BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi membawa dampak yang signifikan terhadap kehidupan manusia. Hampir semua bidang kehidupan manusia sekarang ini sudah memakai teknologi informasi sebagai basisnya. Pendidikan, kesehatan, keamanan, transportasi dan berbagai bidang yang lain sekarang ini sudah menggunakan teknologi informasi sebagai alat bantu yang memiliki peran sangat penting. Istilah - istilah yang berkaitan dengan teknologi informasi sudah sangat familiar di semua kalangan masyarakat. Internet, komputer, *online*, *Facebook*, *Twitter*, *E-mail*, *chatting* adalah beberapa istilah yang populer dan sering ditemui dimana – mana dewasa ini. Beberapa hal tersebut adalah sedikit dari banyak bukti bahwa teknologi informasi sudah "mendarah daging" dalam kehidupan manusia.

Trend penggunaan teknologi informasi dalam kehidupan manusia ini menciptakan banyak disiplin ilmu baru. *Web*, Android, dan GPS adalah salah satu contoh dari banyak disiplin ilmu baru dan terus berkembang sampai sekarang. Tidak hanya muncul disiplin ilmu baru, disiplin ilmu yang sebelumnya sudah ada pun menjadi semakin kaya dan modern dengan implementasi teknologi informasi di dalamnya. Android dan GPS *(Global Positioning System)* adalah salah satu contoh dari disiplin ilmu yang semakin berkembang dengan penggunaan teknologi informasi di dalamnya. Sekarang teknologi informasi sudah menjadi hal yang sangat vital dalam kehidupan manusia.

Di dalam kantor Badan Pertanahan Nasional Indonesia (BPN-RI) sistem pendataan dan pemetaan lahan yang dilakukan masih menggunakan cara yang manual. Walaupun dalam pelaksanaannya sudah memanfaatkan teknologi modern yaitu dengan menggunakan alat GPS, namun pada prakteknya dilapangan teknologi tersebut tidak dimanfaatkan secara optimal. Alat GPS hanya digunakan sebagai pendeteksi atau pencarian titik koordinat saja. Sedangkan alat GPS bisa dimanfaatkan lebih dari itu yaitu menunjukkan lokasi, lama perjalanan yang akan

ditempuh, seberapa banyak bahan bakar yang akan dibutuhkan untuk menempuh suatu perjalanan, dan berbagai manfaat lainnya.

Jika petugas atau *user* akan melakukan pendataan dan pemetaan lahan, petugas atau *user* akan datang ke lahan yang dituju untuk mencari dan mencatat titik koordinat pada sudut-sudut lahan dengan menggunakan GPS dan selembar kertas. Setelah itu petugas kembali ke kantor untuk memasukkan titik koordinat yang diperoleh ke dalam Citra (sebutan google map di kantor BPN). Dengan menggunakan Citra akan diperoleh peta gambar dari lokasi atau lahan.

Selain masalah sistem pendataan dan pemataan yang rumit, petugas atau *user* BPN juga mengalami keterbatasan fasilitas. Contoh di kantor BPN Kota Pati hanya terdapat 12 buah alat GPS, sedangkan pegawai lapangan BPN lebih dari 40 orang. Sehingga petugas atau *user* yang tidak kebagian alat GPS harus bermodal sendiri untuk mendapatkan fasiltas tersebut.

Berdasarkan latar belakang tersebut penulis mengambil judul "Pendataan dan Pemetaan Lahan Menggunakan GPS *Tracker* Berbasis Android Pada Badan Pertanahan Nasional". Sebuah aplikasi GPS yang ada dalam *Smartphone* bersistem operasi Android yang mempunyai fungsi sama dengan alat GPS yaitu untuk menentukan letak koordinat yang akan dituju, dengan harapan perangkat lunak ini nantinya dapat membantu dan mempermudah pegawai atau *user* Badan Pertanahan Nasional dalam usahanya melayani kebutuhan masyarakat untuk pendataan dan pemetaan lahan (studi kasus pada kantor Badan Pertanahan Nasiaonal).

1.2 Rumusan Masalah

Berdasarkan permasalahan yang diangkat pada bagian latar belakang, maka rumusan masalah dikhususkan pada :

1. Bagaimana penjelasan tentang gambaran umum perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan?
2. Bagaimana pembahasan perancangan dan rekayasa perangkat lunak dari perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan?

3. Bagaimana pembahasan teknis implementasi pemrograman dan pengembangan perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan menggunakan bahasa pemrograman Java, PHP, HTML, dan CSS?
4. Bagaimana pembahasan tentang skenario, proses dan hasil pengujian perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan?

1.3 Batasan Masalah

Agar diperoleh hasil pembahasan yang sesuai dengan apa yang diharapkan, maka perlu diberikan batasan masalah pada pengembangan perangkat lunak ini, yaitu :

1. Pembahasan difokuskan pada rekayasa perangkat lunak dari pembuatan perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan.
2. Sumber data yang digunakan adalah sumber data yang diperoleh dari kantor Badan Pertanahan Nasional .
3. Pengembangan perangkat lunak dilakukan dalam lingkungan sistem operasi Microsoft Windows 7 Home Premium 64-bit.
4. IDE (*Integrated Development Environment*) yang dipakai dalam pengembangan perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan adalah Eclipse, Emulator Android, dan Dreamweaver.
5. GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan menggunakan perangkat mobile berbasis Android minimal versi 4.0 ICS *(Ice Cream Sandwich).*
6. Pengembangan perangkat lunak dilakukan dengan bahasa pemrograman Java, PHP, HTML, dan CSS.
7. Versi platform JDK (*Java Development Kit*) yang dipakai adalah JDK 7 Update 21.

8. DBMS (*Database Management System*) yang digunakan dalam pengembangan perangkat lunak ini adalah MySQL dan SQLite.
9. Performa penguncian posisi koordinat *user* tergantung performa GPS dari device.

1.4 Tujuan

Sesuai dengan latar belakang dan rumusan masalah, tujuan dari pengembangan perangkat lunak ini adalah sebagai berikut:

1. Untuk mengetahui gambaran umum perangkat lunak GPS *Tracker* berbasis Android dalam melakukan pendataan dan pemetaan lahan.
2. Untuk mewujudkan perancangan dan rekayasa perangkat lunak dari perangkat lunak GPS *Tracker* berbasis Android dalam melakukan pendataan dan pemetaan lahan.
3. Untuk mewujudkan teknis implementasi pemrograman dan pengembangan perangkat lunak GPS *Tracker* berbasis Android dalam melakukan pendataan dan pemetaan lahan menggunakan bahasa pemrograman Java, PHP, XML, HTML, dan CSS.
4. Untuk memperoleh skenario, proses dan hasil pengujian perangkat lunak GPS *Tracker* berbasis Android dalam melakukan pendataan dan pemetaan lahan.
5. Untuk menghasilkan sebuah dokumen sebagai dokumentasi dari perangkat lunak ini sendiri.

1.5 Manfaat

Manfaat yang nantinya dapat diambil dari pengembangan perangkat lunak ini adalah sebagai berikut :

a. Bagi penulis
    1. Menerapkan ilmu yang telah didapatkan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

2. Mendapatkan pemahaman lebih lanjut dalam pengembangan perangkat lunak GPS *Tracker* berbasis Android Pada Badan Pertanahan Nasional .
3. Sebagai salah satu referensi belajar bagi mahasiswa untuk meningkatkan pengetahuannya dalam bidang perancangan khususnya perangcangan aplikasi perangkat lunak.

b. Bagi pengguna

1. Menyediakan aplikasi yang dapat digunakan untuk pendataan dan pemetaan lahan dengan menggunakan teknologi GPS *Tracker* berbasis Android pada Badan Pertanahan Nasional .
2. Untuk menekan biaya dan mempermudah tugas/pekerjaan pegawai atau *user* Badan Pertanahan Nasional dalam pendataan dan pemetaan lahan.

1.6 Sistematika Pembahasan

Sistematika pembahasan ditunjukkan untuk memberikan gambaran dan uraian dari penulisan skripsi ini secara garis besar yang meliputi beberapa bab, sebagai berikut :

Bab I : Pendahuluan

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

Bab II : Kajian Pustaka dan Dasar Teori

Menguraikan tentang dasar teori dan referensi yang mendasari pengembangan perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan.

Bab III : Metode Penelitian dan Perancangan

Menguriakan dan membahas langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literature, implementasi, perancangan, analisis kebutuhan dan pengujian

perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan sesuai dengan dasar teori yang ada.

Bab IV : Implementasi

Membahas implementasi dari perangkat lunak GPS *Tracker* Berbasis Android untuk pendataan dan pemetaan lahan sesuai dengan perancangan perangkat lunak yang telah dibuat.

Bab V : Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap perangkat lunak yang telah direalisasikan.

Bab VI : Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini berisi pembahasan tentang teori dasar yang berhubungan dengan pengembangan perangkat lunak yang dilakukan. Teori dasar yang akan dibahas pada bab ini yaitu penjelasan tentang konsep dasar perangkat lunak, pengembangan dan rekayasa perangkat lunak, konsep *Component-Based Software Engineering*, konsep dasar *Unified Modelling Language* yang dipakai pada rekayasa perangkat lunak, konsep dasar teknik pengujian perangkat lunak, penjelasan tentang *Web Service*, penjelasan tentang bahasa pemrograman HTML dan CSS, penjelasan tentang bahasa pemrograman PHP, dan penjelasan tentang bahasa pemrograman Java. Konsep dan implementasi basis data, penjelasan tentang MySQL, penjelasan tentang SQLite, penjelasan tentang Google Maps API, pengenalan tentang sistem operasi Android, teknologi GPS dan A-GPS yang digunakan.

2.1. Perangkat Lunak

Banyak orang menyamakan istilah perangkat lunak dengan program komputer. Sesungguhnya, pandangan ini terlalu dangkal. Perangkat lunak tidak hanya mencakup program, tetapi juga semua dokumentasi dan konfigurasi data yang berhubungan, yang diperlukan untuk membuat agar program beroperasi dengan benar. Sistem perangkat lunak biasanya terdiri dari sejumlah program yang terpisah, *file – file* konfigurasi yang digunakan untuk membuat program – program ini, dokumentasi sistem yang mendeskripsikan struktur sistem dan dokumentasi *user* yang menjelaskan bagaimana penggunaan sistem tersebut, dan untuk produk – produk perangkat lunak disediakan situs *web*, agar user bisa men-*download* informasi produk terbaru [SOM-03:5].

Perekayasa perangkat lunak bertugas mengembangkan produk perangkat lunak, yaitu perangkat lunak yang dapat dijual ke pelanggan. Secara umum ada dua tipe produk perangkat lunak, yaitu produk generik dan produk pesanan.

2.1.1. Produk generik

Produk generik merupakan sistem *stand-alone* (berdiri sendiri) standar yang diproduksi oleh organisasi pengembang dan dijual pada pasar terbuka ke siapapun yang bisa membelinya. Kadangkala perangkat lunak ini disebut sebagai perangkat lunak *shrink-wrapped* (dikecilkan dan dikemas). Contoh jenis produk ini adalah *database*, pengolah kata (*word processor*), paket untuk menggambar dan alat bantu manajemen proyek.

2.1.2. Produk pesanan

Produk pesanan (yang disesuaikan) merupakan sistem – sistem yang dipesan oleh pelanggan tertentu. Perangkat lunak dikembangkan khusus bagi pelanggan tersebut oleh kontraktor perangkat lunak. Beberapa contoh perangkat lunak jenis ini adalah sistem kontrol untuk piranti elektronik, sistem yang ditulis untuk mendukung proses tertentu dan sistem kontrol lalu lintas udara.

Perbedaan penting antara tipe – tipe perangkat lunak ini adalah pada produk generik organisasi yang mengembangkan perangkat lunak mengontrol spesifikasi perangkat lunak, sedangkan untuk produk pesanan spesifikasi biasanya dikembangkan dan dikontrol oleh organisasi yang membeli perangkat lunak tersebut. Pengembang perangkat lunak harus bekerja sesuai dengan spesifikasi tersebut.

2.2. Rekayasa Perangkat Lunak

Ian Sommerville dalam bukunya yang berjudul *Software Engineering* menyebutkan bahwa rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan.

Pada definisi ini, ada dua istilah kunci yaitu *'disiplin rekayasa'* dan *'semua aspek produksi perangkat lunak'*.

1. *'disiplin rekayasa'*

   Perekayasa membuat suatu alat bekerja. Mereka menerapkan teori, metode, dan alat bantu yang sesuai, selain itu mereka menggunakannya dengan selektif dan selalu mencoba mencari solusi

terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung. Perekayasa juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan – batasan ini.

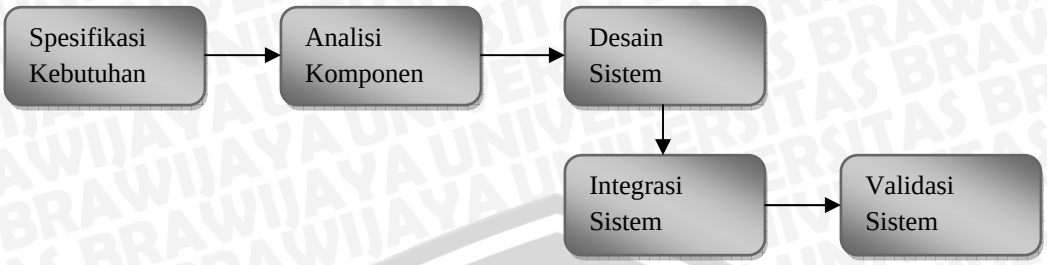
2. *'semua aspek produksi perangkat lunak'*

   Rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode dan teori untuk mendukung produksi perangkat lunak.

Secara umum, perekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun demikian, rekayasa ini sebenarnya mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan [SOM-03:7].

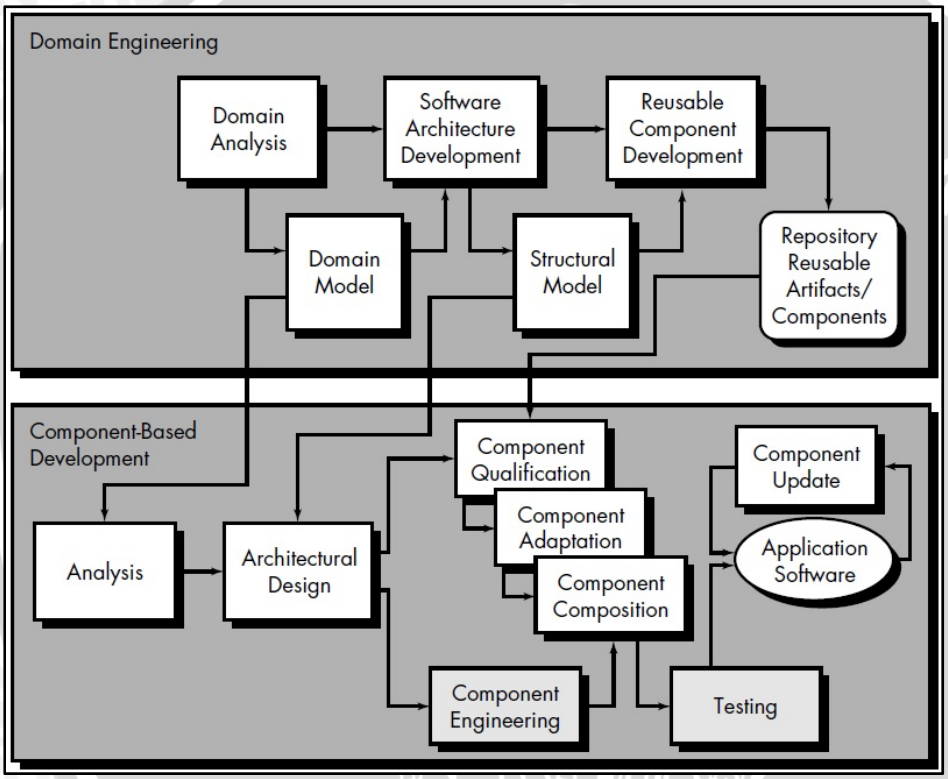
2.3. *Component-Based Software Engineering*

Component-Based Software Engineering (CBSE) adalah metode pengembangan perangkat lunak yang menekankan pada perancangan dan pembangunan perangkat lunak dengan menggunakan komponen perangkat lunak yang sudah ada dan bersifat dapat digunakan kembali. [PRE-01:721]. CBSE memiliki *software process* yang terdiri dari dua bagian utama yang berjalan paralel, yaitu *domain engineering* dan *component-based development.* Alur *software process* dari CBSE digambarkan pada Gambar 2.1 dan 2.2.



Gambar 2.1 *Software process* dari CBSE

Sumber : [PRE-01:725]



Gambar 2.2 *Software process* dari CBSE

Sumber : [PRE-01:725]

2.3.1. *Domain Engineering*

Domain Engineering bertujuan untuk mengidentifikasi, membangun, mengumpulkan, dan mengalokasikan sekumpulan komponen perangkat lunak yang dapat dipakai pada pengembangan perangkat lunak pada saat ini dan di masa yang akan datang.

2.3.1.1. *Domain Analysis*

Pada tahap *domain analysis* akan dilakukan penggalian informasi terkait perangkat lunak yang akan dikembangkan melalui metode analisis kebutuhan sesuai dengan teknik *software analysis*.

2.3.1.2. *Domain Model*

Dalam *Domain Model* akan dilakukan pemodelan terhadap hasil *domain analysis* yang telah dilakukan sebelumnya. Pemodelan dapat dilakukan dengan *unified modelling language* untuk analisis yang bersifat *object-oriented*.

2.3.1.3. *Software Architecture Development*

Software architecture development merupakan tahap dimana terjadi perancangan pola arsitektur perangkat lunak yang nantinya akan dipakai pada pengembangan perangkat lunak yang akan dibangun.

2.3.1.4. *Structural Model*

Structural model adalah tahapan dimana dilakukan analisis lebih lanjut tentang pemodelan struktural perangkat lunak berdasarkan perancangan pola arsitektur perangkat lunak yang telah dilakukan. Analisis ini umumnya dilakukan dengan melakukan *break down* modul – modul fungsional dari perangkat lunak yang akan dibangun.

2.3.1.5. *Reusable Components Development*

Pada tahap *reusable component development* dilakukan proses identifikasi dan pencarian kebutuhan *software component* yang akan digunakan dalam membangun perangkat lunak.

2.3.1.6. *Repository Reusable Artifacts / Components*

Repository reusable artifacts / components adalah tahapan dimana komponen – komponen perangkat lunak dikumpulkan dan disimpan dalam bentuk *repository* sehingga siap diimplementasikan pada bagian *component-based development* atau dapat digunakan kembali di masa yang akan datang.

2.3.1.7. *Component-Based Development*

Component-based development bertujuan untuk melakukan pembangunan sistem perangkat lunak dengan menggunakan komponen – komponen perangkat lunak yang bersifat *reusable*.

2.3.1.8. *Analysis*

Pada tahap *analysis* ini akan dilakukan analisis kebutuhan lebih lanjut terkait dengan kebutuhan komponen – komponen yang dibutuhkan untuk membangun perangkat lunak berdasarkan analisis kebutuhan fungsional.

2.3.1.9. *Architectural Design*

Tahap *architectural design* merupakan tahapan perancangan arsitektural dari perangkat lunak yang akan dibangun. Perancangan arsitektural ini dilakukan berdasarkan pemodelan kebutuhan fungsionalitas perangkat lunak dan kebutuhan komponen yang telah dilakukan. Dalam perancangan arsitektural ini akan dijelaskan arsitektur perangkat lunak yang akan dibangun, seperti arsitektur penyimpanan data dan arsitektur *class* terkait fungsionalitas perangkat lunak. Umumnya, perancangan arsitektural ini dilakukan dengan *unified modelling language* apabila menggunakan *object-oriented analysis.*

2.3.1.10. *Component Qualification*

Pada tahap *component qualification* akan dijelaskan tentang kualifikasi dari komponen perangkat keras dan komponen perangkat lunak yang dibutuhkan dalam pengembangan perangkat lunak yang akan dibangun. Tahap kualifikasi komponen ini bertujuan untuk mengetahui apakah komponen yang digunakan telah sesuai dan memenuhi syarat - syarat dari kebutuhan perangkat lunak yang dibangun.

2.3.1.11. *Component Adaptation*

Dalam tahap *component adaptation* dilakukan proses adaptasi dan modifikasi dari komponen – komponen perangkat lunak yang dipakai sehingga memenuhi kualifikasi kebutuhan komponen dari perangkat lunak yang dibangun.

2.3.1.12. *Component Engineering*

Tahap *component engineering* adalah tahap dimana komponen – komponen dari perangkat lunak yang dibangun, dikembangkan secara mandiri. Pengembangan komponen baru ini umumnya dilakukan dengan teknik *object-oriented programming.*

2.3.1.13. *Component Composition*

Tahap *component composition* adalah tahap utama dimana dilakukan pembangunan perangkat lunak dengan menggunakan komponen- komponen yang sudah ada. Pada tahap ini dilakukan penggabungan komponen – komponen perangkat lunak menjadi satu perangkat lunak yang padu.

2.3.1.14. *Testing*

Tahap *testing* adalah tahap dilakukannya proses pengujian perangkat lunak yang dibangun menggunakan teknik dan metode pengujian perangkat lunak tertentu. Pengujian dilakukan untuk mengetahui apakah fungsionalitas perangkat lunak sudah benar dan dapat memenuhi kebutuhan pengguna.

2.3.1.15. *Application Software*

Proses distribusi perangkat lunak dilakukan pada tahap *application software*. Perangkat lunak yang telah jadi dan telah diuji didistribusikan kepada pengguna atau klien untuk dapat dipakai sebagaimana mestinya.

2.3.1.16. *Component Update*

Pada tahap ini dilakukan pembaharuan (*update*) komponen perangkat lunak yang telah dibangun untuk meningkatkan kinerja dari perangkat lunak tersebut. Proses tersebut juga termasuk *maintenance* perangkat lunak yang telah didistribusikan.

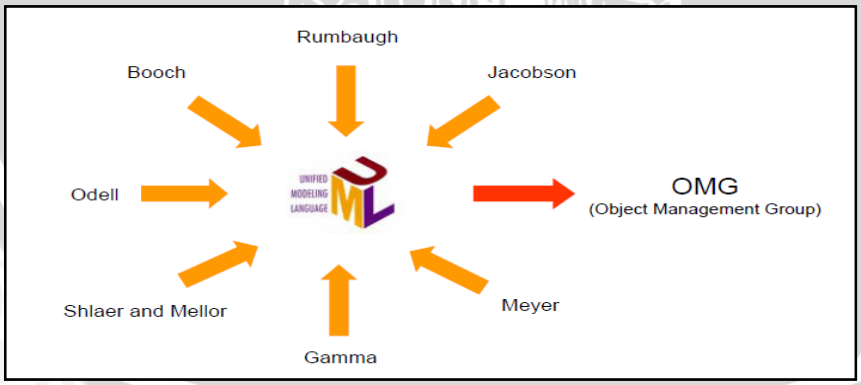
2.4. *Unified Modelling Language*

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa – bahasa berorientasi objek seperti

C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C [DHA-06:2].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi Booch, metodologi Coad, metodologi OOSE, metodologi OMT, metodologi Shlaer-Mellor, metodologi Wirfs-Brock dan sebagainya. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan grup / perusahaan lain yang menggunakan metodologi yang berlainan.



Gambar 2.3 Penyatuan metode UML

Sumber : [DHA-06:3]

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan

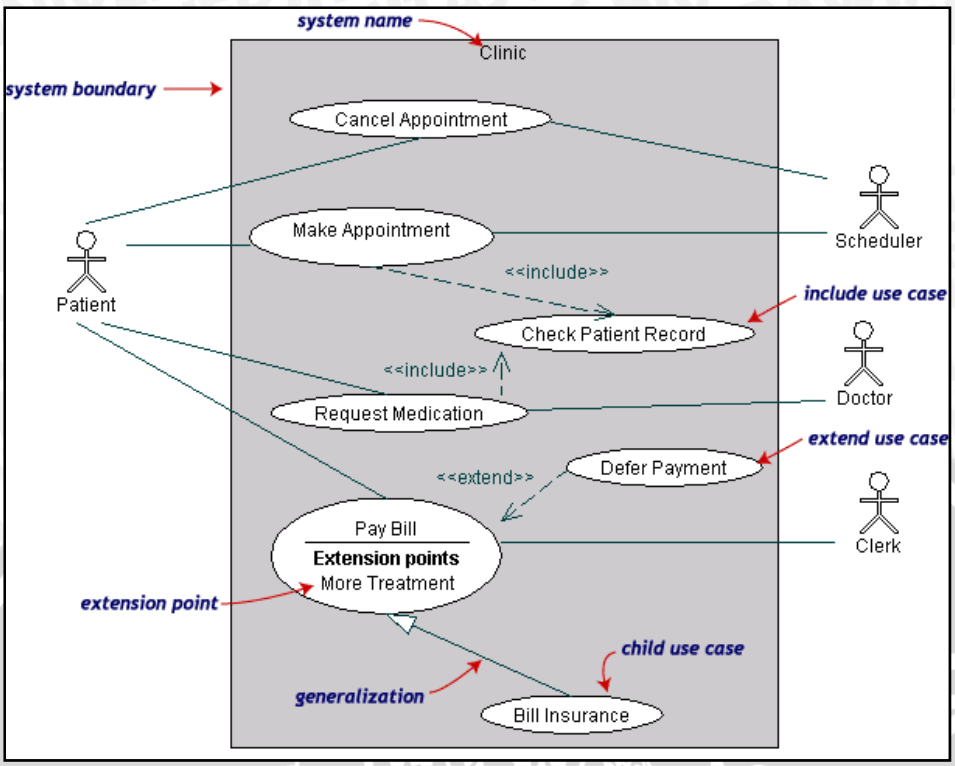
mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 di-*release draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh ***Object Management Group*** (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek [DHA-06:3].

Dalam UML pemodelan digambarkan dengan penggunaan diagram – diagram yang masing – masing memiliki fungsi dan makna sendiri – sendiri. Diagram – diagram ini dapat digunakan sesuai dengan kebutuhan saat kita akan merancang perangkat lunak. Beberapa diantaranya adalah diagram *use case*, diagram *class* dan diagram *activity*.

2.4.1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu [DHA-06:4].

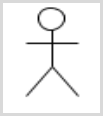
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

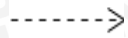
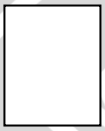
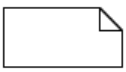


Gambar 2.4 Contoh *use case diagram*

Sumber : [DHA-06:5]

Tabel 2.1 Keterangan simbol - simbol *use case diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya sebagai elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / *ancestor*).

4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit.*
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.4.2. *Class Diagram*

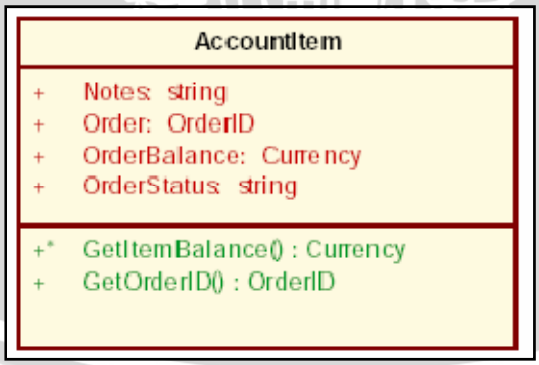
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah *object* dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan *object* beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [DHA-06:5].

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak - anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

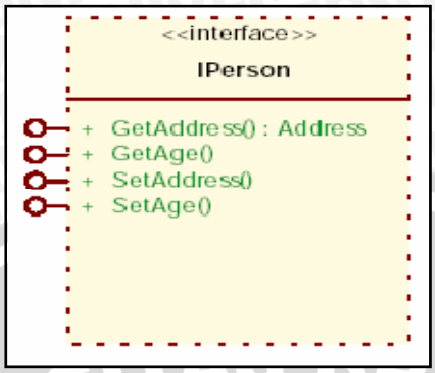


Gambar 2.5 Contoh sebuah *class*

Sumber : [DHA-06:5]

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung

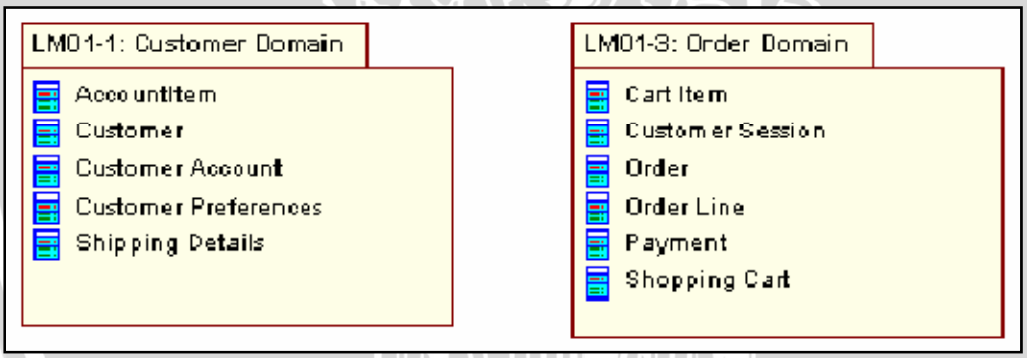
diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



Gambar 2.6 Contoh sebuah *interface*

Sumber : [DHA-06:6]

Sesuai dengan perkembangan *class model*, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.



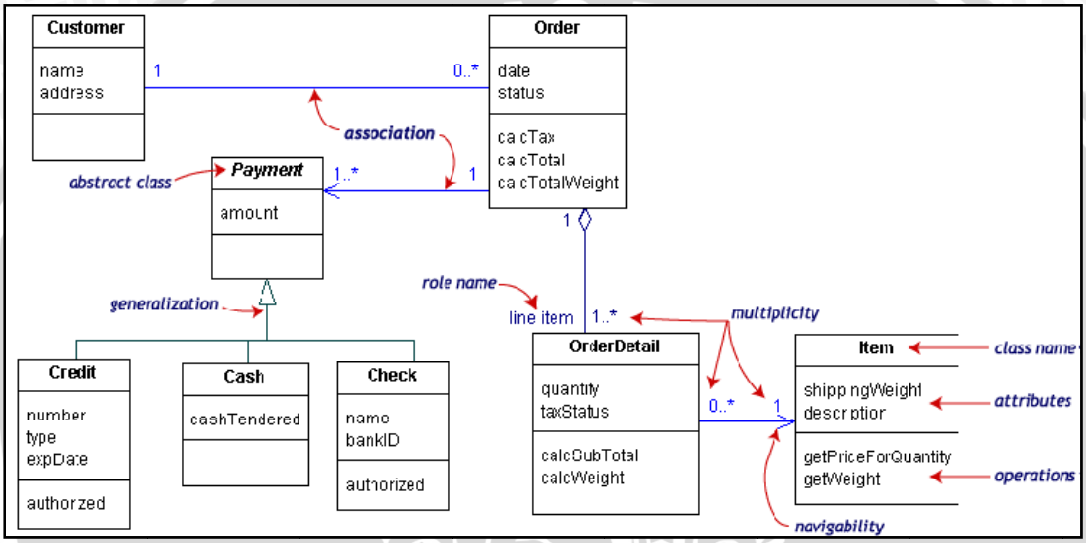
Gambar 2.7 Contoh *package*

Sumber : [DHA-06:6]

Dalam penggunaan metode *object-oriented* dapat dipastikan *class – class* memiliki hubungan – hubungan tertentu sesuai dengan fungsinya sendiri – sendiri. Hubungan antar *class* ada beberapa, yaitu :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

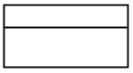


Gambar 2.8 Contoh *class diagram*

Sumber : [DHA-06:6]

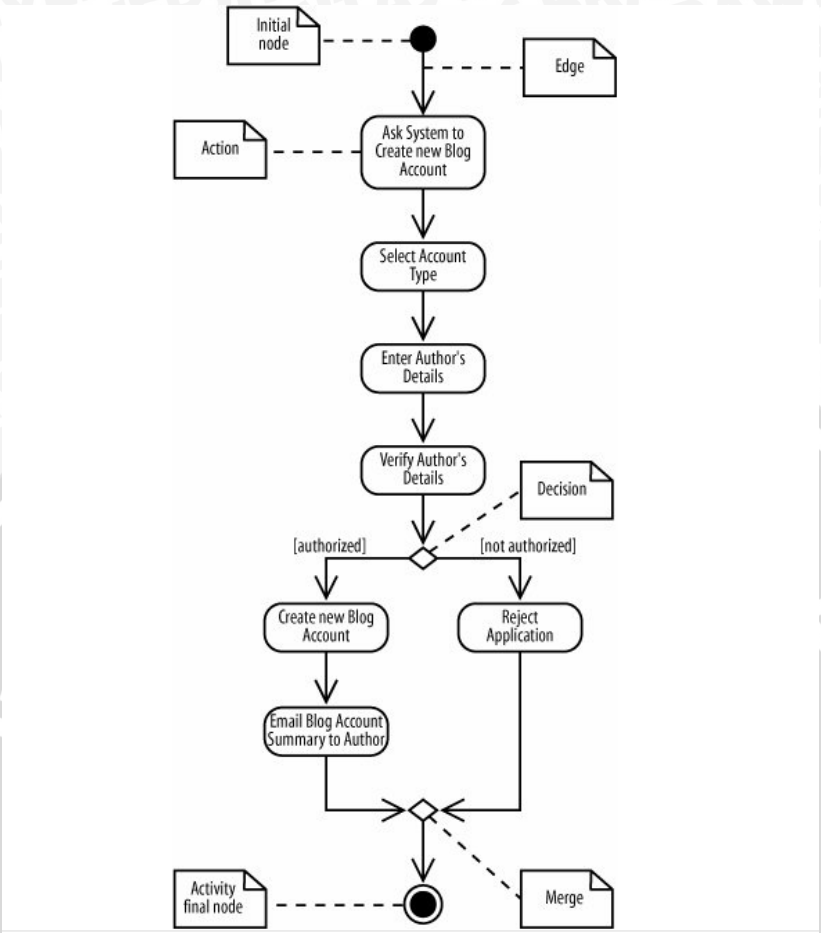
Tabel 2.2 Keterangan simbol - simbol *class diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya (objek induk / *ancestor*).
2		*N-ary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.

