RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA*

SKRIPSI

Oleh:

Faisal Lutfi Afriansyah

NIM. 09650147

JURUSAN TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI (UIN) MAULANA MALIK IBRAHIM MALANG

2013

RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA*

SKRIPSI

Diajukan Kepada:

Fakultas Sains dan Teknologi

Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang

Untuk Memenuhi Salah Satu Persyaratan Dalam

Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

FAISAL LUTFI AFRIANSYAH

NIM. 09650147

JURUSAN TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI (UIN) MAULANA MALIK IBRAHIM MALANG

2013

RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA*

Oleh:

FAISAL LUTFI AFRIANSYAH

NIM. 09650147

Telah Diperiksa dan Disetujui untuk Diuji :

Tanggal, 4 April 2013

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

Mengetahui,

Ketua JurusanTeknik Informatika

Ririen Kusumawati, M.Kom
NIP. 19720309 200501 2 002

RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA*

SKRIPSI

Oleh:

Faisal Lutfi Afriansyah
NIM. 09650147

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

Tanggal, 11 April 2013

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	: Fachrul Kurniawan, M.MT NIP. 19771020 200901 1 001	( )
2. Ketua	: Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
3. Sekretaris	: A'la Syauqi, M.Kom NIP. 19771201 200801 1 007	( )
4. Anggota	: Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )

Mengetahui,

Ketua JurusanTeknik Informatika

Ririen Kusumawati, M.Kom
NIP. 19720309 200501 2 002

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : Faisal Lutfi Afriansyah

NIM : 09650147

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : **RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA***

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.

Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur penjiplakan, maka saya bersedia untuk mempertanggung jawabkan, serta diproses sesuai peraturan yang berlaku.

Malang, 11 April 2013

Yang Membuat Pernyataan,

Faisal Lutfi Afriansyah

NIM. 09650147

فَبِأَيِّ ءَالَآءِ رَبِّكُمَا تُكَذِّبَانِ

“Maka nikmat Tuhan kamu yang manakah yang kamu dustakan?”

PERSEMBAHAN

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Dengan rasa syukur seraya mengharap ridho Ilahi

Kupersembahkan karya ini kepada

Ayahanda dan Ibunda tercinta

Wisono** dan **Mamik Zubaidah

yang selalu mengasihi dan merawatku hingga aku mejadi seperti sekarang.

Adik ku Dina Takti Makhfufah dan

Laili Faiqoti Alfaini

serta Seluruh keluarga besarku

yang selalu mendukung ku hingga aku dapat

menyelesaikan Skripsi ini.

Semoga Allah SWT senantiasa melindungi, menyayangi dan memberikan kita semua kebahagiaan dunia dan akhirat.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah, segala puja dan puji syukur Alhamdulillah kami panjatkan kehadirat Allah SWT yang Maha Pengasih lagi Maha Penyayang yang telah memberikan rahmat, taufik, hidayah dan inayah-Nya kepada kita serta memberikan nikmat Islam dan Iman serta tak lupa nikmat kesehatan yang diberikan kepada penulis khususnya sehingga penulis dapat menyelesaikan karya ilmiah dengan judul "**RANCANG BANGUN APLIKASI *MOBILE* UNTUK MENGETAHUI LOKASI ANAK MENGGUNAKAN ALGORITMA *DIJKSTRA***". Penelitian ini dimaksudkan untuk memenuhi salah satu syarat dalam meraih gelar Sarjana Komputer (S.Kom) di Fakultas Sains dan Teknologi Jurusan Teknik Informatika Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.

Penulis menyadari bahwa banyak pihak yang telah membantu dalam menyelesaikan penulisan tugas akhir ini. Untuk itu, iringan doa dan ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada:

1. Bapak A'la Syauqi, M.Kom selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, memotivasi, mengarahkan, serta memberikan saran, kemudahan, dan kepercayaan kepada penulis dalam menyelesaian skripsi.
2. Ibu Hani Nurhayati, M.T selaku dosen pembimbing II yang telah meluangkan waktu untuk membimbing, mengarahkan, memberikan motivasi, saran, kemudahan, dan kepercayaan kepada penulis dalam menyelesaikan skripsi.

3. Ibu Ririen Kusumawati, M.Kom ketua Jurusan Teknik Informatika Fakultas Sains Dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang yang telah memberi kemudahan, dan melancarkan proses penyelesaian skripsi.
4. Seluruh Dosen Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang yang telah mengalirkan ilmu, pengetahuan, pengalaman, dan wawasan, sebagai pedoman dan bekal bagi penulis.
5. Semua pihak yang tidak dapat penulis sebutkan satu-persatu. Penulis ucapkan terimakasih atas bantuan, dan motivasinya.

Akhirnya atas segala kekurangan dari penyusunan skripsi ini, sangat diharapkan kritik dan saran yang bersifat konstruktif dari semua pembaca demi memperbaiki kualitas penulisan selanjutnya. Semoga apa yang telah tertulis di dalam skripsi ini dapat memberikan kontribusi yang bermanfaat dan menambah khasanah ilmu pengetahuan. Amien...

Wassalamu'alaikum Wr. Wb.

Malang,11 April 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Afriansyah, Faisal Lutfi. 2013. **Rancang Bangun Aplikasi *Mobile* Untuk Mengetahui Lokasi Anak Menggunakan Algoritma *Djikstra***. Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) A'la Syauqi, M.Kom (II) Hani Nurhayati, M.T.

Kata kunci: Lokasi, Anak, Orang tua, Android, Optimasi, Algoritma Dijkstra

Dalam perkembangan awal, kita hanya mengenal adanya telepon selular dan *personal digital Assistant*. Telepon selular pada umumnya digunakan untuk melakukan komunikasi sedangkan *personal digital Assistant* digunakan sebagai asisten pribadi. Perkembangan selanjutnya pada *personal digital Assistant* adalah kemampuan untuk koneksi nirkabel sehingga mampu menerima email dan pada saat yang bersamaan juga telepon selular mendapatkan penambahan fitur yakni kemampuan untuk mengirim pesan. Pada akhirnya *personal digital Assistant* menambahkan fungsi telepon selular pada perangkatnya, begitu pula dengan telepon selular yang menambahkan *personal digital Assistant* didalamnya, sehingga hasilnya adalah sebuah *mobile smartphone*. Oleh karena itu muncul ide untuk mengembangkan sebuah aplikasi pelacakan untuk melacak posisi anak yang dilakukan oleh orang tua yang berbasis sistem operasi Android. Aplikasi ini, mempresentasikannya ke dalam peta Google Maps, dan memiliki menu untuk menuju ke posisi anak yang terakhir kali terupdate ke server. Menggunakan jarak terpendek dari posisi orang tua kepada anak dengan menggunakan Algoritma *Dijkstra* agar rute dan jarak yang diberikan optimal. Hasil uji coba yang dilakukan, apikasi dapat berjalan dengan baik pada semua *device*, tergantung koneksi data internetnya. Dari hasil pengujian terhadap 50 responden dapat disimpulkan bahwa penilaian aplikasi Pelacakan Lokasi Anak menunjukkan penilaian cukup terhadap aspek komunikasi visual dengan persentase 73,3%, Sedangkan dari aspek perangkat lunak dinilai cukup dengan persentase 69%, dan aspek relevansinya dinilai cukup dengan persentase 48%.

ABSTRACT

Afriansyah, Faisal Lutfi. 2013. **Design and Development of Mobile Applications to Finding Kid's Location with Dijkstra Algorithm.** Department of Informatics Engineering, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang,
Promotor : (I) A'la Syauqi, M.Kom (II) Hani Nurhayati, M.T.

In early development, we only know the existence of cellular phones and Personal Digital Assistant. Mobile phones are generally used for communication while used as a personal digital assistant personal assistant. Further development on the personal digital assistant is the ability for a wireless connection so that they can receive emails and at the same time the mobile phone to get additional features: the ability to send messages. In the end personal digital assistant add mobile phone functionality on the device, as well as a mobile phone that adds a personal digital assistant in it, so the result is a mobile smartphone. Hence came the idea to develop a tracking application to track the position of children by parents based on the Android operating system.This application, presenting maps into Google Maps, and has a menu to get to the position that was last updated child to the server. Using the shortest distance from the position of a parent to a child by using Dijkstra algorithm in order to route and distance are given optimal.The results of experiments performed, apikasi can run properly on all devices, depending on internet data connection. From the test results of the 50 respondents, it can be concluded that the assessment of the application appraisal Location Tracking Kids show quite the aspect of visual communication with a percentage of 73.3%, while from the software aspects were considered sufficient by the percentage of 69%, and the relevant aspects considered quite a percentage of 48%.

Keywords: Location,Children,Parents,Android, Optimization, Dijkstra Algorithm

BAB I

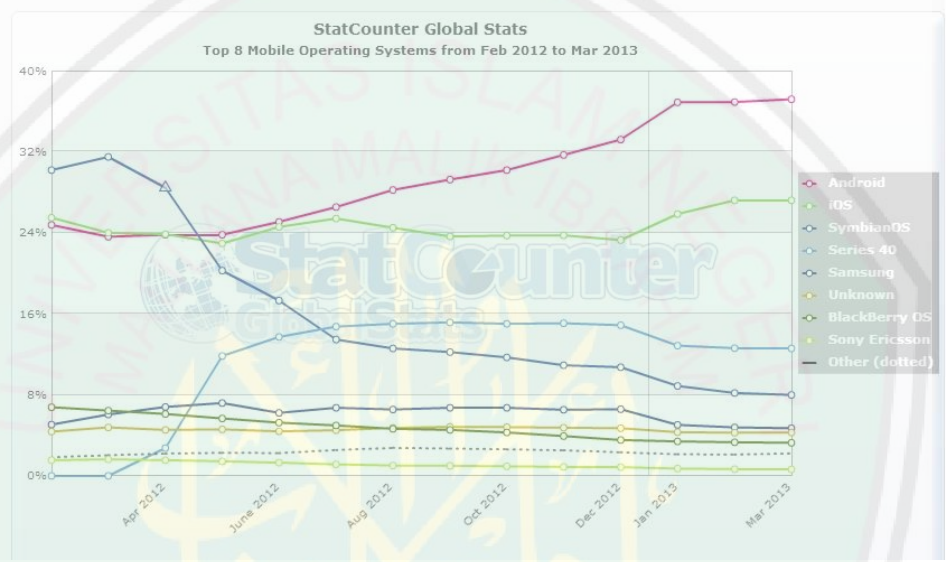
PENDAHULUAN

1.1.Latar Belakang

Teknologi perangkat bergerak sangatlah pesat perkembanganya, perangkat bergerak saat ini menuju kearah *mobile smartphone*, yang memungkinkan untuk melakukan komunikasi dan dapat melakukan fungsi *personal digital Assistant* (PDA) dan memiliki kemampuan layaknya komputer. Dalam perkembangan awal, kita hanya mengenal adanya telepon selular dan PDA. Telepon selular pada umumnya digunakan untuk melakukan komunikasi sedangkan PDA digunakan sebagai asisten pribadi. Perkembangan selanjutnya pada PDA adalah kemampuan untuk koneksi nirkabel sehingga mampu menerima *email* dan pada saat yang bersamaan juga telepon selular mendapatkan penambahan fitur yakni kemampuan untuk mengirim pesan. Pada akhirnya PDA menambahkan fungsi telepon selular pada perangkatnya, begitu pula dengan telepon selular yang menambahkan PDA didalamnya, sehingga hasilnya adalah sebuah *mobile smartphone*.

Salah satu sistem operasi yang sedang berkembang saat ini adalah sistem operasi Android yang diperkenalkan oleh Google. Android merupakan sistem operasi berbasis linux untuk perangkat bergerak. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam perangkat bergerak. Android sendiri memiliki beberapa versi yang selalu berkembang sesuai dengan fitur-fitur baru yang ditambahkan pada telepon selular, mulai dari Android versi 1.0, Android versi

1.1, Android versi 1.5 (*Cupcake*), Android versi 1.6 (*Donut*), Android versi 2.0/2.1 (*Eclair*), Android versi 2.2 (*Froyo*), Android versi 2.3 (*Gingerbread*), Android versi 4.0 (*Ice Cream Sandwich*) dan yang paling baru saat ini adalah Android versi 4.1 (*Jelly Bean*).



