1. Pendahuluan

Wisatawan atau biasa disebut dengan *traveller* biasanya memiliki mobilitas yang tinggi. Terutama bagi yang senang berkendaraan *touring* dan wisata menggunakan motor. Kadangkala ketika melakukan kegiatan tersebut, dibutuhkan media penyimpanan yang tepat agar lokasi dapat disimpan dengan baik. Tidak hanya lokasi, tentunya gambar pada lokasi tertentu juga dibutuhkan sebagai bentuk dokumentasi. Permasalahan yang muncul dari kondisi tersebut adalah mahalnya harga perangkat yang dibutuhkan. Perangkan penyimpan lokasi dan penunjuk arah perjalanan saat ini memiliki harga sekitar delapan juta. Tentunya harga tersebut terbilang mahal mengingat jangkauan masyarakat Indonesia. Masalah lainnya yang timbul adalah mobilitas yang tidak efisien. Perangkat yang ada kurang praktis untuk di bawa padahal pengguna kebanyakan adalah orang yang suka bepergian dan memiliki mobilitas tinggi. Berdasarkan latar belakang dan masalah yang telah disebutkan di atas, maka pada penelitian ini dicari solusi untuk menyimpan lokasi dengan mudah dan biaya yang efisien. Faktor mobilitas juga akan menjadi pertimbangan besar di dalam penelitian ini. Tujuan dari peneltian ini adalah melakukan perancangan aplikasi Android berbasis *Location Based Service* yang mampu menyimpan rute suatu perjalanan dan gambar kemudian menampilkanya kembali di *map* tanpa harus mengeluarkan biaya yang besar dan mudah di bawa-bawa dengan memanfaatkan fitur-fitur di dalam teknologi Android seperti fitur *GPS*, fitur *browse* gambar,*Google Map* dan penyimpanan basis data. Melakukan implementasi dari hasil perancangan yang telah di lakukan menggunakan *platform* Android.

2. Tinjauan Pustaka

Terdapat penelitian terdahulu yang digunakan sebagai acuan dalam penelitian ini. Penelitian tersebut adalah penelitian yang berjudul *"Desain Mobile Agent Pencarian Hotel Berbasis Android*" yang memanfaatkan teknologi pemetaan pada Android berbasis *Google Map API* serta memafaatkan teknologi *GPS* sebagai fitur untuk mendapatkan lokasi pengguna. Data hotel disimpan di dalam basis data *server* dan dimanajemen menggunakan *web*site berbasis *PHP*. Kemudian melalui aplikasi Android, data hotel tersebut diakses dengan menggunakan teknologi *web service*. Setelah data tersebut didapatkan, maka oleh perangkat Android memunculkan lokasi hotel dalam pemetaan *Google Map*. Setiap titik hotel yang muncul, ketika dilakukan *touch* pada *marker* yang ditampilkan akan muncul informasi detail dari setiap hotel. Kemudian berdasarkan lokasi yang didapatkan dari *GPS*, akan ditentukan lokasi pengguna pada *Google Map*. Lokasi pengguna dimanfaatkan untuk memunculkan jalur jalan antara lokasi pengguna dengan lokasi hotel. Pencarian hotel dapat dikategorikan berdasarkan harga hotel serta bintang hotel[2]. Perbedaan antara penelitian pertama dengan penelitian ini adalah jika pada penelitian pertama memanfaatkan *GPS* dan *Google Map* untuk menampilkan daftar hotel beserta jalurnya. Maka pada penelitian ini memanfaatkan *GPS* untuk menyimpan lokasi pengguna saat ini dan menampilkannya kembali apabila diperlukan melalui *Google Map*s.

Android adalah suatu sistem operasi berbasis Java yang berjalan di atas *kernel Linux 2.6*. Sistem ini sangat ringan dan memiliki fitur yang lengkap. Selain itu, Android memiliki berbagai fitur yang menarik seperti grafis 3D, basis data yang cepat menggunakan *SQLite* dan *web browser* yang telah terintegrasi. Jika telah menguasai bahasa pemrograman Java ataupun bahasa pemrograman berbasis objek yang lain maka kemampuan tersebut cenderung dapat digunakan di dalam pengembangan program Android. Penempatan antarmuka dapat ditangani secara langsung dalam kode program. Pada Android, pemrograman antarmuka pengguna juga dapat dilakukan menggunakan pemrograman *XML* sehingga dapat dilakukan dengan lebih mudah menggunakan versi *visual drag-n-drop*. Antarmuka berbasis *XML* merupakan teknologi yang termasuk masih baru di dalam konsep pengembangan aplikasi berbasis *mobile*, dan pada Android keduanya didukung. Arsitektur Android terdiri dari lima lapisan utama yaitu lapisan *Linux Kernal* yaitu lapisan yang bertugas sebagai sistem operasi utama dan melakukan pemrosesan bagian *low level*, Lapisan berikutnya adalah *Libraries* untuk menangani berbagai keperluan, Kemudian ada lapisan *Android Runtime* berisi *Dalvik Virtual Machine* layaknya *Java Virtual Machine* pada Java. Di atasnya terdapat *Application Framework* yang berisi manager aplikasi. Dan yang paling atas adalah *Applications* yaitu aplikasi yang berjalan dan sudah terinstal di atas *platform* Android itu sendiri [3].