3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya atau elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
8	1..*, 0...*, 1...1	*Multiplicity*	Menunjukkan jumlah objek yang diperbolehkan menurut logika.

2.4.3. *Activity Diagram*

Tahap perancangan selanjutnya adalah pembuatan diagram aktivitas. Dengan diagram aktivitas memungkinkan untuk menentukan bagaimana sistem akan mencapai tujuannya [HAM-06]. Sebagai contoh, diagram aktivitas digunakan untuk memodelkan langkah – langkah dalam pembuatan akun sebuah blog. Dalam perancangan perangkat lunak GPS *Tracker* berbasis Android, diagram aktivitas digunakan untuk menggambarkan aliran aktivitas atau proses dari setiap use case dalam perangkat lunak GPS *Tracker* berbasis Android. Gambar 2.8 menunjukan contoh dari diagram aktivitas proses pembuatan akun blog.



Gambar 2.9 Contoh *activity diagram*

Sumber : [HAM-06]

2.5. Pengujian Perangkat Lunak

Pengujian (*testing*) arsitektur dari perangkat lunak berorientasi objek menghasilkan sekumpulan *layered subsystems* yang mengenkapsulasi kelas-kelas yang berkolaborasi. Setiap elemen sistem (subsistem dan *class*) melakukan fungsi yang membantu untuk mencapai kebutuhan sistem. Hal ini sangat penting untuk menguji sebuah *object-oriented system* pada berbagai macam level yang berbeda dalam sebuah usaha untuk menemukan kesalahan-kesalahan yang mungkin terjadi dari kolaborasi kelas-kelas dan komunikasi subsistem melewati *architetural layer* [PRE-01:631].

2.5.1. Teknik Pengujian

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode-metode ini menyediakan *developer* pendekatan sistematis untuk pengujian. Terlebih lagi metode-metode ini menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan-kesalahan dalam perangkat lunak [PRE-01:443]. Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing*.

2.5.1.1. *Black-Box Testing*

Black-box testing atau *behavioral testing* berfokus pada persyaratan fungsional perangkat lunak [PRE-01:459]. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk semua program. Pengujian *black-box* merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi.

Pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi. Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut :

- Bagaimana validitas fungsional diuji ?
- Kelas input apa yang akan membuat *test case* menjadi baik ?
- Apakah sistem sangat sensitif terhadap harga input tertentu ?
- Bagaimana batasan dari suatu data diisolasi ?

- Kecepatan dan volume data apa yang dapat ditolerir oleh sistem ?
- Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem ?

2.5.2. Strategi Pengujian

Strategi untuk pengujian perangkat lunak mengintegrasikan metode desain *test case* perangkat lunak ke dalam sederetan langkah yang direncanakan dengan baik, dan hasilnya adalah konstruksi perangkat lunak yang berhasil [PRE-01:477]. Beberapa strategi pengujian perangkat lunak telah diusulkan di dalam literatur. Strategi pengujian harus mengakomodasi pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang berlawanan dengan kebutuhan pelanggan. Proses pengujian dilakukan dengan pengujian validasi (*validation testing*) dan pengujian peforma [PRE-01:481].

2.5.2.1. Pengujian Validasi

Pada kulminasi pengujian terintegrasi, perangkat lunak secara lengkap dirakit sebagai suatu paket; kesalahan *interfacing* telah diungkap dan dikoreksi, dan seri akhir dari pengujian perangkat lunak, yaitu pengujian validasi dapat dimulai. Validasi dapat ditentukan dengan berbagai cara, tetapi definisi yang sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi dengan cara yang dapat diharapkan secara bertanggung jawab oleh pelanggan. Validasi perangkat lunak dicapai melalui sederetan pengujian *black-box* yang memperlihatkan konformitas dengan persyaratan. Rencana pengujian menguraikan kelas-kelas pengujian yang akan dilakukan, dan prosedur pengujian menentukan *test case* spesifik yang akan digunakan untuk mengungkap kesalahan dalam konformitas dengan persyaratan. Baik rencana dan prosedur didesain untuk memastikan apakah semua persyaratan fungsional dipenuhi; semua persyaratan kinerja dicapai; dokumentasi benar dan direkayasa oleh manusia: dan persyaratan lainnya dipenuhi (transportabilitas, kompatibilitas, pembetulan kesalahan, maintanabilitas) [PRE-01:495].

2.5.2.2. Pengujian Performa

Setelah semua langkah pengujian perangkat lunak secara terstruktur dilakukan, maka perlu dilakukan pengujian sistem di lingkungan dimana dia bekerja untuk mengetahui performa dari perangkat lunak tersebut. Pengujian

sistem dirancang untuk menguji kinerja *run-time* dari perangkat lunak dalam konteks sistem terintegrasi. Pengujian performa melibatkan *monitoring* pemanfaatan sumber daya dari perangkat lunak yang diuji seperti perangkat lunak pendukung dan perangkat keras. Pengujian performa dilakukan secara spesifik sesuai dengan tipe perangkat lunak yang diuji. Pengujian performa bertujuan untuk mengungkap situasi yang menyebabkan degradasi dan kemungkinan kegagalan sistem [PRE-01:498].

2.5.2.3. UAT *(User Acceptance Test)*

UAT *(User Acceptance Test)* atau Uji Penerimaan Pengguna adalah suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang dijadikan bukti bahwa *software* yang telah dikembangkan telah dapat diterima oleh pengguna, apabila hasil pengujian *(testing)* sudah bisa dianggap memenuhi kebutuhan dari pengguna. Proses dalam UAT adalah pemeriksaan dan pengujian terhadap hasil pekerjaan. Diperiksa apakah item-item yang ada dalam dokumen *requirement* sudah ada dalam *software* yang diuji atau tidak. Diuji apakah semua item yang telah ada telah dapat memenuhi kebutuhan penggunanya [MAX-13].

2.6. *Web Service*

Web service adalah suatu sistem perangkat lunak yang didisain untuk mendukung interaksi mesin ke mesin pada suatu jaringan. *Web service* mempunyai suatu *interface* yang diuraikan dalam suatu format *machine-processible* seperti WSDL. Sistem lain yang berinteraksi dengan *web service* dilakukan melalui interface atau antar muka menggunakan pesan seperti pada SOAP. Pada umumnya pesan ini melalui HTTP dan XML yang merupakan salah satu standard web. Perangkat lunak aplikasi yang ditulis dalam berbagai bahasa pemrograman dan berjalan pada berbagai platform dapat menggunakan *web service* untuk pertukaran data pada jaringan komputer seperti Internet. Sebagai contoh adalah antara Java dan Python atau Microsoft Windows dan aplikasi Linux [W3C-11].

Menurut Michael C. Daconta [DAC-05], *web service* adalah aplikasi perangkat lunak yang dapat ditemukan, diuraikan, dan diakses berdasarkan pada XML atau biasa disebut *Application Programming Interface* (API) dan protokol standard web pada intranet, extranet, dan Internet.

2.7. Bahasa Pemrograman HTML dan CSS

Hypertext Markup Language (HTML) adalah sekumpulan *text* atau *file* ASCII yang berisi instruksi atau perintah program untuk web *browser* untuk menampilkan tampilan grafis sebuah halaman website. *File* HTML dapat dibuat menggunakan aplikasi *text editor* pada semua sistem operasi, contohnya adalah *Notepad* di Windows atau simple *text* di Macintosh. HTML merupakan cross platform karena walau pembuatannya menggunakan sistem operasi tertentu, namun akan memiliki tampilan yang sama disemua sistem operasi.

Cascading Style Sheet (CSS) merupakan salah satu bahasa pemrograman web untuk mengendalikan beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. Sama halnya *styles* dalam aplikasi pengolahan kata seperti Microsoft Word yang dapat mengatur beberapa *style*, misalnya *heading*, *subbab*, *bodytext*, *footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa *file*. Pada umumnya CSS dipakai untuk memformat tampilan halaman web yang dibuat dengan bahasa HTML dan XHTML.

CSS dapat mengendalikan ukuran gambar, warna *body teks*, warna tabel, ukuran *border*, warna *border*, warna *hyperlink*, warna mouse over, spasi antar paragraf, spasi antar teks, margin kiri/kanan/atas/bawah, dan parameter lainnya. CSS adalah bahasa *style sheet* yang digunakan untuk mengatur tampilan dokument. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

2.8. Bahasa Pemrograman PHP

Rasmus Ledofrf adalah pencipta bahasa pemorgraman PHP *(Personal Home Page)* pada tahun 1995 yang pada masa itu masih di kenal dengan nama *Form Interpreted* (FI). Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, interpreter PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang PHP: *Hypertext Preprocessing*.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. Hingga saat ini PHP sudah merilis versi 5.6.2.

System kerja dari PHP diawali dengan permintaan yang beasal dari halaman website oleh browser. Berdasarkan URL atau alamat website dalam jaringan internet, *browser* akan menemukan sebuah alamat dari *webserver*, mengidentifikasi halaman yang dikehendaki, dan menyampaikan segala informasi yang dibutuhkan oleh webserver. Selanjutnya webserver akan mencarikan berkas yang diminta dan menampilkan isinya di *browser*. *Browser* yang mendapatkan isinya segera menerjemahkan kode HTML dan menampilkannya.

Pada prinsipnya sama dengan memanggil kode HTML, namun pada saat permintaan dikirim ke *webserver*, *webserver* akan memeriksa tipe file yang

diminta *user*. Jika tipe *file* yang diminta adalah PHP, maka akan memeriksa isi *script* dari halaman PHP tersebut. Apabila dalam *file* tersebut tidak mengandung *script* PHP, permintaan *user* akan langsung ditampilkan ke *browser*, namun jika dalam *file* tersebut mengandung *script* PHP, maka proses akan dilanjutkan ke modul PHP sebagai mesin yang menerjemahkan *script-script* PHP dan mengolah *script* tersebut, sehingga dapat dikonversikan ke kode-kode HTML dan ditampilkan ke browser *user* [PER-13:4].

2.9. Bahasa Pemrograman Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana program yang dibuat dengan bahasa lain seperti Pascal atau C++. Yang lebih menarik, Java juga mendukung sumber daya internet yang saat ini sangat populer, yaitu *world wide web* atau yang sering disebut sebagai web saja. Java juga mendukung aplikasi klien/server, baik dalam jaringan lokal (LAN) maupun jaringan berskala luas (WAN).

Java dikembangkan oleh Sun Microsystems pada Agustus 1991, dengan nama semula Oak. Konon Oak adalah pohon semacam jati yang terlihat dari jendela tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan bahwa Oak adalah singkatan dari *"Object Application Kernel"*, tetapi ada yang mengatakan hal itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java.

Dalam sejumlah literatur disebutkan bahwa Java merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C, C++, Object-C, SmallTalk dan Common LISP. Selain itu Java juga dilengkapi dengan unsur keamanan. Yang tak kalah penting adalah bahwa Java menambahkan paradigma pemrograman yang sederhana. Misalnya dalam bahasa C atau C++ yang menggunakan pointer, namun Java justru meninggalkannya karena sistem pointer dapat dikatakan sangat rumit, sehingga Java menjadi lebih mudah digunakan [KAD-09:2].

2.10. Basis Data

Basis data (bahasa Inggris: *database*), atau sering pula dieja **basis data**, adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi.

Istilah "basis data" berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis.

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data. Model yang umum digunakan sekarang adalah model relasional, yang menurut istilah *layman* mewakili semua informasi dalam bentuk tabel-tabel yang saling berhubungan dimana setiap tabel terdiri dari baris dan kolom (definisi yang sebenarnya menggunakan terminologi matematika). Dalam model ini, hubungan antar tabel diwakili dengan menggunakan nilai yang sama antar tabel. Model yang lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar tabel.

Istilah *basis data* mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai *sistem manajemen basis data* (*database management system/DBMS*). Jika konteksnya sudah jelas, banyak *administrator* dan *programer* menggunakan istilah basis data untuk kedua arti tersebut.

2.11. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL memiliki beberapa keistimewaan, antara lain :

1. **Portabilitas**. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. **Perangkat lunak sumber terbuka**. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. ***Multi-user***. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. ***'Performance tuning'***, MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. **Ragam tipe data**. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed*/*unsigned integer*, *float*, *double*, *char*, *text*, *date*, *timestamp*, dan lain-lain.

6. **Perintah dan Fungsi**. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *select* dan *where* dalam perintah (*query*).
7. **Keamanan**. MySQL memiliki beberapa lapisan keamanan seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. **Skalabilitas dan Pembatasan**. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. **Konektivitas**. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix *socket* (UNIX), atau *Named Pipes* (NT).
10. **Lokalisasi**. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. **Antarmuka**. MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. **Klien dan Peralatan**. MySQL dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk *online*.
13. **Struktur tabel**. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

Terdapat beberapa API (*Application Programming Interface*) tersedia yang memungkinkan aplikasi-aplikasi komputer yang ditulis dalam berbagai bahasa pemrograman untuk dapat mengakses basis data MySQL antara lain: bahasa pemrograman C, C++, C#, bahasa pemrograman Eiffel, bahasa

pemrograman Smalltalk, bahasa pemrograman Java, bahasa pemrograman Lisp, Perl, PHP, bahasa pemrograman Python, Ruby, REALbasic dan Tcl. Sebuah antarmuka ODBC memanggil MyODBC yang memungkinkan setiap bahasa pemrograman yang mendukung ODBC untuk berkomunikasi dengan basis data MySQL. Kebanyakan kode sumber MySQL dalam ANSI C.

MySQL sangat populer dalam aplikasi *web* seperti MediaWiki (perangkat lunak yang dipakai Wikipedia dan proyek-proyek sejenis) dan PHP-Nuke dan berfungsi sebagai komponen basis data dalam LAMP. Popularitas sebagai aplikasi *web* dikarenakan kedekatannya dengan popularitas PHP, sehingga seringkali disebut sebagai *Dynamic Duo*.

2.12. SQLite

SQLite adalah paket aplikasi yang menyediakan sistem database relational (RDBMS), sebagaimana vendor RDBMS lainnya seperti Oracle, MySQL, PostgreSQL dll. Kata *Lite* bukan berarti aplikasi RDBMS ini memiliki kemampuan yang sedikit / minim, tetapi mengacu pada keringanan/kemudahan dalam setup (instalasi), administrasi dan penggunaan. Dan fitur SQLite adalah sebagai berikut:

1. ***Serverless***, SQLite tidak memerlukan proses pada *server* atau sistem untuk menjalankannya, melainkan hanya sebuah *file* yang diakses oleh *library* SQLite.
2. ***Zero Configuration***, Tidak ada *server* berarti tidak perlu setup, membuat sebuah *database* instan adalah semudah anda membuat *file* biasa.
3. ***Cross Platform***, semua instan *database* berada dalam sebuah *file* yang *cross-platform*, tidak memerlukan administrasi.
4. ***Self-Contained***, sebuah *library* mengandung keseluruhan dari sistem database, yang langsung terintegrasi pada sebuah aplikasi program.
5. ***Small Runtime Footprint***, untuk membangun database SQLite hanya membutuhkan kurang dari satu *megabyte library* (kode

program) dan hanya membutuhkan beberapa *megabyte memory*, bahkan dengan beberapa adjustment baik ukuran *library* maupun memory dapat diperkecil.

6. ***Transactional***, SQLite *transaction* memperbolehkan aksi penyimpanan melalui beberapa proses *thread*.
7. ***Full Featured***, SQLite mensupport hampir sebagai besar standar SQL92 (SQL2).
8. ***Highly Reliable***, Tim pengembang SQLite mengembangkan melalui kode program yang sangat serius serta telah melewati proses *testing*.

Secara keseluruhan, SQLite menyediakan *database* relasional yang fungsional dan fleksibel dengan meminimalkan sumber daya dan kerumitan untuk pengembang dan pengguna [JAY-10:23].

Berikut gambaran perbedaan RDBMS dengan SQLite:

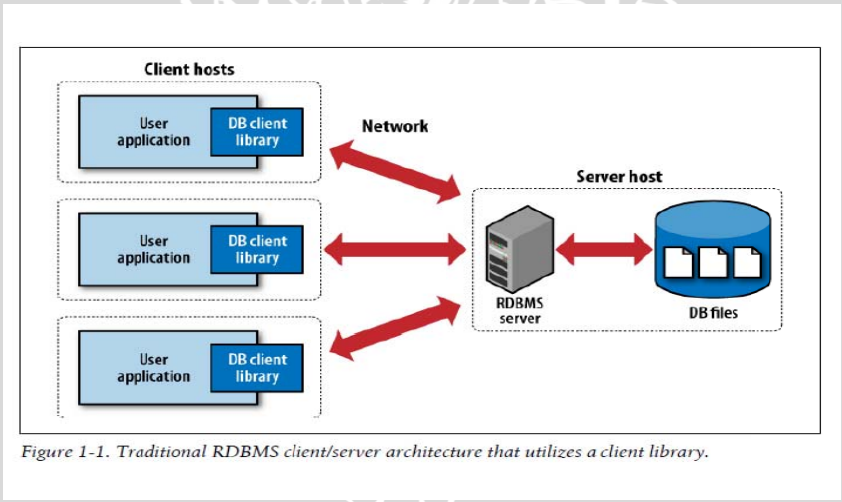


Figure 1-1. Traditional RDBMS client/server architecture that utilizes a client library.

Gambar 2.10 Contoh RDBMS

Sumber: [JAY-10:25]

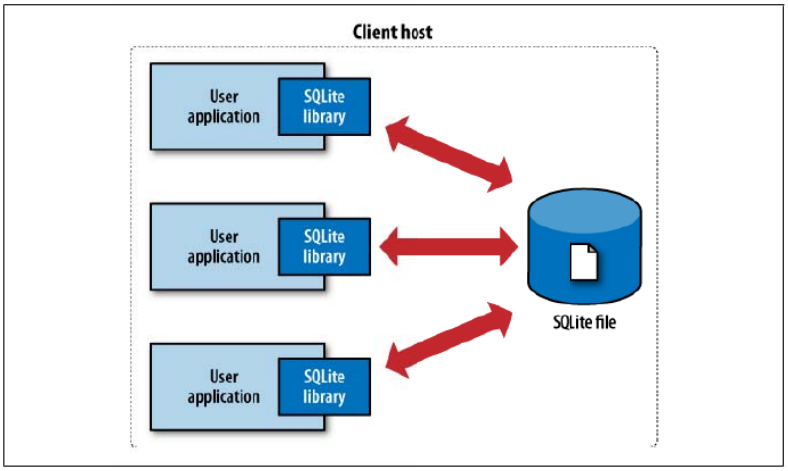


Figure 1-2. The SQLite server-less architecture.

GAMBAR 2.11 Contoh SQLite

Sumber: [JAY-10:25]

2.13. Google Maps Api

Google Maps adalah layanan aplikasi peta online yang disediakan oleh Google secara gratis. Layanan peta Google Maps secara resmi dapat diakses melalui situs http://maps.google.com. Pada situs tersebut dapat dilihat informasi geografis pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan. Layanan ini dibuat sangat interaktif, karena di dalamnya peta dapat digeser sesuai keinginan pengguna, mengubah level zoom, serta mengubah tampilan jenis peta.

Google Maps mempunyai banyak fasilitas yang dapat dipergunakan misalnya pencarian lokasi dengan memasukkan kata kunci, kata kunci yang dimaksud seperti nama tempat, kota, atau jalan, fasilitas lainnya yaitu perhitungan rute perjalanan dari satu tempat ke tempat lainnya.

API *(Application Programming Interface)* merupakan suatu dokumentasi yang terdiri dari interface, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya API ini, maka memudahkan programmer untuk "membongkar" suatu software untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang

memungkinkan programmer menggunakan sistem function. Proses ini dikelola melalui operating system. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya dapat saling berhubungan dan berinteraksi. Bahasa pemrograman yang digunakan oleh Google Maps yang terdiri dari HTML, Javascript dan AJAX serta XML, memungkinkan untuk menampilkan peta Google Maps di website lain.

Google juga menyediakan layanan Google Maps API yang memungkinkan para pengembang untuk mengintegrasikan Google Maps ke dalam website masing-masing dengan menambahkan data point sendiri. Dengan menggunakan Google Maps API, Google Maps dapat ditampilkan pada web site eksternal. Agar aplikasi Google Maps dapat muncul di website tertentu, diperlukan adanya API key. API key merupakan kode unik yang digenerasikan oleh google untuk suatu website tertentu, agar server Google Maps dapat mengenali [AMR-11].

2.14. Sistem Operasi Android

2.14.1 Pengenalan Android

Android adalah software untuk perangkat mobile yang mencakup sistem operasi, middleware dan aplikasi kunci. SDK (*Software Development Kit*) Android menyediakan alat dan API (*Application Programming Interface*) yang diperlukan untuk mengembangkan aplikasi pada platform Android dengan menggunakan bahasa pemrograman Java [HER-11]. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam *mobile devices*.

2.14.2 Fitur Android

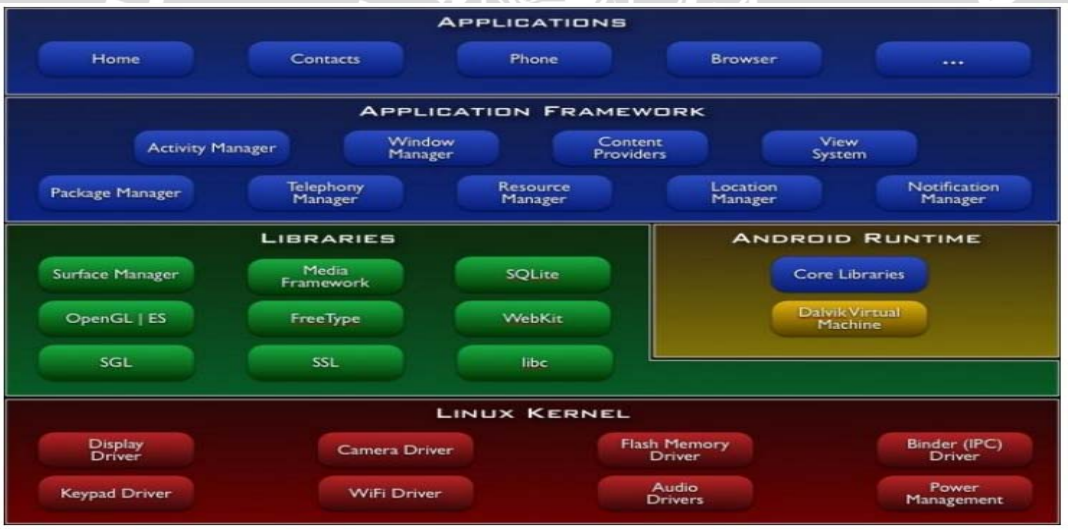
Android juga memiliki beberapa fitur sebagai berikut [HER-11]. :

1. **Application *framework*** memungkinkan penggunaan kembali dan penggantian komponen.
2. ***Dalvik virtual machine*** optimalisasi untuk perangkat bergerak.
3. ***Integrated browser*** berdasarkan open source WebKit mesin.

4. ***Optimized graphics*** didukung *librarygrafis* 2D ; grafis 3D berbasis OpenGL ES 1.0.
5. ***SQLite*** untuk penyimpanan data terstruktur.
6. ***Media support*** untuk audio, video, dan gambar masih dalam format (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
7. ***GSM Telephony.***
8. ***Bluetooth, EDGE, 3G, and WiFi.***
9. ***Camera,* GPS, *compass, and accelerometer***.
10. ***Rich development environment*** termasuk *device emulator*, alat untuk *debugging*, *memory* dan *performance profiling*, dan *plugin* untuk Eclipse IDE.

2.14.3 Arsitektur Android

Diagram Arsitektur Android ditunjukkan pada gambar 2.13.



Gambar 2.12 Arsitektur Android

Sumber : [HER-11]

Sesuai ilustrasi yang ditunjukkan pada gambar 2.13. Arsitektur Android dapat dibagi menjadi lima bagian besar, yaitu:

1. ***Applications***

Android memuat seperangkat aplikasi inti mencakup email *Klien*, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Application Framework*

Android menawarkan kemampuan untuk membangun aplikasi yang sangat kaya dan inovatif. Pengembang bebas untuk mengakses perangkat keras, informasi akses lokasi, menjalankan *background services*, mengatur alarm, tambahkan pemberitahuan ke status bar, dan banyak lagi. Pengembang memiliki akses penuh ke *framework* API yang sama yang digunakan oleh aplikasi inti. Arsitektur aplikasi dirancang untuk menyederhanakan penggunaan kembali komponen, aplikasi apapun dapat mempublikasikan kemampuannya dan aplikasi lain kemudian dapat menggunakan kemampuan mereka (diatur oleh batasan keamanan yang diberlakukan oleh *framework*). Mekanisme ini mengizinkan *user* mengganti komponen. Berikut adalah contoh servis dan *system* yang mendasari seluruh aplikasi Android:

- *View* : tampilan yang dapat digunakan untuk membangun aplikasi, termasuk didalamnya *list, grid*, kotak *teks*, tombol, dan sebagaianya.
- *Content Providers* : memungkinkan aplikasi untuk mengakses data dari aplikasi lain (misal kontak), atau untuk membagi data yang dimiliki.
- *Resource Manager* : menyediakan akses ke *non-code resources* seperti grafik dan layout data.
- *Notification Manager* : memungkinkan aplikasi untuk menampilkan peringatan pada status bar.
- *Activity Manager* : mengatur *lifecycle* aplikasi.

3. *Libraries*

Libraries adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi Android mengakses *libraries* untuk menjalankan aplikasinya. Android mencakup set *library* C/C++ yang digunakan oleh berbagai komponen dari sistem Android. Beberapa library tercantum seperti dibawah ini:

- Sistem C *library* : *library* standard C (libc)
- Media *Libraries* : *library* untuk mendukung memutar dan merekam berbagai format audio dan video, juga file gambar.
- *Surface Manager* : mengatur akses ke tampilan subsistem dan memadukan grafik 2D dan 3D pada beberapa aplikasi.

- LibWeb Core : *web browser engine* yang mendukung Android *browser*.
- SGL : mendasari *engine* grafik 2D.
- 3D *libraries* : implementasi OpenGL ES 1.0 APIs berfungsi mengoptimalkan fungsionalitas 3D.
- FreeType : *rendering bitmap* dan *font*.
- SQLite : *engine* basis data yang ampuh dan ringan yang tersedia untuk semua aplikasi.

4. Android *Runtime*

Android mencakup seperangkat *library* inti yang menyediakan sebagian besar fungsi yang tersedia di library inti dari bahasa pemrograman Java. Setiap aplikasi Android berjalan dalam prosesnya sendiri, dengan Dalvik *Virtual Machine*-nya (VM) sendiri. Dalvik VM mengeksekusi *file* dalam format Dalvix excetubale (.dex) yang dioptimalkan untuk meminimalisir jejak memori.

5. *Linux Kernel*

Android bergantung pada Linux versi 2.6 untuk layanan sistem inti seperti keamanan, manajemen memori, manajemen proses, jaringan *stack*, dan *driver model*.

2.14.4 Fundamental Aplikasi

Aplikasi Android ditulis dalam bahasa pemrograman java. Kode java dikompilasi bersama dengan data *file resource* yang dibutuhkan oleh aplikasi, dimana proses di-*package* oleh *tools* yang dinamakan “apt tools” ke dalam paket Android sehingga menghasilkan *file* dengan ekstensi apk. *File* apk itulah yang kita sebut dengan aplikasi, dan nantinya dapat di-*install* diperangkat mobile. Ada empat jenis komponen pada aplikasi Android, yaitu [SAF-11] :

1. *Activities*

Suatu *activity* akan menyajikan *user interface* (UI) pada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi atau yang bertindak sebagai user interface (UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu

activity ke *activity* lain kita dapat melakukannya dengan satu *event*, misalnya klik tombol, memilih opsi atau menggunakan *triggers* tertentu. Secara hirarki sebuah *windows activity* dinyatakan dengan method *Activity.setContentView*(). *ContentView* adalah objek yang berada pada root hirarki.

2. ***Services***

Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*, sebagai contoh dalam memainkan musik, *service* mungkin memainkan musik atau mengambil data jaringan, tetapi *service* harus berada dalam kelas induknya. Misalnya, media player sedang memutar lagu dari *list* yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu, atau sms dan media player sedang berjalan.

3. ***Broadcast Receiver***

Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh kamera, atau mengubah referensi bahasa yang digunakan. *Broadcast receiver* tidak memiliki *user interface* (UI), tetapi memiliki *activity* untuk merespon informasi yang mereka terima.

4. ***Content Provider***

Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam *file* sistem seperti SQLite. *Content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketikan kita menggunakan aplikasi yang membutuhkan peta (Map), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi *content provider*.

2.15. GPS

GPS *(Global Positioning System)* adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan *(Departemen of Defense)* Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS reciever yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik. Sejak tahun 1980, layanan GPS yang dulunya hanya untuk leperluan militer mulai terbuka untuk publik. Uniknya, walau satelit-satelit tersebut berharga ratusan juta dolar, namun setiap orang dapat menggunakannya dengan gratis. Satelit-satelit ini mengorbit pada ketinggian sekitar 12.000 mil dari permukaan bumi. Posisi ini sangat ideal karena satelit dapat menjangkau area *coverage* yang lebih luas. Satelit-satelit ini akan selalu berada posisi yang bisa menjangkau semua area di atas permukaan bumi sehingga dapat meminimalkan terjadinya blank spot (area yang tidak terjangkau oleh satelit).

Setiap satelit mampu mengelilingi bumi hanya dalam waktu 12 jam. Sangat cepat, sehingga mereka selalu bisa menjangkaudimana pun posisi Anda di atas permukaan bumi. GPS *reciever* sendiri berisi beberapa integrated circuit (IC) sehingga murah dan teknologinya mudah untuk di gunakan oleh semua orang. GPS dapat digunakan utnuk berbagai kepentingan, misalnya mobil, kapal, pesawat terbang, pertanian dan diintegrasikan dengan komputer maupun laptop.

2.15.1 Sistem Satelit GPS

Untuk menginformasikan posisi user, 24 satelit GPS yang ada di orbit sekitar 12,000 mil di atas kita. Bergerak konstan bergerak mengelilingi bumi 12 jam dengan kecepatan 7,000 mil per jam. Satelit GPS berkekuatan energi sinar matahari, mempunyai baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari. Roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya.

Gambar 2.13 Simulasi Posisi Satelit GPS

Sumber: [OFF-13]

Satelit GPS adalah milik Departemen Pertahanan *(Department of Defense)* Amerika, adapun hal-hal lainnya:

1. Nama satelit adalah NAVSTAR.
2. GPS satelit pertama kali adalah tahun 1978.
3. Mulai ada 24 satelit dari tahun 1994.
4. Satelit di ganti tiap 10 tahun sekali.
5. GPS satelit beratnya kira-kira 2,000 pounds.
6. Kekuatan transmiter hanya 50 watts atau kurang.

Satelit-satelit GPS harus selalu berada pada posisi orbit yang tepat untuk menjaga akurasi data yang dikirim ke GPS *reciever*, sehingga harus selalu dipelihara agar posisinya tepat. Stasiun-stasiun pengendali di bumi ada di Hawaii, Ascension Islan, Diego Garcia, Kwajalein dan Colorado Spring. Stasiun bumi tersebut selalu memonitor posisi orbit jam jam satelit dan di pastikan selalu tepat [OFF-13].

2.15.2 Signal Satelit GPS

2.15.2.1 *Carriers*

Satelite GPS mengirim sinyal dalam dua frekuensi. L1 dengan 1575.42 Mhz dengan membawa dua status pesan dan pseudo-random codeuntuk keperluan perhitungan wakt. L2 membawa 1227.60 MHz dengan menggunakaan presesi yang lebih akurat karena untuk keperluan militer. Daya sinyal radio yang dipancarkan hanya berkisar antara 20-50 Watts. Ini tergolong sangat rendah

mengingat jarak antara GPS dan satelit sampai 12.000 mil. Sinyal dipancarkan secara LOS *(line of sight)*, dapat melewati awan, kaca tapi tidak dapat benda padat seperti gedung, gunung.

2.15.2.2 *Pseudo-Random Codes*

GPS yang digunakan untuk publik akan memantau frekuensi L1 pada UHF *(Ultra High Frequency)* 1575,42 MHz. Sinyal L1 yang dikirimkan akan memiliki pola-pola kode digital tertentu yang disebut sebagai pseudorandom. Sinyal yang dikirimkan terdiri dari dua bagian yaitu kode Protected (P) dan Coarse/Acquisition (C/A). Kode yang dikirim juga unik antar satelit, sehingga memungkinkan setiap receiver untuk membedakan sinyal yang dikirim oleh satu satelit dengan satelit lainnya. Beberapa kode Protected (P) juga ada yang diacak, agar tidak dapat diterima oleh GPS biasa. Sinyal yang diacak ini dikenal dengan istilah Anti Spoofing, yang biasanya digunakan oleh GPS khusus untuk keperluan tertentu seperti militer.

2.15.2.3 *Navigation Message*

Ada sinyal frekuensi berkekuatan lemah yang di tambahkan pada kode L1 yang memberikan informasi tentang orbit satelit, *clock correction*-nya dan status sistem lainnya.

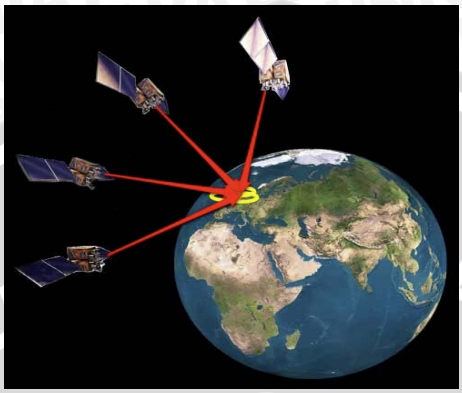
2.15.3 Cara Kerja GPS

Setiap daerah di atas permukaan bumi ini minimal terjangkau oleh 3-4 satelit. Pada prakteknya, setiap GPS terbaru bisa menerima sampai dengan 12 chanel satelit sekaligus. Kondisi langit yang cerah dan bebas dari halangan membuat GPS dapat dengan mudah menangkap sinyal yang dikirimkan oleh satelit. Semakin banyak satelit yang diterima oleh GPS, maka akurasi yang diberikan juga akan semakin tinggi.

