IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)

SKRIPSI

Oleh :
HELGA ADITYA RIZQI GEOVANI
NIM. 09650157

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)

SKRIPSI

Diajukan Kepada :
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
HELGA ADITYA RIZQI GEOVANI
NIM. 09650157

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)

Oleh :
Helga Aditya Rizqi Geovani
NIM. 09650157

Telah Diperiksa dan Disetujui Untuk Diuji :

Tanggal, 16 Juni 2016

Dosen Pembimbing I	Dosen Pembimbing II
Hani Nurhayati, M.T	**A'la Syauqi, M.Kom**
NIP. 19780625 200801 2 006	**NIP. 19771201 200801 1 007**

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)

SKRIPSI

Oleh :
Helga Aditya Rizqi Geovani
NIM. 09650157

Telah Dipertahankan di Depan Dewan Penguji Tugas Akhir dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal, 23 Juni 2016

Susunan Dewan Penguji			Tanda Tangan
1. Penguji Utama	:	Dr. Muhammad Faisal, M.T NIP. 19740510 200501 1 007	( )
2. Ketua	:	Fresy Nugroho, S.T, M.T NIP. 19710722 201101 1 001	( )
3. Sekretaris	:	Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
4. Anggota	:	A'la Syauqi, M.Kom NIP. 19771201 200801 1 007	( )

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

Saya yang bertanda tangan di bawah ini :

Nama : Helga Aditya Rizqi Geovani

NIM : 09650157

Fakultas/Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : **IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)**

Menyatakan dengan sebenar-benarnya bahwa hasil penelitian saya ini tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Malang, 16 Juni 2016

Yang Membuat Pernyataan,

Helga Aditya Rizqi Geovani
NIM. 09650157

MOTTO

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ ٱسْتَعِينُواْ بِٱلصَّبْرِ وَٱلصَّلَوٰةِۚ إِنَّ ٱللَّهَ مَعَ ٱلصَّٰبِرِينَ

Hai orang-orang yang beriman, jadikanlah sabar dan shalat sebagai penolongmu. . Sesungguhnya Allah beserta orang-orang yang sabar.
(QS. Al-Baqarah : 153)

"Jika Engkau Tak Tahan Dengan Lelahnya Belajar, Maka Engkau Akan Menanggung Perihnya Kebodohan" (Imam Syafi'i)

PERSEMBAHAN

Dengan Berjuta Rasa Syukur Kepada Allah SWT Dan

Nabi Muhammad SAW,

Aku Persembahkan Karya Ini Kepada :

Orang Tuaku Tercinta

Kiptiyah** Dan **Suwardi

Yang Dengan Cintanya Dan Kasih Sayangnya Sangat Besar Merawat,

Mendidik Dan Membimbingku Sampai Saat Ini.

Keluarga Besarku, Mb Ida, Mas Mahrus, Mb Nur, Mb Fatim, Om Pipin, Mb Ut, Om Par, Mb Titik, Om Irul, Baihaqi, Abidzar, Syamsa, Taza Dan Mahera.

Dosen Waliku, Ibu Roro Inda Melani,M.T.,M.Sc

Sahabat Terbaikku Achmad Firdaus Sulthoni dan Nursih Dwi Hastuti,

Dan Teman Terbaikku Roro Maylinda Eka Putri.

Terima Kasih Yang Sedalam-Dalamnya Atas Dukungan Kalian Semua

Hingga Aku Dapat Menyelesaikan Skripsi Ini.

Semoga Allah SWT Senantiasa Memberikan Rahmat Dan Kasih Sayangnya

Selalu Di Dunia Dan Akhirat Kelak.

Amiin.

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Alhamdulillah, Penulis Panjatkan Syukur yang tak terhingga kepada Allah SWT yang Maha Pengasih dan Penyayang, dan Juga Shalawat dan Salam Penulis Panjatkan kepada Junjungan Kita Nabi Muhammad SAW. Terima Kasih atas Segala Waktu yang Telah di lalui, Kesehatan dan Nikmat yang tiada henti, sehingga Penulis dapat menyelesaikan karya ilmiah dengan judul "**IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENGETAHUI LOKASI TEMPAT IBADAH UMAT MUSLIM DI KOTA MALANG PADA APLIKASI MOBILE PHONE (STUDI KASUS TEMPAT IBADAH DI WILAYAH KECAMATAN LOWOKWARU)**". Penelitian ini dimaksudkan untuk memenuhi salah satu persyaratan untuk memperoleh gelar Sarjana Komputer (S.Kom) di Jurusan Teknik Indormatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Penulis Menyadari Penuh bahwa banyak pihak-pihak yang telah sangat membantu dalam menyelesaikan penulisan tugas akhir ini, Oleh karena itu, penulis iringkan doa dan ucapan terima kasih yang tak terhingga kepada :

1. Ibu Hani Nurhayati, M.T selaku Dosen Pembimbing I yang telah banyak meluangkan waktu dan membimbing dalam menyelesaikan skripsi ini, Semoga Allah SWT selalu melimpahkan rahmat dan kasih sayang-Nya kepada beliau sekeluarga.
2. Bapak A'la Syauqi, M.Kom selaku Dosen Pembimbing II yang telah banyak meluangkan waktu dan memberikan pengarahan dalam menyelesaikan skripsi ini, Semoga Allah SWT selalu melimpahkan rahmat dan kasih sayang-Nya kepada beliau sekeluarga.

3. Bapak Dr. Cahyo Crysdian selalu Ketua Jurusan Teknik Informatika yang telah banyak meluangkan waktu dan memberikan nasehat dan pengarahan dalam menyelesaikan skripsi ini, Semoga Allah SWT selalu melimpahkan rahmat dan kasih sayang-Nya kepada beliau sekeluarga.
4. Prof. DR. H. Mudjia Rahardjo, M.Si, selaku Rektor Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang beserta seluruh civitas akademik yang turut membesarkan dan mencerdaskan penulis.
5. Seluruh Dosen Teknik Informatika yang telah mendidik penulis dengan sangat baik sampai saat ini,
6. Seluruh Pihak Baik Teman-Teman yang Penulis tidak dapat menyebutkan satu persatu. Penulis ucapkan terima kasih atas bantuannya dan motivasinya.

Akhirnya atas segala kekurangan dari penyusunan skripsi ini, penulis mengharapkan kritik dan saran agar dapat membangun penulis menjadi lebih baik lagi dan dapat membuat karya-karya baru yang lebih bermanfaat. semoga apa yang telah tertulis di skripsi ini dapat memberikan manfaat sebaik mungkin dalam ilmu pengetahuan, Amiin.

Wassalamu'alaikum Wr. Wb

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Rizqi Geovani, Helga Aditya. 2016. **Implementasi Algoritma *Dijkstra* Untuk Mengetahui Lokasi Tempat Ibadah Umat Muslim Di Kota Malang Pada Aplikasi *Mobile Phone* (Studi Kasus Tempat Ibadah Di Wilayah Kecamatan Lowokwaru**). Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing : (1) Hani Nurhayati, M.T (II) A'la Syauqi, M.Kom

---

Kata Kunci : Tempat Ibadah, Masjid, Lokasi, Android, Optimasi, Algoritma Dijkstra

Perkembangan Teknologi membawa kita pada teknologi *mobile phone* menjadi teknologi *mobile smartphone*, dahulu kita mengenal telepon seluler sebagai perangkat teknologi yang mampu membantu kita berkomunikasi hanya untuk menelfon ataupun mengirimkan sms. Kemudian kita mengenal perangkat seluler yang dapat mengakses jaringan internet, yang memungkinkan kita untuk berkomunikasi menggunakan email. Kemudian kita berada pada teknologi perangkat seluler yang memungkinkan kita untuk berkomunikasi menggunakan begitu banyak aplikasi yang kita hanya perlu menambahkan pada perangkat seluler kita. Dengan perkembangan aplikasi pada perangkat seluler seperti pada saat ini, penulis mendapatkan ide untuk membuat aplikasi yang dapat mencari lokasi tempat ibadah umat muslim terdekat dan aplikasi ini dapat memberikan panduan rute terdekat untuk menuju lokasi tempat ibadah tersebut. Aplikasi ini di bangun untuk sistem operasi Android. Dalam aplikasi ini menampilkan pilihan menu untuk memilih tempat ibadah mana yang akan dituju dan aplikasi ini merepresentasikan hasilnya pada *Google Maps*. Algoritma yang digunakan pada aplikasi ini adalah Algoritma *Dijkstra*, Algoritma ini mampu menghitung dan menghasilkan perhitungan jarak yang optimal menuju lokasi tempat ibadah yang dituju. Dari hasil uji coba yang di lakukan, aplikasi dapat berjalan dengan baik pada semua *device* jika kualitas data internet yang di terima baik, bila data yang diterima kurang baik akan terjadi *loading* pada proses menampilkan peta *Google Maps*.

ABSTRACT

Rizqi Geovani, Helga Aditya. 2016. **Implementation *Dijkstra* Algorithm to Find Worship Muslim Place in Malang City On *Mobile Phone* Application (Case Study Worship Muslim Place in Lowokwaru Sub-District**). Department of Informatics Engineering, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang, Promotor : (1) Hani Nurhayati, M.T (II) A'la Syauqi, M.Kom

---

Keywords : Worship Location, Mosque, Location, Android, Optimization, Dijkstra Algorithm

Technology brings us to the development of mobile phone technology into mobile smartphone technology, we know the first cellular phone as a device technology capable of helping us communicate only would call or send sms. Then we know the mobile devices that can access the Internet, which allows us to communicate using email, Then we are on a mobile device technology that allows us to communicate using so many applications that we only need to add on our mobile devices. With the development of applications on mobile devices like at this point, the author got the idea to create applications that can locate the nearest place of worship of Muslims and this application can provide route guidance to the location closest to the place of worship. This application is built on the Android operating system. In this application displays a menu option to select a place of worship which will be addressed and this application represents the results on Google Maps. The algorithm used in this application is Dijkstra's Algorithm, this algorithm is capable of calculating and generating the optimal distance calculations to the designated place of worship. From the test is done, the applications can run well on all devices if the quality of internet data was well received, when the received data is less good will happen loading on the process of displaying a map Google Maps.

الملخص

رزقي جيوفاني، هيلغا أديتيا .٢٠١٦ .تنفيذ خوارزمية ديكسترا لمعرفة أماكن الموقع العبادة المسلمين في مالانج على تطبيقات الهاتف المحمول) دراسة حالة أماكن العبادة في منطقة لوو الكركديه المنطقة .(قسم المعلوماتية، كلية العلوم والتكنولوجيا، وجامعة ولاية الإسلامية مولانا مالك إبراهيم مالانج .مؤدب :١.هني نوره . م. ت ٢.الا سوق م.كوم

كلمات البحث :مكان العبادة، المسجد، الموقع، الروبوت، التحسين، خوارزمية ديكسترا

التكنولوجيا يقودنا إلى تطوير تكنولوجيا الهاتف المحمول في تقنيات الهاتف الذكي المحمول، ونحن نعلم أول هاتف الخلوي كتقنية جهاز قادرة على مساعدتنا التواصل فقط أن الاتصال أو إرسال الرسائل القصيرة .ثم نعرف الأجهزة النقالة التي يمكن الوصول إلى الإنترنت، والذي يسمح لنا بالتواصل باستخدام البريد الإلكتروني .ثم نحن على تكنولوجيا الأجهزة النقالة التي تتيح لنا التواصل باستخدام العديد من التطبيقات التي نحن بحاجة فقط لإضافة على الأجهزة النقالة لدينا .مع تطوير التطبيقات على الأجهزة المحمولة مثل عند هذه النقطة، وحصلت على المؤلف فكرة لإنشاء التطبيقات التي يمكن تحديد موقع أقرب مكان للعبادة للمسلمين ويمكن هذا التطبيق يوفر توجيه المسار إلى الموقع الأقرب إلى مكان للعبادة .تم بناء هذا التطبيق على نظام التشغيل أندرويد .في هذا التطبيق يعرض خيار القائمة لتحديد مكان العبادة التي سيتم تناولها، ويمثل هذا التطبيق النتائج على خرائط جوجل .الخوارزمية المستخدمة في هذا التطبيق هي خوارزمية ديكسترا، هذه الخوارزمية هي قادرة على حساب وتوليد حسابات المسافة المثلى إلى المكان المخصص للعبادة .من الانتهاء من الاختبار، يمكن للتطبيقات تعمل بشكل جيد على كافة الأجهزة إذا كانت نوعية البيانات الانترنت استقبالا حسنا، عندما البيانات الواردة هو أقل جودة سيحدث تحميل على عملية عرض خريطة خرائط جوجل.

BAB I
TINJAUAN PUSTAKA

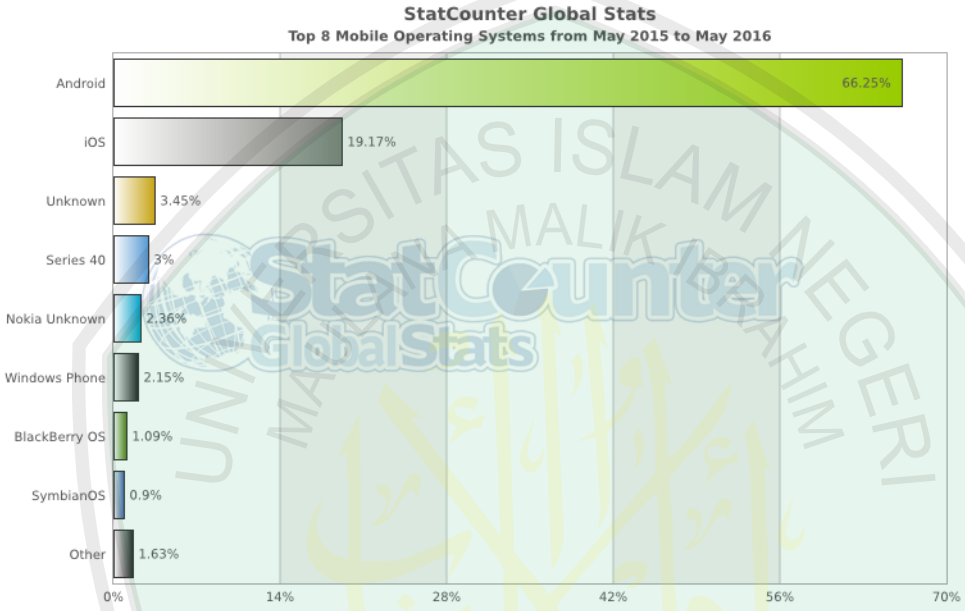
1.1 Latar Belakang

Pada saat ini kebutuhan akan memperoleh dan mengirim informasi dengan cepat sangatlah penting, dan ini tentunya berimbas pada perkembangan teknologi informasi, beragam teknologi informasi telah di bangun untuk dapat menyediakan layanan yang dapat memenuhi permintaan pengguna, baik di kota maupun di pedesaan.

