APLIKASI PENCARIAN RUTE MASJID TERDEKAT DI KOTA MALANG BERBASIS ANDROID

Moh. Sunaryo[1], Yuri Ariyanto[2], Ely Setyo Astuti[3]

Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang
1mohsunaryo99@gmail.com,2yuriariyanto@polinema.ac.id,3elysetyoastuti@polinema.ac.id

Abstrak

Malang sebagai kota pariwisata dan kota pendidikan memiliki banyak obyek wisata dan universitas–universitas negeri yang terkenal. Hal ini menyebabkan banyaknya pendatang dari luar Kota Malang yang datang untuk menimba ilmu atau sekedar berkunjung ke Malang. Pengunjung yang datang ke Malang tersebut mayoritas beragama Islam.

Namun permasalahan yang datang adalah untuk orang yang baru pertama kali ke Malang seringkali kebingungan menemukan masjid terdekat untuk melaksanakan sholat. Tetapi tidak menutup kemungkinan juga untuk yang sering ke Malang masih kebingungan. Dikarenakan masjid yang biasanya dituju masih jauh sedangkan waktu sholat hampir berakhir dan jalanan di Malang sering kali terjadi kepadatan lalu lintas.

Dalam penelitian ini menggunakan Algoritma A*(A Star), aplikasi ini diharapkan dapat membantu pencarian masjid terdekat dengan memberikan rute terpendek. Sehingga memudahkan pengguna dalam pencarian masjid terdekat untuk segera menunaikan sholat. Berdasarkan hasil uji yang telah dilakukan dalam penerapan Algoritma A* untuk pencarian rute terpendek, hasil akurasi yang didapatkan adalah sebesar 80%.

Kata Kunci : Kota Malang, Rute Terpendek, Masjid Terdekat, Algoritma A* (A Star)

1. Pendahuluan

Malang sebagai kota pariwisata dan kota pendidikan memiliki banyak obyek wisata dan universitas–universitas negeri yang terkenal. Hal ini menyebabkan banyaknya pendatang dari luar Kota Malang yang datang untuk menimba ilmu atau sekedar berkunjung ke Malang. Pengunjung yang datang ke Malang tersebut mayoritas beragama Islam.

Masjid merupakan salah satu sarana ibadah untuk umat Islam. Namun permasalahan yang datang adalah untuk orang yang baru pertama kali ke Malang seringkali kebingungan menemukan masjid terdekat untuk melaksanakan sholat. Tetapi tidak menutup kemungkinan juga untuk yang sering ke Malang masih kebingungan. Dikarenakan masjid yang biasanya dituju masih jauh sedangkan waktu sholat hampir berakhir dan jalanan di Malang sering kali terjadi kepadatan lalu lintas.

Kemajuan teknologi informasi pada perangkat mobile yang dipadukan dengan GPS (Global Positioning System) memudahkan pengguna untuk mengetahui lokasi. Dengan adanya teknologi API Google Maps, dapat menampilkan gambar peta secara online. Hal ini akan sangat membantu untuk mengetahui lokasi pengguna dan lokasi masjid yang akan dituju. Sehingga pengguna dapat segera menuju lokasi masjid tersebut.

Berawal dari permasalahan tersebut, penelitian ini akan dibuat sebuah aplikasi informasi pencarian rute terpendek untuk menemukan masjid terdekat di kota Malang berbasis Android. Dengan menggunakan Algoritma A*(A Star), aplikasi ini diharapkan dapat membantu pencarian masjid terdekat dengan memberikan rute terpendek. Sehingga memudahkan pengguna dalam pencarian masjid terdekat untuk segera menunaikan sholat.