Cara kerja GPS secara logika ada 5 langkah:

1. Memakai perhitungan “triangulation”dari satelit.
2. Untuk perhitungan “triangulation”, GPS mengukur jarak menggunakan travel time sinyal radio.
3. Untuk mengukur travel time, GPS memerlukan memerlukan akurasi waktu yang tinggi.

4. Untuk perhitungan jarak, kita harus tahu dengan pasti posisi satelit dan ketingian pada orbitnya.
5. Terakhir harus menggoreksi delay sinyal waktu perjalanan di atmosfer sampai diterima receiver.



GAMBAR 2.14 Satelit GPS Mengirim Signal

Sumber: [OFF-13]

Satelit GPS berputar mengelilingi bumi selama 12 jam di dalam orbit yang akurat dia dan mengirimkan sinyal informasi ke bumi. GPS reciever mengambl informasi itu dan dengan menggunakan perhitungan “triangulation” menghitung lokasi user dengan tepat. GPS reciever membandingkan waktu sinyal di kiirim dengan waktu sinyal tersebut di terima. Dari informasi itu didapat diketahui berapa jarak satelit. Dengan perhitungan jarak-jarak GPS *reciever* dapat melakukan perhitungan dan menentukan posisi user dan menampilkan dalam peta elektronik.

Sebuah GPS reciever harus mengunci sinyal minimal tiga satelit untuk memenghitung posisi 2D (latitude dan longitude) dan track pergerakan. Jika GPS reciever dapat menerima empat atau lebih satelit, makadapat menghitung posisi 3D (latitude, longitude, dan altitude). Jika sudah dapat menentukan posisi user, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur, tujuan perjalanan, jarak tujuan, matahari terbit dan matahari terbenam dan masih banyak lagi.

Satelit GPS dalam mengirim informasi waktu sangat presesi karena Satekit tersebut memakai jam atom. Jam atom yang ada pada satelit jalam dengan partikel atom yang di isolasi, sehingga dapat menghasilkan jam yang akurat

dibandingkan dengan jam biasa. Perhitungan waktu yang akurat sangat menentukan akurasi perhitungan untuk menentukan informasi lokasi kita. Selain itu semakin banyak sinyal satelit yang dapat diterima maka akan semakin presesi data yang diterima karena ketiga satelit mengirim pseudo-random code dan waktu yang sama.

Ketinggian itu menimbulkan keuntungan dalam mendukung proses kerja GPS, bagi kita karena semakin tinggi maka semakin bersih atmosfer, sehingga gangguan semakin sedikit dan orbit yang cocok dan perhitungan matematika yang cocok. Satelit harus teptap pada posisi yang tepat sehingga stasiun di bumi harus terus memonitor setiap pergerakan satelit, dengan bantuan radar yang presesi salalu di cek tentang altitude, posision, dan kecepatannya [OFF-13].

2.15.4 Sinyal Menentukan Lokasi

Sinyal yang dikirimkan oleh satelit ke GPS akan digunakan untuk menghitung waktu perjalanan *(travel time)*. Waktu perjalanan ini sering juga disebut sebagai TOA *(Time of Arrival)*. Sesuai dengan prinsip fisika, bahwa untuk mengukur jarak dapat diperoleh dari waktu dikalikan dengan cepat rambat sinyal. Maka, jarak antara satelit dengan GPS juga dapat diperoleh dari prinsip fisika tersebut. Setiap sinyal yang dikirimkan oleh satelit akan juga berisi informasi yang sangat detail, seperti orbit satelit, waktu, dan hambatan di atmosfir. Satelit menggunakan jam atom yang merupakan satuan waktu paling presisi.

Untuk dapat menentukan posisi dari sebuah GPS secara dua dimensi (jarak), dibutuhkan minimal tiga buah satelit. Empat buah satelit akan dibutuhkan agar didapatkan lokasi ketinggian (secara tiga dimensi). Setiap satelit akan memancarkan sinyal yang akan diterima oleh GPS receiver. Sinyal ini akan dibutuhkan untuk menghitung jarak dari masingmasing satelit ke GPS. Dari jarak tersebut, akan diperoleh jari-jari lingkaran jangkauan setiap satelit. Lewat perhitungan matematika yang cukup rumit, interseksi (perpotongan) setiap lingkaran jangkauan satelit tadi akan dapat digunakan untuk menentukan lokasi dari GPS di permukaan bumi [OFF-13].

2.15.5 Model dan Interkoneksi GPS

Sebuah GPS juga memiliki firmware yang bisa di-upgrade. Upgrade firmware ini biasanya disediakan pada site produsen GPS tersebut. Upgrade firmware biasanya menggunakan kabel yang dibundel atau-pun tersedia sebagai asesoris. Kabel ini juga ternyata bisa digunakan untuk menghubungkan GPS ke komputer (baik itu notebook, PC, maupun PDA dengan sedikit bantuan konverter). Software GPS yang tersedia untuk berbagai platform tersebut juga cukup banyak. Dengan software tersebut, Anda dapat dengan mudah mendownload informasi dari GPS. Memori sebuah GPS memang relatif terbatas, sehingga kemampuan ekstra untuk menyimpan informasi yang ditempuh ke PC/PDA (yang biasanya memiliki memori lebih besar) tentu akan sangat menyenangkan. Untuk media komunikasi GPS dengan hardware lain selain kabel, model GPS sekarang juga ada yang dilengkapi dengan Bluetooth, Infrared.

Berdasarkan fisik, model GPS dibagi menjadi beberapa tipe antara lain model portable/handheld (ukurannya menyerupai ponsel), ada yang lebih besar (biasanya dimount di mobil/kapal), ada pula yang menggunakan interface khusus untuk dikoneksikan ke notebook maupun PDA (Palm,Pocket PC maupun Nokia Com-municator). GPS untuk keperluan out- door biasanya juga dilengkapi dengan perlindungan anti air dan tahan ben-turan. Beberapa GPS keluaran terakhir bahkan sudah menyediakan layar warna dan kemampuan komunikasi radio jarak pendek (FRS/Family Radio Service). Tentu saja, semakin banyak feature yang ditawarkan pada sebuah GPS maka semakin tinggi pula harganya.

Jika suatu saat Anda ingin pergi ke lokasi yang pernah Anda kunjungi dengan menggunakan GPS. Maka, Anda tinggal meng-upload data yang pernah Anda simpan di komputer kembali ke GPS. Selanjutnya, Anda akan mendapatkan rekaman perjalanan Anda terdahulu. Lokasi dan track yang pernah Anda kunjungi akan dapat Anda temui kembali dengan cepat, dan tentu saja meminimalkan resiko tersesat [OFF-13].

2.15.6 *Tracking System*

Prinsip Tracking System adalah dengan menggunakan teknologi GPS dan GSM. Kita hidup dikelilingi jaringan satelit yang mengorbit bumi dan jutaan Administrator GPS di daratan. GPS menentukan sebuah titik lokasi dengan sinyal dari 3 satelit yang berbeda. GPS bekerja dengan mencari jarak dari satelit-satelit dan menghitung satu-satunya tempat dimana ketiga sinyal satelit tertuju secara bersamaan.

Dalam perangkat pelacak juga ditanam jaringan GSM sebagai media untuk berkomunikasi secara langsung (real-time) dengan alat lain didaratan bumi seperti komputer, ponsel atau alat pantau yang lain. GSM atau Global System for Mobile berguna untuk memberi informasi mengenai posisi yang ditemukan dari satelit untuk mereka yang ingin tahu di mana perangkat GPS tersebut berada.

GSM merupakan infrastruktur jaringan yang paling populer di dunia. Dengan jumlah pengguna saat ini 3 milyar orang di seluruh dunia dan tingkat penyebaran yang merata di wilayah bumi, membuat GSM penjadi pilihan terdepan dalam menyediakan sinyal digital untuk semua bentuk komunikasi.

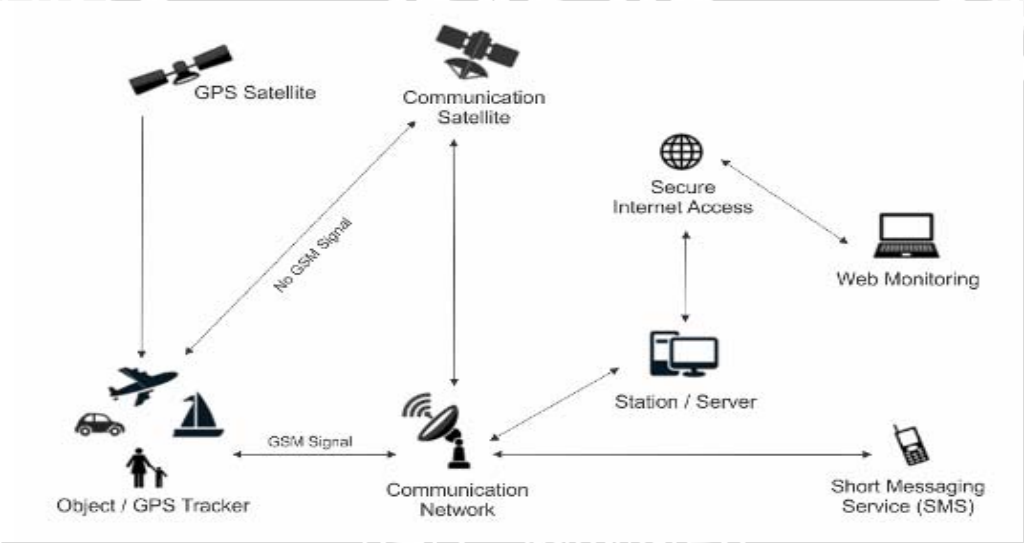
Jadi, secara sederhana, ada dua bagian yang terintegrasi untuk kebutuhan sistem pelacakan ini. Sistem GPS adalah untuk mengumpulkan informasi lokasi yang akurat dari satelit, lalu bagian kedua adalah GSM untuk mengirim informasi lokasi tersebut sehingga dapat dilacak dari tempat lain. Informasi lokasi berupa koordinat yang lalu diterjemahkan ke dalam tampilan posisi pada peta digital.

Dengan Real-Time GPS Tracking System, anda dapat duduk di PC atau komputer laptop, login ke sistem yang telah diinstall atau sistem web-based untuk memantau gerakan, arah dan kecepatan dari setiap kendaraan yang memiliki unit GPS Tracking System. Selain itu, dengan update hardware dan program yang terus dikembangkan saat ini, anda juga dapat memanfaatkan beragam fitur seperti mematikan mesin, alarm bahaya, alarm kecepatan, sensor bahan bakar, sensor penumpang, komunikasi audio, serta data historis, laporan dan analisis.

Selain kombinasi GPS dan GSM yang populer, sebenarnya masih ada teknologi lain yang digunakan dalam Tracking System. Sebut saja Radio Frequency Identification (RFID), Geographic Information Systems (GIS) dan

Wireless Local Area Network (WLAN). Masing-masing teknologi memiliki karakter dan cara kerja berbeda sesuai spesifikasi dan tujuan yang dibutuhkan.

Ada banyak aplikasi berbeda dan dikembangkan untuk memanfaatkan teknologi pelacakan dan membawa sumbangsih besar bagi penelitian IPTEK, pertanahan, pertanian, pertahanan, dan keamanan, hingga usaha komersial [AMA-09].



Gambar 2.15 *Tracking System*
Sumber: [AMA-09]

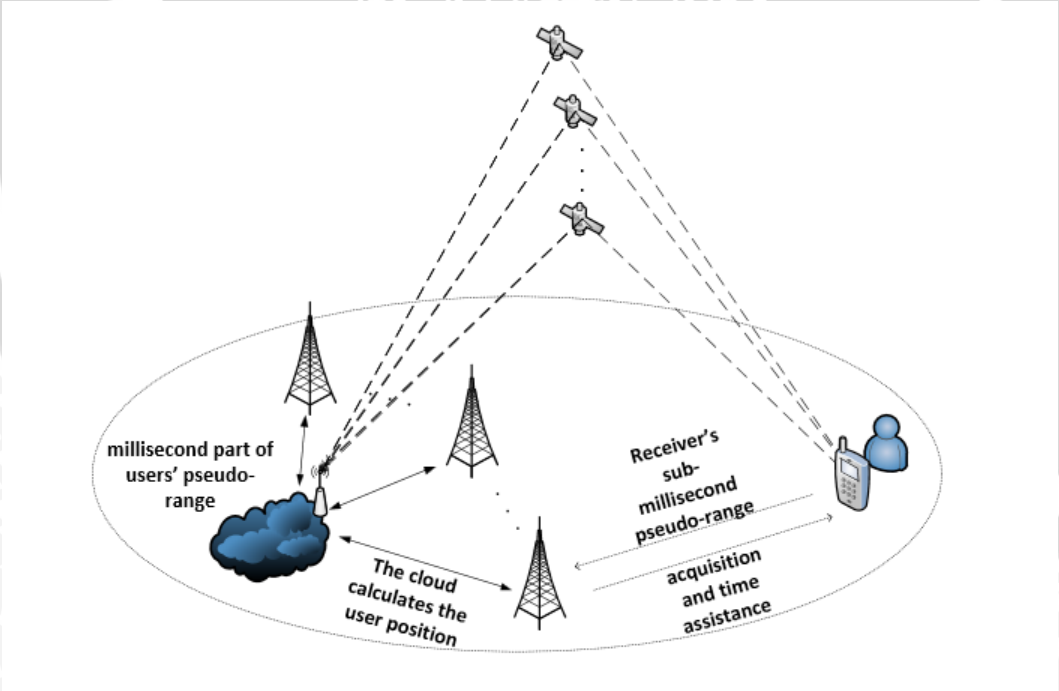
2.16. *Asissted-Global Positioning System* (A-GPS)

Frank Van Diggelen [DIG-09:1] mengatakan bahwa A-GPS merupakan penyempurnaan dari GPS sebagai satelit penentu posisi di belahan bumi. Satelit GPS yang dimiliki bumi mempunyai konstelasi 24 satelit dalam enam orbit yang mendekati lingkaran, setiap orbit ditempati oleh empat buah satelit dengan interval antara yang tidak sama. Orbit satelit GPS berinklinasi 550° terhadap bidang equator dengan ketinggian rata-rata dari permukaan bumi sekitar 20.200 km [ELR-06:1].

Metode *Advanced Positioning* yang terdapat pada A-GPS merupakan metode penentuan posisi yang paling tinggi akurasinya dibandingkan metode

deteksi posisi lainnya seperti *Time Difference Of Arrival* (TDOA) maupun *Enhanced Observed Time Difference* (E-OTD). A-GPS jauh lebih efisien dan efektif dalam mengakses informasi dari satelit karena tidak perlu mencari data satu persatu dari ke-24 satelit yang ada, namun A-GPS telah mengetahui sasaran (satelit) mana yang dibutuhkan atau dituju [ELR-06:125].

Pada sistim A-GPS, telepon genggam akan menangkap sinyal satelit yang lalu dikirimkan keserver penyedia layanan telepon, hasil perhitungan lokasi yang dilakukan oleh server dikirimkan kembali ke telepon genggam. Peta juga dapat dikirimkan oleh *server* tersebut atau sudah disimpan pada telepon genggam. Sistem ini hanya berfungsi bila jaringan telepon genggam mampu disediakan oleh pengguna. Proses perhitungan data dilakukan oleh *server* penyedia jaringan telepon, maka telepon genggam tidak memerlukan prosesor yang canggih [ELR-06:125-126].



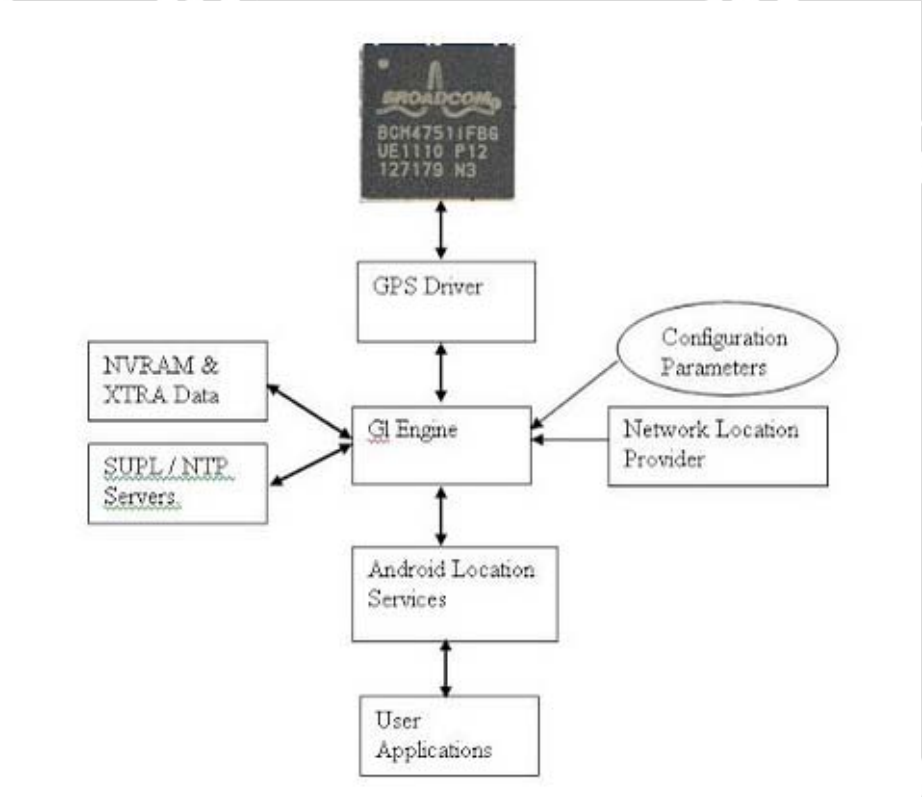
Gambar 2.16 *Assisted* GPS

Sumber: [DIG-09:1]

2.16.1. Cara Kerja GPS pada Android

GPS dalam android merupakan kesatuan beberapa komponen hardware dan software sehingga dapat menjadi semacam location service yang baik. Berikut adalah komponen2 GPS di android:

1. *Chip* GPS
2. GPS *Driver*
3. GL *Engine*
4. Android *Framework*
5. *User Application*



Gambar 2.17 GPS pada Android

Sumber: [ELR-06:125-126].

Chip GPS: Penerima frekuensi radio (radio *frequency*) yang secara langsung berkomunikasi dengan satelit GPS.

GPS *Driver*: semacam system software yang menggunakan low level API's yang mengkomunikasikan antara GPS *chip* dengan OS android. File *driver* GPS

biasanya terdapat pada /system/lib/hw atau /vendor/Lib/Hw. File tersebut biasanya mempunyai nama akhiran so dan awalan GPS, misalnya: gps.default.so atau gps.aries.so dan lain-lain tergantung vendor dan versi android yang digunakan.

GL *Engine*: merupakan system yang penting dalam komponen GPS android. Terdapat pada /system/bin dengan nama *file* glgps atau gpsd tergantung dari *platform* yang digunakan. Konfigurasi GL *Engine* bekerja sesuai parameter yang telah di-*setting* pada *file* dengan akhiran .xml atau .conf (mmisal: glconfic.xml, gps.xml, jupiter.xml, gpsconfig.xml, gps.conf, secgps.conf, dan lain-lain tergantung versi android yg digunakan) biasanya terletak pada /system/etc/gps atau vendor/etc. Sistem ini berfungsi untuk mengetahui lokasi yang diambil pada BTS *transmitter* operator telpon *(cell tower)* untuk membantu A-GPS mengunci lokasi. Kadang juga diperlukan koneksi internet untuk me-lock lokasi. GL *Engine* dapat mendeteksi beberapa satelit gps sekaligus, tapi untuk mengunci memerlukan beberapa informasi lain seperti ketinggian tempat waktu dan lain-lain yang mana informasi ini hanya bisa diperoleh dengan mengunduh dari salahsatu satelit tersebut. Kadang mengunduh informasi dari satelite gps perlu waktu yang lama, sehingga akan digunakan akses internet untuk memperoleh akses SUPL/NTP *server*. Setelah itu data akan disimpan di NVRAM untuk penggunaan selanjutnya.

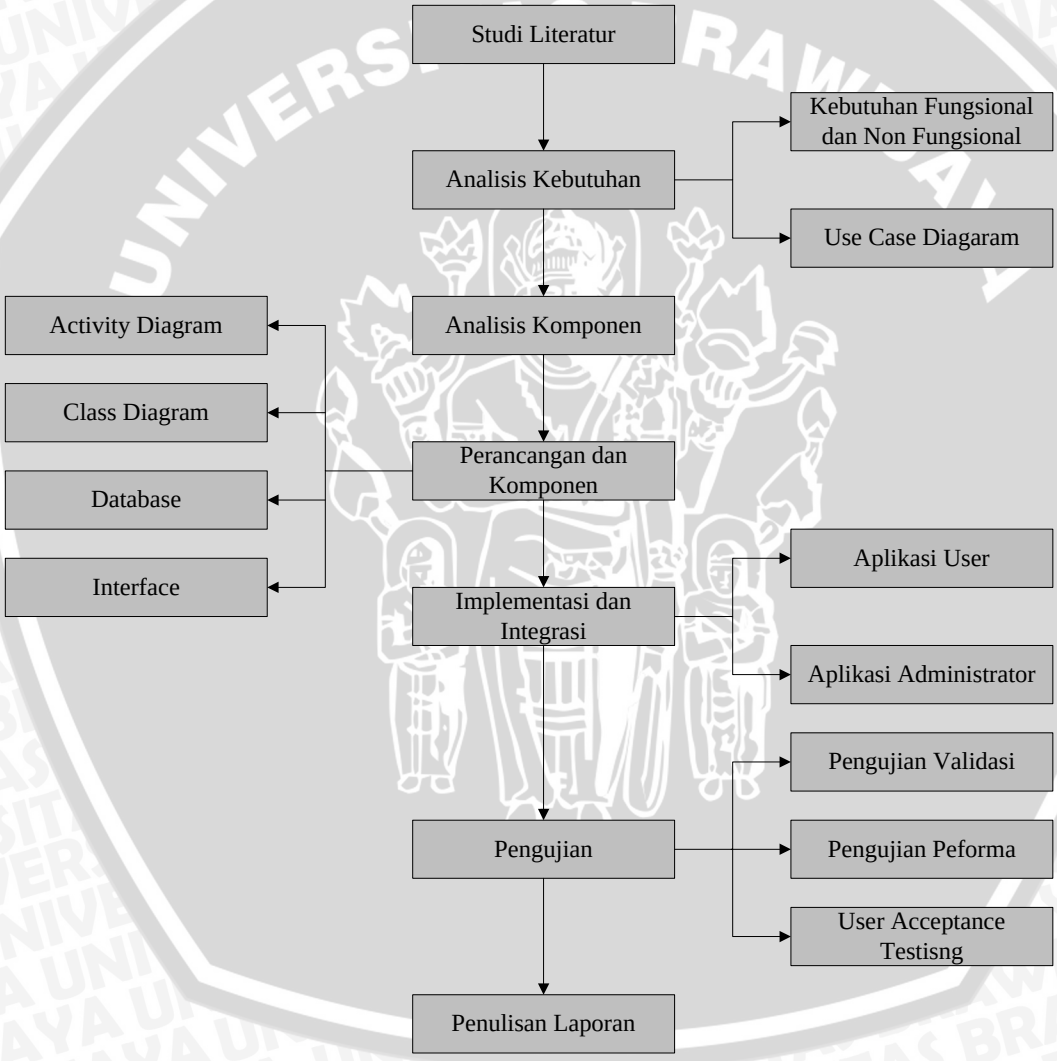
Android *Location Services*: terdiri dari framework classes seperti location manager yang dibutuhkan suatu aplikasi yang menggunakan GL *Engine*.

User Application: Applikasi android yang menggunakan GPS android seperti: google map, navitel, gps essential, dan lain-lain.

BAB III
METODE PENELITIAN DAN PERANCANGAN

3.1. Metode Penelitian

Pada bab ini dijelaskan langkah - langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dikembangkan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan perangkat lunak selanjutnya.



Gambar 3.1 Metode Penelitian CBSE *(Component Based Software Engineering)*

Metodologi penelitian yang digunakan adalah adaptasi dari pengembangan model CBSE *(Component Based Software Engineering).* Adaptasi yang dilakukan adalah menggunakan tahap-tahap yang terdapat pada gambar 3.1 di atas.

3.1.1. Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori - teori pendukung tersebut meliputi :

1. Perangkat Lunak
2. Rekayasa Perangkat Lunak
3. *Component-Based Software Engineering*
    a. *Domain Engineering*
        i. *Domain Analysis*
        ii. *Domain Model*
        iii. *Software Architecture Development*
        iv. *Structural Model*
        v. *Reusable Component Development*
        vi. *Repository Reusable Artifacts / Components*
    b. *Component-Based Development*
        i. *Analysis*
        ii. *Architectural Design*
        iii. *Component Qualification*
        iv. *Component Adaptation*
        v. *Component Engineering*
        vi. *Component Composition*
        vii. *Testing*
        viii. *Application Software*
        ix. *Component Update*
4. *Unified Modelling Language*
    a. *Use Case Diagram*
    b. *Class Diagram*
    c. *Activity Diagram*
5. Pengujian Perangkat Lunak

a. Teknik Pengujian
    1. *Black-Box Testing*
b. Strategi Pengujian
    1. Pengujian Validasi
    2. Pengujian Performa
    3. UAT *(User Acceptance Testing)*

6. *Webservice*
7. Bahasa Pemrograman HTML dan CSS
8. Bahasa Pemrograman PHP
9. Bahasa Pemrograman Java
10. Basis Data
11. MySQL
12. SQLite
13. Google Maps Api
14. Sistem Operasi Android
    a. Pengenalan Android
    b. Fitur Android
    c. Arsitektur Android
        1. *Applications*
        2. *Application Framework*
        3. *Libraries*
        4. Android *Runtime*
        5. *Linux Kernel*
    d. Fundamental Aplikasi
        1. *Activities*
        2. *Services*
        3. *Broadcast Receiver*
        4. *Content Provider*
15. GPS
    a. Sistem Satelit GPS
    b. Sinyal Satelit GPS

1. *Carriers*
2. *Pseudo-Random Codes*
3. *Navigation Message*

c. Cara Kerja GPS
d. Sinyal Menentukan Lokasi
e. Model dan Interkoneksi GPS
f. *Tracking System*

16. *Asissted-Global Positioning System* (A-GPS)

3.1.2. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem perangkat lunak yang akan dibangun. Metode analisis yang digunakan adalah *object-oriented analysis* dengan menggunakan bahasa pemodelan UML *(Unified Modeling Language).* Diagram *use case* digunakan untuk mendeskripsikan kebutuhan - kebutuhan dan fungsionalitas perangkat lunak dari perspektif *end-user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem perangkat lunak GPS *Tracking* Berbasis Android yang kemudian akan dimodelkan dalam diagram *use case*. Diagram *use case* perangkat lunak GPS *Tracking* Berbasis Android terbagi menjadi 2 buah subsistem, yaitu subsistem *user* dan subsistem *administrator.* Tiap *use case* dalam diagram *use case* tersebut juga akan dijelaskan lebih rinci dalam skenario *use case.* Analisis penyimpanan data pada perangkat lunak GPS *Tracking* Berbasis Android juga dilakukan pada tahap ini.

3.1.3. Perancangan

Perancangan perangkat lunak dilakukan setelah semua kebutuhan perangkat lunak didapatkan melalui tahap analisis kebutuhan. Perancangan perangkat lunak berdasarkan *object-oriented analysis* dan *object-oriented design* yaitu menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan dimulai dari perancangan struktur basis data berdasarkan analisis penyimpanan data yang telah dilakukan. Kemudian dilanjutkan dengan perancangan sistem perangkat lunak GPS *Tracking* Berbasis Android yang dilakukan dengan mengidentifikasi *class – class* dan *interface – interface* yang dibutuhkan, serta

kemudian dimodelkan dalam diagram *class*. Hubungan interaksi antar objek yang telah diidentifikasi, dimodelkan dalam *activity diagram* dan kemudian tahap perancangan dilanjutkan dengan perancangan antarmuka pengguna.

3.1.4. Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Perancangan perangkat lunak digunakan sebagai dasar acuan implementasi perangkat lunak yang akan dilakukan. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan pengembangan perangkat lunak. Kemudian dilanjutkan dengan implementasi basis data dengan menggunakan MySQL dan SQLite. Selanjutnya dijabarkan *mapping class* dengan *file* program saat implementasi perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman Java pada subsistem *user* dan pemrograman HTML, CSS, dan PHP pada subsistem *administrator*. Tahap terakhir dari implementasi adalah implementasi antarmuka berdasarkan perancangan yang telah dilakukan.

3.1.5. Pengujian dan Analisis

Pengujian perangkat lunak dilakukan untuk mengetahui apakah kinerja dan performa sistem perangkat lunak GPS *Tracking* Berbasis Android telah memenuhi spesifikasi kebutuhan yang melandasinya. Strategi pengujian perangkat lunak yang akan digunakan yaitu pengujian UAT, pengujian performa, dan pengujian validasi (*validation testing*). Metode pengujian yang akan digunakan adalah *black-box testing*. Proses pengujian perangkat lunak dilakukan dengan pengujian validasi dan berakhir pada pengujian peforma. Pada tahap pengujian validasi digunakan metode *black-box testing.* Analisis juga dilakukan untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dari pengembangan perangkat lunak yang telah dilakukan.

3.1.6. Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang

dimaksudkan untuk memperbaiki kesalahan – kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan perangkat lunak lebih lanjut.

3.2. Perancangan

Bab ini membahas mengenai perancangan perangkat lunak GPS *Tracking* Berbasis Andorid. Perancangan yang dilakukan meliputi dua tahap. Proses analisis kebutuhan dilakukan pada tahap pertama dan tahap kedua adalah proses perancangan perangkat lunak. Tahap analisis kebutuhan terdiri atas lima langkah yaitu melakukan penjabaran tentang gambaran umum perangkat lunak GPS *Tracking* Berbasis Andorid, melakukan proses identifikasi aktor yang terlibat dalam sistem perangkat lunak, melakukan proses analisis data yang diperlukan, membuat daftar kebutuhan pengguna berdasarkan penjabaran gambaran umum perangkat lunak dan menggunakan pemodelan diagram *use case* untuk menggambarkan kebutuhan tersebut. Proses perancangan perangkat lunak memiliki lima tahap, yaitu perancangan arsitektural, perancangan basis data, pemodelan diagram *class* untuk menggambarkan perancangan struktur *class* – *class* yang menyusun perangkat lunak GPS *Tracking* Berbasis Andorid, pemodelan diagram *sequence* untuk menggambarkan interaksi antar objek atau *class* di dalam perangkat lunak GPS *Tracking* Berbasis Andorid, dan perancangan antarmuka pengguna dari perangkat lunak GPS *Tracking* Berbasis Andorid.

3.2.1. Analisis Kebutuhan

Proses analisis kebutuhan mengacu pada gambaran umum perangkat lunak GPS *Tracking* Berbasis Andorid dan hasil pengumpulan, pemahaman dan penetapan kebutuhan – kebutuhan yang ingin didapatkan oleh pengguna. Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum perangkat lunak GPS *Tracking* Berbasis Andorid, identifikasi aktor yang terlibat, analisis data yang akan disimpan, penjabaran tentang daftar kebutuhan dan kemudian memodelkannya ke dalam diagram *use case.* Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan – kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

3.2.2. Gambaran Umum Perangkat Lunak GPS *Tracking* Berbasis Andorid

Pembahasan gambaran umum perangkat lunak GPS *Tracking* Berbasis Andorid terdiri atas dua bagian, yaitu deskripsi umum perangkat lunak GPS *Tracking* Berbasis Andorid dan cara penggunaan perangkat lunak GPS *Tracking* Berbasis Andorid.

3.2.2.1 Deskripsi Perangkat Lunak GPS *Tracking* Berbasis Andorid

Perangkat lunak yang akan dikembangkan dalam proyek skripsi ini adalah perangkat lunak GPS *Tracking* berbasis Andorid. Perangkat lunak GPS *Tracking* Berbasis Andorid adalah perangkat lunak yang dapat digunakan untuk mendata lahan dan mencari data koordinat lahan klien pada *mobile phone* bersistem operasi Android yang nantinya dikrim ke dalam *webservice* kantor Badan Pertanahan Nasional dan dapat menampilkan bidang lahan/*polygon* pada *Google Maps Api*.

Perangkat lunak GPS *Tracking* berbasis Andorid memiliki 2 bagian utama yaitu, subsistem aplikasi *User* dan subsistem aplikasi *Administrator*.

1. Aplikasi *User*

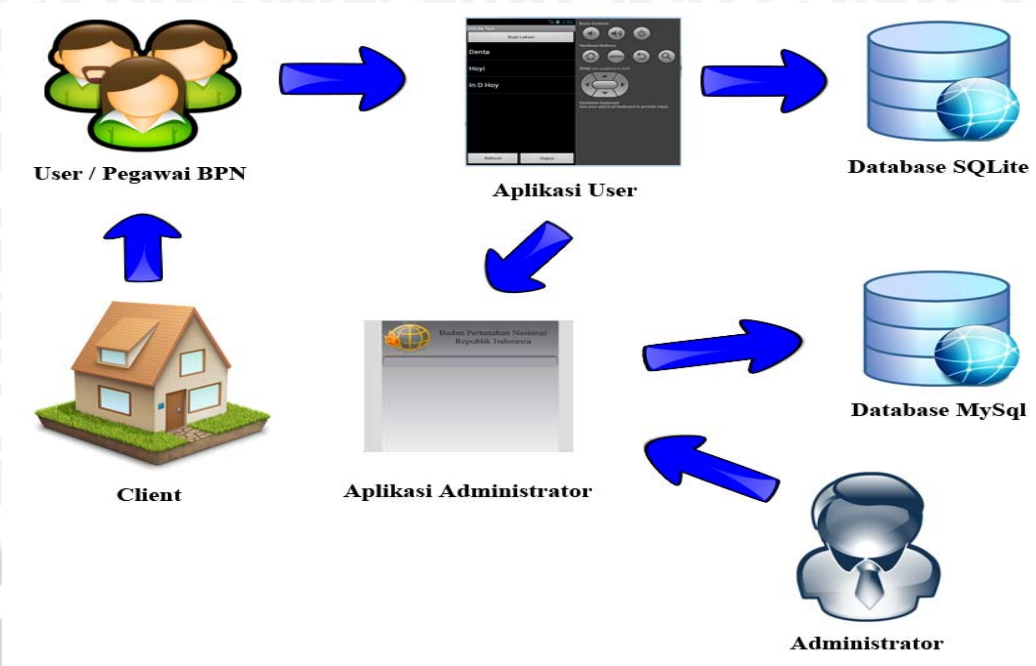
   Aplikasi *User* adalah aplikasi GPS *Tracking* Berbasis Andorid yang dapat digunakan untuk mendata lahan dan mencari data koordinat lahan klien pada *mobile phone* bersistem operasi Android yang nantinya dikrim ke *webservice* kantor Badan Pertanahan Nasional dan dapat menampilkan bidang lahan/*polygon* pada *Google Maps Api*.