Dalam perkembangannya, saat ini teknologi informasi yang paling berkembang adalah teknologi informasi berbasis *mobile phone*, pengguna saat ini lebih menyukai segala sesuatu yang di kemas dalam aplikasi berbasis *mobile*. Aplikasi berbasis *mobile* mempunyai beberapa keunggulan, salah satunya yaitu dapat di jalankan di ponsel, sehingga pengguna dapat mengakses dimanapun mereka berada, dengan di dukungnya jaringan internet saat ini maka aplikasi berbasis *mobile* akan mempunyai kelebihan tersendiri bagi pengguna. Terdapat berbagai macam sistem operasi *mobile* yang berkembang saat ini, diantaranya adalah sistem operasi Android, Blackberry dan iOS. Diantara sistem operasi tersebut yang paling menyita perhatian bagi penulis adalah sistem operasi Android, karena sistem operasi Android merupakan sistem operasi yang bersifat terbuka dan boleh di kembangkan oleh siapapun, sehingga mempunyai versi yang cukup banyak.

Sistem operasi Android adalah sistem operasi yang di perkenalkan oleh Google, yaitu perusahaan yang berperan besar pada mesin *search engine* pada internet. Android merupakan sistem operasi berbasis linux untuk perangkat bergerak. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk di gunakan oleh bermacam-macam perangkat bergerak. Android memiliki beberapa versi yang selalu berkembang dengan fitur-fitur baru yang di tambahkan pada telepon selular,

Android di mulai pada Android versi 1.0 (*Apple Pie*), Android versi 1.1 (*Banana Bread*), Android versi 1.5 (*Cupcake*), Android versi 1.6 (*Donut*), Android versi 2.0 (*Eclair*), Android versi 2.2 (*Froyo*), Android 2.3 (*Gingerbread*), Android versi 3.0 (*Honeycomb*), Android versi 4.0 (*Ice Cream Sandwich*), Android versi 4.1 (*Jelly Bean*), Android versi 4.4 (*Kitkat*), Android versi 5.0 (*Lollipop*) dan Android versi 6.0 (*Marshmallow*).



Gambar 1.1 Pengguna *Mobile Operating System* di Dunia (sumbergs.statcounter.com)

Pada saat ini sistem operasi Android adalah sistem operasi yang dapat diandalkan, sistem operasi ini berhasil memasuki ke seluruh segment pasar, pada kalangan bawah, kalangan menengah dan kalangan atas. Dengan dukungan teknologi yang mumpuni membuat Android mempunyai penggemar yang banyak, di bandingkan sistem operasi yang lain.

Dalam sistem operasi Android, Google memberikan aplikasi Google Maps yang merupakan layanan gratis untuk melihat peta digital, di dalamnya di tampilkan peta dan citra bumi dari satelit. Google juga menyediakan API (*Application Program Interface*) untuk para pengembang, agar mereka dapat menghubungkan Google Maps dengan

Aplikasi mereka. seiring dengan berkembangnya Android, berbagai macam aplikasi telah di buat dengan memanfaatkan GPS (*Global Positioning System*) dan Peta Google Maps. Dengan adanya GPS dan Google Maps dapat di buat aplikasi yang dapat mencari rute terdekat dan mendapatkan peta jalan.

Dalam rancangan pembuatan Aplikasi Mobile untuk mengetahui Lokasi Tempat Ibadah Umat Muslim se-Kecamatan Lowokwaru ini nantinya akan di tampilkan peta Kecamatan Lowokwaru, fitur utama dari aplikasi ini adalah pengoperasiannya dapat mencari lokasi tempat ibadah terdekat dari posisi penggunanya saat itu dan juga dapat memberikan panduan jalan yang di tempuh menggunakan metode dijkstra. Pembuatan aplikasi ini di khususkan agar para pengguna tidak kesulitan dalam mencari jalur untuk menuju lokasi tempat ibadah tersebut.

Dalam Al-Qur'an Surat Yunus Ayat 5, Allah menjelaskan tanda-tanda kebesarannya dengan di ciptakannya matahari dan bulan serta peredarannya pada garis lintang dan bujur bumi, sehingga dengannya dapat di tentukan waktu siang dan malam dan juga dapat di tentukan perhitungan Tahun.

هُوَ ٱلَّذِى جَعَلَ ٱلشَّمۡسَ ضِيَآءً وَٱلۡقَمَرَ نُورٗا وَقَدَّرَهُۥ مَنَازِلَ لِتَعۡلَمُواْ عَدَدَ ٱلسِّنِينَ وَٱلۡحِسَابَۚ مَا خَلَقَ ٱللَّهُ ذَٰلِكَ إِلَّا بِٱلۡحَقِّۚ يُفَصِّلُ ٱلۡأٓيَٰتِ لِقَوۡمٖ يَعۡلَمُونَ ٥

"*Dia-lah yang menjadikan matahari bersinar dan bulan bercahaya dana di tetapkan-Nya manzilah-manzilah (tempat-tempat) bagi perjalanan bulan itu, supaya kamu mengetahui bilangan tahun dan perhitungan (waktu). Allah tidak menciptakan yang sedemikian itu melainkan dengan hak. Dia menjelaskan tanda-tanda (kebesaran-Nya) kepada orang-orang yang mengetahui*" (QS. Yunus, 10:5)

Ayat di atas menjelaskan bahwa semua yang di ciptakan Allah adalah bermanfaat dan mengandung hikmah, diantarannya adalah Matahari dan Bulan. Selain merupakan sumber energi utama di bumi, matahari dan bulan juga di jadikan sebagai penanda dalam mengetahui bilangan tahun dan bulan. Pada zaman Yunani kuno para ilmuwan telah menggunakan matahari untuk menghitung jarak antara dua buah tempat yang berjauhan. Dengan Matahari pula para ilmuwan kuno telah berhasil menghitung diameter bumi, walaupun tingkat akurasi kebenarannya masih rendah.

Oleh karena itu penulis bermaksud membuat aplikasi *mobile* yang dapat membantu mencari lokasi tempat ibadah terdekat, dalam penelitian ini penulis membuat aplikasi *mobile* menggunakan sistem operasi Android, karena sistem ini sangat popular saat ini, dan dengan dukungan pengembang aplikasi yang begitu banyak, diharapkan dapat mendukung dalam pembuatan aplikasi ini.

Penulis memanfaatkan Peta Google *Maps* untuk menampilkan rute yang akan di tempuh pengguna, dan metode yang di gunakan adalah Algoritma *dijkstra* untuk menentukan rute terpendek menuju lokasi tempat ibadah.

Algoritma dapat di definisikan sebagai urutan langkah – langkah logis dan sistematis dalam mencari suatu solusi dari suatu permasalahan yang ada (Wahid, Fathul, 2004). Langkah –langkah dalam memecahkan masalah bisa dilakukan dalam berbagai cara dengan karakteristik yang berbeda-beda dari masing-masing langkah. Tiap-tiap algoritma memiliki cara kerja yang berbeda-beda dalam menentukan solusi paling optimal.

Algoritma *dijkstra* dipilih karena algoritma in efektif untuk menemukan rute terpendek. Karena tiap node yang sudah dilewati akan kembali dihitung ulang sehingga dapat menentukan mana yang terpendek. (Lubis, 2009)

Device Android dipilih karena *device* ini memiliki banyak pengguna dan di dukung oleh Google, jadi penggunaan Google *Maps* akan lebih baik dan tidak menutup kemungkinan jumlah ini akan terus meningkat, mengingat teknologi *smartphone* Android terus meningkat.

Penelitian terkait yang pernah di lakukan oleh Faqih Hamami dari Institut Teknologi Sepuluh Nopember Surabaya pada bulan Desember 2011 dengan judul "Implementasi Sistem Informasi Geografis Cityguide Surabaya Pada *Smartphone* Android Berbasis Global Positioning System (GPS)" merupakan penelitian tentang system informasi pada *mobile phone* berbasis Android, aplikasi ini membantu penggunanya dalam menjelajah tempat-tempat wisata di kota Surabaya, data-data mengenai tempat wisata di Kota Surabaya, data-data mengenai tempat wisata telah di simpan dalam *server* dan *mobile phone* adalah sebagai *client*, pengguna cukup mengakses aplikasi melalui *mobile phone*, fitur-fitur utama yang terdapat di aplikasi tersebut adalah daftar wisata lengkap di kota Surabaya dan pencarian jarak menuju tempat wisata menggunakan layanan GPS.

Penelitian setelahnya yaitu penelitian oleh I wayan Gede Suma Wijaya dan Eko Heri Susanto dari STIKOM PGRI banyuwangi pada bulan Februari 2012 dengan Judul "Penerapan Algortima *Dijkstra* Untuk menemukan Rute Terpendek Daerah Wisata di Kabupaten Banyuwangi Pada *Location Based Service* di *Platform* Android" merupakan penelitian yang memanfaatkan teknik LBS (*Location Based Service*), aplikasi ini memungkinkan pengguna untuk mendeteksi posisi dirinya saat itu, kemudian membandingkan jarak dengan posisi tempat wisata, setelah itu barulah di lakukan perhitungan untuk mencari jarak terdekat ke tempat wisata tersebut.

1.2. Rumusan Masalah

Berdasarkan penjelasan pada latar belakang maka rumusan masalah yang di buat adalah, Bagaimana mengetahui lokasi tempat ibadah umat muslim dan rute terpendek menuju lokasi tempat ibadah tersebut.

1.3 Tujuan Penelitian

Tujuan dari pembuatan aplikasi ini adalah untuk membantu pengguna mengetahui rute terdekat menuju lokasi tempat ibadah menggunakan Algoritma *dijkstra* yang hasilnya dapat di tampilan pada peta Google *Maps*.

1.4 Manfaat Penelitian

Manfaat yang dapat diambil dari aplikasi yang dibangun ini adalah untuk memberikan kemudahan bagi umat muslim untuk mengetahui rute terdekat menuju lokasi tempat ibadah, sehingga memudahkan dalam beribadah.

1.5 Batasan Masalah

Pembatasan masalah yang di maksudkan untuk mebatasi ruang lingkup permasalahan yang akan di bahas, mengingat waktu yang tersedia terbatas, demikian pula biaya dan tenaga, bukan untuk mengurangi sifat ilmiah suatu pembahasan. Batasan masalah penelitian ini adalah sebagai berikut :

a. Bahasa Pemrograman yang di gunakan adalah Java untuk android yang diimplementasikan ke dalam perangkat *mobile* dengan sistem operasi Android,
b. Database menggunakan SQLite,
c. Peta digital menggunakan API dari Google Maps,
d. Jaringan 3G/HSDPA untuk mengakses peta dari Google,

e. Platform Android yang di gunakan adalah Android versi 4.1.1, GPS built-in mobile phone harus dalam posisi menyala untuk mengetahui posisi pengguna,

f. Implementasi node untuk Algortima d*ijkstra* hanya bisa di lakukan pada node yang telah terdifinisi yaitu, Jalan Gajayana, Jalan Sumbersari, Jalan Sigura-Gura dan Jalan Sunan Kalijaga.

1.6 Metode Penelitian

Dalam pembuatan aplikasi untuk mengetahui Lokasi tempat ibadah Umat Muslim terdekat menggunakan Algoritma *dijkstra* ini mempunyai beberapa tahapan metode pengerjaanya, tahapan-tahapannya yaitu di awali dengan melakukan studi kepustakaan kepada sumber-sumber yang berkaitan dengan penelitian, tahap berikutnya yaitu mengumpulkan data Koordinat Tempat Ibadah di Kecamatan Lowokwaru, kemudian melakukan perancangan aplikasi dengan database SQLite dan dan aplikasi berbasis Java Android yang di bangun menggunakan IDE Eclipse dan Android SDK sebagai *development tools*.

1.7 Sistematika Penulisan

Laporan tugas akhir ini dibuat dengan sistem penulisan sebagai berikut :

BAB 1 PENDAHULUAN

Berisi tentang latar belakang pemilihan judul, maksud dan tujuan, batasan masalah dan sistematika penulisan laporan.

BAB II TINJAUAN PUSTAKA

Pada bab ini membahas tentang teori-teori yang menjadi acuan dalam pembuatan analisa dan pemecahan dari permasalahan yang dibahas.

BAB III ANALISA DAN PERANCANGAN

Bab ini menjelaskan mengenai analisis dan perancangan pembuatan aplikasi pencarian rute terdekat.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan kebutuhan peralatan, cara instalasi program, cara pemakaian dan penjelasan proses aplikasi yang terjadi pada system

BAB V PENUTUP

Berisi tentang kesimpulan yang diambil dari pembahasan program aplikasi pencarian rute terdekat dan saran untuk pengembangannya.

BAB II

TINJAUAN PUSTAKA

2.1 GPS (*Global Positioning System*)

Adalah sistem untuk menentukan letak permukaan bumi dengan bantuan penyelarasan sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini di terima oleh alat penerima di permukaan dan di gunakan untuk menentukan letak, kecepatan, arah dan waktu, Sistem yang serupa dengan GPS antara lain GLONASS Rusia, Galileo Uni Eropa dan IRNSS india.

2.2 Android

Adalah kumpulan perangkat lunak yang di tujukan bagi perangkat bergerak mencakup sistem operasi, *midddleware* dan aplikasi kunci. *Android Standart Development Kit* (SDK) menyediakan perlengkapan dan *Application Programming Interface* (API) yang di perlukan untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java.

Android di kembangkan oleh Google bersama Open Handset Allience (OHA) yaitu aliansi perangkat selular terbuka yang terdiri dari 47 perusahaan *hardware*, *software* dan perusahaan telekomunikasi di tujukan untuk mengembangkan standar terbuka bagi perangkat selular.

2.2.1 Arsitektur Android

Secara garis besar arsitektur Android dapat di jelaskan dan digambarkan sebagai berikut :

2.2.1.1 *Application Layer*

Application Layer adalah *layer* dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi termasuk klien email, program sms, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi di tulis menggunakan bahasa pemrograman java.

2.2.1.2 *Application Framework*

Android adalah "*Open Development Platform*" yaitu android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur alarm, menambahkan status notifikasi dan sebagainya. Pengembang memiliki akses penuh menuju API framework seperti yang di lakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi di rancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah di gunakan (*reuse*).

2.2.1.3 *Linux Kernel*

Linux Kernel adalah *layer* dimana inti dari operasi sistem di Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers* dan

sistem-sistem operasi android lainnya. Linux kernel yang di gunakan Android adalah linux kernel release 2.6.

2.2.1.4 *Libraries*

Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya berjalan diatas kernel, layer ini meliputi berbagai library C/C++ inti seperti Libe dan SSl, serta :

- *Media Library* untuk pemutaran media audio dan video,
- *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi,
- *Graphic Library* termasuk di dalamnya terdapat SGL dan OpenGL untuk tampilan 2D dan 3D,
- *Libraries LiveWebcore* mencakup *web browser* dengan *engine embedee web view*,

2.2.1.5 *Android Run Time*

Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan implementasi linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam *Android Run Time* dibagi menjadi dua bagian, yaitu :

- *Core Libraries* : Aplikasi Android di bangun dalam bahasa java, sementara Dalvik sebagai *virtual* mesinnya bukan *virtual* machine Java, sehingga di perlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java yang di tangani oleh *Core Libraries*.
- *Dalvik Virtual Machine* : *Dalvik* merupakan sebuah mesin *virtual* yang di kembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah

perkampungan yang berada di Iceland. *Dalvik* hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex)

2.2.2 Aplikasi Fundamental

Aplikasi Android ditulis dengan bahasa pemrograman Java. Kode Java dikompilasi bersama dengan *file* data dan sumber daya yang dibutuhkan oleh aplikasi, dipaketkan dengan alat apt ke dalam Android, *file* arsip ditandai oleh akhiran *apk. *File* ini adalah bagian dari mendistribusikan aplikasi dan menginstall pada perangkat *mobile*. Semua kode di sebuah *file* *apk tunggal, dianggap sebagai satu aplikasi. Dalam hal ini, masing-masing aplikasi Android hidup di dunia sendiri meliputi :

1. Secara *default*, semua aplikasi berjalan dalam proses Linux masing-masing. Android memulai proses ketika salah satu kode aplikasi harus di jalankan, dan menutup proses ketika itu tidak lagi diperlukan dan sumber daya sistem yang dibutuhkan oleh aplikasi lain.
2. Setiap proses memiliki virtual machine (VM) sendiri, maka kode aplikasi berjalan secara terpisah dari kode dari semua aplikasi lainnya.
3. Secara *default*, setiap aplikasi diberikan sebuah ID pengguna Linux yang unik. Perizinan ditetapkan sehingga *file* aplikasi terlihat hanya untuk pengguna yang dan hanya untuk aplikasi itu sendiri, meskipun ada cara untuk ekspor ke aplikasi lain juga.