2. Landasan Teori

2.1 Android

Android adalah sistem operasi untuk perangkat mobile berbasis Linux yang mencakup sistem operasi, middleware, dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat piranti lunak untuk ponsel. kemudian dalam pengembangan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance mendukung pengembangan open source pada perangkat mobile. Google juga merilis kode-kode Android di bawah lisensi Apache yaitu lisensi perangkat lunak dan open platform perangkat seluler. (Safaat H:2011)

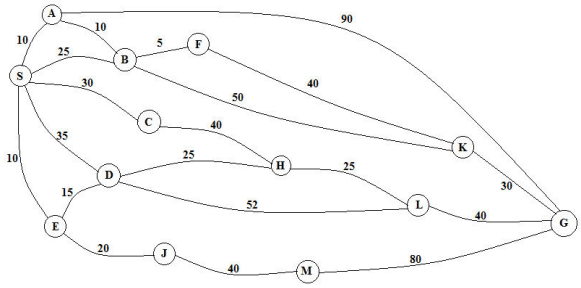
2.2 Algoritma A* (A Star)

Algoritma ini merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best-First Search.* Jarak yang diperhitungkan didapat dari jarak sebenarnya

(*actual cost*) ditambah dengan jarak perkiraan (*estimated cost*). (Rusel, Stuart dan Norvig, Peter: 1995) Dalam notasi matematika dituliskan sebagai:

$f(n) = g(n) + h(n)$. (1)

Dengan perhitungan jarak seperti ini, algoritma A* adalah *complete* dan optimal. Misalkan terdapat 13 kota yang dinyatakan oleh simpul-simpul dalam suatu *graph* dua arah. Setiap angka pada busur menyatakan jarak sebenarnya (*actual cost*) antara satu kota dengan yang lainnya. Nilai $h(n)$ adalah fungsi heuristik, yaitu jarak garis lurus dari simpul n menuju simpul G dalam satuan kilometer.



Gambar 1. Contoh kasus algoritma A*

Berikut ini adalah tabel nilai $h(n)$ yang menyatakan jarak perkiraan dari setiap simpul-simpul menuju simpul G :

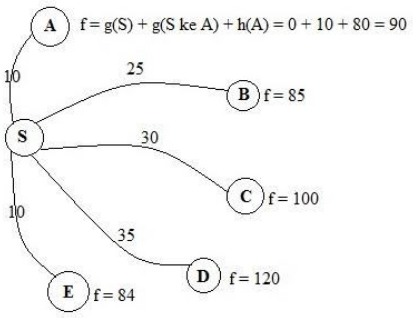
Tabel 1. Nilai h(n) contoh kasus algoritma A*

n	S	A	B	C	D	E
h(n)	80	80	60	70	85	74

F	G	H	J	K	L	M
70	0	40	100	30	20	70

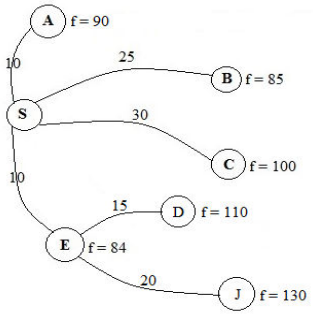
Langkah-langkah pencarian rute berdasarkan algoritma tersebut adalah sebagai berikut:

1. Langkah pertama, karena di *OPEN* hanya terdapat satu simpul (S), maka S terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Kemudian bangkitkan semua suksesor S, yaitu: A, B, C, D ,E. Karena kelima suksesor tidak ada di *OPEN* maupun *CLOSED*, maka kelimanya dimasukkan ke *OPEN*. Menghasilkan *OPEN* = [A,B,C,D,E] dan *CLOSED* = [S].