Google Map telah memiliki efek mendasar pada dunia pemetaan berbasis *web*. Sementara orang lain masih menggunakan gambar statis, pengembang *Google* diam-diam mengembangkan antarmuka *slickest*. Kemudian mereka mengambil citra satelit dari jalan-jalan dan lokasi dengan ukuran *terabyte* kemudian memberikan semuanya secara gratis[4]. Banyak teknologi *web* yang penting dan bermanfaat bermunculan setelah dikembangkannya *Google Map API*. Katakanlah teknologi *Ajax* atau *Web* 2.0 merupakan teknologi yang sangat mendukung pemanfaatkan *Goole Map API*. Tidak diperlukan membeli perangkat lunak yang mahal untuk menampilkan *map*, cukup menggunakan *Google Map API*. Tidak diperlukan banyak pengalaman di semua bidang komputer, namun diperlukan hanyalah mempelajari *Google Map API*. Kebutuhan sistem adalah salah satu data penting dan ide apa yang dapat dikembangkan untuk menghadirkan data secara persuasif melalui pemetaan. Saat ini *Google Map API* tidak hanya dapat digunakan sebagai pemetaan berbasis *web*, namun dapat juga digunakan di dalam perangkat *mobile* yang berbasis Android. Android merupakan produk *Google* yang tentu saja mendukung penuh fitur *Google Map API*. Telah disediakan *Android Google Map API* yang cukup lengkap untuk membangun aplikasi berbasis pemetaan tanpa harus mengeluarkan biaya mahal dan pembelajaran yang rumit. *Google Map API* menyediakan tampilan yang lengkap di dalam pemetaan sehingga pihak pengembang sistem cukup memanipulasi data yang akan ditampilkan. Hal tersebut mempermudah membangun sistem pemetaan tanpa harus membuat citra digital *map* dari awal.

SQLite adalah basis data relasional yang berbasis *open source* dan berbasis *embedded*. Awal mula dirilis pada tahun 2000, dirancang untuk menyediakan cara mengelola data yang nyaman untuk aplikasi tanpa adanya *overhead* yang sering muncul pada basis data relasional. *SQLite* memiliki reputasi

untuk menjadi sangat *portable*, mudah digunakan, kompak, efisien dan dapat diandalkan[5]. *SQLite* sendiri merupakan jenis basis data berbasis *embedded*. Daripada basis data berjalan sendiri secara mandiri dan independen, *SQLite* menyediakan layanan basis data yang akan selalu mendampingi aplikasi yang berjalan dan memerlukan datanya. Pengguna aplikasi dapat menginstal aplikasi tanpa mengetahui aplikasi tersebut menggunakan basis data atau tidak. Karena tidak diperlukan instalasi khusus. Basis data ini akan melakukan pekerjaan basis data relasional pada umumnya tanpa harus melakukan instalasi secara mandiri dan terpisah dari aplikasi yang sedang berjalan, namun basis data akan bekerja di dalam aplikasi. Salah satu keuntungan menggunakan basis data yang ada di dalam aplikasi adalah tidak perlu adanya konfigurasi jaringan atau administrasi yang diperlukan. Baik *klien* maupun *server* dijalankan bersama-sama dalam proses yang sama. Hal ini akan mengurangi *overhead* yang berkaitan dengan proses jaringan, penyederhanaan proses administrasi basis data, dan akan menjadi lebih mudah di dalam penyebaran aplikasi. Segala sesuatu yang diperlukan telah dikolaborasi bersamaan dengan aplikasi. Saat ini banyak sekali bahasa pemrograman yang dapat dikolaborasikan menggunakan *SQLite* seperti C/C++, PHP, Java (termasuk Android). Basis data dapat dipakai bersama-sama walaupun berada pada berkas yang sama. Masing-masing proses mewakili sebuh *server* secara independen. *SQLite* memanfaatkan sistem operasi untuk melakukan proses *sinkronisasi* dan penguncian data. Bahkan multi akses dapat dilakukan oleh bahasa pemrograman yang berbeda. Teknologi *SQLite* sangatlah bermanfaat karena merupakan sebuah basis data, perpustakaan pemrograman, serta berbagai baris perintah yang disediakan juga merupakan alat belajar yang baik dan cara yang baik untuk mengenalkan basis data *relasional*. Memang ada banyak cara yang disediakan untuk memanfaatkan jenis *embedded*. Untuk programmer, *SQLite* adalah seperti lakban *digital*, menyediakan cara yang mudah untuk mengikat aplikasi dengan data mereka dan tidak ada akhir untuk menggunakan potensinya. Di lingkungan *web*, *SQLite* dapat membantu dengan mengelola informasi sesi yang kompleks. Daripada menggunakan serialisasi pada pengelolaan sesi, potongan individu dapat ditulis secara selektif dan dibaca sebagai basis data individu. *SQLite* juga berfungsi sebagai basis data relasional yang baik untuk pengembangan dan pengujian. Selain sebagai media penyimpanan, *SQLite* dapat berfungsi sebagai alat murni fungsional serta pengelolaan data umum. Tergantung pada ukuran dan kompleksitas, mungkin lebih mudah mengoperasikan data aplikasi dalam struktur tabel. Dengan cara ini, dapat dioperasikan data relasional menggunakan *SQLite* untuk melakukan pekerjaan berat daripada harus menulis algoritma sendiri untuk memanipulasi dan mengurutkan struktur data.