Gambar 1.1. Pengguna *Mobile Operating System* di dunia (sumber: gs.statcounter.com)

Saat ini Android sebagai sistem operasi *mobile smartphone* yang handal telah mendukung teknologi konektivitas yang cukup lengkap untuk mendukung fleksibilitas antara lain GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth dan Wi-Fi. Selain itu, sebagai sebuah *platform* yang terpadu, Android memiliki fitur-fitur teknis yang dibutuhkan untuk menyusun sebuah sistem operasi bergerak (*mobile OS*) misalnya fasilitas GPS untuk membantu navigasi.

Sedangkan Google juga memiliki fasilitas Google *Maps* sebagai layanan gratis untuk peta digital yang menawarkan tampilan peta dan gambar dari satelit. Google *Maps* menyediakan *Application Program Interface* (API) untuk

memungkinkan pengembang untuk mengintegrasikan Google *Maps* kedalam situs web. Pemanfaat API yang telah disediakan oleh Google *Maps* memungkinkan untuk melakukan *overlay* dengan data tertentu pada peta misalnya menampilkan posisi dengan menggambarkan sebuah penanda.

Seiring dengan perkembangan sistem operasi Android, telah banyak aplikasi yang memanfaatkan fasilitas GPS seperti untuk mencari rute, mendapatkan peta jalan sekitar. GPS sendiri dapat memberikan informasi yang tepat dan akurat mengenai posisi, kecepatan, arah dan waktu. Akan tetapi seringkali pengguna perangkat bergerak kesulitan ketika ingin mengetahui lokasi seseorang berada dengan melihat lokasi perangkat bergeraknya, termasuk jika kita ingin mengetahui lokasi anak kita berada, kemudian jika kita ingin menuju lokasi tertentu, terkadang jika kita mengikuti rute jalan yang diberikan, rute jalan yang di berikan tidak optimal sehingga kita semakin lama menuju lokasi tujuan. selain itu, Pada saat ini orang tua semakin sibuk dengan urusannya masing-masing dan hanya memiliki sedikit waktu bersama dengan anak. Hal itu menyebabkan pengawasan orang tua ke anaknya sangat berkurang, tak jarang anaknya berbohong ketika akan pergi ke suatu tempat, anak kabur dari rumah bahkan terjadinya penculikan.

Pendidikan Islam merupakan pendidikan seumur hidup, maka perlu dibedakan antara pendidikan orang dewasa dan pendidikan anak-anak. Pendidikan Islam merupakan pendidikan yang memperhatikan perkembangan jiwa anak. Oleh karena itu, pendidikan yang tidak berorientasi pada perkembangan kejiwaan akan mendapatkan hasil yang tidak maksimal, bahkan bisa membawa kepada kefatalan

anak, karena anak tumbuh dan berkembang sesuai dengan irama dan ritme perkembangan kejiwaan anak. Masing-masing periode perkembangan anak memiliki tugas-tugas perkembangan yang harus dipenuhi anak secara baik tanpa ada hambatan.

Statemen di atas, mengisyaratkan bahwa sebenarnya orang tua mempunyai tanggung jawab yang sangat besar terhadap pendidikan anaknya. Dan keluarga yang merupakan lembaga pendidikan yang pertama dan utama tersebut, wajib memberikan pendidikan agama Islam dan menjaga anaknya dari api neraka. Allah berfirman dalam Q.S.al-Tahrim/66:6

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ قُوٓاْ أَنفُسَكُمۡ وَأَهۡلِيكُمۡ نَارٗا وَقُودُهَا ٱلنَّاسُ وَٱلۡحِجَارَةُ عَلَيۡهَا مَلَٰٓئِكَةٌ غِلَاظٞ شِدَادٞ لَّا يَعۡصُونَ ٱللَّهَ مَآ أَمَرَهُمۡ وَيَفۡعَلُونَ مَا يُؤۡمَرُونَ ٦

Artinya: "*Hai orang-orang yang beriman, peliharalah dirimu dan keluargamu dari api neraka yang bahan bakarnya adalah manusia dan batu; penjaganya malaikat-malaikat yang kasar, keras, dan tidak mendurhakai Allah terhadap apa yang diperintahkan-Nya kepada mereka dan selalu mengerjakan apa yang diperintahkan*".

Oleh karena itu peneliti bermaksud untuk merancang dan membangun sebuah aplikasi pelacakan untuk melacak posisi anak yang dilakukan oleh orang tua yang berbasis sistem operasi Android dan menampilkan ke dalam peta Google *Maps*,selain itu juga ditambahkan menu untuk menuju ke posisi anak yang terakhir kali terupdate ke server. Menggunakan jarak terpendek dari posisi orang tua kepada anak dengan menggunakan Algoritma *Dijkstra* agar rute dan jarak yang diberikan optimal,sehingga mempermudah orang tua untuk menuju posisi terakhir yang *update* keserver yg dikirm oleh anak.

Routing adalah kemampuan untuk menunjukkan jalur jalan yang harus dipilih dari titik A menuju ke titik B (Riftadi,Mohammad, 2007). *Routing* pada peta goegrafis mempunyai tujuan untuk menentukan jalur paling optimal, yaitu jalur terpendek dan biaya terkecil. *Routing* pada peta geografis dilakukan karena terdapat beberapa pilihan rute (*alternative route*) yang bisa dilalui dengan jarak dan waktu tempuh berbeda pada masing –masing rute

Algortima dapat didefinisikan 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 algortima memiliki cara kerja yang berbeda-beda dalam menentukan solusi yang paling optimal.

Algoritma *Djikstra* dipilih karena dirasa algortima yang paling efektif untuk menemukan rute terpendek. Karena tiap node yang sudah dilewati 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 tentang *monitoring* lokasi anak pernah dilakukan oleh Muhammad Amrin Hakim (Hakim, 2011) Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember pada penelitian nya yang berjudul “Aplikasi *monitoring* lokasi anak menggunakan *handphone* ber-

GPS" merupakan aplikasi yang memungkinkan pengguna (orang tua) mengetahui lokasi *handphone* anak. Dalam proses tersebut pengguna (orang tua) dapat mengetahui lokasi *handphone* anak setiap saat, mengetahui lokasi *handphone* anak dalam rentang waktu tertentu dan mengetahui lokasi *handphone* anak jika anak dalam keadaan darurat, Untuk mengetahui lokasi *handphone* anak, pengguna mengirimkan *request* berupa pesan singkat dengan format tertentu. Aplikasi akan mengecek kebenaran format pesan singkat tersebut dan akan mengaktifkan *Global Positioning System*, membuat *link* peta lokasi kemudian mengirimkannya ke *handphone* pengguna.

Pada penelitian lain yang membahas tentang Algoritma *Dijkstra* di lakukan oleh Amirullah Andi Bramantya (Bramantya, 2012) jurusan Teknik Informatika UIN Maulana Malik Ibrahim Malang dalam penelitiannya yang berjudul "Aplikasi *location based service* untuk menentukan rute terpendek lokasi atm di kota malang menggunakan algoritma *dijkstra*" merupakan penelitian tentang lokasi ATM terdekat berdasarkan posisi *user*, kemudian *user* bisa menuju lokasi ATM dengan jalur terpendek menggunakan algoritma *Dijkstra* menuju lokasi ATM terdekat tersebut.

1.2.Rumusan Masalah

Berdasarkan penjelasan latar belakang di atas, maka perumusan masalah dalam penelitian ini adalah bagaimana mengetahui lokasi anak dan menuju lokasi anak dengan jarak terpendek menggunakan Algoritma *Dijkstra*?

1.3.Tujuan Penelitian

Tujuan dari pembuatan perangkat lunak ini adalah untuk membangun sebuah aplikasi yang dapat membantu orang tua mengetahui dan melakukan pencarian lokasi sebuah perangkat bergerak yang digunakan oleh anak kemudian menampilkan hasilnya pada sebuah peta digital dan menentukan jarak dan rute maksimal untuk menuju lokasi anak menggunakan Algoritma *Dijkstra*.

1.4.Manfaat Penelitian

Manfaat yang dapat diambil dari aplikasi yang dibangun ini adalah untuk memberikan kontribusi perkembangan teknologi perangkat bergerak untuk membantu orang tua mengetahui lokasi anaknya dan penentuan rute jarak terpendek untuk menuju lokasi anak.

1.5.Batasan Masalah

Pembatasan masalah yang dimaksudkan untuk membatasi ruang lingkup permasalahan yang akan dibahas 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 digunakan adalah Java untuk Android yang diimplementasikan ke dalam suatu perangkat bergerak dengan sistem operasi Android
b. Server menggunakan bahasa pemrograman PHP serta menggunakan *database* MySQL
c. Peta digital yang digunakan memanfaatkan API dari Google *Maps*.

d. Minimal *platform* Android yang digunakan adalah Android versi 2.3 (*Gingerbread*).

e. GPS *built-in* ponsel selalu menyala.

f. Koneksi Internet selalu aktif,Jika tidak aktif maka aplikasi tidak dapat berjalan dengan semestinya.

g. Segmentasi *user* (anak) aplikasi ini adalah anak usia 6 – 17 Tahun

h. Implementasi *node* untuk Algoritma *Djikstra* hanya bisa dilakukan pada *node* yang telah terdefinisi yaitu jalan Gajayana, Jalan Bendungan Sutami, Jalan Galunggung, Jalan Kawi atas, Jalan Ijen dan Jalan Veteran.

1.6.Metode Penelitian

Dalam rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan Algoritma *Djikstra* ini mempunyai beberapa tahapan dalam metode pengerjaannya. Tahapan-tahapannya yaitu diawali dengan melakukan studi kepustakaan terhadap berbagai pustaka yang berkaitan dengan penelitian. Tahap berikutnya yaitu melakukan perancangan sistem di sisi server, yaitu menyiapkan *database* server dengan MySQL dan menyiapkan web server menggunakan Apache dengan modul PHP. Sedangkan persiapan aplikasi di sisi *client* menggunakan aplikasi java berbasis Android yang dibangun dengan IDE Eclipse dan Android SDK sebagai *development tools*.

1.7. Sistematika Penulisan

Laporan tugas akhir ini dibuat dengan sistem penulisan sebagai berikut :

BAB I 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 pelacakan anak.

BAB IV HASIL DAN PEMBAHASAN

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

BAB V PENUTUP

Berisi tentang kesimpulan yang diambil dari pembahasan program aplikasi pelacakan anak dan saran untuk pengembangannya.

BAB II

TINJAUAN PUSTAKA

2.1. GPS (*Global Positioning System*)

GPS (*Global Positioning System*) adalah sistem untuk menentukan posisi di permukaan bumi dengan bantuan sinkronisasi sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan posisi, kecepatan, arah dan waktu. Sistem yang serupa dengan GPS antara lain GLONASS Rusia, Galileo Uni Eropa, IRNSS India. Sistem ini dikembangkan oleh Departemen Pertahanan Amerika Serikat, dengan nama lengkapnya adalah NAVSTAR GPS (Parkinson, B.W.1996).

2.2. Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi (Safaat,Nazruddin 2011).

2.2.1. Arsitektur Android

Android akan bekerja dengan serangkaian aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2.2.1.1. Linux Kernel

Android dibangun dia atas kernel Linux 2.6. namun secara keseluruhan Android bukanlah linux, karena di dalam Android tidak terdapat paket standar yang dimiliki Linux lainnya. Kernel Linux menyediakan *driver* layar, kamera,

keypad, WiFi, *Flash Memory*, *audio*, dan IPC (*Inter Process Communication*) untuk mengatur aplikasi dan celah keamanan.

2.2.1.2. *Libraries*

Android menggunakan beberapa paket *library* yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari aslinya untuk tertanam pada kernel Linux. Beberapa *library* diantaranya:

a. Media *Library* untuk memutar dan merekam berbagai macam format audio dan video.
b. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
c. *Graphic Library* termasuk didalamnya SGL dan OpenGL untuk tampilan 2D dan 3D.
d. SQLite untuk mengatur relasi *database* yang digunakan pada aplikasi.
e. SSL dan WebKit untuk *browser* dan keamanan internet.

