SPATIAL DECISION SUPPORT SYSTEM (SDSS) UNTUK PERENCANAAN PEMBANGUNAN APOTEK BERBASIS ANDROID MENGGUNAKAN METODE *SIMPLE ADDITIVE WEIGHTING* (SAW)

SKRIPSI

oleh :

Sandi Prasekti

112410101071

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

SPATIAL DECISION SUPPORT SYSTEM (SDSS) UNTUK PERENCANAAN PEMBANGUNAN APOTEK BERBASIS ANDROID MENGGUNAKAN METODE *SIMPLE ADDITIVE WEIGHTING* (SAW)

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Pendidikan Sarjana (S1) Program Studi Sistem Informasi dan mencapai gelar Sarjana Komputer

oleh :

Sandi Prasekti

112410101071

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Alloh Subhanahu Wa Ta'ala;
2. Kedua Orangtua saya, Ayahanda Dariyono dan Ibunda Winarsih;
3. Saudara-saudaraku berserta seluruh keluarga besar;
4. Guru-guruku sejak taman kanak-kanak hingga perguruan tinggi;
5. Seluruh teman-teman yang selalu memberikan bantuan dan dukungan;
6. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTO

عَسَى أَنْ تَكْرَهُوا شَيْئًا وَهُوَ خَيْرٌ لَكُمْ وَعَسَى أَنْ تُحِبُّوا شَيْئًا وَهُوَ شَرٌّ لَكُمْ وَاللَّهُ يَعْلَمُ وَأَنْتُمْ لا تَعْلَمُونَ (٢١٦)

“Boleh jadi, kamu membenci sesuatu, padahal ia amat baik bagimu, dan boleh jadi (pula) kamu menyukai sesuatu, padahal ia amat buruk bagimu. Allah yang paling mengetahui, sedangkan kamu tidak mengetahui.” (QS. Al-Baqarah:216)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Sandi Prasekti

NIM : 112410101071

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "*Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)", adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, 16 Juni 2015

Yang menyatakan,

Sandi Prasekti
NIM. 112410101071

PENGESAHAN PEMBIMBING

Skripsi berjudul "***Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)**", telah diuji dan disahkan pada :

Hari, tanggal : Kamis, 27 Agustus 2015

Tempat : Program Studi Sistem Informasi Universitas Jember.

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Dr. Saiful Bukhori ST., M.Kom	Yanuar Nurdiansyah ST,.M.Cs.
NIP. 196811131994121001	NIP. 198201012010121004

SKRIPSI

***SPATIAL DECISION SUPPORT SYSTEM* (SDSS) UNTUK PERENCANAAN PEMBANGUNAN APOTEK BERBASIS ANDROID MENGGUNAKAN METODE *SIMPLE ADDITIVE WEIGHTING* (SAW)**

oleh :

Sandi Prasekti

112410101071

Pembimbing :

Dosen Pembimbing Utama : Dr. Saiful Bukhori ST., M.Kom

Dosen Pembimbing Pendamping : Yanuar Nurdiansyah ST,.M.Cs.

PENGESAHAN

Skripsi berjudul "***Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)**", telah diuji dan disahkan pada :

Hari, tanggal : Kamis, 27 Agustus 2015

Tempat : Program Studi Sistem Informasi Universitas Jember.

Tim Penguji :

Penguji I,	Penguji II,
Prof. Drs. Slamin, M.Comp.Sc.,Ph.D	Muhamad Arief Hidayat S.Kom,. M.Kom.
NIP 196704201992011001	NIP 198101232010121003

Mengesahkan
Ketua Program Studi,

Prof. Drs. Slamin, M.Comp.Sc.,Ph.D
NIP 196704201992011001

RINGKASAN

***Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)**; Sandi Prasekti, 112410101071; 2015; 101 halaman; Program Studi Sistem Informasi Universitas Jember.