GPS Photo Tagging, juga dikenal sebagai *'Geotagging'*, merupakan proses penambahan informasi posisi data *GPS* (Latitude, Longitude, Altitude) dalam sebuah *photo* digital. Ponsel-ponsel kamera yang memiliki *GPS receiver internal* umumnya memiliki fitur unik ini[6]. Dengan *GeoTagging* pada informasi digital *photo*, pada waktu yang lain pengguna dapat mengetahui letak pengambilan gambar *photo* tersebut. Hal ini juga menambah kategori baru untuk mempermudah pencarian berkas, semisal saat koleksi *photo* sudah mencapai ribuan, bisa mencarinya dengan kategori lokasi pemotretan. Atau bisa juga hal ini

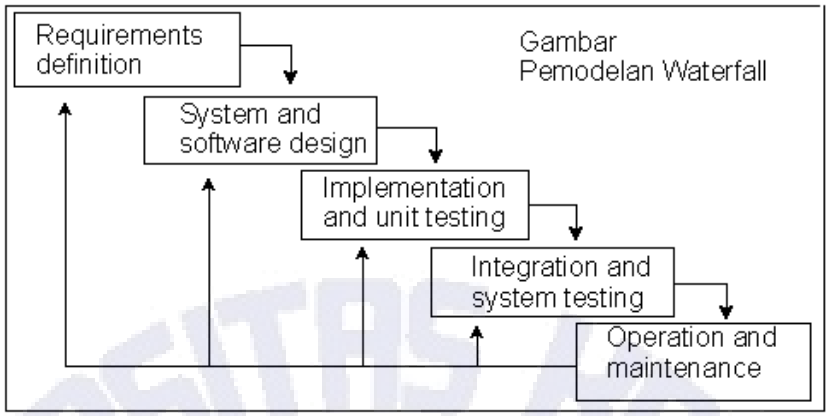
berlaku secara global di dunia *internet*. Di *internet*, untuk mempermudah pencarian pada sebuah '*Search Engine*', semisal Google, bisa dilakukan *entry tag* dengan kata-kata kunci yang menarik, semisal "Bill Gates" dan "Malaria". Demikian juga dengan *photo*, dengan melakukan sharing *photo* pada sebuah situs seperti *Flickr*, seseorang dapat melakukan *tag* pada *photo* yang diambilnya dengan memberikan informasi tentang siapa atau apa yang ada pada *photo* tersebut dan dimana diambilnya, termasuk *GeoTagging* yang bersifat otomatis ini. Orang lain dapat melakukan pencarian atau penjelajahan dengan *tag* untuk mencari konten yang paling relevan. *GPS phototagging* dapat mengurangi pekerjaan saat melakukan *tag* pada gambar di *internet*. Jika mengungah sebuah *photo tag* ke *Flickr*, situs tersebut secara otomatis akan membuat *map* dunia interaktif *photo* tersebut. Pengguna lain dapat mencari dan melihat dimana *photo* itu diambil dengan posisi lokasi yang tepat. *GPS phototagging* juga memungkinkan melakukan pengaturan pustaka *photo* personal. Pada masa lalu, jika ingin mencari sebuah *photo* terpisah, harus mengingat kapan *photo* itu diambil. Namun jika semua *photo* digital diberi *tag* dengan informasi lokasi, dapat mencarinya dengan mengetikkan sebuah lokasi dan melihat semua *photo* yang diambil pada lokasi tersebut. *GPS phototagging* juga memiliki aplikasi professional. Jika seorang pencari lokasi sebuah film, dapat dengan akurat memetakan semua *photo* yang diambil dalam sebuah perjalanan menuju suatu tempat. Bila sebagai agen sebuah *real estate* dapat menunjukkan pada *klien* dengan *map photo* interaktif dari semua properti yang dimiliki. Bahkan seorang Arkeolog dapat melakukan *'link'* sebuah *photo* dengan GIS *(Geographic Information System)* ketika sebuah rencana penggalian dilakukan. Mekanisme *GPS Photo Tagging* ketika mengambil sebuah *photo* menggunakan kamera (digital atau ponsel) yang memiliki *geotag*, kamera atau ponsel tersebut mencatat lebih banyak informasi atau data dibandingkan sebuah *photo*. Informasi tersebut termasuk waktu dan data ketika sebuah *photo* diambil, orientasi dari kamera (*Portrait* atau *Landscape*), apakah menggunakan lampu *flash* dan detail kamera lainnya yang digunakan seperti *Aperture*, *Exposure* dan *Focal Length*. Semua data ini disimpan pada sesuatu yg disebut *EXIF Header*. *EXIF (Exchangeable Image File Format) header* berisi petunjuk *photo* dengan data yang dapat dibaca oleh aplikasi *photo management* atau situs *photo*. Inilah sebabnya mengapa sebuah PC/Komputer secara otomatis dapat mengetahui dimana untuk meletakkan *photo* tersebut di sebuah *folder* tertentu. *EXIF Headers* adalah tersedia sebuah ruang untuk mengisi koordinat *Longitude*, *Latitude*, dan *Altitude*. Belakangan, perangkat keras dan perangkat lunak yang dapat menambahkan lokasi data pada *EXIF Headers* menggunakan teknologi *GPS*. Aplikasi *GPS Photo Tagging* kini telah ada kamera yang *Build-in* dengan *GPS receiver*. Seperti kamera Ricoh 500SE, yang setiap kali melakukan pengambilan *photo* data *GPS* akan ditulis pada *EXIF header photo* tersebut. Terdapat juga kamera digital yang *Build-In* dengan *chip GPS*. Atau dengan membeli *GPS Receiver* mini yang dapat ditambahkan pada kamera. Nikon juga telah menyediakan perangkat *GPS Phototagging* pada kamera SLR model terbarunya. Bahkan kini ponsel seperti Nokia N78, telah menerapkan kamera *Build-In* yg telah ada *chip GPS*. Jika mencari *GPS Phototagging* yang dapat bekerja pada semua kamera digital, dapat mengandalkan sebuah *GPS*

