Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 12, Desember 2017, hlm. 1779-1787 http://j-ptiik.ub.ac.id

Implementasi Algoritma *Dijkstra* Dalam Menemukan Jarak Terdekat Dari Lokasi Pengguna Ke Tanaman Yang Di Tuju Berbasis Android (Studi Kasus di Kebun Raya Purwodadi)

Muhammad Syamsuddin Yusuf[1], Hanifah Muslimah Az-Zahra[2], Diah Harnoni Apriyanti[3]

[1,2]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
[3]Balai Konservasi Tumbuhan Kebun Raya Purwodadi, Lembaga Ilmu Pengetahuan Indonesia
Email: [1]syamsuddinyusuf88@gmail.com, [2]hanifah.azzahra@ub.ac.id, [3]diah007@lipi.go.id

Abstrak

Kebun raya adalah institusi yang memegang dokumentasi mengenai koleksi tumbuhan. Kebun raya Purwodadi terletak di di Jalan Raya Surabaya Malang, Km. 65, Desa Purwodadi, Kecamatan Purwodadi, Kabupaten Pasuruan. Kebun Raya Purwodadi memiliki luas mencapai 85 hektar. Kebun raya purwodadi memiliki koleksi tanaman sejumlah 2002 jenis/*spesies*, 178 suku/*family*, 962 marga/*genus* dan 11.669 *specimen*. Dengan jumlah tanaman yang begitu banyak, dibutuhkan aplikasi yang dapat menunjukkan jalan dari lokasi pengguna ke lokasi tanaman yang dituju. Dalam pembuatan aplikasi, dibutuhkan suatu metode/algoritma untuk melakukan perhitungan guna mendapatkan jarak terdekat. Algoritma yang digunakan pada penelitian ini menggunakan algortima *dijkstra* yang dipilih karena memiliki waktu *running time* lebih cepat dibandingkan algoritma *Bellman-Ford*. Untuk merancang aplikasi yang dibutuhkan, tahap identifikasi kebutuhan fungsional berdasarkan kebutuhan dari pengunjung kebun raya. Sedangkan untuk kebutuhan non-fungsional adalah tentang *usability* dan *compatibility*. Implementasi yang dibuat berdasarkan perancangan yang telah dibuat sebelumnya. *Web server* dibangun menggunakan bahasa PHP, sedangkan aplikasi android menggunakan bahasa Java dengan tools android studio. Pada pengujiannya dilakukan secara *black-box* untuk menguji fungsional dari aplikasi dan semuanya valid. Sedangkan pengujian *white-box* digunakan untuk menguji algoritma *dijkstra* yang digunakan. Selain itu dilakukan pengujian *usability* dan menunjukkan hasil yang memuaskan dengan presentase sebesar 70.916% dengan jumlah responden sebanyak 30 orang.

Kata Kunci: kebun raya Purwodadi, jarak terdekat, Algoritma *Dijkstra, usability*

Abstract

Botanical gardens is an institution that keeps documentation about living plants collection Purwodadi Botanical Gardens located on Surabaya Malang Km. 65 Highway, Purwodadi Village, Purwodadi District, Pasuruan Regency. Purwodadi Botanical Garden has an area of 85 hectares. Purwodadi Botanical Garden has plants collection of 2.002 species, 178 family, 962 genus and 11.669 specimens. With abundance of plants, an application is required that can lead the way from user's location to intended plant's location. In developing the application, it takes a method/algorithm to perform calculations to get the closest distance. The algorithm used in this study is dijkstra algorithm. The dijkstra algorithm is chosen because it has faster running time than the Bellman-Ford algorithm. To design the required applications, functional requirements identification stage is performed based on visitors' needs. As for non-functional requirements, it is about usability and compatibility. Implementation was created based on the design that has been made before. Web server was built using PHP language, while the android app uses Java language with tools Android Studio. The test was performed in black-box to test the functionality of the application and every results showed valid. While white-box testing was performed to test the dijkstra algorithm. In addition, usability testing was performed and showed satisfactory results with a percentage of 70.916% by 30 respondents.

Keywords: *Purwodadi botanical garden, shortest distance, Dijkstra Algorithm, usability*

1. PENDAHULUAN

Dalam kehidupan sehari-hari, sering kali seseorang melakukan suatu perjalanan. Dan mayoritas perjalanan tersebut ingin dilalui dengan jarak yang terdekat. Alasannya adalah untuk penghematan bahan bakar, waktu dan tenaga (Ardiani, 2011). Hasil penentuan jarak terdekat tersebut menjadi acuan dalam menentukan jalur mana yang akan dilewati. Selain itu, harapan dari setiap orang dalam melakukan perjalanan tersebut agar sampai pada tujuan dan tidak tersesat di tengah jalan.