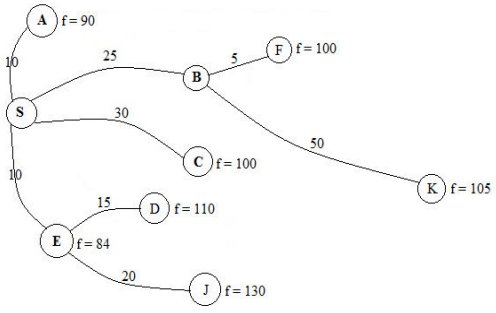
Gambar 2. Langkah pertama contoh kasus algoritma A*

2. Langkah kedua, E dengan biaya terkecil (yaitu 84) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Selanjutnya, semua suksesor E dibangkitka, yaitu: D dan J. Karena belum pernah ada di *OPEN* maupun *CLOSED*, maka J dimasukkan ke *OPEN*. Sedangkan simpul D sudah ada di *OPEN*, maka harus dicek apakah *parent* dari D perlu diganti atau tidak, Ternyata, jarak dari S ke D melalui E (yaitu 10 + 15 = 25) lenih kecil daripada jarak dari S ke D (yaitu 35). Oleh karena itu, *parent* dari D harus diubah, yang semula S menjadi E. Dengan perubahan *parent* ini, maka nilai g dan f pada D juga diperbarui (nilai g yang semula 35 menjadi 35, dan nilai f dari 120 menjadi 110). Langkah kedua ini menghasilkan *OPEN* = [A,B,C,D,J] dan *CLOSED* = [S,E].



Gambar 2. Langkah kedua kasus algoritma A*

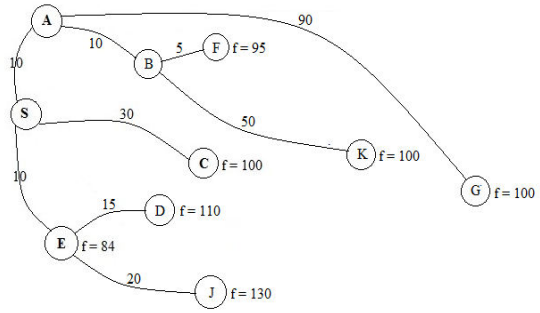
3. Langkah ketiga, B dengan jarak terkecil (yaitu 85) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Selanjutnya, semua suksesor B dibangkitkan, yaitu: A, F dan K. Karena belum pernah ada di *OPEN* maupun *CLOSED*, maka F dan K dimasukkan ke *OPEN*. Sedangkan simpul A sudah ada di *OPEN*, maka harus dicek apakah *parent* dari A perlu diganti atau tidak. Ternyata, jarak dari S ke A melalui B (yaitu 25 + 10 = 35) lebih besar daripada jarak dari S ke A (yaitu 10). Oleh karena itu, *parent* dari A tidak perlu diubah (tetap S), Akhir dari langkah ketiga ini menghasilkan *OPEN* = [A,C,D,F,J,K] dan *CLOSED* = [S,E,B].



Gambar 3. Langkah ketiga kasus algoritma A*

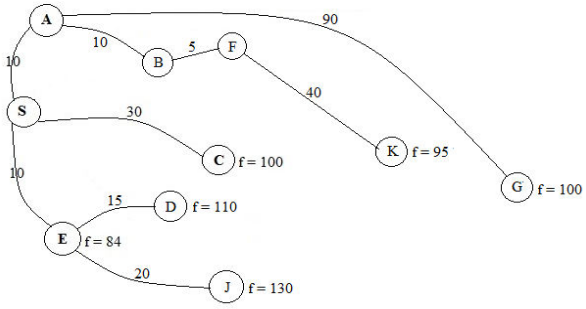
4. Langkah keempat, A dengan jarak terkecil (yaitu 90) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Selanjutnya,