Ruteer. Cukup dengan menghidupkan kamera tersebut, kemudian ambil gambar. Sebelum mengunggah gambar tersebut ke PC/Komputer, ambil kartu memori dan masukkan ke *GPS Ruteer*. Semua data pada *GPS* secara otomatis akan ditulis pada setiap *EXIF Header* gambar yang diambil, selanjutnya dapat diunggah seperti biasa. Ada banyak situs yang mampu menampilkan *photo* dengan *geotagging* dan dapat di *sharing* dengan semua orang, *Flickr* adalah salah satunya. *Flickr* dapat melakukan pengaturan sederhana seperti memberi tanda *'Yes' pada 'Import EXIF location data*'. Situs lainnya adalah *Picasa* dan *Panoramio* (keduanya milik Google). Tidak hanya memungkinkan mengunggah *photo* dan membuat *map* interaktifnya, dengan menggunakan aplikasi *Google Earth*, dapat melakukan pembesaran dan menampilkan gambar 3 dimensi pada *map virtual*. Bedanya dengan *Flickr*, *Picasa* dan *Panoramio* akan memberikan opsi untuk membuat sebuah file berformat KML (*Keyhole Markup Language*) dari *photo* yang diunggah dan menampilkannya kembali lengkap dengan lokasi geografis pada permukaan planet *virtual*[7].

3. Metode Penelitian

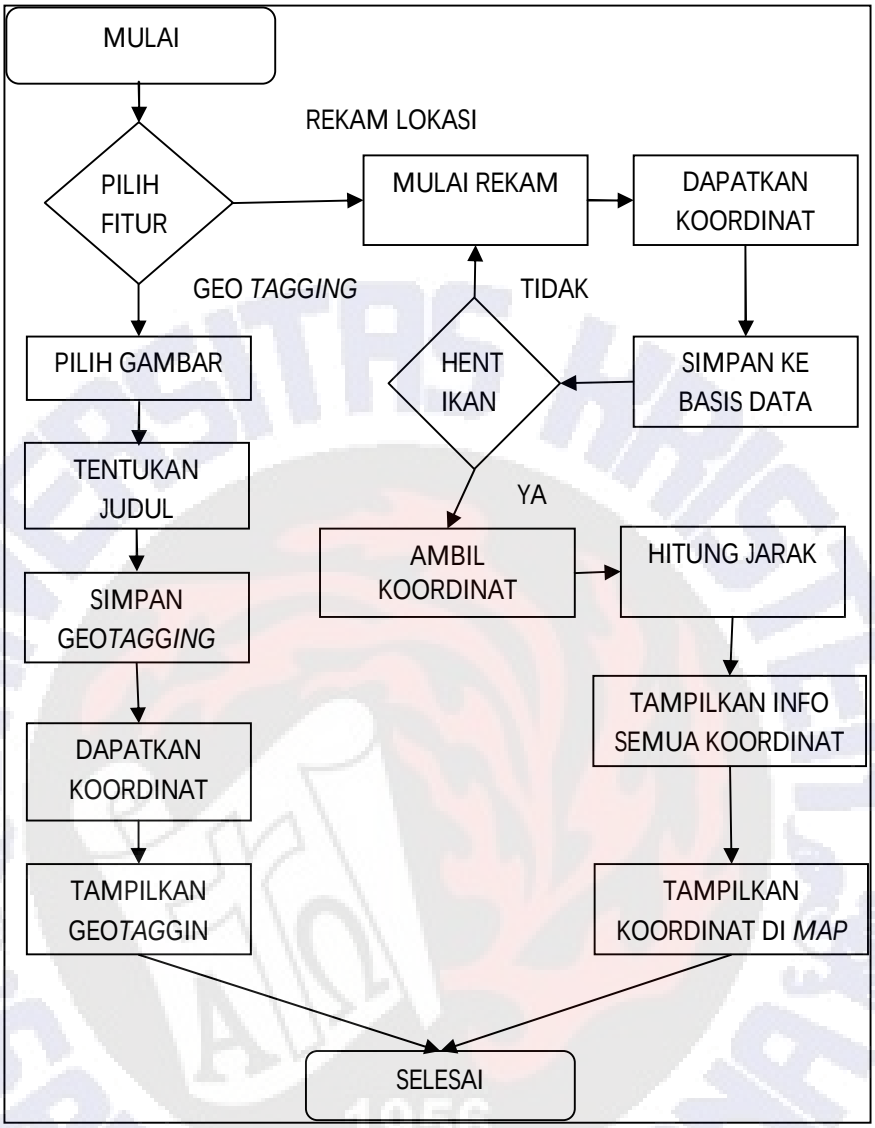
Metode pengembangan perangkat lunak yang dipakai di dalam penelitian ini adalah metode *waterfall*. Metode *waterfall* memiliki proses yang berjalan secara sekuen yaitu proses mengalir dari proses awal sampai dengan proses akhir. Alur di dalam *Waterfall* terdiri dari beberapa proses yaitu *Requirement Definition, System and Software Design, Implementation and Unit Testing* dan *Operaton and Maintenance*. Lebih tepatnya dapat melihat Gambar 1.