Meraih kesuksesan suatu bisnis merupakan impian tiap-tiap orang. Banyak faktor yang mempengaruhi kesuksesan suatu bisnis. Salah satu faktor yang mempengaruhi berhasil atau tidaknya suatu bisnis adalah pemilihan tempat bisnis. Semakin strategis tempat untuk bisnis, maka peluang bisnis pun semakin besar. Begitu pula sebaliknya, tempat bisnis yang kurang strategis memiliki peluang bisnis yang lebih kecil. Maka dari itu, untuk menghindari salah satu hambatan bisnis apotek adalah memilih tempat bisnis apotek yang strategis. Dengan adanya sistem pendukung keputusan pemilihan lokasi pembangunan apotek, seseorang yang akan membangun apotek dapat melakukan proses pemilihan lokasi apotek dengan cepat dan tepat, serta mampu memberikan rekomendasi keputusan lokasi bisnis apotek terpilih secara lebih objektif. Sistem ini dibangun menggunakan bahasa pemrograman *JAVA* dan *Extensible Markup Language* (XML). Berdasarkan hasil pengujian terhadap sistem, SDSS dapat memberikan rekomendasi lokasi pembangunan Apotek yang strategis berdasarkan penilaian kelompok berupa perankingan kecamatan. Kecamatan yang mendapat ranking pertama merupakan kecamatan yang direkomendasikan untuk menjadi lokasi pembangunan apotek. Lokasi yang direkomendasikan akan ditampilkan pada peta.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "*Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Slamin, M.Comp.Sc., Ph.D., selaku Ketua Program Studi Sistem Informasi Universitas Jember;
2. Dr. Saiful Bukhori ST., M.Kom selaku Dosen Pembimbing Utama dan Yanuar Nurdiansyah ST,.M.Cs. selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi;
3. Dr. Saiful Bukhori, ST., M.Kom., selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember;
5. Teman-teman seperjuangan dan juga teman yang saya perjuangkan.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 16 Juni 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1. PENDAHULUAN

Bab ini menjelaskan latar belakang, perumusan masalah, tujuan dan manfaat, batasan masalah, dan sistematika penulisan.

1.1. Latar Belakang

Dunia bisnis atau wirausaha semakin berkembang dari waktu ke waktu. Berbagai macam jenis bisnis telah ada pada saat ini. Mulai dari bisnis kecil-kecilan sampai bisnis yang dapat meraih ratusan juta rupiah. Perkembangan ini disebabkan oleh perkembangan manusia dan teknologi yang semakin canggih.

Bisnis yang ada pada saat ini telah banyak. Salah satu bisnis yang ada atau dapat dilakukan pada saat ini adalah bisnis apotek. Apotek adalah salah satu sarana pelayanan kesehatan dalam membantu mewujudkan tercapainya derajat kesehatan yang optimal bagi masyarakat. Keberadaan apotek menjadi salah satu kebutuhan yang penting bagi masyarakat. Oleh karena itu, apotek merupakan salah satu peluang bisnis yang bagus. Tetapi, tampaknya sekarang ini telah banyak orang yang memiliki bisnis apotek. Menjamurnya keberadaan apotek dapat menjadi hambatan dalam kelancaran bisnis apotek.

Meraih kesuksesan suatu bisnis merupakan impian tiap-tiap orang. Banyak faktor yang mempengaruhi kesuksesan suatu bisnis. Salah satu faktor yang mempengaruhi berhasil atau tidaknya suatu bisnis adalah pemilihan tempat bisnis. Semakin strategis tempat untuk bisnis, makan peluang bisnis pun semakin besar. Begitu pula sebaliknya, tempat bisnis yang kurang strategis memiliki peluang bisnis yang lebih kecil. Maka dari itu, untuk menghindari salah satu hambatan bisnis apotek adalah memilih tempat bisnis apotek yang strategis.

Seringkali ketika memulai suatu bisnis orang tidak mempertimbangkan tempat bisnisnya. Kebanyakan hanya menghandalkan perkiraan saja dan tidak mempertimbangkan faktor-faktor penting dalam menentukan tempat bisnisnya. Sehingga kesuksesan bisnis yang diinginkan tidak dapat dicapai. Demikian juga halnya dengan bisnis apotek, dalam pembangunan apotek harus memperhatikan faktor-faktor penting dalam penentuan tempat apotek. Misalnya, Keberadaan apotek lain di tempat sekitar, dan kemudahan akses menuju apotek. Maka dalam mempertimbangkan pembangunan apotek akan lebih baik dengan bantuan sistem penunjang keputusan.

Sistem Pendukung Keputusan (SPK) digunakan sebagai alat bantu bagi para pengambil keputusan untuk memperluas kapabilitas para pengambil keputusan, namun tidak untuk menggantikan penilaian para pengambil keputusan (Turban, Aronso, & Liang, 2005). Dengan adanya sistem pendukung keputusan pemilihan lokasi pembangunan apotek, seseorang yang akan membangun apotek dapat melakukan proses pemilihan lokasi apotek dengan cepat dan tepat, serta mampu memberikan rekomendasi keputusan lokasi apotek terpilih secara lebih objektif. Dengan adanya sistem tersebut diharapkan lokasi pembangunan apotek yang terpilih benar-benar sesuai dengan yang diinginkan oleh seseorang yang akan membangun apotek.

