dasarnya adalah perangkat lunak khusus yang bertugas melayani permintaan – permintaan dari browser web akan dokumen – dokumen yang tesimpan di dalamnya. Perangkat lunak web server sekarang telah tersedia untuk berbagai macam *platform* dan lingkungan sistem operasi untuk lingkungan Unix, yang paling populer adalah *Apache*, *Netscape FastTrack*, dan NCSA HTTPD . Sedangkan unutk lingkungan windows tersedia *Microsoft Internet Information Server* (IIS), *Netscape FastTrack O'reilly Website*.

Beberapa perangkat lunak *server* web mempunyai *feature* seperti *server-side programming*, *security control* dan lain sebagainya. Meskipun beragam macamnya, secara fungsional semua jenis *server* web adalah sama saja, yaitu berfungsi melayani permintaan-permintaan dari *browser* web (www.ikc.cbn.net.id).

2.10.1 *Web Server Apache*

Dapat berjalan pada platform Windows dan Linux
Software yang bersifat open source
Versi stabil Apache 2.0.47
Merupakan Web server yang paling terkenal
Efisien
Portability

2.10.2 Mengakses *Web Servers*

Request documents dari Web servers.
Nama Host.
Local Web servers.

- Mengakses komputer lokal dengan localhost

Remote Web servers.

- Mengakses dengan nama komputernya

Domain name atau Internet Protocol (IP) address.

- Domain name server (DNS)

Komputer yang mengatur database dari host names dan IP address yang bersangkutan.

2.10.3 *Apache & mySQL Running*

Jika *apache* dijalankan akan ditampilkan tampilan seperti pada gambar 2.18.

Gambar 2.18 *Apache & mySQL Running*

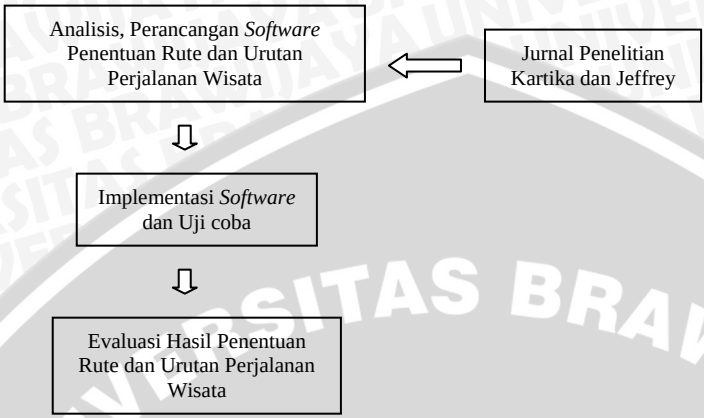
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian penentuan rute dan urutan perjalanan wisata yang efektif dan efisien.

Penelitian dilakukan dengan tahapan – tahapan berikut ini :

1. Mempelajari algoritma yang digunakan pada sistem penentuan rute terpendek yang telah ada sebelumnya, yaitu algoritma *Dijkstra*.
2. Menganalisa dan merancang perangkat lunak penentuan rute dan urutan perjalanan wisata yang akan dilalui dengan analisis pencarian rute terpendek berdasarkan jarak lintasan, jarak lintasan dengan tidak mempertimbangkan waktu terjadinya kemacetan dan jarak lintasan dengan mempertimbangkan waktu terjadinya kemacetan.
3. Pada analisa hasil juga akan dilakukan uji coba dengan membandingkan ketiga analisa yang terdapat pada *point* 2 yaitu terhadap hasil nilai total waktu dan total jarak terpendek, yang ditujukan untuk mengetahui, apakah dengan analisa pencarian lintasan tependek yang dilakukan dapat menghasilkan jalur tersingkat, apabila mempertimbangkan variabel kemacetan.
4. Uji coba perangkat lunak yang telah dibuat dengan menjalankannya di *browser*.

Langkah – langkah penelitian ini dapat digambarkan pada gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.1 Deskripsi Umum Sistem

Sistem yang akan dibuat merupakan penentuan rute dan urutan perjalanan wisata yang dapat mempermudah wisatawan untuk mengetahui keseluruhan waktu dan jarak yang diperlukan untuk melakukan perjalanan wisata ke berbagai tempat wisata dalam suatu kota yang diinginkannya, urutan kunjungan yang akan dilakukan dan rute perjalanan yang akan dilalui, sehingga segala sesuatu yang akan dilakukan menjadi lebih efektif dan efisien. Dengan suatu *interface* yang dapat mempermudah *user* karena ditampilkan pada *browser* (internet) dan *user* juga dapat memperoleh informasi geografis mengenai tempat wisata tersebut melalui peta yang telah disediakan.

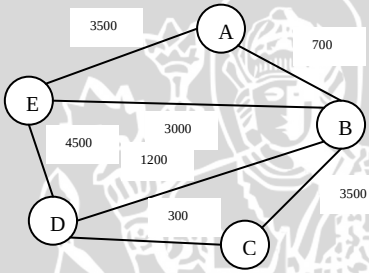
Metode yang digunakan pada sistem ini adalah dengan menggunakan Algoritma *Shortest Path* yaitu algoritma *Dijkstra*, yang merupakan salah satu algoritma yang berguna untuk mencari lintasan terpendek dari satu titik ke titik lain. Metode *Dijkstra* dipilih karena metode ini dapat mendukung dalam menghasilkan nilai total waktu maupun total jarak lintasan dan urutan kunjungan dan algoritma *Dijkstra* dapat diulang-ulang sebanyak n kali yang dimulai dari semua titik (Jong, 2002), algoritma *Dijkstra* juga lebih menguntungkan dari sisi *runing time* dibandingkan algoritma *Bellman-ford* (Gilang, dkk., 2005) serta berdasarkan bobot-bobot yang berbeda, menunjukkan bahwa semakin banyak bobot yang diberikan, maka semakin akurat pula data yang dihasilkan, sehingga menghasilkan waktu yang efisien (Ariyanto, dkk., 2006). Diharapkan

dengan analisis yang akan dilakukan dapat disimpulkan apakah dalam pencarian lintasan terpendek ini dapat menghasilkan jalan tersingkat menuju suatu tempat dengan adanya variabel lain yang mempengaruhi, dalam penelitian ini yaitu waktu kemacetan yang terjadi.

Penentuan rute dan urutan perjalanan wisata dibuat berdasarkan pada analisis yang dilakukan meliputi 2 aspek, yaitu:

1. Berdasarkan graf berbobot yang bobotnya merupakan panjang lintasan (jarak).
2. Berdasarkan graf berbobot yang bobotnya merupakan waktu terjadinya kemacetan .

Contoh analisis Algoritma *Dijkstra* untuk mencari lintasan terpendek:



Gambar 3.2 Penggambaran Graf Berbobot

Keterangan, dimisalkan:

A : Jl. Kahuripan
B : Jl. Tugu
C : Jl. Tumapel
D : Jl. Mojopahit
E : Jl. Brawijaya

Tabel 3.1 Panjang lintasan dari satu tempat ke tempat lainnya (dalam meter)

	A	B	C	D	E
A	0	700	0	0	3500
B	700	0	850	1200	3000
C	0	850	0	300	0
D	0	1200	300	0	4500
E	3500	3000	0	4500	0

Contoh persoalan : Carilah lintasan terpendek dai Jl Mojopahit menuju Jl. Brawijaya!

Dapat dilihat bahwa lintasan terpendek dari Jl. Mojopahit menuju Jl. Brawijaya adalah: D – C – B – E, yaitu sepanjang: 300*m*+850*m*+3000*m*=4500*m*.

Tabel 3.2 Waktu Kemacetan yang Paling Sering Terjadi

	A	B	C	D	E
A	0	Pagi, Sore	0	0	Siang
B	Pagi, Sore	0	Pagi, Sore	0	Siang
C	0	0	0	0	0
D	0	0	0	0	0
E	Siang	Siang	0	0	0

Jarak yang digunakan dari satu tempat ke tempat lainnya sama dengan jarak yang digunakan pada tabel 3.1.

Contoh persoalan: Carilah lintasan terpendek dari Jl Mojopahit menuju Jl. Brawijaya pada pukul 12.00!

Ditinjau dari tingkat kemacetan yang terjadi maka lintasan terpendek dari Jl Mojopahit menuju Jl. Brawijaya adalah: D – E = 4500*m.*

3.2 Batasan Sistem

Batasan dari sistem yang akan dikembangkan adalah:

1. Informasi peta diperoleh dari *software* Arcview 3.3 dengan ekstensinya yaitu toWKT *interface extension.*
2. Informasi koordinat peta pada layar diperoleh dari *software Handy Image Mapper*.
3. Pembuatan peta dalam bentuk *shapefile* pada *software* Arcview 3.3 tidak dijelaskan lebih lanjut dalam penentuan rute dan urutan perjalanan wisata.
4. Peta ditampilkan di halaman web dalam format *image*.

3.3 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan ini didasarkan pada pembangunan sistem penentuan rute dan urutan perjalanan wisata yang dapat memuaskan pengguna, dan informatif. Kebutuhan perangkat ini meliputi hal pokok yang harus ada dan tambahan.

Hal pokok :

- Sistem dapat mengidentifikasi semua masukkan dari *user*.
- Sistem dapat menghasilkan rute terpendek sesuai dengan algoritma yang telah ditentukan.
- Sistem dapat menghasilkan urutan perjalanan wisata yang diinginkan *user*.
- Sistem dapat menghasilkan total jarak perjalanan wisata.
- Sistem dapat menghasilkan total waktu/berapa jam perjalanan wisata diprediksi akan berlangsung.
- Sistem dapat memberikan informasi tempat wisata berdasarkan pilihan *user*.

Tambahan :

- Interaksi pengguna dengan sistem akan melalui GUI (*Graphical User Interface*) melalui web.
- Dapat dilakukan penambahan titik hotel maupun wisata pada peta.

3.4 Pengumpulan Data

Pengumpulan data merupakan usaha untuk memperoleh data atau dokumen yang dibutuhkan dalam penelitian dan untuk selanjutnya data tersebut akan diproses sesuai dengan kebutuhan.

3.4.1 Studi literatur

Studi literatur merupakan cara pengumpulan data yang diperoleh dengan mengumpulkan berbagai sumber kepustakaan, baik berupa buku-buku, jurnal, laporan penelitian, dan lain sebagainya untuk ditelaah lebih lanjut sebagai bahan pendukung penelitian.

3.4.2 Pengumpulan data lapangan

Metode ini digunakan untuk mengumpulkan data-data yang dibutuhkan dalam penelitian. Data-data yang dimaksud ialah data kemacetan, yaitu lokasi dan waktu. Sumber data yang digunakan ialah Dinas Perhubungan Kota Malang.

3.4.3 Dialog, diskusi, dan konsultasi

Metode ini dilakukan dengan cara melakukan konsultasi dengan pembimbing skripsi serta melakukan dialog maupun diskusi dengan sumber-sumber lain yang berhubungan dengan penelitian ini untuk memperoleh solusi pengambilan data dan metode pengujian yang efektif.

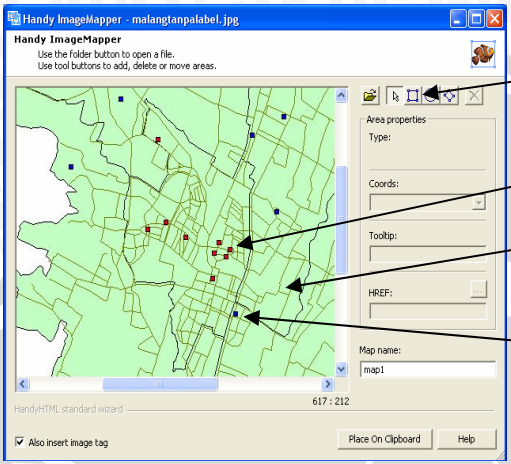
3.5 Perancangan Sistem

Berdasarkan analisis yang telah dilakukan, berikut ini akan dibahas mengenai proses yang terjadi pada sistem penentuan rute dan urutan perjalanan wisata**.** Penelitian ini terdiri dari beberapa tahap, yaitu proses menampilkan peta, rancangan basisdata, penentuan rute dan urutan perjalanan wisata, perancangan antarmuka, dan perancangan uji coba.

3.5.1 Proses Menampilkan Peta

Peta ditampilkan dengan memasukkan gambar peta di ArcView, kemudian menambah *new theme,* baik dijitasi pada *point, polygon* maupun *line.* Proses dijitasi tersebut, dikonversi menjadi *image* menggunakan fasilitas export yang ada pada Arcview GIS 3.3 ke dalam *type* JPEG.

Setelah dalam bentuk JPEG, proses selanjutnya menggunakan *software Handy Image Mapper*, untuk memperoleh koordinat pixel, agar dapat dipergunakan pada saat menampilkan informasi dengan meng-*click* titik pada peta, yang dapat dilihat pada gambar 3.3, kemudian diproses untuk di *load* pada halaman web.



Gambar 3.3 Pengolahan Koordinat Peta pada *Handy Image Mapper*.

3.5.2 Rancangan Basisdata

Dalam penelitian ini, untuk menghasilkan sebuah rute perjalanan wisata yang efektif dari satu tempat wisata ke tempat wisata lainnya, dibutuhkan data jarak antara satu titik dengan titik lainnya di dalam peta (jarak jalan), lokasi dan waktu kemacetan yang terjadi.

Tabel yang digunakan pada rancangan basisdata ini dibuat manual dan sebagian tabel hasil konversi software *toWKT interface Extension ArcView* ke *file SQL* (dapat dipilih atau disesuaikan dengan kebutuhan). Tabel – tabel hasil konversi adalah tabel wisata, tabel hotel dan tabel jalan, sedangkan tabel tambahan adalah tabel *graph*, tabel *temp*, tabel rute dan tabel rutemacet. Dimana database yang digunakan adalah database MySQL.

Tabel 3.3 adalah tabel yang digunakan untuk menyimpan data-data tempat wisata, antara lain:

Tabel 3.3 Tabel wisata

Field	Type	Allow Null
Id	Decimal(10.0)	No
Kd_jln	Decimal(10.0)	No
Nama	Varchar(30)	Yes
Jenis	Varchar(16)	Yes

Deskripsi	Varchar(254)	Yes
X	Double	No
Y	Double	No
Lokasi (pixel)	Varchar(30)	No
Cek	Int(2)	No

Keterangan :

Id	:	Id dari tempat wisata
Nama	:	Nama dari tempat wisata
Jenis	:	Jenis dari tempat wisata
Kd_jln	:	Kode untuk mengakses jalan tempat wisata berada
Deskripsi	:	Penjelasan singkat mengenai tempat wisata tesebut
X	:	Koordinat X titik wisata yang didapat dari *software arcview* 3.3
Y	:	Koordinat Y titik wisata yang didapat dari *software arcview* 3.3
Lokasi (pixel)	:	Menyimpan lokasi koordinat titik wisata untuk menampilkan informasi pada peta, dengan satuan pixel yang didapat dari *software Handy ImageMapper*.
Cek	:	Untuk menandai 1 jika tempat wisata telah ditampilkan pada peta

Tabel 3.4 adalah tabel yang digunakan untuk menyimpan data-data jalan, antara lain:

Tabel 3.4 Tabel Jalan

Field	Type	Allow Null
Kd_jln	Decimal(10.0)	No
Namajln	Varchar(20)	Yes
Xmin	Double	No
Ymin	Double	No
Xmax	Double	No
Ymax	Double	No
Fnode_	Decimal(10.0)	No
Tnode_	Decimal(10.0)	No
Length	Decimal(10.0)	No

Pagi	Int(5)	Yes
Siang	Int(5)	Yes
Sore	Int(5)	Yes

Keterangan :

Kd_jln	:	Kode dari setiap jalan
Namajln	:	Nama dari jalan
XMin	:	Koordinat XMin jalan
XMax	:	Koordinat XMax jalan
YMin	:	Koordinat Ymin jalan
YMax	:	Koordinat Ymax jalan
Fnode_	:	Penanda suatu titik terhubung (sebagai x)
Tnode_	:	Penanda suatu titik terhubung (sebagai y)
Length	:	Jarak jalan /lintasan dua titik
Pagi	:	Menandai jalan tertentu macet di pagi hari
Siang	:	Menandai jalan tertentu macet di siang hari
Sore	:	Menandai jalan tertentu macet di sore hari

Tabel 3.5 adalah tabel yang digunakan untuk menyimpan data-data hotel, sebagai tempat dimulainya para wisatawan melakukan perjalanan, antara lain:

Tabel 3.5 Tabel Hotel

Field	Type	Allow Null
Hotel_id	Decimal(10.0)	No
Kd_jln	Decimal(10.0)	No
Fasilitas_	Varchar(200)	Yes
Jumlah_kam	Varchar(50)	Yes
Nama	Vachar(50)	Yes
Alamat	Varchar(50)	Yes
Telepon	Varchar(50)	Yes
Kelas	Varchar(50)	Yes
X	Double	No
Y	Double	No
Lokasi (pixel)	Varchar(30)	No
Cek	Int(2)	Yes

Keterangan :

Hotel_id	:	Menyimpan id hotel
Kd_jln	:	Kode untuk mengakses jalan tempat wisata berada
Fasilitas_	:	Fasilitas yang disediakan oleh hotel
Jumlah_kam	:	Jumlah kamar yang disediakan oleh hotel
Nama	:	Nama hotel
Alamat	:	Alamat hotel
Telepon	:	Nomor telepon hotel
Kelas	:	Status kelas dari hotel
X	:	Koordinat X titik hotel yang didapat dari *software arcview* 3.3
Y	:	Koordinat Y titik hotel yang didapat dari *software arcview* 3.3
Lokasi (pixel)	:	Menyimpan lokasi koordinat titik wisata untuk menampilkan informasi pada peta, dengan satuan pixel yang didapat dari *software Handy ImageMapper*
Cek	:	Untuk menandai 1 jika tempat wisata telah ditampilkan pada peta

Tabel 3.6 adalah tabel yang digunakan untuk menyimpan data-data yang akan digunakan dalam penentuan rute, urutan, total jarak, dan total waktu, antara lain:

Tabel 3.6 Tabel *Graph*

Field	Type	Allow Null
Fnode_	Decimal(10.0)	No
Tnode_	Decimal(10.0)	No
Namajln	Varchar(30)	No
Waktu_lintasan	int(15)	No
Length	Decimal(10.0)	No
Pagi	Int(5)	Yes
Siang	Int(5)	Yes
Sore	Int(5)	Yes

Keterangan :

Fnode_	:	Penanda suatu titik terhubung (sebagai x)
Tnode_	:	Penanda suatu titik terhubung (sebagai y)
Namajln	:	Nama jalan yang dipakai sebagai rute yang akan dilalui
Waktu_lintasan	:	Waktu yang ditempuh untuk satu jalan
Length	:	Jarak suatu lintasan dua titik
Pagi	:	Menandai jalan tertentu macet di pagi hari
Siang	:	Menandai jalan tertentu macet di siang hari
Sore	:	Menandai jalan tertentu macet di sore hari

Tabel 3.7 adalah tabel yang digunakan untuk menentukan semua node yang digunakan (baik fnode_ maupun tnode_), antara lain:

Tabel 3.7 Tabel Temp

Field	Type	Allow Null
Node	Decimal(10.0)	No

Keterangan:

Node	:	Menyimpan node yang dipakai dalam perhitungan rute terpendek.