1. Pembuatan perangkat lunak dimulai dari tahap *requirements definition*, yaitu tahap analisis kebutuhan pengguna. Pengguna adalah orang-orang yang sering bepergian. Analisisi dilakukan dengan melakukan wawancara terhadap *traveller* yang tentunya akan mengetahui kebutuhan dari sistem berupa definisi. Adapun kebutuhan sistem yang didapatkan berdasarkan hasil wawancara adalah aplikasi dapat menyimpan titik dan gambar pada lokasi tertentu.Kemudian pengguna dapat menyimpan dan menampilkannya kembali pada *map*.
2. Tahap berikutnya adalah *system design* atau desain sistem. Pada tahap ini dilakukan desain sistem yaitu desain alur sistem dan desain basis data.
3. Tahap berikutnya adalah *implementasi*. Implementasi merupakan tahapan melakukan pengkodean program pada *platform* Android.
4. Tahap berikutnya adalah *system testing* atau pengujian sistem. Tahap pengujian sistem terhadap pengguna yaitu *traveller*.
5. Tahapan yang terakhir adalah *operation and maintenance* atau tahap perbaikan. Hasil implementasi dan pengujian akan membuat sistem perlu diperbaiki pada tahap ini.



Gambar 1 Metode *Waterfall*[8]

Pertama-tama aplikasi dijalankan akan masuk ke dalam menu aplikasi. Menu aplikasi terdiri dari mulai rekam, lihat hasil rekam, mulai *GeoTagging*, Hasil *GeoTagging*, informasi pembuat dan reset. Jika pengguna memilih mulai rekam maka akan masuk ke proses berikutnya yaitu ”Mulai Rekam”. Pada proses ”Mulai Rekam” aplikasi akan secara terus-menerus melakukan proses pengambilan koordinat *GPS* untuk kemudian koordinat tersebut ”disimpan ke dalam basis data”. Setelah disimpan ke dalam basis data akan selalu diperiksa apakah proses ini dihentikan oleh pengguna atau tidak. Jika belum dihentikan oleh pengguna, maka akan terus dilakukan mendapatkan koordinat *GPS* dan menyimpannya ke dalam basis data. Ketika pengguna menghentikan proses rekam, maka proses rekam yang terjadi akan dihentikan dan tidak mendapatkan posisi *GPS* lagi. Proses yang dilakukan berikutnya adalah proses mendapatkan semua data di dalam basis data yang sudah tersimpan. Ketika semua data telah terambil maka dihitung satu-per satu jarak antar titik. Jarak setiap antar titik disimpan sementara, kemudian setiap titik yang didapatkan ditampilkan semua sebagai informasi pengguna untuk kemudian dari dalam basis data titik tersebut dimasukkan ke dalam *Google Map*. Setiap titik yang ditekan akan memunculkan informasi koordinat dan informasi jarak yang telah dihitung sebelumnya. Pada tahapan ini, proses rekam dan lihat hasil selesai. Untuk menu selanjutnya adalah mulai *GeoTagging*, proses yang terjadi adalah memilih *photo* dari *library*, kemudian memberi nama pada *photo*, proses yang dilakukan berikutnya adalah menyimpan *photo*, kemudian mengambil koordinat *photo* dan menampilkanya di *Google Map*, menu yang lainnya adalah menu informasi pembuat untuk menampilkan informasi pembuat aplikasi ini, sedangkan menu reset digunakan untuk menghapus data dari basis data.



Gambar 2 *FlowChart*

Keperluan data aplikasi hanya perlu menyimpan satu jenis data yaitu data koordinat *map*. Di mana nantinya data tersebut disimpan ke dalam basis data *SQLite* dengan struktur table seperti pada tabel.

Tabel 1. Desain Data Tabel Posisi

Nama *Field*	Tipe Data	Keterangan
Id	*Integer*	*Primary key auto increment*
Nama	Text	Nama posisi
Latitude	Text	Lokasi *GPS*
Longitude	Text	Lokasi *GPS*

Tabel posisi pada Tabel 1 dapat dijelaskan bahwa *Field* "*id*" digunakan sebagai *primary key* atau *field* yang unik dan digunakan sebagai pembeda setiap data. Nilai "*id*" dibuat *auto increment* supaya tidak perlu menentukan nilainya.