2.2.1.3. Android *Runtime*

Android *Runtime* merupakan mesin *virtual* yang membuat aplikasi Android menjadi lebih tangguh dengan paket *library* yang telah ada. Dalam Android Runtime terdapat 2 bagian utama,diantaranya:

a. *Core Libraries*. Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. *Core Libraries* Android menyediakan hampir semua fungsi

yang terdapat pada *library* Java serta beberapa *library* khusus Android.

b. *Dalvik Virtual Machine*. Dalvik merupakan sebuah mesin *virtual* yang dikembangkan 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).

2.2.1.4. *Application Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

a. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan "*Backstack*" untuk navigasi penggunaan.
b. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
c. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan *file layout*.
d. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.

e. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

2.2.1.5. *Application Layer*

Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam *Android Runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi.

2.2.2. Aplikasi Fundamental

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi-bersama dengan *file* data dan sumber daya yang dibutuhkan oleh aplikasi, dipaketkan dengan alat apt ke dalam paket Android, *file* arsip ditandai oleh akhiran APK. *File* ini adalah bagian untuk mendistribusikan aplikasi dan menginstal 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 dijalankan, 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 VM 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 mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan 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* dideklarasikan 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 tapi 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 dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih.
d. *Inactive*, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* 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 atau tidak tampak.

2.2.3.3. Intents

Intens 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, misal 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 baterei sudah hampir 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* sistem, 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 Activity

Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika di perlukan akan memberi 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 karenanya perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.2.5. Kelebihan Android

a. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan *open source*.

b. Arsitektur komponen dasar Android terinspirasi dari teknologi internet *Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.

c. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database* SQL, *browser* dan penggunaan peta.

d. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil.

e. Dukungan grafis, dengan adanya dukungan 2D grafis dan animasi yang di ilhami oleh *Flash* menyatu dalam 3D menggunakan OpenGL memungkinkan membuat aplikasi maupun *game* yang berbeda.

2.2.6. Google *Maps*

Google *Maps* merupakan layanan dari Google yang mempermudah pengunanya untuk melakukan kemampuan pemetaan untuk aplikasi yang dibuat. Sedangkan Google *Maps* API memungkinkan pengembangan untuk mengintegrasikan Google *Maps* ke dalam situs web. Dengan menggunakan Google *Maps* API memungkinkan untuk menanamkan situs Google *Maps* ke dalam situs eksternal, di mana situs data tertentu dapat dilakukan *overlay*.

Meskipun pada awalnya hanya JavaScript API, API Maps sejak diperluas untuk menyertakan sebuah API untuk Adobe Flash, layanan untuk mengambil gambar peta statis, dan layanan web untuk melakukan *geocoding*, menghasilkan petunjuk arah mengemudi, dan mendapatkan profil elevasi.

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 jenis tampilan di atas peta.

Secara umum, kelas *MapView* menyediakan lapisan luar di Google *Maps* API yang memungkinkan aplikasi tersebut memanipulasi data Google *Maps* melalui kelas *method*, dan itu memungkinkan dikerjakan dengan data *Maps* seperti jenis lain *Views*. Perpustakaan *Maps* eksternal bukan bagian dari perpustakaan Android standar, sehingga tidak mungkin ada pada beberapa perangkat Android biasa. Demikian pula, perpustakaan *Maps* eksternal tidak termasuk dalam perpustakaan Android standar yang disediakan dalam SDK. Google API menyediakan perpustakaan *Maps* untuk sehingga dapat mengembangkan, membangun, dan menjalankan aplikasi berbasis peta di SDK Android, dengan akses penuh ke data Google *Maps*.

2.3. Optimasi

2.3.1. Definisi Optimasi dan Nilai Optimal

Optimisasi ialah suatu proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dicapai). Dalam disiplin matematika, optimisasi merujuk pada studi permasalahn 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 menjadi suatu solusi jawaban yang paling baik dari semua solusi yang ada.

2.3.2. Penyelesain 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 tependek, diantaranya: algortima *Dijkstra*, algoritma *Floyd-Marshall*, dan algoritma *Bellman-Ford*.

2.3.2.2. Metode Heuristik

Metode heuristic adalah sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi. Ada beberapa algortima pada metode heuristic yang biasa digunakan dalam permasalahan optimasi, diantaranya algortima genetika, algortima semut, logika fuzzy, jaringan syaraf tiruan, *tabu search*, *simulated annealing*, dan lain-lain.

2.3.3. Permasalahan Jalur Terpendek (*Shortest Path Problem*)

Awal mula dari permasalahan ini adalah untuk menemukan jalur terpendek dari *vertex* S ke *vertex* T pada jaringan yang berbobot. Untuk melakukannya, kita bergerak melawati jaringan tersebut dari kiri ke kanan, menghitung jarak terpendek dari S ke setiap *node* yang mengarah pada tujuan kita.di setiap langkah algortima, kita melihat semua simpul ke tujuan dari *vertex* sekarang ke *vertex* yang memiliki nilai yang terdekat dari S dengan garis. Sebenarnya setiap *vertex* membutuhkan label permanen (potensinya, dan disimbolkan di sekeliling tabel), yang menunujukkan 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 *Dijkstra*

Dijkstra Algorithms (Shortest Path Algorithms) adalah algoritma untuk menemukan jarak terpendek dari suatu *vertex* ke *vertex* yang lainnya pada suatu *graph* yang berbobot, dimana jarak antar *vertex* adalah bobot dari tiap *edge* pada *graph* tersebut. Algoritma *Djikstra* mencari jarak terpendek untuk tiap *vertex* dari

suatu *graph* yang berbobot. Algoritma *Dijkstra* mencari jarak terpendek dari *node* asal ke *vertex* terdekatnya, kemudian ke *vertex* kedua, dan seterusnya. Secara umum, sebelum dilakukan iterasi, algoritma sudah mengidentifikasi 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 dari node asal.

Algoritma *Dijkstra* merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana (*straightforward*). Sesuai dengan arti *greedy* yang secara harafiah berarti tamak atau rakus - namun tidak dalam konteks *negative*, algoritma *greedy* ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan.

2.4.1. Elemen Penyusun Prinsip *Greedy* pada 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 yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

2.4.1.4. **Fungsi kelayakan**

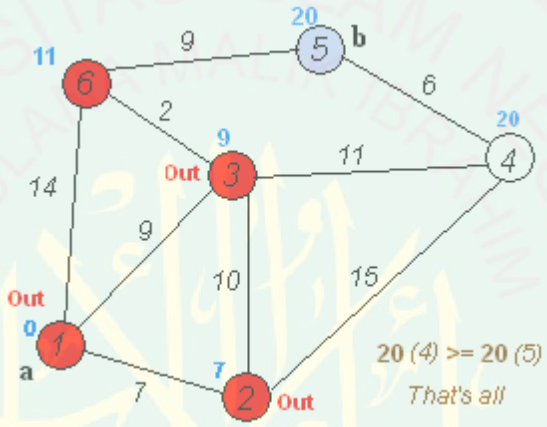
Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar constraint atau tidak. Apabila kandidat melanggar constraint maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

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 *Djikstra* 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 digunakan dalam mencari lintasan terpendek adalah *graph* berbobot. Bobot pada sisi *graph* dapat menyatakan jarak antar kota, waktu, biaya dan sebagainya. Dalam hal ini bobot harus bernilai positif seperti contoh pada Gambar 2.2.

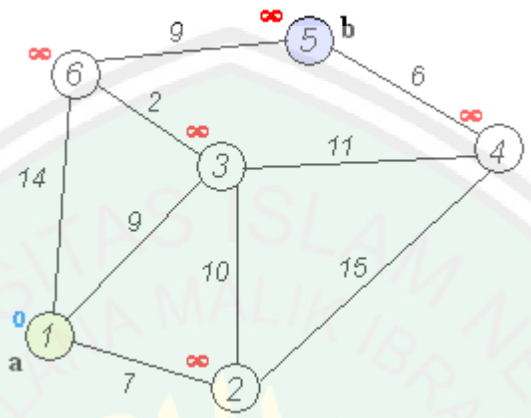


Gambar 2.1 Algoritma *Dijkstra*

Aplikasi ini akan menjelaskan implementasi dari algoritma *Dijkstra*. Aplikasi ini sudah menyediakan *graf*, untuk itu pada perancangannya hanya perlu memasukkan node tujuan (lokasi Anak).

Pemilihan *node*-nya sendiri berdasarkan pada bobot yang terendah dari tabel, dan *predecessor* merupakan node 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 dihasilkan 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. Contoh 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 per satu seperti pada Gambar 2.3, *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 melewati B menjadi 6+2=8. Jika jarak ini

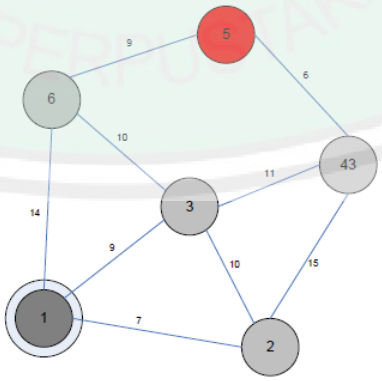
lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

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 Algortima *Dijkstra*

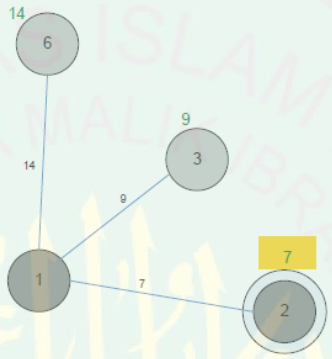
Dibawah ini penjelasan langkah per langkah pencarian jalur terpendek secara rinci dimulai dari *node* awal sampai *node* tujuan dengan nilai jarak terkecil.

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



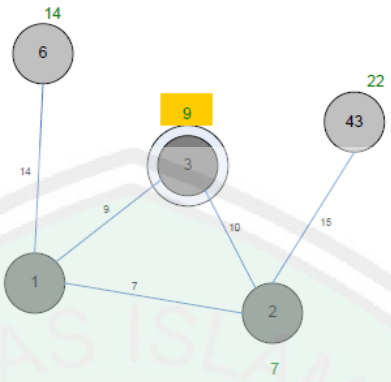
Gambar 2.3 Contoh kasus Djikstra - Langkah 1

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).



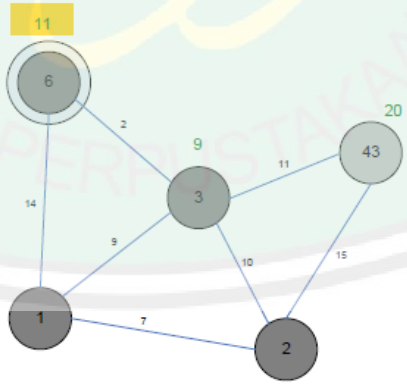
Gambar 2.4 Contoh kasus Djikstra - Langkah 2

3. *Node* 2 diset menjadi *node* keberangkatan dan ditandai sebagi *node* yang telah terpilih. *Dijkstra* melakukan kalkulasi kembali terhadap *node-node* tetangga yang terhubung langsung dengan *node* yang telah terpilih. Kemudian pada Gambar 2.5, kalkulasi *Dijkstra* menunjukan bahwa *node* 3 yang menjadi node keberangkatan selanjutnya karena bobotnya yang paling kecil dari hasil kalkulasi terakhir, nilai 9 (0+9).



Gambar 2.5 Contoh kasus Djikstra - Langkah 3

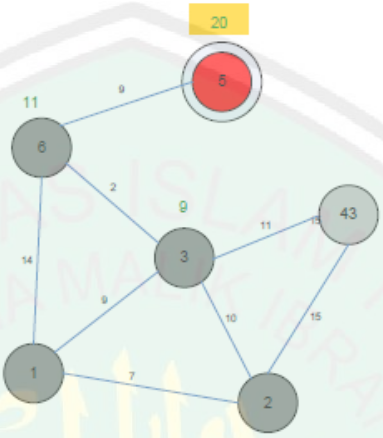
4. Perhitungan berlanjut pada Gambar 2.6, dengan *node* 3 ditandai menjadi *node* yang telah terpilih. Dari semua node tetangga belum terlewati yang terhubung langsung dengan node terlewati, node selanjutnya yang ditandai menjadi node terpilih adalah node 6 karena nilai bobot yang terkecil, nilai 11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



Gambar 2.6 Contoh kasus Djikstra - 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 1-3-6-5, dan nilai bobot yang

didapat adalah 20 (11+9). Bila *node* tujuan telah tercapai maka kalkulasi *dijkstra* dinyatakan selesai.