Tabel 3.8 adalah tabel yang digunakan untuk menyimpan rute yang dilalui dalam proses pembandingan yang hanya digunakan untuk proses penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan waktu kemacetan, antara lain terdapat pada tabel 3.8.

Tabel 3.8 Tabel Rute

Field	Type	Allow Null
No	Int(11)	No
Jalur	Varchar(20)	No
Tujuan	Int(11)	No
F	Int(11)	No
D	Int(11)	No
Pilih	Int(11)	No
Waktu	Int(11)	No
Jarak	Int(11)	No
Berangkat	Char(5)	No

Keterangan:

No	:	Menyimpan id setiap jalur/rute
Jalur	:	Menyimpan nama jalan dari rute yang dilalui
Tujuan	:	Menyimpan node tujuan
F	:	Menyimpan iterasi keberapa rute yang didapat
D	:	Menyimpan iterasi keberapa rute yang didapat
Pilih	:	Untuk menentukan apakah rute pertama yang dipilih atau rute kedua yang dipilih
Waktu	:	Menyimpan waktu perhitungan dengan kemacetan maupun tidak
Jarak	:	Menyimpan total jarak perhitungan dengan kemacetan maupun tidak
Berangkat	:	Menyimpan jam/waktu terakhir dari perjalanan, berdasarkan rute yang diambil

Tabel 3.9 adalah tabel yang digunakan untuk menyimpan rute dari hasil perhitungan berdasarkan jarak lintasan dan jarak lintasan tanpa mempertimbangkan kemacetan, antara lain:

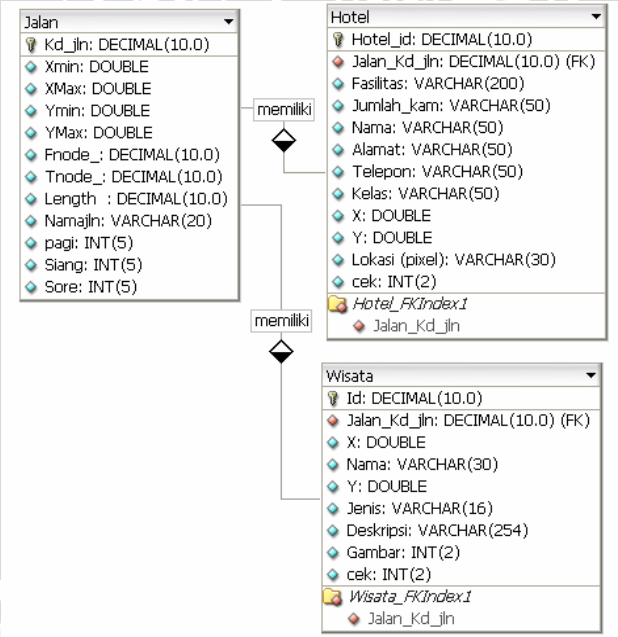
Tabel 3.9 Tabel Rutemacet

Field	Type	Allow Null
No	Int(11)	No
Jalur	Varchar(20)	No
Tujuan	Int(11)	No
F	Int(11)	No
D	Int(11)	No

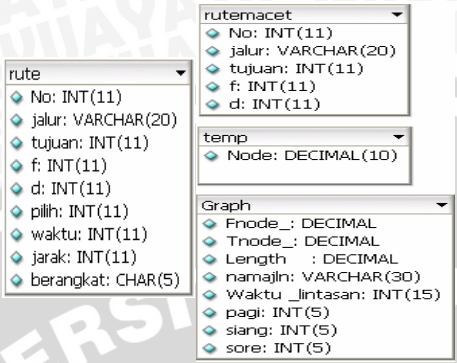
Keterangan:

No	:	Menyimpan id setiap jalur/rute
Jalur	:	Menyimpan nama jalan dari rute yang dilalui
Tujuan	:	Menyimpan node tujuan
F	:	Menyimpan iterasi keberapa rute yang didapat
D	:	Menyimpan iterasi keberapa rute yang didapat

Struktur dan relasi antar tabel digambarkan pada gambar 3.4 dan 3.5.



Gambar 3.4 Struktur database Utama



Gambar 3.5 Struktur Database Pendukung

3.5.3 Penentuan Rute dan Urutan Perjalanan Wisata

Ketika penentuan rute dan urutan perjalanan wisata akan ditentukan, proses yang dilakukan ada 4 tahapan antara lain:

1. Menginisialisasi node setiap titik asal dan titik-titik tujuan. Langkah-langkah yang dilakukan antara lain:
   - Input hotel maupun wisata sebaga asal dan tujuan.
   - Mencari dan menghitung jarak terpendek titik – titik yang ada disekitar titik awal (node-node jalan) dengan titik awal itu sendiri, dan jarak yang terpendek terhadap suatu node jalan yang dihasilkan dijadikan sebagai *node* awal, dengan menggunakan rumus:

$$AB = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \qquad (3.1)$$

   Dimana:

   AB = jarak antara titik asal dengan titik tujuan.

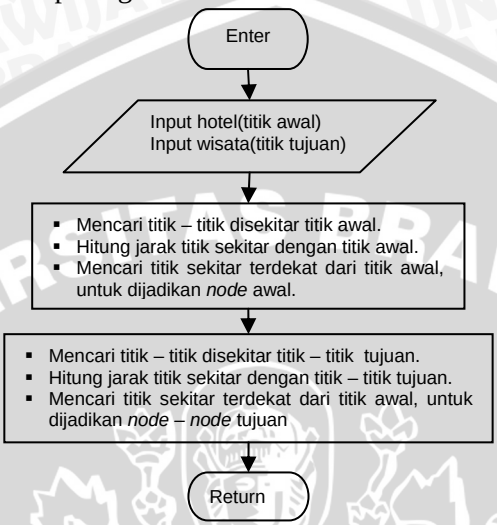
   X_1, X_2 = Koordinat x pada peta.

   Y_1, Y_2 = Koordinat y pada peta.

   (Neutron, 2003)

   - Sama halnya dengan titik-titik tujuan, untuk masing-masing titik tujuan dicari titik-titik terdekat yang berada disekitarnya, yang kemudian *node* jalan dengan jarak terpendek dengan titik tujuan dijadikan sebagai *node- node* tujuan.

Flowchart inisialisasi *node* asal dan *node* tujuan pada suatu peta jalan diperlihatkan pada gambar 3.6.



Gambar 3.6 Inisialisasi *Node*

2. Menentukan graf dari *node-node* jalan yang terdapat pada peta wisata dengan menghubungkan setiap titik – titik yang berada di sekitar titik awal dan titik – titik tujuan yang digunakan untuk mempermudah perhitungan penentuan rute dan urutan perjalanan wisata yang efektif dan efisien serta meminimalkan waktu pencarian.

   Secara umum, langkah-langkah yang dilakukan, antara lain:

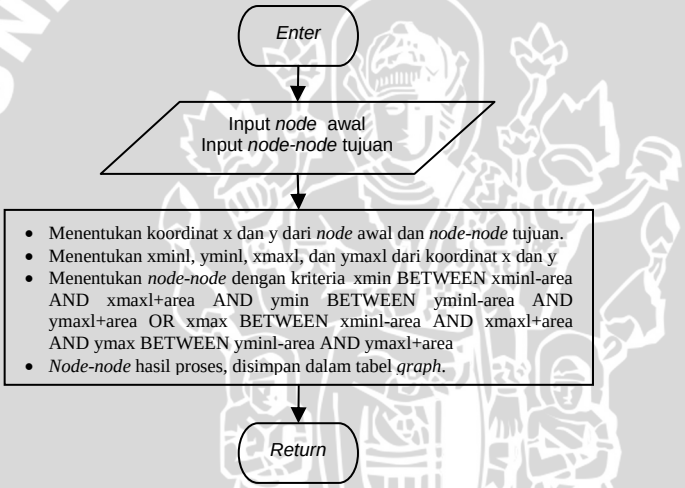
   - *User* memasukkan tempat penginapan (titik awal) dan tempat – tempat wisata tujuan (titik – titik tujuan).
   - Menentukan koordinat x dan y dari *node* awal dan *node-node* tujuan yang diperoleh pada proses *point* 1.
   - Dari semua koordinat titik x yang didapatkan, dicari yang paling min dan max. Begitu juga dengan semua koodinat titik y, yang masing-masing disimpan dalam variabel xminl, yminl, xmaxl, dan ymaxl.
   - Dengan menggunakan perintah *select* pada tabel jalan menggunakan kriteria xmin BETWEEN xminl-area AND xmaxl+area AND ymin BETWEEN yminl-area AND ymaxl+area OR xmax BETWEEN xminl-area AND

xmaxl+area AND ymax BETWEEN yminl-area AND ymaxl+area.

Dimana area adalah wilayah untuk memperlebar daerah *graph* agar *node* yang seharusnya termasuk dalam himpunan *graph* tidak terbuang.

- Dari proses diatas akan menghasilkan *node-node* yang merupakan lintasan-lintasan atau yang menghubungkan antara *node* awal dengan *node-node* tujuan.
- *Node-node* hasil proses, disimpan dalam tabel *graph* (dalam database).

Flowchart pembuatan *graph* dari *node-node* jalan yang terdapat pada suatu peta wisata diperlihatkan pada gambar 3.7.



Gambar 3.7 *Flowchart* pembuatan *graph*.

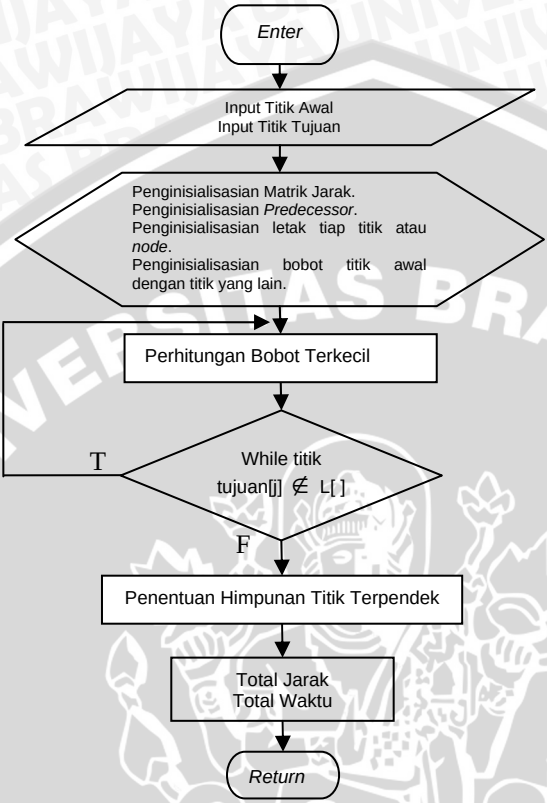
3. Menentukan rute terpendek, total waktu, total jarak yang diperlukan dari graf tersebut dengan menggunakan algoritma *dijkstra*.

Secara umum, langkah – langkah yang dilakukan, antara lain:

- *User* memasukkan inputan berupa *node* awal, *node* tujuan.
- Penginisialisasian Matrik Jarak.
- Penginisialisasian *Predecessor*.
- Penginisialisasian letak tiap titik atau *node*.
- Penginisialisasian bobot titik awal dengan titik yang lain

- Perhitungan bobot terkecil dari titik awal.
- Jika titik yang sudah terpilih bukan titik tujuan dan belum termasuk dalam lintasan terpendek, maka dilakukan perhitungan bobot terkecil dengan menggunakan rumus yaitu D[j] > D[k] + W[k,j].
  Dimana j adalah indek tiap *node*, k adalah titik yang sudah terpilih dalam lintasan terpendek.
- Penentuan Himpunan Titik Terpendek.
- Menghitung total jarak dan total waktu dari setiap lintasan tempat wisata yang didapat dengan melakukan proses pengecekan pada lintasan dan mengambil data waktu dari database.

Untuk lebih jelasnya, langkah penentuan rute diatas dapat dilihat prosesnya dalam *flowchart* pada gambar 3.8.



Gambar 3.8 Diagram Alir Perhitungan Rute menggunakan algoritma *Dijkstra*

4. Menentukan urutan perjalanan wisata yang akan dilalui dengan melakukan perhitungan yang telah dilakukan pada *point* 1, 2 dan 3 diatas.

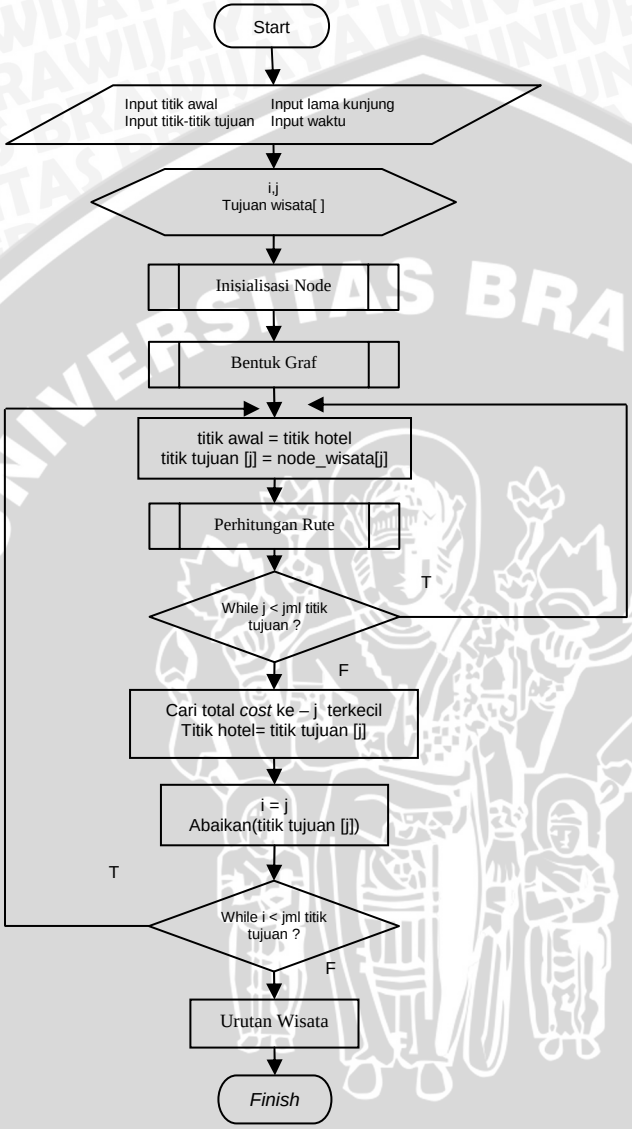
   Pada penentuan urutan ini, tempat wisata yang terakhir telah dikunjungi adalah titik asal keberangkatan selanjutnya.

   Langkah-langkah yang dilakukan antara lain:

   a. *User* memasukkan inputan berupa *node* awal, *node* tujuan, lama kunjung tempat wisata dan waktu keberangkatan.
   b. Penginisialisasian terhadap variabel i,j dan tujuan wisata [ ].
   c. Menginisialisasi node, yang prosesnya terdapat pada gambar 3.6.
   d. Pembentukan graf, prosesnya dapat dilihat pada gambar 3.7.

e. Melakukan perulangan terhadap variabel i, sampai i ≤ jumlah tempat wisata tujuan, untuk melakukan proses pada *point* f dan g.
f. Melakukan perulangan terhadap variabel j, sampai j ≤ jumlah tempat wisata tujuan, untuk melakukan proses titik awal = titik hotel, titik tujuan [j] = node_wisata[j] dan perhitungan rute dengan algoritma *Dijkstra*, yang prosesnya dapat dilihat pada gambar 3.8.
g. Jika j = jumlah tempat wisata tujuan, melakukan proses cari total jarak[i,j] terkecil dan titik awal = titik tujuan [j], apabila i = j, titik tujuan[i] atau titik tujuan[j] tidak dipakai/diabaikan, karena sudah dikunjungi.
h. Jika i ≥ jumlah tempat wisata tujuan, menyusun urutan wisata yang akan dilalui wisatawan.

Langkah-langkah tersebut dapat dilihat dalam *flowchart* pada gambar 3.9.



Gambar 3.9 Diagram Alir Penentuan rute dan Urutan Perjalanan Wisata

3.5.4 Perancangan Antarmuka

Berdasarkan hasil analisis secara keseluruhan terdapat beberapa bagian yang dibutuhkan dalam antarmuka penentuan rute dan urutan perjalanan wisata ini, yaitu :

1. Terdapat 4 bagian dalam 1 halaman yang masing-masing adalah halaman judul, halaman samping, halaman isi dan halaman *footer*.
2. Untuk halaman judul, menampilkan judul dari tugas akhir ini.
3. Halaman samping, digunakan untuk menampilkan menu-menu yang ada, atau menu lain yang bisa ditambahkan, namun dalam tugas akhir ini hanya menggunakan menu yang ada pada gambar 3.10.
4. Menu-menu pada gambar 3.10 akan ditampilkan pada halaman isi.
5. Halaman *footer* hanya utnuk menampilkan *Created By.*

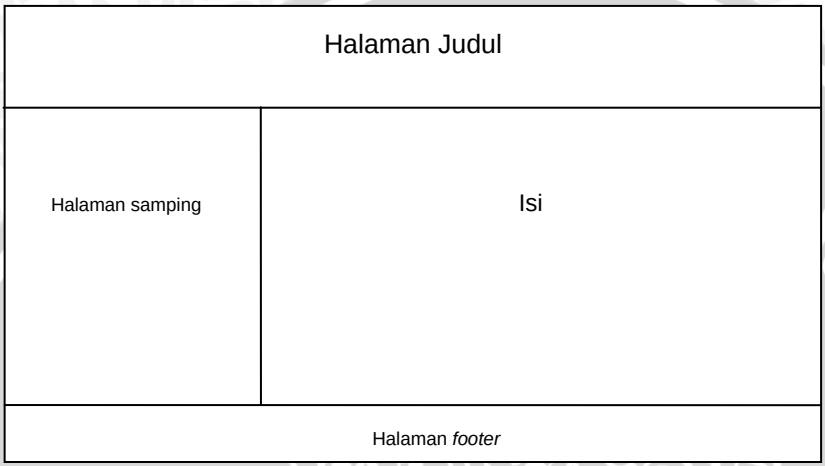
Navigation	Rute Perjalanan	Map	Tambah Titik
- Home - Abaut	Penentuan Rute dan Urutan Perjalanan Berdasarkan: - Jarak Lintasan - Jarak Lintasan tanpa mempertimbangkan kemacetan - Jarak Lintasan dengan mempertimbangkan kemacetan	- Select	- Tambah Titik - Wisata - Hotel

Links
Brawijaya
Ilmu komputer

Gambar 3.10 Rancangan Menu

Sedangkan visualisasi perancangan antarmuka dapat dilihat pada gambar dibawah ini:

1. Halaman Utama

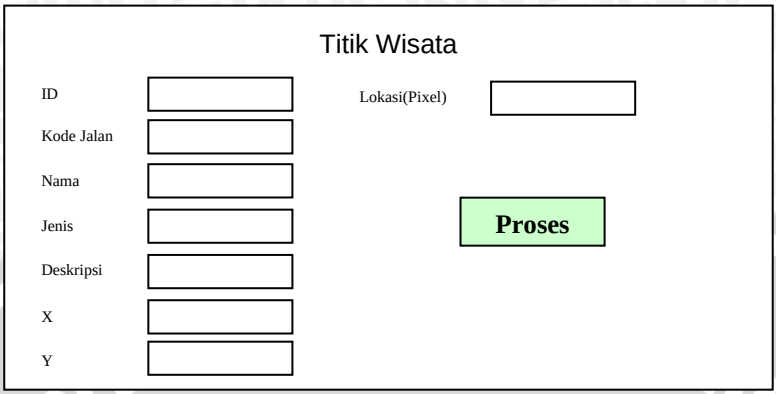


Gambar 3.11 Perancangan Desain halaman utama Web

2. Tambah Titik

Gambar 3.12 Halaman Tambah Titik

Halaman diatas adalah halaman untuk memilih menambah titik tempat wisata atau titik hotel, jika tombol wisata dipilih, maka halaman isi akan menampilkan halaman seperti pada gambar 3.13.