Salah satu metode pencarian jarak terdekat adalah dengan menggunakan algoritma *Dijkstra*. Algoritma *Dijkstra* adalah algoritma yang menerapkan graph berarah dan berbobot, dimana jarak antar titik adalah bobot dari tiap panah tersebut (Lubis, 2009). Algoritma ini akan mencari jalur dengan cost yang paling minimum antara titik yang satu dengan titik yang lainnya. Selain itu, algoritma *Dijkstra* juga bisa digunakan untuk menghitung total biaya atau cost dari lintasan terpendek yang sudah terbentuk (Fakhri, 2008).

Alasan dipilihnya algoritma *Dijkstra* adalah algoritma ini adalah salah satu jenis dari algoritma greedy. Karena algoritma *Dijkstra* beroperasi secara menyeluruh terhadap alternatif fungsi yang ada, dan dihasilkan lintasan terpendek dari semua node (Lubis 2009). Selain itu, juga ada penelitian dari Michi Purna Irawan (2011) yang menjelaskan bahwa algoritma *Dijkstra* lebih cepat mengeksekusi programnya dibandingkan dengan algoritma Bellman-Ford dengan asumsi tidak ada nilai negatif. Nilai negatif memiliki arti bahwa setiap jarak antar persimpangan pasti ada nilainya, dan satuan nilai tersebut selalu positif. Selain itu algoritma *Dijkstra* dapat menyelesaikan beberapa kasus jalur terpendek, antara lain: pencarian jalur terpendek antara dua buah simpul (*a pair shortest path*), pencarian jalur terpendek antara semua pasangan simpul *(all pairs shortest path),* pencarian jalur terpendek dari simpul tertentu ke semua simpul yang lain *(single-source shortest path)* dan pencarian jalur terpendek antara dua buah simpul yang melalui beberapa simpul tertentu *(intermediate shortest path)* (Andriani, 2011).

Kebun raya adalah institusi yang memegang dokumentasi mengenai koleksi tumbuhan hidup dan digunakan untuk tujuan penelitian ilmiah, konservasi, pameran dan pendidikan (Hulme, 2011). Kebun raya Purwodadi adalah salah satu kebun raya di Indonesia yang berada di Kecamatan Purwodadi, Kabupaten Pasuruan. Kebun raya Purwodadi didirikan pada tanggal 30 Januari 1941 oleh Dr. L.G.M. Baas Becking, dengan luas 85 hektar.

Dengan luas area kebun raya Purwodadi yang mencapai 85 hektar, membuat pengunjung yang pertama kali datang akan merasa kebingungan. Bahkan pengunjung yang sudah berkali-kali datang pun kemungkinan juga akan merasa kebingungan untuk mencari tempat ataupun tanaman yang ingin dicari. Tanpa bantuan seorang pemandu, hampir pasti pengunjung mengalami kesulitan menemukan tanaman yang dicari (Oktavianingsih, 2014). Kurangnya papan informasi yang disediakan pihak pengelola kebun raya, menambah kebingungan dari pengunjung. Papan penunjuk jalan tidak selalu ada di persimpangan jalan. Dan gambar peta dari kebun raya Purwodadi hanya ada satu buah yang berada sekitar 50 meter dari pintu gerbang.

Di dalam kebun raya Purwodadi, terdapat banyak jalan dan persimpangan. Masing-masing persimpangan dapat direpresentasikan sebagai *vertex*. Dan masing-masing jalan bisa direpresentasikan sebagai *edge*. Sedangkan jarak antar persimpangan dapat didefinisikan sebagai *bobot* dari *edge* tersebut. Sehingga dengan melihat peta kebun raya Purwodadi dapat di identifikasi berapa jumlah *vertex* dan *edge* beserta bobotnya. Dan data tersebut bisa digunakan untuk menentukan jarak terpendek dari titik seorang pengguna ke tanaman yang ingin dituju dengan menggunakan algoritma *Dijkstra.*

Kebanyakan orang menginginkan suatu kemudahan dalam penggunaan suatu aplikasi. Kemudahan penggunaan aplikasi tersebut dapat meningkatkan minat pengguna untuk menggunakan aplikasi. Kemudahan dalam menggunakan aplikasi dapat di ukur dengan adanya pengujian tingkat *usability*. Pengujian *usability* bertujuan untuk mengetahui seberapa besar kemudahan yang didapat oleh pengguna untuk mengoperasikan suatu aplikasi (Munaiseche, 2012). *Usability* sendiri memiliki atribut kualitas yang menunjukkan seberapa mudah antarmuka tersebut digunakan oleh pengguna. Dan secara umum *usability* mengacu pada bagaimana pengguna tersebut mempelajari dan menggunakan suatu sistem (Nielsen, 2012). Hasil dari pengukuran tingkat *usability* nantinya bisa digunakan untuk mengembangkan aplikasi

agar lebih baik dari sebelumnya.