2. Aplikasi *Administrator*

   Aplikasi *Administrator* dari perangkat lunak GPS *Tracking* Berbasis Andorid adalah aplikasi yang digunakan untuk mencari dan melihat informasi data klien yang diperoleh dari perangkat lunak GPS *Tracking* Berbasis Andorid.

3.2.2.2. Cara Penggunaan Perangkat Lunak GPS *Tracking* Berbasis Andorid

Perangkat lunak GPS *Tracking* Berbasis Andorid memiliki urutan langkah kerja seperti yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Urutan langkah kerja perangkat lunak GPS *Tracking* Berbasis Andorid

Pada perangkat lunak GPS *Tracking* Berbasis Andorid, data dan koordinat klien dikelola oleh *user* melalui aplikasi *user*. Kemudian, data dan koordinat klien yang dikelola disimpan pada *database SQLite*. Selanjutnya data dan koordinat klien tersebut dikirim ke aplikasi *administrator/webservice* kantor Badan Pertanhan Nasional sebagai dasar menampilkan bidang/*polygon* lahan di dalam *google maps api* yang sudah ada dalam aplikasi *administrator*.

3.2.3. Identifikasi Aktor

Tahap ini adalah tahap untuk melakukan identifikasi terhadap aktor - aktor yang akan berinteraksi dengan perangkat lunak GPS *Tracking* Berbasis Andorid. Tabel 3.1 memperlihatkan aktor – aktor yang terlibat beserta penjelasannya masing-masing yang merupakan hasil dari proses identifikasi aktor.

Tabel 3.1 Identifikasi aktor

Aktor	Deskripsi
User	*User* adalah sebagai pegawai/petugas lapangan di dalam kantor Badan Pertanahan yang hanya bisa menggunakan

	aplikasi *user* dari perangkat lunak GPS *Tracking* Berbasis Andorid.
Administrator	*Administrator* adalah pengguna aplikasi *administrator* yaitu *webservice* kantor Badan Pertanahan yang memiliki hak untuk mengelola data dan koordinat klien dari perangkat lunak GPS *Tracking* Berbasis Andorid lebih lanjut.

3.2.4. Analisis Data

Analisis data bertujuan untuk mendapatkan struktur penyimpanan data yang dibutuhkan perangkat lunak GPS *Tracking* Berbasis Andorid. Struktur penyimpanan data pada perangkat lunak GPS *Tracking* Berbasis Andorid disusun berdasarkan analisis data sebagai berikut :

a. Aplikasi *User*
   - Data klien yang terdiri dari *id*, nama, dan alamat klien.
   - Data lokasi yang terdiri atas *id*_lokasi, id_kilen, dan alamat_lokasi klien
   - .Data Koordinat terdiri dari id_lokasi, longitude, dan latitude.

b. Aplikasi *administrator*
   - Data pegawai yang terdiri atas *username*, *password*, dan nama.
   - Data klien yang terdiri dari *id*_klien, nama, dan alamat klien.
   - Data lokasi yang terdiri dari *id*_lokasi, longitude, latitude, *username*_pegawai, *id*_klien, dan waktu_terima.

3.2.5. Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional. Pada daftar kebutuhan akan dispesifikasikan menjadi dua yaitu spesifikasi kebutuhan fungsional *user* dan spesifikasi kebutuhan fungsional *administrator*. Spesifikasi kebutuhan fungsional *user* ditunjukkan pada Tabel 3.2.

Tabel 3.2 Spesifikasi kebutuhan fungsional *user*

Nomor SRS	Kebutuhan	*Use Case*
SRS_001_01	Perangkat lunak harus menyediakan fasilitas untuk login sehingga hanya *user* saja yang dapat mengelola sistem.	*Login*
SRS_001_02	Perangkat lunak harus menyediakan fasilitas untuk menambah informasi dari data klien.	Menambah Informasi Data Klien
SRS_001_03	Perangkat lunak harus menyediakan fasilitas untuk menghapus informasi dari data klien.	Menghapus Informasi Data Klien
SRS_001_04	Perangkat lunak harus menyediakan fasilitas untuk pengiriman informasi data klien.	Mengirim Informasi Data Klien
SRS_001_05	Perangkat lunak dapat menghapus session dari *user*.	*Logout*

Spesifikasi kebutuhan fungsional *administrator* ditunjukkan pada Tabel 3.3.

Tabel 3.3 Spesifikasi kebutuhan fungsional *administrator*

Nomor SRS	Kebutuhan	*Use Case*
SRS_002_01	Sistem harus menyediakan fasilitas untuk membuat akun pegawai.	Pembuatan Akun Pegawai
SRS_002_02	Sistem harus menyediakan fasilitas untuk melihat semua informasi data pegawai.	Melihat Informasi Data Pegawai
SRS_002_03	Sistem harus menyediakan fasilitas untuk melihat semua informasi data klien.	Melihat Informasi Data Klien
SRS_002_04	Sistem harus menyediakan fasilitas untuk mencari dan mendapatkan semua data kiriman dari aplikasi *user*.	Melihat Seluruh Informasi Data Kiriman
SRS_002_05	Sistem harus menyediakan fasilitas untuk melihat peta dan luas lokasi dari koordinat yang dikirim pada aplikasi *user*.	Melihat Peta dan Luas Lokasi

Daftar kebutuhan non-fungsional perangkat lunak GPS *Tracking* Berbasis Andorid ditunjukkan pada Tabel 3.4.

Tabel 3.4 Spesifikasi kebutuhan non-fungsional

Parameter	Deskripsi Kebutuhan
Availability	Perangkat lunak harus dapat beroperasi terus - menerus selama waktu yang diinginkan (24 jam atau bahkan melebihi jam kerja).
Response Time	Perangkat lunak harus dapat berjalan dengan waktu proses tidak lebih dari 10 detik dengan *bandwith* minimal 64 *kbps*.

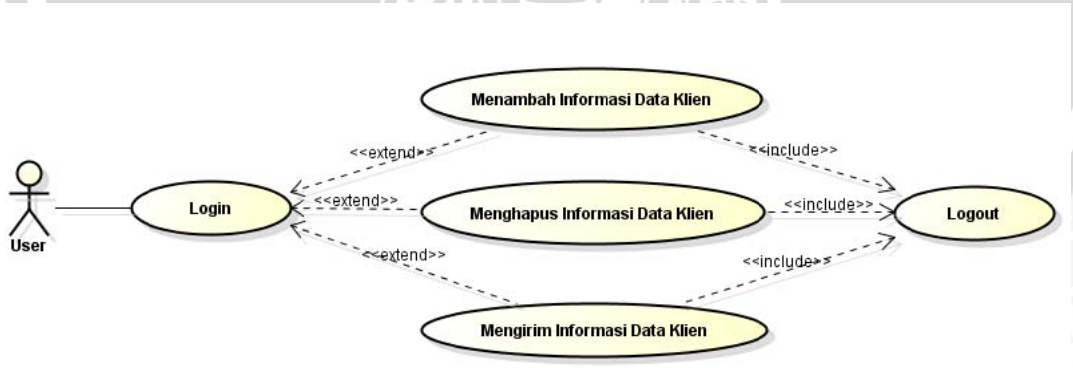
Control	Membatasi akses penggunaan perangkat lunak dengan meyediakan 2 sisi aplikasi, yaitu *user* dan *administrator*.
Usability	Perangkat lunak harus dapat dengan mudah digunakan pengguna.

3.2.6. Diagram *Use Case*

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku sistem. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor. Pemodelan diagram *use case* yang menggambarkan fungsionalitas perangkat lunak GPS *Tracking* Berbasis Andorid dibagi menjadi dua yaitu diagram *use case* untuk aplikasi *user* dan diagram *use case* untuk aplikasi *administrator*.

3.2.6.1. Diagram *Use Case* Aplikasi *User*

Diagram *use case* ini melibatkan *user* sebagai aktor dan 5 buah *use case*. 5 buah *use case* ini termasuk dalam bagian perangkat lunak GPS *Tracking* Berbasis Andorid. Satu buah *use case* ini juga akan disertai dengan skenario *use case* untuk menjelaskan rangkaian aktivitas yang terjadi di *use case* tersebut. Diagram *use case* untuk aplikasi *user* ditunjukkan pada Gambar 3.3 berikut ini :



Gambar 3.3 Diagram *use case user*

1. Skenario *Use Case* Menunjukkan *Login*

Tabel 3.5 *Use case* Menunjukkan *Login*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_01
Nama	*Login*
Tujuan	Untuk menyeleksi *user* yang sah.
Deskripsi	*Use case* ini menjelaskan bagaimana *user* melakukan login untuk dapat menampilkan halaman utama *user*.
Aktor	*User*
Skenario Utama	
Kondisi Awal	Perangkat lunak memunculkan halaman *login*.
Aksi Aktor	Reaksi Sistem
User memasukkan data login (*username* dan *password*), lalu menekan tombol "*Login*".	1. Perangkat lunak menerima data login dan kemudian aplikasi *administrator* melakukan pengecekan terhadap data yang dimasukkan oleh *user*. Jika data benar, maka akan ditampilkan halaman utama *user*.
Skenario Alternatif 1 : Jika *username* atau *password* kosong	
	2. Perangkat lunak akan menampilkan pesan peringatan bahwa *username* atau *password* kosong.
Skenario Alternatif 2 : Jika *username* dan *password* salah	
	3. Perangkat lunak akan menampilkan pesan peringatan bahwa *username* dan *password* salah.
Kondisi Akhir	Sistem akan menampilkan halaman utama dari aplikasi *user*.

2. Skenario *Use Case* Menambah Informasi Data Klien di *Database*

Tabel 3.6 *Use case* Menambah Informasi Data Klien di *Database*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_02
Nama	Menambah informasi data klien di database.
Tujuan	Untuk menambakan informasi data klien.
Deskripsi	*Use Case* ini menjelaskan bagaimana *user* menambahkan informasi data tentang klien.
Aktor	*User*
Skenario Utama	
Kondisi Awal	*User* berhasil melakukan login dan masuk halaman utama *user*.
Aksi Aktor	Reaksi Sistem
User menekan tombol "Buat Lokasi" dan berhasil masuk halaman tambah data lokasi.	1. Perangkat lunak menerima informasi data dan penambahan data klien yang akan dikirim ke aplikasi *administrator*.
Skenario Alternatif 1 : Jika *user* menekan tombol *back*	
	2. Perangkat lunak akan kembali ke halaman utama.
Kondisi Akhir	Informasi data klien berhasil ditambahkan pada aplikasi *user*.

3. Skenario *Use Case* Menghapus Informasi Data Klien di *Database*

Tabel 3.7 *Use case* Menghapus Informasi Data Klien di *Database*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_03
Nama	Menghapus informasi data klien di database.
Tujuan	Untuk menghapus informasi data klien.

Deskripsi	*Use Case* ini menjelaskan bagaimana *user* menghapus informasi data tentang klien.
Aktor	*User*
Skenario Utama	
Kondisi Awal	*User* berhasil melakukan login dan masuk halaman. utama *user*.
Aksi Aktor	Reaksi Sistem
User menekan tombol "Hapus/Hapus Semua" pada *context* menu aplikasi.	Perangkat lunak menerima data berupa *id* informasi kemudian melakukan proses penghapusan data dengan *id* informasi tersebut.
Kondisi Akhir	Informasi data klien berhasil dihapus dan halaman utama kembali kosong.

4. Skenario *Use Case* Mengirim Informasi Data Klien di *Database*

Tabel 3.8 *Use case* Mengirim Informasi Data Klien di *Database*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_04
Nama	Mengirim informasi data klien di database.
Tujuan	Untuk mengirim informasi data klien ke aplikasi *administrator*.
Deskripsi	*Use Case* ini menjelaskan bagaimana *user* mengirim informasi data tentang klien ke aplikasi *administrator*.
Aktor	*User*
Skenario Utama	
Kondisi Awal	*User* berhasil melakukan login dan masuk halaman. utama *user*.
Aksi Aktor	Reaksi Sistem
User memilih salah satu nama dari *Klien*	Perangkat lunak mengirim data berupa *id* informasi klien dan melakukan proses pengiriman data dengan *id*

dan kemudian menekan tombol "kirim".	informasi tersebut ke aplikasi *administrator*.
Kondisi Akhir	Informasi data klien berhasil dikirim pada aplikasi *user*.

5. Skenario *Use Case Logout*

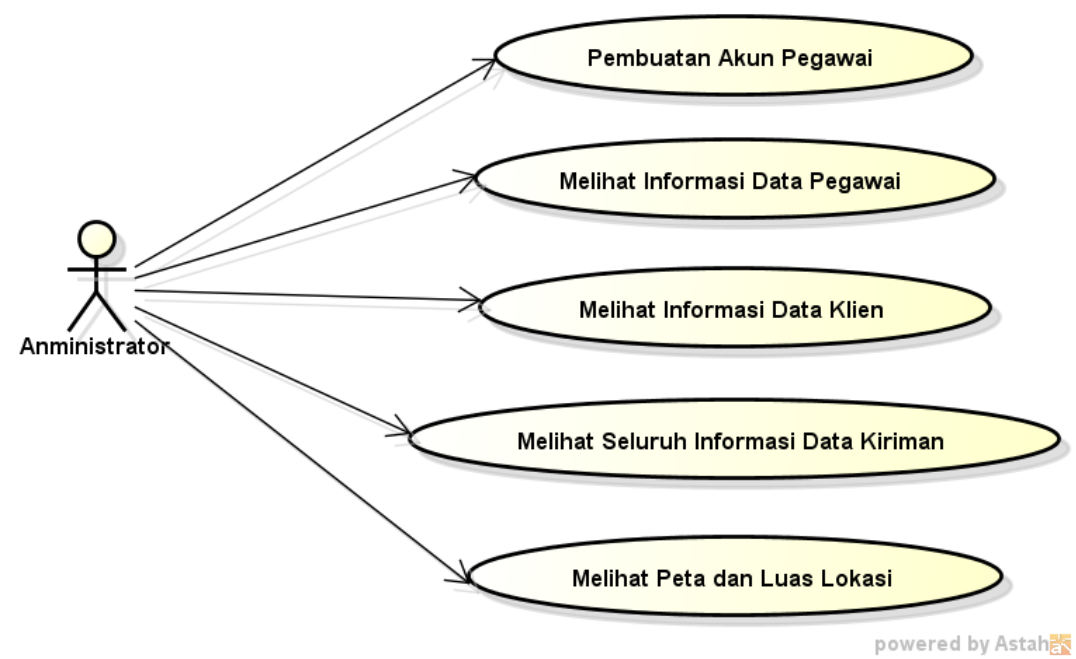
Tabel 3.9 *Use case Logout*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_05
Nama	*Logout*
Tujuan	Untuk melakukan *logout*.
Deskripsi	*Use Case* ini menjelaskan bagaimana *user* melakukan *logout* .
Aktor	*User*
Skenario Utama	
Kondisi Awal	*User* berhasil melakukan login dan masuk halaman utama *user*.
Aksi Aktor	Reaksi Sistem
User menekan tombol "*Logout*" pada *context* menu aplikasi.	Perangkat lunak melakukan penghapusan *session user* dan masuk halaman *login*.
Kondisi Akhir	*User* berhasil *logout* dan tidak dapat kembali kehalaman utama tanpa melakukan *use case login*.

3.2.6.2. Diagram *Use Case* Aplikasi *Administrator*

Diagram *use case* ini melibatkan *administrator* sebagai aktor dan 5 buah *use case*. 5 buah *use case* ini termasuk dalam bagian perangkat lunak GPS *Tracking* Berbasis Andorid. 5 buah *use case* ini juga akan disertai dengan skenario *use case* untuk menjelaskan rangkaian aktivitas yang terjadi di masing -

masing *use case* tersebut. Diagram *use case* untuk aplikasi *administrator* ditunjukkan pada Gambar 3.4 berikut ini :



Gambar 3.4 Diagram *use case administrator*

1. Skenario *Use Case* Membuat Akun Pegawai

Tabel 3.10 *Use Case* Membuat Akun Pegawai

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_01
Nama	Membuat Akun Pegawai.
Tujuan	Untuk membuat akun pegawai.
Deskripsi	*Use Case* ini menjelaskan bagaimana *administrator* melakukan pembuatan akun untuk pegawai atau *user*.
Aktor	*Administrator*
Skenario Utama	
Kondisi Awal	*Administrator* masuk ke halaman utama aplikasi *administrator*.
Aksi Aktor	Reaksi Sistem
Administrator	Sistem akan menampilkan *form* membuat akun

memilih kata "Buat Akun".	pegawai.
Kondisi Akhir	Sistem menyimpan akun yang dibuat ke *database*.

2. Skenario *Use Case* Melihat Informasi Data Pegawai

Tabel 3.11 *Use Case* Melihat Informasi Data Pegawai

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_02
Nama	Melihat Informasi Data Pegawai.
Tujuan	Untuk melihat informasi data pegawai.
Deskripsi	*Use Case* ini menjelaskan bagaimana *administrator* melakukan pencarian informasi data Pegawai.
Aktor	*Administrator*
Skenario Utama	
Kondisi Awal	*Administrator* masuk ke halaman utama aplikasi *administrator*.
Aksi Aktor	Reaksi Sistem
Administrator melihat informasi data klien dan menekan tombol "Data Pegawai".	Sistem akan menampilkan halaman data pegawai.
Kondisi Akhir	Sistem akan menampilkan data pegawai (*username*, nama, dan *password*).

3. Skenario *Use Case* Melihat Informasi Data Klien

Tabel 3.12 *Use Case* Melihat Informasi Data Klien

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_03
Nama	Melihat Informasi Data klien.
Tujuan	Untuk melihat informasi data klien.

Deskripsi	*Use Case* ini menjelaskan bagaimana *administrator* melakukan pencarian informasi data Klien yang akan ditampilkan pada *Google Maps Api*.
Aktor	*Administrator*
Skenario Utama	
Kondisi Awal	*Administrator* masuk ke halaman utama aplikasi *administrator*.
Aksi Aktor	Reaksi Sistem
Administrator melihat informasi data klien dan menekan tombol "Data Klien".	Sistem akan menampilkan halaman data klien.
Kondisi Akhir	Sistem akan menampilkan data klien (No. KTP, nama, dan alamat klien).

4. Skenario *Use Case* Melihat Seluruh Informasi Data Kiriman

Tabel 3.13 *Use Case* Melihat Seluruh Informasi Data Kiriman

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_04
Nama	Melihat Seluruh Informasi Data Kiriman.
Tujuan	Untuk melihat informasi data Kiriman.
Deskripsi	*Use Case* ini menjelaskan bagaimana *administrator* melakukan pencarian seluruh informasi data kiriman.
Aktor	*Administrator*
Skenario Utama	
Kondisi Awal	*Administrator* masuk ke halaman utama aplikasi *administrator*.
Aksi Aktor	Reaksi Sistem
Administrator menekan "Logo BPN"	Sistem akan menampilkan seluruh informasi data kiriman.

Kondisi Akhir	Sistem akan menampilkan menampilkan seluruh informasi data kiriman.

5. Skenario *Use Case* Melihat Peta dan Luas Lokasi

Tabel 3.14 *Use Case* Melihat Peta dan Luas Lokasi

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_05
Nama	Melihat Peta dan Luas Lokasi.
Tujuan	Untuk melihat peta dan luas lokasi.
Deskripsi	*Use Case* ini menjelaskan bagaimana *administrator* melakukan penelusuran dari data kiriman yaitu koordinat untuk dikonversikan menjadi sebuah bidang pada peta sekaligus luas lokasi.
Aktor	*Administrator*
Skenario Utama	
Kondisi Awal	*Administrator* masuk ke halaman utama aplikasi *administrator*.
Aksi Aktor	Reaksi Sistem
Administrator menekan tombol “Lihat Peta”	Sistem akan mengkonversi koordinat yang ada ke dalam Google Maps.
Kondisi Akhir	Sistem akan menampilkan sebuah bidang pada peta Google Maps sekaligus luas bidang lokasi tersebut.

3.2.7. Perancangan Perangkat Lunak

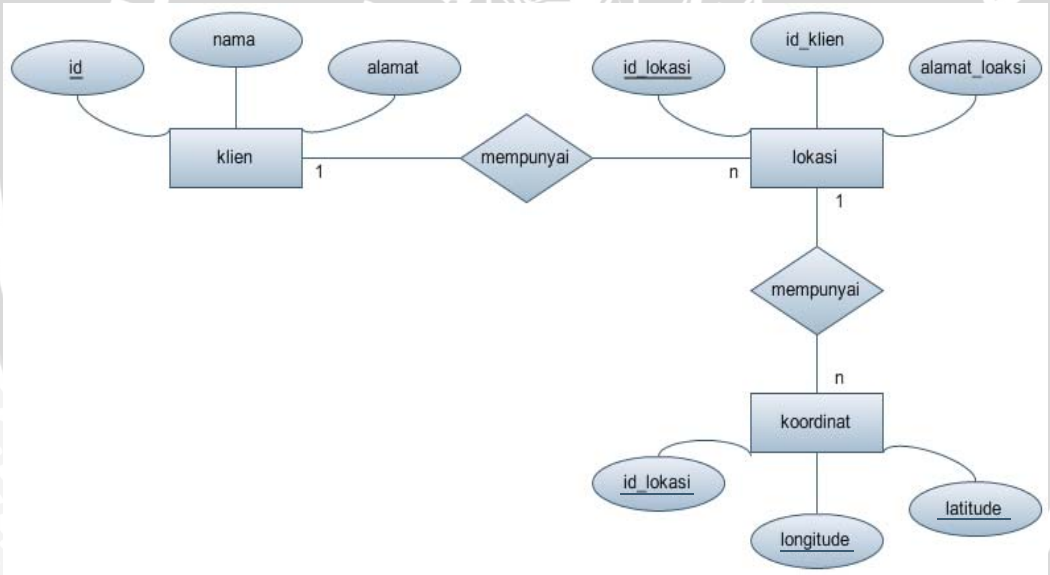
Perancangan perangkat lunak dilakukan dalam empat tahap, yaitu perancangan basis data, pemodelan diagram *class* untuk menggambarkan perancangan struktur *class* – *class* yang menyusun perangkat lunak perangkat lunak GPS *Tracking* Berbasis Andorid, pemodelan diagram *activity* memungkinkan untuk menentukan bagaimana sistem akan mencapai tujuannya di dalam perangkat lunak perangkat lunak GPS *Tracking* Berbasis Andorid, dan

perancangan antarmuka pengguna dari perangkat lunak perangkat lunak GPS *Tracking* Berbasis Andorid. Perancangan perangkat lunak pada skripsi ini menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan menggunakan UML (*Unified Modelling Language*).

3.2.7.1. Perancangan Basis Data

Basis data berfungsi sebagai tempat menyimpan data. Perancangan basis data digunakan untuk merancang basis data yang akan dibuat agar masukan dan keluaran program sesuai dengan apa yang diharapkan. Perancangan basis data mengambil acuan dari proses analisis data yang dilakukan pada tahap analisis kebutuhan. Dalam perangkat lunak GPS Tracker berbasis Android terdapat 2 basis data yaitu basis data pada aplikasi klien disebut SQLite dan basis data pada aplikasi *administrator* disebut MySql.

3.2.7.1.1. Basis Data Aplikasi *User*



Gambar 3.5 Diagram Entity Relationship Aplikasi *User*

Rancangan masing – masing tabel adalah sebagai berikut :

1. Tabel Klien

Nama tabel : Klien

Jumlah *field* : 3

Fungsi : Untuk menyimpan data klien

Tabel 3.15 Struktur tabel klien

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	Id	TEXT	-	Id Klien
2	Nama	TEXT	-	Nama Klien
3	Alamat	TEXT	-	Alamat Klien

2. Tabel Lokasi

Nama tabel : Lokasi

Jumlah *field* : 3

Fungsi : Untuk menyimpan data lokasi

Tabel 3.16 Struktur tabel lokasi

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	id_lokasi	TEXT	-	Id Lokasi
2	id_klien	TEXT	-	Id Klien
3	alamat_lokasi	TEXT	-	Alamat Lokasi

3. Tabel Koordinat

Nama tabel : Koordinat

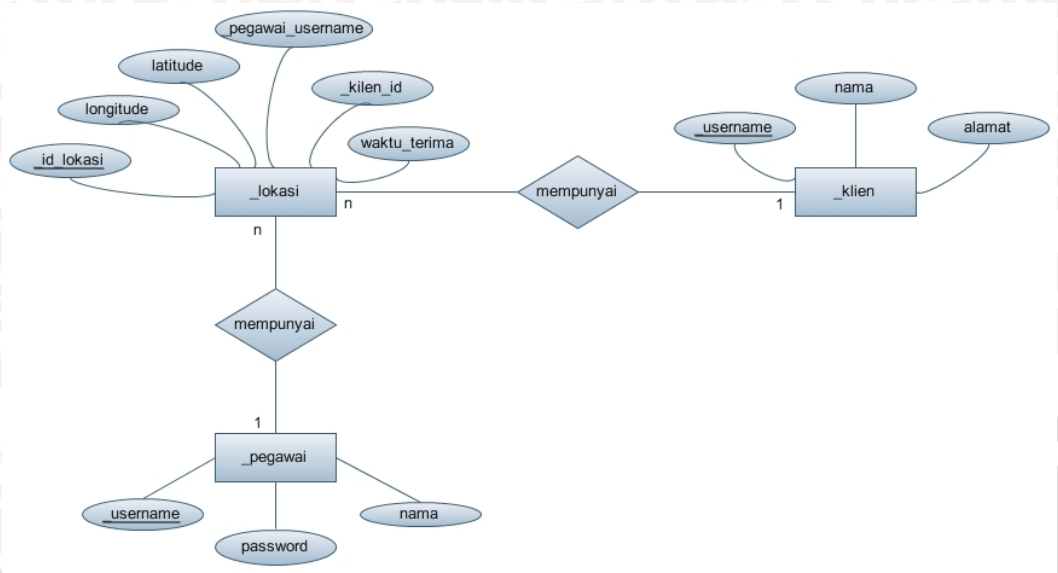
Jumlah *field* : 3

Fungsi : Untuk menyimpan data koordinat

Tabel 3.17 Struktur tabel koordinat

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	id_lokasi	TEXT	-	Id Lokasi
2	Longitude	REAL	-	longitude
3	Latitude	REAL	-	latitude

3.2.7.1.2. Basis Data Aplikasi *Administrator*



Gambar 3.6 Diagram Entity Relationship Aplikasi *Administrator*

Rancangan masing – masing tabel adalah sebagai berikut :

1. Tabel Klien

Nama tabel : Klien

Jumlah *field* : 3

Fungsi : Untuk menyimpan data klien

Tabel 3.18 Struktur tabel klien

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	_id	varchar	255	Id Klien
2	Nama	varchar	255	Nama Klien
3	Alamat	varchar	255	Alamat Klien

2. Tabel Pegawai

Nama tabel : Pegawai

Jumlah *field* : 3

Fungsi : Untuk menyimpan data pegawai

Tabel 3.19 Struktur tabel pegawai

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	_username	varchar	255	*Username* Pegawai
2	Password	varchar	255	*Password* Pegawai
3	Nama	varchar	255	Nama Pegawai

3. Tabel Lokasi

Nama tabel : Lokasi

Jumlah *field* : 3

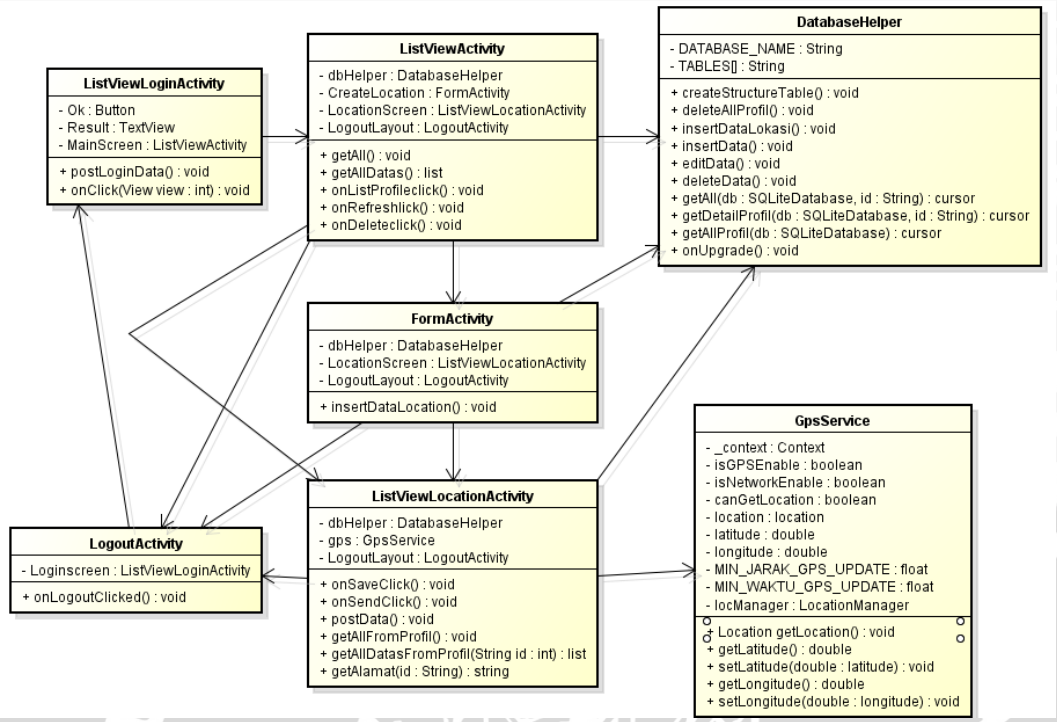
Fungsi : Untuk menyimpan data lokasi

Tabel 3.20 Struktur tabel lokasi

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	_id_lokasi	varchar	255	*Username* lokasi
2	Latitude	double	-	latitude lokasi
3	Longitude	double	-	longitude lokasi
4	_pegawai_username	varchar	255	*Username* Pegawai
5	_klien_id	varchar	255	Id Klien
6	Waktu_terima	timestamp	-	Waktu Terima

3.2.7.2 Diagram *Class*

Diagram *class* memberikan gambaran pemodelan elemen – elemen *class* yang membentuk sebuah perangkat lunak. *Class* bisa didapatkan dengan menganalisis secara detail terhadap *use case* yang dimodelkan. Gambar 3.6 menunjukkan diagram *class* dari perangkat lunak GPS *Tracker* berbasis Android yang akan dibuat.



Gambar 3.7 Diagram c*lass* perangkat lunak GPS *Tracker* berbasis Android

Pada diagram *class* di atas digambarkan beberapa *class* utama yang menyusun perangkat lunak GPS *Tracker* berbasis Android. Deskripsi beberapa *class* utama akan dijelaskan pada table sebagai berikut :

Tabel 3.21 Deskripsi Diagram *class* Perangkat Lunak GPS *Tracker* berbasis Android.

No	Kelas	Deskripsi
1	ListViewLoginActivity	*Class* ini menyediakan *form* untuk melakukan *login* dengan memasukkan *username* dan *password* yang terdapat pada *database administrator* dan untuk masuk ke dalam *mainscreen* atau halaman utama.
2	ListViewActivity	*Class* ini berfungsi untuk membuat informasi data *Klien*.
3	FormActivity	*Class* ini menyediakan *form* untuk melakukan pengisian data *Klien* dengan memasukkan nama dan alamat *Klien*.

4	ListViewLocationActivity	*Class* ini berfungsi untuk pencarian koordinat lokasi dan digunakan untuk pengiriman informasi data *Klien* ke *administrator*.
5	GpsService	*Class* ini adalah *Class* yang menangani kerja dari sistem GPS *Tracking* pada Andorid.
6	DatabaseHelper	*Class* ini berfungsi untuk menyimpan semua informasi data Klien (pembuatan data, pengeditan data, dan penghapusan data).
7	LogoutActivity	*Class* ini berfungsi untuk men-*destroy* semua aktivitas *user* dan kembali ke dalam *loginscreen.*

3.2.7.3. Diagram *Activity*

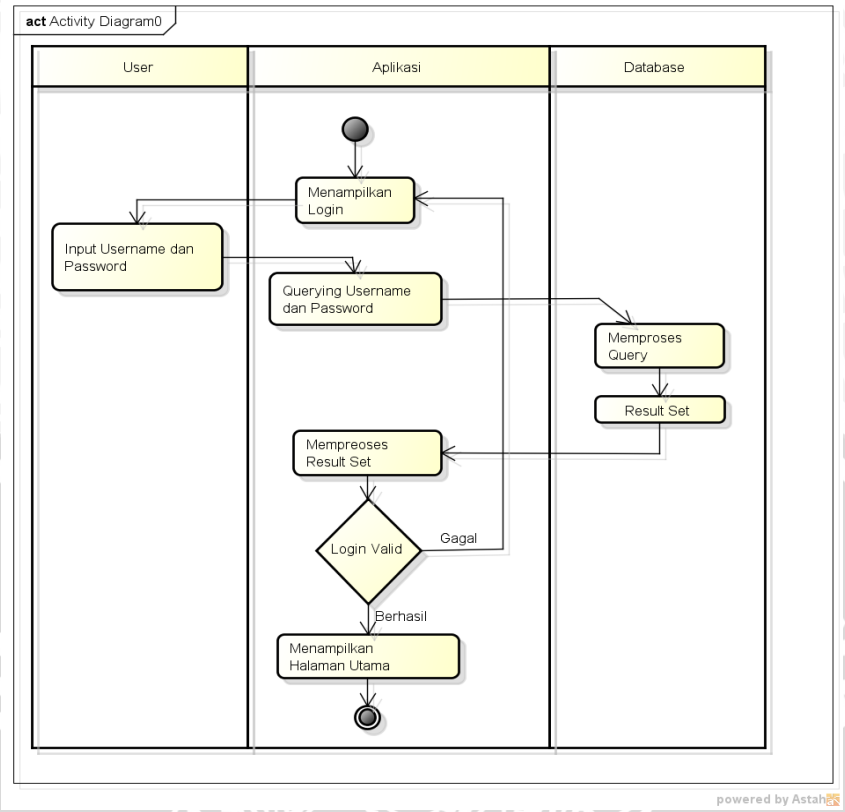
Pembuatan *activity diagram* ini bertujuan untuk menggambarkan urutan aktivitas dari proses pada setiap *use case* yang ada. Berikut merupakan *activity diagram* dari *user* dan *administrator.*

3.2.7.3.1 *Activity Diagram User*

Pada *Activity Diagram User* terdapat 5 *Activity Diagram* yaitu *Activity Diagram Login*, *Activity Diagram* Menghapus Informasi Data Klien, *Activity Diagram* Pengiriman Informasi Data Klien, dan *Activity Diagram Logout*.