Hal tersebut memungkinkan untuk mengatur dua aplikasi untuk berbagi ID pengguna yang sama, dalam hal ini akan dapat melihat *file* masing-masing. Untuk menghemat sumber daya sistem, aplikasi dengan ID yang sama juga dapat mengatur untuk menjalankan Linux dalam proses yang sama, berbagi *Virtual Machine* yang sama.

2.2.3 Komponen Aplikasi Android

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mangembangkan fitur *scroller* yang lebih baik dan memungkinkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan dan pemanggilan objek java untuk bagian itu. Oleh karena itu Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.2.3.1 *Activities*

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *activity* di deklarasikan dalam sebuah kelas yang bertugas untuk menampilkan *user interface* yang terdiri dari *Views* dan respon terhadap *Event*.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

a. *Active*, setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus dan menerima masukkan dari pengguna

b. *Paused*, dalam beberapa kasus *activity* akan terlihat rapi tidak terfokus, pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna.

c. *Stopped*, ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped*. *Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk di eksekusi oleh *sistem* ketika membutuhkan sumberdaya lebih.

d. *Inactive*, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive Acitvity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

2.2.3.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *activity* tidak aktif.

2.2.3.3 *Intents*

Intents merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intents* tidak selalu

dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misalkan jika terdapat pesan masuk.

2.2.3.4 *Broadcast Receivers*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *broadcast* berasal dari sistem, misalnya baterai yang sudah hamper habis, informasi zona waktu telah berubah atau pengguna telah merubah bahasa *default* pada perangkat.

2.2.3.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam *file system,* dalam *database* SQLite, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data.

2.2.4 Tipe Aplikasi Android

Terdapat 3 kategori aplikasi pada Android :

2.2.4.1 *Foreground Activity*

Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini pasti mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2.2.4.2 *Background Service*

Aplikasi yang memiliki interaksi terbatas dengan user, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tampak pada layar.

2.2.4.3 *Intermittent Actvity*

Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan mamberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik, untuk aplikasi yang kompleks akan sulit untuk menentukan kategori aplikasi tersebut , apalagi aplikasi memiliki ciri-ciri dari semua kategori. Oleh karena itu perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.2.5 Kelebihan Android

a) Keterbukaan, Pengambangan dapat dilakukan dengan bebas tanpa dikenakan biaya terhadap sistem, karena Android berbasiskan Linux dan *Open Source*.
b) Arsitektur komponen dasar Android terinspirasi dari teknologi internet *Mashup*. Bagian dalam Aplikasi dapat digunakan oleh aplikasi yang lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
c) *Support* yang banyak, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, database SQL dan peta.
d) Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga sistem bekerja lebih stabil.
e) Dukungan Grafis, dengan adanya dukungan Grafis 2D dan animasi yang digabung menjadi 3D menggunakan OpenGL memungkinkan membuat berbagai aplikasi dan game yang berbeda.

2.2.6 Google *Maps*

Adalah layanan aplikasi peta online yang di sediakan oleh Google secara gratis. Layanan peta Google *Maps* secara resmi dapat di akses melalui situs

www.maps.google.com. Pada situs tersebut dapat dilihat informasi geografis pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan. Layanan ini di buat dangat interaktif, karena di dalamnya peta dapat di geser sesuai keinginan pengguna, mengubah level *zoom* , serta mengubah tampilan jenis peta.

Google Maps API mamungkinkan pengembang untuk mengintegrasikan Google Maps ke dalam Aplikasi yang dibuatnya. Dengan menggunakan Google Maps API memungkinkan untuk menanamkan situs Google Maps ke situs eksternal, dan si situs tersebut dapat dilakukan *Overlay*.

Meskipun pada awalnya hanya JavaScript API, Google Maps API diperluas untuk menyertakan API untuk Adobe Flash, layanan untuk mengambil gambar peta statis dan layanan web untuk melakukan geocoding, menghasilkan petunjuk rute.

Kelas Kunci dalam perpustakaan Maps adalah *MapView*, sebuah *subclass* dari *view group* dalam *standar library* Android. Sebuah *MapView* menampilkan peta dengan data yang diperoleh dari layanan Google Maps. Bila MapView memiliki fokus, dapat menangkap tombol yang ditekan dan gerakan sentuh untuk pan dan zoom peta secara otomatis, termasuk penanganan permintaan jaringan untuk peta tambahan. Ini juga menyediakan semua elemen UI yang diperlukan bagi pengguna untuk mengendalikan peta. Aplikasi tersebut juga dapat menggunakan *method* kelas *MapView* untuk mengontrol *MapView* secara terprogram dan menarik sejumlah tampilan di atas peta.

2.3 Optimasi

2.3.1 Definisi Optimasi dan Nilai Optimasi

Optimasi ialah suatu proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dicapai). Dalam disiplin matematika, optimisasi merujuk pada studi permasalahan yang mencoba untuk mencari nilai minimal atau maksimal dari suatu fungsi nyata. Untuk mencapai nilai minimum dan maksimum tersebut, secara sistematis dilakukan pemilihan integer atau bilangan nyata yang akan memberikan solusi optimal.

Nilai optimal adalah nilai yang didapat dengan melalui suatu proses dan dianggap sebagai suatu solusi jawaban yang paling baik dari semua solusi yang ada.

2.3.2 Penyelesaian Masalah Optimasi

Secara umum, penyelesaian masalah pencarian jalur terpendek dapat dilakukan dengan menggunakan dua metode, yaitu metode konvensional dan heuristik. Metode konvensional diterapkan dengan perhitungan matematis biasa, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan.

2.3.2.1 Metode Konvensional

Metode Konvensional adalah metode yang menggunakan perhitungan matematis biasa. Ada beberapa metode konvensional yang biasa digunakan untuk melakukan pencarian jalur terpendek, diantaranya, Algoritma *Dijkstra*, Algoritma *Floyd-Marshall* dan Algoritma *Bellman-Ford.*

2.3.2.1 Metode Heuristik

Adalah sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi. Ada beberapa Algoritma pada Metode Heuristik yang umum dilakukan dalam permasalahan optimasi, diantaranya Algoritma Genetika, Algoritma Semut, Logika *Fuzzy* dan Jaringan Syaraf Tiruan.

2.3.3 Permasalahan Jalur Terpendek

Awal mula dari permasalahan ini adalah untuk menemukan jalur terpendek dari *vertex* S ke *vertex* T pada jaringan yang berbobot. Untuk melakukannya, kita bergerak melewati jaringan tersebut dari kiri ke kanan, menghitung jarak terpendek dari S ke setiap *node* yang mengarah pada tujuan kita. Di setiap langkah algoritma, kita melihat semua simpul ke tujuan dari *vertex* sekarang ke *vertex* yang memiliki nilai yang terdekat dari S dengan garis. Setiap *vertex* membutuhkan label permanen (potensinya dan disimbolkan di sekeliling tabel), yang menunjukkan jarak terdekat dari S ke *vertex* tersebut. Sekali T memiliki nilai, kemudian kita menemukan jarak terdekat dari S ke T. (Jungnikel, 2005)

2.4 Algoritma *Djikstra*

Algoritma *Dijkstra* ditemukan oleh Edsger W. Dijkstra yang merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi dan bersifat sederhana. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek (sebuah lintasan yang mempunyai panjang minimum) dari *vertex* a ke *vertex* z dalam *graph* berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh *node* negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah infiniti.

Secara umum, sebelum melakukan iterasi, algoritma sudah mengidentifikasikan jarak terdekat dari i-1 *vertex* terdekatnya. Selama seluruh *edge* berbobot tertentu yang (positif), maka *vertex* terdekat berikutnya dari node asal dapat ditemukan selama *vertex* berdekatan dengan *vertex* Ti. Kumpulan *vertex* yang berdekatan dengan *vertex* di Ti dapat

dikatakan sebagai "*fringe vertices*". *Vertex* inilah yang merupakan kandidat dari Algoritma *Dijkstra* untuk memilih *vertex* berikutnya datri node asal.

Algoritma *Dijkstra* mempunyai sifat sederhana (*straightforward*). Sesuai dengan arti *greedy* yang secara harfiah berarti tamak atau rakus, namun tidak dalam konteks negatif, Algortima *Greedy* ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi kedepan.

2.4.1 Elemen-Elemen Penyusun Greedy dalam Algoritma Dijkstra

2.4.1.1 Himpunan Kandidat

Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam *graf*, himpunan kandidat ini adalah himpunan simpul pada *graf* tersebut.

2.4.1.2 Himpunan Solusi

Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah bagian dari himpunan kandidat.

2.4.1.3 Fungsi Seleksi

Fungsi Seleksi adalah fungsi yang akan memilih setiap kandidat yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

2.4.1.4 Fungsi Kelayakan

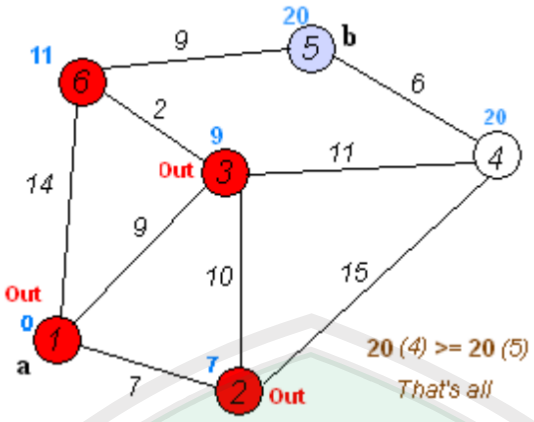
Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar *constraint* atau tidak. Apabila kandidat melanggar *constraint* maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

2.4.1.5 Fungsi Objektif

Fungsi Objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

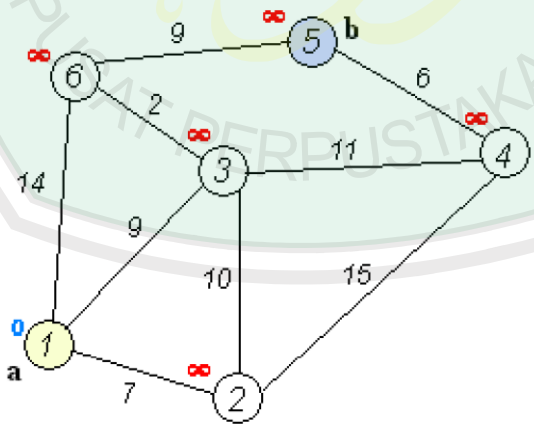
Jika menggunakan Algoritma *Dijkstra* untuk menentukan jalur terpendek dari suatu *graph* maka pasti akan menemukan jalur yang terbaik. Karena pada waktu penentuan jalur yang akan terpilih, akan dianalisis bobot dari *node* yang belum terpilih. Lalu dipilih *node* dengan bobot yang terkecil, jika ternyata ada bobot yang lebih kecil jika melalui node tertentu maka bobot dapat berubah. Algoritma ini akan berhenti jika semua node sudah terpilih dan dengan Algoritma *Dijkstra* ini dapat menemukan jarak terpendek dari node yang dipilih.

Persoalan mencari lintasan terpendek di dalam *graf* merupakan salah satu persoalan optimasi. *Graf* yang di gunakan dalam mencari lintasan terpendek dalam graph berbobot. Bobot pada sisi *graph* dapat menyatakan jarak antar kota, waktu, biaya dan sebagainya. Dalam hal ini bobot harus bernilai positif seperti gambar berikut :



Gambar 2.1 Algoritma Dijkstra

Pemilihan *node*-nya sendiri berdasarkan pada bobot yang terendah dari tabel, dan *predecessor* merupakan kode yang bertetangga dengan node, dengan bobot yang terendah. Program juga akan menampilkan jalur yang terpilih untuk jalur yang terpendek dan juga besarnya jarak . Dan node awal (*start*) dan akhir (*finish*) akan berwarna kuning. Program tidak akan melayani jika *node* awal dan akhir sama, karena jarak yang di hasilkan akan nol, dan jika kondisi ini ternyata terpenuhi maka akan ada report yang menyatakan bahwa *node* awal dan akhir tidak boleh sama.



Gambar 2.2 Hubungan antar Titik dalam Algoritma *Dijkstra*

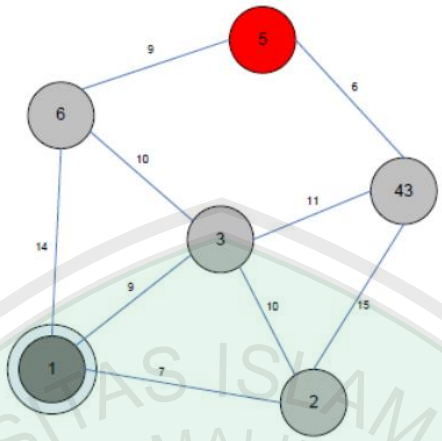
Pertama – tama tentukan titik mana yang akan menjadi *node* awal, lalu beri bobot jarak pada node pertama ke node terdekat satu persatu seperti pada Gambar 2.2, *Dijkstra* akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Inilah urutan logika dari Algoritma *Dijkstra* :

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada *node* awal dan nilai tak hingga terhadap *node* lain (belum terisi),
2. Set semua *node* "belum terpilih" dan set *node* awal sebagai "*node* keberangkatan",
3. Dari *node* keberangkatan, pertimbangkan *node* tetangga yang belum terpilih dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke node C berjarak 2, maka jarak ke C melewat B menjadi 6+2 =8. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru,
4. Saat kita selesai mempertimbangkan setiap jarak terhadap *node* tetangga, tandai *node* yang telah terpilih sebagai "*Node* terpilih". *Node* terpilih tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya,
5. Set "*Node* belum terpilih" dengan jarak terkecil (dari *node* keberangkatan) sebagai "*Node* keberangkatan" selanjutnya dan lanjutkan dengan kembali ke step 3,

2.4.2 Langkah-Langkah Dalam Algoritma *Dijkstra*

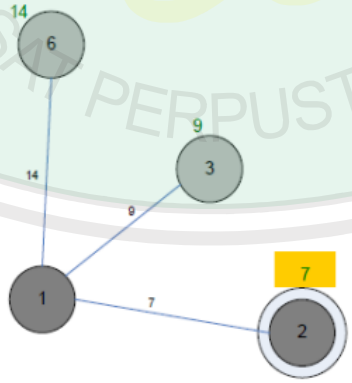
Dibawah ini penjelasan langkah per langkah pencarian jalur terpendek secara rinci dimulai dari node awal sampai node tujuan dengan nilai jarak terkecil.