Dalam penelitian ini, penulis menggunakan metode *Simple Additive Weighting* (SAW). Metode yang dipilih adalah metode *Simple Additive Weighting* (SAW) karena pemilihan lokasi pembangunan digolongkan ke dalam masalah yang bersifat *multicriteria* (ada banyak kriteria untuk mencapai tujuan). Melalui metode *Simple Additive Weighting* (SAW) dapat menentukan sendiri bobot kepentingan dari masing-masing kriteria. Metode *Simple Additive Weighting* (SAW) ini menentukan nilai bobot untuk setiap atribut, kemudian dilanjutkan dengan proses perankingan yang akan menyeleksi alternatif terbaik dari sejumlah alternatif, dalam hal ini alternatif yang dimaksud adalah lokasi pembangunan apotek yang memiliki kriteria sesuai

dengan yang diinginkan seseorang yang akan membangun apotek. Dengan metode perangkingan tersebut, diharapkan penilaian akan lebih tepat karena didasarkan pada nilai kriteria dan bobot yang sudah ditentukan sehingga akan mendapatkan hasil yang lebih akurat dan optimal terhadap lokasi pembangunan apotek terpilih yang akan dipertimbangkan oleh pengambil keputusan. Berdasarkan latar belakang diatas maka dalam tugas akhir ini penulis mengambil judul ”*Spatial Decision Support System* (SDSS) untuk Perencanaan Pembangunan Apotek Berbasis Android Menggunakan Metode *Simple Additive Weighting* (SAW)”.

1.2 Perumusan Masalah

Berdasarkan latar belakang diatas maka dirumuskan permasalahan sebagai berikut :

1. Bagaimana sistem dapat membantu dalam pengambilan keputusan perencanaan pembangunan apotek di Kabupaten Jember ?
2. Bagaimana menghasilkan keputusan dengan menggunakan *Simple Additive Weighting* (SAW) ?

1.3 Tujuan dan Manfaat

Tujuan dan manfaat dalam penelitian ini merupakan jawaban dari perumusan masalah yang telah disebutkan.

1.3.1. Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Sistem dapat membantu dalam pengambilan keputusan perencanaan pembangunan apotek di Kabupaten Jember.
2. Menghasilkan keputusan dengan menggunakan metode *Simple Additive Weighting* (SAW).

1.3.2. Manfaat

Manfaat diperoleh dari adanya ini adalah aplikasi sistem pendukung keputusan seleksi penerimaan program kreatifitas mahasisiswa Universitas Jember adalah sebagai berikut:

a. Manfaat Akademis

Hasil penelitian ini diharapkan dapat memberikan kontribusi dan masukan bagi siapa saja yang membutuhkan informasi yang berhubungan dengan judul penelitian ini.

b. Manfaat bagi peneliti

Mengetahui proses penerapan metode *Simple Additive Weighting* (SAW) pada *Spatial Decision Support System* (SDSS) untuk perencanaan pembangunan apotek.

c. Manfaat bagi objek penelitian

Membantu pemilik apotek mengembangkan usaha apoteknya dalam menentukan lokasi pembangunan apotek yang strategis.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Implementasi *Spatial Decision Support System* (SDSS) perencanaan penempatan apotek yang dibangun berbasis android.
2. Metode yang digunakan dalam pembuatan sistem adalah metode *Simple Additive Weighting* (SAW).
3. Wilayah yang tercakup hanya wilayah kecamatan yang berada di kabupaten Jember.

1.5. Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Pendahuluan

   Bab ini terdiri atas latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah dan sistematika penulisan.

2. Tinjauan Pustaka

   Bab ini berisi tentang kajian materi, penelitian terdahulu dan informasi apa saja yang digunakan dalam penelitian ini. Dimulai dari kajian pustaka mengenai pengertian dari sistem informasi hingga metode *Simple Additive Weighting* (SAW).

3. Metodologi Penelitian

   Bab ini menguraikan tentang metode apa yang dilakukan selama penelitian. Dimulai dari tahap pencarian permasalahan hingga pengujian aplikasi yang akan dibuat.

4. Analisis dan Pengembangan Sistem

   Bab ini menjelaskan penerapan desain dan pengembangan sistem, menggunakan model pengembangan *waterfall*, meliputi analisis kebutuhan fungsional dan *non*-fungsional sistem, dilanjutkan dengan pembuatan *usecase diagram*, skenario, *activity diagram*, *sequence diagram*, *class diagram dan entity relation diagram* (ERD).

5. Hasil dan Pembahasan

   Bab ini menjelaskan tentang hasil dan pembahasan dari penelitian yang telah dilakukan. Dengan memaparkan hasil penelitian dan hasil percobaan pengimplementasian sistem.