Gambar 2.7 Contoh kasus Djikstra - Langkah 5

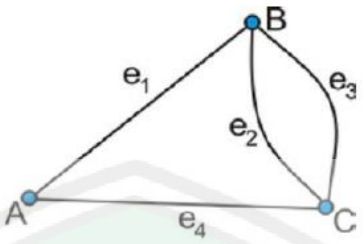
2.5. Teori *Graf*

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, sperti 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*

Pada Gambar 2.8, *graf* terdiri dari himpunan V dan E yaitu :

V = {A, B, C}

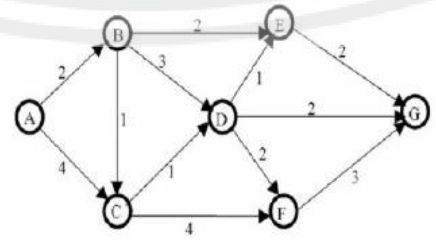
E = {e1, e2, e3, e4}

2.5.2. Macam-macam *Graf*

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

2.5.2.1. *Graf* berarah dan berbobot

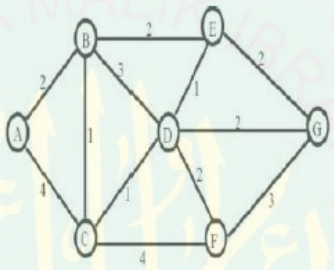
Tiap busur mempunyai anak panah dan bobot. Gambar 2.9 menunjukkan *graf* berarahan berbobot yang terdiri dari tujuh titik yaitu titik A, B, C, D, E, F, G. Titik menunjukkan arah ke titik B dan titik C, titik B menunujukkan arah ke titik D dan titik C dan seterusnya. Bobot antar titik A dan titik B pun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



Gambar 2.9 *Graf* Berarah dan Berbobot

2.5.2.2. *Graf* tidak berarah dan berbobot

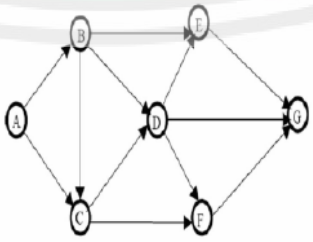
Tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.10 menunjukkan *graf* tidak berarah dan berbobot. *Graf* terdiri tujuh titik yaitu titik A, B, C, D, E, F, G. Pada Gambar 2.10, titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui.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



Gambar 2.10 *Graf* Tidak Berarah dan Berbobot

2.5.2.3. Graf berarah dan tidak berbobot

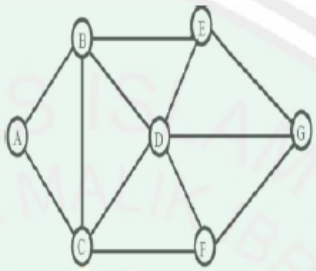
Tiap busur mempunyai anak panah yang tidak berbobot. Gambar 2.11 menunjukkan graf berarah dan tidak berbobot.



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 didefinisikan 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 = derjat 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,v2),…, en = (vn-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 terdapat 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 kuat (*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 ⊂ V dan E1 ⊂ 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 terbagi menjadi dua macam, yaitu komponen *software* dan *hardware*.

3.1.1. Software

Software yang dibutuhkan dalam proses pembuatan aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *dijkstra* adalah sebagai berikut:

a. *Windows* 7

   Windows 7 digunakan karena lebih banyak *support* dan *compatible* dengan *software* lain yang dibutuhkan dalam pembuatan aplikasi *mobile* ini.

b. *Java Development Kit* (JDK) versi 7.1

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

c. *Java Runtime Evironment* (JRE) versi 7

   Supaya sebuah 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 Interface*) yang diperlukan untuk mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman java. Pada pembuatan aplikasi pelacakan ini, menggunakan Android SDK versi 21.1.

e. Eclipse

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

f. *Database* MySQL

Database MySQL merupakan suatu perangkat lunak *database* yang berbentuk *database* relasional atau disebut *Relational Database Management System* (RDBMS) yang menggunakan bahasa SQL (*Structured Query Language*).

3.1.2. Hardware

Dalam pembuatan aplikasi pelacakan anak ini, *hardware* yang dibutuhkan antara lain:

a. Komputer

Komputer yang digunakan untuk membangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* mempunyai spesifikasi sebagai berikut:

1. Intel (R) Core (TM) 2 Duo CPU T6600 @2.20 GHz
2. RAM 4096 MB

3. *Hardisk* 320 GB

b. Smartphone

Selain menggunakan emulator android yang diintegrasikan dengan *Eclipse,* peneliti juga menguji aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* ini pada *smartphone* android Sony Xperia Go dengan spesifikasi sebagai berikut:

1. Prosesor Novathor U8500 Dual-Core 1 GHz Cortex-A9
2. Memori *internal* 8 GB
3. RAM 512 MB
4. Ukuran Layar 3.5, 320 x 480 piksel 16 M *Color*

3.2.Spesifikasi Aplikasi

a. Menyediakan layanan bagi orang tua untuk mengetahui posisi lokasi anak terkini.
b. Menyediakan fasilitas pada anak untuk mengirimkan lokasi anak tersebut ke server secara periodik.
c. Menyediakan fasilitas untuk menuju posisi anak terkini dengan jarak terpendek.

3.3.Spesifikasi Pengguna

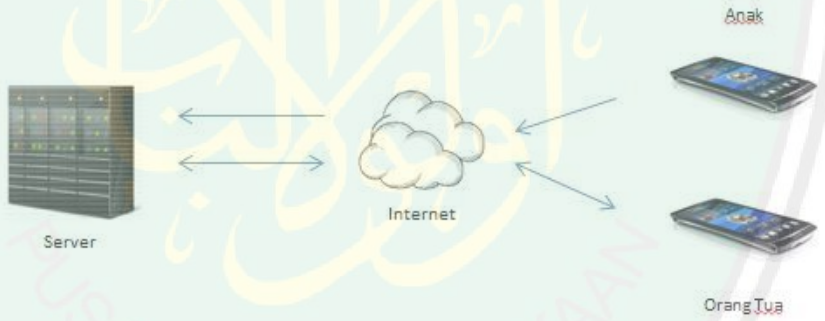
Aplikasi *mobile* ini digunakan oleh orang tua dan anak. aplikasi *mobile* untuk orang tua di instal dan digunakan di *smartphone* Android milik orang tua untuk menampilkan posisi anak kemudian rute menuju anak, Sedangkan aplikasi *mobile* untuk anak di install di *smartphone* Android milik anak berfungsi untuk mengirim secara berkala posisi anak ke server.

3.4.Desain Sistem

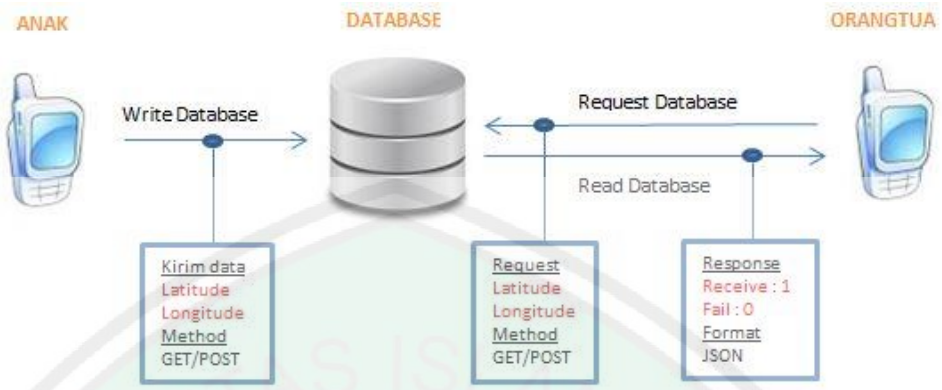
Analisa sistem rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* terbagi dalam dua aplikasi sebagai berikut :

a. Aplikasi Anak yaitu aplikasi berbasis sistem operasi Android yang berfungsi untuk mengirimkan *Latitude Longitude* posisi anak ke server secara berkala.
b. Aplikasi Orang tua yaitu aplikasi berbasis sistem operasi Android yang berfungsi untuk mengakses posisi anak yang dikirim oleh anak kemudian ditampilkan pada aplikasi orang tua kemudian juga ada fitur menuju posisi anak menggunakan algoritma *Dijkstra*.

Desain sistem yang digambarkan, dibawah ini:



Gambar 3.1 Desain Sistem



Gambar 3.2 Koneksi antara Aplikasi Ayah, Aplikasi Anak dan Database

Pada Gambar 3.1 menunjukkan bahwa secara fisik arsitektur Rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan Algoritma *Dijkstra* ini dibagi menjadi dua,yaitu aplikasi untuk anak untuk mengirim *latitude longitude* posisi anak tersebut ke server,kemudian aplikasi orang tua untuk membuat request *latitude longitude* posisi anak kemudian nanti oleh server dikirimkan *latitude longitude* anak ke aplikasi orang tua untuk ditampilkan dalam bentuk *maps* dan ada *marker* anak pada posisi *latitude longitude* yg dikirim oleh server tersebut.

Dengan demikian, Proses pertukaran data dari aplikasi anak ke server kemudian dari server ke orang tua dan sebaliknya orang tua *request* ke server menggunakan koneksi protokol HTTP, Format data yang dikirim dari anak ke server menggunakan aturan yang sesuai dengan protokol HTTP seperti *GET* dan *POST*. Sedangkan respon dari server, data dikirim dalam format JSON (Lihat Gambar 3.2).

3.4.1. Fungsi Sistem

Fungsi- fungsi yang dapat di identifikasi dari rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* ini adalah sistem mampu untuk :

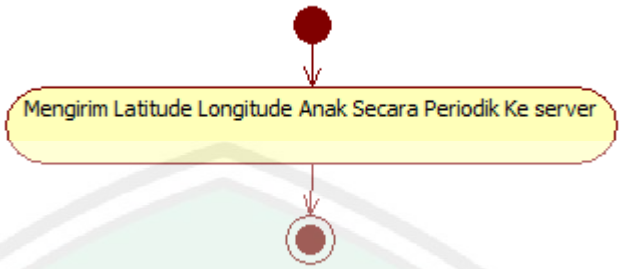
a. Melacak *latitude longitude* anak menggunakan GPS yg ada pada aplikasi anak.
b. Orang tua bisa mengetahui lokasi anak dan Lokasi anak tersebut ditampilkan pada aplikasi orang tua dalam bentuk *maps* dan ada *marker* anak pada *latitude longitude* anak tersebut.
c. Orang tua dapat menuju lokasi anak dengan menggunakan Jalur rute terpendek menggunakan algoritma *Dijkstra*.

3.4.2. Analisa *Activity* Diagram

Berikut adalah *activity* diagram secara keseluruhan proses yang terjadi di dalam sistem rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* yang meliputi anak dan orang tua

3.4.2.1. *Activity* Diagram Anak

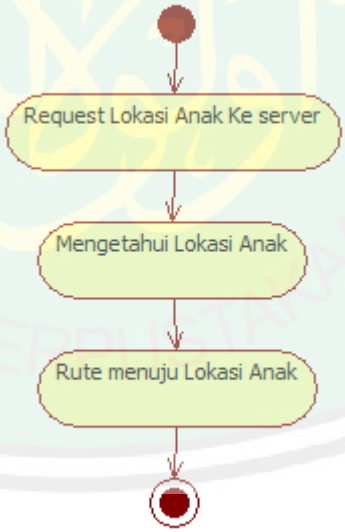
Pada Anak ada aplikasi untuk anak mengirim *latitude longitude* anak secara otomatis dan berkala ke server sehingga keberadaan anak akan selalu terekam di server.