1. Pada Gambar 2.3, *node* awal 1, *node* tujuan 5. Setiap *edge* yang terhubung antar node telah diberi nilai



Gambar 2.3 Contoh kasus *Dijkstra* – Langkah 1

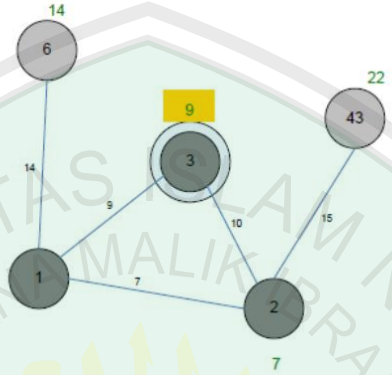
2. *Dijkstra* melakukan kalkulasi terhadap node tetangga yang terhubung langsung dengan node keberangkatan (*node* 1), terlihat pada Gambar 2.4, hasil yang didapat adalah *node* 2 karena bobot nilai *node* 2 paling kecil dibandingkan nilai pada node lain, nilai = 7 (0+7).



Gambar 2.4 Contoh kasus *Dijkstra* – Langkah 2

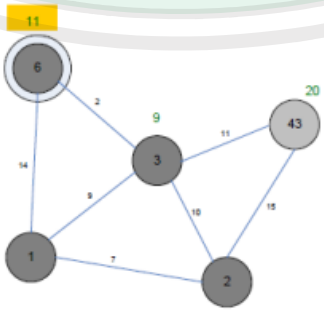
3. *Node* 2 diset menjadi *node* keberangkatan dan ditandai sebagai *node* yang terpilih. *Dijkstra* melakukan kalkulasi kembali terhadap *node-node* tetangga

yang terhubung langsung dengan *node* yang terpilih. Kemudian pada Gambar 2.5, kalkulasi *Dijkstra* menunjukkan bahwa *node* 3 yang menjadi yang menjadi *node* keberangkatan selanjutnya karena bobotnya yang paling kecil dari hasil kalkulasi terakhir, nilai 9 (0+9).



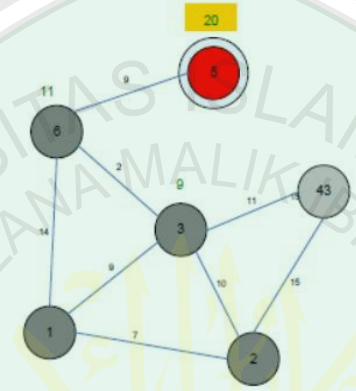
Gambar 2.5 Contoh kasus *Dijkstra* – Langkah 3

4. Perhitungan berlanjut pada gambar 2.6, dengan *node* 3 ditandai menjadi *node* yang telah terpilih. Dari semua *node* tetangga belum terlewati yang terhubung langsung dengan *node* terlewati, *node* selanjutnya yang ditandai menjadi node terpilih adalah *node* 6 karena nilai bobot yang terkecil, nilai 11 (9+2).



Gambar 2.6 Contoh kasus *Dijkstra* – Langkah 4

5. Node 6 menjadi node terpilih, *Dijkstra* melakukan kalkulasi kembali, dan menemukan bahwa *node* 5 (node tujuan) telah tercapai lewat *node* 6. Pada Gambar 2.7, alur terpendeknya adalah 1-3-5-6, dan nilai bobot yang didapat adalah 20 (11+9). Bila *node* tujuan telah tercapai maka kalkulasi *dijkstra* dinyatakan selesai.



Gambar 2.7 Contoh kasus *Dijkstra* – Langkah 5

2.5 Teori *Graf*

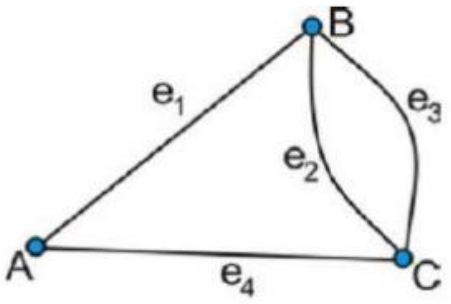
2.5.1 Definisi *Graf*

Graf adalah kumpulan simpul (*node*) yang dihubungkan satu sama lain melalui sisi/busur (*edges*) (Zakaria, 2006). Suatu Graf G terdiri dari dua himpunan yaitu himpunan V dan himpunan E, seperti pada Gambar 2.10.

a. *Vertex* (simpul) - V = Himpunan simpul yang terbatas dan tidak kosong,
b. *Edge* (sisi/busur) – E = Himpunan busur yang menghubungkan sepasang simpul.

Dapat dikatakan *graf* adalah kumpulan dari simpul-simpul yang dihubungkan oleh sisi-sisi.

‘



Gambar 2.8 *Graf*

V = {A,B,C}

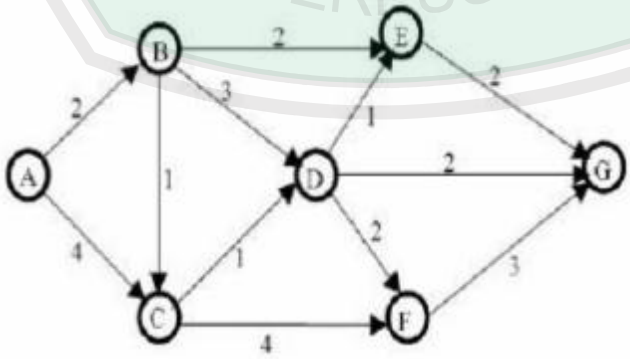
E = {e1, e2, e3, e4}

2.5.2 Macam – Macam *Graf*

Menurut arah dan bobotnya, *graf* dibagi menjadi empat bagian :

2.5.2.1 Graf Berarah Dan Berbobot

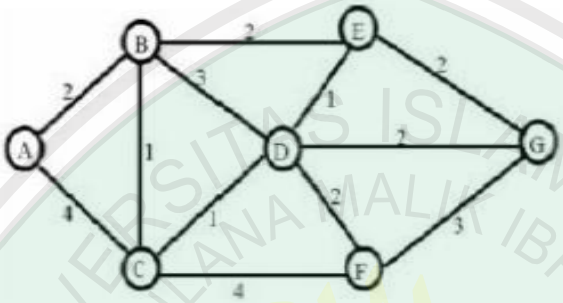
Tiap busur mempunyai anak panah dan bobot. Gambar 2.9 menunjukkan *graf* berarahan berbobot yang terdiri dari tujuh titik yaitu titik A, B, C, D, E, F, G. Titik menunjukkan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan C dan seterusnya. Bobot antar titik A dan titik B pun telah diketahui.



Gambar 2.9 *Graf* Berarah dan Berbobot

2.5.2.2 *Graf* Tidak Berarah Dan Berbobot

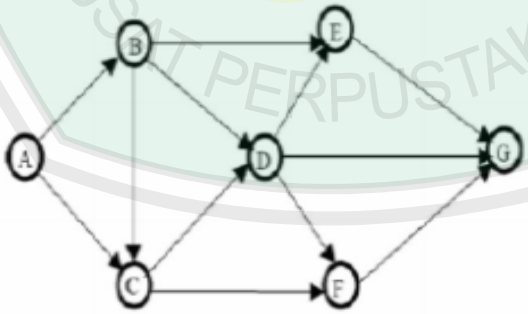
Tiap busur tidak mempunyai anak panah tapi mempunyai bobot. Gambar 2.10 menunjukkan *graf* tidak berarah dan berbobot. *Graf* terdiri tujuh titik yaitu titik A, B, C, D, E, F, G. Pada gambar 2.10, titik A tidak menunjukkan arah ke titik B dan C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



Gambar 2.10 *Graf* Tidak Berarah dan Berbobot

2.5.2.3 *Graf* Berarah Dan Tidak Berbobot

Tiap busur mempunyai anak panah yang tidak berbobot. Gambar 2.11 menunjukkan *graf* berarah dan tidak berbobot.

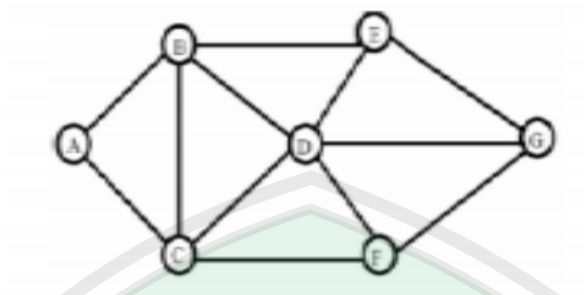


Gambar 2.11 *Graf* Berarah dan Tidak Berbobot

2.5.2.4 *Graf* Tidak Berarah Dan Tidak Berbobot

Pada gambar 2.12, tiap busur tidak mempunyai anak panah dan tidak mempunyai bobot

.



Gambar 2.12 *Graf* Tidak Berarah dan Tidak Berbobot

2.5.3 Terminologi Dasar *Graf*

Dalam teori *Graf* terdapat beberapa terminologi (istilah) yang berkaitan dengan *graf* yang akan di definisikan satu persatu sebagai berikut :

2.5.3.1 Bertetangga (*Adjacent*)

Dua buah simpul pada *graf* dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain, vj bertetangga dengan vk jika (vj, vk) adalah sebuah sisi pada *graf*.

2.5.3.2 Bersisian (*Incident*)

Untuk sembarang sisi e = (vj, vk), sisi e dikatakan bersisian dengan simpul vj dan vk.

2.5.3.3 Simpul Terpencil (*Isolated Vertex*)

Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau dapat dinyatakan bahwa simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya\

2.5.3.4 *Graf* Kosong (*Null Graph* atau *Empty Graph*)

Graf yang himpunan sisinya merupakan himpunan kosong disebut sebagai *graf* kosong dan ditulis sebagai Nn, yang dalam hal ini n adalah jumlah simpul

2.5.3.5 Derajat (*Degree*)

Derajat suatu simpul pada *graf* tak berarah adalah jumlah sisi yang bersisian dengan simpul tersebut. Pada *graf* berarah, derajat simpul v dinyatakan din (v) dan dout (v), yang dalam hal ini din (v) = derajat masuk = derajat busur yang masuk ke simpul. Dout (v) = derajat keluar = derajat busur yang keluar dari simpul. Untuk sembarang *graf* G, banyaknya simpul yang berderajat ganjil selalu genap.

2.5.3.6 Lintasan (*Path*)

Lintasan yang panjangnya n dari simpul awal v0 ke simpul tujuan vn didalam *graf* G adalah barisan berselang seling simpul-simpul dan sisi-sisi yang berbentuk v0, e1, v1, e2, v2, e3, v3…, vn-1, en, vn sedemikian sehingga e1 = (v0, v1), e2 = (v1-1, vn) adalah sisi-sisi dari *graf* G.

2.5.3.7 Siklus (*Cycle*) Atau Sirkuit (*Circuit*)

Lintasan yang berawal dan berakhir di simpul yang sama disebut sirkuit atau siklus.

2.5.3.8 Terhubung (*Connected*)

Graf tak berarah disebut *graf* terhubung (*connected graph*) jika untuk setiap pasang simpul vi dan vj di dalam himpunan V terhadap lintasan dari vi ke vj (yang berarti ada lintasan dari vi ke vj). Jika tidak maka disebut *graf* tak terhubung. *Graf* berarah dikatakan terhubung jika *graf* tak berarahnya terhubung (*Graf* tak berarahnya dari *graf* diperoleh dengan menghilangkan arahnya). *Graf* berarah disebut *graf* terhubung terkuat (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang vi dan vj di G terhubung kuat. Kalau tidak, G disebut *graf* terhubung lemah.

2.5.3.9 *Upagraf* (*Subgraph*) Dan Komplemen *Upagraf*

Misalkan G = (V,E) adalah sebuah *graf* G1 = (V1, E1) adalah *upagraf* (*subgraph*) dari G jika V1 $\subset$ V dan E1 $\subset$ E, complement dari *upagraf* G1 terhadap *graf* G adalah graf G2 = (V2, E2) sedemikian sehingga E2 = E – E1 dan V2 adalah himpunan simpul anggota – anggota E2 bersisian dengannya.

2.5.3.10 *Upagraf* Merentang (*Spanning Supgraph*)

Upagraf G1 = (V1, E1) dari G = (V, E) dikatakan *upagraf* merentang jika V1 = V (yaitu G1 mengandung semua simpul dari G).

2.5.3.11 *Cut-Set*

Cut set dari *graf* terhubung G adalah himpunan sisi yang bisa dibuang dari G menyebabkan G tidak terhubung. Jadi, *cut-set* selalu menghasilkan dua buah komponen terhubung.

2.5.3.12 *Graf* Berbobot.

Graf berbobot adalah *graf* yang setiap sisinya diberi sebuah harga (bobot), (Munir, 2010)

BAB III

ANALISA DAN PERANCANGAN

3.1 Analisa Kebutuhan

Komponen yang dibutuhkan dalam penelitian ini terbagi menjadi dua macam, yaitu *Software* dan *Hardware*.

3.1.1 Software

Software yang dibutuhkan untuk pembuatan aplikasi *mobile phone* untuk mengetahui rute terdekat ke tempat ibadah menggunakan Algoritma *Dijkstra* adalah :

a. Sistem Operasi Windows 7

   Windows 7 digunakan karena sistem operasi ini paling *compatible* dengan berbagai macam *software*, dan mempunyai dukungan yang banyak.

b. Java Development Kit (JDK) versi 1.8

   JDK adalah paket *platform* java yang terdiri dari berbagai macam *library*, JVM, *compiler* dan *debugger*.

c. Java Runtime Environment (JRE) versi 1.8

   Agar program java dapat dijalankan, maka file berekstensi *.java harus dikompilasi menjadi file *bytecode*. JRE berfungsi untuk mengeksekusi file *bytecode* yang memungkinkan pemakai untuk menjalankan program java di berbagai *platform*.

d. Android SDK (Software Development Kit)

Android SDK adalah tools API (Application Programming Insterface) yang diperlukan untuk mengembangkan aplikasi pada platform Android menggunakan baha pemrograman java. Pada pembuatan aplikasi ini, menggunakan Android SDK versi 21.1.

e. Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat di jalankan di semua *platform* (*platform-independent*).

f. Database SQLite

Adalah database yang berukuran kecil, berdiri sendiri, bukan database *client server*, tanpa konfigurasi yang rumit namum mempunyai dapat menjalankan banyak fitur-fitur perintah SQL.