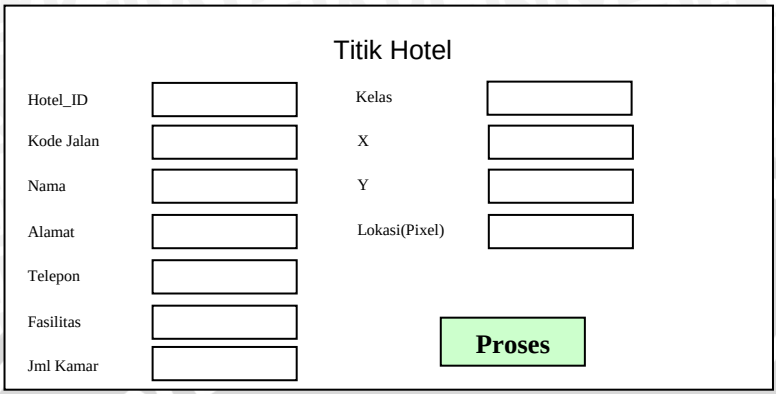


Gambar 3.13 Halaman tambah Titik Tempat Wisata

Keterangan:

ID	:	Input ID tempat wisata yang akan ditambahkan
Kode Jalan	:	Inputan kode jalan dari tempat wisata
Nama	:	Inputan nama tempat wisata
Jenis	:	Inputan jenis tempat wisata
Deskripsi	:	Inputan penjelasan mengenai tempat wisata yang akan ditambahkan
X	:	Inputan koordinat x titik wisata
Y	:	Inputan koordinat y titik wisata
Lokasi(Pixel)	:	Inputan lokasi dalam bentuk *pixel* untuk diproses pada saat melakukan penampilan data titik wisata pada halaman web

Dan jika dipilih tambah titik untuk hotel, maka halaman isi akan menampilkan halaman seperti pada gambar 3.14.



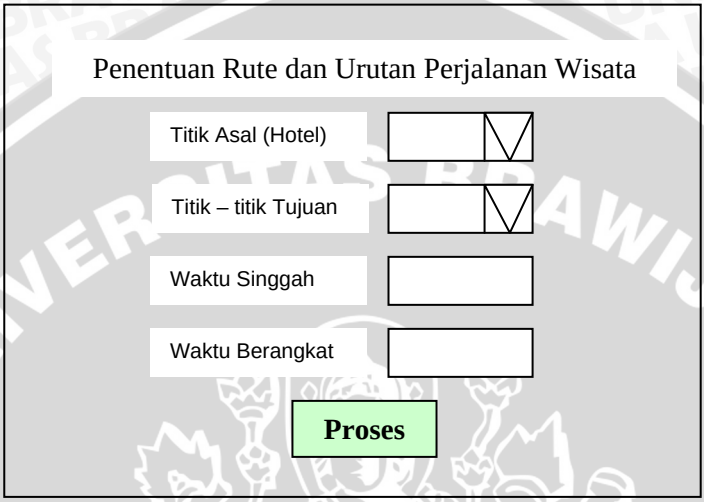
Gambar 3.14 Halaman Tambah Titik Hotel

Keterangan:

Hotel_ID	:	Input ID hotel yang akan ditambahkan
Kode Jalan	:	Inputan kode jalan dari hotel
Nama	:	Inputan nama hotel
Alamat	:	Inputan alamat hotel
Telepon	:	Inputan nomor telepon hotel yang akan ditambahkan
Fasilitas	:	Inputan fasilitas yang dimiliki hotel
Jml Kamar	:	Inputan jumlah kamar yang dimiliki hotel
Kelas	:	Inputan kelas yang disandang oleh hotel
X	:	Inputan koordinat x titik hotel
Y	:	Inputan koordinat y titik hotel
Lokasi(Pixel)	:	Inputan lokasi dalam bentuk *pixel* untuk diproses pada saat melakukan penampilan data titik wisata pada halaman web

3. Halaman *User*.

Secara umum halaman *user* yang digunakan adalah seperti pada gambar 3.15.



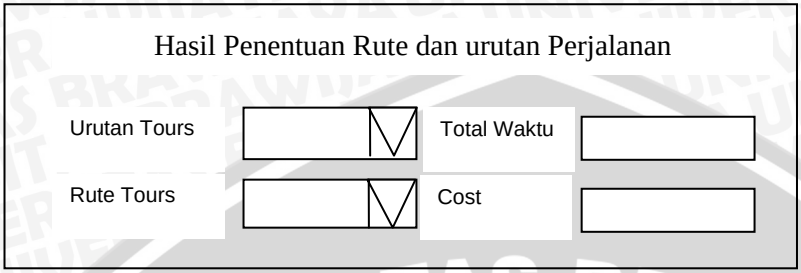
Gambar 3.15 Halaman *Input* Penentuan Rute dan Urutan Perjalanan Wisata

Keterangan:

Titik asal (Hotel)	:	Inputan berupa nama dari hotel.
Titik – titik tujuan	:	Inputan berupa nama dari tempat-tempat wisata.
Waktu singgah	:	Berapa jam wisatawan berada pada suatu tempat wisata
Waktu berangkat	:	Waktu dimulainya perjalanan, yaitu dari hotel.

Pada penentuan rute dan urutan perjalanan wisata berdasarkan Jarak lintasan saja, waktu berangkatnya tidak digunakan.

Jika tombol proses diatas di *click*, akan ditampilkan pada halaman isi, *output*-nya, yang dapat dilihat pada gambar 3.16.



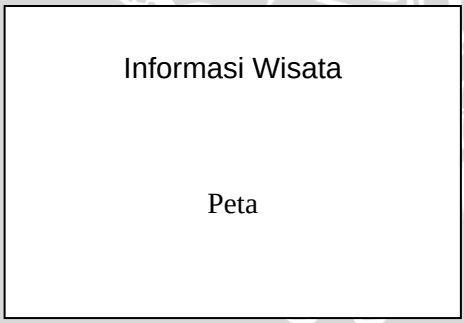
Gambar 3.16 Halaman *Output* Penentuan Rute dan Urutan Perjalanan Wisata

Keterangan:

Urutan Tours	:	Urutan tempat wisata mana dulu yang akan disinggahi.
Rute tours	:	Jalan yang dilewati untuk setiap pergantian kunjungan tempat wisata.
Total waktu	:	Total waktu yang dibutuhkan untuk semua perjalanan yang dilalui.
cost	:	Total jarak lintasan yang dilalui.

4. Halaman Peta.

Halaman Peta akan ditampilkan pada halaman isi jika *User* memilih menu *Map* dan *option select.*



Gambar 3.17 Halaman Peta dan Informasi Titik pada Peta

3.5.5 Perancangan Uji Coba

3.5.5.1 Bahan Pengujian

Bahan yang akan digunakan pada proses pengujian ini, yaitu beberapa tempat wisata, total waktu, total jarak dan waktu berangkat.

3.5.5.2 Tujuan Pengujian

Beberapa hal yang menjadi tujuan dari pelaksanaan pengujian terhadap sistem penentuan rute dan urutan perjalanan wisata ini, yaitu :

- Keluaran dari perangkat lunak mampu merepresentasikan hasil yang telah dianalisa (analisa hasil) dari tiga *item* penentuan rute yang telah ditentukan.
- Dengan analisa yang dilakukan, apakah pencarian rute terpendek dapat menghasilkan waktu tersingkat dengan adanya variabel kemacetan.

3.5.5.3 Skenario dan Kriteria Pengujian

Pengujian yang dilaksanakan pada Tugas Akhir yaitu uji terhadap analisa hasil.

3.5.5.3.1 Pengujian Analisa Hasil

Pada analisa hasil akan dilakukan pengujian dengan mengganti nilai waktu keberangkatan. Nilai waktu keberangkatan yang digunakan antara lain 04.00, 06.00, 08.00, 09.00, 12.00, 13.00, 14.00, 16.00, 18.00, 21.00. Jam ini diambil selain untuk mempermudah perhitungan juga agar dapat dibedakan antara total waktu dan total jarak yang dihasilkan untuk tiap waktu keberangkatan.

Uji coba dilakukan masing-masing 10x percobaan dengan jumlah tempat wisata yang ditentukan yaitu 1, 3, 6, 9 dan 13, hal ini juga dilakukan untuk mengetahui keterkaitan antara variabel waktu keberangkatan dengan total jarak ataupun total waktu dalam penentuan dan urutan rute perjalanan wisata.

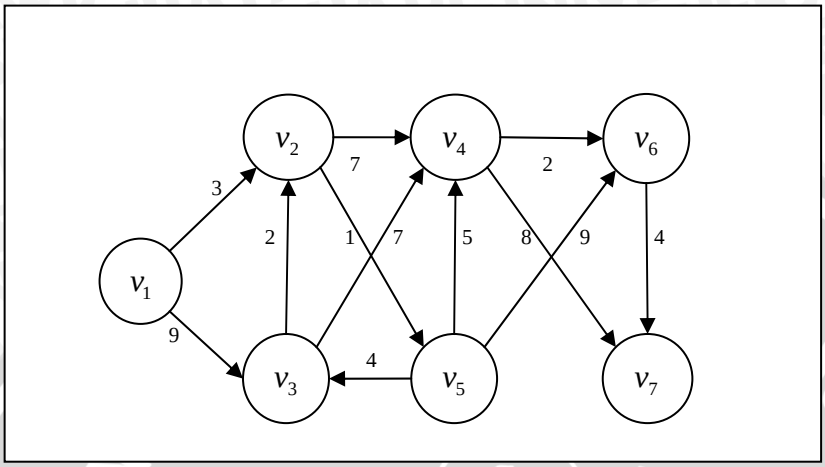
3.6 Contoh Perhitungan

Berikut ini dicontohkan bagaimana cara kerja sistem dalam membentuk suatu penentuan rute dan urutan perjalanan wisata:

➔ Misalkan ada 6 tempat wisata, dan 1 tempat penginapan (hotel), pencarian rutenya adalah sebagai berikut:

Titik Asal : v_1

Titik Tujuan : v_7



Gambar 3.18 Posisi tempat wisata dan hotel

Contoh diatas adalah contoh penerapan dari Algoritma *Dijkstra*, dalam mendukung pencarian rute dan urutan perjalanan wisata.

1. Matrik hubung W untuk menyatakan graf pada gambar 3.18 adalah sebagai berikut:

$$W = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} \infty & 3 & 9 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 7 & 1 & \infty & \infty \\ \infty & 2 & \infty & 7 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 2 & 8 \\ \infty & \infty & 4 & 5 & \infty & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 6 \\ \infty & \infty & \infty & \infty & \infty\infty & \infty & \infty \end{bmatrix} \end{array}$$

2. Mula – mula $L = \{\ \}$ dan $V = \{v_2, v_3, ..., v_7\}$

   D(2) = W(1,2) = 3 D(5) = W(1,5) = ∞
   D(3) = W(1,3) = 9 D(6) = W(1,6) = ∞
   D(4) = W(1,4) = ∞ D(7) = W(1,7) = ∞
   V – L = $\{v_2, v_3, ..., v_7\}$ - $\{\ \}$ = $\{v_2, v_3, ..., v_7\}$

   $v_n = v_7 \notin L$, sehingga langkah 3(a) dan 3(b) dilakukan:

3. (a). D(k) terkecil adalah D(2), sehingga $v_k = v_2$

$L = L \cup \{v_k\} = \{\ \} \cup \{v_2\} = \{v_2\}$

(b). $V - L = \{v_2, v_3, ..., v_7\} - \{v_2\} = \{v_3, v_4, v_5, v_6, v_7\}$

k = 2 (dai langkah 3(a)).

Diselidiki tiap titik dalam V – L.

Untuk j = 3:

D(j) = D(3) = 9

D(k) + W(k,j) = D(2) + W(2,3) = 3 + ∞ = ∞

Karena D(3) > $\neq$ D(2) + W(2,3) maka D(3) tetap = 9

Untuk j = 4

D(j) = D(4) = ∞

D(k) + W(k,j) = D(2) + W(2,4) = 3 + 7 = 10

Karena D(4) > D(2) + W(2,4) maka harga D(4) dirubah menjadi 10.

Ini berarti bahwa untuk mencapai titik v_4 dari v_1, jalur melalui v_2, yaitu : v_1, v_2, v_4 (D(2) + W(2,4)) mempunyai bobot lebih kecil dibandingkan jalur langsung v_1, v_4 (D(4)).

Untuk j = 5

D(j) = D(5) = ∞

D(k) + W(k,j) = D(2) + W(2,5) = 3 + 1 = 4

Karena D(5) > D(2) + W(2,5) maka harga D(5) dirubah menjadi 4.

Untuk j = 6

D(j) = D(6) = ∞

D(k) + W(k,j) = D(2) + W(2,6) = 3 + ∞ = ∞

Karena D(6) > $\neq$ D(2) + W(2,6) maka harga D(6) tetap seperti sebelumnya yaitu ∞.

Untuk j = 7

D(j) = D(7) = ∞

D(k) + W(k,j) = D(2) + W(2,7) = 3 + ∞ = ∞

Karena D(6) > $\neq$ D(2) + W(2,6) maka harga D(7) tetap seperti sebelumnya yaitu ∞.

Langkah selanjutnya kembali ke langkah 3(a). Karena $v_7 \notin L$.

$V - L = \{v_3, v_4, v_5, v_6, v_7\} \neq \emptyset$

Di antara D(k) (k = 3, 4, 5, ..., 7) hasil iterasi langkah 3(b), D(k) yang terkecil adalah D(5), sehingga $v_k = v_5$

Maka sekarang, $L = L \cup \{v_5\} = \{v_2\} \cup \{v_5\} = \{v_2, v_5\}$

3(b) : V – L = $\{v_2, v_3, v_4, v_5, v_6, v_7\}$ - $\{v_2, v_5\}$ = $\{v_3, v_4, v_6, v_7\}$

Langkah 3(b) untuk mengecek setiap titik dalam V – L diulangi lagi.

Langkah 3(a) dan 3(b) diulang - ulang terus hingga $v_7 \notin L$. Hasil iterasi selengkapnya dapat dilihat pada tabel 3.10.

Karena $v_n = 7 \in L$, maka iterasi dihentikan. *Path* terpendek dari v_1 ke v_7 adalah 15 dengan jalur yang dibaca mundur sebagai berikut:

Pada iterasi dengan indeks k = 6, diperoleh penurunan jarak pada kolom D(7) dari 17 ke 15.

Ini berarti titik yang terpilih pada baris tersebut (v_6) adalah jalur *path* (jalur $v_6 \rightarrow v_7$).

Berikutnya pada kolom D(6) diperoleh penurunan jarak dari 13 ke 11. ini berarti titik pada indeks k = 4 (v_4) adalah jalur *path* (jalur adalah $v_4 \rightarrow v_6 \rightarrow v_7$).

Pada kolom D(4) tidak terjadi penurunan jarak (dari 9 tetap 9). Ini berarti bahwa titik pada indeks k = 3 (v_3) bukan jalur *path*. Naik satu baris diatasnya, terjadi penurunan jarak dari 10 ke 9 yaitu pada baris yang sesuai dengan indeks k = 5. Jadi v_5 adalah jalur *path* (jalur adalah $v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$).

Pada kolom D(5) terjadi penurunan jarak dari ∞ ke 4. Baris tersebut sesuai dengan indeks k = 2. Jadi v_2 adalah jalur *path* (jalur $v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$).

Jadi *path* terpendek adalah $v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$ dengan total panjang (*cost*) adalah 15.

Hasil akhir dari algoritma *Dijkstra* adalah *path* terpendek dari titik v_1 ke semua titik lain.

BAB IV
IMPLEMENTASI DAN UJI COBA SISTEM

Implementasi merupakan proses transformasi representasi rancangan ke bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini akan dibahas hal-hal yang berkaitan dengan implementasi sistem penentuan rute terpendek dengan algoritma *Dijkstra*.

4.1 Implementasi

Implementasi yang akan dipaparkan disini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan penentuan rute dan urutan perjalanan ini adalah sebagai berikut :

1. Prosesor Intel Pentium IV 1.80 Ghz
2. RAM 256 MB
3. *Harddisk* dengan kapasitas 40 GB
4. Monitor 15”
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan penentuan rute dan urutan perjalanan ini adalah :

1. Sistem Operasi Windows XP
2. *Macromedia Dreamweaver* 4
3. *Text editor* Notepad 2
4. *XAMPP* 2.1, yang didalamnya terdapat:
   - PHP *Version* 5.0.5
   - *Apache* 2.0.55
   - *mySQL* 5.0.15
5. *ArcView GIS 3.3*
6. *Handy ImageMapper*

4.1.3 Implementasi *Database*

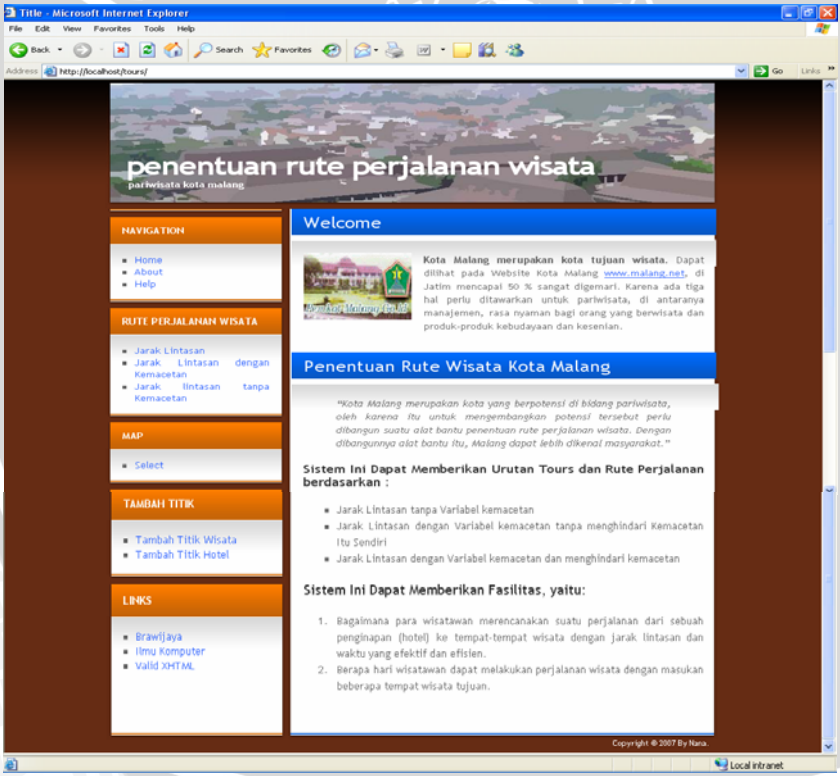
Pada bab sebelumnya telah dijelaskan rancangan tabel dan relasi basis data yang terdiri dari 7 buah tabel. Jenis kolom dan fungsinya sudah dijelaskan lebih detail pada bab III.