Gambar 3.3 *Activity* Diagram Anak

3.4.2.2. *Activity* Diagram Orang tua

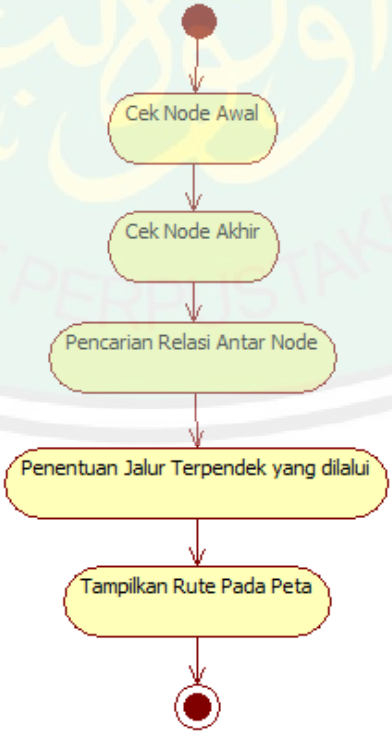
Aplikasi Orang tua *request latitude longitude* anak keserver kemudian menampilkan *latitude longitude* anak dalam bentuk *maps* kemudian orang tua bisa menuju lokasi anak dengan rute terpendek menggunakan algoritma *Dijkstra*.



Gambar 3.4 *Activity* Diagram Orang tua

3.4.2.3. *Activity* Diagram Penentuan Rute

Setelah mendapatkan lokasi orang tua, maka program akan melakukan pengecekan lokasi *node* simpang jalan terdekat dari orang tua saat ini, yang akan dijadikan *node* awal dari berjalannya algoritma *Dijkstra*, Setelah lokasi anak diambil dari server, program akan menentukan *node* simpang jalan yang terdekat dari lokasi anak, Setelah *node* awal dan *node* akhir diketahui, maka relasi antar *node* yang berhubungan akan dicari dan ditentukan *node* mana yang menghubungkan *node* awal ke *node* akhir, setelah node yang berhubungan diketahui kemudian dicari *node* yang memiliki jarak terpendek dari *node* awal menuju *node* akhir, *Node-node* yang terpilih akan dijadikan titik dari jalur rute jalan terpendek yang dilalui *user* untuk menuju lokasi anak.



Gambar 3.5 *Activity* Diagram Penentuan Rute

3.4.3. Struktur *Database*

Tabel struktur data yang digunakan dalam aplikasi ini, yaitu :

3.4.3.1. Tabel Posisi

Tabel

Tabel 3.1 adalah tabel untuk menyimpan data *latitude* dan *longitude* yang dikirm oleh aplikasi anak, mempunyai struktur tabel sebagai berikut :

Tabel 3.1. Tabel Posisi

Kolom	Jenis
ID	int(50)
Latitude	varchar(12)
Longitude	varchar(12)
Waktu	timestamp

Keterangan dari Tabel diatas adalah :

a. ID, untuk memberikan id ketika anak mengirim *latitude* dan *longitude* ke server,selain itu ID ini juga di gunakan untuk *shorting* lokasi anak terakhir yang di *parsing* ke aplikasi orang tua.
b. *Latitude*, untuk menerima data *latitude* yang dikirim anak.
c. *Longitude*, untuk menerima data *longitude* yang dikirim oleh anak.
d. Waktu, untuk menyimpan data waktu ketika *latitutde longitude* diterima oleh server.

3.4.3.2. Tabel *Graf*

Tabel 3.2 adalah tabel untuk menyimpan data *node graf* data pada tabel ini di inputkan secara manual berisi *node* yang nantinya akan menghubungkan dari titik awal ke titik tujuan, mempunyai struktur tabel sebagai berikut :

Tabel 3.2. Tabel *Graf*

Kolom	Jenis
Id_Node	int(50)
Latitude	varchar(12)
Longitude	varchar(12)
Titik_*Graf*	int(2)

Keterangan dari Tabel diatas adalah :

a. Id_*Node*, untuk memberikan id tiap-tiap *node graf*.
b. *Latitude*, untuk menyimpan data *latitude* dari *node graf*.
c. *Longitude*, untuk menyimpan data *longitude* dari *node graf*.
d. Titik_*Graf*, untuk memberikan aturan tiap-tiap *node graf*,agar memudahkan ketika dalam proses perhitungan.

3.4.3.3. Tabel Jalan

Table 3.3 adalah tabel untuk menyimpan data jalan dan relasi antar *node* yang saling berhubungan dan juga untuk menentukan aturan apakah jalur yang dilalui searah atau dua arah, untuk datanya dinputkan secara manual, mempunyai struktur tabel sebagai berikut :

Tabel 3.3. Tabel Jalan

Kolom	Jenis
Id_Jalan	int(50)
*Node*_Awal	varchar(12)
*Node*_Akhir	varchar(12)
Panjang_Jalan	double

Keterangan dari Tabel diatas adalah :

a. Id_Jalan, untuk memberikan id setiap jalan antar *node graf*.
b. *Node*_Awal, untuk menyimpan *id graf node* awal yang akan di hitung.
c. *Node*_Akhir, untuk menyimpan *id graf node* akhir yang akan di hitung.
d. Panjang_Jalan, untuk menyimpan panjang jalan antar *Node*_Awal ke *Node*_Akhir atau sebaliknya.

3.4.3.4. Tabel *Temp*

Tabel 3.4 adalah tabel untuk menyimpan hasil sementara dari perhitungan algoritma, mempunyai struktur tabel sebagai berikut :

Tabel 3.4. Tabel *temp*

Kolom	Jenis
Awalan	int(50)
Akhiran	varchar(12)
Panjang	varchar(12)

Keterangan dari Tabel diatas adalah :

a. Awalan, digunakan untuk menyimpan awalan titik *graf* yang dihitung.
b. Akhiran, digunakan untuk menyimpan akhiran titik *graf* yang dihitung.
c. Panjang, digunakan untuk menyimpan panjang lintasan dari Awalan ke Akhiran.

3.4.4. Desain *Interface*

Hal yang perlu diperhatikan dalam mendesain sebuah sistem adalah rancangan tersebut harus dapat memudahkan pengguna dalam menggunakan sistem aplikasi yang dibuat. Sehingga perlu diperhatikan dalam mengatur letak menu, ataupun komponen visual yang lain sehingga tidak membingungkan pengguna dalam pemakaian. Berikut adalah perancangan menu utama sistem Rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra*.

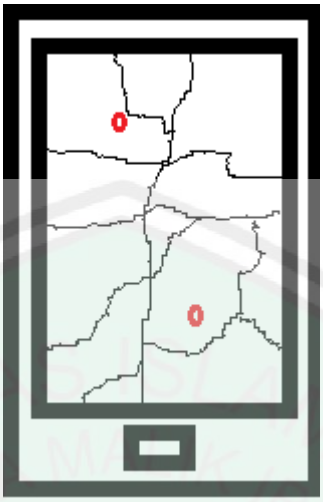
3.4.4.1. Halaman Utama Aplikasi Anak

Gambar 3.6 menampil informasi *latitude longitude* dan setiap anak berubah posisi nilai inipun akan *update* setiap 30 detik sekali. Halaman ini berfungsi mengirimkan *latitude longitude* lokasi anak keserver secara berkala sesuai dengan nilai *latitude longitude* yg tertera pada aplikasi, aplikasi ini akan tetap aktif mengirim posisi anak meskipun aplikasi ini tidak harus dibuka.

Gambar 3.6 Halaman Utama Aplikasi Anak

3.4.4.2. Halaman Utama Aplikasi Orang tua

Halaman utama pada Gambar 3.7 merupakan halaman yang pertama kali tampil setelah halaman *Splash Screen*. Halaman ini berfungsi menampilkan lokasi anak dan lokasi orang tua, jadi pada halaman utama ini akan tampil posisi anak yg di *parsing* dari server dan juga posisi orang tua yang di ambil melalui GPS *device* orang tua.



Gambar 3.7 Halaman Utama Aplikasi Orang tua

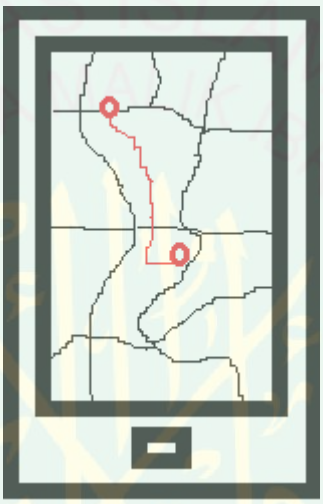
3.4.4.3. Halaman Menu

Halaman ini tampil untuk memberitahukan kepada orang tua bahwa orang tua bisa memilih menggunakan menu rute untuk menggambarkan rute menuju lokasi anak, menu ini muncul ketika orang tua menyentuh *marker* anak maupun marker orang tua sendiri,maka nanti akan muncul notifikasi seperti pada Gambar 3.8.

Gambar 3.8 Halaman Menu Aplikasi Orang tua

3.4.4.4. Rute Menuju Lokasi Anak

Halaman ini menampilkan jalur rute jalan menuju lokasi anak, sehingga orang tua tahu jalur rute terdekat menuju lokasi anak menggunakan algoritma *Dijkstra*, pada Gambar 3.9 tampilan jalur rute di tunjukkan dengan garis berwarna merah



Gambar 3.9 Rute Menuju Lokasi Anak pada Aplikasi Orang tua

Pada Gambar 3.10 menjelaskan *flowchart* dari proses perhitungan *Dijkstra* pada aplikasi pelacakan anak, langkah pertama yaitu mengambil *vertex* tujuan yg ada pada tabel_posisi di *database* yang dikirimkan oleh anak dengan id_node pada tabel_graf adalah titik_graf 3, titik_graf dengan id 3 adalah kode untuk titik graf tujuan, kemudian sistem akan mengecek relasi *vertex*, relasi *vertex* ini tersimpan di *database* pada tabel_graf, kemudian menghitung relasi *vertex* yg terdekat yang ada pada tabel_graf adalah titik_graf 2, titik_graf dengan id 2 adalah kode untuk titik graf simpangan jalan. maksudnya menghitung disini adalah menghitung bobot antar *node*,bobot antar *node* tersebut ada dalam *database* pada

tabel_jalan,kemudian sistem akan terus *looping* ke semua *node* untuk mencari bobot terkecil dan disimpan pada tabel_temp dan setiap menemukan bobot terpendek maka tabel_temp akan terus *update* dan menghapus bobot yg lebih panjang dan di *update* dengan bobot yg lebih pendek,sampai nantinya *node* yg terpilih adalah *node* tujuan dengan id_node pada tabel_graf adalah 3.jika syarat itu telah terpenuhi maka *node* yang terpilih dengan bobot terkecil yang telah terpilih tadi akan di gambar dari titik awal yaitu posisi GPS *device* orang tua menuju *vertex* tujuan yaitu lokasi anak pada peta dengan menggunakan *maps* API.

Gambar 3.10. *Flowchart* Perhitungan *Dijkstra*

BAB IV

HASIL DAN PEMBAHASAN

Dalam bab ini akan dibahas mengenai hasil uji coba terhadap sistem rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* yang telah dibuat. Uji coba ini bertujuan untuk mengetahui apakah sistem aplikasi yang dibuat telah dapat berjalan sebagaimana mestinya sesuai dengan rancangan sistem pada BAB 3. Pada bab ini juga akan dibahas mengenai fitur dan *interface* yang terdapat di dalam sistem rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra*.

4.1. Implementasi Sistem

Dalam tahap implementasi ini, sistem yang telah didesain mulai diterapkan dengan membangun komponen-komponen yang telah direncanakan.

4.1.1. Implementasi Aplikasi Anak

Desain rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* pada sisi anak diimplementasikan dengan MySQL dan PHP pada Apache *web server*. MySQL sebagai basis data dan PHP sebagai pemroses untuk menerima data yang dikirim oleh aplikasi anak. Dalam hal ini, *file* PHP tersebut merupakan penjembatan antara *database* MySQL dengan aplikasi anak. Sedangkan Apache *web server* berfungsi sebagai jembatan komunikasi dengan protokol HTTP.