semua suksesor A dibangkitkan, yaitu : B dan G. Karena belum pernah ada di *OPEN* maupun *CLOSED*, maka G dimasukkan ke *OPEN*. Sedangkan simpul B sudah ada di *CLOSED*, maka harus dicek apakah *parent* dari B perlu diganti atau tidak. Ternyata, jarak dari S ke B melalui A (yaitu 10 + 10 = 20) lebih kecil daripada jarak dari S ke B (yaitu 25). Oleh karena itu, *parent* dari B harus diubah, yang semula S menjadi A, nilai *g* dan *f* pada B juga harus diperbarui (nilai *g* yang semula 25 menjadi 20, dan nilai *f* dari 85 menjadi 80). Dalam kasus ini, B hanya mempunyai dua jalur, yaitu F dan K. Nilai *g*(F) yang semula 30 diubah menjadi 25, dan nilai *f*(F) dari 100 menjadi 95. Nilai *g*(K) yang semula 75 diubah menjadi 70, dan nilai *f*(K) dari 105 menjadi 100. Akhirnya, OPEN = [C,D,F,G,J,K] dan CLOSED = [S,E,B,A].



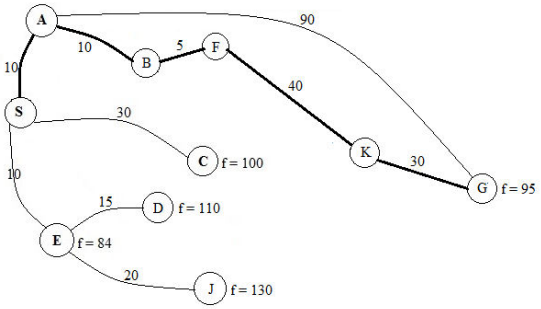
Gambar 4. Langkah keempat kasus algoritma A*

5. Langkah kelima, F dengan jarak terkecil (yaitu 95) terpilih sebagai *BestNode* dan dipindahkan ke CLOSED. Selanjutnya, semua suksesor F dibangkitkan, yaitu: K. Karena K sudah ada di OPEN, maka harus dicek apakah *parent* dari K perlu diganti atau tidak. Jarak dari S ke K melalui F ternyata lebih kecil daripada jarak dari S ke K melalui *parent* lama (B). Oleh karena itu, *parent* dari K harus diubah, yang semula B menjadi F. Selanjutnya, nilai *g*(K) yang semula 70 diubah menjadi 65, dan nilai *f*(K) dari 100 menjadi 95. Akhirnya, OPEN = [C,D,F,G,J,K] dan CLOSED = [S,E,B,A,F].



Gambar 5. Langkah kelima kasus algoritma A*

6. Langkah keenam, K dengan jarak terkecil (yaitu 95) terpilih sebagai *BestNode* dan dipindahkan ke CLOSED. Selanjutnya, semua suksesor K dibangkitkan, yaitu: G. Karena D sudah ada di OPEN, maka harus dicek apakah *parent* dari G perlu diganti atau tidak. Jarak dari S ke G melalui K ternyata lebih kecil daripada jarak S ke G melalui *parent* lama (A). Oleh karena itu, *parent* dari G harus diubah, yang semula A menjadi K. Selanjutnya, nilai *g*(G) yang semula 100 diubah menjadi 95, dan nilai *f*(G) dari 100 menjadi 95. Pada akhir langkah keenam ini, OPEN = [C,D,G,J] dan CLOSED = [S,E,B,A,F,K].



Gambar 6. Langkah keenam kasus algoritma A*

Selanjutnya, G dengan jarak terkecil (yaitu 95) terpilih sebagai *BestNode*. Karena *BestNode* sama dengan *goal*, berarti solusi sudah ditemukan. Rute dan total jarak bisa ditelusuri balik dari G menuju S karena setiap simpul hanya memiliki satu *parent* dan setiap simpul memiliki informasi jarak sebenarnya (*g*). Penelusuran menghasilkan rute S-A-B-F-K-G dengan total jarak sama dengan 95. (Suyanto:2014)

2.3 Google Map

Google Maps adalah sebuah jasa peta virtual gratis yang disediakan oleh Google. Google Maps menawarkan peta yang dapat diseret dan gambar satelit untuk seluruh dunia, juga menawarkan perencana rute dan pencari letak bisnis. Kita dapat menambahkan fitur Google Maps dalam web yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan Google Maps API. Google Maps API adalah suatu library yang berbentuk JavaScript. API sendiri adalah singkatan dari Application Progrmming Interface.