3.1.2 Hardware

Hardware yang dibutuhkan untuk pembuatan aplikasi mobile phone untuk mengetahui rute terdekat ke tempat ibadah menggunakan Algoritma *Dijkstra* adalah :

a. Notebook

Notebook yang di gunakan menggunakan spesifikasi berikut :

1. Intel(R) Core(TM) i3 M350 @2.27 GHz

2. RAM 2048 MB
3. *Hardisk* 500 GB

b. Smartphone

Selain menggunakan emulator android yang terdapat pada Eclipse, peneliti juga menguji menggunakan smartphone android yaitu Asus Zenfone 2 dengan spesifikasi sebagai berikut :

1. Intel Atom Z3580 2.3 GHz
2. RAM 4096 MB
3. Memori Internal 32 GB
4. Ukuran layar 5,5” 1080x1920 *pixels*

3.2 Spesifikasi Aplikasi

a. Menyediakan layanan bagi pengguna untuk mengetahui lokasi tempat ibadah di sekitar lokasi mereka,

b. Menyediakan rute terpendek untuk menuju lokasi tempat ibadah mereka.

3.3 Spesifikasi Pengguna

Aplikasi mobile ini bebas di gunakan oleh siapapun, dengan hanya menginstall aplikasi ini pada smartphone android, akan dapat menampilkan rute terdekat untuk menuju lokasi tempat ibadah.

3.4 Desain Sistem

Analisa Aplikasi Pencarian Lokasi Tempat Ibadah Terdekat menggunakan Algoritma Dijkstra ini adalah Aplikasi yang terdapat pada smartphone pengguna sudah memiliki database lokasi-lokasi tempat ibadah, pengguna akan mendapatkan pilihan lokasi masjid dan rute akan di tampilkan setelah aplikasi mendapatkan koordinat lokasi titik keberangkatan pengguna.

Gambar 3.1 Desain Sistem

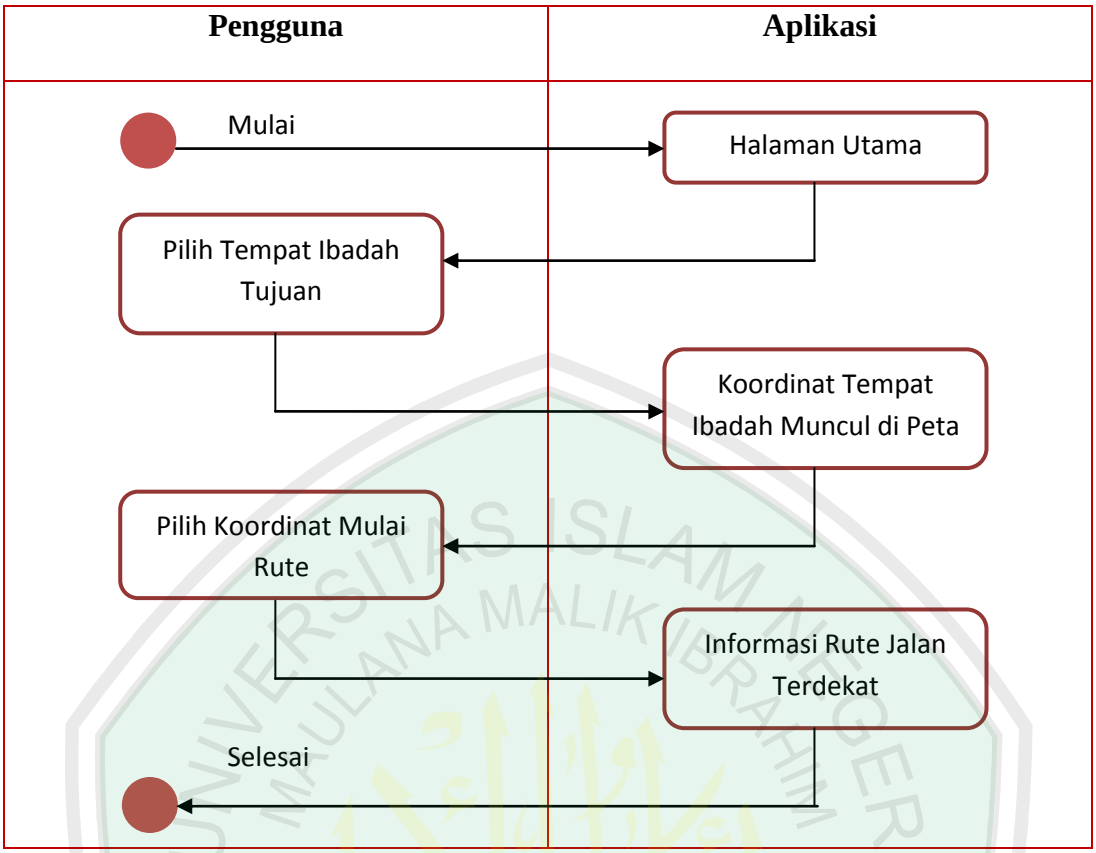
3.4.1 Fungsi Sistem

Fungsi-fungsi yang akan di terapkan pada aplikasi pencarian lokasi tempat ibadah terdekat ini adalah :

a. Melacak Koordinat Pengguna menggunakan GPS pada smartphone,
b. Pengguna dapat mengetahui lokasi tempat ibadah di sekitarnya dalam bentuk peta,
c. Pengguna dapat menuju lokasi tempat ibadah yang di tuju menggunakan rute yang di tampilkan pada peta.

3.4.2 Analisa *Activity* Diagram

Berikut *Activity* Diagram yang terjadi pada seluruh proses aplikasi pencarian lokasi tempat ibadah terdekat.



Gambar 3.2 *Activity* Diagram

3.4.3 Struktur Database

Terdapat tiga tabel yang di buat di dalam aplikasi ini, yaitu :

3.4.3.1 Tabel Graph

Tabel ini berisikan panjang simpul yang berisikan simpul-simpul yang menyimpan panjang jalan setiap simpul yang di akan lalui.

Dengan struktur tabel seperti berikut :

Tabel 3.1. Tabel *Graph*

Kolom	Tipe Data
Id	Integer
Simpul_awal	Integer
Simpul_tujuan	Integer

Jalur	Text
Bobot	Double

Keterangan untuk tabel diatas adalah :

1. Id : Memberikan id ke setiap simpul,
2. Simpul Awal : Berisikan Simpul awal deklarasi rute,
3. Simpul Tujuan : Berisikan Simpul Tujuan deklarasi rute,
4. Jalur :Menyimpan koordinat-koordinat pada simpul yang akan dilalui,
5. Bobot : Nilai simpul dalam satuan meter.

3.4.3.2 Tabel Jalur Mobil

Tabel ini berisikan jalur pada peta yang dapat dilalui oleh kendaraan mobil. Di dalamnya setiap jalurnya menghubungkan antar simpul

Dengan struktur tabel seperti berikut :

Tabel 3.2. Tabel Jalur_Mobil

Kolom	Tipe Data
Id	Integer
Jalur	Varchar
Simpul	Varchar

Keterangan untuk tabel diatas adalah :

1. Id : Memberikan id ke setiap jalur,
2. Jalur : Nama Jalur,
3. Simpul : Berisi simpul yang menghubungkan node

3.4.3.3 Tabel Masjid

Tabel ini berisikan Koordinat Lokasi Masjid

Dengan struktur tabel seperti berikut :

Tabel 3.3. Tabel Masjid

Kolom	Tipe Data
Id	Integer
Masjid	Varchar
Koordinat	Text

Keterangan untuk tabel diatas adalah :

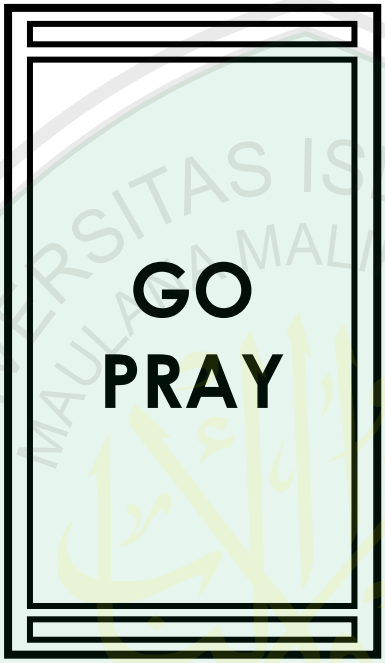
1. Id : Memberikan id ke setiap jalur,
2. Masjid : Nama Masjid,
3. Koordinat : Berisi Koordinat Lokasi Masjid

3.4.4 Desain Interface

Perlu di perhatikan dalam perancangan desain interface aplikasi yaitu interface dibuat dengan semudah mungkin agar pengguna tidak kesulitan saat menggunakannya. Sehingga perlu diperhatikan komponen-komponen

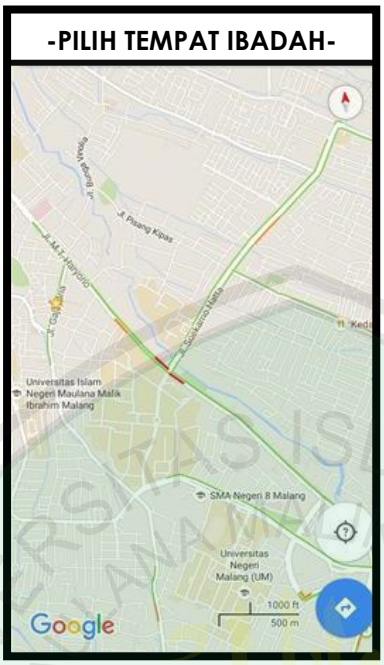
dari tampilan seperti letak menu, tampilan peta dan komponen visual lainnya. Berikut ini adalah perancangan menu utama aplikasi untuk mencari lokasi tempat ibadah terdekat.

3.4.4.1 Halaman Aplikasi



Gambar 3.3 Halaman Splash Screen

Gambar 3.4 Halaman Menu Utama

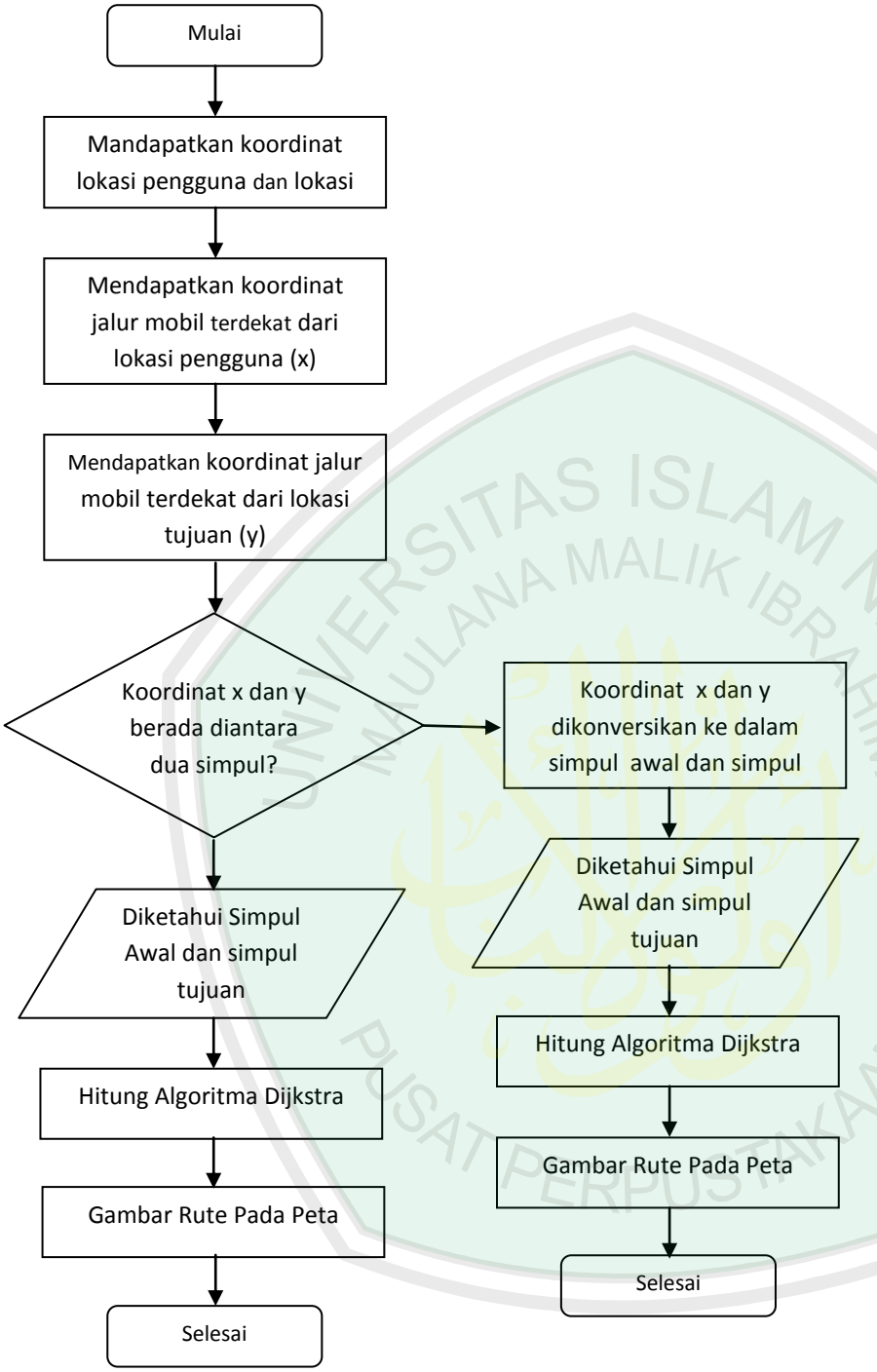


Gambar 3.5 Halaman Peta Pada Menu "*Find Location*"

Gambar 3.6 Halaman Pilihan Nama Masjid

Gambar 3.7 Halaman Rute Menuju Masjid

Gambar 3.8 Halaman Menu *About*



Gambar 3.9 *Flowchart* Perhitungan Algoritma *Dijkstra*

Pada gambar 3.9 adalah *flowchart* yang menjelaskan proses perhitungan dari Algortima *Dijkstra*, langkah pertama adalah mendapatkan koordinat dari lokasi pengguna dan lokasi tujuan, kemudian mencari jalur mobil terdekat dengan lokasi pengguna dan lokasi tujuan, pada tabel jalur mobil telah di definisikan pada kolom simpul, kemudian apakah lokasi pengguna (x) dan lokasi tujuan (y) berada di antara dua simpul, jika iya maka perlu di buat simpul baru lagi yang menjadikan lokasi pengguna (x) manjadi lokasi awal dan lokasi tempat ibadah (y) menjadi lokasi tujuan, setelah di ketahui simpul baru yang berisi lokasi awal dan lokasi tujuan yang baru maka dilakukanlah perhitungan Algortima *Dijkstra*, setelah mendapatkan hasilnya maka di gambarlah rute pada peta menggunakan *polyline*.

BAB IV
HASIL DAN PEMBAHASAN

Di dalam bab ini penulis akan menjelaskan hasil uji coba terhadap aplikasi pencarian lokasi tempat ibadah terdekat menggunakan algoritma *dijkstra* yang telah dibuat, uji coba ini bertujuan untuk mengetahui apakah aplikasi yang telah di buat telah sesuai dan berjalan dengan semestinya sesuai dengan yang telah penulis buat pada Bab III. Pada bab ini juga akan di bahas mengenai fitur aplikasi dan *interface* yang telah terdapat di dalam aplikasi pencarian lokasi tempat ibadah terdekat menggunakan algoritma *dijkstra*.

4.1 Implementasi Sistem

Dalam tahap ini aplikasi yang telah di desain mulai diterapkan dengan membangun komponen-komponen yang telah di jelaskan pada bab sebelumnya.