Tabel-tabel tersebut beserta relasinya diimplementasikan dengan menggunakan *mySQL* 5.0.15.

Sebagian data dalam tabel dihasilkan dari hasil *export ToWKT Extension* dimana contoh sebagian hasil dapat dilihat pada lampiran 1, yang langkah-langkahnya dapat dilihat pada subbab 2.1.2.

4.1.4 Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada subbab 3.5.4 dihasilkan antarmuka seperti pada gambar 4.1.

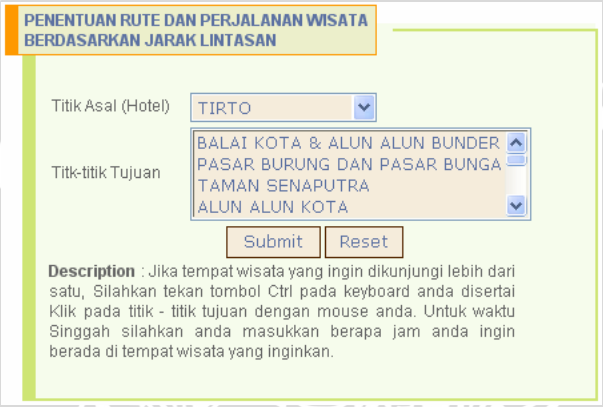


Gambar 4.1 Hasil Perancangan Antarmuka Tampilan Halaman Utama

4.1.5 Input Data

4.1.5.1 *Form* Jarak Lintasan.

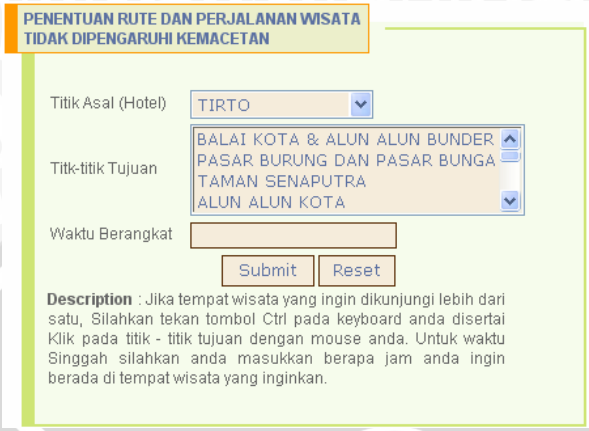
Penentuan urutan wisata dan rute perjalanan berdasarkan jarak lintasan, memerlukan inputan dari *user* berupa titik asal (hotel), titik-titik tujuan (tempat wisata), dengan antarmuka seperti pada gambar 4.2.



Gambar 4.2 *Form* Input Berdasarkan Jarak Lintasan.

4.1.5.2 *Form* Jarak Lintasan baik yang dipengaruhi kemacetan maupun tidak.

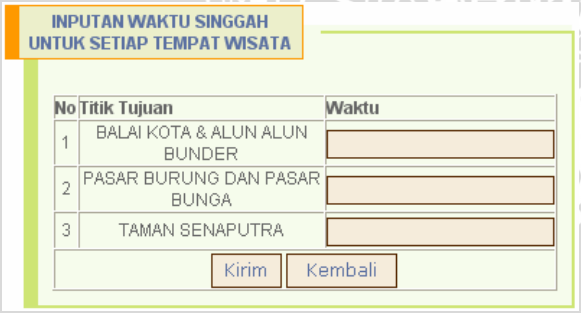
Penentuan urutan wisata dan rute perjalanan berdasarkan jarak lintasan baik dengan mempertimbangkan kemacetan maupun tanpa mempertimbangkan kemacetan, memerlukan inputan dari *user* berupa titik asal (hotel), titik-titik tujuan (tempat wisata) serta waktu berangkat, dengan antarmuka seperti pada gambar 4.3.



Gambar 4.3 *Form* Input Berdasarkan Jarak Lintasan baik dengan mempertimbangkan kemacetan maupun tanpa mempertimbangkan.

4.1.5.3 *Form* Waktu Singgah.

Form ini akan ditampilkan apabila *user* men-*submit* tombol *submit* pada saat melakukan proses penentuan rute dan urutan wisata, maka sesuai dengan inputan tempat wisata yang *user* inginkan, user akan diminta untuk memasukkan masing-masing waktu singgah dari tempat wisata yang ingin dikunjungi. Untuk inputan waktu singgah masing-masing tempat wisata dapat dilihat pada gambar 4.4.

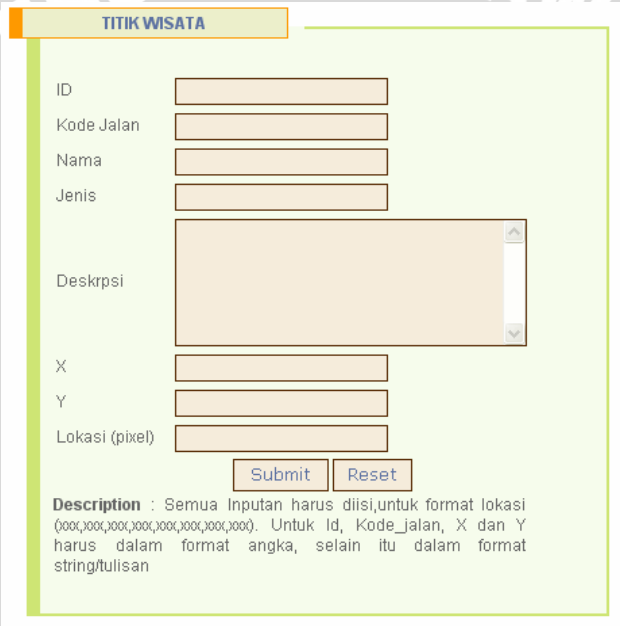


Gambar 4.4 *Form* Waktu Singgah

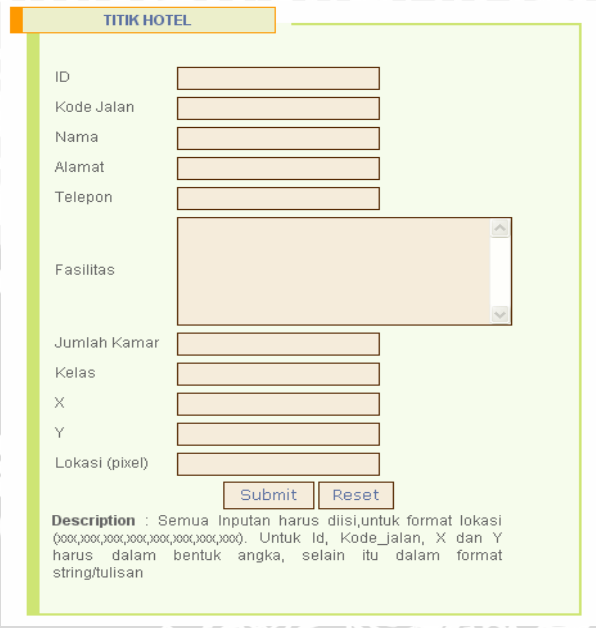
4.1.5.4 *Form* Tambah Titik.

Diperlukan pula inputan untuk menambah titik baik titik hotel ataupun titik wisata, yang merupakan kebutuhan tambahan dalam sistem ini, yang digunakan jika sewaktu-waktu terdapat tambahan hotel dan wisata pada kota tertentu, yang dapat dilihat pada gambar 4.5 dan 4.6.

Namun untuk inputan koordinat x dan y serta lokasi masih memerlukan *software* lain sebagai alat bantu, yaitu *ArcView GIS 3.3* dan *Handy ImageMapper*.



Gambar 4.5 Tambah Titik Wisata

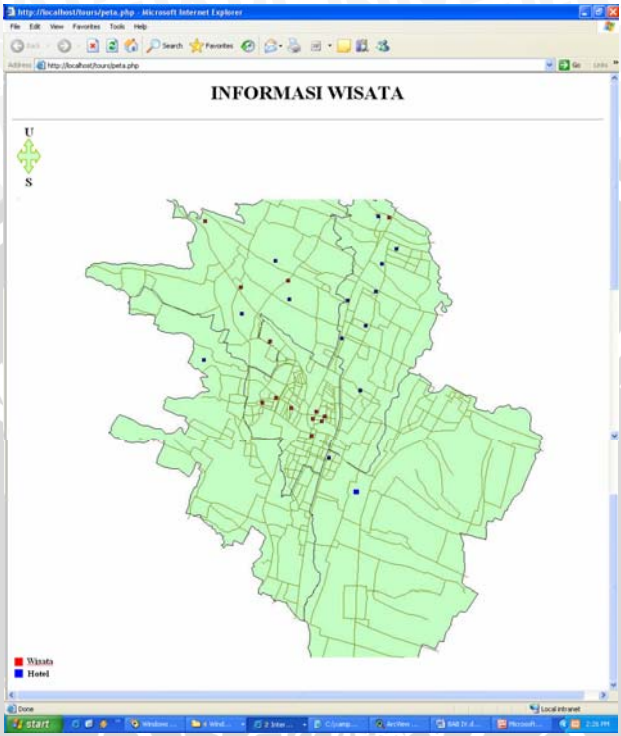


Gambar 4.6 Tambah Titik Hotel

Untuk penjelasan tiap *item* inputan telah dijelaskan pada subbab 3.5.2. Hasil tambah titik tersebut ditambahkan pada peta yang dapat dilihat pada halaman peta.

4.1.5.5 Halaman Peta.

Pada halaman peta *user* dapat memberikan inputan dengan meng – *click* titik yang berwarna merah atau biru pada peta, dapat dilihat pada gambar 4.7.



Gambar 4.7 Halaman Peta

4.1.6 Deskripsi Program

Implementasi aplikasi untuk pencarian rute terpendek dari hasil perancangan yang telah dilakukan, baik berdasarkan jarak lintasan, jarak lintasan tanpa mempertimbangkan kemacetan dan jarak lintasan dengan mempertimbangkan kemacetan. Untuk ketiga *item* tersebut dibentuk dari beberapa *script* PHP, antara lain:

4.1.6.1 *Script* connect.php

Script ini berfungsi untuk melakukan koneksi ke *database server* mySQL. Diimplementasikan dalam *script* connect.php:

```
<?
        $db = mysql_connect("localhost", "root","");
        mysql_select_db("wisata",$db);
?>
```

4.1.6.2 *Script* titikasal.php

Script ini berfungsi melakukan proses *query* Kd_jln, xmin, ymin dalam tabel hotel berdasarkan nama hotel yang dikirimkan oleh aplikasi penentuan rute perjalanan wisata. Nama hotel yang dikirimkan oleh aplikasi tersebut merupakan titik awal dimulainya perjalanan. Hasil *query* Kd_jln digunakan kembali untuk melakukan *query* xmin, ymin, xmax, ymax, fnode_, tnode_ dari tabel jalan. *Script* proses *query* dalam titikasal.php:

```
<?
    foreach ($asal as $value) {
       $titik_asal=$value;}
mysql_select_db("wisata") or
die("tidak dapat memilih database");
$query1="select Kd_jln, xmin, ymin from hotel where
nama='$titik_asal'";
$result=mysql_query($query1);
        while ($row1=mysql_fetch_array($result)){
               $kdjlnasal=$row1[Kd_jln];
               $xasal=$row1[xmin];
               $yasal=$row1[ymin];}
$i=0;
$query1="select namajln, xmin, xmax, ymin, ymax, fnode_,
tnode_ from jalan where Kd_jln='$kdjlnasal'";
$result1=mysql_query($query1);
...
```

Hasil *query* tersebut kemudian digunakan untuk menghasilkan node dari hotel yang proses perhitungannya dapat dilihat dalam *script* titikasal.php:

```
…
//menghitung jarak titik min node jalan dengan titik hotel
        $angka1=$xasal-$xminAsal[$i];
        $angka2=$yasal - $yminAsal[$i];
        $angka3=pow($angka1,2);
        $angka4=pow($angka2,2);
        $jmlmin=$angka3 + $angka4;
        $distmin[$i]=sqrt($jmlmin);
//menghitung jarak titik max node jalan dengan titik hotel
        $angka5=$xasal - $xmaxAsal[$i];
        $angka6=$yasal - $ymaxAsal[$i];
        $angka7=pow($angka5,2);
        $angka8=pow($angka6,2);
        $jmlmax=$angka7 + $angka8;
        $distmax[$i]=sqrt($jmlmax);
//Pengurutan distmin untuk mencari titik asal dengan jarak
terpendek dari titik min
         $jml=mysql_num_rows($result1);
         $min=$distmin[0];
         $idx=0;
         for($asli=0;$asli<$jml;$asli++){
            if ($distmin[$asli]<$min){
                $min=$distmin[$asli];
                 $idx=$asli;}
            else $min = $min;
              $idx=$idx;}
            $fnodeawal=$nodeA[$idx];
            $xmin=$xminAsal[$idx];
            $ymin=$yminAsal[$idx];
//Pengurutan distmax untuk mencari titik asal dengan jarak
terpendek dari titik max
        $max=$distmax[0];
        $index=0;
        for($asli1=0;$asli1<$jml;$asli1++){
           if ($distmax[$asli1]<$max){
               $max=$distmax[$asli1];
                $index=$asli1;}
           else $max = $max;
                $index=$index;}
           $tnodeawal=$nodeB[$index];
           $xmax=$xmaxAsal[$index];
           $ymax=$ymaxAsal[$index];
...
```

Perhitungan diatas untuk mendapatkan *node* dari titik asal dan titik tujuan, yaitu dengan mencari titik terdekat antara titik hotel atau titik wisata dengan *node-node* jalan. Satu jalan tediri dari 2 *node*, yaitu fnode_ dan tnode_ yang masing-masing mewakili koordinat

min dan max, yang kemudian antara keduanya dicari jarak yang paling dekat/terkecil, jika sudah didapatkan maka hasilnya adalah perwakilan *node* dari titik hotel maupun titik tempat wisata. Variabel-variabel yang terdapat pada *script* tersebut adalah variabel yang mewakili perhitungan pada rumus jarak antara dua titik:

$$AB = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

4.1.6.3 *Script* titiktujuan.php

Struktur data untuk *script* titiktujuan.php dalam penentuan node wisata sama dengan struktur data pada *script* titikasal.php.

4.1.6.4 *Script* initgraph.php

Script ini berfungsi untuk membuat graf yang merepresentasikan lintasan-lintasan jalan raya dalam suatu kota serta membuat matrik jarak, matrik macet, matrik waktu dan matrik *template* dari graf tersebut. Matrik jarak, matrik macet, matrik waktu dan matrik *template* ini berisi data input untuk pencarian lintasan terpendek dalam suatu graf yang disimpan dalam suatu *array* dua dimensi. Untuk matrik macet ada 3 matrik, yaitu matrik macet_pagi, matrik macet_siang dan matrik macet_sore.

Matrik jarak, matrik macet, matrik waktu dan matrik *template* tersebut dapat menunjukkan bahwa titik-titik yang bertetangga (mempunyai bobot berhingga) adalah mempunyai *edge* yang terhubung, sedangkan titik-titik yang tidak bertetangga (mempunyai bobot tak berhingga) mempunyai *edge* yang tidak terhubung.

Script yang memperlihatkan proses pembuatan graf:

```
…
$area=600;
mysql_select_db("wisata") or
die("tidak dapat memilih database");
$query="SELECT namajln, fnode_, tnode_, length, pagi, siang,
  sore FROM jalan WHERE xmin BETWEEN $xminl-$area AND
  $xmaxl+$area AND ymin BETWEEN $yminl-$area AND
  $ymaxl+$area OR xmax BETWEEN $xminl-$area AND $xmaxl+$area
  AND ymax BETWEEN $yminl-$area AND $ymaxl+$area";
$i=0;
$result=mysql_query($query);
       while ($row=mysql_fetch_array($result)){
              $gnamajln[$i]=$row[namajln];
              $gfnode[$i]=$row[fnode_];
              $gtnode[$i]=$row[tnode_];
              $glength[$i]=$row[length];
              $gpagi[$i]=$row[pagi];
              $gsiang[$i]=$row[siang];
              $gsore[$i]=$row[sore];
              $waktu[$i]=round($glength[$i]/16.67);
$sql="INSERT INTO graph
  (Fnode_,Tnode_,Length,namajln,waktu_lintasan,pagi,siang,so
  re) VALUES ('$gfnode[$i]', '$gtnode[$i]', '$glength[$i])',
  '$gnamajln[$i]', '$waktu[$i]', '$gpagi[$i]',
  '$gsiang[$i]', '$gsore[$i]')";
$tambah=mysql_query($sql);
$i++;}
…
```

Area adalah konstanta yang dibutuhkan untuk memperluas area graf, agar jalan yang seharusnya termasuk dalam himpunan graf yang merupakan kumpulan jalan disekitar *node* awal dan *node* tujuan tidak terpotong, apabila terpotong akan menyebabkan *node* yang terhubung menjadi tidak terhubung. Graf tersebut diambil dari kumpulan *node-node* jalan pada peta kota malang.

Script yang memperlihatkan proses pembuatan matrik jarak, matrik macet, matrik waktu dan matrik *template* yang merepresentasikan suatu graf lintasan-lintasan jalan dalam suatu kota:

```
...
$queryV2="SELECT DISTINCT node FROM temp ORDER BY node ASC";
$resultV2=mysql_query($queryV2);
        $l=0;
        while($rowV2=mysql_fetch_array($resultV2)){
                $V[$l]=$rowV2[node];
        $l++;}
//menentukan matrik jarak dan macet serta waktu jarak
lintasan
$jmlnode=$l;
        for($a=0;$a<$jmlnode;$a++){
                for($b=0;$b<$jmlnode;$b++){
                    if(($a==$b)|($b==$a)){
                        $edge[$V[$a]][$V[$b]]=0;
                        $waktu_lintasan[$V[$a]][$V[$b]]=0;
                        $template[$V[$a]][$V[$b]]=0;
                        $macet_pagi[$V[$a]][$V[$b]]=0;
                        $macet_siang[$V[$a]][$V[$b]]=0;
                        $macet_sore[$V[$a]][$V[$b]]=0;}
                else{
                $edge[$V[$a]][$V[$b]]=$infinity;
                $waktu_lintasan[$V[$a]][$V[$b]]=$infinity;
                $template[$V[$a]][$V[$b]]=infinity;
                $macet_pagi[$V[$a]][$V[$b]]=$infinity;
                $macet_siang[$V[$a]][$V[$b]]=$infinity;
                $macet_sore[$V[$a]][$V[$b]]=$infinity;}}
        }
...
```

Inisialisasi bobot yang dipakai, jika *node* awal dan *node* akhir sama, maka bobotnya adalah 0, dan selain itu adalah tidak berhingga atau *infinity*.

Matrik *template* merupakan penggandaan dari matrik jarak, matrik ini digunakan pada saat, mencari jarak sebenarnya dari jalan yang dilalui kemacetan, karena jalan yang mengalami kemacetan pada waktu-waktu tertentu bobot jaraknya diperbesar.