Nilai yang dimasukkan akan dibuat secara otomatis dan ditentukan oleh *SQLite* basis data. *Field* "nama" digunakan untuk menyimpan nama posisi. Nama posisi digunakan untuk mengelompokkan pengambilan posisi. Sehingga bisa digunakan untuk mencatat beberapa kelompok posisi pada saat yang berbeda. *Field "latitude"* adalah *field* yang digunakan untuk menyimpan posisi *GPS* dengan tipe *latitude*. *Field latitude* menggunakan tipe data text karena agar mudah ditampilkan di dalam *map* serta mudah dilemparkan sebagai parameter ke dalam *Intent* yang lain. Untuk keperluan perhitungan jarak, akan dikonversi menjadi nilai *double* oleh kode program aplikasi. *Field "longitude"* adalah field yang digunakan untuk menyimpan posisi *GPS* dengan tipe *longitude*. *Field longitude* menggunakan tipe data *text* karena agar mudah ditampilkan di dalam *map* serta mudah dilemparkan sebagai parameter ke dalam *Intent* yang lain. Untuk keperluan perhitungan jarak, akan dikonversi menjadi nilai *double* oleh kode program aplikasi.

Tabel 2. Desain Data Tabel *Tag*

Nama *Field*	Tipe Data	Keterangan
Id	*Integer*	*Primary key auto increment*
Uri	Text	Lokasi gambar
Nama	Text	Nama *Tag*
Latitude	Text	Lokasi *GPS*
Longitude	Text	Lokasi *GPS*

Tabel posisi pada Tabel 2 dapat dijelaskan bahwa *field* "*id*" digunakan sebagai *primary key* atau *field* yang unik dan digunakan sebagai pembeda setiap data. Nilai "*id*" dibuat *auto increment* supaya tidak perlu menentukan nilainya ketika melakukan penambahan data. Nilai yang dimasukkan akan dibuat secara otomatis dan ditentukan oleh *SQLite* basis data. *Field* "*uri*" digunakan untuk menyimpan lokasi gambar pada posisi tertentu. Gambar sebenarnya tidak disimpan di dalam basis data, namun lokasi gambarlah yang disimpan di dalam basis data. *Field* "nama" digunakan untuk menyimpan nama dari lokasi yang disimpan. Nama setiap lokasi akan disimpan dan ditampilkan kembali seiring dengan gambar. *Field* "*latitude*" dan "*longitude*" digunakan untuk menyimpan lokasi gambar dalam koordinat *GPS*. Nantinya akan dipakai di dalam *Google Map* untuk menentukan lokasi gambar di dalam *map*.

4. Hasil dan Pembahasan

Kode program 1 merupakan kode program pembuatan *thread* sehingga program dapat secara berkala melakukan *record* data pada satuan waktu tertentu. Lokasi *GPS* didapatkan dengan menggunakan kelas *LocationManager* yang ada pada baris 5 dan 6. Kemudian jeda setiap pengambilan data lokasi *GPS* digunakan method sleep(10000) yang ada pada baris 32. 10000 memiliki arti 10000ms atau 10 detik. Sehingga setiap 10 detik sekali posisi *GPS* pengguna akan disimpan.

Kode Program 1 Proses Penyimpanan Lokasi

```
1. public void run() {
2.      jalan = true;
3.      while (jalan) {
4.              Location location =
```

```
5.          locationManager.getLastKnownLocation(
6.          LocationManager.GPS_PROVIDER);
7.             if (location != null) {
8.                    boolean baru = true;
9.                    for (Position p:list) {
10.                          if (
11.                    p.latitude.equals(
12.                    ""+location.getLatitude())
13.                          &&
14.                    p.longitude.equals(
15.                    ""+location.getLongitude())) {
16.                                 baru = false;
17.                          }
18.                   }
19.
20.                   if (baru) {
21.                          Position p = new Position();
22.                          p.nama = Main.nama;
23.                          p.latitude =
24.                    ""+location.getLatitude();
25.                          p.longitude =
26.                    ""+location.getLongitude();
27.                          list.add(p);
28.                   }
29.            }
30.
31.            try {
32.                   Thread.sleep(10000);
33.            } catch (InterruptedException e) {
34.                   e.printStackTrace();
35.            }
36.    }
37.}
```

Gambar 3 merupakan tampilan ketika proses *record* berjalan. Ketika proses berjalan maka akan ditampilkan *progress dialog* yang akan hilang jika pengguna sudah menekan tombol *back* pada perangkat Android.

Gambar 3. Tampilan *Record* Lokasi

Kode program 2 merupakan kode program untuk menampilkan hasil *record* lokasi. Hasil *record* ditampilkan ke dalam *mapView*. Pertama kali perlu melakukan *clear* pada *overlay* agar *marker* sebelumnya dihilangkan. Berikutnya dengan melakukan perulangan dari basis data, gambarkan setiap *marker* atau titik penyimpanan lokasi.