1. ***Activity Diagram Login***

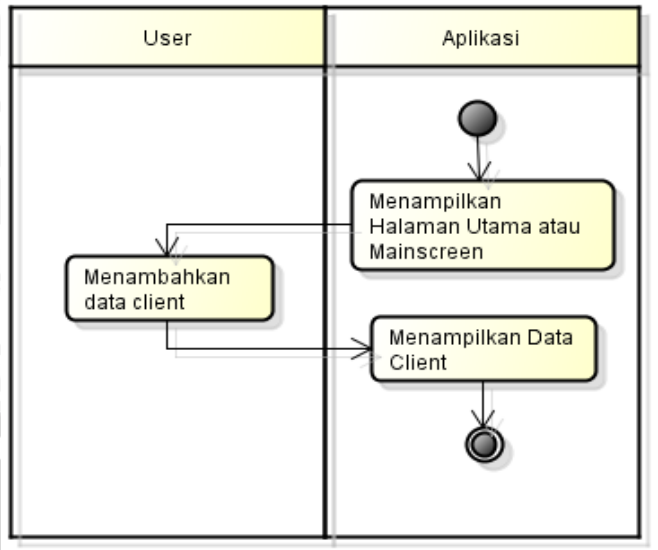
   Pada gambar 3.8 dibawah ini menunjukkan *Activity Diagram Login*, pertama kali aplikasi menampilkan halaman *login*, *user* memasukkan *username* dan *password* yang sudah terdaftar dalam *database* aplikasi *administrator*, *database* akan melakukan *result set* dan aplikasi akan memproses *result set*, jika *username* dan *password* yang dimasukkan *user* benar maka *user* akan masuk ke halaman pertama atau *mainscreen*, dan jika gagal *user* akan tetap di halaman *login*.



Gambar 3.8 *Activity Diagram Login*

2. *Activity Diagram* Menambah Informasi Data Klien

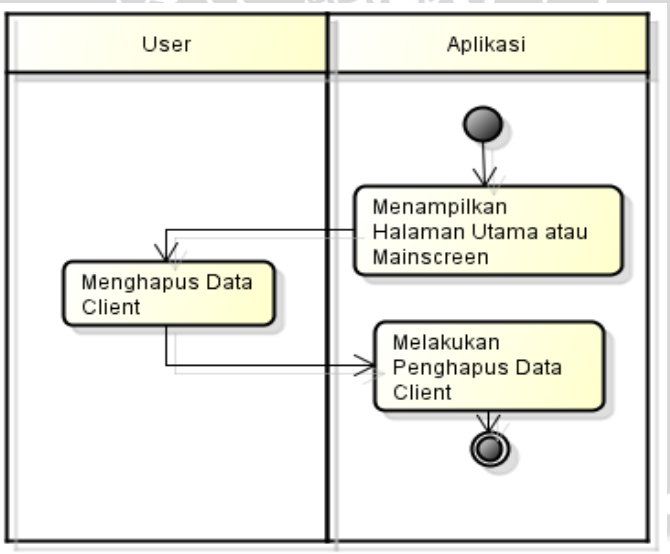
Pada gambar 3.9 dibawah ini menunjukkan *Activity Diagram* Menambah Informasi Data Klien. Pada aplikasi GPS *Tracking* berbasis Android akan menampilkan halaman utama, *user* dapat menambah semua informasi data klien. Aplikasi akan menampilkan data klien yang dibuat oleh *user*.



Gambar 3.9 *Activity Diagram* Menambah Informasi Data Klien

3. *Activity Diagram* Menghapus Informasi Data Klien

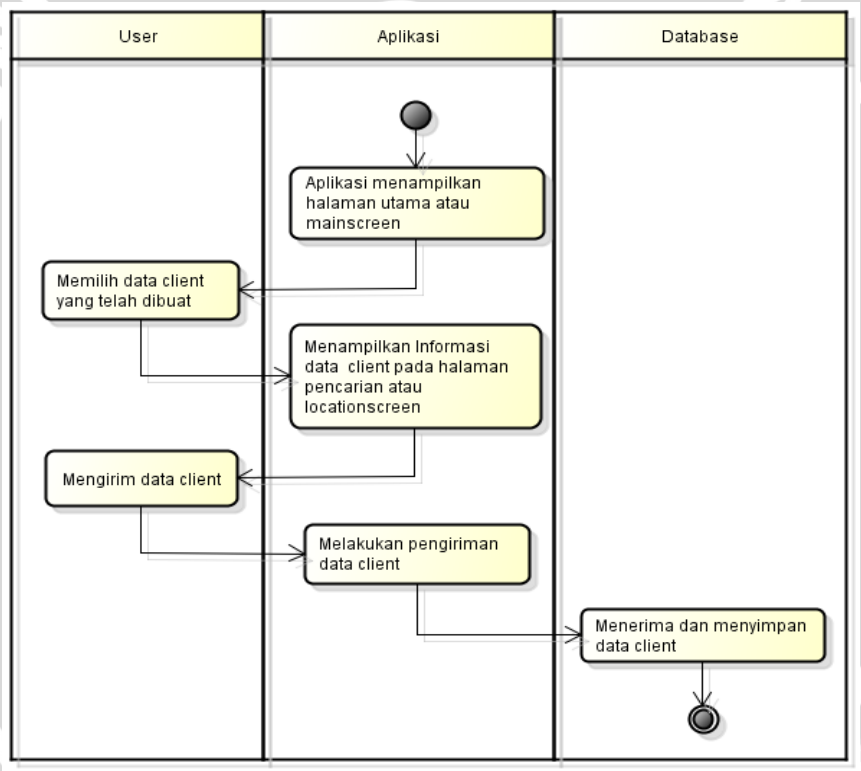
Pada gambar 3.10 dibawah ini menunjukkan *Activity Diagram* Menghapus Informasi Data Klien. Pada aplikasi GPS *Tracking* berbasis Android akan menampilkan halaman utama, *user* dapat menghapus semua informasi data klien. Aplikasi akan menghapus semua data klien yang dibuat oleh *user*.



Gambar 3.10 *Activity Diagram* Menghapus Informasi Data Klien

4. *Activity Diagram* Pengiriman Informasi Data *Klien*

Pada gambar 3.11 dibawah ini menunjukkan *Activity Diagram* Pengiriman Informasi Data Klien. Pada aplikasi GPS *Tracking* berbasis Android akan menampilkan halaman utama, *user* memilih data klien yang telah dibuat. Setelah *user* memilih data klien yang telah dibuat, aplikasi akan menampilkan data klien yang dipilih oleh *user*. *User* mengirim data klien yang telah dibuat dan aplikasi menampilkan halaman pencarian atau *locationscreen*. Aplikasi akan mengirimkan data klien yang dipilih ke aplikasi *administrator*, *database* aplikasi *administrator* akan menerima dan menyimpan data klien yang telah dikirim.

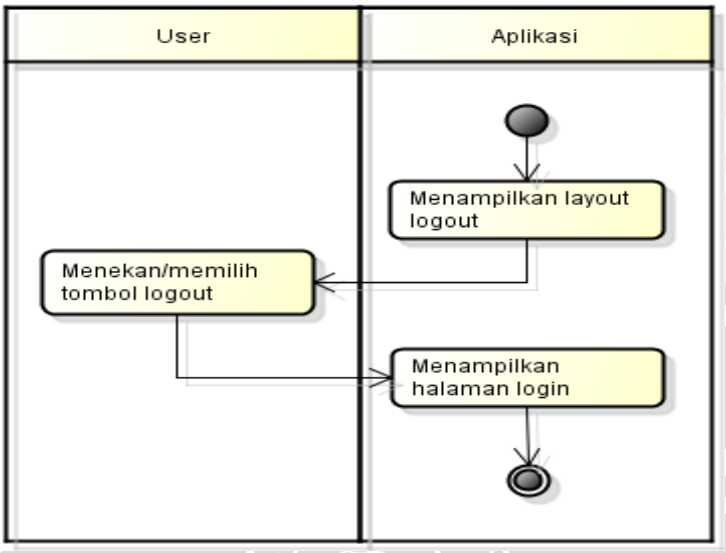


Gambar 3.11 *Activity Diagram* Pengiriman Informasi Data Klien

5. *Activity Diagram Logout*

Pada gambar 3.12 dibawah ini menunjukkan *Activity Diagram Logout.* Apilkasi GPS *Tracking* berbasis Android menyediakan *layout logout* pada semua halaman kecuali pada halaman *login*. *User* menekan *logout* pada *layout* aplikasi GPS *Tracking* berbasis Android,

dan aplikasi GPS *Tracking* berbasis Android akan kembali menampilkan halaman *login*.



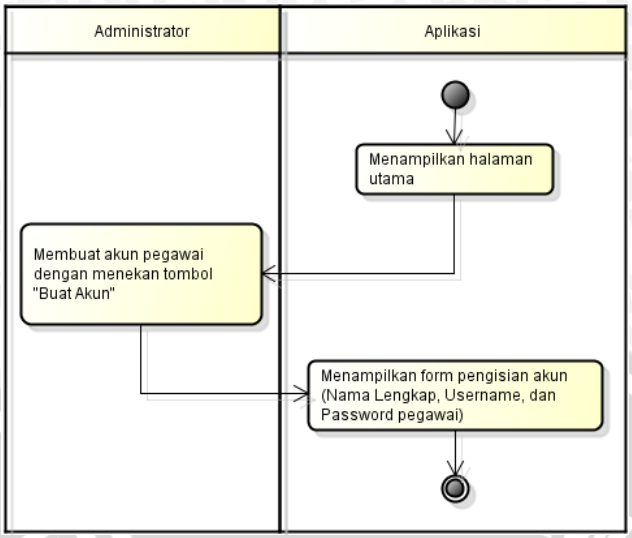
Gambar 3.12 *Activity Diagram Logout*

3.2.7.3.2 *Activity Diagram Administrator*

Pada *Activity Diagram Administrator* terdapat 5 *Activity Diagram* yaitu *Activity Diagram* Membuat Akun Pegawai, *Activity Diagram* Melihat Informasi Data Pegawai, *Activity Diagram* Melihat Informasi Data klien, *Activity Diagram* Melihat Seluruh Informasi Data Kiriman dan *Activity Diagram* Melihat Peta dan Luas Lokasi.

1. ***Activity Diagram* Membuat Akun Pegawai**

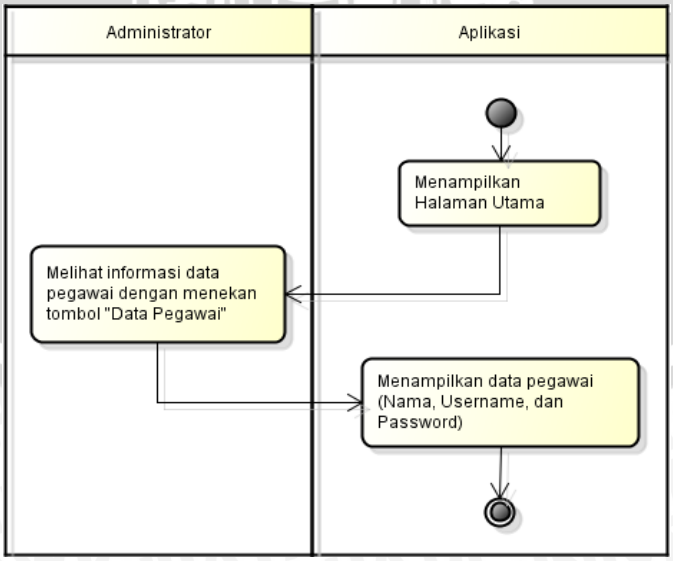
   Pada gambar 3.13 dibawah ini menunjukkan *Activity Diagram* Membuat Akun Pegawai. Apliaksi *administrator* menampilkan halaman utama yaitu *webservice* kantor Badan Pertanahan Nasional . *Administrator* melakukan pembuatan akun untuk para pegawai atau *user* dengan menekan tombol “Buat Akun”. Aplikasi *administrator* akan menampilkan halaman pembuatan akun yang berisi form (Nama Lengkap, *Username*, dan *Password*).



Gambar 3.13 *Activity Diagram* Membuat Akun Pegawai

2. *Activity Diagram* Melihat Informasi Data Pegawai

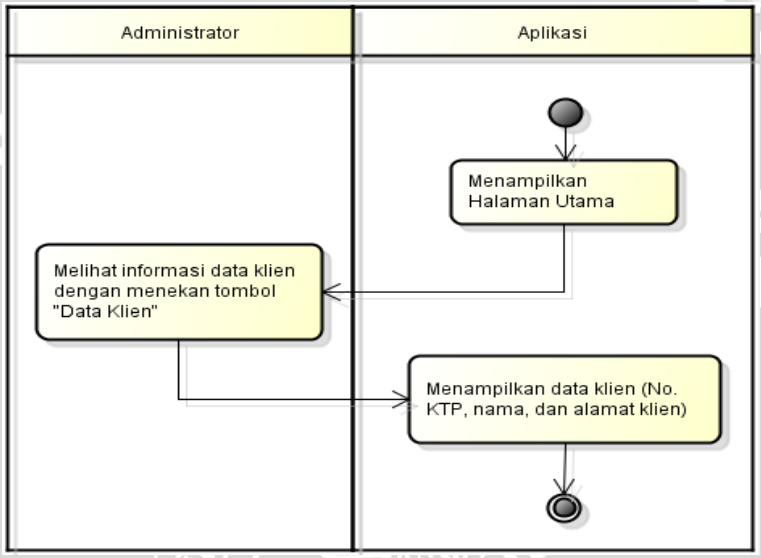
Pada gambar 3.14 dibawah ini menunjukkan *Activity Diagram* Melihat Informasi Data Pegawai. Apliaksi *administrator* menampilkan halaman utama yaitu *webservice* kantor Badan Pertanahan Nasional . *Administrator* melihat informasi data pegawai dengan menekan tombol "Data Pegawai". Aplikasi *administrator* akan menampilkan seluruh informasi data dari data yang dipilih (*username*, nama, dan *password* pegawai).



Gambar 3.14 *Activity Diagram* Melihat Informasi Data Pegawai

3. *Activity Diagram* Melihat Informasi Data Klien

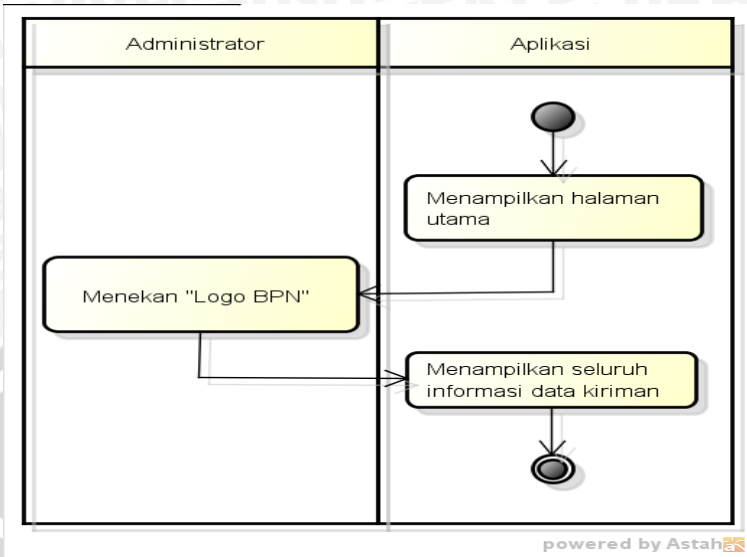
Pada gambar 3.15 dibawah ini menunjukkan *Activity Diagram* Melihat Informasi Data klien. Apliaksi *administrator* menampilkan halaman utama yaitu *webservice* kantor Badan Pertanahan Nasional . *Administrator* melihat informasi data klien dengan menekan tombol "Data Klien". Aplikasi *administrator* akan menampilkan informasi data dari data yang dipilih (No. KTP, nama klien, dan alamat klien).



Gambar 3.15 *Activity Diagram* Melihat Informasi Data Klien

4. *Activity Diagram* Melihat Seluruh Informasi Data Kiriman

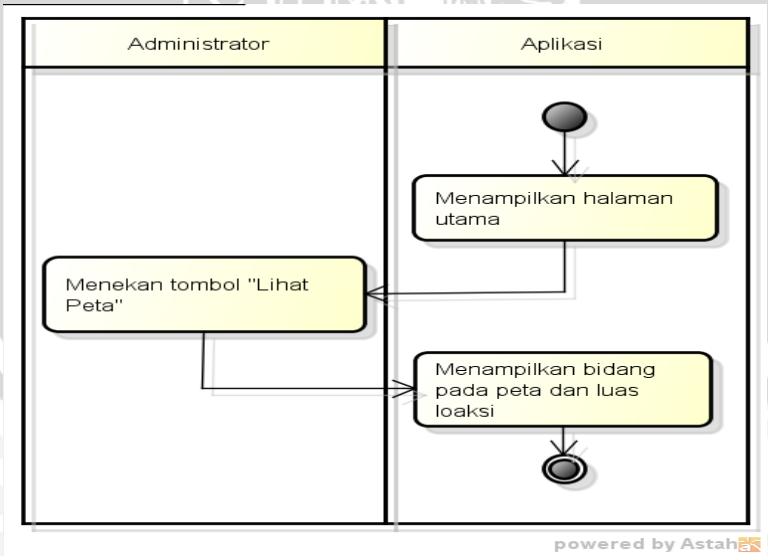
Pada gambar 3.16 dibawah ini menunjukkan *Activity Diagram* Melihat Seluruh Informasi Data Kiriman. Apliaksi *administrator* pertama kali akan menampilkan halaman utama yaitu *webservice* kantor Badan Pertanahan Nasional . *Administrator* menekan "Logo BPN", dan aplikasi *administrator* akan menampilkan seluruh informasi data kiriman.



Gambar 3.16 *Activity Diagram* Melihat Seluruh Informasi Data Kiriman

5. *Activity Diagram* Melihat Peta dan Luas Lokasi

Pada gambar 3.17 dibawah ini menunjukkan *Activity Diagram* Melihat Peta dan Luas Lokasi. Apliaksi *administrator* pertama kali akan menampilkan halaman utama yaitu *webservice* kantor Badan Pertanahan Nasional . *Administrator* menekan tombol “Lihat Peta”, dan aplikasi *administrator* akan menampilkan sebuah bidang pada Google Maps sekaligus luas loaksi.



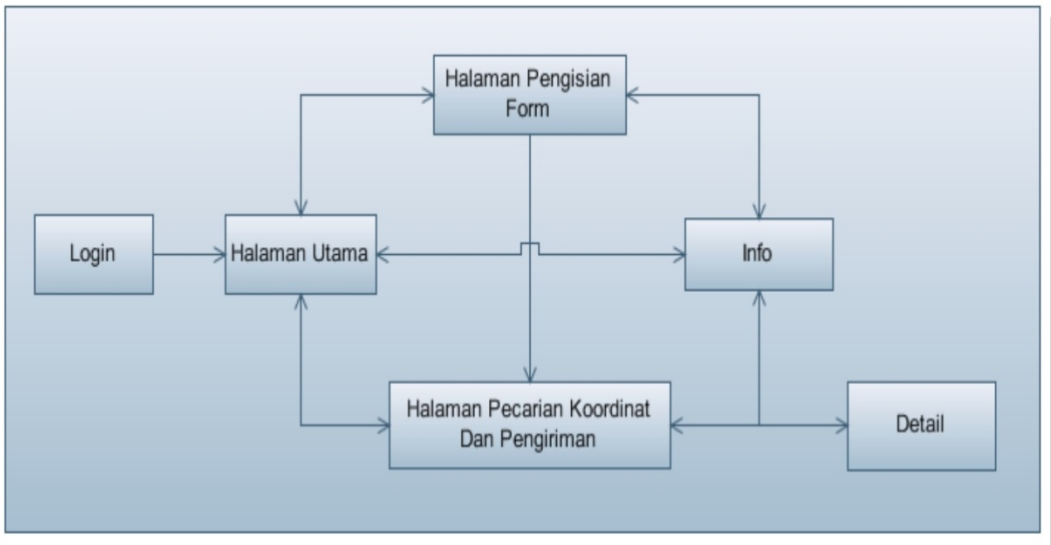
Gambar 3.17 *Activity Diagram* Melihat Peta dan Luas Lokasi

3.2.7.4 Perancangan Antarmuka

Pada bagian ini akan dijelaskan tentang perancangan antarmuka perangkat lunak GPS *Tracker* Berbasis Android. Antarmuka perangkat lunak ini akan digunakan oleh pengguna untuk berinteraksi dengan sistem perangkat lunak GPS *Tracker* Berbasis Android. Antarmuka perangkat lunak ini dibagi menjadi dua, yaitu antarmuka untuk aplikasi *user* dan aplikasi *administrator*.

3.2.7.4.1 Perancangan Antarmuka Aplikasi *User*

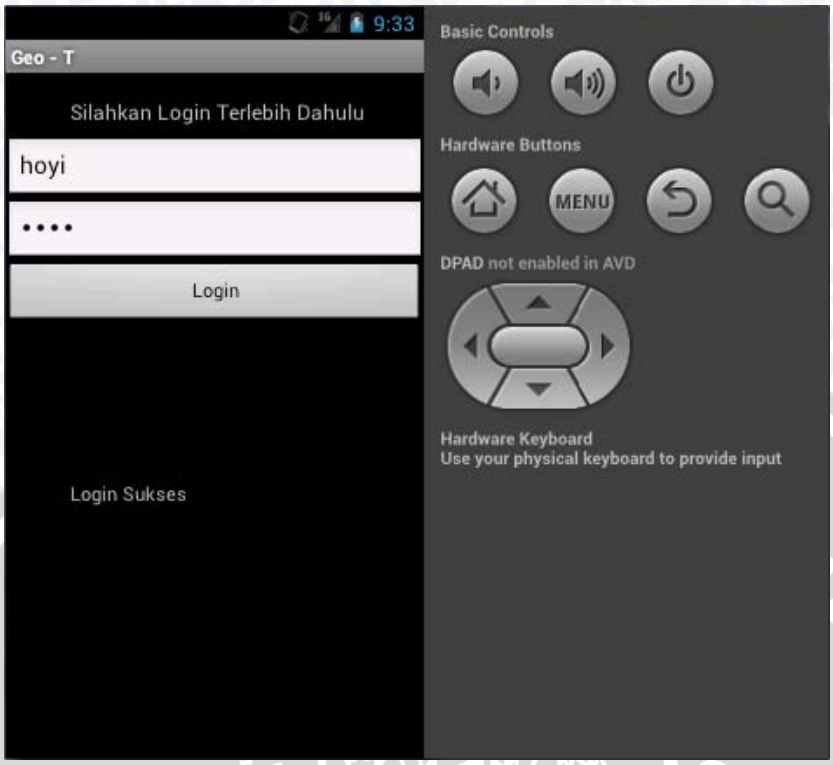
Antarmuka pada aplikasi pengguna berupa layout Android yang memudahkan pengguna untuk menggunakan aplikasi GPS *Tracker* Berbasis Android.



Gambar 3.18 *Sitemap* Antarmuka Aplikasi *User*

1. Halaman Login

Halaman login merupakan salah satu antarmuka pengguna untuk aplikasi *user* yang berfungsi untuk *user* dalam melakukan *login*. Gambar 3.19 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman *login*.



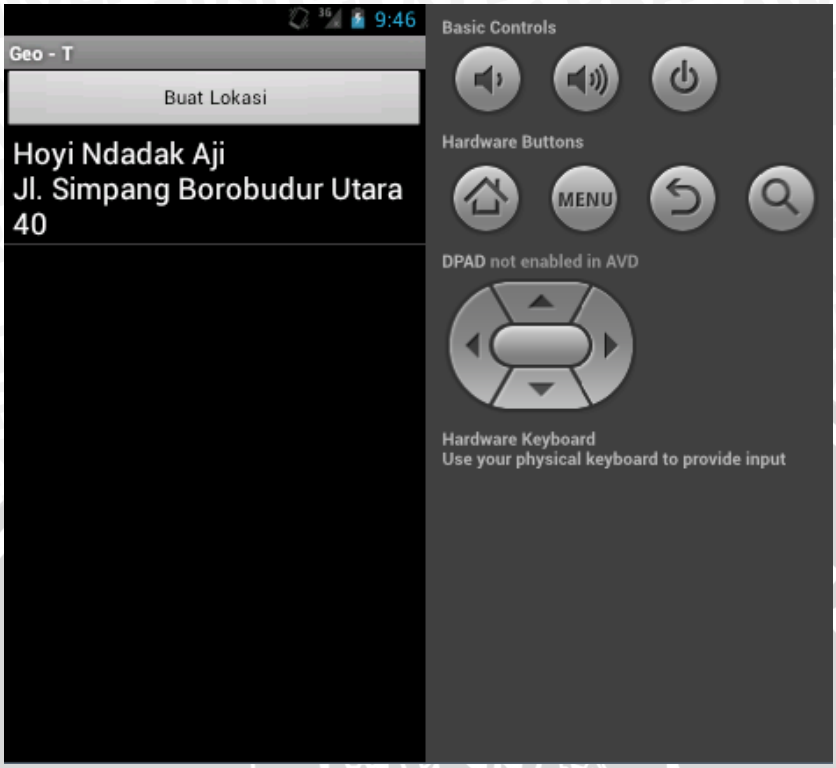
Gambar 3.19 Halaman *Login* Pada Aplikasi *User*

Keterangan gambar 3.19:

1. Form untuk mengisi *username*.
2. Form untuk mengisi *password.*
3. Tombol *Login* untuk melakukan proses *login*.
4. Pesan “sukses” jika berhasil melakukan *login* dan pesan “gagal” jika tidak berhasil melakukan *login*.

2. Halaman Utama

Halaman utama adalah halaman untuk menampilkan data – data klien yang telah dicatat sebelumnya. Gambar 3.20 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman utama.



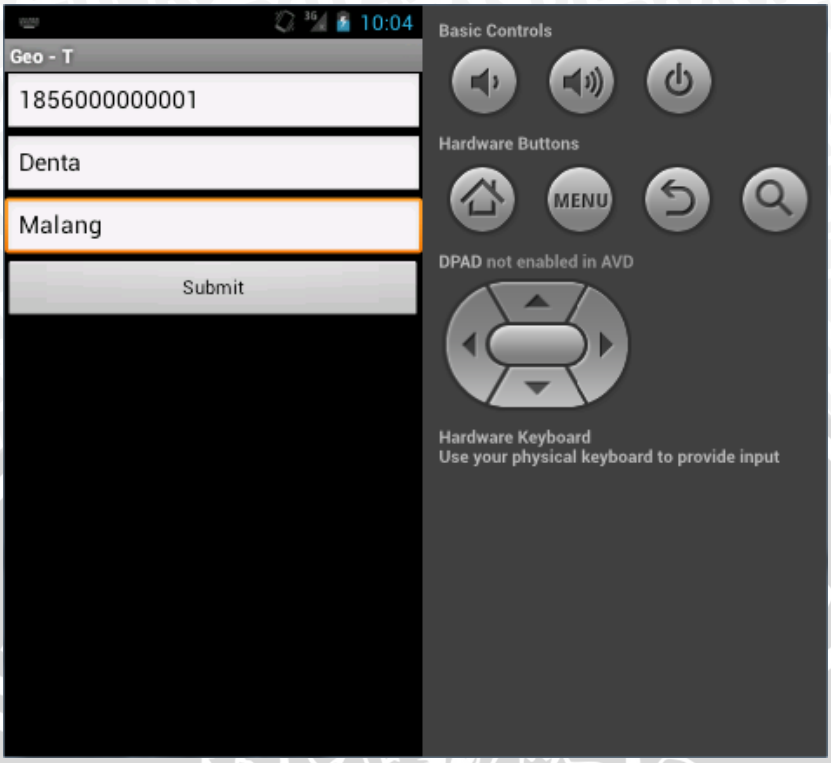
Gambar 3.20 Halaman Utama Pada Aplikasi *User*

Keterangan gambar 3.20:

1. Tombol "Buat Lokasi" untuk melakukan proses pencatatan data klien pada halaman pengisian *form*.
2. *Listview* yang berisi nama dan alamat adalah listview untuk melihat koordinat klien pada halaman pencarian koordinat dan pengiriman.

3. Halaman Pengisian *Form*

Halaman pengisian *form* adalah halaman untuk mencatat data – data klien. Gambar 3.21 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman pengisian *form.*



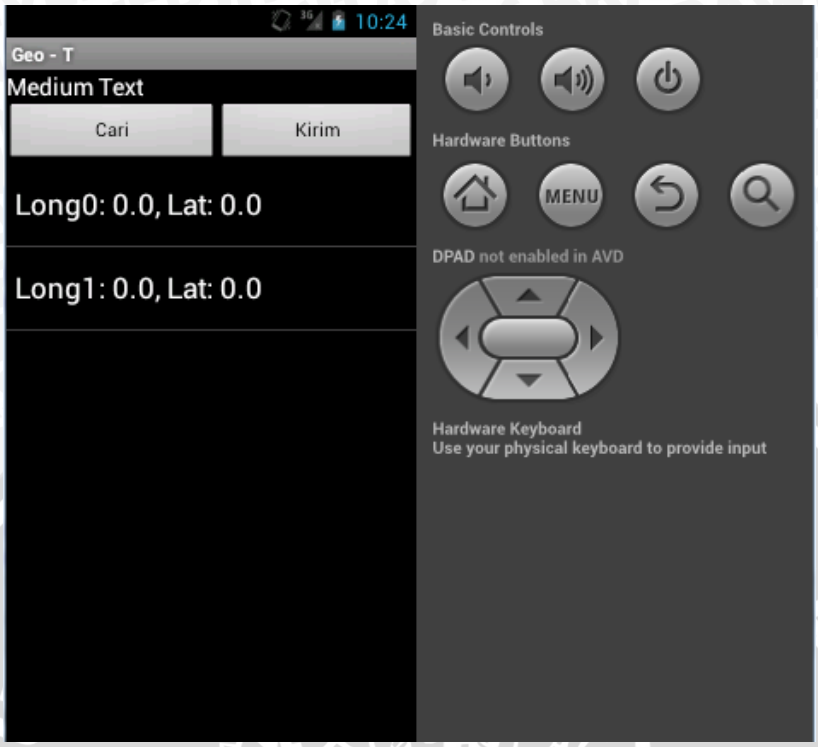
Gambar 3.21 Halaman Pengisian *form* Pada Aplikasi *User*

Keterangan gambar 3.21:

1. *Form* untuk mengisi No. KTP klien.
2. *Form* untuk mengisi nama klien.
3. *Form* untuk mengisi alamat klien.
4. Tombol "Submit" untuk melakukan proses pencatatan dan penyimpanan data klien.

4. Halaman Pencarian Koordinat dan Pengiriman

Halaman pencarian koordinat dan pengiriman adalah halaman untuk mencari koordinat lahan klien dan untuk melakukan pengiriman semua data klien yang telah tercatat. Gambar 3.22 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman pencarian koordinat dan pengiriman.



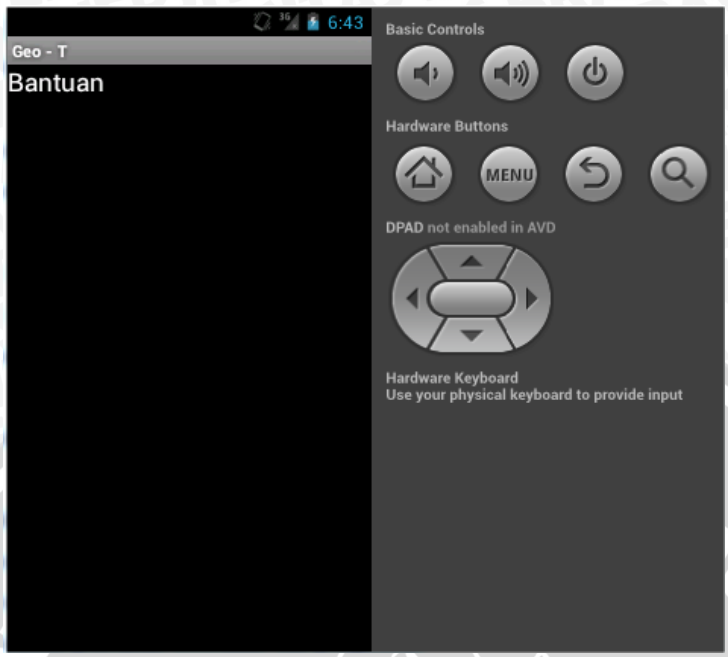
Gambar 3.22 Halaman pencarian koordinat dan pengiriman

Keterangan gambar 3.22:

1. Tombol “Cari” untuk melakukan proses pencarian koordinat pada lahan klien.
2. Tombol “Kirim” untuk melakukan proses pengiriman semua data klien ke server atau aplikasi *administrator*.