Script yang memperlihatkan proses pengambilan data dalam pembuatan matrik jarak, macet, waktu, nama jalan dan *template*:

```
...
//mengambil bobot dalam database
$querylength="SELECT fnode_, tnode_, length, namajln,
  waktu_lintasan, pagi, siang, sore FROM graph";
$lengthresult=mysql_query($querylength);
while($rowlength=mysql_fetch_array($lengthresult)){
       $edge[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[length];
       $namajalan[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[namajln];
       $waktu_lintasan[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[waktu_lintasan];
       $template[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[length];
       $macet_pagi[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[pagi];
       $macet_siang[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[siang];
       $macet_sore[$rowlength[fnode_]][$rowlength[tnode_]]=
       $rowlength[sore];
       $edge[$rowlength[tnode_]][$rowlength[fnode_]]=
       $edge[$rowlength[fnode_]][$rowlength[tnode_]];
       $namajalan[$rowlength[tnode_]][$rowlength[fnode_]]=
       $namajalan[$rowlength[fnode_]][$rowlength[tnode_]];
       $waktu_lintasan[$rowlength[tnode_]][$rowlength[fnode_]]=
       $waktu_lintasan[$rowlength[fnode_]][$rowlength[tnode_]];
       $template[$rowlength[tnode_]][$rowlength[fnode_]]=
       $template[$rowlength[fnode_]][$rowlength[tnode_]];
       $macet_pagi[$rowlength[tnode_]][$rowlength[fnode_]]=
       $macet_pagi[$rowlength[fnode_]][$rowlength[tnode_]];
       $macet_siang[$rowlength[tnode_]][$rowlength[fnode_]]=
       $macet_siang[$rowlength[fnode_]][$rowlength[tnode_]];
       $macet_sore[$rowlength[tnode_]][$rowlength[fnode_]]=
       $macet_sore[$rowlength[fnode_]][$rowlength[tnode_]];
}
…
```

Bobot berubah sesuai dengan nilai yang disimpan dalam database untuk setiap lintasan jalan, dan bobot jalan awal ke akhir adalah sama dengan bobot jalan akhir ke awal, karena jalan yang dilalui adalah 2 arah.

Untuk matrik waktu, data waktu lintasan diperoleh dari rumus yang terdapat pada subbab 2.9, yaitu:

$$waktutotal = \frac{jaraktotal}{kelajuanrata - rata}$$

Dengan jaraktotal yang diperoleh dari data hasil convert *ToWKT Extension* dan kelajuan rata-rata sebesar 60 km/jam, diambil 60 km/jam, karena kelajuan rata-rata tersebut adalah kelajuan yang umum digunakan orang dalam berkendaraan (Dinas Perhubungan). Dalam program menggunakan satuan m/s, sehingga kelajuan rata-ratanya menjadi 16.67 m/s.

4.1.6.5 *Script* berangkat.php

Script ini berfungsi untuk menentukan skala dari jam-jam keberangkatan yang diinputkan oleh *user*, yang nantinya akan digunakan pada saat pengecekan waktu kemacetan pada setiap lintasan jalan raya. Skala akan berubah setiap kali melintasi jalan yang pada jam tertentu terdapat kemacetan atau tidak terdapat kemacetan. Jam-jam terjadi kemacetan disimpan dalam array, yaitu array pagi, siang dan sore, dimana masing-masing *interval* jam terjadinya kemacetan yaitu, 06.30-09.30, 11.00-14.30, 15.05-17.30 (Dinas Perhubungan). *Script* yang memperlihatkan penentuan skala untuk jam berangkat yang diinputkan *user*:

```
…
if(in_array("$waktuberangkat",$pagi)){
        $skala=1;}
else if(in_array("$waktuberangkat",$siang)){
        $skala=2;}
else if(in_array("$waktuberangkat",$sore)){
        $skala=3;}
else{
        $skala=0;}
$pisah=explode(".", $waktuberangkat,2);
        while (list($index,$data)=each($pisah)){
                $datapisah[$index]=$data;}
settype($datapisah[0],"integer");
settype($datapisah[1],"integer");
$jam=$datapisah[0];
$menit=$datapisah[1];
?>
```

Skala 1, 2 dan 3 menunjukkan adanya kemacetan untuk pagi, siang dan sore, sedangkan 0 menunjukkan tidak ada kemacetan. Skala diperoleh dari jam keberangkatan, dengan melakukan pengecekan apakah jam keberangkatan termasuk dalam waktu-waktu terjadinya kemacetan.

4.1.6.6 *Script* shortpath.php

Script shortpath.php digunakan untuk menentukan lintasan terpendek dari salah satu elemen himpunan titik awal ke salah satu elemen himpunan titik tujuan. *Script* ini menggunakan algoritma *dijkstra* untuk menentukan lintasan terpendek dari titik awal ke semua titik yang ada dalam graf, sampai titik yang dicari temasuk didalamnya.

Ketika *script* ini dipanggil, maka terlebih dahulu dilakukan proses inisialisasi matrik jarak, *predecessor* dan letak tiap titik. *Script* yang memperlihatkan implementasi proses inisialisasi matrik jarak, *predecessor* dan letak tiap titik dengan algoritma *dijkstra*.

```
vx[0]=$nodehtl;
$dist[$vx[0]]=0;
$path[$vx[0]]=-1;
$ins[$vx[0]]=1;
if(($vx[0]!=$node_wisata[$d]) and
($node_terpilih[$node_wisata[$d]]==0)){
        $indek=0;
        for($k0=0;$k0<$jmlnode;$k0++){
                if($V[$k0]!=$vx[0]){
                        $indek=$indek+1;
                        $vx[$indek]=$V[$k0];
                        $ins[$vx[$indek]]=0;
        $dist[$vx[$indek]]=$edge[$vx[0]][$vx[$indek]];
                        if($dist[$vx[$indek]]==$infinity){
                                $path[$vx[$indek]]=-1;}
                        else{
                                $path[$vx[$indek]]=$vx[0];}
                }
        }
```

Selanjutnya dilakukan pencarian lintasan terpendek dari titik awal ke titik dicari menggunakan algoritma *dijkstra*. Setelah titik dicari sudah termasuk dalam himpunan lintasan terpendek, maka algoritma *dijkstra* berhenti.

Script yang memperlihatkan implementasi algoritma *dijkstra*:

```
$notshort=false;
for($k1=1;$k1<$jmlnode;$k1++){
        $mindist=$infinity;
        for($k2=1;$k2<$jmlnode;$k2++){
                if($ins[$vx[$k2]]==0){
                        if($dist[$vx[$k2]]<$mindist){
                                $mindist=$dist[$vx[$k2]];
                                $k3=$vx[$k2];
        }}}
//k3 sudah termasuk dalam titik yang sudah terpilih
$ins[$k3]=1;
        $minbobot=$infinity;
        if($dist[$k3]>=$minbobot){
                $notshort=true;
                break;}
        $instujuan=false;
        if($k3==$node_wisata[$d]){
                $instujuan=true;}
        if($instujuan==true){
                break;}
//dicari yang belum termasuk L atau himpunan dari v
for($k4=0;$k4<$jmlnode;$k4++){
   if($ins[$vx[$k4]]==0){
    if(($dist[$k3]+$edge[$k3][$vx[$k4]])<$dist[$vx[$k4]]) {
        $dist[$vx[$k4]]=$dist[$k3]+$edge[$k3][$vx[$k4]];
        $path[$vx[$k4]]=$k3;
    }}}}
$vakhir=$k3;
```

Himpunan titik lintasan terpendek diperoleh dengan mencari *predecessor* tiap titik dalam lintasan terpendek, dimulai dari titik dicari sampai titik awal. Total jarak lintasan dan total waktu diperoleh dengan menjumlahkan seluruh bobot tiap *edge* dan tiap waktu_lintasan dari titik-titik yang terhubung dalam lintasan.

Script yang memperlihatkan proses penentuan himpunan titik dan penghitungan total waktu dan total jarak lintasan terpendek:

```
if($notshort==true){
                $weight=$minbobot;
                $jalur[0]=" ";}
else{
                $temp=$vakhir;
                $k5=-1;
                $nopath=false;
                while($nopath==false){
                        $k5+=1;
                        $jalur[$k5]=$temp;
                        $temp=$path[$jalur[$k5]];
                        if($temp==-1){
                                $nopath=true;
                        }}
 $k6=$k5;
  $weight=0.00;
  $time=0;
  for($k7=$k6;$k7>0;$k7--){
        $weight+=$edge[$jalur[$k7]][$jalur[$k7-1]];
        $time+=$waktu_lintasan[$jalur[$k7]][$jalur[$k7-1]];
 }
}
        $bobot[$d]=$weight;}
…
```

Script shorthpath.php diatas untuk penentuan rute terpendek berdasarkan jarak lintasan. Untuk penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan kemacetan, terdapat sedikit perbedaan pada saat penghitungan total waktu lintasan, dari jalur yang telah diperoleh apabila jalan yang akan dilalui terindikasi kemacetan maka bobot waktu akan dikalikan dengan skala, yaitu 4 dengan tujuan untuk menghasilkan waktu lintasan yang yang lebih besar dari waktu semula (secara umum apabila jalan yang dilalui macet maka waktu yang ditempuh akan lebih lama).

S*cript* yang memperlihatkan bagaimana proses perhitungan total waktu dengan melakukan pengecekan waktu lintasan tiap jalan yang dilalui dalam shortpathjarakdenganmacet.php:

```
…
$k6=$k5;
$weight=0.00;
$timemacet=0;
for($k7=$k6;$k7>0;$k7--){
  $weight+=$edge[$jalur[$k7]][$jalur[$k7-1]];
  include("berangkat.php");
if(($skala==$macet_pagi[$jalur[$k7]][$jalur[$k7-1]]) or
            ($skala==$macet_siang[$jalur[$k7]][$jalur[$k7-1]]) or
            ($skala==$macet_sore[$jalur[$k7]][$jalur[$k7-1]])){
        $time=$waktu_lintasan[$jalur[$k7]][$jalur[$k7-1]]*4;
}else{
        $time=$waktu_lintasan[$jalur[$k7]][$jalur[$k7-1]]*1;}
$detik=$menit*60+$time;
$menit=round($detik/60);
$detik=$menit%60;
if($detik>=60){
        $menit+=1;}
if($menit>=60){
        $jam+=floor($menit/60);
        $menit=$menit%60;}
$berangkat=sprintf("%02s.%02s",$jam,$menit);
$timemacet+=$time;
}}
$berangkatlagi[$d]=$berangkat;
$bobot[$d]=$weight;
…
```

Dari waktu lintasan yang diperoleh karena hasil akhirnya dalam *format* detik, maka untuk melakukan pengecekan pada setiap lintasan jalan raya perlu diubah dalam format jam dan menit, dengan pembagian 60 untuk mendapatkan *format* menit, hasil dari menit apabila lebih dari 60 maka dilakukan *modulus* (sisa pembagian) dengan 60.

4.1.6.7 *Script* jalur.php

Script jalur.php ini berfungsi untuk menentukan jalur baru atau lintasan baru apabila terjadinya kemacetan, yang hanya digunakan untuk *item* berdasarkan jarak lintasan dengan mempertimbangkan kemacetan. Dalam penentuan himpunan titik dan perhitungan total jarak serta total waktu pada penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan kemacetan, yaitu dengan

melakukan pengecekan waktu pada setiap lintasan jalan yang diperoleh, di mana struktur data dari pengecekan waktu sama dengan pengecekan waktu pada penentuan rute berdasarkan jarak tanpa mempertimbangkan kemacetan. Jika terdapat kemacetan maka akan dicarikan jalur lain yang dapat menghindari kemacetan dengan cara memperbesar bobot jarak pada lintasan jalan yang terjadi kemacetan (secara umum apabila jalan yang dilalui macet maka waktu yang ditempuh akan lebih lama, waktu yang lama tersebut seharusnya dapat untuk melintasi jalan yang lebih panjang) dengan dikalikan skala sebesar 4. Proses penentuan himpunan titik dan penghitungan total jarak dalam *weight* dan waktu lintasan dalam *time* untuk penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan kemacetan yang disimpan dalam *file* jalur.php:

```
<?
if(($vx[0]!=$node_wisata[$d]) and
($node_terpilih[$node_wisata[$d]]==0)){
$time=0;
for($k7=$k5;$k7>0;$k7--){
        include("berangkat.php");
        if(($skala!=0) and
(($macet_pagi[$jalur[$k7]][$jalur[$k7-1]]==$skala) or
($macet_siang[$jalur[$k7]][$jalur[$k7-1]]==$skala) or
($macet_sore[$jalur[$k7]][$jalur[$k7-1]]==$skala))){
                $edge[$jalur[$k7]][$jalur[$k7-1]]*=4;
        }
        else{
                $edge[$jalur[$k7]][$jalur[$k7-1]]*=1; }
$time=$waktu_lintasan[$jalur[$k7]][$jalur[$k7-1]];
$detik=$menit*60+$time;
$menit=round($detik/60);
$detik=$menit%60;
if($detik>=60){
        $menit+=1;}
if($menit>=60){
        $jam+=floor($menit/60);
        $menit=$menit%60;}
$berangkat=sprintf("%02s.%02s",$jam,$menit); }

$berangkat=$jamberangkat;
include("shortpathjarakdenganmacet.php");
                $k11=$k6;
                include("pilih1.php"); }}
?>
```

4.1.6.8 *Script* indekss.php

Script ini berfungsi untuk mendapatkan indek terkecil dari semua bobot hasil perhitungan jarak antara hotel dan tempat wisata, atau tempat wisata ke tempat wisata lainnya, yang akan digunakan dalam pencarian urutan wisata. *Script* yang memperlihatkan pencarian indek terkecil:

```
<?
$bobotmin=$bobot[0];
$idx=0;
for($i=0;$i<$n;$i++){
        if ($bobot[$i]<$bobotmin){
                $bobotmin=$bobot[$i];
                $idx=$i;}
        else $bobotmin = $bobotmin;
                $idx=$idx;}
$idxs[$f]=$idx;
?>
```

4.1.6.9 *Script* totaldantujuan.php

Script ini digunakan untuk menghitung total jarak dan total waktu ke semua tempat wisata yang ingin dikunjungi, mendapatkan urutan tempat wisata dan mengembalikan node hasil dari titikasal.php atau titiktujuan.php ke nama jalan sebenarnya. *Script* yang merupakan proses untuk mendapatkan apa yang diterangkan diatas:

```
<?
        $cost+=$bobot[$idx];
        $cost_length=$cost/1000;      //kilometer
        $cost_length1=$cost%1000;     //meter
        $total+=$cost_time[$idx];
        $bagi=$total/60;
        if($bagi>=60){
                $plusjam=floor($bagi/60);
                $menit=round(($total%3600)/60);
                $detik=$menit%60;}
        else{$menit=round($bagi);
             $detik=$bagi%60;}
        $hour_time=$singgah+$plusjam;
        $mnt_time=$menit;
        $tujuan_wisata[$f]=$node_wisata[$idx];
        $nodehtl=$tujuan_wisata[$f];
        $node_terpilih[$node_wisata[$idx]]=1;
        $namawisata[$f]=$titik_tujuan[$idx];
?>
```

Dari total waktu tempuh yang diperoleh karena hasil akhirnya dalam *format* detik, maka untuk ditampilkan di *browser* perlu diubah dalam format jam dan menit, dengan pembagian 60 untuk mendapatkan *format* menit yang disimpan dalam variabel bagi, hasil dari variabel bagi apabila lebih dari sama dengan 60 maka jam ditambah hasil bagi antara variabel bagi dengan 60 dan *modulus* (sisa pembagian) antara variabel menit dengan 60 untuk mendapatkan *format* detik. Jika hasil variabel bagi kurang dari 60 maka menit adalah variabel bagi dan detik adalah modulus antara variabel bagi dengan 60.

4.1.6.10 *Script* namajalan.php

Script ini berfungsi untuk menyimpan nama jalan yang dilalui ke dalam *database* tabel rute, yang nantinya akan digunakan oleh rutejalan.php. *Script* yang memperlihatkan proses menyimpan nama jalan:

```
…
$k9=0;
for($k8=$k11;$k8>0;$k8--){
        $namajln[$k9]=$namajalan[$jalur[$k8]][$jalur[$k8-1]];
        $sqljlr="INSERT INTO rutemacet (jalur,tujuan,f,d)
            VALUES ('$namajln[$k9]', '$node_wisata[$d]', '$f',
            '$d')";
                                $tambahjlr=mysql_query($sqljlr);
                                $k9++;}
…
```

4.1.6.11 *Script* rutejalan.php

Script ini digunakan untuk menampilkan rute jalan yang dilalui pada browser, prosesnya adalah:

```
<?
for($f=0;$f<$n;$f++){
        echo "Rute Jalan dalam Perjalanan Wisata yang akan
        dikunjungi untuk $namawisata[$f]: ";
        $queryjalan="SELECT DISTINCT jalur FROM rutemacet
        where f='$f' and d='$idxs[$f]'";
        $resultjalan=mysql_query($queryjalan);
    $jln=0;
    while($rowjalan=mysql_fetch_array($resultjalan)){
        $jalantujuan[$jln]=$rowjalan[jalur];
        if($jalantujuan[$jln]!=null){
            echo "-> Jalan $jalantujuan[$jln] ";
        }
    $jln++;
}}
…
```

4.1.6.12 *Script* urutanwisata.php

Script ini berfungsi untuk menampilkan urutan wisata pada browser, baik untuk satu pilihan tempat wisata ataupun lebih, dapat dilihat pada *script* urutanwisata.php:

```
<?
        echo "Perjalanan Wisata yang akan dikunjungi : ";
        for($f=0;$f<$n;$f++){
                echo "  -> $namawisata[$f] ";
        }
?>
```

4.1.6.13 *Script* WaktuSinggah.php

Script ini berfungsi untuk menampilkan inputan waktu singgah yang digunakan agar didapatkan lama perjalanan sesuai dengan keinginan *user* pada setiap tempat wisata:

```
...
for ($i=0; $i<$n_tujuan; $i++) {
  $name = 'waktu_'.$i;
  $name2= 'tujuan_'.$i;
  echo '<tr><td align="center">'.($i+1).'</td>
  <td align="center">'.$titik_tujuan[$i].'</td>
  <input type="hidden" name="'.$name2.'"
  value="'.$titik_tujuan[$i].'">
  <td align="center"><input type="text"
  name="'.$name.'"></td></tr>';}
  echo '<tr><td colspan="3" align="center">
  <input type="submit" value="Kirim" name="kirim" />
  <input type="button" value="Kembali"
  onclick="self.history.back()"</td></tr>'; echo'</table>';
  include("titiktujuan.php");
  for ($g=0; $g<$n_tujuan; $g++) {
        $name3 = 'node_'.$g;
        $name4 = 'xnode_'.$g;
        $name5 = 'ynode_'.$g;
        echo '<input type="hidden" name="'.$name3.'"
         value="'.$titikwisata[$g].'">';
        echo '<input type="hidden" name="'.$name4.'"
         value="'.$x_nodewisata[$g].'">';
        echo '<input type="hidden" name="'.$name5.'"
         value="'.$y_nodewisata[$g].'">';}
…
```

4.1.6.14 *Script* Utama

Script ini merupakan *script* induk sebagai tempat berlangsungnya proses dari mulai koneksi ke *database server* sampai pembuatan paket data lintasan terpendek yang siap ditampilkan oleh aplikasi penentuan rute perjalanan wisata.