Maka berdasarkan latar belakang yang sudah dipaparkan tersebut, dibutuhkan suatu solusi yang bisa menunujukkan kepada pengunjung jalan dari titik pengguna ke tempat tanaman yang akan dituju. Aplikasi tersebut diharapkan memiliki nilai *usability* yang baik, agar pengguna dari aplikasi nantinya bisa dengan mudah mempelajari dan menggunakan aplikasi. Oleh karena itu fokus dari penilitian ini adalah untuk mencari jalur terpendek dari titik lokasi pengguna ke tempat tanaman yang ingin di cari oleh pengguna dengan berbasis android. Sehingga diharapkan aplikasi ini dapat membantu pengguna menemukan tanaman yang ingin di cari.

2. LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada penelitian sebelumnya telah dijelaskan oleh Yi-zhou Chen et al, dalam penelitiannya yang berjudul *"Path Optimization Study for Vehicles Evacuation Based on Dijkstra algorithm"* yang diterbitkan pada tahun 2014. Pada penelitian tersebut membahas ketika terjadi kaeadaan darurat yang disebabkan oleh gempa bumi, angin topan, kebakaran, kecelakaan kimia, kecelakaan nuklir, serangan terror dan ancaman lainnya yang dapat menyebabkan cedera ataupun membahayakan kehidupan manusia. Dan orang-orang berada disekitar daerah darurat tersebut dianjurkan untuk mengungsi ke daerah yang lebih aman. Maka diperlukan sebuah model jaringan jalan dinamis yang dibangun untuk jalur evakuasi kendaraan menggunakan algoritma *Dijkstra.* Dan hasil yang diperoleh adalah dengan memberikan metode yang optimal dalam memilih jalur evakuasi, terutama dengan tingkat kepadatan penduduk yang tinggi.

Penelitian kedua adalah penelitan mengenai perbandingan algoritma *Greedy* dan *Dijkstra* untuk menentukan lintasan terpendek yang dilakukan Henny Syahriza Lubis (2009). Dari penelitian tersebut dapat ditarik kesimpulan bahwa algoritma *Greedy* menghasilkan jarak yang lebih besar daripada algoritma *Dijkstra*. Algortima *Greedy* tidak mengihitung semua alternatif fungsi yang ada, sehingga vertex yang dihasikan hanya dari titik awal ke titik tujuan. Sedangkan pada algortima *Dijkstra* beroperasi secara menyeluruh terhadap semua alternatif yang ada, sehingga didapatkan lintasan terpendek dan lintasan terpendek dari semua node.

Pada penelitan ini akan dikembangkan mengenai implementasi algoritma *Dijkstra* dalam menemukan jarak terdekat dari lokasi titik pengguna ke tanaman yang di tuju berbasis android. Studi kasus yang digunakan penulis adalah pada Kebun Raya Purwodadi. Aplikasi ini nantinya akan dikembangkan dengan menggunakan online storage pada *web service* yang terkoneksi dengan *server* sebagai pusat pengambilan data.

2.2 Kebun Raya Purwodadi

Menurut Philip E. Hulme (2011), Kebun raya adalah institusi yang memegang dokumentasi mengenai koleksi tumbuhan hidup dan digunakan untuk tujuan penelitian ilmiah, konservasi, pameran dan pendidikan. Kebun Raya Purwodadi terletak di Jalan Raya Surabaya Malang, Km. 65, Desa Purwodadi, Kecamatan Purwodadi, Kabupaten Pasuruan. Kebun Raya Purwodadi memiliki luas mencapai 85 hektar pada ketinggian sekitar 300 mdpl (meter diatas permukaan laut). Kebun Raya Purwodadi memiliki curah hujan rata-rata per tahun 2366 mm dengan bulan basah antara bulan November sampai Maret. Dengan suhu sekitar 22^{o} – 32^{o}C, membuat kebun ini terasa sejuk untuk dikunjungi (Kebun Raya Purwodadi, 2016).

Kebun Raya Purwodadi memiliki sekitar 2002 jenis koleksi pohon dan tumbuhan. Jumlah keseluruhan koleksi Kebun Raya Purwodadi per Oktober 2013, seperti pada Tabel 1 berikut:

Tabel 1. Jumlah koleksi kebun raya Purwodadi

	Suku	Marga	Jenis	Specimen
Jumlah	178	962	2002	11.669

2.3 Lintasan Terpendek

Lintasan terpendek adalah jarak yang ditempuh dari suatu titik ke titik yang lain dengan jarak tempuh yang paling pendek. Untuk mencari lintasan terpendek dalam suatu *graph*, berarti membicarakan masalah optimasi. *Graph* yang digunakan dalam mencari lintsan terpendek, menggunakan *graph* berbobot. Bobot pada *graph* bisa berupa jarak, waktu, biaya dan sebagainya. Biasanya, bobot yang ada pada *graph* berupa nilai positif. Tetapi tidak menutup kemungkinan terdapat nilai yang negatif (Lubis,

2009).