5. Halaman Bantuan

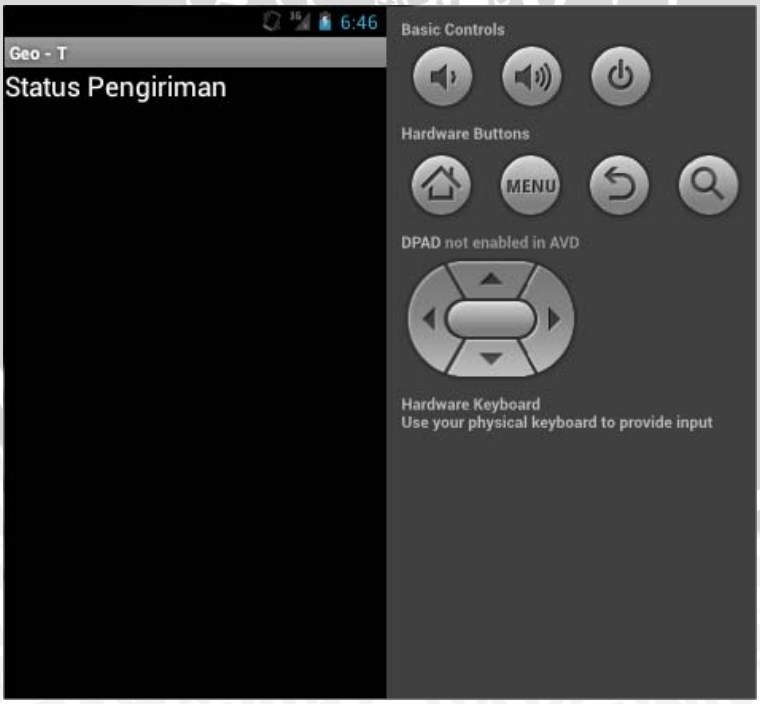
Halaman bantuan adalah halaman untuk mengetahui informasi bantuan, tata cara menggunakan aplikasi GPS *Tracker* berbasis Android secara keselurahan. Gambar 3.23 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman bantuan.



Gambar 3.23 Halaman Bantuan

6. Halaman Detail

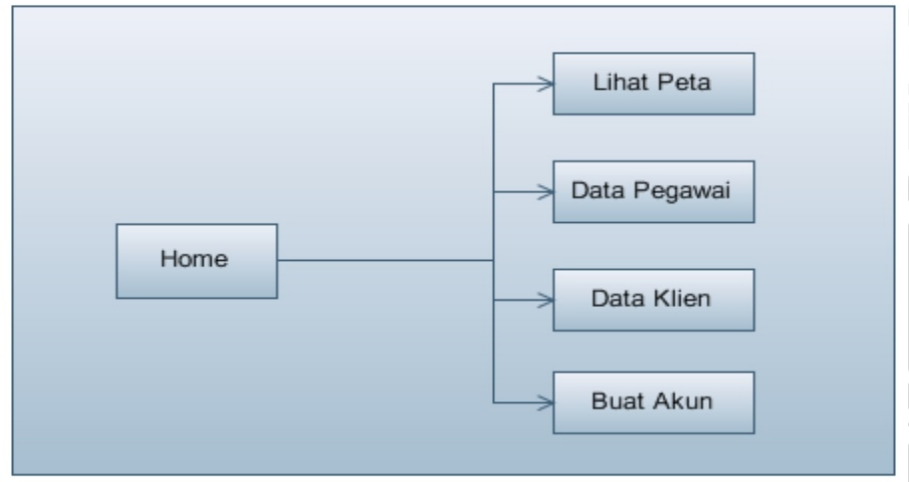
Halaman detail adalah halaman untuk mengetahui status pesan dari data klien yang telah terkirim atau belum terkirim. Gambar 3.24 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman info.



Gambar 3.24 Halaman Detail

3.2.7.4.2 Perancangan Antarmuka Aplikasi *Administrator*

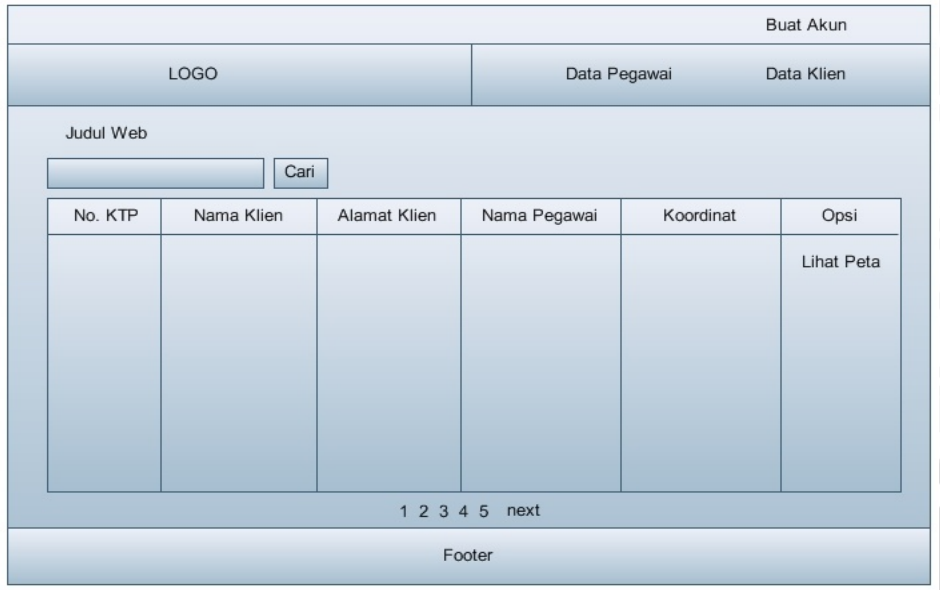
Antarmuka pada sisi *admisnitrator* berupa halaman *web*. Perancangan antarmuka pada sisi *administrator* bertujuan untuk memudahkan *admin* untuk membuat akun pegawai, melihat, mencari, dan menghapus data klien di *server*. Gambar 3.25 menunjukkan *sitemap* dari aplikasi *administrator*.



Gambar 3.25 *Sitemap* Antarmuka Aplikasi *Administrator*

1. Halaman Home

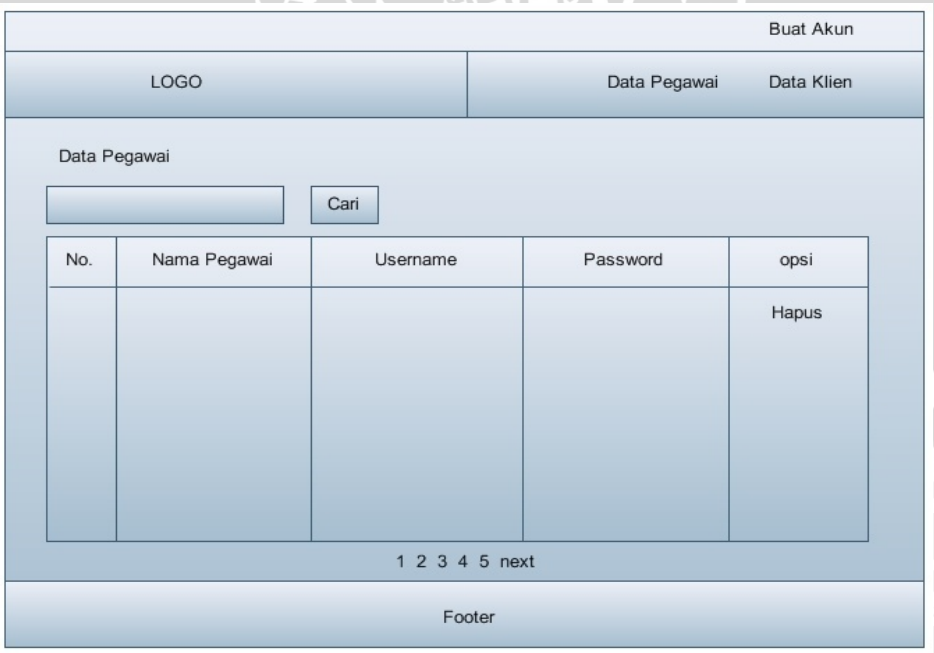
Halaman home merupakan halaman utama yang menampilkan data-data yang disimpan di basis data. Informasi-informasi tentang seluruh data klien dari kiriman aplikasi *user* dapat dilihat di halaman ini. Halaman ini berisi tabel yang menampung informasi tentang data-data klien. Halaman ini juga berisi link untuk melihat data klien (No. KTP, nama klien, dan alamat klien), melihat data pegawai (*username*, *password*, dan nama pegawai), dan melihat peta. Gambar 3.26 menunjukkan rancangan antarmuka halaman home.



Gambar 3.26 Rancangan Antarmuka Halaman Home

2. Halaman Data Pegawai

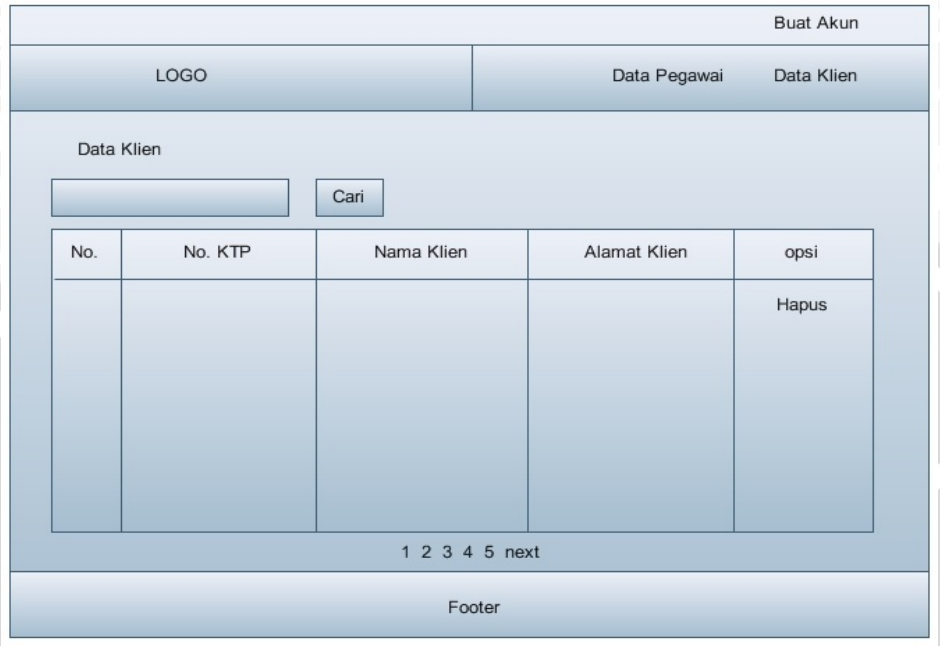
Halaman data pegawai adalah halaman yang menampilkan data-data pegawai yang disimpan di dalam basis data. Halaman ini juga berisi link untuk menon-aktifkan data pegawai. Gambar 3.27 menunjukkan rancangan antarmuka halaman data pegawai.



Gambar 3.27 Rancangan Antarmuka Halaman Data Pegawai

3. Halaman Data Klien

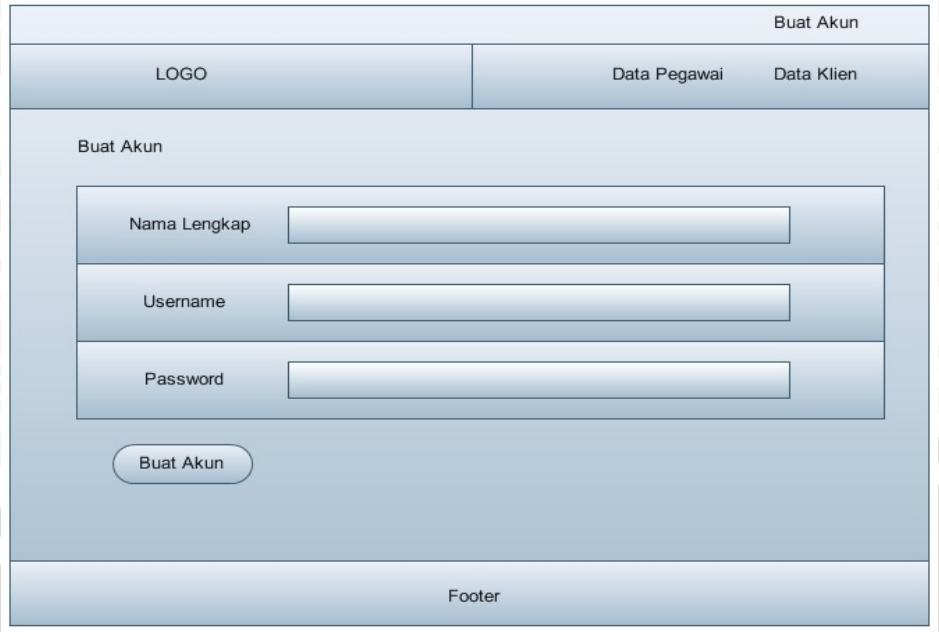
Halaman data klien adalah halaman yang menampilkan data-data klien yang disimpan di dalam basis data. Halaman ini juga berisi link untuk menghapus data klien. Gambar 3.28 menunjukkan rancangan antarmuka halaman data klien.



Gambar 3.28 Rancangan Antarmuka Halaman Data Klien

4. Halaman Buat Akun

Halaman buat akun adalah halaman yang menampilkan pembuatan akun untuk pegawai atau *user* agar dapat *login* dalam aplikasi *user*. Gambar 3.29 menunjukkan rancangan antarmuka halaman buat akun.



Gambar 3.29 Rancangan Antarmuka Halaman Buat Akun

Keterangan Gambar 3.29:

1. Form untuk mengisi nama lengkap pegawai.
2. Form untuk mengisi nama *username* pegawai.
3. Form untuk mengisi nama *password* pegawai.
4. Tombol “Buat Akun” untuk melakukan proses pembuatan akun pegawai dan menyimpan ke dalam database pegawai.

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan terdiri atas penjelasan tentang spesifikasi sistem, implementasi basis data, implementasi tiap *class* pada *file* program, implementasi algoritma komponen-komponen yang digunakan, dan implementasi antarmuka.

4.1. Spesifikasi Sistem

Perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

4.1.1. Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan pada Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat keras komputer

Notebook Acer Aspire 4920	
Processor	Intel (R) Core (TM) 2 Duo Processor T5550
Memory (RAM)	1 Gb DDR2
Harddisk	160 Gb
Motherboard	Acer Notebook Intel Motherboard
Graphic Card	Mobile Intel (R) Graphics Media Accelerator X3100
Monitor	14.1” WXGA Acer CrystalBrite (TM) LCD

4.1.2. Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan dijelaskan pada Tabel 4.2.

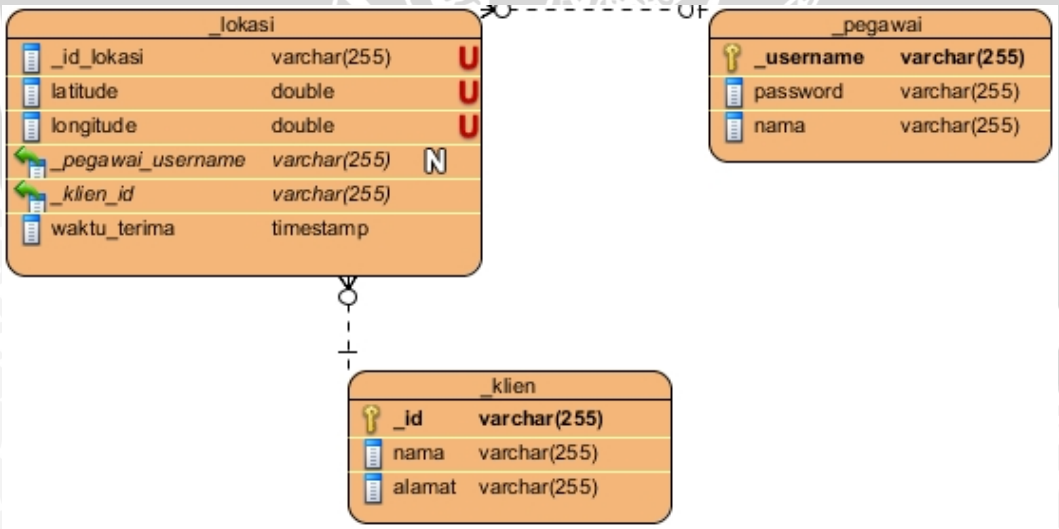
Tabel 4.2 Spesifikasi perangkat lunak komputer

Notebook Acer Aspire 4920	
Operating System	Microsoft Windows 7 Home Premium Acer Enhanced Experience Edition 64-bit
DirectX Version	DirectX 11
Programming Language	Java
Software Development Kit	Java Development Kit 7 Update 21
Programming Environment	Java Runtime Environment 7
Database Management System	MySQL 5.1
Integrated Development Environment	Eclipse, Emulator Android, dan Dreamweaver

4.2. Implementasi Basis Data

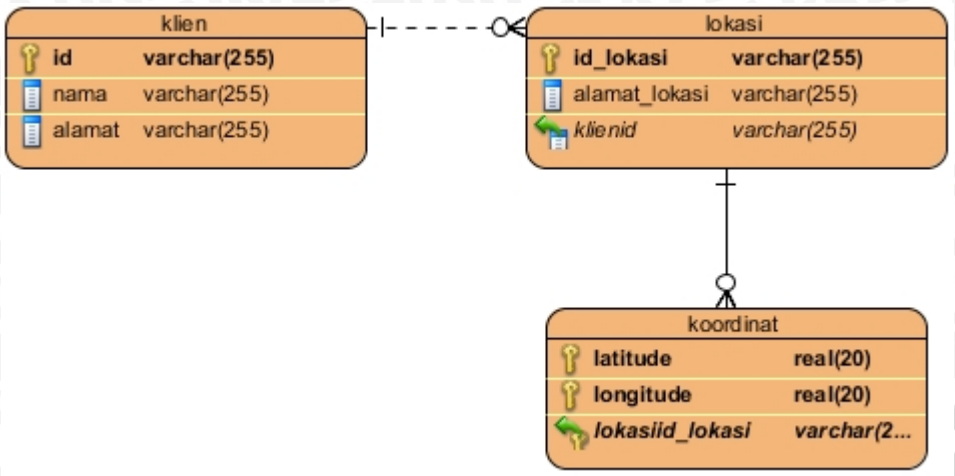
Implementasi penyimpanan data dilakukan dengan *database management system* MySQL pada aplikasi *administrator* dan *database management system* SQLite pada aplikasi *user*. Hasil implementasi penyimpanan data ini berupa *script – script* SQL. Hasil implementasi SQL pada *database* ini dimodelkan dalam diagram konseptual *entity relationship.*

1. *Entity relationship* Aplikasi *Administrator*



Gambar 4.1 Diagram *Entity relationship* Aplikasi *Administrator*

2. *Entity relationship* Aplikasi *User*



Gambar 4.2 Diagram *Entity relationship* Aplikasi *User*

4.3. Implementasi *Class* dan *Interface* Pada *File* Program

Setiap *class* yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program dengan ekstensi *.java. Tabel 4.3 menjelaskan mengenai pasangan antara *class* dengan *file* program GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan yang digunakan untuk mengimplementasikannya

Tabel 4.3 Implementasi *class* pada kode program *.java

No.	*Package*	Nama *Class*	Nama *File* Program
1	bpn_pati.*activity*	Bantuan	Bantuan.java
2	bpn_pati.*activity*	Beranda	Beranda.java
3	bpn_pati.*activity*	CariKoordinat	CariKoordinat.java
4	bpn_pati.*activity*	CariLokasi	CariLokasi.java
5	bpn_pati.*activity*	Detail	Detail.java
6	bpn_pati.*activity*	*FormClient*	*FormClient*.java
7	bpn_pati.*activity*	FormLokasi	FormLokasi.java
8	bpn_pati.*activity*	*Login*	*Login*.java
9	bpn_pati.*adapter*	Klien_adapter	Klien_adapter.java
10	bpn_pati.*adapter*	Klien_objek	Klien_objek.java
11	bpn_pati.*adapter*	Lokasi_adapter	Lokasi_adapter.java

12	bpn_pati.*adapter*	Lokasi_objek	Lokasi_objek.java
13	bpn_pati.*helper*	DBMS	DBMS.java
14	bpn_pati.*helper*	GpsService	GpsService.java
15	bpn_pati.*helper*	UserData	UserData.java
16	bpn_pati.koneksi	Httpclient	Httpclient.java
17	bpn_pati.model	Klien	Klien.java
18	bpn_pati.model	Lokasi	Lokasi.java

4.4. Implemementasi Algoritma Komponen-Komponen Perangkat Lunak

Perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan memiliki beberapa proses atau *method* yang terdapat pada beberapa *class*. Beberapa *method* yang akan dicantumkan dalam penulisan makalah skripsi ini hanya untuk algoritma dari beberapa proses (operasi) saja sehingga tidak semua *method* akan dicantumkan.

Semua *class* yang terdapat pada perangkat lunak GPS *Tracker* berbasis Android ini menggunakan komponen-komponen yang sudah ada atau menggunakan metode CBSE *(Component Based Software Engineering)*. Berikut ini adalah beberapa komponen yang digunakan dalam perangkat lunak GPS *Tracker* berbasis Android.

4.4.1. Implemementasi Algoritma dari Komponen *Class* Beranda

Komponen *class* beranda ini digunakan untuk menampilkan halaman utama dalam perangkat lunak GPS *Tracker* berbsis Android. Di dalam *class* ini terdapat 4 method diantaranya:

1. Method *onCreate* adalah method yang dijalankan pertama kali pada *class* beranda yang digunakan untuk menginisialisasi antarmuka dan komponen-komponennya.
2. Method *onClick* digunakan untuk menangani setiap *even* yang dilakukan oleh *user*.
3. Method *onItemClick* digunakan untuk menangani setiap *even* yang dilakukan oleh *user* dikhususkan pada komponen antarmuka *listview*.

4. Method *updateListKlien_cb* digunakan untuk mengupdate data dari tampilan *listview* yang diambil dari *table* klien di DBMS.

Implementasi algoritma *class* beranda terdapat pada Gambar 4.3.

```
public class Beranda extends Activity implements OnClickListener,
OnItemClickListener{

	Button b_create_client;
	ListView lv_klien;
	String onUser;
	ArrayList<String> Klien_list;
	ArrayAdapter<String> Klien_adapter;
	Klien klien_model;
	ArrayList<Klien_objek> klienListCheckBox;
	Klien_adapter klienAdapterCheckBox;
	String[] jmlKlien;
	boolean isNothingCB = true;

	SharedPreferences prefs;
	Editor editor;

	DBMS db;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.beranda);

   prefs = getSharedPreferences("myData", MODE_PRIVATE);
   onUser = prefs.getString("username", null);

   b_create_client (Button) findViewById(R.id.b_create_new_client);
   lv_klien = (ListView) findViewById(R.id.lv_klien);
   b_create_client.setOnClickListener(this);
   lv_klien.setOnItemClickListener(this);
   db = DBMS.getInstance(this);
   this.updateListKlien_cb();
}
@Override
public void onClick(View v) {
	switch(v.getId()){
	case R.id.b_create_new_client:
		try {
			Intent pr = new Intent(this, FormClient.class);
				startActivity(pr);
		}
		catch (Exception e) {
		  Log.e("ERROR", e.getMessage());
		  return;
		}
		break;
		 }
	}
```

```
@Override
public void onItemClick(AdapterView<?> l, View v, int position, long
id) {
       Klien_objek klien = (Klien_objek)
    klienAdapterCheckBox.getItem(position);
       Toast.makeText(getApplicationContext(),
       "item Click: "+klien.getId(), Toast.LENGTH_LONG).show();

       Intent myIntent = new Intent(this, CariLokasi.class);
       myIntent.putExtra("id_klien", klien.getId());
       startActivityForResult(myIntent,1);
       }

public void updateListKlien_cb(){
       klienListCheckBox = db.getAllKlien_list();
       jmlKlien = new String[klienListCheckBox.size()];
    klienAdapterCheckBox = new Klien_adapter(this, jmlKlien,
    klienListCheckBox);
    lv_klien.setAdapter(klienAdapterCheckBox);
       }
```

Gambar 4.3 Implementasi Algoritma *Class* Beranda

4.4.2. Implemementasi Algoritma dari Komponen *Class* CariKoordinat

Komponen *class* cariKoordinat ini digunakan untuk menadapatkan koordinat *longitude* dan *latitude* menggunakan fitur GPS pada Android. Di dalam *class* ini terdapat 3 method diantaranya:

1. Method *onCreate* adalah method yang dijalankan pertama kali pada *class* cariKoordinat yang digunakan untuk menginisialisasi antarmuka dan komponen-komponennya.
2. Method *onClick* digunakan untuk menangani setiap *even* yang dilakukan oleh *user*.
3. Method *updateListKoordinat* digunakan untuk mengupdate nilai longitude dan latitude pada *listview* yang diambil dari fitur GPS Android.

Implementasi algoritma *class* cariKoordinat terdapat pada Gambar 4.4.

```
public class CariKoordinat extends Activity implements
OnClickListener, OnItemClickListener{

       GpsService gps;
       Double longitude, latitude;
       Httpclient client;
       Klien klien_model;
       Lokasi lokasi_model;
```

```
        Button b_cari,b_kirim;
        ListView lv_koordinat;
        SharedPreferences prefs;
        Editor editor;

        String id_lokasi, id_klien, username, klien, alamat, onUser;
        Intent intent;

        ArrayList<String> koordinat_list;
        ArrayAdapter<String> koordinat_adapter;

        DBMS db;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.carikoordinat);

    prefs = getSharedPreferences("myData", MODE_PRIVATE);
    onUser = prefs.getString("username", null);
    client = new Httpclient();
    intent = getIntent();
            id_lokasi = intent.getStringExtra("id_lokasi");
            id_klien = intent.getStringExtra("id_klien");
            username = intent.getStringExtra("username");
            String delims = "[_]";
            klien = id_lokasi.split(delims)[0];
            alamat = id_lokasi.split(delims)[1];

    b_cari = (Button) findViewById(R.id.b_cari_koordinat);
    b_kirim = (Button) findViewById(R.id.b_kirim_koordinat);
    lv_koordinat = (ListView) findViewById(R.id.lv_koordinat);
    b_cari.setOnClickListener(this);
    b_kirim.setOnClickListener(this);
    lv_koordinat.setOnItemClickListener(this);

    db = DBMS.getInstance(this);
    this.updateListKoordinat();
  }

@Override
public void onClick(View v) {
            switch(v.getId()){
            case R.id.b_cari_koordinat:
                  try {
                        gps = new GpsService(this);
                        if (gps.canGetLocation())
                        {
                              latitude = gps.getLatitude();
                              longitude = gps.getLongitude();
                              db.insertKoordinat(id_lokasi,
longitude, latitude);

                              updateListKoordinat();
```

```
      Toast.makeText(getApplicationContext(),
                                                "Lokasi mu latitude: " +
latitude + " Longitude : " + longitude +" kembalian ",
Toast.LENGTH_LONG).show();

                        } else
                        {
                              gps.showSettingAlert();
                        }
            }
            catch (Exception e) {
              Log.e("ERROR", e.getMessage());
              return;
            }
            break;
               case R.id.b_kirim_koordinat:
                     try {
                        Cursor cursor2 =
db.getKoordinat_id(id_lokasi);
                        client.addParam("id_lokasi", id_lokasi);
                        client.addParam("username", onUser);
                        client.addParam("id_klien", id_klien);
                        client.addParam("jml",
String.valueOf(cursor2.getCount()));
                        for(int i=0; i<cursor2.getCount(); i++){
                              cursor2.moveToNext();
                              client.addParam("longitude"+i,
cursor2.getString(1));
                              client.addParam("latitude"+i,
cursor2.getString(2));
                        }

                        client.setUrl("terimadata.php");
                        client.executeRequest();


                        Toast.makeText(getApplicationContext(),
                                    "respon:
"+client.getResponse(), Toast.LENGTH_LONG).show();
            }
            catch (Exception e) {
              Log.e("ERROR", e.getMessage());
              return;
            }
            break;

               }
      }
private void updateListKoordinat() {
            koordinat_list = new ArrayList<String>();
    koordinat_adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, db.getKoordinatById(id_lokasi));
    lv_koordinat.setAdapter(koordinat_adapter);
      }
```

Gambar 4.4 Implementasi Algoritma *Class* CariKoordinat

4.4.3. Implementasi Algoritma dari Komponen *Class* FormClient

Komponen *class form Client* ini digunakan untuk menampilkan beberapa *form* untuk registrasi klien baru. Di dalam *class* ini terdapat 3 method diantaranya:

1. Method *onCreate* adalah method yang dijalankan pertama kali pada *class* cariKoordinat yang digunakan untuk menginisialisasi antarmuka dan komponen-komponennya.
2. Method *onClick* digunakan untuk menangani setiap *even* yang dilakukan oleh *user*.

Implementasi algoritma *class formClient* terdapat pada Gambar 4.5.

```
public class FormClient extends Activity implements OnClickListener{

       EditText et_ktp, et_nama, et_alamat;
       Button b_submit;
       Intent myIntent;
       Klien klien_model;
       Httpclient client;
       Beranda beranda;
       Lokasi lokasi_model;

       DBMS db;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.formclient);

    et_ktp = (EditText) findViewById(R.id.et_no_ktp);
             et_nama = (EditText) findViewById(R.id.et_nama_klien);

             et_alamat = (EditText)
findViewById(R.id.et_alamat_klient);
             b_submit = (Button) findViewById(R.id.b_klien_kirim);
             b_submit.setOnClickListener(this);
             client = new Httpclient();
             db = DBMS.getInstance(this);
  }

@Override
public void onClick(View v) {
             switch(v.getId()){
             case R.id.b_klien_kirim:
                    try {
                           db.insertKlien(et_ktp.getText().toString(),
                                  et_nama.getText().toString(),
                                  et_alamat.getText().toString()
                           );
```

```
                        myIntent = new Intent(this,
CariLokasi.class);
                        myIntent.putExtra("id_lokasi",
et_nama.getText().toString()+"_"+et_alamat.getText().toString());
                        myIntent.putExtra("id_klien",
et_ktp.getText().toString());
                        this.startActivity(myIntent);
                        client.addParam("id",
et_ktp.getText().toString());
                        client.addParam("nama",
et_nama.getText().toString());
                        client.addParam("alamat",
et_alamat.getText().toString());
                        client.setUrl("postKlien.php");
                        client.executeRequest();
                        Toast.makeText(this, "respon:
"+client.getResponse(), Toast.LENGTH_LONG).show();
                        finish();
                }
                catch (Exception e) {
                  Log.e("ERROR", e.getMessage());
                  return;
                }
                break;
                   }

        }
```

Gambar 4.5 Implementasi Algoritma *Class* FormClient

4.4.4. Implemementasi Algoritma dari Komponen *Class* GPS*Service*

Komponen *class* GpsService adalah sebagai *library* GPS pada Android. Di dalam *class* ini terdapat 11 method diantaranya:

1. Method Gps*Service* adalah method konstruktor yang dijalankan pertama kali pada *class* GPS*Service*..
2. Method *Location getLocation*
3. Method *getLatitude* digunakan untuk mendapatkan nilai *latitude*.
4. Method *setLatitude* digunakan untuk memberikan nilai *latitude*.
5. Method *getLongitude* digunakan untuk mendapatkan nilai *longitude*.
6. Method *setLongitude* digunakan untuk memberikan nilai *longitude*.
7. Method *canGetLocation* digunakan untuk mendapakan nilai *canGetLocation* yang bertipe *boolean*.

Implementasi algoritma *class* GpsService terdapat pada Gambar 4.6.

```
public class GpsService extends Service implements LocationListener
{
```

```
	private final Context			_context;

	// cek apakah GPS aktif ?
	boolean 					isGPSEnable
			= false;
	// cek network aktif ?
	boolean 					isNetworkEnable
			= false;

	boolean 					canGetLocation
			= false;

	Location 				location;
	double 					latitude;
	double 					longitude;

	// GPS akan update ketika jarak sudah berubah lebih dari 10
meter
	private static final floatMIN_JARAK_GPS_UPDATE	= 0.1f;
				// meter

	// GPS akan update pada waktu interval
	private static final long MIN_WAKTU_GPS_UPDATE	= 1000;

	protected LocationManager locManager;

public GpsService(Context context)
	{
		_context = context;
		getLocation();
	}

private Location getLocation()
	{

		try
		{
			locManager = (LocationManager)
_context.getSystemService(LOCATION_SERVICE);

			// cek GPS status
			isGPSEnable =
locManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
			// cek status koneksi
			isNetworkEnable =
locManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

			if (!isGPSEnable && !isNetworkEnable)
			{
				// tidak ada koneksi ke GPS dan Jaringan
			} else
			{
				// bisa dapatkan lokasi
				canGetLocation = true;
```

```
                        // cek apakah koneksi internet bisa ?
                        if (isNetworkEnable)
                        {
                                // ambil posisi berdasarkan Network

      locManager.requestLocationUpdates(LocationManager.NETWORK_PROVID
ER, MIN_WAKTU_GPS_UPDATE,
                                                MIN_JARAK_GPS_UPDATE,
this);
                                if (locManager != null)
                                {
                                        // ambil posisi terakhir user
menggunakan Network
                                        location =
locManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                                        // jika lokasi berhasil didapat
                                        if (location != null)
                                        {
                                                // ambil latitude
                                                latitude =
location.getLatitude();
                                               // ambil longitude
                                                longitude =
location.getLongitude();
                                        }
                                }
                        }

                        // jika gps bisa digunakan
                        if (isGPSEnable)
                        {
                                if (location == null)
                                {
                                        // ambil posisi berdasar GPS

      locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
MIN_WAKTU_GPS_UPDATE,

      MIN_JARAK_GPS_UPDATE, this);
                                        if (locManager != null)
                                        {
                                                // dapatkan posisi
terakhir user menggunakan GPS
                                                location =
locManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                                                // jika lokasi berhasil
didapat
                                                if (location != null)
                                                {
                                                        // ambil latitude
                                                        latitude =
location.getLatitude();

                                                        // ambil longitude
                                                        longitude =
location.getLongitude();
```

```
                                                }
                                            }
                                        }
                                    }
                                }
                            } catch (Exception e)
                            {
                                e.printStackTrace();
                            }
                            return location;

                        }

                        public double getLatitude()
                        {
                            if (location != null)
                                latitude = location.getLatitude();
                            return latitude;
                        }

                        public void setLatitude(double latitude)
                        {
                            this.latitude = latitude;
                        }

                        public double getLongitude()
                        {
                            if (location != null)
                                longitude = location.getLongitude();
                            return longitude;
                        }

                        public void setLongitude(double longitude)
                        {
                            this.longitude = longitude;
                        }

                        public boolean canGetLocation()
                        {
                            return this.canGetLocation;
                        }
```

Gambar 4.6 Implementasi Algoritma *Class* GpsService

4.4.5. Implementasi Algoritma dari Komponen *Class* Httpclient

Komponen *class* Httpclient ini digunakan untuk membuat koneksi antara aplikasi *user* dengan aplikasi *administrator*/*server*. Di dalam *class* ini terdapat 7 method diantaranya:

1. Method *Httpclient* adalah method konstruktor yang dijalankan pertama kali pada *class Httpclient*.

2. Method *addParam* digunakan untuk menambahkan parameter yang dikirim ke aplikasi *administrator/server*.
3. Method *setUrl* digunakan untuk mendefinisikan alamat *server* yang ingin digunakan.
4. Method *String getUrl* digunakan untuk mendapatkan alamat *server*.
5. Method *executeRequest* digunakan untuk mengeksekusi *request* yang dikirimkan ke *server*.
6. Method *String getResponse* digunakan untuk mendapakan respon dikirimkan oleh *server*.
7. Method *StringBuilder inputStreamToString* digunakan untuk membuat hasil respon menjadi bentuk *string*.