File-file *include* yang dipanggil dalam *script* utama adalah connect.php, titikasal.php, titiktujuan.php, initgraph.php, sortpath.php untuk jarak lintasan, shortpathjarakdenganmacet.php dan jalur.php untuk jarak lintasan dengan mempertimbangkan kemacetan, namajalan.php, indekss.php, serta totaldantujuan.php.

Hasil dari pemanggilan *script* titikasal.php dan titiktujuan.php adalah *node* awal dan *node* tujuan beserta koordinatnya. Setelah

diketahui koordinat dari masing-masing *node*, dilakukan pencarian *node-node* yang termasuk dalam himpunan graf.

Jika *node* awal dan *node* tujuan saling beririsan tidak perlu dilakukan pencarian lintasan terpendek karena sudah diketahui bahwa bobotnya adalah 0. Jika tidak beririsan maka akan dilakukan pencarian lintasan terpendek dari *node* awal ke *node* tujuan dengan memanggil *script* shortpath untuk masing-masing *item*, yang akan menghasilkan lintasan terpendek.

Output dari *script* utama ini adalah data *text* urutan tempat wisata, nama jalan lintasan yang akan dilalui, total waktu dan total jarak.

Untuk masing-masing *item* penentuan memilliki *script* utama yang berbeda, berdasarkan jarak lintasan dapat dilihat pada *script* utama yaitu hasiljarak.php:

```
<?
include("connect.php");
$minbobot=100000;
$cost=0.00;
$total=0;
if((!empty($asal)) & (!empty($tujuan)) & (!empty($stay)) &
(ereg("^([0-9]{1,2})$", $stay))){
        include ("titikasal.php");
        $nodehtl=$Node_Hotel;
        include ("titiktujuan.php");
        $node_terpilih[$nodewisata[$d]]=0;
        include("initgraph.php");
        if(($nodehtl!=$node_wisata[$d]) and
              $node_terpilih[$node_wisata[$d]]==0)){
            for($f=0;$f<$n;$f++){
                for($d=0;$d<$n;$d++){
                    $node_wisata[$d]=$nodewisata[$d];
                    include("shortpath.php");
                    $singgah=$total_waktu;
                    $cost_time[$d]=$time;
                    $k11=$k5;
                    include("namajalan.php");}
        include("indekss.php");
        include("totaldantujuan.php");}}
}else{
        $komentar="Salah Satu Inputan yang Anda Masukan
Salah";
        $komentar0="Masukkan waktu Singgah yang Benar format
angka(numeric)";
        echo $komentar;
        echo "<br>";
        echo " 1.  $komentar0";}
?>
```

Penentuan berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan dapat dilihat pada *script* dibawah, yang disimpan dalam hasiljarakmacet.php:

```
…
$jamberangkat=$berangkat;
$x=0;
if(($nodehtl!=$node_wisata[$d])
          and($node_terpilih[$node_wisata[$d]]==0)){
  for($f=0;$f<$n;$f++){
    for($d=0;$d<$n;$d++){
                $node_wisata[$d]=$nodewisata[$d];
                include("shortpathjarakdenganmacet.php");
                $singgah=$total_waktu;
                $cost_time[$d]=$timemacet;
                $k11=$k5;
                include("namajalan.php");
                if($d==$d+$x){
                   $berangkat=$jamberangkat;
       }}
                include("indekss.php");
                include("totaldantujuan.php");
      if ($f==$f+$x){
        $jamberangkat=$berangkatlagi[$idx];
        include("explode.php");
        $jamberangkat=sprintf("%02s.%02s",$jam+$waktu_singgah
        [$idx],$menit);}}}}
…
```

Pemanggilan *script* explode.php adalah untuk memisahkan antara jam dan menit pada jam keberangkatan, misalnya pukul 11.30, maka dengan jam = 11 dan menit = 30.

Setelah dipisahkan, diproses untuk menambahkan jam pada waktu keberangkatan selanjutnya dan kemudian digabung kembali untuk diproses kembali pada saat melakukan pengecekan waktu kemacetan pada setiap jalan.

Penentuan rute dengan mempertimbangkan kemacetan disimpan dalam *script* utama yaitu hasiljaraktanpamacet.php:

```
…
if(($nodehtl!=$node_wisata[$d]) and
($node_terpilih[$node_wisata[$d]]==0)){
        for($f=0;$f<$n;$f++){
          for($d=0;$d<$n;$d++){
            if($d==$d+$x){
                $berangkat=$jamberangkat;}
            $node_wisata[$d]=$nodewisata[$d];
            include("shortpathjarakdenganmacet.php");
            $k11=$k6;
            include("pilih0.php");
            include("jalur.php");
            $singgah=$total_waktu;
            include("ambil0.php");
            include("ambil1.php");
                if($waktu0>$waktu1){
                   $cost_time[$d]=$waktu1;
                   $bobotfree[$d]=$jarak1;
                   $berangkatlagi[$d]=$berangkat1;
                   $ambil=1;}
                else{
                   $cost_time[$d]=$waktu0;
                   $bobotfree[$d]=$jarak0;
                   $berangkatlagi[$d]=$berangkat0;
                   $ambil=0;}
                include("rutediambil.php");}
                include("indeksfree.php");
                include("totaldantujuanfree.php");
                if ($f==$f+$x){
                   $jamberangkat=$berangkatlagi[$idx];
                   include("explode.php");
        $jamberangkat=sprintf("%02s.%02s",$jam+$waktu_singgah
[$idx],$menit);
        }}}
...
```

Perbedaannya dengan kedua *script* utama yang lain yaitu hanya pada *script* hasiljaraktanpamacet.php, melakukan proses pembandingan dengan hasil yang diperoleh dari *script* hasiljarakmacet.php, yaitu apabila lintasan yang dihasilkan oleh *script* hasiljarakmacet.php mengalami kemacetan, maka akan dicari lintasan lain yang dapat menghindari kemacetan, namun apabila lintasan yang didapat juga terdapat kemacetan, maka dilakukan pembandingan hasil waktu yang diperoleh dari keduanya agar didapatkan waktu yang lebih efisien.

Sedangkan pemanggilan dari pilih0.php adalah untuk mengambil hasil-hasil yang diperoleh dari script shortpathjarakdenganmacet.php untuk dimasukkan ke dalam database yang kemudian diambil oleh ambil0.php, ambil1.php dan dilakukan proses pengecekan hasil waktu mana yang lebih kecil, hasil akhirnya dimasukan ke dalam database dengan *script* rutediambil.php, hasil akhir berupa nama jalan yang merupakan rute yang diambil.

Implementasi aplikasi untuk pembuatan titik pada peta yang didasarkan oleh data yang disimpan dalam *database*, dibentuk dari beberapa *script* PHP, yaitu:

4.1.6.15 *Script* map.php

Script ini berfungsi untuk menggambar titik pada peta. Baik titik hotel maupun titik wisata. Pada penambahan titik yang dibutuhkan adalah data lokasi yang tersimpan pada *database*. Pengambilan data lokasi ini untuk sementara masih memerlukan alat bantu dari *software Handy ImageMapper*. *Script* yang merupakan proses menggambar titik pada peta dalam map.php:

```
…
header("content-type:image/jpeg");
$img=imagecreatefromjpeg("images/malangtanpalabel.jpg");
    $blue=imagecolorallocate($img,0,0,255);
    $red=imagecolorallocate($img,255,0,0);
  $sql1="SELECT lokasi from hotel where cek='0' ";
  $ambil=mysql_query($sql1);
  while($baris=mysql_fetch_array($ambil)){
      $lokasi=explode(",", $baris[lokasi]);
      $lokasi[count($lokasi)]=$lokasi[0];
      $lokasi[count($lokasi)]=$lokasi[1];
      imagefilledpolygon($img,$lokasi,count($lokasi)/2,$blue);
      imagepolygon($img,$lokasi,count($lokasi)/2,$blue);}
  $sql2="SELECT lokasi from wisata where cek='0' ";
  $ambil1=mysql_query($sql2);
  while($baris1=mysql_fetch_array($ambil1)){
      $lokasi=explode(",", $baris1[lokasi]);
      $lokasi[count($lokasi)]=$lokasi[0];
      $lokasi[count($lokasi)]=$lokasi[1];
      imagefilledpolygon($img,$lokasi,count($lokasi)/2,$red);
      imagepolygon($img,$lokasi,count($lokasi)/2,$red);}
       imagejpeg($img);
       imagedestroy($img);
```

Berdasarkan data pada lokasi, titik digambar dengan kotak segi empat. Penggambaran dilakukan dengan menggunakan fungsi *imagefilledpolygon* dan *imagepolygon.* Fungsi pertama digunakan untuk mengisi warna pada kotak tersebut, sedangkan fungsi kedua digunakan untuk mewarnai tepinya saja.

4.1.6.16 *Script* peta.php

Script ini digunakan untuk menampilkan informasi setiap titik dengan mengarahkan *cursor* atau meng-*click*-nya. *Script* yang memperlihatkan proses menampilkan informasi titik, baik titik hotel maupun wisata:

```
…
$query="SELECT lokasi,nama from hotel";
$result=mysql_query($query);
$stringPeta_hotel="";
while($row=mysql_fetch_array($result)){
 $lokasi=explode(",", $row[lokasi]);
 $stringPeta_hotel.="<AREA ALT=\"" .  $row[nama] .
 "\" SHAPE=\"poly\" COORDS=\"" .$row[lokasi] ."\"
HREF=\"detilhotel.php?id=" .$row[nama] ."\">\n";}
$query1="SELECT lokasi,nama from wisata";
$result1=mysql_query($query1);
$stringPeta_wisata="";
while($row1=mysql_fetch_array($result1)){
  $lokasi=explode(",", $row1[lokasi]);
  $stringPeta_hotel.="<AREA ALT=\"" .$row1[nama] .
  "\" SHAPE=\"poly\" COORDS=\"" .$row1[lokasi] ."\"
HREF=\"detilwisata.php?id=" . $row1[nama] ."\">\n";}
echo "<IMG SRC=images\arah.bmp BORDER=0>";
echo "<IMG SRC=\"map.php\" USEMAP=\"#malang\" BORDER=0>";
echo "<IMG SRC=images\legenda.bmp BORDER=0>";
?>
<MAP NAME="malang">
<?php
  print($stringPeta_hotel)
?>
</MAP>
```

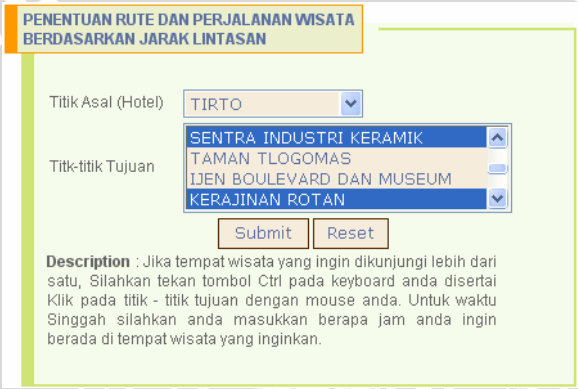
Dari titik yang telah dibuat agar *user* dapat memperoleh informasi tentang masing-masing titik, dibuat tag HTML yang disebut MAP. Daerah untuk setiap titik didefinisikan melalui tag AREA. SHAPE yang berisi “poly” menyatakan bahwa data pada COORDS ditangani dengan bentuk poligon. HREF menyatakan *script* yang akan dijalankan kalau daerah poligon yang disebutkan diklik. Sedangkan nama akan muncul jika *cursor* berada pada titik, yang digunakan sebagai identitas untuk mengakses detil dari titik.

4.2 Penerapan Aplikasi

User memasukkan titik hotel adalah Tirto dan titik tujuan wisata yang diinginkan adalah Sentra Industri Keramik dan Kerajinan rotan, waktu singgah yang diambil untuk tiap tempat wisata adalah sama yaitu 1 jam, serta waktu berangkat 08.00.

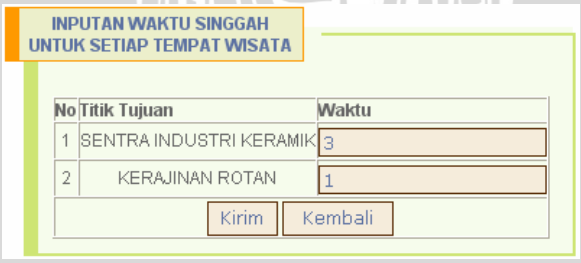
4.2.1 Jarak Lintasan.

Penerapan aplikasi untuk penentuan rute perjalanan wisata berdasarkan jarak lintasan berdasarkan inputan *user* dapat dilihat pada gambar 4.8.



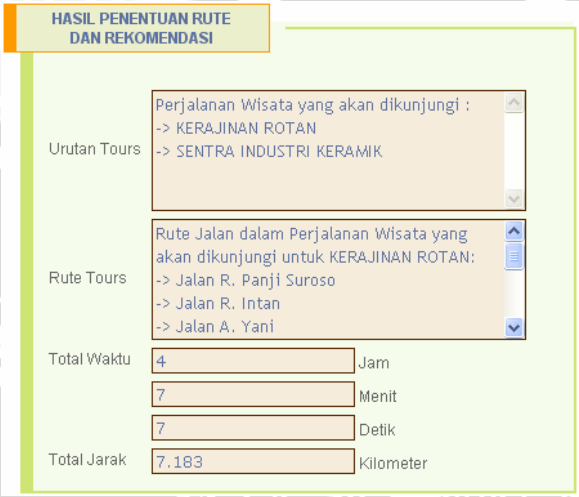
Gambar 4.8 Input untuk jarak lintasan

Setelah *user* melakukan *submit*, sebelum mendapatkan hasil penentuan rute dan urutan tempat wisata yang diingikan, *user* akan diminta untuk memasukkan masing-masing waktu singgah dari tempat wisata tersebut, dapat dilihat pada gambar 4.9.



Gambar 4.9 Input Waktu Singgah

Setelah *user* melakukan kirim, maka hasil dari proses yang dilakukan aplikasi penentuan rute perjalanan wisata terdapat pada gambar 4.10.



Gambar 4.10 Hasil jarak lintasan

Dapat dilihat hasil dari inputan *user* bahwa perjalanan wisata yang akan dikunjungi adalah:

1. KERAJINAN ROTAN
2. SENTRA INDUSTRI KERAMIK

Untuk rute jalan yang diambil adalah sebagai berikut:

Rute Jalan dalam Perjalanan Wisata yang akan dikunjungi untuk KERAJINAN ROTAN:

-> Jalan R. Panji Suroso
-> Jalan R. Intan
-> Jalan A. Yani

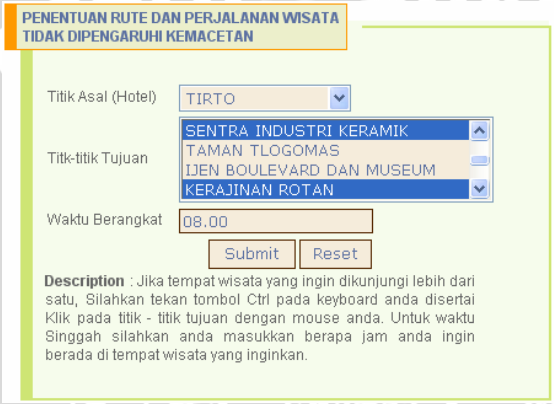
Rute Jalan dalam Perjalanan Wisata yang akan dikunjungi untuk SENTRA INDUSTRI KERAMIK:

-> Jalan A. Yani
-> Jalan Letjend. Suparman
-> Jalan Borobudur
-> Jalan Sukarno Hatta
-> Jalan Keramik

Total waktu yang ditempuh untuk waktu singgah dan keseluruhan perjalanan adalah 4 jam 7 menit dengan jarak tempuh 7.183 kilometer.

4.2.2 Jarak Lintasan tanpa mempertimbangkan kemacetan.

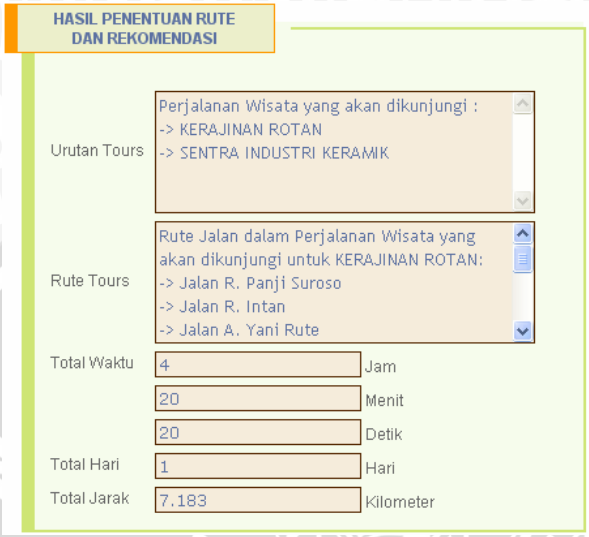
Penerapan aplikasi untuk penentuan rute perjalanan wisata berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan dapat dilihat pada gambar 4.11.



Gambar 4.11 Input jarak lintasan tanpa mempertimbangkan waktu terjadinya kemacetan

Setelah *user* melakukan *submit*, sebelum mendapatkan hasil penentuan rute dan urutan tempat wisata yang diingikan, *user* juga akan diminta untuk memasukkan masing-masing waktu singgah dari tempat wisata tersebut, dapat dilihat pada gambar 4.9.

Hasil dari proses yang dilakukan aplikasi penentuan rute perjalanan wisata terdapat pada gambar 4.12.



Gambar 4.12 Hasil jarak lintasan tanpa mempertimbangkan waktu terjadinya kemacetan

Urutan perjalanan wisata, rute jalan yang dilalui dan total jarak tempuh untuk jarak lintasan tanpa mempertimbangkan waktu terjadinya kemacetan sama dengan urutan, rute jalan yang dilalui dan total jarak tempuh untuk jarak lintasan.

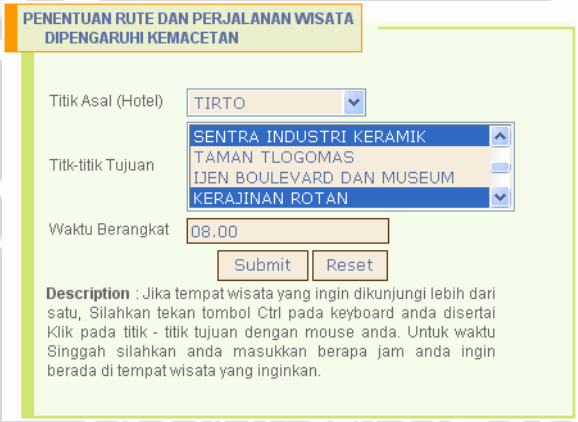
Total waktu yang ditempuh untuk waktu singgah dan keseluruhan perjalanan adalah 4 jam 20 menit, lebih lama dibandingkan dengan penentuan rute perjalanan berdasarkan jarak lintasan saja. Hal ini disebabkan karena karena pada jam tertentu, pada salah satu atau lebih jalan yang dilalui terdapat kemacetan. Lama perjalanan yang akan dilalui dengan jam berangkat dan waktu singgah yang diinputkan adalah 1 hari. Untuk kasus diatas, jalan yang mengalami kemacetan antara lain:

Tabel 4.1 Jalan yang mengalami kemacetan dan waktu kemacetan

Nama Jalan	Mulai waktu kemacetan
R. Intan	08.00
Borobudur	09.01
Sukarno Hatta	09.08

4.2.3 Jarak Lintasan dengan mempertimbangkan kemacetan.

Penerapan aplikasi untuk penentuan rute perjalanan wisata berdasarkan jarak lintasan dengan mempertimbangkan kemacetan dapat dilihat pada gambar 4.13.