Menurut Ardiani (2011), terdapat beberapa macam persmasalahan lintasan terpendek, antara lain:

1. Pencarian lintasan terpendek antara dua buah simpul tertentu *(a pair shortest path).*
2. Pencarian lintasan terpendek antara semua pasangan simpul *(all pairs shortest path).*
3. Pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain *(single-source shortest path).*
4. Pencarian lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu *(intermediate shortest path).*

2.4 Algoritma *Dijkstra*

Algoritma *Dijkstra* bisa disebut juga sebagai algoritma *greedy*. Algoritma ini merupakan salah satu algoritma yang digunakan untuk menyelesaikan jalur terpendek dan tidak memiliki *cost* yang negatif. Strategi *greedy* yang digunakan dalam algoritma ini adalah setiap simpul dalam *graph* akan mencari nilai yang minimum. Algoritma ini akan mencari jalur dengan *cost* yang paling minimum antara titik yang satu dengan titik yang lainnya. Selain itu, algoritma *Dijkstra* juga bisa digunakan untuk menghitung total biaya atau *cost* dari lintasan terpendek yang sudah terbentuk (Fakhri, 2008).

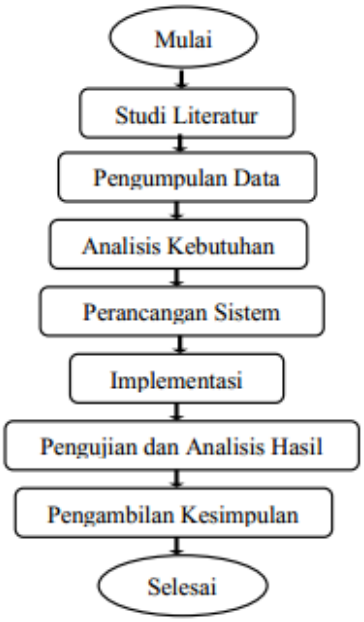
Menurut Fakhri (2008), di dalam melakukan perhitungan dengan algoritma *Dijkstra* ada beberapa skema umum yang digunakan pada pencarian jarak terpendek, antara lain:

1. Membuat 3 buah *list*, yaitu *list* jarak (*list* 1), *list* simpul-simpul sebelumnya (*list* 2) dan *list* yang sudah dikunjungi (*list* 3), serta sebuah variabel untuk menampung simpul pada saat ini (*current vertex*).
2. Dalam *list* jarak, diisi dengan nilai tak hingga, kecuali simpul awal yang diisi dengan nilai 0.
3. Pada *list* 2, diisi dengan *false*.
4. Pada *list* 3, diisi dengan *null*.
5. *Current vertex* diisi dengan simpul awal (*start*).
6. Menandai *current vertex* sebagai simpul yang telah dikunjungi.
7. *Update list* 1 dan 2 berdasarkan simpul-simpul yang dapat langsung dicapai dari *current vertex.*
8. *Update current vertex* dengan simpul yang paling dekat dengan simpul awal.
9. Lakukan langkah no 6 sampai semua simpul telah dikunjungi.

Algoritma *Dijkstra* memiliki kompleksitas $O(n^2)$. Sehingga total waktu *asimptotik* komputasi dalam mencari pasangan *vertex* terpendek adalah $T(n) = n.(n^2) = O(n^3)$. Dengan demikian algoritma *Dijkstra* memiliki keuntungan lebih dalam hal *running time*.

3. METODOLOGI PENELITIAN

Metodologi penelitian adalah langkah-langkah yang digunakan untuk menyelesaikan penelitian ini. Penulis menggunakan strategi ataupun langkah-langkah agar penelitian ini bisa terselesaikan tepat waktu. Tahapan-tahapan metodologi penelitian yang digunakan penulis dalam menyelesaikan penelitian ini dapat dilihat pada Gambar 1 berikut.



Gambar 1. Strategi Penelitian

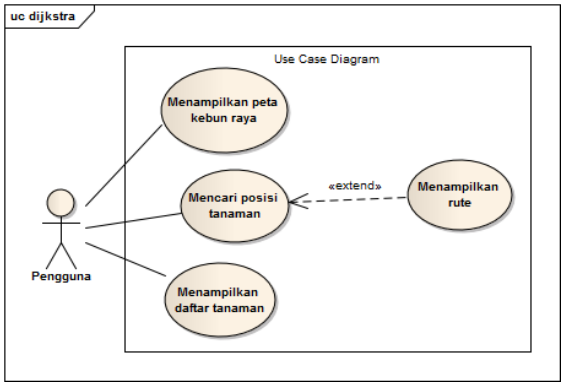
4. REKAYASA KEBUTUHAN

Proses identifikasi kebutuhan yang terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional pengguna menggunakan penomoran *Software Requirement Specification (SRS).*

4.1 Kebutuhan Fungsional

Kebutuhan fungsional, dapat dimodelkan pada *use case diagram. Use case diagram* adalah diagram yang memodelkan perilaku dari sistem. Di dalam *use case diagram* terdapat sekumpulan *use case*, aktor dan hubungan antara *use case* dan aktor. *Use case diagram* dapat dilihat pada

Gambar 2.