Sistem pemroses dalam Apache *web server* berisi *file-file* php yang berfungsi untuk menangkap kiriman dari aplikasi dengan *Method GET/POST*.

4.1.2. Implementasi Aplikasi Orang tua

Desain pada Aplikasi Orang tua di implementasikan dengan *database* MySQL dan PHP pada Apache *web server*. MySQL sebagai basis data dan PHP sebagai pemroses untuk menerima data yang diminta oleh aplikasi orang tua. Dalam hal ini, *file* PHP tersebut merupakan penjembatan antara *database* MySQL dengan *request* dari aplikasi orang tua Sedangkan Apache *web server* berfungsi sebagai jembatan komunikasi dengan protokol HTTP.

Sistem pemroses dalam Apache *web server* berisi *file-file* php yang berfungsi untuk memberikan *response* kepada aplikasi Ayah dengan format JSON. Pada apliksi orang tua uji coba perangkat menggunakan *smartphone* Sony Xperia Go yang bersistem operasi android dengan versi 4.0.4 Sedangkan koneksi antara *client* dan *server* menggunakan internet.

4.1.3. Ruang Lingkup Perangkat Keras

Ruang lingkup perangkat keras yang diperlukan oleh sistem rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* ini antara lain:

- Prosesor Intel (R) Core (TM) 2 Duo CPU T6600 @2.20 GHz
- *Hard Disk* 320 GB
- Monitor 14”
- *Keyboard*

- *Mouse Wireless*
- *Smartphone* Android

4.1.4. Ruang Lingkup Perangkat Lunak

Ruang lingkup perangkat lunak yang diperlukan oleh sistem antara lain:

- *Windows 7 Home Premium*
- *Java Development Kit* (JDK) versi 7.1
- *Java Runtime Environment* (JRE) versi 7
- Eclipse
- Android SDK
- ADT 21.0.1

4.2. Implementasi *Interface*

Pada rancang bangun aplikasi *mobile* untuk mengetahui lokasi anak menggunakan algoritma *Dijkstra* ini terbagi dalam 2 *interface*, yaitu:

1. Aplikasi Anak

   1.1.Halaman utama sebelum menerima Sinyal GPS

   1.2.Halaman setelah menerima sinyal GPS

2. Aplikasi Orang tua

   2.1.Halaman utama aplikasi orang tua

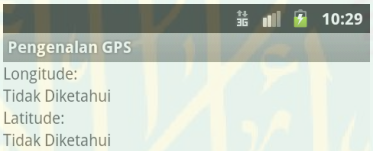
   2.2.Halaman menu rute

   2.3.Halaman rute menuju lokasi anak

4.2.1. Aplikasi Anak

4.2.1.1. Halaman Utama Sebelum menerima Sinyal GPS

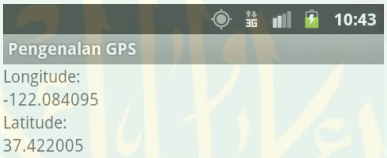
Halaman ini merupakan halaman yang pertama kali tampil ketika aplikasi anak dibuka. Dijelaskan pada Gambar 4.1 aplikasi ini statusnya tidak diketahui jika GPS ponsel dimatikan atau posisi anak berada didalam gedung ataupun penghalang yg menyebabkan satelit tidak bisa menemukan nilai *latitude longitude* anak.



Gambar 4.1. Halaman Utama Sebelum Menerima Sinyal GPS

4.2.1.2. Halaman Utama Setelah Menerima Sinyal GPS

Pada Gambar 4.2 menampil informasi *latitude longitude* lokasi anak dan setiap anak berubah posisi, nilai ini akan *update* setiap 30 detik. Halaman ini berfungsi mengirimkan *latitude longitude* lokasi anak ke server secara berkala sesuai dengan nilai *latitude longitude* yg tertera pada aplikasi,aplikasi ini akan tetap aktif menggunakan *service* (*Running in Background*).



Gambar 4.2. Halaman Utama Setelah Menerima Sinyal GPS

4.2.2. Aplikasi Orang tua

4.2.2.1.Halaman *Splash Screen*

Halaman *Splash Screen* pada Gambar 4.3 adalah halaman yang pertama muncul pada saat aplikasi orang tua dijalankan, halaman ini tampil selama 3 detik kemudian setelah itu akan langsung menuju halaman utama aplikasi orang tua, halaman ini sekaligus sebagai pengganti *loading* ketika aplikasi orang tua *request latitude longitude* lokasi anak ke sever dan juga mengambil lokasi orang tua saat itu melalui GPS *device* orang tua.

Gambar 4.3. Halaman *Splash Screen*

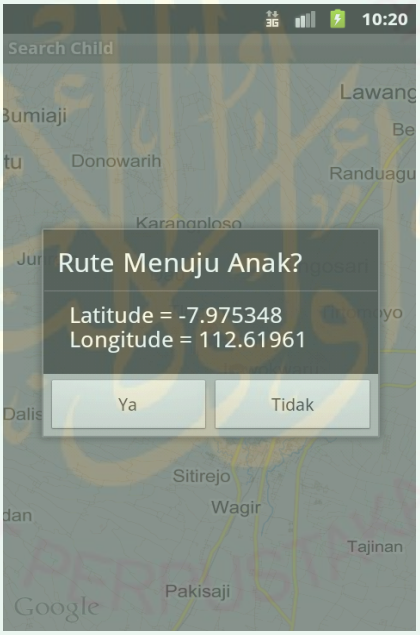
4.2.2.2.Halaman Utama Aplikasi Orang tua

Halaman utama merupakan halaman yang pertama kali tampil setelah halaman *Splash Screen*. Ditampilkan pada Gambar 4.4 halaman ini berfungsi menampilkan lokasi anak dan lokasi orang tua, jadi pada halaman utama ini akan tampil posisi anak yang di *parsing* dari server dan juga posisi orang tua.

Gambar 4.4. Halaman Utama Aplikasi Orang tua

4.2.2.3. Halaman Menu

Halaman ini tampil untuk memberitahukan kepada orang tua bahwa orang tua bisa memilih menggunakan menu rute untuk menggambarkan rute menuju lokasi anak, menu ini muncul ketika orang tua menyentuh *marker* anak maupun *marker* orang tua sendiri, maka nanti akan muncul notifikasi seperti yang ditampilkan pada Gambar 4.5.



Gambar 4.5. Halaman Menu

4.2.2.4. Halaman Rute

Gambar 4.6 menampilkan jalur rute menuju lokasi anak, sehingga orang tua tahu jalur rute terdekat menuju lokasi anak menggunakan algoritma *Dijkstra*, pada tampilan jalur rute di tunjukkan dengan garis berwarna Merah.

Gambar 4.6. Halaman Rute

```
//Posisi Bapak Berdasarkan GPS
GeoPoint geopoint = new GeoPoint((int) (latBapak * 1E6),
                          (int) (logBapak * 1E6));
//Parsing Posisi Anak dari Server
jObject = new JSONObject(xResultAnak);
JSONArray menuitemArray = jObject.getJSONArray("Posisi");
String latitudeAnak = menuitemArray.getJSONObject(0).getString(
                          "Latitude");
String longitudeAnak = menuitemArray.getJSONObject(0.getString(
                          "Longitude");
//titik node
int temp =2;
//relasi antar node
for (int i=0;i<anid.size();i+=2)
{
//Berangkat
if(i==0)
{
latx=Double.valueOf(anid.get(anid.size()-4));
lonx=Double.valueOf(anid.get(anid.size()-3));
GeoPoint endGP = new GeoPoint((int) (latx * 1E6),
(int) (lonx*E6));
Route route = getDirections(startGP, endGP);
mapView.getOverlays().add(routeOverlay);
}
//Perhitungan antar node
if (i%2==0&&i>2)
{
lat1=Double.valueOf(anid.get(anid.size()-(4+temp)));
lon1=Double.valueOf(anid.get(anid.size()-(3+temp))); temp +=2;

//diambil pada kondisi perhitungan terakhir
if (i==anid.size()2)
{
GeoPoint startGP2 = new GeoPoint((int)(latx*1E6),(int)(lonx*1E6));
GeoPoint endd = new GeoPoint ((int)(lattujuan*1E6),(int)(longtujuan*1E6));
Route route2 = getDirections(startGP2, endd);
mapView.getOverlays().add(routeOverlay2);
}
}
//Kondisi setiap perhitungan
else
{
GeoPoint startGP1 = new GeoPoint ((int)(latx*1E6), (int)(lonx*1E6));
latx=lat1;
lonx=lon1;
GeoPoint endGP1 = new GeoPoint ((int)(lat1*1E6), (int)(lon1*1E6));
Route route1 = getDirections(startGP1, endGP1);
RouteOverlay routeOverlay1 = new RouteOverlay(route1, Color.RED);
mapView.getOverlays().add(routeOverlay1);
}
}
```

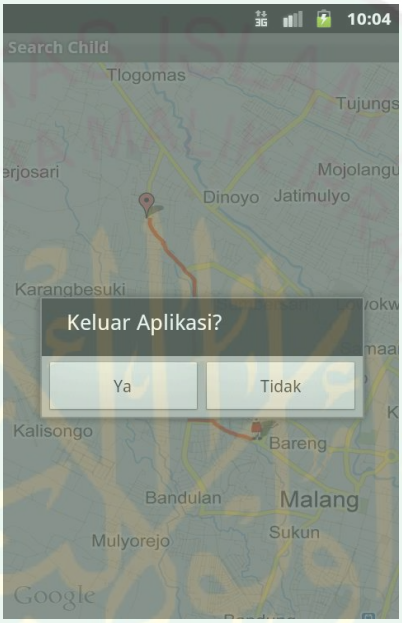
4.2.2.5. Tentang Program

Tentang program seperti yang tampil pada Gambar 4.7 berisi tentang info aplikasi cara menggunakan dan identitas pembuat termasuk maksud dari pembuatan aplikasi.

Gambar 4.7. Halaman Tentang Program

4.2.2.6. Notifikasi Keluar

Notifikasi keluar seperti ditampilkan pada Gambar 4.8 ini adalah notifikasi ketika *user* akan keluar aplikasi, notifikasi ini berfungsi menanyakan apakah *user* akan keluar dari aplikasi atau tidak.



Gambar 4.8. Notifikasi Keluar Aplikasi

4.3. Uji Coba Aplikasi

4.3.1. Uji Coba Alur Aplikasi

Sewaktu kita memulai aplikasi ini pastikan koneksi data dan GPS di *smartphone* aktif, dan pastikan *smartphone* berada pada tempat terbuka dan tidak terhalang oleh apapun, karena jika berada di tempat tertutup sepeti didalam gedung, maka GPS tidak akan menemukan *Latitude Longitude* dari *device*.

1. Pilih Aplikasi *Search Child*, dalam uji coba pada Gambar 4.9 aplikasi yang dimaksud ditandai dengan *icon* yang ada lingkaran merah.

Gambar 4.9. Uji Coba Tampilan Icon pada *Drawer*

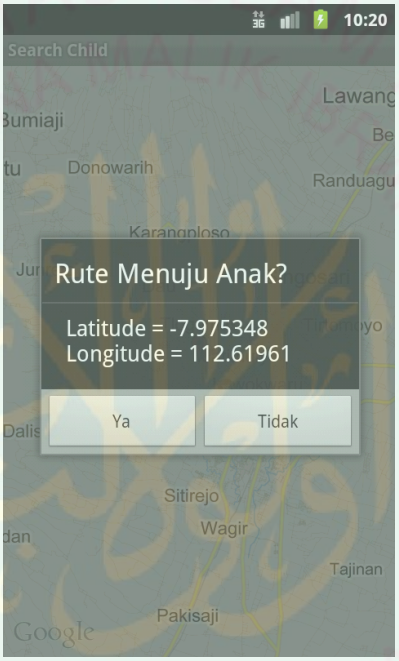
2. Kemudian setelah *Icon Search Child* dipilih maka akan muncul *splash screen* yang ditampilkan pada Gambar 4.10 yang berfungsi pengganti *loading*, karena pada saat *splash screen* muncul, aplikasi akan *request* posisi anak ke server dan *device* orang tua mendapatkan lokasi nya dari GPS pada *device* orang tua, nantinya *marker* untuk orang tua dan anak akan muncul pada halaman utama aplikasi.