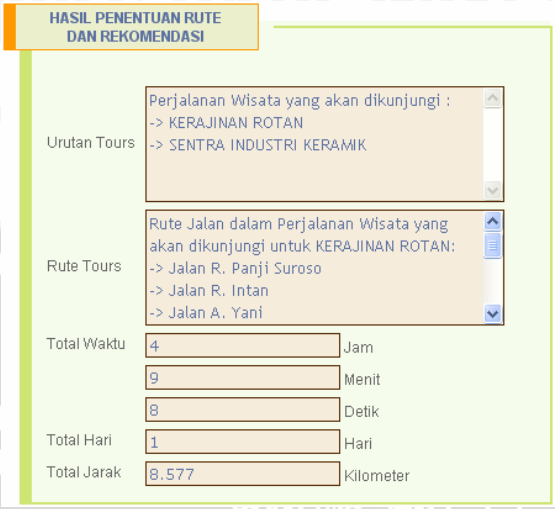


Gambar 4.13 Input jarak lintasan dengan mempertimbangkan waktu terjadinya kemacetan

Setelah *user* melakukan *submit*, sebelum mendapatkan hasil penentuan rute dan urutan tempat wisata yang diingikan, *user* juga akan diminta untuk memasukkan masing-masing waktu singgah dari tempat wisata tersebut, dapat dilihat pada gambar 4.9.

Setelah *user* melakukan Submit, maka hasil dari proses yang dilakukan aplikasi penentuan rute perjalanan wisata terdapat pada gambar 4.14

.



Gambar 4.14 Hasil jarak lintasan dengan mempertimbangkan waktu terjadinya kemacetan

Dapat dilihat hasil dari inputan *user* bahwa perjalanan wisata yang akan dikunjungi adalah:

1. Kerajinan rotan
2. Sentra Industri Keramik

Untuk rute jalan yang diambil adalah sebagai berikut:

Rute Jalan dalam Perjalanan Wisata yang akan dikunjungi untuk KERAJINAN ROTAN:

-> Jalan R. Panji Suroso
-> Jalan R. Intan
-> Jalan A. Yani

Rute Jalan dalam Perjalanan Wisata yang akan dikunjungi untuk SENTRA INDUSTRI KERAMIK:

-> Jalan Polowijen II
-> Jalan Sembilang
-> Jalan Cakalang
-> Jalan Tombro
-> Jalan Kakap
-> Jalan Gurami
-> Jalan Pohon Payung
-> Jalan Tunggul Yudo
-> Jalan Blimbing Indah Utara

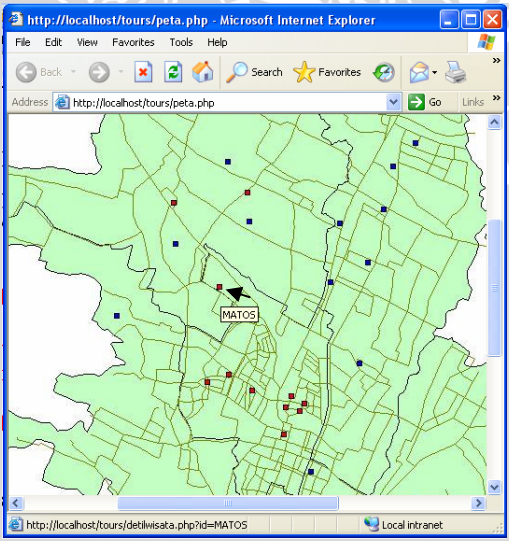
-> Jalan Paku Ningrat
-> Jalan Keramik

Total waktu yang ditempuh untuk waktu singgah dan keseluruhan perjalanan adalah 4 jam 9 menit dengan jarak tempuh 8.577 kilometer. Lama perjalanan yang akan dilakukan berdasarkan jam keberangkatan dan waktu singgah setiap tempat wisata adalah 1 hari.

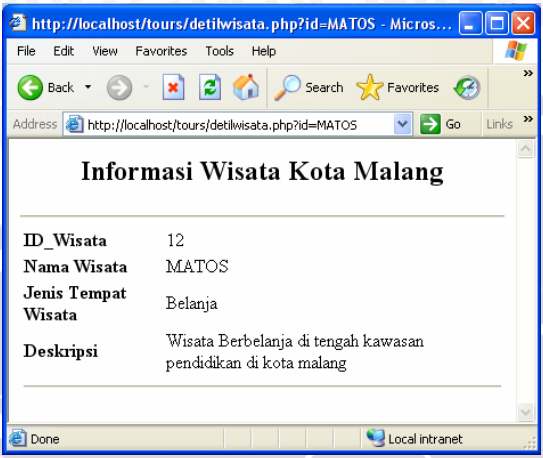
4.2.4 Peta

Peta digunakan untuk menampilkan informasi keberadaan hotel dan tempat-tempat wisata, tampilan utama peta pada saat *user* meng-*click option select* pada *browser* yang diperlihatkan pada gambar 4.6.

Jika *user* ingin mengetahui informasi titik-titik yang terdapat pada peta, maka *user* dapat meletakkan *cursor* pada titik dipilih, maka akan muncul informasi nama hotel jika titik yang dipilih berwarna biru atau nama tempat wisata jika titik yang dipilih berwarna merah, dan untuk detailnya user dapat meng-*click* titik tersebut. Gambar 4.15 dan 4.16 memperlihatkan informasi tersebut.



Gambar 4.15 Informasi nama wisata dengan meletakkan *cursor* di titik berwarna merah



Gambar 4.16 informasi detail dari titik yang dipilih

4.3 Analisa Hasil

Aplikasi yang sudah dibuat digunakan untuk melakukan pengujian seperti yang sudah dijelaskan pada subbab 3.5.5.

Untuk pengujian hotel yang dipakai adalah Montana Dua hotel, dan waktu singgah adalah sama yaitu 1 jam untuk mempermudah dalam perbandingan waktu tempuh lintasan yang diperoleh, serta nama-nama hotel yang dipakai adalah sebagai berikut:

1. 1 tempat wisata
    - Balai Kota & Alun Alun Bunder
2. 3 tempat wisata
    - Alun Alun Kota
    - Pulosari
    - Sentra Industri Keramik
3. 6 tempat wisata
    - Taman Tlogomas
    - Matos
    - Taman Senaputra
    - Balai Kota & Alun Alun Bunder
    - Pasar Burung Dan Pasar Bunga
    - Ijen Boulevard Dan Museum
4. 9 tempat wisata
    - Taman Tlogomas
    - Matos

- Pasar Wisata Tugu
- Ijen Boulevard Dan Museum
- Sentra Industri Keramik
- Balai Kota & Alun Alun Buder
- Pasar Burung Dan Pasar Bunga
- Taman Krida Budaya
- Pulosari

5. 13 tempat wisata
   - Alun Alun Kota
   - Pasar Burung Dan Pasar Bunga
   - Taman Senaputra
   - Taman Rekreasi Rakyat
   - Balai Kota & Alun Alun Bunder
   - Pasar Wisata Tugu
   - Ijen Boulevard Dan Museum
   - Pulosari
   - Matos
   - Sentra Industri Keramik
   - Taman Krida Budaya
   - Kerajinan Rotan
   - Taman Tlogomas

Penentuan rute perjalanan wisata berdasarkan jarak lintasan untuk total waktu dan total jarak, dapat dilihat pada tabel 4.2. Hasil yang diperoleh tersebut tidak dipengaruhi oleh waktu keberangkatan, sehingga berangkat pukul berapapun hasil yang diperoleh adalah sama.

Tabel 4.2 Penentuan total waktu dan total jarak berdasarkan jarak lintasan

Cost	**Banyaknya Tempat Wisata**				
	1	**3**	**6**	**9**	**13**
Waktu	1.6	3.9	6.15	9.19	13.29
Jarak	5.499	8.896	15.355	19.545	29.440

Keterangan:
Untuk waktu, tanda titik hanya untuk memisahkan jam dan menit. Misalnya 3.9 adalah 3 jam 9 menit.
Untuk jarak satuannya adalah Kilometer.

Penentuan rute perjalanan wisata berdasarkan jarak lintasan tanpa mempertimbangkan waktu kemacetan pada jam-jam tertentu, dapat dilihat pada tabel 4.3.

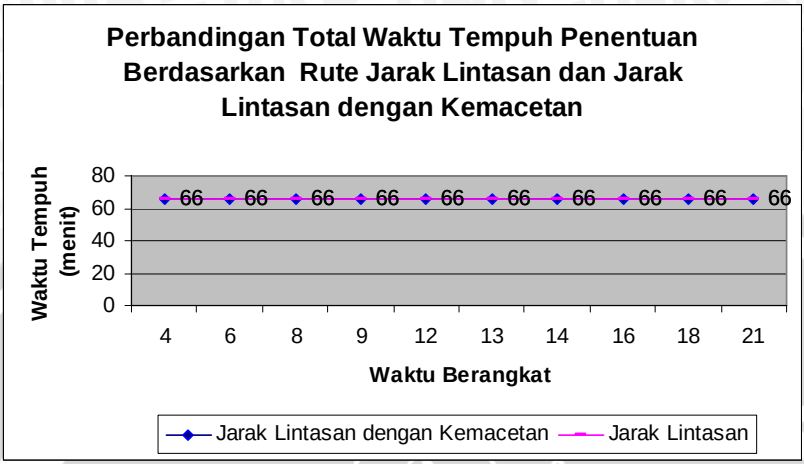
Tabel 4.3 Penentuan total waktu berdasarkan jarak lintasan tanpa mempertimbangkan waktu kemacetan

Waktu Berangkat	Banyaknya Tempat Wisata				
	1	3	6	9	13
04.00	1.6	3.9	6.26	9.28	13.45
06.00	1.6	3.15	6.26	9.42	13.52
08.00	1.6	3.21	6.35	9.35	13.37
09.00	1.6	3.15	6.35	9.32	13.34
12.00	1.6	3.21	6.35	9.35	13.45
13.00	1.6	3.15	6.24	9.35	13.45
14.00	1.6	3.21	6.21	9.25	13.35
16.00	1.6	3.21	6.24	9.27	13.37
18.00	1.6	3.9	6.15	9.19	13.29
21.00	1.6	3.9	6.15	9.19	13.29

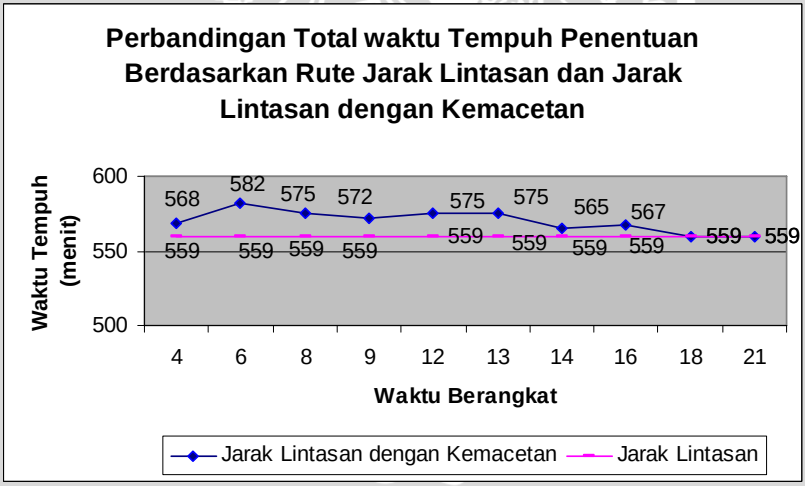
Keterangan:
Untuk waktu, tanda titik hanya untuk memisahkan jam dan menit. Misalnya 3.7 adalah 3 jam 7 menit.

Pada perhitungan total jarak tempuh dalam penentuan rute perjalanan wisata ini, memiliki nilai yang sama dengan total jarak yang ditempuh dalam penentuan rute perjalanan berdasarkan jarak lintasan, karena rute-rute yang ditempuh juga sama, hanya pada waktu-waktu tertentu, terdapat kemacetan pada suatu lintasan jalan yang membuat total waktu menjadi berbeda.

Perbandingan waktu tempuh yang dihasilkan antara penentuan berdasarkan jarak lintasan dan jarak lintasan tanpa mempertimbangkan kemacetan disajikan dalam grafik, waktu dijadikan dalam *format* menit dan tempat wisata yang digunakan adalah 1 tempat wisata dan 9 tempat wisata yang dapat dilihat pada gambar 4.17 dan 4.18.



Gambar 4.17 Grafik Perbandingan Total Waktu Tempuh antara Penentuan Jarak Lintasan dan jarak lintasan tanpa mempertimbangkan kemacetan untuk 1 Tempat Wisata



Gambar 4.18 Grafik Perbandingan Total Waktu Tempuh antara Penentuan Jarak Lintasan dan jarak lintasan tanpa mempertimbangkan kemacetan untuk 9 Tempat Wisata

Pada grafik gambar 4.17 dan 4.18 dapat dilihat bahwa waktu tempuh yang dihasilkan dalam penentuan rute berdasarkan jarak lintasan dengan kemacetan pada jam-jam tertentu lebih besar

dibandingkan dengan total waktu tempuh yang dihasilkan berdasarkan jarak lintasan, dan bernilai sama apabila waktu berangkat atau waktu lintasan bukan merupakan waktu-waktu sering terjadinya kemacetan sehingga jalan yang dilalui tidak mengalami kemacetan, seperti dicontohkan pada gambar 4.15. Waktu-waktu terjadinya kemacetan yaitu untuk pagi pukul 06.30 – 09.30, siang pukul 11.00 – 14.30, dan sore pukul 15.05 – 17.30 (Dinas Perhubungan) dan jalan yang sering terjadi kemacetan dapat dilihat pada lampiran 2. Begitu pula untuk kunjungan beberapa tempat wisata yang lain.

Penentuan rute perjalanan wisata berdasarkan jarak lintasan dengan mempertimbangkan waktu kemacetan pada jam-jam tertentu, dapat dilihat pada tabel 4.4.

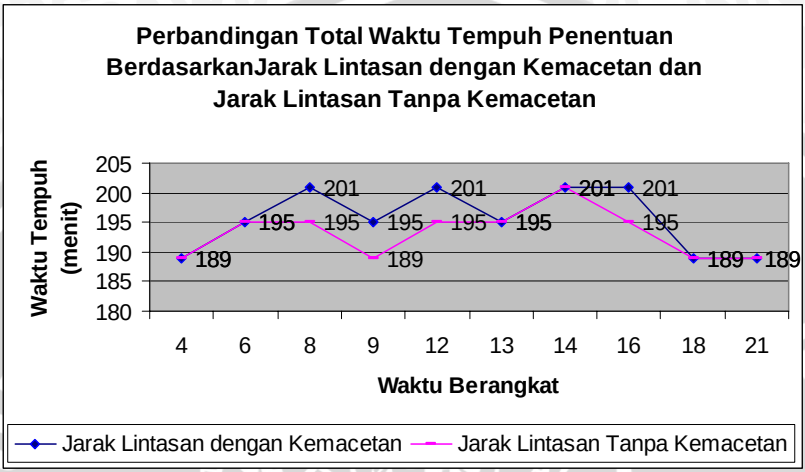
Tabel 4.4 Penentuan total waktu berdasarkan jarak lintasan dengan mempertimbangkan waktu kemacetan

Waktu Berangkat	Banyaknya Tempat Wisata				
	1	3	6	9	13
04.00	1.6	3.9	6.17	9.21	13.32
06.00	1.6	3.15	6.17	9.25	13.29
08.00	1.6	3.15	6.18	9.24	13.28
09.00	1.6	3.9	6.18	9.24	13.32
12.00	1.6	3.15	6.19	9.25	13.28
13.00	1.6	3.15	6.17	9.24	13.27
14.00	1.6	3.21	6.17	9.24	13.26
16.00	1.6	3.15	6.17	9.23	13.25
18.00	1.6	3.9	6.15	9.19	13.29
21.00	1.6	3.9	6.15	9.19	13.29

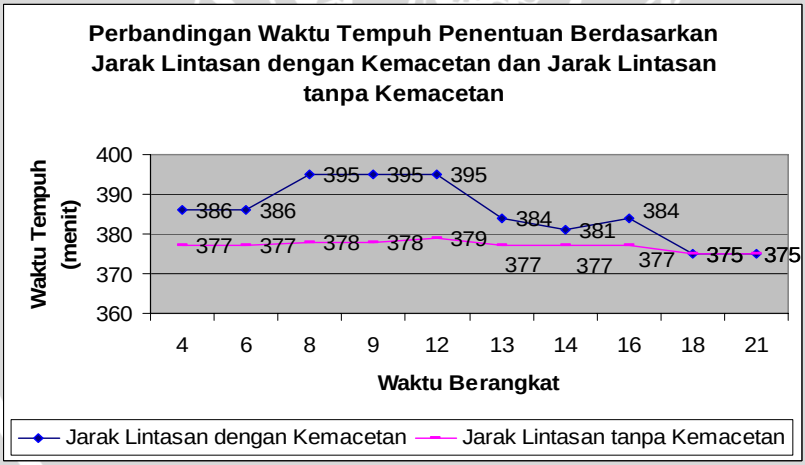
Keterangan:
Untuk waktu, tanda titik hanya untuk memisahkan jam dan menit. Misalnya 3.9 adalah 3 jam 9 menit.

Perbandingan waktu tempuh yang dihasilkan antara penentuan berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan dan jarak lintasan dengan mempertimbangkan kemacetan disajikan dalam grafik, untuk mempermudah waktu juga dijadikan dalam *format*

menit dan tempat wisata yang digunakan adalah 3 tempat wisata dan 6 tempat wisata yang dapat dilihat pada gambar 4.19 dan 4.20.



Gambar 4.19 Grafik Perbandingan Total Waktu Tempuh antara Penentuan Jarak Lintasan tanpa mempertimbangkan kemacetan dan jarak lintasan dengan mempertimbangkan kemacetan untuk 3 Tempat Wisata



Gambar 4.20 Grafik Perbandingan Total Waktu Tempuh antara Penentuan Jarak Lintasan tanpa mempertimbangkan kemacetan dan jarak lintasan dengan mempertimbangkan kemacetan untuk 6 Tempat Wisata

Pada grafik gambar 4.19 dan 4.20 dapat dilihat bahwa waktu tempuh yang dihasilkan dalam penentuan rute berdasarkan jarak lintasan dengan kemacetan pada jam-jam tertentu juga lebih besar dibandingkan dengan total waktu tempuh yang dihasilkan berdasarkan jarak lintasan tanpa kemacetan, dan juga bisa bernilai sama apabila waktu berangkat atau waktu lintasan bukan merupakan waktu-waktu sering terjadinya kemacetan, karena juga akan berakibat kemacetan pada sejumlah jalan yang rawan kemacetan pada jam-jam tersebut. Sama halnya untuk kunjungan beberapa tempat wisata yang lain.