Kode Program 2. Menampilkan Hasil *Record*

```
1. mapOverlays.clear();
2.
3.      int i = 1;
4.      double totalDistance = 0;
5.      Position temp = new Position();
6.      for (Position p : poses) {
7.              if (i == 1) {
8.                      temp = p;
9.                      GeoPoint point1 = new GeoPoint(
10.                             (int)(Double.parseDouble(
11.                           temp.latitude) * 1E6),
12.                             (int) (Double.parseDouble(
13.                           temp.longitude) * 1E6));
14.                  mapOverlays.add(
15.                  new Titik("0 km", point1));
16.             } else {
17.                     double distance = round(
18.                  CalculationByDistance(
19.                             poses.get(i - 2).latitude,
20.                         poses.get(i - 2).longitude,
21.                             p.latitude, p.longitude),2);
22.
23.                          totalDistance += distance;
24.
25.                     GeoPoint point1 = new GeoPoint(
26.                             (int) (Double.parseDouble(
27.                       temp.latitude) * 1E6),
28.                             (int) (Double.parseDouble(
29.                       temp.longitude) * 1E6));
30.                     GeoPoint point2 = new GeoPoint(
31.                             (int) (Double.parseDouble(
32.                        p.latitude) * 1E6),
33.                             (int) (Double.parseDouble(
34.                           p.longitude) * 1E6));
35.
36.             DrawRoute(
37.                   point1, point2, Color.GREEN, mapView);
38.             mapOverlays.add(
39.                new Titik(distance + " km", point2));
40.             }
41.             i++;
42.     }
43.     Toast.makeText(
44.             this,
45.             "Total Jarak: " + totalDistance
46.              + " km dari " + poses.size()
47.                 + " titik.", Toast.LENGTH_SHORT).show();
```

Gambar 4 merupakan tampilan ketika perhitungan jarak telah selesai. Sesuai tampilan gambar, terdapat total jarak dan jumlah titik yang telah dicatat oleh aplikasi. Dari setiap titik dihitung jarak serta digambarkan rutenya. Hasilnya adalah setiap titik ditampilkan beserta dengan informasi jarak dan setiap titik dihubungkan oleh rute jalan berwarna hitam.

Gambar 4. Hasil Perhitungan Jarak dan Titik

Kode program 3 merupakan kode program untuk menampilkan pemilihan gambar yang akan dijadikan *GeoTagging*. Untuk menampilkan *activity* pemilihan *library* gambar dapat memanggil *Android.Images.Media*.

Kode Program 3. Memilih Gambar dari *Library*

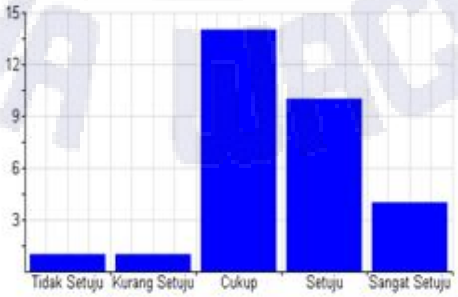
```
1. Intent intent = new Intent(Intent.ACTION_PICK,
2.         Android.provider.MediaStore.Images.Media.
3.        EXTERNAL_CONTENT_URI);
4. startActivityForResult(intent, SELECT_PICTURE);
5.
6. public void onActivityResult(
7.    int requestCode, int resultCode, Intent data) {
8.     super.onActivityResult(
9.      requestCode, resultCode, data);
10.
11.      if (requestCode == SELECT_PICTURE) {
12.            if (data != null) {
13.            Uri targetUri = data.getData();
14.
15.                  finish();
16.                  Intent intent =
17.                 new Intent(this, SetGeoTag.class);
18.                  intent.putExtra("uri",
19.                 targetUri.toString());
20.                  startActivity(intent);
21.            }
22.     }}
```

Gambar 5 merupakan hasil dari proses *GeoTagging*. Setelah pengguna memilih gambar, maka akan ditangani oleh method *onActivityResult* di mana jika data yang dipilih tidak *null* (gambar ada yang dipilih) maka disimpan *Uri* atau lokasi gambar yang dipilih. Setelah *uri* gambar terpilih didapatkan, selanjutnya memanggil *activity* baru untuk ditampilkan yaitu *activity SetGeoTag* yang berguna untuk menampilkan gambar yang dipilih pengguna beserta inputan nama *GeoTagging*.

Gambar 5. *GeoTagging*

Hasil pengujian dari peneltian ini diujikan kepada wisatawan atau *traveller* dengan menggunakan kuesioner, dan mendapatkan hasil jumlah tertinggi yang ada di setiap pertanyaan adalah 46.67% menyebutkan aplikasi cukup bermanfaat, kemudian 30% menyatakan aplikasi telah berjalan dengan baik, 46.67% menyatakan aplikasi cukup digunakan dengan dengan mudah, 46.67% menyatakan setuju aplikasi telah menampilkan yang sesuai, dan 40% menyatakan setuju aplikasi yang dibuat dapat menyimpan data yang sesuai. Hasil grafik dari kuesioner adalah sebagai berikut:

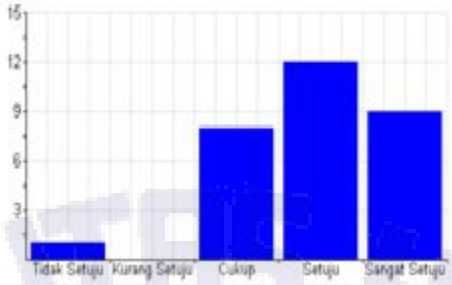
Pada pertanyaan pertama "Apakah Aplikasi yang dibuat bermanfaat?" sesuai dengan Gambar 6. hasilnya 1 orang atau 3.33% menyatakan tidak setuju, 1 orang atau 3.33%, 14 orang atau 46.67% menyatakan cukup, 10 orang atau 33.33% menyatakan setuju dan 4 orang atau 13.33% menyatakan sangat setuju.



Gambar 6. Grafik Pertanyaan Pertama

Untuk pertanyaan kedua "Apakah Aplikasi yang dibuat berjalan dengan baik?" sesuai dengan Gambar 7 hasilnya tidak setuju sebanyak satu orang atau 3.33%, kurang setuju tidak ada atau 0%, delapan orang atau 26.67% menyatakan

cukup, duabelas orang atau 40% menyatakan setuju dan sembilan orang atau 30% menyatakan sangat setuju.

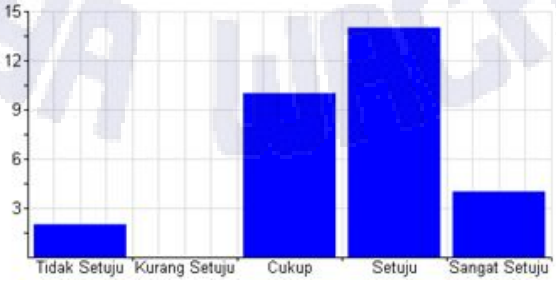


Gambar 7. Grafik Pertanyaan Kedua

Untuk pertanyaan ketiga "Apakah Aplikasi yang dibuat dapat digunakan dengan mudah?" sesuai pada Gambar 8 hasilnya satu orang atau 3.33% menyatakan tidak setuju, satu orang atau 3.33% menyatakan kurang setuju, sepuluh orang atau 33.33% menyatakan cukup, empatbelas orang atau 46.67% menyatakan setuju dan empat orang atau 13.33% menyatakan sangat setuju.

Gambar 8. Grafik Pertanyaan Ketiga

Untuk pertanyaan keempat "Apakah Aplikasi yang dibuat memberikan hasil yang sesuai?" hasilnya dua orang atau 6.67% menyatakan tidak setuju, tidak ada yang menyatakan kurang setuju atau 0%, sepuluh orang atau 33.33% menyatakan cukup, empatbelas orang atau 46.67% menyatakan setuju dan empat orang atau 13.33% menyatakan sangat setuju.



Gambar 9. Grafik Pertanyaan Keempat

Untuk pertanyaan kelima "Apakah Aplikasi yang dibuat dapat menyimpan data yang sesuai?" sesuai pada Gambar 10 hasilnya 3 orang atau 10% menyatakan tidak setuju, satu orang atau 3.33% menyatakan kurang setuju,

sembilan orang atau 30% menyatakan cukup, duabelas orang atau 40% menyatakan setuju dan lima orang atau 16.67% menyatakan sangat setuju.

Gambar 10. Grafik Pertanyaan Kelima

5. Simpulan

Berdasarkan proses penelitian yang telah dilakukan maka dapat diambil kesimpulan proses implementasi LBS (*Location Based Service*) yang sesuai dengan rumusan masalah pertama, dapat dilakukan dengan baik. Hal tersebut dilakukan dengan memanfaatkan fitur *GPS* di dalam perangkat dan kemudian ditampilkan ke dalam *Google Map*. Proses perhitungan jarak dan menampilkan rute setiap titik juga sudah berjalan dengan baik sesuai dengan pengujian. Proses implementasi *GeoTag* yang sesuai dengan rumusan masalah kedua, dapat dilakukan dengan baik. Hal tersebut dilakukan dengan memanfaatkan *Activity Image Library* dan penggambaran *Bitmap* ke dalam *Google Map* sesuai lokasi *GPS*. Proses *GeoTag* telah berjalan dengan baik sesuai dengan hasil pengujian yang telah dilakukan.

6. Daftar Pustaka

[1] Works, It. 2012. *Seri Creative Project: Membedah Kehebatan Android*. Jakarta: Grasindo.

[2] Murphy. 2010. *Beginning Android*. United States of America: Apress.

[3] Edy, Kurniawan. 2011. *Pembuatan dan Desain Agen Pencarian Hotel Berbasis Android*. Salatiga: FTI-UKSW.

[4] Jerome. 2008. *Beginning Android 2*. United States of America: Apress.

[5] Purvis. 2006. *Google Maps Programming*. United States of America: Apress.

[6] Owens, Michael. 2006. *Definitive Guide to SQLite*. United States of America: Apress.

[7] Meier, Reto. 2012. *Professional Android Application Development*. United States of America: Wrozz Press.

[8] Jalote, Pankaj. 2002. *Software Project Management in Practice*. United States of America: Addison Wisley.