Gambar 2. *Use Case Diagram*

4.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah batasan dalam layanan atau fungsi yang ditawarkan sistem. Dalam penelitian ini, kebutuhan nonfungsional yang digunakan ada dua macam, seperti pada Tabel 2.

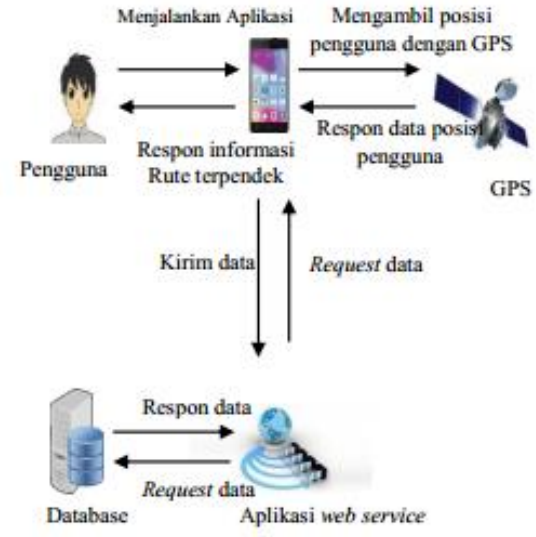
Tabel 2. Kebutuhan Non Fungsional

Kode	Jenis Kebutuhan	Deskripsi
SRS_NF_01	*Usability*	Sistem yang dibuat harus dapat dipelajari dan digunakan dengan mudah oleh pengguna. Kemudahan-kemudahan tersebut diantaranya kemudahan dalam hal penggunaan sistem.
SRS_NF_02	*Compatibility*	Sistem dapat diakses melalui beberapa versi sistem operasi android

5. PERANCANGAN DAN IMPLEMENTASI

5.1 Gambaran Umum Aplikasi

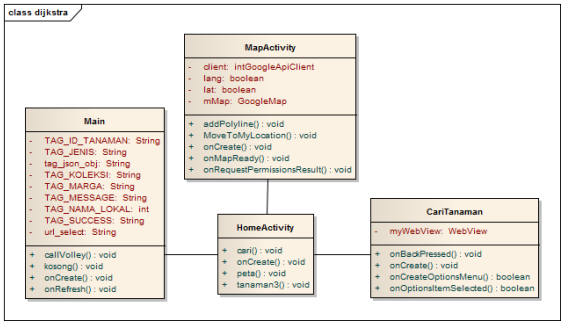
Perancangan arsitektur menjelaskan hubungan tentang elemen apa saja yang berhubungan dengan aplikasi.



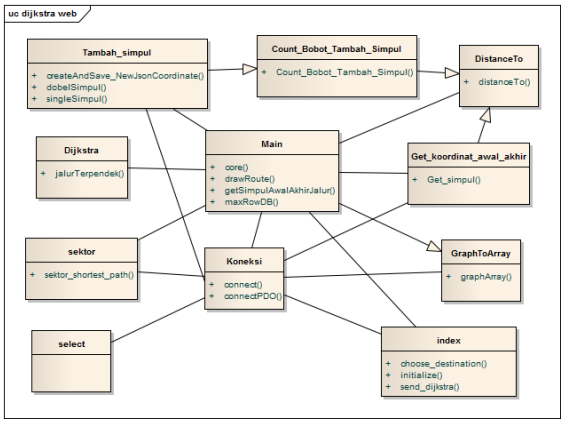
Gambar 3. Diagram Blok Perancangan Arsitektur

5.2 Perancangan Diagram Klas

Berikut adalah diagram klas yang digunakan dalam implementasi algoritma *dijkstra* dalam menemukan jarak terdekat dari lokasi titik pengguna ke tanaman yang di tuju berbasis android.