Perlu diketahui bahwa perkembangan penggunan Gogle Map di Android yang ada saat ini dimulai dengan adanya Google Map V1, yang penggunaannya telah dihentikan pada akhir tahun 2012. Mulai tahun 2013, aplikasi Android yang ingin menampilkan Google Map harus menggunakan layanan Google Map V2. Mulai dari penggunaan SHA1 yang menggantikan MD5 untuk mendapatkan Google API Key hingga penggunaan Fragment yang menggantikan MapView. Kita juga

harus menginstal library google-play-service terlebih dahulu pada Android SDK. (Mufti:2015)

3. Pembahasan

3.1 Analisa Sistem

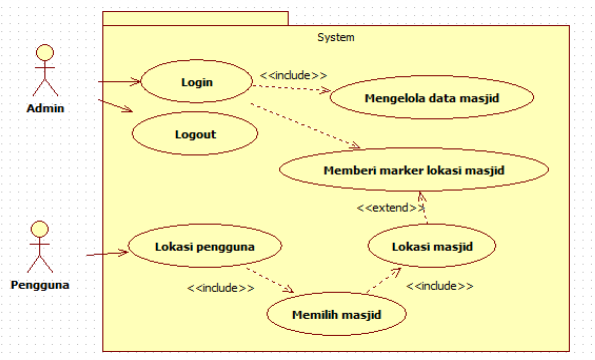
Analisa sistem merupakan tahapan yang akan menentukan benar atau tidaknya langkah selanjutnya dalam pembuatan aplikasi. Tujuan dari analisa sistem antara lain adalah untuk mempelajari aktivitas sistem untuk mendapatkan gambaran yang menyeluruh tentang sistem yang sedang berjalan dan permasalahan yang terjadi serta analisa dari algoritma yang digunakan.

3.2 Perancangan Sistem

Dalam perancangan sistem Aplikasi Informasi Pencarian Rute Terpendek Untuk Menemukan Masjid Terdekat Di Kota Malang Berbasis Android dapat menggunakan berbagai model untuk menggambarkan alur proses aplikasi. Model yang digunakan seperti berikut:

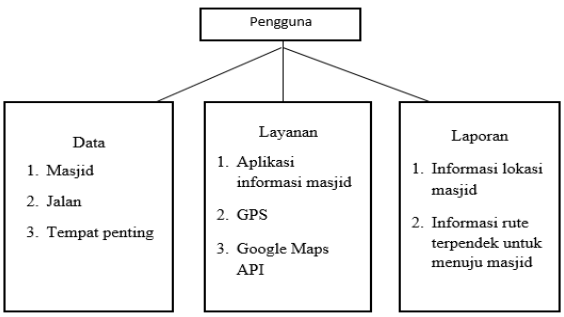
3.2.1 *Use Case*

Use case diagram adalah diagram yang menyajikan interaksi antara usecase dan aktor. Dimana aktor dapat berupa orang, peralatan atau sistem lain yang berinteraksi dengan sistem yang sedang dibangun. *Use case* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai.



Gambar 8. Diagram *Use Case*

3.2.2 WBS (*Work Breakdown Structure*)



Gambar 9. WBS (*Work Breakdown Structure*)

Pengguna mendapat koordinat lokasinya dari layanan GPS, kemudian sistem akan menampilkan peta yang mengambil dari API Google Maps. Pengguna juga dapat menginputkan tempat penting sebagai lokasi awal untuk menuju masjid. Hasil output dari sistem akan menampilkan informasi lokasi masjid dan rute terpendek untuk menuju lokasi tersebut.