Gambar 4.10. Uji Coba Tampilan *Splash Screen*

3. Kemudian setelah halaman *splash screen* akan muncul halaman utama yang tampilannya seperti pada Gambar 4.11 berisi informasi lokasi orang tua dan juga lokasi anak, jika pada halaman peta tidak muncul marker maka kemungkinan koneksi data atau GPS belum aktif.

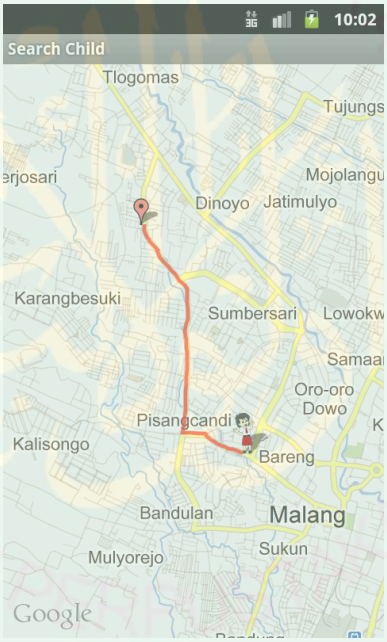
Gambar 4.11. Uji Coba Tampilan Halaman Utama

4. Selanjutnya jika orang tua menginginkan rute terdekat dari posisi orang tua ke anak, maka sentuh *marker* anak, kemudian muncul notifikasi sepetti pada Gambar 4.12, pilih "Ya" jika orang tua ingin mengetahui rute menuju anak atau tekan "Tidak" Jika orang tua tidak ingin mengetahui rute menuju lokasi anak.



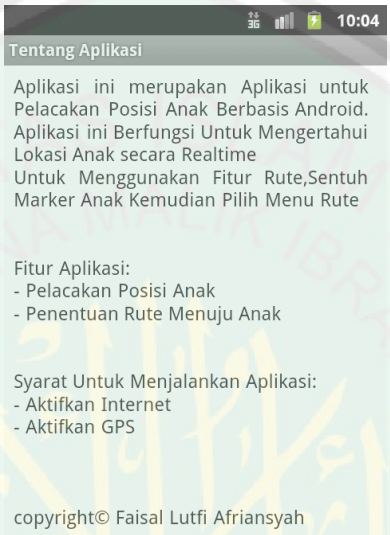
Gambar 4.12. Uji Coba Tampilan Halaman Menu Rute

5. Untuk uji coba ini pada Gambar 4.12 orang tua memilih "Ya" berarti orang tua ingin mengetahui rute terpendek menuju lokasi anak, setelah memilih "Ya", maka akan muncul rute dari lokasi orang tua, untuk waktu *loading* penentuan rute ini tergantung koneksi data, karena pada proses ini perlu koneksi internet untuk *parsing* data dari *database* untuk penentuan rutenya. kemudian setelah sukses akan muncul halaman seperti pada Gambar 4.13.



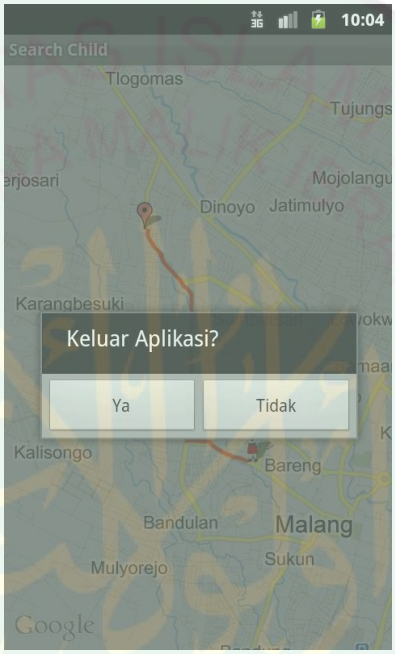
Gambar 4.13. Uji Coba Tampilan Rute Menuju Lokasi Anak

6. Selanjutnya adalah halaman info seperti pada Gambar 4.14, halaman ini muncul dengan menyentuh tombol menu dan memilih menu Info, Halaman ini berisi tentang info aplikasi dan cara menggunakan aplikasi.



Gambar 4.14. Uji Coba Tampilan Halaman Info

7. Selanjutnya jika orang tua ingin keluar dari aplikasi, pilih tombol menu kemudian pilih menu Keluar setelah ada *popup* pertanyaan seperti pada Gambar 4.15 pilih "Ya" untuk keluar dan pilih "Tidak" jika tetap ingin menggunakan aplikasi.



Gambar 4.15. Uji Coba Tampilan Halaman Menu

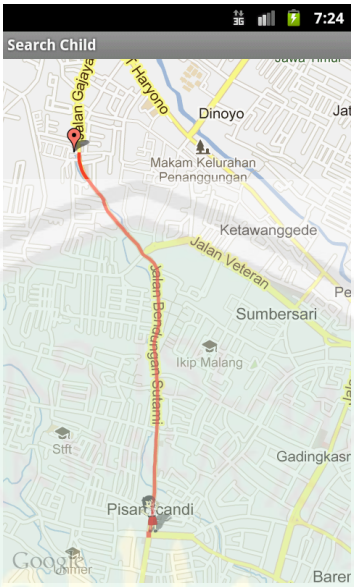
4.3.2. Uji Coba Penggunaan memori,Waktu dan Akurasi Algoritma *Dijkstra*

Pengujian memori untuk mendapatkan data memori yang digunakan pada setiap pengujian yang dilakukan. Konsumsi waktu, untuk mengetahui waktu yang dibutuhkan sistem mulai dari mendapatkan titik awal sampai dengan menemukan rute terpendek, akurasi algoritma *Dijkstra* untuk mengetahui perhitungan jarak data yang diproses.

Pada Uji Coba ini akan menguji 3 lokasi tujuan berbeda, Yaitu:

Tabel 4.1. Tabel Uji Coba 1

Uji Coba	**Awalan**		**Tujuan**	
	Latitude	**Longitude**	**Latitude**	**Longitude**
1	-7.950793	112.60864	-7.973173	112.613071
	Relasi Vertex : 1-3-4-5-6-7 = 2.727 Km			
	Relasi Vertex : 1-3-4-11-12-13-14-15-16-17-18-19-20-10-9-8-7=4.91 Km			
	Relasi Vertex Terpilih : 1-3-4-5-6-7 = 2.727 Km			
Penggunaan Memori	82MB			
Waktu	2 Menit 24 Detik			



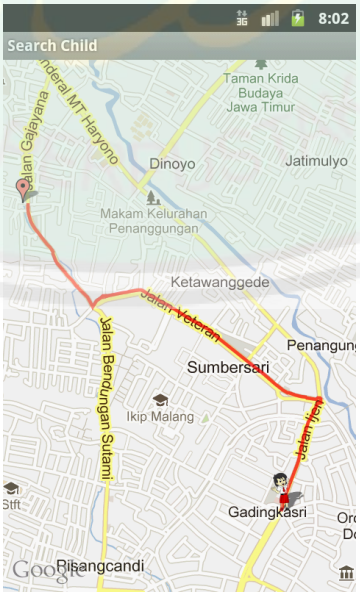
Gambar 4.16. Hasil Uji Coba 1

Pada Uji coba 1 ini lokasi awalan yaitu lokasi orang tua di kampus UIN Maulana Malik Ibrahim Malang, kemudian untuk lokasi tujuan yaitu lokasi anak ada di Jl.Galunggung, untuk relasi *vertex* nya ada 2 relasi *vertex* yang terpilih, relasi tersebut menentukan bobot panjang jarak dari lokasi awal menuju tujuan, pada uji coba 1 ini relasi *vertex* pertama dengan panjang 2,727 Km dan relasi *vertex* yang kedua dengan panjang 4,91 Km, untuk relasi *vertex* yang pertama jalur rute nya yaitu dari lokasi awal melalui Jl.Gajayana, kemudian lurus melalui Jl.Bedungan Sutami, kemudian lurus melalui Jl.Galunggung dan sampai pada lokasi tujuan, untuk relasi *vertex* kedua Jalur rute yang dilalui yaitu dari lokasi awal kemudian melalui Jl.Gajayana, Belok kiri melalui Jl.Veteran kemudian belok kanan melalui Jl.Ijen kemudian Belok kanan melalui Jl.Kawi Atas dan sampai pada lokasi tujuan, Untuk *Output* rute yang terpilih adalah relasi *vertex* pertama dengan panjang 2,72 Km dengan hasil seperti pada Gambar 4.16. Sedangkan

konsumsi memori untuk uji coba 1 ini adalah 82 MB dengan waktu mulai dari mendapatkan titik awal sampai dengan menemukan rute terpendek selama 2 menit 24 detik.

Tabel 4.2. Tabel Uji Coba 2

Uji Coba	Awalan		Tujuan	
	Latitude	Longitude	Latitude	Longitude
2	-7.950793	112.60864	-7.968617	112.623661
	Relasi Vertex : 1-3-4-5-6-7-8-9-10-20-19-18-17-16 = 4.62 Km			
	Relasi Vertex : 1-3-4-11-12-13-14-15-16 = 3.34 Km			
	Relasi Vertex Terpilih : 1-3-4-11-12-13-14-15-16 = 3.34 Km			
Penggunaan Memori	89MB			
Waktu	2 Menit 40 Detik			

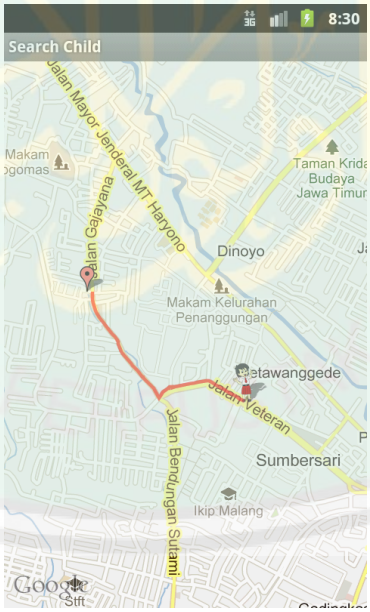


Gambar 4.17. Hasil Uji Coba 2

Pada Uji coba 2 ini lokasi awalan yaitu lokasi orang tua di kampus UIN Maulana Malik Ibrahim Malang,kemudian untuk lokasi tujuan yaitu lokasi anak ada di Jl. Ijen, untuk relasi *vertex* nya ada 2 relasi *vertex* yang terpilih, relasi tersebut menentukan bobot panjang jarak dari lokasi awal menuju tujuan, pada uji coba 2 ini relasi *vertex* pertama dengan panjang 4,62 Km dan relasi *vertex* yang kedua dengan panjang 3,34 Km, untuk relasi *vertex* yang pertama jalur rute nya yaitu dari lokasi awal melalui Jl.Gajayana, kemudian lurus melalui Jl.Bedungan Sutami, kemudian lurus melalui Jl.Galunggung, kemudian belok kiri melalui Jl.Kawi atas, kemudian belok kiri melalui Jl.Ijen dan sampai pada lokasi tujuan, untuk relasi *vertex* kedua Jalur rute yang dilalui yaitu dari lokasi awal kemudian melalui Jl.Gajayana, Belok kiri melalui Jl.Veteran kemudian belok kanan melalui Jl.Ijen dan sampai pada lokasi tujuan. Untuk *Output* rute yang terpilih adalah relasi *vertex* kedua dengan panjang 3,34 Km dengan hasil seperti pada Gambar 4.17. Sedangkan konsumsi memori untuk uji coba 2 ini adalah 89 MB dengan waktu mulai dari mendapatkan titik awal sampai dengan menemukan rute terpendek selama 2 menit 40 detik.