4.1.1 Implementasi Aplikasi Pengguna

Aplikasi pencarian lokasi tempat ibadah terdekat menggunakan algoritma *dijkstra* ini menggunakan database SQLite sebagai basisdata dan menggunakan java android untuk pemrograman aplikasinya,

4.1.2 Ruang Lingkup Perangkat Keras

lokasi tempat ibadah terdekat menggunakan algoritma *dijkstra* ini adalah :

4. Intel(R) Core(TM) i3 M350 @2.27 GHz
5. RAM 2048 MB
6. *Harddisk* 500 GB
7. *Mouse*
8. *Smartphone* Android

4.1.3 Ruang Lingkup Perangkat Lunak

1. Windows 7 *Ultimate*
2. JDK (*Java Development Kit*) versi 1.8
3. JRE (*Java Runtime Environment*) versi 1.8
4. Eclipse
5. Android SDK

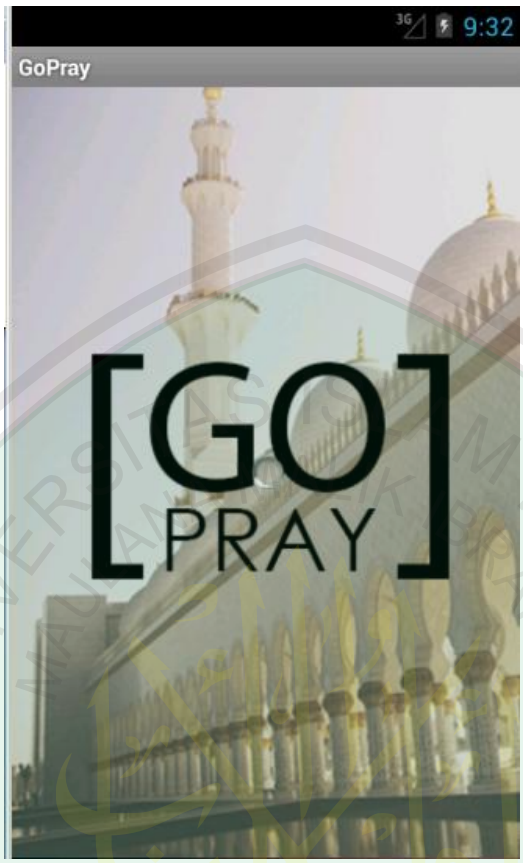
4.2 Implementasi *Interface*

Dalam aplikasi pencarian lokasi tempat ibadah terdekat menggunakan algoritma *dijkstra* ini mengimplementasikan beberapa *interface*, diantaranya :

1. Halaman aplikasi dengan menampilkan *splash screen*,
2. Halaman aplikasi dengan menampilkan menu utama,
3. Halaman aplikasi dengan manampilkan nama-nama tempat ibadah,
4. Halaman aplikasi dengan menampilkan peta lokasi tujuan tempat ibadah,
5. Halaman aplikasi dengan menampilkan rute algortima *dijkstra* menuju lokasi tempat ibadah.

4.2.1 Halaman Splash Screen

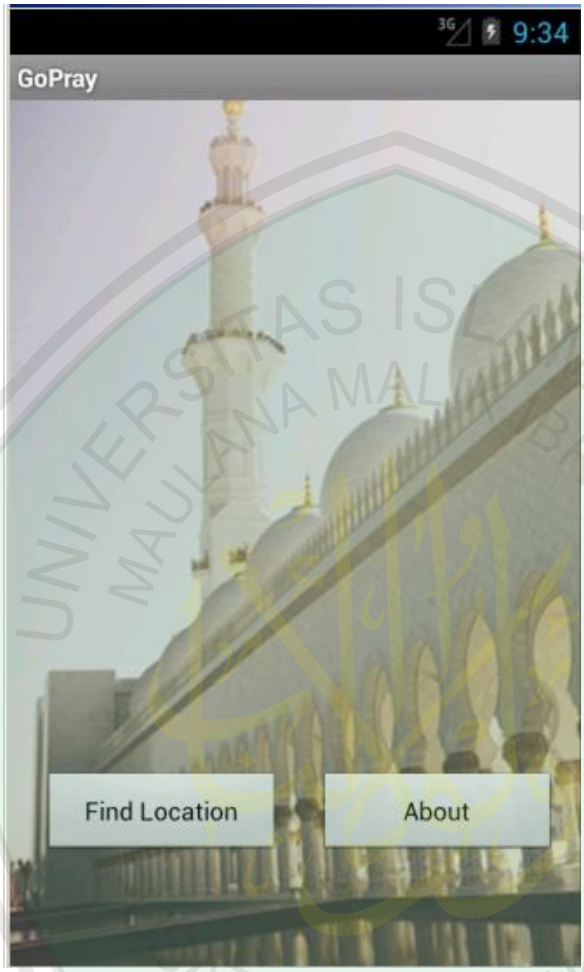
Merupakana halaman yang ditampilkan saat pertama kali aplikasi di buka,



Gambar 4.1 *Interface* Halaman *Splash Screen*

4.2.2 Halaman Menu Utama

Merupakan halaman yang menampilkan menu utama dari aplikasi, pada tampilan ini dua pilihan menu yaitu “Menu *Find Location*” dan “Menu *About*”



Gambar 4.2 *Interface* Halaman Menu Utama

4.2.3 Halaman Pilihan Nama-Nama Tempat Ibadah

Merupakan halaman yang menampilkan nama-nama tempat ibadah yang ada di database. Dengan menampilkan menu *drop-down*.

Gambar 4.3 Halaman Menu Pilihan Nama Tempat Ibadah

4.2.4 Halam Peta Menampilkan Lokasi Tujuan Tempat Ibadah

Merupakan halaman yang menampilkan lokasi masjid yang sudah dipilih dan lokasi titik berangkat pengguna.

Gambar 4.4 Halaman Peta Dengan Lokasi Masjid dan Pengguna

4.2.5 Halaman Peta Menampilkan Rute Algoritma *Dijkstra*

Merupakan halaman yang menampilkan lokasi masjid yang sudah dipilih dan lokasi titik berangkat pengguna dengan sudah menampilkan rute *dijkstra* yang di tandai dengan warna biru,

Gambar 4.5 Halaman Peta dengan Rute Algoritma *Dijkstra*

```
graph = arg_graph;
int simpul_awal = simpulAwal;
int simpul_maju = simpulAwal;
int simpul_tujuan = simpulTujuan;

if(simpul_maju != simpul_tujuan){
//HITUNG JUMLAH SIMPUL
int jml_simpul = 0;
for(String[] array : graph){
if(array[0] != null){
jml_simpul += 1;
}
}
//System.out.println("Jumlah Simpul : "+jml_simpul);

//System.out.println("==================================");
//System.out.println();

//TANDAI SIMPUL YANG AKAN DIKERJAKAN
List<Integer> simpulYangDikerjakan = new
ArrayList<Integer>();

//UNTUK MENYIMPAN NILAI-NILAI * YANG DITANDAI
List<Integer> simpulYangSudahDikerjakan_bawah = new
ArrayList<Integer>();

double nilaiSimpulYgDitandai = 0;
double nilaiSimpulFixYgDitandai = 0;

//PERULANGAN HANDLE
for(int perulangan = 0; perulangan < 1; perulangan++){

//UNTUK MNDAPATKAN 1 BOBOT PALING MINIMUM DARI SETIAP SIMPUL
List<Double> perbandinganSemuaBobot = new
ArrayList<Double>();

//DAFTARKAN SIMPUL pertama YANG AKAN DIKERJAKAN KE DALAM
ARRAY

if(!simpulYangDikerjakan.contains(simpul_maju)){
                    simpulYangDikerjakan.add(simpul_maju);
                }

//PERULANGAN SIMPUL SIMPUL YANG DITANDAI
for(int perulanganSimpul = 0; perulanganSimpul <
simpulYangDikerjakan.size(); perulanganSimpul++){
```

```
//HITUNG JUMLAH BARIS PER KOLOM SIMPUL
int jml_baris_fix = 0;
for(int min_batas_bris = 0; min_batas_bris < 100;
min_batas_bris++){

if(graph[simpulYangDikerjakan.get(perulanganSimpul)][min_bata
s_bris] != null){
jml_baris_fix += 1;
}
}

//CARI BOBOT MINIMUM di 1 simpul berdasarkan baris scr
urut[0][0],[0][1] dst
List<Double> bobot = new ArrayList<Double>();
int status_baris = 0;

//perulangan CARI BOBOT2 DI 1 SIMPUL
for(int min_batas_bris_fix = 0; min_batas_bris_fix <
jml_baris_fix; min_batas_bris_fix++){
String bobot_dan_ruas =
graph[simpulYangDikerjakan.get(perulanganSimpul)][min_batas_b
ris_fix];//pasti berurutan [0][0],[0][1] dst
//print isi dr baris[0][0],[0][1],[0][2] dst

//System.out.println("bobot_dan_ruas : "+bobot_dan_ruas);
String[] explode;
explode = bobot_dan_ruas.split("->");
//System.out.println("bobot_ : "+explode[0]);
//cari bobot yg belum dikerjakan (yg tidak ada tanda ->y)
if(explode.length == 2){
status_baris += 1; // masih ada yg belum ->y

//Cek simpul apakah sudah ditandai apa blom, klo udh berarti
nilai * tidak ditambah lagi / 0
//kalo blm ditandai, berarti nilai * bernilai
nilaiSimpulYgditandai

if(!simpulYangSudahDikerjakan_bawah.isEmpty()){

if(simpulYangSudahDikerjakan_bawah.contains(simpulYangDikerja
kan.get(perulanganSimpul))){
nilaiSimpulYgDitandai = 0;
}else{
nilaiSimpulYgDitandai = nilaiSimpulFixYgDitandai;
}}
```

```
bobot.add((Double.parseDouble(explode[1])+nilaiSimpulYgDitan
dai));//bs acak bobot[0],bobot[2]// 0 dan 2 berdasarkan
baris yg akan dikerjakan

graph[simpulYangDikerjakan.get(perulanganSimpul)][min_batas_
bris_fix] =
String.valueOf(explode[0]+"-
>"+(Double.parseDouble(explode[1])+nilaiSimpulYgDitandai));
}}

//jika baris di kolom belum ->y semua, maka lakukan if di
bawah ini :
if(status_baris > 0){

//DAPATKAN BOBOT MINIMUM
for(int index_bobot = 0; index_bobot < bobot.size();
index_bobot++){
if(bobot.get(index_bobot) <= bobot.get(0)){
bobot.set(0, bobot.get(index_bobot));
}}

perbandinganSemuaBobot.add(bobot.get(0));
}//end if jika ->y atau ->t belum semua dikerjakan
else{//Jika baris di kolom sudah ->y semua, maka lakukan
else di bawah ini
//System.out.println("=======||Baris sudah ->y
semua||=======");}

//DAFTARKAN SIMPUL SIMPUL YANG baru selesai DIKERJAKAN

if(!simpulYangSudahDikerjakan_bawah.contains(simpulYangDiker
jakan.get(perulanganSimpul))){

simpulYangSudahDikerjakan_bawah.add(simpulYangDikerjakan.get
(perulanganSimpul));}
}//end for perulanganSimpul

//DAPATKAN 1 BOBOT PALING MINIMUM DARI SIMPUL YG DITANDAI
for(int min_indexAntarBobotYgDitandai = 0;
min_indexAntarBobotYgDitandai <
perbandinganSemuaBobot.size();
min_indexAntarBobotYgDitandai++){

if(perbandinganSemuaBobot.get(min_indexAntarBobotYgDitandai)
<= perbandinganSemuaBobot.get(0)){
                        perbandinganSemuaBobot.set(0,
perbandinganSemuaBobot.get(min_indexAntarBobotYgDitandai));
}}
```

```
//DAPATKAN INDEX SIMPUL+BOBOTNYA YG ASLI DARI SIMPUL YG
DITANDAI

int indexAwalAsli = 0; //index simpulnya
int status_baris1 = 0;
int dapat_indexAsliBobot = 0;
int simpul_lama = 0;
for(Integer indexAsli_bobot : simpulYangDikerjakan){
for(int baris1 = 0; baris1 < 100; baris1++){

if(graph[simpulYangDikerjakan.get(indexAwalAsli)][baris1]
!= null){
String bobot_dan_ruas1 =
graph[simpulYangDikerjakan.get(indexAwalAsli)][baris1];

//System.out.println(bobot_dan_ruas1);
String[] explode1;
explode1 = bobot_dan_ruas1.split("->");
if(explode1.length == 2){
// System.out.println("----------;"+explode1[1]);

if(perbandinganSemuaBobot.get(0) ==
Double.parseDouble(explode1[1])){
dapat_indexAsliBobot = baris1;
simpul_lama = simpulYangDikerjakan.get(indexAwalAsli);
simpul_maju = Integer.parseInt(explode1[0]);
status_baris1 += 1;
}
}//end if cek ->y atau ->t
}//end if cek baris != null
}//end for limit baris = 100
indexAwalAsli++; //index simpul di tambah 1
}//end for simpul yang dikerjakan
//BULETIN BOBOT MINIMUM YANG UDH DIDAPAT  dan HAPUS RUAS
YANG BERHUBUNGAN
if(status_baris1 > 0){
graph[simpul_lama][dapat_indexAsliBobot] =
graph[simpul_lama][dapat_indexAsliBobot]+"->y";
```