4. Implementasi

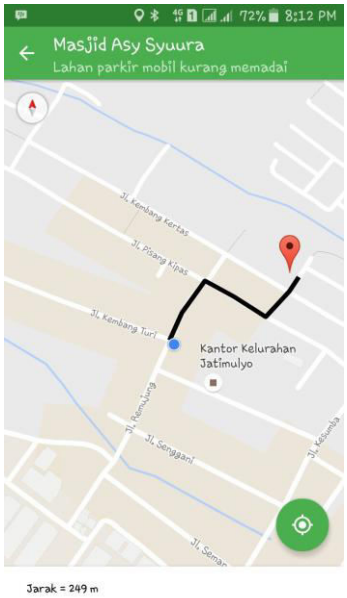
4.1 Implementasi Algoritma A*

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (starting point) menuju simpul tujuan dengan memperhatikan jarak (F) terkecil. Algoritma *A** (*A Star*) memperhitungkan *cost* dari *current state* ke tujuan denga fungsi heuristik, Algoritma ini juga mempertimbangkan *cost* yang telah ditempuh selama ini dari *initial state* ke *current state*. Jadi jika ada jalan yang telah ditempuh sudah terlalu panjang dan ada jalan lain yang *cost*-nya lebih kecil tetapi memberikan posisi yang sama dilihat dari *goal*, jalan yang lebih pendek yang akan dipilih. Berikut adalah langkah dalam melakukan perhitungan Algoritma *A** (*A Star*):

1. Masukan node awal ke openlist
2. *Loop* langkah – langkah di bawah ini :
   a. Cari *node* (*n*) dengan nilai *f*(*n*) yang paling rendah dalam *open list*. *Node* ini sekarang menjadi *current node*.
   b. Keluarkan *current node* dari *open list* dan masukan *ke close list.*
   c. Untuk setiap tetangga dari *current node* lakukan berikut :
   - Jika tidak dapat dilalui atau sudah ada dalam *close list*, abaikan.
   - Jika belum ada di *open list* . Buat *current node parent* dari *node* tetangga ini. Simpan nilai *f*,*g* dan *h* dari *node* ini.
   - Jika sudah ada di *open list*, cek bila *node* tetangga ini lebih baik, menggunakan nilai *g* sebagai ukuran. Jika lebih baik ganti *parent* dari *node* ini di *open list* menjadi *current node*, lalu kalkulasi ulang nilai *g* dan *f* dari *node* ini.

   d. Hentikan *loop* jika :
   - *Node* tujuan telah ditambahkan ke *open list*, yang berarti rute telah ditemukan.
   - Belum menemukan *node goal* sementara *open list* kosong atau berarti tidak ada rute.
3. Simpan rute. Secara '*backward*', urut mulai dari *node goal* ke *parent*-nya terus sampai mencapai *node* awal sambil menyimpan *node* ke dalam sebuah *array*.

Hasil rute dan jarak pada sistem ditunjukkan pada Gambar 10.



Gambar 10. Implementasi Algoritma A*

4.2 Hasil Uji Coba Akurasi Algoritma A*

Uji coba yang dilakukan adalah dengan membandingkan hasil perhitungan manual dengan hasil perhitungan sistem. Diambil 20 data *sample* dari data masjid yang ada untuk diuji. Titik awal yang digunakan berada di Jalan Kembang Turi berdasarkan dari koordinat lokasi GPS penulis. Berikut adalah tabel hasil uji coba tersebut:

Tabel 2. Hasil Perhitungan Manual dan Sistem

No	Nama Masjid	Manual	Sistem
1	Masjid Al Muqorrobun	260	257
2	Masjid Asy Syuura	350	342
3	Masjid Al Ikhlas	450	543
4	Masjid Nurul Huda	400	356
5	Masjid Al Muhajirin Griya Santa	1200	1200
6	Masjid An Nur Polinema	350	279
7	Masjid Al Muhajirin	800	798
8	Masjid Nurul Jannah	1800	1700
9	Masjid Darul Muttaqin	2200	2200
10	Masjid Al Ghifari	2100	1300
11	Masjid Baabul Jannah	1500	1300
12	Masjid Raden Patah	1500	1300
13	Masjid Al Istiqomah	1700	1500
14	Masjid Jami H. Agus Salim	3000	3000
15	Masjid Sabilillah	4200	4300
16	Masjid Al Firdaus	3400	3400
17	Masjid Al Hikmah UM	3700	3200
18	Masjid Jami	5600	5400
19	Masjid Al Huda Embong Arab	6500	5900
20	Masjid Jami Manarul Huda	2100	2300