Gambar 4. Diagram Klas pada Android



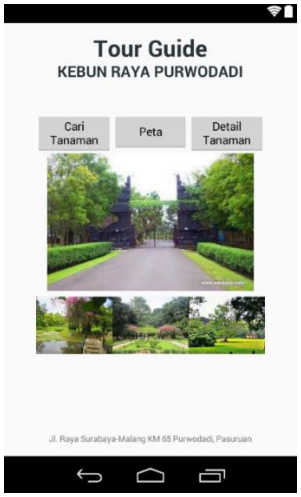
Gambar 5. Diagram Klas *Web Service*

6. IMPLEMENTASI DAN PENGUJIAN

6.1 Implementasi

6.1.1 Implementasi Halaman *Home*

Pada implementasi halaman *Home* terdapat judul dari aplikasi, menu utama, beberapa gambar kebun raya dan alamat kebun raya Purwodadi. Hasil implementasi antarmuka halaman *Home* dapat dilihat pada Gambar 6.



Gambar 6. Implementasi Halaman *Home*

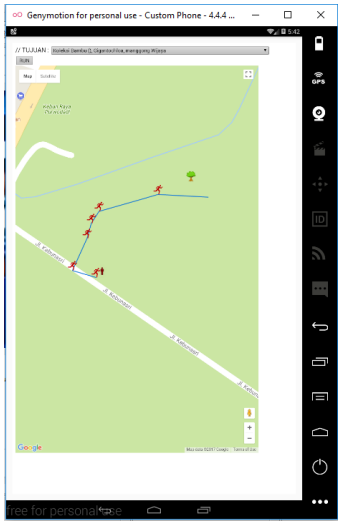
6.1.2 Implementasi Halaman Peta

Pada implementasi halaman peta aplikasi akan menampilkan peta secara keseluruhan beserta dengan jalan-jalan yang ada di kebun raya Purwodadi. Hasil implementasi antarmuka halaman peta dapat dilihat pada Gambar 7.

Gambar 7. Peta Kebun Raya Purwodadi

6.1.3 Implementasi Halaman Cari Tanaman

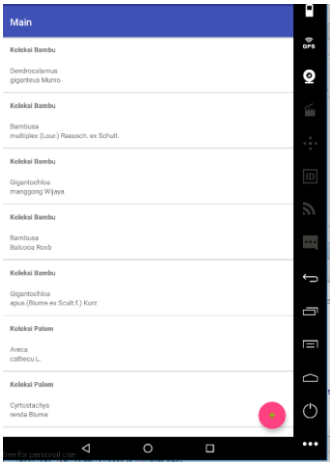
Pada implementasi halaman cari tanaman, pengguna akan masuk pada halaman peta kebun raya. Kemudian pengguna diminta memilih tanaman apa yang akan dicari yang sudah tersedia di sistem. Setelah itu pengguna menentukan posisi dari mana asal pengguna tersebut. Dan yang terakhir pengguna menekan tombol cari. Sesuai pada Gambar 8.



Gambar 8. Halaman Cari Tanaman

6.1.4 Implementasi Halaman Daftar Tanaman

Pada implementasi halaman daftar tanaman, pengguna akan melihat tanaman apa saja yang sudah masuk dalam sistem. Data tersebut meliputi koleksi, nama lokal (tidak semua ada), jenis dan marga. Sesuai pada Gambar 9.



Gambar 9. Daftar Tanaman

6.1.5 Implementasi Algoritma

1. Membuat 3 buah *list*, yaitu *list* jarak (*list* 1), *list* simpul-simpul sebelumnya (*list* 2) dan *list* yang sudah dikunjungi (*list* 3), serta

sebuah variabel untuk menampung simpul pada saat ini (*current vertex*).
2. Dalam *list* jarak, diisi dengan nilai tak hingga, kecuali simpul awal yang diisi dengan nilai 0.
3. Pada *list* 2, diisi dengan *false*.
4. Pada *list* 3, diisi dengan *null*.
5. *Current vertex* diisi dengan simpul awal (*start*).

```
$graph    = $arg_graph;
$simpul_awal  = $simpulAwal;
$simpul_maju  = $simpulAwal;
$simpul_tujuan  = $simpulTujuan;
$jml_simpul  = count($arg_graph);
$simpulYangDikerjakan = array();
$simpulYangSudahDikerjakan_bawah =
array();
$nilaiSimpulYgDitandai = 0;
$nilaiSimpulFixYgDitandai = 0;
```

6. Menandai *current vertex* sebagai simpul yang telah dikunjungi.

```
for($perulanganSimpul = 0;
$perulanganSimpul <
count($simpulYangDikerjakan);
$perulanganSimpul++){
    $jumlah_baris = count($graph[
$simpulYangDikerjakan[$perulanganSim
pul] ]);
    $bobot_baris = array();
    $baris_belum_dikerjakan = 0;
```

7. *Update list* 1 dan 2 berdasarkan simpul-simpul yang dapat langsung dicapai dari *current vertex.*

```
for($min_indexAntarBobotYgDitandai
= 0; $min_indexAntarBobotYgDitandai
< count($perbandinganSemuaBobot);
$min_indexAntarBobotYgDitandai++)
  {

if($perbandinganSemuaBobot[$min_inde
xAntarBobotYgDitandai] <=
$perbandinganSemuaBobot[0]){

      $perbandinganSemuaBobot[0] =
$perbandinganSemuaBobot[$min_index
AntarBobotYgDitandai];
  }
}
```