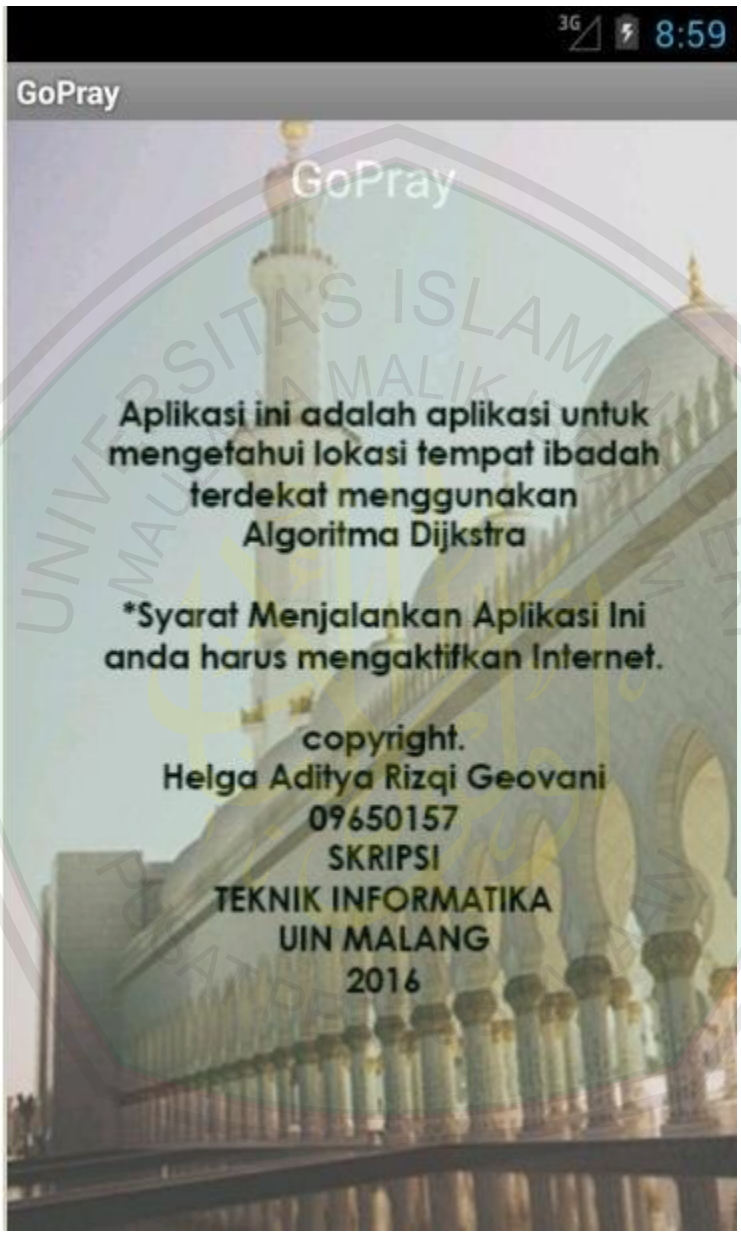
```
//HAPUS RUAS LAIN
for(int min_kolom = 0; min_kolom < jml_simpul;
min_kolom++){
for(int min_baris = 0; min_baris < 100; min_baris++){

if(graph[min_kolom][min_baris] != null){
String ruasYgAkanDihapus = graph[min_kolom][min_baris];
String[] explode3 = ruasYgAkanDihapus.split("->");
if(explode3.length == 2){
if(explode3[0].equals(String.valueOf(simpul_maju))){
graph[min_kolom][min_baris] =
graph[min_kolom][min_baris]+"->t";
}
}//end if cek ->y atau ->t
}//end if cek baris != null
}//end for baris
}//end for kolom
}//end if cek status_baris sudah ->y atau ->t semua apa
belum
//Nilai * yg ditandai
nilaiSimpulFixYgDitandai = perbandinganSemuaBobot.get(0);
//System.out.println("nilaiSimpulFixYgDitandai :
"+nilaiSimpulFixYgDitandai);
//System.out.println("perbandingan simpul? :
"+simpul_maju+" = "+simpul_tujuan+" ..?");
if(simpul_maju != simpul_tujuan){
--perulangan;
}
else{
break; //akhiri perulangan
}
}//end for handle perulangan
//System.out.println("--SELESAI--");
//taruh simpul gabungan ke array; misal : simpul 6-10
List<String> gabungSimpulPilihan = new ArrayList<String>();
            for(int h = 0; h < jml_simpul; h++){
                for(int n = 0; n < 100; n++){
                    if(graph[h][n] != null){
                        String str_graph = graph[h][n];
```

```
if(str_graph.substring(str_graph.length()-1,
str_graph.length()).equals("y")){
String[] explode4 = graph[h][n].split("->");
String simpulGabung = h+"-"+explode4[0];

gabungSimpulPilihan.add(simpulGabung);
}
}//end if cek isi graph != null
}//end for looping baris
}//end looping kolom (simpul)

//masukkan simpul yg sudah diurutkan (dari simpul tujuan ke
simpul awal). (nanti direverse arraynya)
List<Integer> simpulFix_finish = new ArrayList<Integer>();
//masukkan pertama kali simpul tujuan (simpul akhir) ke
array dgn index 0. (nanti dibalik(reverse) arraynya)
simpulFix_finish.add(simpul_tujuan);

int simpul_explode = simpul_tujuan;

for(int v = 0; v < 1; v++){
for(int w = 0; w < gabungSimpulPilihan.size(); w++){
String explode_simpul = gabungSimpulPilihan.get(w);
String[] explode5 = explode_simpul.split("-");

if(simpul_explode == Integer.parseInt(explode5[1])){
simpulFix_finish.add(Integer.parseInt(explode5[0]));
simpul_explode = Integer.parseInt(explode5[0]);}
if(simpul_explode == simpul_awal){
break;
}}

if(simpul_awal != simpul_explode){
                    --v;
                }else{
                    break;
                }
            }//end for cari simpul yang dibuletin lalu
dibandingkan dgn simpul_tujuan
```

```
//array di balik indexnya; jadi simpul tujuan di pindah
posisi ke akhir index array
Collections.reverse(simpulFix_finish);
String jalur_terpendek = "";
for(int x = 0; x < simpulFix_finish.size(); x++){
if(x == simpulFix_finish.size()-1){
jalur_terpendek += simpulFix_finish.get(x);
}else{
jalur_terpendek += simpulFix_finish.get(x)+"->";
}
}

//System.out.println("... "+jalur_terpendek);
//Toast.makeText(getBaseContext(), "... "+jalur_terpendek,
Toast.LENGTH_LONG).show();
jalur_terpendek1 = jalur_terpendek;
}//end if start != finish
}
}
```

4.2.6 Halaman About

Halaman ini berisi informasi mengenai tata cara penggunaan aplikasi dan identitas pembuat aplikasi.



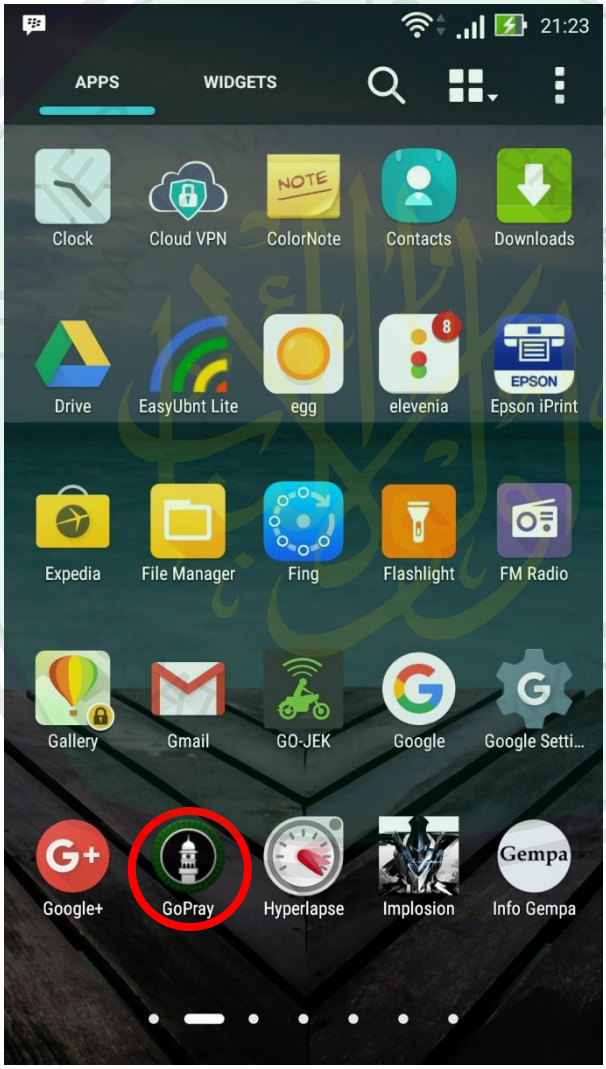
Gambar 4.6 Halaman Tentang Aplikasi

4.3 Uji Coba Aplikasi

4.3.1 Uji Coba Alur Aplikasi

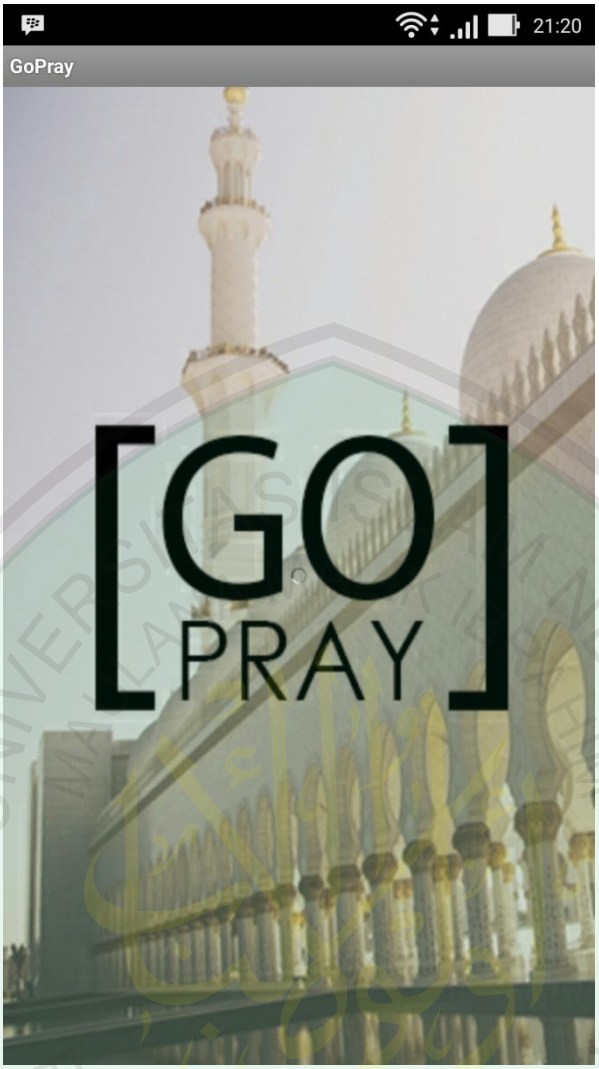
Pada saat menggunakan aplikasi ini pastikan koneksi internet di *smartphone* aktif dan mempunyai kualitas jaringan yang baik.

1. Pilih aplikasi GoPray pada menu (bertanda lingkaran merah) :



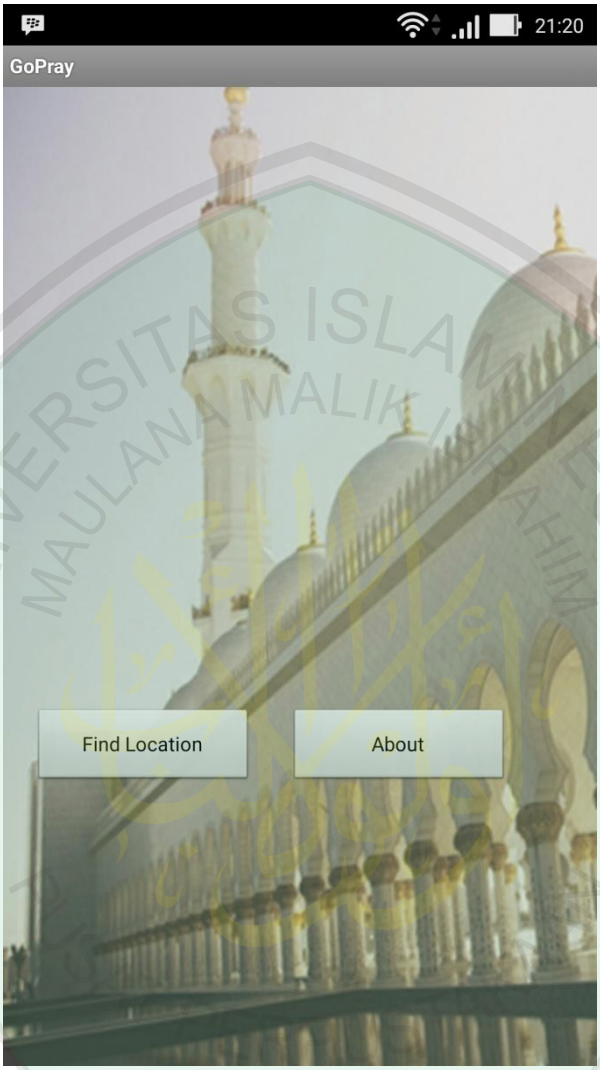
Gambar 4.7 Pengujian Tampilan Pada Menu *App Drawer*

2. Setelah icon di pilih akan terbuka aplikasi GoPray dan akan muncul halaman *splash screen* seperti berikut :



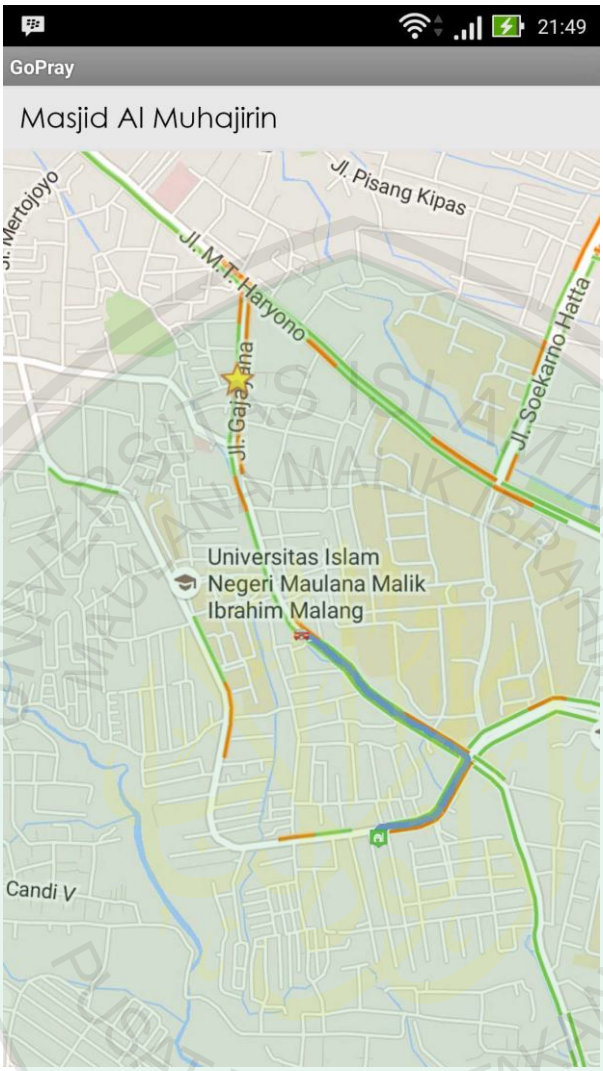
Gambar 4.8 Pengujian Tampilan *Splash Screen*

3. Kemudian di tampilkan menu utama dari aplikasi :



Gambar 4.9 Pengujian Tampilan Menu Utama

4. Pengujian menggunakan rute, dalam menu ini di pilih menuju Masjid Al Muhajirin, dan di tampilkan rute seperti pada Gambar 4.9 berikut ini :



Gambar 4.10 Pengujian Menampilkan Rute

5. Pengujian menampilkan halaman info aplikasi , halaman ini muncul jika pada menu utama di sentuh menu "*about*",

Gambar 4.11 Pengujian Menampilkan Halaman *About*

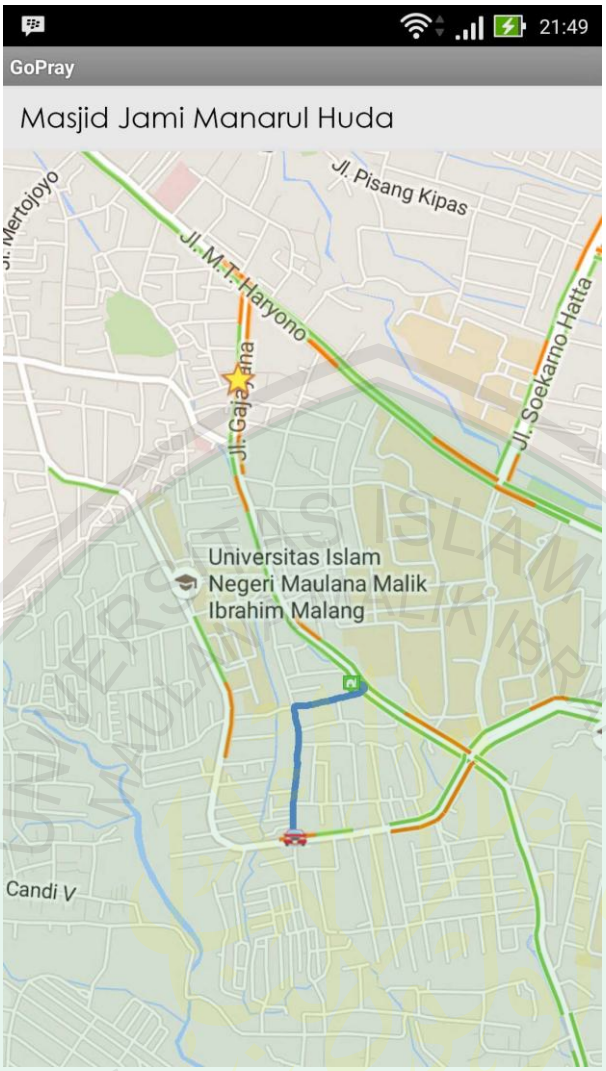
4.3.2 Uji Coba Penggunaan Algoritma *Dijkstra* Pada Peta

Penulis melakukan pengujian terhadap perhitungan Algoritma *Dijkstra* pada peta, apakah Algoritma *Dijkstra* dapat menyelesaikan dan menampilkannya dalam peta.