Tabel 4.3. Tabel Uji Coba 3

Uji Coba	Awalan		Tujuan	
	Latitude	Longitude	Latitude	Longitude
3	-7.950793	112.60864	-7.968617	112.623661
	Relasi Vertex : 1-3-4-5-6-7-8-9-10-20-19-18-17-16-15-14-13 = 6.96			
	Relasi Vertex : 1-3-4-11-12-13 = 1.46			
	Relasi Vertex Terpilih : 1-3-4-11-12-13 = 1.46			
Penggunaan Memori	78MB			
Waktu	2 Menit 13 detik			



Gambar 4.18. Hasil Uji Coba 3

Pada Uji coba 3 ini lokasi awalan yaitu lokasi orang tua di kampus UIN Maulana Malik Ibrahim Malang, kemudian untuk lokasi tujuan yaitu lokasi anak

ada di Jl. Veteran, untuk relasi *vertex* nya ada 2 relasi *vertex* yang terpilih, relasi tersebut menentukan bobot panjang jarak dari lokasi awal menuju tujuan, pada uji coba 3 ini relasi *vertex* pertama dengan panjang 6,96 Km dan relasi *vertex* yang kedua dengan panjang 1,46 Km, untuk relasi *vertex* yang pertama jalur rute nya yaitu dari lokasi awal melalui Jl.Gajayana,kemudian lurus melalui Jl.Bedungan Sutami, kemudian lurus melalui Jl.Galunggung, kemudian belok kiri melalui Jl.Kawi atas,kemudian belok kiri melalui Jl.Ijen kemudian belok kiri melalui Jl.Veteran dan sampai pada lokasi tujuan, untuk relasi *vertex* kedua Jalur rute yang dilalui yaitu dari lokasi awal kemudian melalui Jl.Gajayana, Belok kiri melalui Jl.Veteran dan sampai pada lokasi tujuan. Untuk *Output* rute yang terpilih adalah relasi *vertex* kedua dengan panjang 1,46 Km dengan hasil seperti pada Gambar 4.18. Sedangkan konsumsi memori untuk uji coba 3 ini adalah 78 MB dengan waktu mulai dari mendapatkan titik awal sampai dengan menemukan rute terpendek selama 2 menit 13 detik.

4.3.3. Hasil Uji Coba pada Responden

Jumlah pengguna yang turut serta dalam uji coba aplikasi adalah 50 orang, aplikasi yang di uji coba yaitu aplikasi untuk orang tua dan aplikasi untuk anak, hal ini dilakukan untuk menguji apakah aplikasi anak tersebut memang benar bisa berjalan sebagaimana mestinya. Setelah dilakukan ujicoba secara acak pada aplikasi orang tua dan juga aplikasi anak maka dilakukan analisis hasil untuk mengetahui tingkat akurasi dari aplikasi yang dibangun.

Tabel 4.4.Pengolahan Data Kuisioner Aspek Komunikasi Visual

No.	Komponen Penilaian	Nilai					
		Baik		Cukup		Kurang	
		F	%	F	%	F	%
1.	GUI	6	12	33	66	11	22
2.	Komunikatif	4	8	46	92	0	0
3.	Kemudahan Pengguna	19	38	31	62	0	0
Rata-rata		9,67	19,33	36,67	73,3	1,22	7,33

Tabel 4.5.Pengolahan Data Kuisioner Aspek Perangkat Lunak

No.	Komponen Penilaian	Nilai					
		Baik		Cukup		Kurang	
		F	%	F	%	F	%
1.	Kepraktisan	10	20	40	80	0	0
2.	Alur program	21	42	29	58	0	0
Rata-rata		10,33	31	34,5	69	0	0

Tabel 4.6.Pengolahan Data Kuesioner Aspek Fungsi Pelacakan

Komponen Penilaian	Nilai					
	Baik		Cukup		Kurang	
	F	%	F	%	F	%
Relevansi Aplikasi terhadap Pelacakan Anak	24	48	23	46	3	6

Keterangan: F = Frekuensi, % = Persentase

Dari hasil uji coba kepada 50 responden tersebut dapat disimpulkan bahwa hasil penilaian produk terhadap aplikasi pelacakan lokasi anak menunjukkan penilaian yang baik terhadap aspek komunikasi visual, aspek perangkat lunak dan aspek desain pembelajaran. Oleh karena itu, aplikasi pelacakan lokasi Anak ini layak untuk diimplementasikan. Pelacakan lokasi anak juga dapat membantu mempermudah kontrol orang tua terhadap anak.

4.4. Aplikasi Pelacakan Anak dalam pandangan Islam

Rancang bangun aplikasi *Mobile* untuk mengetahui lokasi anak yang telah dibuat oleh penulis membantu orang tua dalam menjaga anaknya yang awalnya

masih konvensional dengan Telepon dan SMS, menjadi lebih modern dengan bersentuhan teknologi digital yang secara *Realtime* terus melaporkan lokasi putra dan putrinya secara berkala. Beberapa manfaat nya, yaitu:

a. Dapat digunakan dimana- pun pada waktu kapan- pun.
b. Harga *device* bergerak memiliki harga yang relatif lebih murah di banding harga GPS *Tracker* yang memang didesain untuk pelacakan.
c. Ukuran perangkat yang praktis, kecil, ringan dan kebanyakan orang dewasa maupun anak-anak memiliki perangkat tersebut, sehingga mudah dibawa kemana- mana.

Seorang ayah bersama seorang ibu harus bekerja sama untuk menunaikan tanggung jawab terhadap anak, baik di dalam maupun di luar rumah. Anak harus terus mendapatkan pengawasan di mana saja mereka berada, dijauhkan dari teman yang jelek dan teman yang rusak. Anak diperintahkan untuk mengerjakan yang ma'ruf dan dilarang dari mengerjakan yang mungkar.

Allah Berfirman dalam surat At Tahrim ayat : 6

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ قُوٓاْ أَنفُسَكُمۡ وَأَهۡلِيكُمۡ نَارٗا وَقُودُهَا ٱلنَّاسُ وَٱلۡحِجَارَةُ عَلَيۡهَا مَلَٰٓئِكَةٌ غِلَاظٞ شِدَادٞ لَّا يَعۡصُونَ ٱللَّهَ مَآ أَمَرَهُمۡ وَيَفۡعَلُونَ مَا يُؤۡمَرُونَ ٦

Artinya : *Hai orang-orang yang beriman, peliharalah dirimu dan keluargamu dari api neraka yang bahan bakarnya adalah manusia dan batu; penjaganya malaikat-malaikat yang kasar, keras, dan tidak mendurhakai Allah terhadap apa yang diperintahkan-Nya kepada mereka dan selalu mengerjakan apa yang diperintahkan.*

Jika kita mengingat lagi hadist Nabi Muhammad SAW yang diriwayatkan oleh Ath thusi,Bukhori dan Muslim.

Seorang datang kepada Nabi Muhammad SAW dan bertanya,"Ya Rasulullah, apa hak anakku ini?" Nabi Muhammad SAW mejawab, "Memberinya nama yang baik,mendidik adab yang baik dan memberinya kedudukan yang baik." (HR.Aththusi)

Bertaqwalah kepada Allah dan berlakulah adil terhadap anak anak mu (HR.Bukhari dan Muslim)

Rasulullah SAW ditanya tentang peranan kedua orang tua,beliau lalu menjawab,"Mereka adalah (yang menyebabkan) surgamu atau nerakamu." (HR.Ibnu Majah)

Maka mulai sekarang hendaknya para orang tua sadar terhadap kewajiban mereka untuk mendidik istri dan anak nya agar menjadi hamba Allah SWT yang taat. Memilihkan pendidikan anak yang kondusif untuk perkembangan iman dan otaknya. Bukannya membiarkan anak-anak mereka begitu saja tanpa pengawasan terhadap bacaan yang mereka gemari, apa saja yang suka mereka saksikan dan aktivitas yang mereka gandrungi. Kelalaian dalam hal ini, berarti penyia-nyiaan terhadap amanat Allah.

BAB V

PENUTUP

5.1. Kesimpulan

Dari pendefinisian masalah serta analisa dan pembuatan aplikasi ini, dapat diambil kesimpulan bahwa dari hasil uji coba, Rancang bangun aplikasi *Mobile* untuk mengetahui lokasi anak menggunakan Algoritma *Dijkstra* ini dapat di implementasikan guna membantu orang tua untuk mengetahui lokasi anaknya secara *realtime* dan periodik. Dalam aplikasi pencarian lokasi anak ini terdapat kelemahan, karena Algoritma *Dijkstra* dapat digunakan untuk perhitungan pada wilayah yang telah terdefinisi sehingga ketika *node* dan bobot jarak antar *node* tidak terdefinisi, maka Algoritma *Dijkstra* tidak dapat berjalan sebagaimana mestinya.

Berdasarkan *device* yang telah di uji coba,apikasi dapat berjalan dengan baik pada semua *device*, tergantung koneksi data internetnya. Dari hasil pengujian terhadap 50 responden dapat disimpulkan bahwa penilaian aplikasi Pelacakan Lokasi Anak menunjukkan penilaian cukup terhadap aspek komunikasi visual dengan persentase 73,3%, Sedangkan dari aspek perangkat lunak dinilai cukup dengan persentase 69%, dan aspek relevansi nya dinilai cukup dengan persentase 48%.

5.2. Saran

1. Perlu diadakan penelitian lebih lanjut untuk memaksimalkan Fungsi kecepatan pada saat menampilkan peta dan tambahan fitur lain untuk aplikasi pelacakan lokasi anak ini.
2. Perlu pengkajian lanjutan untuk penggunaan Algoritma yang lebih sesuai.

DAFTAR PUSTAKA

Bramantya, Amirullah Andi. 2012. *aplikasi Location Based Service untuk menentukan rute terpendek lokasi atm di kota malang menggunakan Algoritma Djikstra*. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang.

DiMarzio,J.F, .2008..*Android a Programmer's Guide*.New York: McGraw-Hill Companies.

Faiz Almath,Muhammad. 2012.*1100 Hadits Terpilih*.Depok:Gema Insani

Gabriel,Svennerberg. 2010..*Beggining Google Maps API 3*. New York: Apress.

Haseman,Chris. 2008..*Android Esential.New*.New York:Apress

Hakim,Muhammad Amrin. 2011. Monitoring Lokasi Anak Menggunakan Handphone ber-GPS, Skripsi, Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember.

Junaedi,Moh. 2008.*Pengantar XML*.Jakarta:IlmuKomputer.com

Jungnikel, D. 2005. G*raphs, Network and Algorithms*. Springer-Verlag

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

Meier,Reto. 2010.*Professional Android 2 ApplicationDevelopment*. Indianapolis: Wiley Publishing, Inc.

Munir, Rinaldi. (2005*). Matematika Diskrit edisi Ketiga*,Bandung: Informatika

Parkinson, B.W. (1996), *Echosounder : Theory and Applications, chap. 1: Introduction and Heritage of NAVSTAR, the Global Positioning System. pp. 3*-28, American Institute of Aeronautics and Astronautics, Washington, D.C.

Riftadi, Mohammad. 2007. *Variasi Penggunaan Fungsi Heuristik Dalam. Pengaplikasian Algoritma A**, Makalah,Institut Teknologi Bandung

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

Wahid, Fathul. 2004. *Dasar-dasar Algoritma dan Pemrograman.* Yogyakarta:Andi

http://gs.statcounter.com/#mobile_os-ww-monthly-201202-201302,diakses pada tanggal 28 Juni 2012 pukul 12.33

LAMPIRAN

Kuesioner

No	Pertanyaan
1.	Menurut anda, apakah aplikasi ini mudah untuk dijalankan ? a. sangat mudah b. mudah c. biasa
2.	Menurut anda, bagaimana tampilan antar muka dari aplikasi ini ? a. sangat menarik b. menarik c. biasa
3.	Menurut anda, apakah aplikasi ini membantu dalam proses pelacakan ? a. sangat membantu b. membantu c. Kurang Membantu
4.	Menurut anda, bagaimana penilaian keseluruhan anda terhadap aplikasi ini? a. Sangat Baik b. Baik c. Biasa
5.	Seberapa mudah anda dapat mempelajari aplikasi ini? a. Sangat Mudah b. Mudah c. Sulit
6.	Seberapa Sulit anda menggunakan aplikasi ini? a. Sangat Mudah b. Mudah c. Sulit