Implementasi algoritma *class* Httpclient terdapat pada Gambar 4.7.

```
public class Httpclient {

       private String baseUrl = "http://172.21.13.165/BPN/";
       private String url;
       ArrayList<NameValuePair> params;
       DefaultHttpClient client;
       HttpResponse httpResponse;
       String response;

public Httpclient() {
       params = new ArrayList<NameValuePair>();
    client = new DefaultHttpClient();
       }

public void addParam(String name, String value) {
    params.add(new BasicNameValuePair(name, value));
  }


public void setUrl(String url) {
       this.url = "";
             this.url = this.baseUrl+url;
       }

public String getUrl(){
       return this.url;
  }

public void executeRequest() throws Exception {
    HttpPost request = new HttpPost(url);
    try{
       if (!params.isEmpty()) {
             request.setEntity(new UrlEncodedFormEntity(params,
```

```
HTTP.UTF_8));
        }
        httpResponse = client.execute(request);

        response =
inputStreamToString(httpResponse.getEntity().getContent()).toString();
        Log.w("postKlien", response);
        Log.w("postKlien", "FALSE");

        params.clear(); //mengahapus nilai tombol yang ditekan
    } catch (ClientProtocolException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
  }

public String getResponse() {
    return response;

private StringBuilder inputStreamToString(InputStream is) {
            String line = "";
            StringBuilder total = new StringBuilder();
            BufferedReader rd = new BufferedReader(new
InputStreamReader(is));
            try {
                while ((line = rd.readLine()) != null) {
                    total.append(line);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            return total;
        }
}
```

Gambar 4.7 Implementasi Algoritma *Class* Httpclient

4.5. Implementasi Antarmuka

Implementasi antarmuka perangkat lunak GPS *Tracker* berbasis Android untuk pendataan dan pemetaan lahan terdiri dari 2 bagian, yaitu implementasi antarmuka aplikasi *user* dan implementasi antarmuka aplikasi *administrator*. Implementasi antarmuka aplikasi *user* terdapat pada *smartphone* dengan operasi sistem Android. Implementasi antarmuka aplikasi *administrator* berbentuk *webservice*.

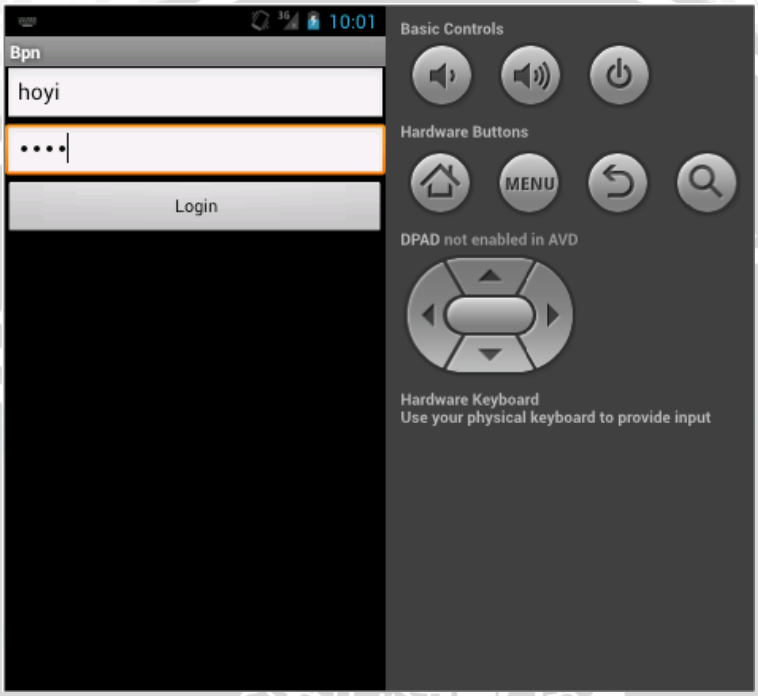
4.5.1. Implementasi Antarmuka Aplikasi *User*

Antarmuka pengguna aplikasi *user* merupakan sebuah perangkat lunak GPS *Tracker* berbasis Android. Antarmuka pengguna untuk aplikasi *user* terdiri

atas halaman login, halaman beranda, halaman *form* klien, halaman *form* lokasi, halaman cari koordinat, dan halaman cari lokasi.

1. Halaman *Login*

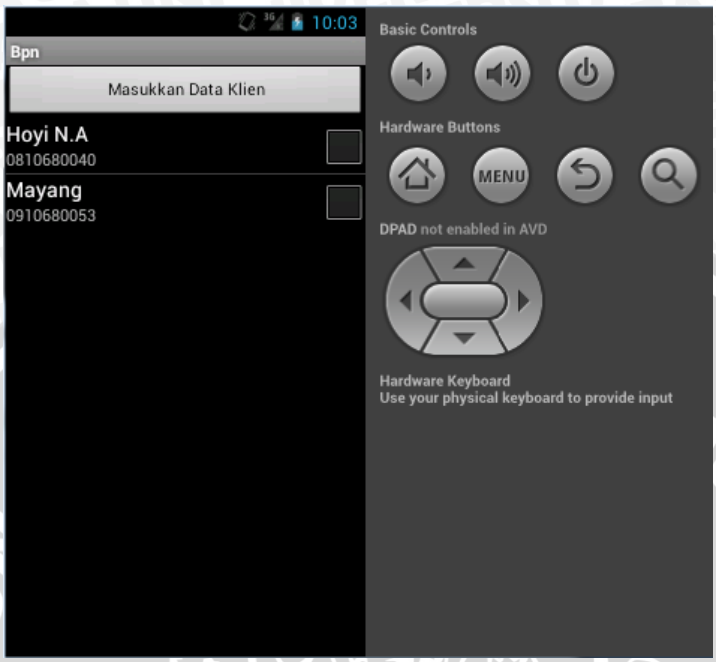
Halaman login merupakan salah satu antarmuka pengguna untuk aplikasi *user* yang berfungsi untuk *user* dalam melakukan *login*. Gambar 4.7 di bawah ini akan menunjukkan antarmuka dari halaman *login*.



Gambar 4.8 Implementasi antarmuka halaman *login*

2. Halaman Beranda

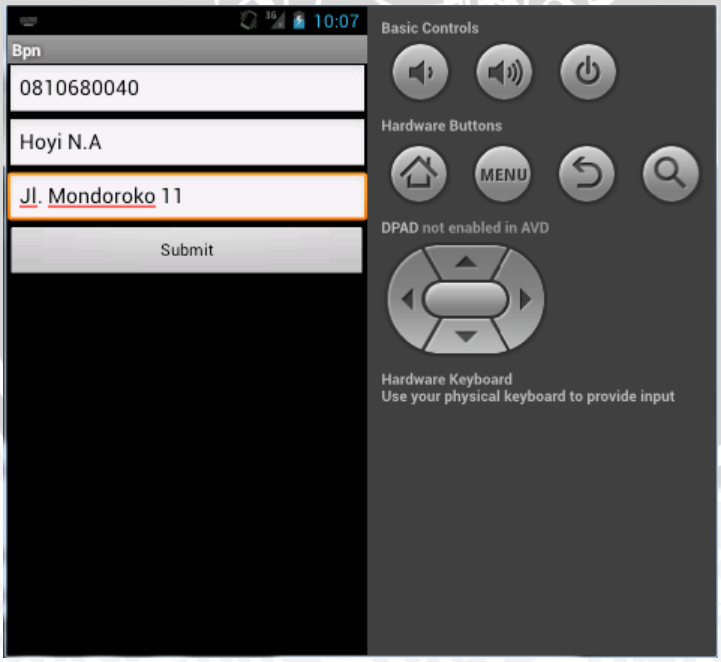
Halaman beranda adalah halaman untuk menampilkan data – data klien yang telah dicatat sebelumnya. Gambar 4.8 di bawah ini akan menunjukkan antarmuka dari halaman beranda.



Gambar 4.9 Implementasi antarmuka halaman beranda

3. Halaman *Form* Klien

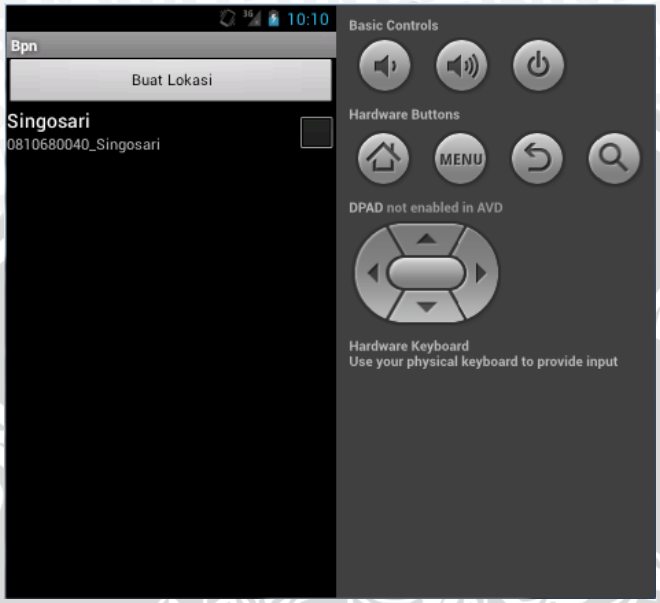
Halaman *form* klien adalah halaman untuk mencatat data – data klien. Gambar 4.9 di bawah ini akan menunjukkan antarmuka dari halaman *form* klien.



Gambar 4.10 Implementasi antarmuka halaman *form* klien

4. Halaman Cari Lokasi

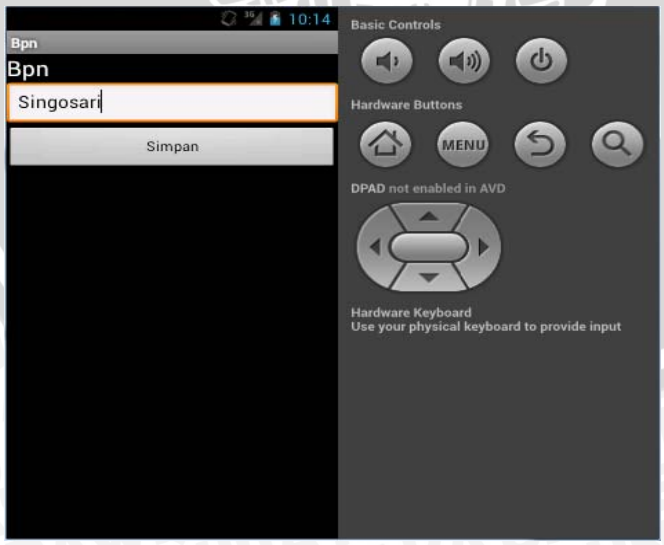
Halaman cari lokasi adalah halaman untuk menampilkan data lokasi klien yang telah dicatat sebelumnya. Gambar 4.10 di bawah ini akan menunjukkan perancangan tampilan antarmuka dari halaman cari lokasi.



Gambar 4.11 Implementasi antarmuka halaman cari lokasi

5. Halaman *Form* Lokasi

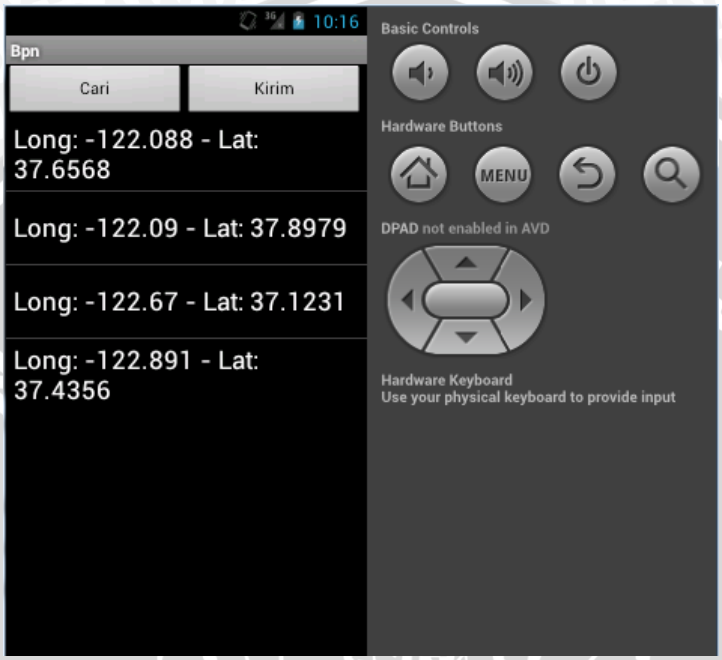
Halaman *form* lokasi adalah halaman untuk mencatat data lokasi klien. Gambar 4.11 di bawah ini akan menunjukkan antarmuka dari halaman *form* lokasi.



Gambar 4.12 Implementasi antarmuka halaman *form* lokasi

6. Halaman Cari Koordinat

Halaman cari koordinat adalah halaman untuk mencari koordinat pada sudut-sudut lahan klien dan untuk melakukan pengiriman semua data klien yang telah tercatat. Gambar 4.12 di bawah ini akan menunjukkan antarmuka dari halaman cari koordinat.



Gambar 4.13 Implementasi antarmuka halaman cari koordinat

4.5.2. Implementasi Antarmuka Aplikasi *Administrator*

Antarmuka pengguna aplikasi *administrator* merupakan halaman adminsitrasi yang dapat digunakan untuk mengelola data dari perangkat lunak GPS *Tracker* berbasis Android. Antarmuka pengguna aplikasi *administrator* ini diimplementasikan melalui sebuah *web service*.

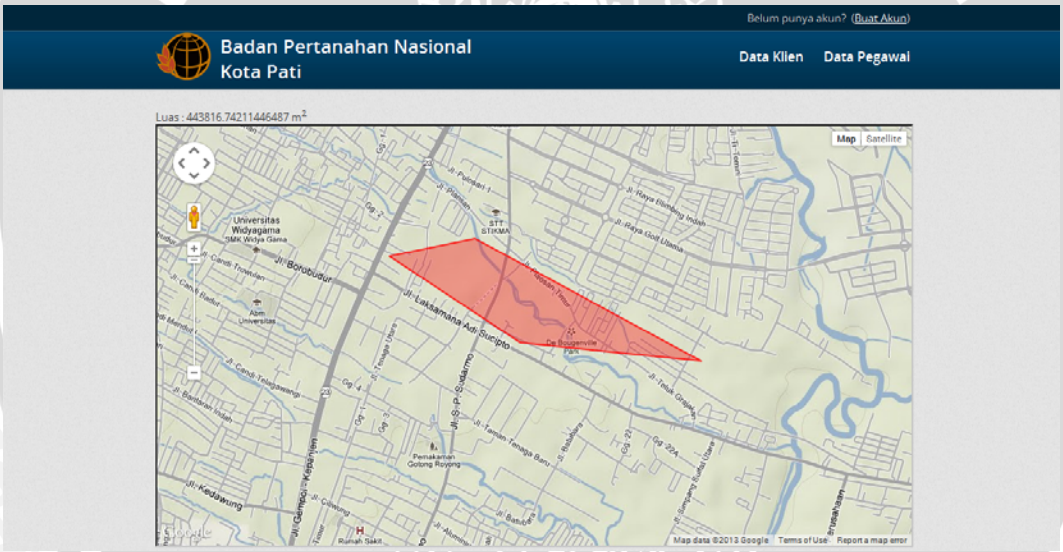
Antarmuka pengguna untuk aplikasi *administrator* terdiri atas halaman *home*, halaman data pegawai, halaman data klien, dan halaman buat akun.

1. Halaman *Home*

Halaman home merupakan halaman utama yang menampilkan data-data yang disimpan di basis data. Informasi-informasi tentang seluruh data klien dari kiriman aplikasi *user* dapat dilihat di halaman ini. Halaman ini berisi tabel yang menampung informasi tentang data-data klien. Halaman ini juga berisi link untuk melihat data klien (No. KTP, nama klien, dan alamat klien), melihat data pegawai

(*username*, *password*, dan nama pegawai), dan melihat peta. Gambar 4.13 akan menunjukkan implementasi antarmuka dari halaman login.

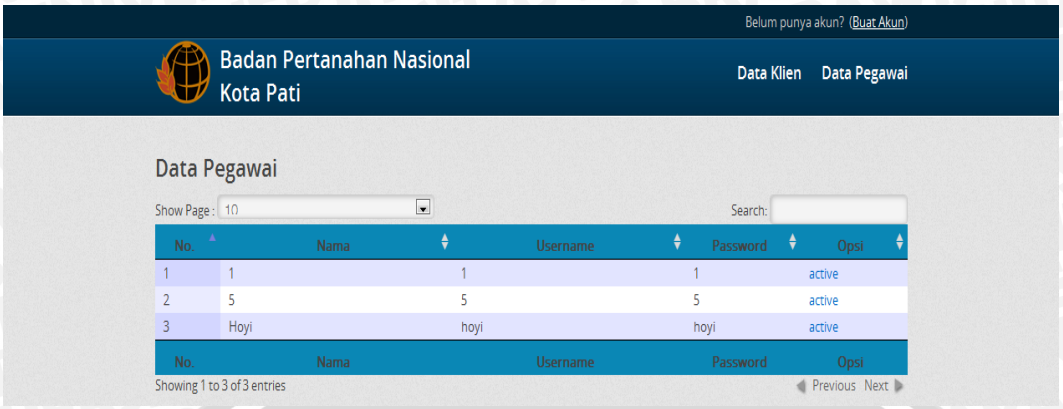
Gambar 4.14 Implementasi antarmuka halaman *home*



Gambar 4.13 Implementasi antarmuka halaman *home* (Peta)

2. Halaman Data Pegawai

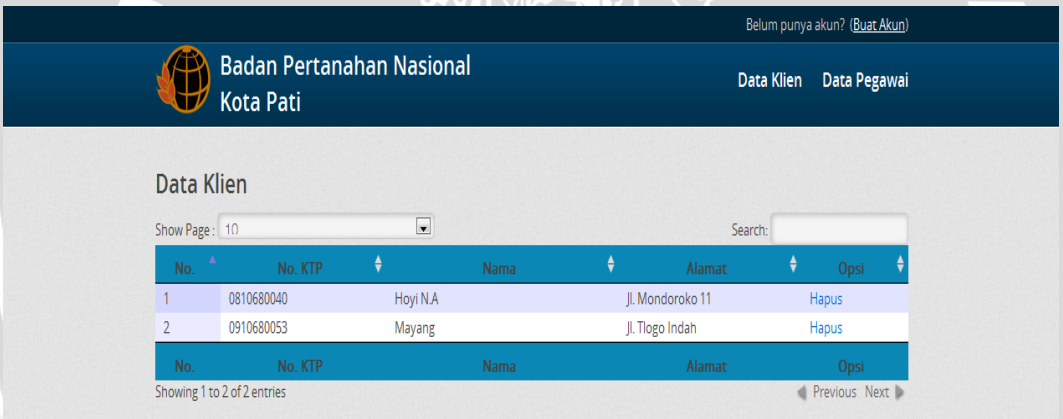
Halaman data pegawai adalah halaman yang menampilkan data-data pegawai yang disimpan di dalam basis data. Halaman ini juga berisi link untuk menon-aktifkan data pegawai.



Gambar 4.16 Implementasi antarmuka halaman data pegawai

3. Halaman Data Klien

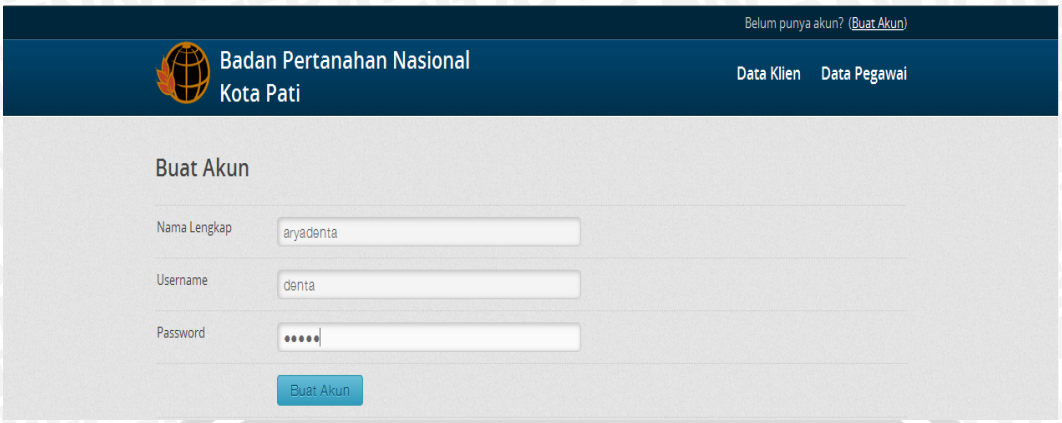
Halaman data klien adalah halaman yang menampilkan data-data klien yang disimpan di dalam basis data. Halaman ini juga berisi link untuk menghapus data klien.



Gambar 4.17 Implementasi antarmuka halaman data klien

4. Halaman Buat Akun

Halaman buat akun adalah halaman yang menampilkan pembuatan akun untuk pegawai atau *user* agar dapat *login* dalam aplikasi *user*.



Gambar 4.18 Implementasi antarmuka halaman buat akun

BAB V
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis perangkat lunak GPS *tracker* berbasis Android yang telah dikembangkan. Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian validasi, pengujian performa, dan UAT *(User Acceptance Testing)*. Pada pengujian validasi akan digunakan teknik pengujian *Black-Box* (*Black-Box Testing*). Pada pengujian performa akan dilakukan perbandingan antara alat GPS asli dengan GPS *tracker* yang berada dalam Android untuk mengetahui ketepatan titik koordinat yang dihasilkan. Pada UAT *(User Acceptance Testing)* dilakukan pada pegawai BPN yang telah menggunakan perangkat lunak ini, dengan mengajukan beberapa pertanyaan.

5.1. Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. *Item - item* yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap perangkat lunak GPS *tracker* berbasis Android untuk pendataan dan pemetaan lahan.

5.1.1. Kasus Uji Validasi Aplikasi *User*

1. Kasus Uji *Login*

Tabel 5.1 Kasus uji untuk pengujian validasi *login*

Nama Kasus Uji	*Login*
Objek Uji	Kebutuhan Fungsional (SRS_001_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk

	melakukan login bagi *User*.
Prosedur Uji	1. *User* mengeksekusi *class Login.* 2. Karena data *User* sudah ada, maka akan dilakukan *redirecting* ke halaman login *User.* 3. *User* memasukkan data yang diperlukan di dalam halaman *login.* 4. *User* menekan tombol *Login.*
Hasil yang Diharapkan	Sistem dapat menjalankan proses login terlebih dahulu sebelum mengoperasikan aplikasi lebih lanjut.

2. Kasus Uji Menambah Informasi Data Klien

Tabel 5.2 Kasus uji untuk pengujian validasi menambah informasi data klien

Nama Kasus Uji	Menambah Informasi Data Klien
Objek Uji	Kebutuhan Fungsional (SRS_001_02)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk menambahkan informasi data klien.
Prosedur Uji	1. *User* mengeksekusi *class* Beranda. 2. *User* menekan tombol Buat Lokasi. 3. *User* mengeksekusi *class* FormKlien. 4. *User* memasukkan data yang diperlukan di dalam halaman FormKlien (No. Identitas, Nama, dan Alamat). 5. *User* menekan tombol *Submit*. 6. *User* mengeksekusi class CariLokasi. 7. *User* menekan tombol Buat Lokasi. 8. *User* memasukkan data yang diperlukan di dalam halaman FormKlien (Alamat Lokasi). 9. *User* menekan tombol Simpan. 10. *User* mengeksekusi *class* CariKoordinat. 11. *User* melakukan pencarian koordinat dengan

	menekan tombol Cari.
Hasil yang Diharapkan	Sistem dapat menjalankan proses penambahan informasi data klien terlebih dahulu sebelum mengoperasikan aplikasi lebih lanjut.

3. Kasus Uji Menghapus Informasi Data Klien

Tabel 5.3 Kasus uji untuk pengujian validasi menghapus informasi data klien

Nama Kasus Uji	Menghapus Informasi Data Klien
Objek Uji	Kebutuhan Fungsional (SRS_001_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk menghapus informasi data klien.
Prosedur Uji	1. *User* mengeksekusi *class* Beranda atau class CariLokasi. 2. *User* memilih data yang akan dihapus dengan menekan *checkbox* di samping data. 3. *User* melakukan penghapusan data dengan menekan tombol hapus atau hapus semua pada *context menu* Android.
Hasil yang Diharapkan	Sistem dapat menjalankan proses penghapusan informasi data klien yang diinginkan.

4. Kasus Uji Mengirim Informasi Data Klien

Tabel 5.4 Kasus uji untuk pengujian validasi mengirim informasi data klien

Nama Kasus Uji	Mengirim Informasi Data Klien
Objek Uji	Kebutuhan Fungsional (SRS_001_04)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengirim informasi data klien.
Prosedur Uji	1. *User* mengeksekusi *class* CariKoordinat.

	2. *User* mengirim data klien ke aplikasi *administrator* atau *server* dengan menekan tombol kirim.
Hasil yang Diharapkan	Sistem dapat menjalankan proses pengiriman informasi data klien.

5. Kasus Uji *Logout*

Tabel 5.5 Kasus uji untuk pengujian validasi *logout*

Nama Kasus Uji	*Logout*
Objek Uji	Kebutuhan Fungsional (SRS_001_05)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan logout.
Prosedur Uji	1. *User* masuk ke dalam *context menu* Android. 2. *User* menekan tombol *Logout*.
Hasil yang Diharapkan	Sistem dapat menjalankan fungsi *Logout* pada aplikasi.

5.1.1.1. Hasil Pengujian Validasi Aplikasi *User*

Hasil pengujian pada setiap kasus uji pada aplikasi *user* akan dijabarkan pada tabel 5.6 untuk mengetahui apakah setiap fungsi pada aplikasi telah bernilai valid untuk memenuhi kebutuhan fungsional aplikasi GPS *tracker* berbasis Android untuk pendataan dan pemetaan lahan.

Tabel 5.6 Hasil pengujian validasi

No	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validasi
1	Login (SRS_001_01)	Sistem dapat menjalankan proses login terlebih dahulu sebelum	Sistem dapat menjalankan proses login terlebih dahulu sebelum	Valid

		mengoperasikan aplikasi lebih lanjut.	mengoperasikan aplikasi lebih lanjut.	
2	Menambah Informasi Data Klien (SRS_001_02)	Sistem dapat menjalankan proses penambahan informasi data klien terlebih dahulu sebelum mengoperasikan aplikasi lebih lanjut.	Sistem dapat menjalankan proses penambahan informasi data klien terlebih dahulu sebelum mengoperasikan aplikasi lebih lanjut.	Valid
3	Menghapus Informasi Data Klien (SRS_001_03)	Sistem dapat menjalankan proses penghapusan informasi data klien yang diinginkan.	Sistem dapat menjalankan proses penghapusan informasi data klien yang diinginkan.	Valid
4	Mengirim Informasi Data Klien (SRS_001_04)	Sistem dapat menjalankan proses pengiriman informasi data klien.	Sistem dapat menjalankan proses pengiriman informasi data klien.	Valid
5	Logout (SRS_001_5)	Sistem dapat menjalankan fungsi Logout dari aplikasi.	Sistem dapat menjalankan fungsi Logout dari aplikasi.	Valid

5.1.2. Kasus Uji Validasi Aplikasi *Administrator*

1. Kasus Uji Membuat Akun Pegawai

Tabel 5.7 Kasus uji untuk pengujian validasi membuat akun pegawai

Nama Kasus Uji	Membuat Akun Pegawai
Objek Uji	Kebutuhan Fungsional (SRS_002_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk membuat akun pegawai.
Prosedur Uji	1. *Administrator* mengeksekusi *class* daftar atau halaman buat akun. 2. *Administrator* memasukkan data yang diperlukan di dalam halaman daftar. 3. *Administrator* menekan tombol Buat akun.
Hasil yang Diharapkan	Sistem dapat menjalankan proses pembuatan akun pegawai.

2. Kasus Uji Melihat Informasi Data Pegawai

Tabel 5.8 Kasus uji untuk pengujian validasi melihat informasi data pegawai

Nama Kasus Uji	Melihat Informasi Data Pegawai
Objek Uji	Kebutuhan Fungsional (SRS_002_02)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat informasi data pegawai.
Prosedur Uji	1. *Administrator* mengeksekusi *class* pegawai atau halaman data pegawai. 2. *Administrator* mendapat akses untuk mengubah data pegawai dengan menekan tombol *active/deactive* pada kolom opsi.
Hasil yang Diharapkan	Sistem dapat menjalankan proses melihat informasi data pegawai.

3. Kasus Uji Melihat Informasi Data Klien

Tabel 5.9 Kasus uji untuk pengujian validasi melihat informasi data klien

Nama Kasus Uji	Melihat Informasi Data Klien
Objek Uji	Kebutuhan Fungsional (SRS_002_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat informasi data klien.
Prosedur Uji	*Administrator* mengeksekusi *class* klien atau halaman data klien.
Hasil yang Diharapkan	Sistem dapat menjalankan proses melihat informasi data klien.

4. Kasus Uji Melihat Seluruh Informasi Data Kiriman

Tabel 5.10 Kasus uji untuk pengujian validasi melihat seluruh informasi data kiriman

Nama Kasus Uji	Melihat Seluruh Informasi Data Kiriman
Objek Uji	Kebutuhan Fungsional (SRS_002_04)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat seluruh informasi data kiriman.
Prosedur Uji	*Administrator* mengeksekusi *class* index atau halaman *home*.
Hasil yang Diharapkan	Sistem dapat menjalankan proses melihat seluruh informasi data kiriman.

5. Kasus Uji Melihat Peta dan Luas Lokasi

Tabel 5.11 Kasus uji untuk pengujian validasi melihat peta dan luas lokasi

Nama Kasus Uji	Melihat Peta Dan Luas Lokasi
Objek Uji	Kebutuhan Fungsional (SRS_002_05)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat peta dan luas lokasi.
Prosedur Uji	1. *Administrator* mengeksekusi *class maps* atau halaman lokasi. 2. *Administrator* melihat peta dan luas lokasi dengan menekan tombol Lihat peta pada kolom opsi halaman index.
Hasil yang Diharapkan	Sistem dapat menjalankan proses melihat peta dan luas lokasi.

5.1.2.1. Hasil Pengujian Validasi Aplikasi *Administrator*

Hasil pengujian pada setiap kasus uji pada aplikasi a*dministrator* akan dijabarkan pada tabel 5.11 untuk mengetahui apakah setiap fungsi pada aplikasi telah bernilai valid untuk memenuhi kebutuhan fungsional aplikasi GPS *tracker* berbasis Android untuk pendataan dan pemetaan lahan.

Tabel 5.12 Hasil pengujian validasi

No	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validasi
1	Buat Akun (SRS_002_01)	Sistem dapat menjalankan proses pembuatan akun pegawai.	Sistem dapat menjalankan proses pembuatan akun pegawai.	Valid
2	Melihat Informasi Data	Sistem dapat menjalankan proses	Sistem dapat menjalankan proses	Valid

	Pegawai (SRS_001_02)	melihat informasi data pegawai.	melihat informasi data pegawai.	
3	Melihat Informasi Data Klien (SRS_001_03)	Sistem dapat menjalankan proses melihat informasi data klien.	Sistem dapat menjalankan proses melihat informasi data klien.	Valid
4	Melihat Seluruh Informasi Data Kiriman (SRS_001_04)	Sistem dapat menjalankan proses melihat seluruh informasi data kiriman.	Sistem dapat menjalankan proses melihat seluruh informasi data kiriman.	Valid
5	Melihat Peta dan Luas Lokasi (SRS_001_5)	Sistem dapat menjalankan proses melihat peta dan luas lokasi.	Sistem dapat menjalankan proses melihat peta dan luas lokasi.	Valid

5.2. Pengujian Performa

Pengujian performa dilakukan untuk mengetahui bagaimana performa perangkat lunak GPS *tracker* berbasis Android untuk pendataan dan pemetaan lahan dalam ketepatan pengambilan titik koordinat. Pengujian performa dilakukan dengan membandingkan perangkat GPS dan GPS pada *Smartphone* Android atau *Assisted* GPS.

Pengujian performa dilakukan dengan menggunakan GPS GARMIN 60 dan Samsung Galaxy S3 dengan spesifikasi perangkat keras yang ditunjukkan oleh Tabel 5.13 dan Tabel 5.14.