Total jarak yang dihasilkan berbeda dengan jarak yang dihasilkan berdasarkan kedua penentuan rute yang lainnya, karena jarak yang dihasilkan lebih besar, sama ataupun lebih kecil bergantung pada waktu keberangkatan yang diinputkan. Tabel 4.3 memperlihatkan total jarak tempuh yang dihasilkan dalam penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan kemacetan.

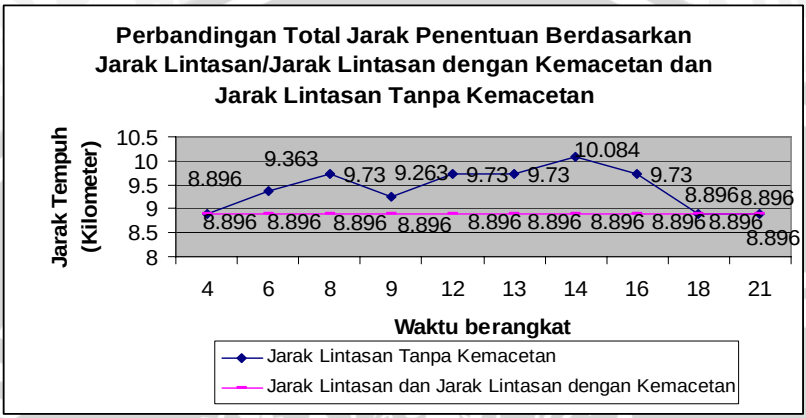
Tabel 4.5 Penentuan total jarak berdasarkan jarak lintasan dengan mempertimbangkan waktu kemacetan

Waktu Berangkat	Banyaknya Tempat Wisata				
	1	3	6	9	13
04.00	5.499	8.896	17.096	20.784	31.741
06.00	5.499	9.363	17.096	23.904	29.068
08.00	5.499	9.730	18.149	23.557	27.385
09.00	5.499	9.263	18.149	22.129	31.217
12.00	5.499	9.730	16.790	24.625	27.425
13.00	5.499	9.730	16.790	23.557	26.066
14.00	5.499	10.084	16.790	23.557	24.878
16.00	5.499	9.730	16.790	22.318	24.350
18.00	5.499	8.896	15.355	19.545	29.440
21.00	5.499	8.896	15.355	19.545	29.440

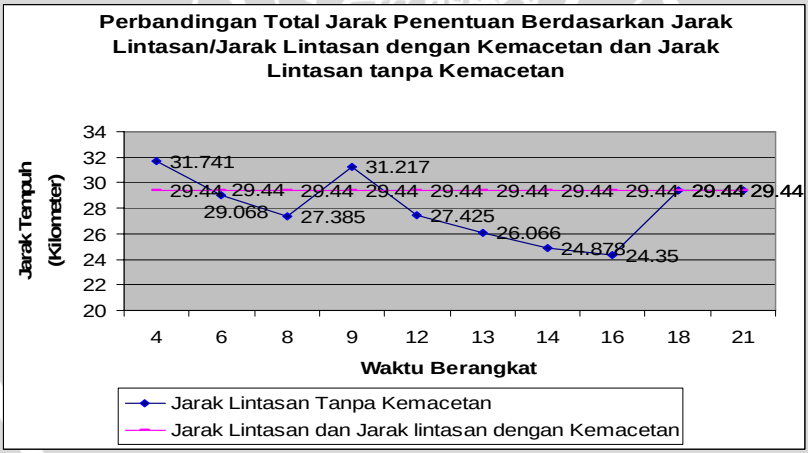
Keterangan:
Untuk jarak satuannya adalah Kilometer.

Penyajian untuk perbandingan jarak tempuh yang dihasilkan antara penentuan berdasarkan jarak lintasan baik jarak lintasan saja

maupun jarak lintasan tanpa mempertimbangkan kemacetan dan jarak lintasan dengan mempertimbangkan kemacetan pada pembuatan grafik, tempat wisata yang digunakan adalah 3 dan 13 tempat wisata yang dapat dilihat pada gambar 4.21 dan 4.22.



Gambar 4.21 Grafik Perbandingan Total Jarak Tempuh antara Penentuan Jarak Lintasan/Jarak Lintasan tanpa mempertimbangkan kemacetan dan Jarak Lintasan dengan mempertimbangkan Kemacetan untuk 3 Tempat Wisata



Gambar 4.22 Grafik Perbandingan Total Jarak Tempuh antara Penentuan Jarak Lintasan/Jarak Lintasan tanpa mempertimbangkan kemacetan dan Jarak Lintasan dengan mempertimbangkan Kemacetan untuk 13 Tempat Wisata

Dari grafik 4.21 dan 4.22 tersebut dapat diketahui bahwa penentuan rute berdasarkan jarak lintasan tanpa kemacetan atau dengan mempertimbangkan waktu terjadinya kemacetan memberikan jalur/rute lain dengan jarak tempuh yang dapat lebih besar dibandingkan dengan jarak tempuh yang dihasilkan jarak lintasan atau jarak lintasan dengan kemacetan, dapat dilihat pada subbab 4.2, bernilai sama apabila waktu berangkat bukan merupakan waktu terjadinya kemacetan dan lintasan jalan yang dilalui tidak mengalami kemacetan pada waktu-waktu tersebut seperti terlihat pada gambar 4.15 atau dikarenakan memang memiliki rute yang sama dengan penentuan berdasarkan jarak lintasan dengan kemacetan karena waktu tempuhnya yang lebih efisien, dan dapat pula bernilai lebih kecil, hal ini disebabkan karena rute yang didapatkan memang lebih pendek, dapat dilihat pada lampiran 4.

Jarak tempuh yang dihasilkan merupakan hasil dari pencarian rute terpendek karena pada saat pencarian, bobot jarak pada lintasan yang terjadi kemacetan diperbesar. Sebagai contoh, lintasan jalan tanpa kemacetan dengan jarak 100 meter dan waktu tempuh 1 menit, apabila terjadi kemacetan waktu tempuh menjadi 2 kali lipatnya, yaitu 2 menit, maka tanpa kemacetan seharusnya berdasarkan waktu tersebut jarak yang ditempuh adalah 200 meter.

Dari semua perbandingan yang dipaparkan diatas, waktu berangkat juga sangat berpengaruh pada hasil baik urutan, rute, total waktu maupun total jarak.

Dicontohkan pula bagaimana kerja sistem dalam membentuk suatu penentuan rute dan urutan perjalanan wisata untuk perhitungan manualnya yang dapat dilihat pada lampiran 3.

Dimisalkan bahwa rute ditentukan untuk hotel Kalpataru dan Taman Krida Budaya (untuk mempermudah dalam perhitungan manual) dan waktu berangkat pukul 08.00.

Dari *node-node* yang telah diperoleh untuk penentuan berdasarkan jarak lintasan didapat pula rute dari hotel Kalpataru ke Taman Krida Budaya, yaitu jalan Sukarno Hatta, Sukarno Hatta. Sukarno Hatta yang kedua adalah terusan Sukarno Hatta yang pertama.

Waktu yang diperlukan dari 87→67 = 52 detik, dan dari 67→53 = 19 detik. jadi total waktu yang dibutuhkan untuk 1188 m atau 1.188 km adalah 52 + 19 = 71 detik atau dengan pembulatan adalah sekitar 1 menit, tanpa waktu singgah.

Untuk perhitungan dalam penentuan rute perjalanan wisata berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan menghasilkan rute yang sama, hanya saja untuk total waktu yang ditempuh dikalikan dengan skala kemacetan yaitu 4 untuk jalan yang mengalami macet, dengan cara mengecek semua jalan yang akan dilewati berdasarkan waktu keberangkatan, misalnya berangkat pukul 08.00. Rute yang telah didapat diatas jalan Sukarno Hatta adalah jalan yang mengalami kemacetan pada pagi hari, sehingga waktunya dikalikan dengan skala 4, jadi total waktu yang ditempuh menjadi (52 x 4) + (19 x 4) = 284 detik atau dengan pembulatan adalah 5 menit, tanpa waktu singgah.

Node-node yang diperoleh pada penentuan berdasarkan jarak lintasan dengan mempertimbangkan kemacetan, didapatkan rute jalan yang dilalui dari hotel Kalpataru ke Taman Krida Budaya antara lain jalan Kalpataru, jalan Cengger Ayam, jalan Candi Mendut, jalan Candi Sari, jalan Candi Sari Utara, dan jalan Borobudur.

Rute baru tersebut ternyata pada pukul 08.03, jalan borobudur mengalami kemacetan, sehingga waktu yang diperlukan untuk menempuh jarak 3.85 km tersebut adalah 91 + 55 + 7 + 11 + 29 + (38 x 4) = 345 detik, atau dengan pembulatan waktu yaitu 6 menit, tanpa waktu singgah, waktu tersebut lebih lama dibandingkan dengan waktu pada rute yang diperoleh dengan penentuan rute berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan, maka dilakukan pembandingan waktu tempuh diantara hasil keduanya, rute yang mempunyai waktu yang lebih kecil dijadikan rute yang akan dilalui.

Berdasarkan lampiran 3, hasil yang dicapai baik dengan perhitungan manual maupun dengan sistem yang telah dibuat, menghasilkan nilai dan hasil yang sama.

Untuk waktu singgah yang berbeda-beda hanya akan merubah total lama perjalanan tidak merubah waktu tempuh lintasan, jadi dengan waktu singgah yang berbeda-beda pada setiap tempat wisata mengakibatkan berapa hari perjalanan wisata tersebut dapat dilalui.

4.4 Pengujian.

Untuk dapat menyimpulkan apakah *output* sistem sudah optimal atau belum, maka dilakukan pengujian terhadap bobot skala yang dipakai, jarak, rute dan waktu yang dapat dilihat pada tabel 4.6 dan 4.7.

Bobot skala ini hanya untuk menandakan waktu kemacetan yang memperbesar bobot jarak.

Titik asal dan titik tujuan yang dipakai adalah hotel Pelangi Dua dan Ijen Boulevard dan Museum, rute yang dihasilkan adalah sebagai berikut:

- ➔ Jalan Gajahyana (A)
- ➔ Jalan Sutami (B)
- ➔ Jalan Bondowoso (C)
- ➔ Jalan Retawu (D)
- ➔ Jalan Besar Ijen (E)

Pengujian bobot skala di mana waktu-waktu kemacetan tidak terjadi:

Tabel 4.6 Pengujian bobot skala pada waktu tanpa terjadi kemacetan

Waktu Berangkat	Bobot Skala	Rute	Total waktu	Total Jarak
10.00	1	A-B-C-D-E	1 jam 4 menit	4.185 Km
10.00	2	A-B-C-D-E	1 jam 4 menit	4.185 Km
10.00	3	A-B-C-D-E	1 jam 4 menit	4.185 Km
10.00	4	A-B-C-D-E	1 jam 4 menit	4.185 Km
10.00	5	A-B-C-D-E	1 jam 4 menit	4.185 Km

Berdasarkan uji coba yang dilakukan terhadap bobot skala pada waktu yang sama menghasilkan rute dan waktu yang sama, jadi bobot skala pada waktu-waktu tidak terjadi kemacetan tidak berpengaruh pada total waktu dan total jarak yang dihasilkan.

Pengujian bobot skala di mana waktu-waktu kemacetan sering terjadi:

Tabel 4.7 Pengujian bobot skala pada waktu sering terjadi kemacetan

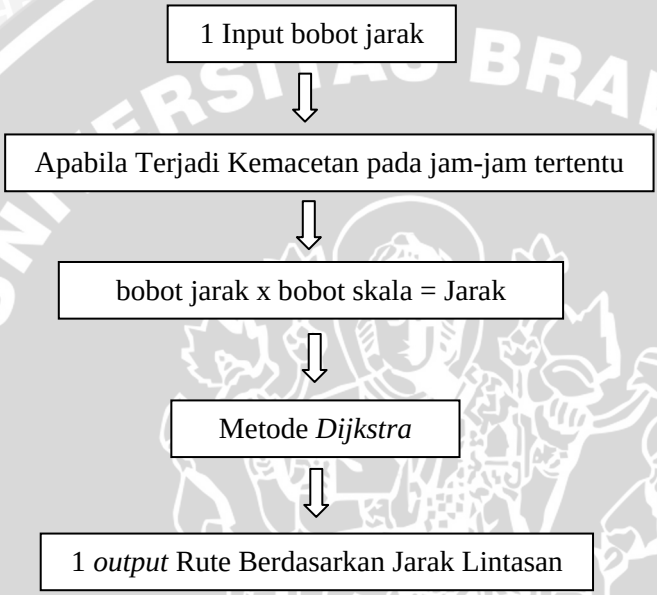
Waktu Berangkat	Bobot Skala	Rute	Total waktu	Total Jarak
08.00	1	A-B-C-D-E	1 jam 4 menit	4.185 Km
08.00	2	A-B-C-D-E	1 jam 5 menit	4.185 Km
08.00	3	A-B-C-D-E	1 jam 7 menit	4.185 Km
08.00	4	A-B-C-D-E	1 jam 9 menit	4.185 Km
08.00	5	A-B-C-D-E	1 jam 11 menit	4.185 Km

Berdasarkan uji coba yang dilakukan terhadap bobot skala pada waktu yang sama menghasilkan rute yang sama dan total waktu yang berbeda, jadi bobot skala pada waktu-waktu terjadinya kemacetan berpengaruh pada total waktu yang dihasilkan. Semakin besar bobot skala yang diberikan maka semakin besar pula total waktu yang dihasilkan.

Untuk rute yang dapat dihasilkan pada saat bobot diperbesar, juga akan menghasilkan total waktu dan total jarak yang sama untuk setiap waktu keberangkatan yang sama dan tidak terjadi kemacetan, untuk waktu-waktu terjadinya kemacetan bobot skala mempengaruhi rute yang akan dihasilkan, namun rute yang dihasilkan merupakan perhitungan yang lain dari metode *Dijkstra*, bukan merupakan suatu alternatif.

Dari analisis hasil diatas dapat dijelaskan bahwa sistem dengan menggunakan metode *Dijkstra* hanya mengeluarkan satu *output* yang dicontohkan pada hasil rute pada tabel 4.6 dan 4.7.

Penerapan Algoritma *Dijkstra* pada pencarian lintasan terpendek dimana menghasilkan solusi yang didapatkan dari satu aspek tanpa memperhatikan aspek lain, yaitu aspek jarak lintasan sehingga akan dicari lintasan terpendek berdasarkan jarak lintasan tanpa memperhatikan aspek yang lain seperti waktu terjadi kemacetan. Hal ini disebabkan karena:



Gambar 4.23 Skema hasil dari Algoritma *Dijkstra* dengan bobot skala

Dari skema diatas dapat diketahui bahwa input bobot yang dilakukan dengan menggunakan algoritma Dijkstra hanya satu yaitu jarak lintasan dan *output* yang dihasilkan juga satu rute berdasarkan jarak lintasan tersebut, tidak dapat memberikan alternatif solusi rute yang lain.

BAB V
PENUTUP

5.1 Kesimpulan

Beberapa kesimpulan yang dapat diambil dari hasil penelitian ini adalah:

1. Pada kasus serta analisa dari contoh tersebut, dapat disimpulkan bahwa pencarian lintasan terpendek dapat menghasilkan jalan tersingkat untuk menuju suatu tempat dimana terdapat suatu variabel lain yang juga mempengaruhi. Dalam hal ini adalah waktu kemacetan yang dapat digunakan sebagai variabel yang mempengaruhi kecepatan perjalanan.
2. Waktu tempuh yang dihasilkan dalam penentuan rute berdasarkan jarak lintasan dengan mempertimbangkan kemacetan dapat sama ataupun lebih efisien daripada berdasarkan jarak lintasan tanpa mempertimbangkan kemacetan, meskipun jarak yang ditempuh dapat lebih besar.
3. Penerapan Algoritma *Dijkstra* pada pencarian lintasan terpendek belum tentu menghasilkan nilai yang optimum karena hanya mengacu pada satu aspek dan akan mencari nilai maksimum pada aspek tersebut dengan mengabaikan aspek yang lain.

5.2 Saran

Saran yang ingin disampaikan penulis ialah agar para peneliti selanjutnya yang ingin melanjutkan penelitian ini dapat:

1. Memberikan suatu kemudahan pada *user* pada saat memilih hotel dan tempat wisata yang ingin dikunjungi ataupun hasil dari penentuan rute terpendek langsung berinteraksi pada peta, bukan berupa hasil tekstual namun berupa grafikal.
2. Selain itu, saran yang lain adalah adanya fasilitas *update* atau kemampuan untuk mengubah data yang telah ada. Hal tersebut berguna apabila terdapat data yang baru.

DAFTAR PUSTAKA

Anynomous, *Flowcharting Basic*, http://www.cic.ac.id/modulkuliah/download/Flowchart%201_files/frame.htm#slide0001.htm , tanggal akses : 3 maret 2007.

Anynomous, *toWKT : An ArcView 3.x Extension for Web Feature Serving*. http://geoserver.sourceforge.net, tanggal akses : 23 febuari 2007.

Aziz, Muh dan Slamet pujiono. 2006. *Sistem Informasi Geografis berbasis Desktop dan Web*. Gava Media, Yogyakarta.

Dwi Prasetyo, Didik. 2005. *Solusi Menjadi Web Master Melalui Manajemen Web dengan PHP*. Elex Media Komputindo, Jakarta.

Forta, Ben. 2000. *Belajar sendiri dalam 10 menit SQL*. Andi, Yogyakarta.

Hakim, Lukmatul dan Uus Musalini. 2004. *150 Rahasia dan Trik menguasai PHP*. Elex Media Komputindo, Jakarta.

http://www.sanctaursula-jkt.sch.id, tanggal akses : 4 mei 2007.

Mark, 1999, *Aplikasi web dengan PHP*, http://www.ikc.cbn.net.id, tanggal akses 23 febuari 2007.

Kurniawan S.T, Yahya. 2002. *Aplikasi web database dengan PHP dan MySQL*. Elex Media Komputindo, Jakarta.

Ladjamuddin, Al-Bahra, 2004. *Konsep Sistem Basis Data dan Implementasinya*. Graha Ilmu, Yogyakarta.

Munir, Rinaldi. 2001. *Matematika Diskrit*. Informatika, Bandung.

Neutron Yogyakarta. 2003. *Paket Teori Lengkap*. Neutron, Yogyakarta.

Purwanto, Yudhi. 2001. *Pemogramna Web dengan PHP*. Elex Media Komputindo, Jakarta.

Siang, Jong Jek. 2002. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Andi, Yogyakarta.

Syamsuddin, Aries dkk. 2007. *WebGIS Wisata Kota Malang*. Tugas Mata Kuliah Sistem Informasi Geografi, Malang.

Ariyanto, Taufiq Reza dkk. 2006. *Strategi Greedy pada Kasus Pencarian Lintasan Terpendek*. Sekolah Tinggi Teknologi Telkom, Bandung.

Tipler, Paul A. 1991. *Fisika untuk Sains dan Teknik*. Erlangga, Jakarta.

Yulia, kartika dan Jeffrey, *Perencanaan rute perjalanan di jawa timur Dengan dukungan gis menggunakan Metode dijkstra's* http://puslit.petra.ac.id/, tanggal akses : 31 September 2007.