8. *Update current vertex* dengan simpul yang paling dekat dengan simpul awal.

```
    foreach($simpulYangDikerjakan as
$idx => $v){
     $length_baris =
$graph[$simpulYangDikerjakan[$idx]];
     for($baris1 = 0; $baris1 <
$length_baris; $baris1++){
      if( isset($graph[
$simpulYangDikerjakan[$indexAwalAsli
] ][$baris1 ]) ){
       $bobot_baris_dan_ruas1 =
$graph[
$simpulYangDikerjakan[$indexAwalAsli
] ][$baris1 ];
       $explode1 = array();
       $explode1 = explode('->',
$bobot_baris_dan_ruas1);
       if(count($explode1) == 2){

if($perbandinganSemuaBobot[0] ==
$explode1[1]){
        $dapat_indexAsliBobot =
$baris1;
        $simpul_lama =
$simpulYangDikerjakan[$indexAwalAsli
];
        $simpul_maju =
$explode1[0];
        $baris_belum_dikerjakan1 +=
1;
       }
      }
     }

     else{
      break;
     }
    }

    $indexAwalAsli++;
   }
```

9. Lakukan langkah no 6 sampai semua simpul telah dikunjungi.

6.2 Pengujian

6.2.1 Pengujian Unit

Pengujian unit yang digunakan dalam menguji aplikasi adalah pengujian *white-box* dan *black-box*. Pengujian *black-box* digunakan untuk menguji kesesuaian antara daftar kebutuhan fungsional dengan sistem yang sudah dibuat. Sedangkan pengujian *white-box* digunakan untuk menguji algoritma yang diterapkan di aplikasi. Pada penelitan ini semua fungisonal telah sesuai dengan apa yang diharapkan. Sedangkan untuk pengujian *white-box* juga menghasilkan nilai *cyclometic complecity* yang juga sesuai dengan jalur independen yang dihasilkan.

6.2.2 Pengujian Usability

Pengujian *usability* digunakan untuk dapat diketahui seberapa mudah aplikasi tersebut dapat dijalankan oleh pengguna. Teknik pengujian *usability* yang digunakan adalah dengan memberikan kuesioner kepada pengunjung kebun raya Purwodadi. Jenis pertanyaan yang ada pada kuesioner mengacu pada kuesioner SUS. Dalam pengujian usability ini, melibatkan 30 orang pengunjung yang berbeda. Adapun hasil rekap kuesioner dapat dilihat pada Tabel 3.

Tabel 3. Rekap Hasil Pengisian Kuesioner Pengunjung

No.	Responden	Jumlah	Skor
1.	Elma Puspaningtiyas	31	31*2.5 = 77.5
2.	Siti Khoiriyah	30	30*2.5 = 75
3.	Mega Goe Arsa	26	26*2.5 = 65
4.	Astrid Ruliyati Kuswardani	28	28*2.5 = 70
5.	Usman Supriadi	24	24*2.5 = 60
6.	Yunita	26	26*2.5 = 65
7.	Leli Resita	33	33*2.5 = 82.5
8.	Yizra Salsa	24	24*2.5 = 60
9.	Sevi Alfrida	27	27*2.5 = 67.5
10.	Dian Noviana	25	25*2.5 = 62.5
11.	Rochmad Alfian D.M.	29	29*2.5 = 75
12.	Nizam Mauluddin	30	30*2.5 = 75
13.	Riki Wahyu Saputra	33	33*2.5 = 82.5
14.	M. Rizki AL J.	28	28*2.5 = 70
15.	Rachmad Ramadani	27	27*2.5 = 67.5
16.	Apriliani Eka Wulandari	31	31*2.5 = 77.5
17.	M. Dzul Fikri	31	31*2.5 = 77.5
18.	Iin Mufarida	27	27*2.5 = 67.5
19.	Tri Fauzi Ningrum	24	24*2.5 = 60
20.	Nuril Masdharul Khusna	29	29*2.5 = 72.5
21.	Eka Ratih Wulan	26	26*2.5 = 65
22.	Windy Trismawati	24	24*2.5 = 60
23.	Farih Azzuhri	31	31*2.5 = 77.5
24.	M. Figo AL faro	29	29*2.5 = 72.5
25.	Wardatun Nabila	26	26*2.5 = 65
26.	Laila M.	26	26*2.5 = 65
27.	Rizky Diah Aryani	31	31*2.5 = 77.5
28.	Ima Anjasari	30	30*2.5 = 75
29.	Bustanul Syaifuddin	31	31*2.5 77.5
30.	Moh. Abdul Rozak	33	33*2.5 = 82.5

Keterangan:

Total Jumlah = Jumlah responden

Total Skor =

- Jika pertanyaan 1,3,5,7,9 = skor yang di dapat dikurangi 1
- Jika pertanyaan 2,4,6,8,10 = skor yang didapat adalah 5, dan dikurangi posisi skala.