Pada pengujian ini penulis melakukan 3 uji tes dengan lokasi yang berbeda-beda :

Tabel 4.1. Tabel Uji Coba 1

PENGUJIAN 1	
Koordinat Awal : -7.9576, 112.6076	
Koordinat Tujuan : Masjid Manarul Huda : -7.9542, 112.6105	
Relasi Vertex 1 : 5-4-3	988 meter
Relasi Vertex 2 : 5-1-3	655 meter
Relasi Vertex 3 : 5-2-1-3	1.024 meter
Relasi Vertex 4 : 5-2-0-3	1.757 meter
Relasi Vertex Terpilih : 5-1-3 = 655 meter	

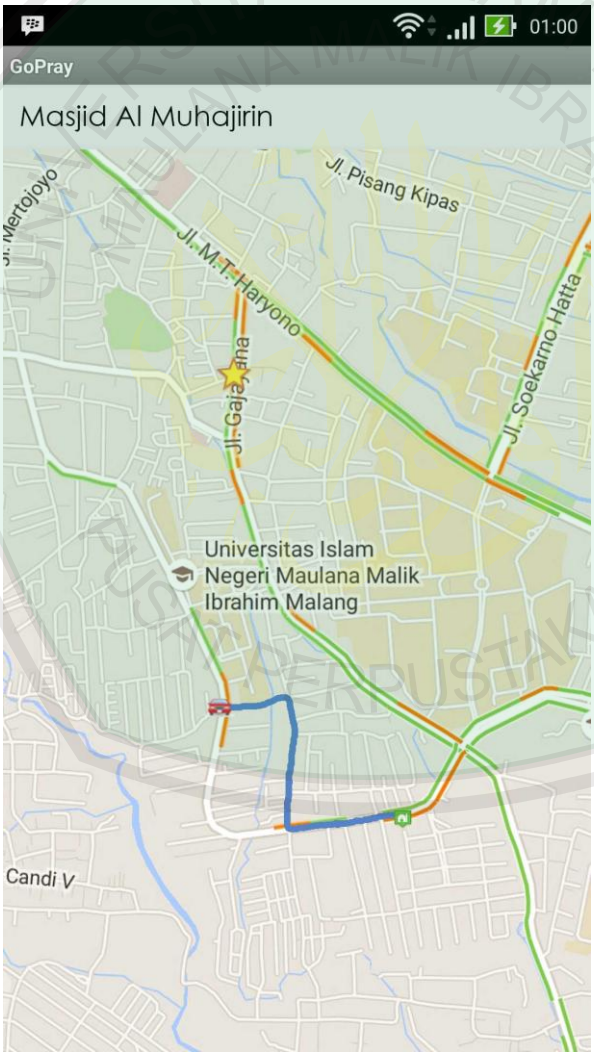


Gambar 4.12 Pengujian 1

Pengujian 1, pada menu *dropdown* nama-nama Masjid di pilih Masjid Jami Manarul Huda yang berada di jalan sumbersari. Lalu pada titik keberangkatan atau lokasi pengguna berada pada jalan bendungan sigura-sigura barat, pada rute menuju Masjid Jami Manarul Huda bila penulis lihat hanya pada tampilan peta terdapat dua jalur yang mengarah ke lokasi, bila kita lihat di tabel, jalur pertama adalah relasi *vertex* 1 yang mempunyai panjang 988 meter, dan jalur kedua adalah relasi *vertex* 2 yang mempunyai panjang 655 meter, Algoritma *Dijkstra* menyelesaikan pada relasi *vertex* 2 dengan *vertex* 5-1-3.

Tabel 4.2. Tabel Uji Coba 2

PENGUJIAN 2	
Koordinat Awal : -7.9540, 112.6068	
Koordinat Tujuan : Masjid Muhajirin : -7.9580, 112.6107	
Relasi Vertex 1 : 2-1-5-4	937 meter
Relasi Vertex 2 : 2-5-4	942 meter
Relasi Vertex 3 : 2-1-3-4	1.071 meter
Relasi Vertex 4 : 2-0-3-4	1.804 meter
Relasi Vertex Terpilih : 2-1-5-4 = 937 meter	

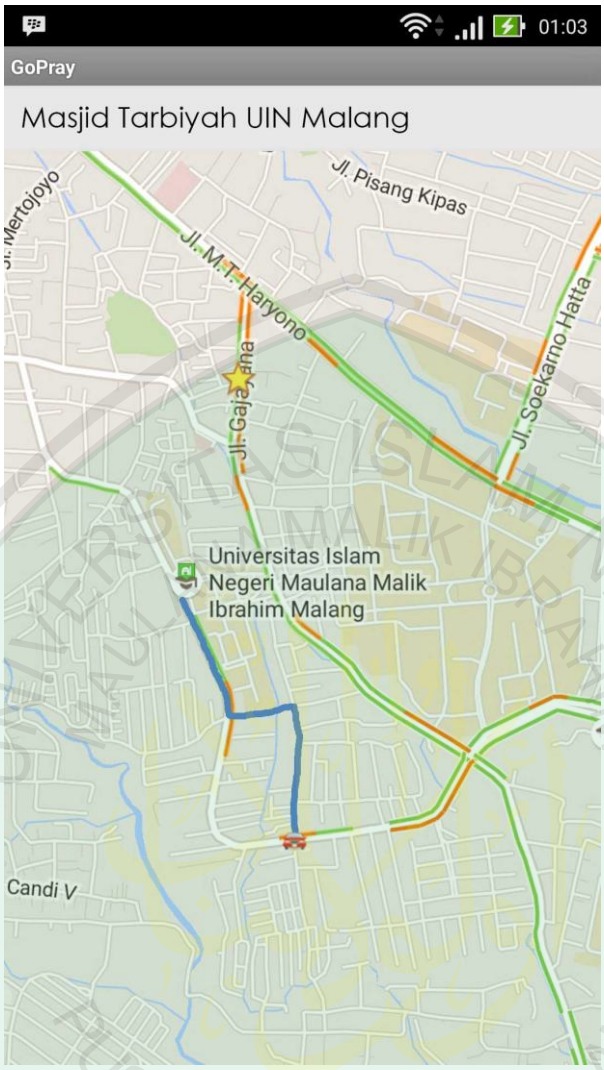


Gambar 4.13 Pengujian 2

Pengujian 2, pada menu *dropdown* nama-nama Masjid di pilih Masjid Al Muhajirin yang berada di jalan bendungan sigura-gura barat. Lalu pada titik keberangkatan atau lokasi pengguna berada pada jalan sunan kalijaga, pada rute menuju Masjid Al Muhajirin bila penulis lihat hanya pada tampilan peta terdapat dua jalur yang mengarah ke lokasi, bila kita lihat di tabel, jalur pertama adalah relasi *vertex* 1 yang mempunyai panjang 937 meter, dan jalur kedua adalah relasi *vertex* 2 yang mempunyai panjang 942 meter, Algoritma *Dijkstra* menyelesaikan pada relasi *vertex* 1 dengan vertex 2-1-5-4.

Tabel 4.3. Tabel Uji Coba 3

PENGUJIAN 3	
Koordinat Awal : -7.9576, 112.6076	
Koordinat Tujuan : Masjid Tarbiyah UIN Malang : -7.9503, 112.6069	
Relasi Vertex 1 : 5-2-0	1.084 meter
Relasi Vertex 2 : 5-1-3-0	1.298 meter
Relasi Vertex 3 : 5-4-3-0	1.661 meter
Relasi Vertex 4 : 5-1-2-0	1.080 meter
Relasi Vertex Terpilih : 5-1-2-0 = 1.080 meter	



Gambar 4.14 Pengujian 3

Pengujian 3, pada menu *dropdown* nama-nama Masjid di pilih Masjid tarbiyah UIN Malang yang berada di jalan Sunan Kalijaga. Lalu pada titik keberangkatan atau lokasi pengguna berada pada jalan bendungan sigura-gura barat, pada rute menuju Masjid Tarbiyah UIN Malang bila penulis lihat hanya pada tampilan peta terdapat dua jalur yang mengarah ke lokasi, bila kita lihat di tabel, jalur pertama adalah relasi *vertex* 1 yang mempunyai panjang 1.084 meter, dan jalur kedua adalah relasi *vertex* 4 yang mempunyai panjang 1.080 meter, Algoritma *Dijkstra* menyelesaikan pada relasi *vertex* 4 dengan vertex 5-1-2-0.

4.4 Aplikasi Pencarian Lokasi Tempat Ibadah Dalam Pandangan Islam

Aplikasi ini dibuat oleh penulis yang berguna untuk membantu umat islaam yang ingin segera melaksanakan ibadah di masjid, aplikasi ini dapat menggantikan dari cara pada umumnya seperti bertanya pada seseorang di pinggir jalan menegenai lokasi masjid terdekat. Beberapa manfaat dari aplikasi ini adalah:

a. Aplikasi ini sangatlah mobilitas, dengan *smartphone* anda dapat menggunakannya di manapun anda berada,
b. *Smartphone* saat ini sudah banyak yang mempunyai harga murah, bila di bandingkan dengan alat GPS, tentunya jauh lebih mahal, dengan aplikasi ini akan menjadi lebih efisien biaya,
c. Ukuran *smartphone* yang kecil, sehingga akan praktis di bawa kemanapun dan di akses kapanpun.

Firman Allah SWT dalam Surat Al Baqarah : 153

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ ٱسْتَعِينُواْ بِٱلصَّبْرِ وَٱلصَّلَوٰةِۚ إِنَّ ٱللَّهَ مَعَ ٱلصَّٰبِرِينَ

"*Hai Orang-Orang yang beriman, jadikanlah sabar dan shalat sebagai penolongmu. Sesungguhnya Allah beserta orang-orang yang sabar*".

Dan Juga Hadits Rasulullah SAW :

Abdullah Ibnu Mas'ud RA berkata : "Aku bertanya kepada Rasulullah, "Ya Rasulullah, amal perbuatan apa yang paling afdhal?, beliau menjawab, "Shalat tepat pada waktunya", Aku bertanya lagi,"Lalu apa lagi?,"Beliau menjawab,"Berbakti kepada kedua orang tua". Aku bertanya lagi, "Kemudian apa lagi, ya Rasullullah?" Beliau menjawab, "Berjihad di jalan Allah" (H.R Bukhari).

Mendirikan shalat sudah menjadi rutinitas kita sebagai muslim, karena memang itu salah satu hal yang wajib dari perintah wajib lainnya yang harus ditunaikan. Begitu pentingnya shalat ini sehingga tidak ada ruang untuk kita melalaikannya (terutama bagi laki-laki yang sudah baligh), tidak mampu berdiri, kita bisa dengan duduk, tidak bisa duduk dengan berbaring, dan sebagainya sampai kita bisa melakukannya. Atau ketika tidak ada air kita bisa bertayamum, ketika dalam perjalanan kita bisa mengatur waktu shalat kita dengan menjamak atau mengqashar shalat kita. Inilah yang membedakan shalat dengan ibadah lain. Oleh karena itu, hendaklah kita sebagai seorang muslim terus meningkatkan kualitas ibadah shalat yang kita lakukan setiap harinya dengan baik dan benar dengan salah satunya selalu mengerjakan shalat tepat pada waktunya.

BAB V
PENUTUP

5.1 Kesimpulan

Dari keseluruhan bab di atas, dalam aplikasi ini, dapat diambil kesimpulan bahwa, Aplikasi implementasi algoritma *dijkstra* untuk mengetahui lokasi tempat ibadah umat muslim pada *mobile phone* ini di implementasikan yang gunanya adalah untuk membantu umat muslim untuk melakukan aktivitas ibadahnya dengan baik.

Kelemahan yang terdapat pada aplikasi ini adalah algoritma d*ijkstra* hanya akan menghitung pada area yang telah terdefinisi, sehingga bila terdapat area baru yang belum terdifinisi seperti pada *node* dan *graf*-nya, maka algortima tidak dapat berjalan sebagaimana mestinya.

5.2 Saran

1. Dengan hasil di atas perlu diadakan penelitian yang lebih lanjut untuk memaksimalkan fungsi aplikasi dan penambahan fitur agar dapat lebih berfungsi dengan baik.
2. Untuk metode yang digunakan perlu pengkajian yang lebih sesuai dan efektif untuk mendapatkan rute yang lebih optimal.

DAFTAR PUSTAKA

Faqih Hamami, (2011), *Implementasi Sistem Informasi Geografis City Guide Surabaya Pada Smartphone Android Berbasis Global Positioning System (GPS)*, *Skripsi*, Teknik Informatika, Institut Teknologi Sepuluh Nopember, Surabaya.

Bramantya, Amirullah Andi, (2012), *Aplikasi Location Based Service Untuk Menentukan Rute Terpendek Lokasi ATM di Kota Malang Menggunakan Algoritma Dijkstra*, *Skripsi*, Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

I Wayan Gede Suma Wijaya, Eko Heri Susanto, (2012), *Penerapan Algoritma Dijkstra Untuk Menemukan Rute Terpendek Daerah Wisata di Kabupaten Banyuwangi Pada Location Based Service di Platform Android*, Skripsi, Teknik Informatika, Stikom Banyuwangi.

Lubis, H.S, (2009), *Perbandingan Algoritma Greedy dan Dijkstra untuk menentukan lintasan terpendek* , Skripsi, Matematika dan Ilmu Pengetahuan Alam, Universitas Sumatra Utara Medan.

Junaedi, Moh. (2008). *Pengantar XML*, Jakarta:Ilmukomputer.com

Safaat, Nzruddin, 2011, *Pengembangan Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Bandung:Informatika.

http://gs.statcounter.com/#mobile_os-ww-monthly-201505-201605-bar

diakses pada 7 Juni 2016.