Kolom yang berwarna hijau menunjukkan hasil yang sesuai sedangkan kolom yang berwarna merah menunjukkan hasil yang tidak sesuai. Dari tabel tersebut di dapatkan 16 data yang sesuai dari 20 data *sample*. Dengan hasil perbandingan tersebut maka didapatkan hasil 80%.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan oleh penulis mengenai Aplikasi Informasi Pencarian Rute Terpendek Untuk Menemukan Masjid Terdekat di Kota Malang Berbasis Android, dapat disimpulkan sebagai berikut :

1. Berdasarkan pada pengujian yang telah dilakukan, user dapat melakukan pencarian masjid terdekat dengan menggunakan aplikasi informasi pencarian rute terpendek untuk menemukan masjid terdekat di Kota Malang ini sesuai dengan lokasi user berada pada saat pengujian tersebut.
2. Dalam penerapan pencarian rute terpendek memerlukan *marker* atau penanda sebagai proses pencarian. Semakin banyak *marker* yang dibuat, maka akan menjadikan hasil pencarian rute bisa lebih detail lagi.
3. Berdasarkan hasil uji yang telah dilakukan dalam penerapan Algoritma A* untuk pencarian rute terpendek, hasil akurasi yang didapatkan adalah sebesar 80%.

5.2 Saran

Aplikasi ini dapat dikembangkan menggunakan platform lain misalnya, IOS, Windows Phone dan Blackberry OS dan juga

dapat dikembangkan dengan menggunakan algoritma pencarian lainnya.

Daftar Pustaka

Data Keterangan Kota Pariwisata dan Kota Pendidikan.http://budpar.malangkota.go.id/. Diakses pada 08 Desember 2015. Pukul 21.00 WIB

Djojo, Michael Alexander dan Karyono. 2013. "Pengukuran Beban Komputasi Algoritma Dijkstra, A*, dan Floyd-Warshall pada Perangkat Android". *Jurnal ULTIMA Computing*. 5(1). 15-16

Kadir, Abdul. 2010. *Mudah Mempelajari Database MySQL*. Yogyakarta: Andi.

Mufti, Yusuf. 2015. *Panduan Mudah Pengembangan Google Map Android*. Yogyakarta : Andi.

Munir, Rinaldi. 2012. *Matematika Diskrit*. Bandung: Informatika.

Mustofa, Andri., dkk. 2014. "Aplikasi Penentuan Rute Lokasi Kuliner di Kota Malang Berbasis GIS Menggunakan Metode A*". Jurnal Mahasiswa

Program Studi Ilmu Komputer, Program Teknologi Informatika dan Ilmu Komputer Universitas Brawijaya Malang. 8-9.

Mutiana, Veronica., dkk. 2013. "Optimasi Pencarian Jalur dengan Metode A-Star". *Jurnal ULTIMATICS*. 5(2). 43-47.

Rusel, Stuart dan Norvig, Peter. 1995. *Artificial Intelligence: A Modern Approach.* Prentice Hall International Inc.

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

Siregar, Ivan Michael. 2011. *Membongkar Source Code Berbagai Aplikasi Android.* Yogyakarta: Gava Media

Suyanto. 2014. *Artificial Intelligence Searching, Reasoning, Planning, Learning*. Bandung: Informatika.

Winarno Edi., dkk. 2011. *Membuat Sendiri Aplikasi Android Untuk Pemula*. Jakarta: Elexmedia Komputindo.