Kesimpulan

$$=77.5+75+65+70+60+65+82.5+60+67.5+62.5+75+75+82.5+70+67.5+77.5+77.5+67.5+60+72.5+65+60+77.5+72.5+65+65+77.5+75+77.5+82.5$$
$$= 2127.5/30$$
$$= 70.916$$

6.2.3 Pengujian Compatibility

Pengujian *compatibility* dilakukan untuk menguji, apakah aplikasi dapat digunakan di beberapa versi android yang digunakan. Skenario pengujian *compatibility* dilakukan bersamaan dengan menyebar kuesioner *usability*. Responden diminta untuk menyebutkan jenis android yang digunakan untuk pengujian *compatibility*. Selain itu pengujian *compatibility* dilakukan dengan spesifikasi minimum dan maksimum dari spesifikasi sistem.

Tabel 4. Pengujian Compatibility

No.	Jenis Android	Pengguna	Keterangan
1.	Android 2.3 Gingerbread		
2.	Android 4.0 Ice Cream Sandwich		

3.	Android 4.1 Jelly Bean	16	Aplikasi dapat dijalankan
4.	Android 4.4 Kitkat	4	Aplikasi dapat dijalankan
5.	Android 5.0 Lollipop	6	Aplikasi dapat dijalankan
6.	Android 6.0 Marshmallow	4	Aplikasi dapat dijalankan

7. KESIMPULAN

Berdasarkan penelitian yang sudah dilakukan, terdapat beberapa kebutuhan fungsional yang meliputi: menampilkan peta kebun raya, mencari posisi tanaman, menampilkan rute dan menampilkan daftar tanaman. Selain itu terdapat kebutuhan non-fungsional *usability* dan *compatibility*. Fungsional yang menerapkan implementasi algortima *dijkstra* adalah menampilkan rute. Rute yang didapatkan berdasarkan perhitungan setiap jalan dan persimpangan yang ada di kebun raya. Persimpangan yang ada di kebun raya Purwodadi direpresentasikan sebagai *vertex* atau *node*. Sedangkan jarak antar persimpangan direpresentasikan sebagai *edge*. Sehingga dari data yang didapatkan di lapangan, bisa dihitung jarak terpendek antara titik pengguna dan titik tujuan atau tanaman.

Hasil implementasi algoritma *dijkstra* dalam menentukan jarak terpendek dari lokasi pengguna ke tanaman yang di tuju menghasilkan nilai *usability* sebesar 70.916%, yang diambil dari 30 responden. Nilai tersebut berada di rentang interval 60% - 80% yang dapat dikategorikan sebagai aplikasi yang memuaskan.

DAFTAR PUSTAKA

Ardiani, Farida. 2011. Penentuan jarak terpendek dan waktu tempuh menggunakan algoritma Dijkstra dengan pemrograman berbasis objek. Skripsi Universitas Islam Negeri Sunan Kalijaga

Chen, Yi-zhou, et al. 2014. Path optimization study for vehicles evacuation based on Dijkstra algorithm. Elsevir, Procedia Engineering 71, 159 - 165

Fakhri. 2008. Penerapan algoritma Dijkstra dalam pencarian solusi maximum flow problem. Makalah IF2251 Strategi Algoritmik

Hulme, Philip E. 2011. *Addressing the threat to biodiversity from Botanic Garden.* The Bio-Protection Research Centre. Christchurch: Trends in Ecology and Evaluation, Vol. 26, No. 4

Kebun Raya Purwodadi. 2016. Tentang Kebun Raya Purwodadi. Tersedia di <http://krpurwodadi.lipi.go.id/> [di akses pada 28 Januari 2016]

Lubis, Henny Syahriza. 2009. Perbandingan algoritma greedy dan dijkstra untuk menentukan lintasan terpendek. Skripsi Universitas Sumatera Utara

Munaiseche, Cindi P.C. 2012. Pengujian web aplikasi DSS berdasarkan pada aspek usability. Orbith Vol. 8, No. 2

Nielsen, Jakob. 2012. Usability 101: Introduction to Usability. Tersedia di <https://www.nngroup.com/articles/usability-101-introduction-tousability/> [di akses pada 2 Maret 2016]

Oktavianingsih, Desi. 2014. Keragaman Hayati di Kebun Raya Bogor. Tersedia di <http://desioktavianingsih04.blogspot.co.id/2014/03/keragaman-hayati-di-kebun-raya-bogor.html> [di akses pada 4 April 2016]