Tabel 5.13 Spesifikasi perangkat keras GPS

GPS GARMIN 60	
Waypoints/icons	500 *with name and graphic symbol,* 10 *nearest (automatic),* 10 *proximity*
Routes	50 *reversible routes with up to* 250 *points each, plus* MOB *and TracBack® modes*
Tracks	10K *point automatic track log;* 20 *saved tracks* 500 *points each let you retrace your path in both directions*
Trip computer	*Current speed, average speed, resettable max. speed, trip timer and trip distance*
Tables	*Built-in celestial tables for best times to fish and hunt, sun and moon rise, set and location*
Map datums	*More than 100 plus user datum*
Position format	Lat/Lon, UTM/UPS, *Maidenhead,* MGRS, *Loran* TDs *and other grids, including user grid*
GPS *accuracy*	*Position:* < 15 *meters,* 95% *typical, Velocity:* 0.05 *meter/sec steady state*
DGPS (WAAS) *accuracy*	*Position:* < 3 *meters,* 95% *typical, Velocity:* 0.05 *meter/sec steady state*
Interfaces	USB, RS232 *with* NMEA 0183, RTCM 104 DGPS *data format and proprietary Garmin*
Antenna	*Built-in quadrifilar, with external antenna connection* (MCX)
Differential	DGPS, USCG *and* WAAS *capable*
Receiver	WAAS *enabled,* 12 *parallel channel* GPS *receiver continuously tracks and uses up to* 12 *satellites to compute and update your position*

Tabel 5.14 Spesifikasi perangkat lunak *Smartphone* Android

Samsung Galaxy S3	
Network/Bearer and Wireless Connectivity	GSM, HSPA + 21, EDGE / GPRS (850 / 900 / 1.800, 1.900MHZ), HSPA + 21 (850 / 900 / 1.900 / 2.100), WiFi *Direct*, GAP, SSP, HSP, HFP1.5, A2DP, SPP, OPP, PBAP, MAP, AVRCP 1.3, HID, NFC *available*, DLNA, MHL 1.0, HDMI 2 *support* KIES, KIES Air *support*
Display	HD sAMOLED, 16M, 4,8", 720 x 1280 (HD)
Chipset	*Quad Core Application Processor*, 1,4GHz CPU *Speed*
Audio and Video	MPEG4, H.264, H.263, VC-1. DivX, VP8, WMV7/8, *Sorenson Spark*, DivX3.11, *Full* HD (1080p) *Video Recording & Playback, Recording up to* 30fps, MP3, AMR-NB / WB, AAC / AAC + / eAAC+, WMA, OGG *(Vorbis)*, FLAC, AC-3, apt-X
Location	*Assisted* GPS / GLONASS *available*

Perhitungan perbandingan perangkat GPS dan GPS pada *Smartphone* Android atau *Assisted* GPS dalam memperoleh titik koordinat akan dijelas pada tabel 5.14 di bawah ini:

Tabel 5.15 Perbandingan alat GPS dan *Assited* GPS

No.	GPS		A-GPS		Selisih Jarak (m)
	Latitude	Longitude	Latitude	Longitude	
1.	-7.95344	112.61567	-7.953492	112.615688	2
2.	-7.95347	112.61591	-7.953487	112.616133	9
3.	-7.95325	112.61626	-7.953175	112.616271	6
4.	-7.95325	112.61526	-7.953116	112.616528	3
5.	-7.95315	112.61668	-7.952305	112.616678	2
6.	-7.95199	112.61604	-7.951849	112.616091	1

7.	-7.95247	112.61298	-7.952631	112.612291	3
8.	-7.95251	112.61337	-7.952501	112.612801	3
9.	-7.95223	112.61356	-7.952121	112.613499	4
10.	-7.95219	112.65344	-7.952101	112.612905	6
11.	-7.95364	112.61491	-7.954042	112.614748	12
12.	-7.95365	112.61478	-7.953868	112.614445	10
13.	-7.95356	112.61465	-7.953626	112.614506	4
14.	-7.95355	112.61492	-7.953608	112.614911	2
15.	-7.96025	112.61797	-7.960617	112.617842	6
16.	-7.95993	112.61729	-7.960218	112.617943	4
17.	-7.96086	112.61774	-7.960194	112.617057	4
18.	-7.96019	112.61703	-7.959908	112.617263	7
19.	-7.96273	112.61414	-7.962991	112.614686	5
20.	-7.96291	112.61446	-7.962797	112.614846	1
21.	-7.96279	112.61475	-7.962669	112.614217	4
22.	-7.96255	112.61430	-7.962505	112.614489	6
23.	-7.96161	112.61639	-7.962143	112.617116	2
24.	-7.96074	112.61691	-7.961614	112.616458	6
25.	-7.96123	112.61754	-7.961297	112.617568	8
26.	-7.96223	112.61715	-7.960694	112.616913	3
27.	-7.95965	112.61861	-7.960366	112.618928	5
28.	-7.95994	112.61915	-7.960028	112.618345	5
29.	-7.95999	112.61835	-7.959966	112.619201	3
30.	-7.96041	112.61909	-7.959599	112.618591	3
31.	-7.95964	112.61848	-7.959602	112.618298	3
32.	-7.95941	112.61777	-7.959244	112.617674	3
33.	-7.95868	112.61756	-7.958909	112.618781	4
34.	-7.95886	112.61880	-7.958508	112.618078	9
35.	-7.95626	112.61656	-7.956509	112.616706	7
36.	-7.95616	112.61636	-7.956287	112.616399	6
37.	-7.95653	112.61665	-7.956242	112.616925	5
38.	-7.95541	112.61678	-7.955951	112.616526	8
39.	-7.95622	112.61331	-7.956164	112.613452	2
40.	-7.95584	112.61536	-7.955857	112.615534	2
41.	-7.95273	112.61565	-7.953066	112.615534	5
42.	-7.95281	112.61561	-7.952931	112.614562	1
43.	-7.95076	112.61327	-7.950716	112.613508	1
44.	-7.95028	112.61328	-7.950558	112.612309	8
45.	-7.95007	112.61236	-7.950186	112.613498	9
46.	-7.95076	112.61230	-7.950061	112.612374	6

47.	-7.95363	112.61489	-7.953647	112.614862	10
48.	-7.95373	112.61523	-7.953684	112.615284	4
49.	-7.95393	112.61523	-7.953962	112.615211	9
50.	-7.95389	112.61492	-7.953874	112.614916	3
Rata – rata = $\frac{Selisih\ Jarak\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Koordinat}$					4,88

Dari 50 titik koordinat yang diambil, diperoleh rata-rata selisih jarak antara perangkat GPS dengan A-GPS dengan hasil akhir 4,88 Meter, sehingga dapat disimpulkan bahwa keakuratan antara perangkat GPS dengan A-GPS mempunyai perbedaan yaitu kurang lebih 4-5 Meter disetiap pengambilan titik koordinat.

Perhitungan perbandingan perangkat GPS dan GPS pada *Smartphone* Android atau *Assisted* GPS saat menemukan titik koordinat/*respone time* pada cuaca cerah dan pada cuaca mendung akan dijelas pada tabel 5.16 di bawah ini:

Tabel 5.16 Perbandingan *Respone Time* alat GPS dan *Assited* GPS

No	GPS				A-GPS			
	Cuaca Cerah		Cuaca Mendung		Cuaca Cerah		Cuaca Mendung	
	Respone Time (s)	Akurasi (m)	*Respone Time (s)*	Akurasi (m)	*Respone Time (s)*	Akurasi (m)	*Respone Time (s)*	Akurasi (m)
1	4	± 5	60	± 40	0	± 5	2	± 10
2	2	± 8	43	± 26	0	± 5	6	± 15
3	0	± 7	40	± 48	0	± 5	1	± 5
4	0	± 4	15	± 56	0	± 5	0	± 5
5	0	± 5	27	± 43	0	± 5	0	± 5
6	0	± 5	42	± 17	0	± 5	0	± 5
7	3	± 6	38	± 33	0	± 5	2	± 5
8	2	± 5	30	± 28	0	± 5	2	± 15
9	2	± 9	7	± 23	0	± 5	1	± 15
10	3	± 10	16	± 19	0	± 5	1	± 10
11	0	± 5	54	± 56	0	± 5	3	± 5
12	1	± 10	28	± 22	0	± 5	1	± 10
13	1	± 6	10	± 15	0	± 5	0	± 5
14	1	± 6	9	± 16	0	± 5	2	± 5
15	0	± 8	72	± 12	0	± 5	0	± 5
16	4	± 5	42	± 32	0	± 5	0	± 5
17	3	± 6	81	± 5	0	± 5	5	± 10

18	2	± 5	22	± 11	0	± 5	0	± 10
19	2	± 5	61	± 48	0	± 5	2	± 10
20	0	± 8	5	± 20	0	± 5	2	± 5

Keterangan :

- Pada GPS:
  - ✓ Rata-rata respon time cuaca cerah:

    $$\text{Rata} - \text{rata} = \frac{Respon\ Time\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Respon\ Time} = 1.5\ second$$

    $$\text{Rata} - \text{rata} = \frac{Akurasi\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Akurasi} = \pm\ 6.4 \text{ meter}$$

  - ✓ Rata-rata respon time cuaca Mendung:

    $$\text{Rata} - \text{rata} = \frac{Respon\ Time\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Respon\ Time} = 35.1\ second$$

    $$\text{Rata} - \text{rata} = \frac{Akurasi\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Akurasi} = \pm\ 28.5 \text{ meter}$$

- Pada A-GPS:
  - ✓ Rata-rata respon time cuaca cerah:

    $$\text{Rata} - \text{rata} = \frac{Respon\ Time\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Respon\ Time} = 0\ second$$

    $$\text{Rata} - \text{rata} = \frac{Akurasi\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Akurasi} = \pm\ 5 \text{ meter}$$

  - ✓ Rata-rata respon time cuaca Mendung:

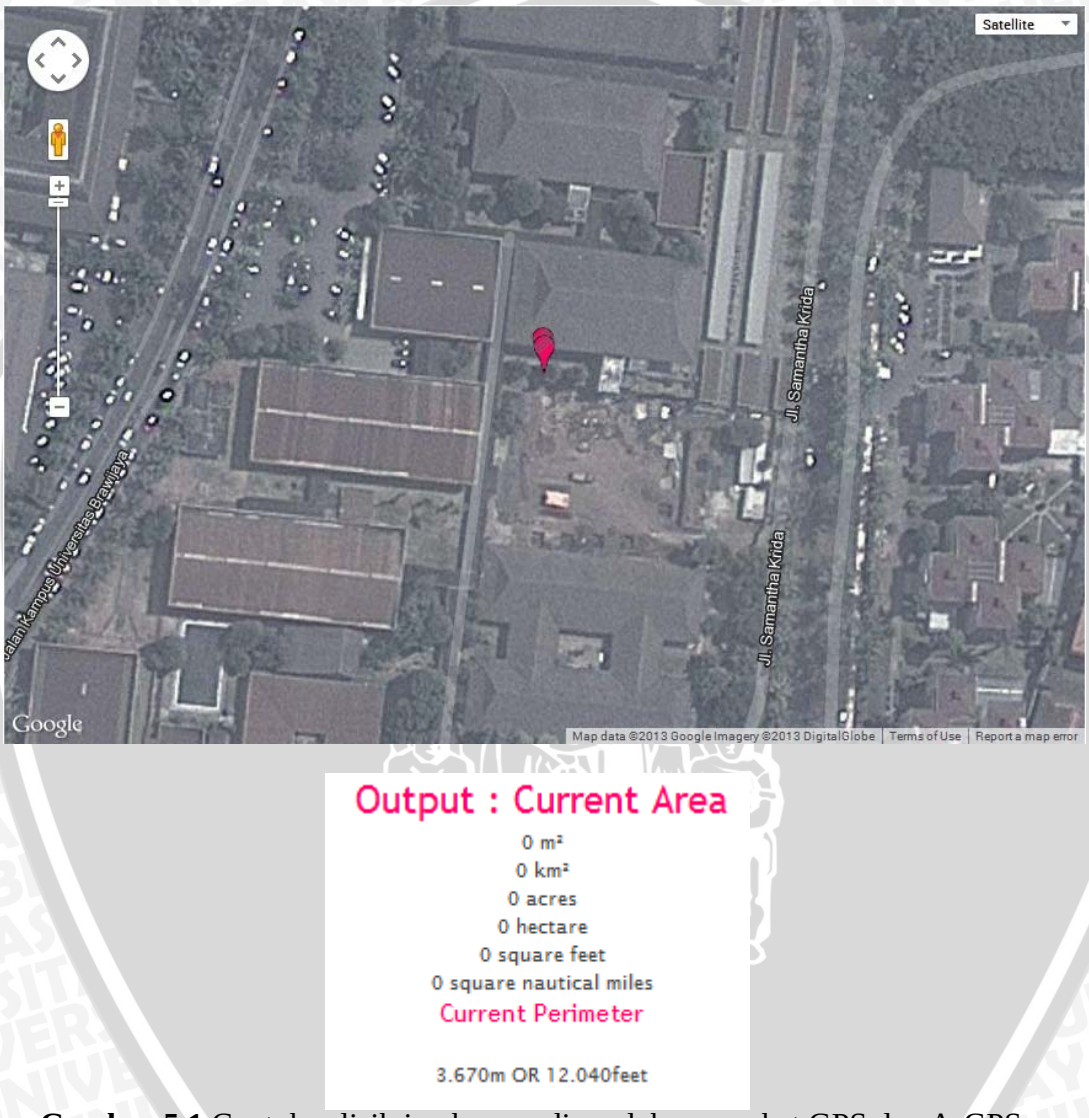
    $$\text{Rata} - \text{rata} = \frac{Respon\ Time\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Respon\ Time} = 1.5\ second$$

    $$\text{Rata} - \text{rata} = \frac{Akurasi\ (n1 + n2 + n3 + n....)}{Jumlah\ Total\ Akurasi} = \pm\ 8 \text{ meter}$$

5.2.1. Gambar Hasil Perbandingan Antara Perangkat GPS dengan A-GPS

Berikut beberapa gambar hasil perbandingan selisih jarak dan luas bidang pada peta antara perangkat GPS dengan A-GPS dalam memperoleh titik koordinat.

5.2.1.1. Gambar Hasil Perbandingan Selisih Jarak



Gambar 5.1 Contoh selisih jarak yang diperoleh perangkat GPS dan A-GPS.

Output : Current Area

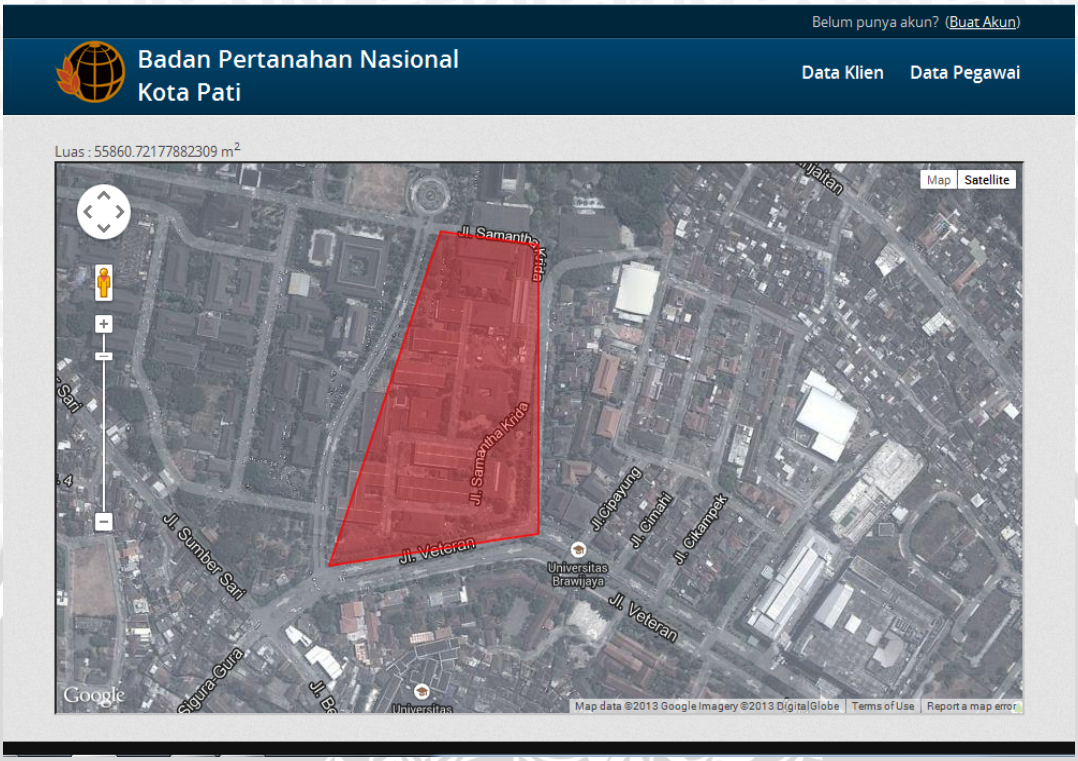
0 m²
0 km²
0 acres
0 hectare
0 square feet
0 square nautical miles

Current Perimeter

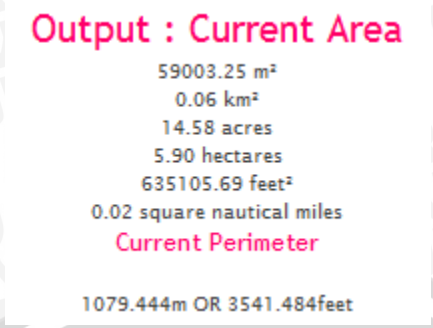
8.265m OR 27.116feet

Gambar 5.2 Contoh selisih jarak yang diperoleh perangkat GPS dan A-GPS.

5.2.1.2. Gambar Hasil Perbandingan Luas Bidang Dalam Peta



Gambar 5.3 Luas bidang dalam peta pada pengukuran menggunakan A-GPS



Gambar 5.4 Luas bidang dalam peta pada pengukuran menggunakan GPS

Perangkat GPS menentukan lokasi dari minimal 3 satelit yang membentuk kawasan segitiga dengan mencari longitude, latitude, lokasi dalam peta, dan data lainnya yang diperlukan. Hal ini membutuhkan waktu kira- kira 12 menit untuk menentukan lokasi ketika perangkat khusus GPS dinyalakan. Selain itu perangkat GPS harus berada diluar ruangan, bahkan harus berada di bawah langit terbuka. Kekuatan sinyal dapat berkurang kalau perangkat GPS berada dibawah pohon, di bawah gedung-gedung pencakar langit, atau didalam kendaraan. Namun setidaknya *user* tidak perlu memiliki akses ke operator ponsel untuk mengoperasikan GPS.

Sedangkan A-GPS dapat menemukan lokasi dengan lebih cepat, bahkan kurang dari 5 detik. Ini dikarenakan A-GPS langsung mencari satelit yang terdekat dengan lokasi ponsel saat itu melalui operator telekomunikasi. Dibutuhkan 3 komponen untuk bisa melakukan hal ini, yaitu satelit, *assistance* server (GSM) dan *receiver* A-GPS. Kekurangannya adalah masih tergantung pada *coverage* GSM operator.

5.3. UAT *(User Acceptance Testing)*

UAT *(User Acceptance Test)* UAT digunakan untuk mengetahui apakah sistem yang dibangun atau dikembangkan telah dapat diterima atau belum oleh pengguna sistem. UAT dilakukan dengan memberikan kuesioner kepada 30 pegawai BPN untuk menilai keseluruhan sistem dan memberikan komentar maupun saran terhadap sistem. Pada tabel 5.17 adalah beberapa pertanyaan pada kuisioner yang diajukan kepada 30 orang pegawai BPN.

Tabel 5.17 Kuisioner UAT

No	Pertanyaan	Nilai				
		Sangat Kurang	**Kurang**	**Cukup**	**Bagus**	**Sangat Bagus**
1	Bagaimana kemudahan navigasi atau menu dalam aplikasi?					
2	Bagaimana penampilan visual aplikasi?					
3	Bagaimana sistem/kinerja aplikasi?					
4	Bagaimana ketepatan *Assisted* GPS pada Android?					
5	Bagaimana kualitas keseluruhan aplikasi?					
6	Komentar dan saran					

Pada Tabel 5.18 menjelaskan bahwa aplikasi yang sudah dibuat memiliki tingkat kelayakan dalam kriteria sangat bagus sesuai dengan tanggapan dari beberapa responden atau pengguna aplikasi.

Tabel 5.18 Hasil kuisioner UAT

Pertanyaan	Jumlah Yang Diperoleh				
	Sangat Kurang	Kurang	Cukup	Bagus	Sangat Bagus
Bagaimana kemudahan navigasi atau menu dalam aplikasi?	0	0	0	0	30

Bagaimana penampilan visual aplikasi?	0	0	0	0	30
Bagaimana sistem/kinerja aplikasi?	0	0	0	0	30
Bagaimana ketepatan *Assisted* GPS pada Android?	0	0	0	2	28
Bagaimana kualitas keseluruhan aplikasi?	0	0	0	2	28

5.4. Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian perangkat lunak GPS *tracker* berbasis Android yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap metode pengembangan perangkat lunak dan hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis metode pengembangan perangkat lunak, analisis hasil pengujian validasi, analisis hasil pengujian performa, dan analisis hasil UAT *(User Acceptance Testing).*

5.4.1. Analisis Metode Pengembangan Perangkat Lunak

Rekayasa perangkat lunak CBSE *(Component Based Software Engineering)* memiliki keuntungan jelas dalam mengurangi biaya dan resiko.

- Biaya, dari segi waktu pengerjaan penggunaan metode CBSE *(Component Based Software Engineering)* dapat mempercepat pengerjaan proyek. Sehingga dapat meminimalisir biaya yang dikeluarkan.
- Resiko, dengan menggunakan komponen-komponen yang sudah ada dan sudah diuji sebelumnya, maka akan tidak ada atau sedikit resiko yang timbul.

Pada umumnya kendala yang timbul adalah komponen-komponen yang ada tidak benar-benar mampu memenuhi spesifikasi yang diharapkan. Namun

pada kasus penelitian ini, semua komponen sudah dapat memenuhi spesifikasi kebutuhan.

5.4.2. Analisis Hasil Pengujian Validasi

Proses analisis terhadap hasil pengujian validasi dilakukan dengan melihat konformitas antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas perangkat lunak GPS *Tracker* Berbasis Android telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

5.4.3. Analisis Hasil Pengujian Performa

Proses analisis terhadap hasil pengujian performa dilakukan dengan analisis hasil pengujian perbandingan ketepatan perangkat GPS dengan A-GPS dan hasil pengujian perbandingan *respon time* perangkat GPS dengan A-GPS dan.

5.4.3.1. Analisis Hasil Pengujian Perbandingan Ketepatan Perangkat GPS dengan A-GPS

Berdasarkan pada tabel 5.15, proses analisis terhadap hasil pengujian perbandingan ketepatan perangkat GPS dengan A-GPS dilakukan dengan menghitung rata-rata dari selisih jarak pada 50 titik koordinat yang diperoleh perangkat GPS dan A-GPS adalah 4,88 Meter, sehingga dapat disimpulkan bahwa keakuratan antara perangkat GPS dengan A-GPS mempunyai perbedaan yaitu kurang lebih 4-5 Meter disetiap pengambilan titik koordinat.

5.4.3.2. Analisis Hasil Pengujian Perbandingan *Respon Time* Perangkat GPS dengan A-GPS

Berdasarkan pada tabel 5.16, proses analisis terhadap hasil pengujian perbandingan *respon time* perangkat GPS dengan A-GPS dilakukan 20 kali pengambilan titik koordinat pada saat cuaca mendung dan pada saat cuaca cerah dengan menghitung rata-rata dari *respon time* dan akurasi. Pada saat cuaca cerah A-GPS bekerja lebih maksimal daripada perangkat GPS dengan hasil rata-rata *respon time* 0 second dan akurasi $\pm$ 5 meter. Sedangkan dengan perangkat GPS diperoleh hasil rata-rata *respon time* 1.5 second dan akurasi $\pm$ 6.4 meter. Dan pada saat cuaca mendung pun A-GPS bekerja lebih maksimal daripada perangkat GPS dengan hasil rata-rata *respon time* 1.5 second dan akurasi $\pm$ 8 meter. Sedangkan

dengan perangkat GPS diperoleh hasil rata-rata *respon time* 35.1 second dan akurasi ± 28.5 meter.

5.4.4. Analisis Hasil UAT *(User Acceptance Testing)*

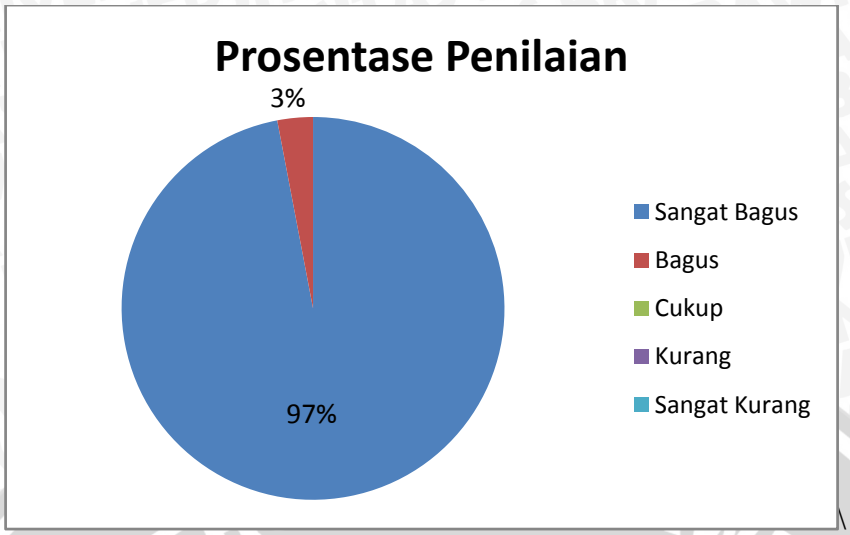
Proses analisis terhadap hasil pengujian sistem terhadap pengguna dilakukan dengan menghitung jumlah tiap nilai dari semua tanggapan responden. Pengujian sistem terhadap pengguna memiliki lima tingkatan penilaian dengan mengajukan lima pertanyaan terhadap sistem kepada responden.

Adapun tabel 5.19 menjelaskan tentang prosentase dari jumlah keseluruhan dari setiap penilaian yang diberikan oleh responden. Setiap nilai dari masing-masing pertanyaan akan dijumlahkan. Penjumlahan tersebut dimaksudkan untuk mengetahui rata-rata keseluruhan penilaian yang diberikan oleh responden.

Tabel 5.19 Prosentase Jumlah Keseluruhan Tiap Tingkatan Penilaian.

Tingkatan Penilaian	Jumlah Keseluruhan Penilaian	Prosentase
Sangat Bagus	146	97%
Bagus	4	3%
Cukup	0	0%
Kurang	0	0%
Sangat Kurang	0	0%

Gambar 5.5 menunjukkan bahwa perangkat lunak yang sudah dibuat memiliki tanggapan yang sangat bagus dari responden. 97% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori sangat bagus dan 3% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori bagus.



Gambar 5.5 Prosentase Jumlah Keseluruhan Tiap Tingkatan Penilaian

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Perancangan perangkat lunak GPS *Tracker* berbasis Android telah dibuat sesuai dengan spesifikasi kebutuhan yang telah dianalisa.
2. Metode pengembangan perangkat lunak yang digunakan dalam perangkat lunak GPS *tracker* berbasis Android menggunakan CBSE *(component-based sofware engineering)*. Dengan model tersebut dapat mempermudah dalam proses pengembangan sistem dari segi biaya dan resiko dengan menggunakan komponen yang telah tersedia.
3. Perangkat lunak GPS *Tracker* berbasis Android telah dibuat sesuai perancangan dan diimplementasikan dari komponen-komponen yang telah ditentukan.
4. Aplikasi *user* perangkat lunak GPS *tracker* berbasis Android mengirimkan koordinat ke aplikasi *administrator/server* yang dikonversikan menjadi sebuah bidang dalam peta dan menampilkan luas (m^2) dari bidang dalam peta tersebut.
5. Pada pengujian perbandingan *respone time* perangkat GPS dengan A-GPS, pengambilan titik koordinat dalam kondisi pengukuran pada saat cuaca mendung/hujan A-GPS bekerja lebih maksimal dengan rata-rata *respon time* 1.5 *second* dibandingkan perangkat GPS dengan rata-rata *respon time* 35.1 *second* ketika perangkat GPS dinyalakan.
6. Pada pengujian perbandingan ketepatan perangkat GPS dengan A-GPS dilakukan dengan menghitung rata-rata dari selisih jarak pada 50 titik koordinat yang diperoleh perangkat GPS dan A-GPS kurang lebih 4-5 Meter.

7. Hasil pengujian validasi dengan metode *Black-Box Testing* pada sistem menunjukkan nilai dengan prosentase 100%. Sistem sudah memenuhi spesifikasi kebutuhan yang telah dianalisa.
8. Berdasarkan dari hasil UAT, sistem telah dapat diterima oleh pengguna. Hasil yang diperoleh menunjukkan 97% tanggapan responden memberikan penilaian sangat bagus terhadap navigasi menu, *interface*, kinerja, dan kualitas keseluruhan aplikasi. Dan 3% tanggapan responden memberikan penilaian bagus terhadap ketepatan *Assited* GPS.

6.2. Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak ini antara lain :

1. Komponen – komponen perangkat lunak yang digunakan dalam *component-based sofware engineering* sebaiknya dipilih dengan cermat agar tidak terjadi permasalahan dalam proses pengembangan perangkat lunak.
2. Untuk pengembangan lebih lanjut perangkat lunak ini dapat dikembangkan dan digunakan pada semua *smartphone* yang telah tertanam GPS.
3. Untuk pengembangan lebih lanjut pada sistem informasi *administrator* mampu mengkonversi secara langsung koordinat-koordinat yang dikirim oleh perangkat lunak GPS *tracker* berbasis Android menjadi bidang-bidang pada sebuah peta yang dilengkapi informasi data klien pada tiap bidang dalam peta.
4. Untuk pengembangan lebih lanjut pada sistem informasi *administrator* dapat menjadi salah satu bagian dari *webservice company profile* Badan Pertanahan Nasional, sehingga dapat dilihat oleh masyarakat umum.

5. Untuk pengembangan lebih lanjut dapat dilakukan optimasi pada ketepatan *Assisted* GPS dengan membandingkan lagi antara beberapa perangkat *Asisted* GPS dengan perangkat GPS.
6. Untuk pengembangan lebih lanjut dapat dilakukan optimasi dari sisi keamanan jaringan agar pada saat pengiriman data dari perangkat lunak GPS *tracker* berbasis Android tidak disalahgunakan oleh pihak yang tidak bertanggungjawab.
7. Untuk pengembangan lebih lanjut dapat dilakukan pengambilan koordinat secara *real-time* pada aplikasi *administrator* dari perangkat lunak GPS *tracker* berbasis Android, sehingga secara langsung terbentuk bidang dalam peta.

DAFTAR PUSTAKA

[ADA-12] Adam H, Brata. 2012. *Pengembangan Perangkat Lunak Magic Profile Book Teknik Informatika Universitas Brawijaya Dengan Menggunakan Teknologi Augmented Reality*. Universitas Brawijaya Malang.

[AMR-11] Amri, Syaiful. 2011. *Membangun Sistem Navigasi Di Surabaya Menggunakan Google Maps Api*. Institut Teknologi Sepuluh Nopember Surabaya.

[AMA-09] Amadeo Global Perkasa. 2009. *Mengenal Teknologi Tracking System.* Akses dari http://www.gpstracker.co.id. Tanggal Akses : 30 April 2013.

[DHA-06] Dharwiyanti, Sri. 2006. *Pengantar Unified Modeling Language (UML).* Akses dari http://ilmukomputer.org/2006/08/25/pengantar-uml/. Tanggal Akses : 30 April 2013.

[DIG-09] Diggelen, Frank van. 2009. *A-GPS: Assisted* GPS, *GNSS, and SBAS*. Artech House.

[ELR-06] El-Rabbany, Ahmed. 2006. *Introduction to* GPS*: The Global Positioning System, second edition*. Norwood: Artech House.

[HER-11] Hermawan, Stephanus. 2011. *Mudah membuat aplikasi android*, Yogyakarta : Pemerbit ANDI.

[IBM-04] IBM Developer Works. 2004. *UML Basics: The Sequence Diagram*. Akses dari http://www.ibm.com/developerworks/rational/library/3101.html. Tanggal Akses : 2 Mei 2013.

[JAY-10] Jay A. Kreibich. 2012. *Using SQLite*, United Statesof Amerika: Publisher O'Reilly Media. ISBN: 978-0-596-52118-9.

[KAD-09] Kadir, Abdul. 2009. *Dasar Pemrograman Java™ 2,* Yogyakarta : Penerbit ANDI.

[MAX-13] Maxim, Bruce R. 2013. *Software Testing Strategies*. Akses dari http://www.learningace.com/doc/2027567/0b918c897e643de4e7447f663a111bc7/lec25. Tanggal Akses: 19 November 2013.

[OFF-13] Official U.S. Government information. 2013. *Global Positioning System (*GPS*) and related topics.* Akses dari http://www.gps.gov/systems/gps/. Tanggal Akses : 2 Mei 2013.

[PER-13] Permana, Budi. 2013. Pemrograman PHP: *Cepat Mahir Bahasa Pemrograman PHP*. Akses dari http://ilmukomputer.org/2013/01/22/cepat-mahir-bahasa-pemrograman-php/. Tanggal Akses: 30 April 2013.

[PRE-01] Pressman, Roger S. 2001. *Software Engineering : A Practitioner's Approach, Fifth Edition*. McGraw Hill.

[SAF-11] Safaat H, Nazaruddin. 2011. *Pemrograman Aplikasi Mobile Smartphone dan Tabet PC berbasis Android*, Bandung : Penerbit Informatika Bandung. ISBN : 978-602-8758-23-9.

[SOM-03] Sommerville, Ian. 2003. *Software Engineering (Rekayasa Perangkat Lunak) / Edisi 6 / Jilid 1*, Jakarta : Penerbit Erlangga.

[W3C-11] W3C. 2011. *Web Services Tutorial*. Akses dari http://www.w3schools.com/webservices/default.asp. Tanggal Akses : 30 April 2013.