6. Penutup

   Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Dalam penelitian pengimplementasian *Simple Additive Weighting* (SAW) dalam sistem informasi dan rekomendasi pemilihan lokasi pembangunan apotek, dibutuhkan beberapa landasan teori yang digunakan untuk memperkuat dan mengarahkan penelitian agar tidak keluar dari kaidah keilmuan yang ada.

2.1 Penelitian Terdahulu

Penelitian yang berjudul "*MODEL PETA DIGITAL RAWAN SAMBARAN PETIR DENGAN MENGGUNAKAN METODE SAW (SIMPLE ADDITIVE WEIGHTING) : STUDI KASUS PROPINSI LAMPUNG"* dilakukan oleh Sugiyono yang merupakan mahasiswa Magister Ilmu Komputer Program Pascasarjana Universitas Budi Luhur. Penelitian ini mengimplementasikan metode *SIMPLE ADDITIVE WEGHTING* (SAW) untuk menganalisa daerah-daerah yang rawan sambaran petir di provinsi Lampung. Peta digital yang dibuat tersebut dapat menginformasikan daerah-daerah yang mempunyai tingkat kerawanan sambaran petir yang tinggi sehingga masyarakat maupun pihak-pihak terkait bisa melakukan usaha mengurangi dampak dan kerugian akibat sambaran petir.

2.2 Apotek

2.2.1 Pengertian

Menurut KepMenKesNo.1027/MENKES/SK/IX/2004, apotek adalah tempat tertentu tempat dilakukan pekerjaan kefarmasian dan penyaluran sediaan kefarmasian, perbekalan kesehatan lainnya kepada masyarakat.

Definisi apotek menurut PP 51 Tahun 2009. Apotek merupakan suatu tempat atau terminal distribusi obat perbekalan farmasi yang dikelola oleh apoteker sesuai standar dan etika kefarmasian.

Berdasarkan dari definisi di atas, dapat diketahui bahwa apotek merupakan salah satu sarana pelayanan kesehatan dalam membantu mewujudkan tercapainya

derajat kesehatan yang optimal bagi masyarakat, selain itu juga sebagai salah satu tempat pengabdian dan praktek profesi apoteker dalam melakukan pekerjaan kefarmasian.

2.2.2 Tugas dan Fungsi

Menurut Syamsuni (2006) Apotek memiliki tugas dan fungsi sebagai

1. Tempat pengabdian profesi apoteker yang telah mengucapkan sumpah jabatan.
2. Sarana farmasi untuk melaksanakan peracikan, pengubahan bentuk, pencampuran, dan penyerahan obat atau bahan obat.
3. Sarana penyaluran perbekalan farmasi dalam menyebarkan obat-obatan yang diperlukan masyarakat secara luas dan rata.

2.3 Android

Android merupakan sebuah sistem operasi telepon selular dan komputer tablet layar sentuh yang berbasis linux. Namun seiring perkembangannya, Android berubah menjadi platform yang begitu cepat dalam melakukan inovasi. Hal ini tidak lepas dari pengembang utama dibelakangnya yaitu Google. Google membuatkan sebuah *platform*. *Platform* android terdiri dari sistem operasi berbasis linux, sebuah *Graphic User Interface* (GUI), sebuah web browser dan aplikasi yang dapat di download dan juga para pengembang bisa dengan leluasa berkarya menciptakan aplikasi yang baik dan dapat untuk digunakan pada berbagai macam perangkat. (Kasman, 2013).

Android adalah sistem operasi yang mengadopsi sistem operasi linux, namun telah dimodifikasi. Android diambil oleh google pada tahun 2005 dari Android sebagai bagian strategi untuk mengisi pasar sistem operasi bergerak. Google mengambil alih seluruh hasil kerja Android termasuk tim yang mengembangkan android. (Agustina, 2012).

Model pengembangannya yang sederhana membuat android menarik bagi *vendor – vendor* perangkat keras. Keuntungan utama dari android adalah andanya pendekatan aplikasi secara terpadu. Pengembang hanya berkonsentrasi pada aplikasi saja, aplikasi tersebut bisa berjalan pada beberapa perangkat yang berbeda selama masih ditenagai oleh android dimana pengembang tidak perlu mempertimbangkan kebutuhan jenis perangkatnya (Supriyanto, 2012 : 9). Jadi, aplikasi yang dibuat berbasis android saat ini memiliki banyak keuntungan dan selain itu android kini menjadi *trend* saat ini di dunia teknologi.

2.4 Geographic Information System (GIS)

Sistem Informasi Geografis (SIG) atau lebih terkenal dengan istilah *Geographical Information Sistem (GIS)* sebagai salah satu penerapan di bidang teknologi informasi yang mampu mendapatkan gambaran atau informasi ruang muka bumi, mengelola data-data spasial atau tata ruang, dan mampu menghasilkan informasi dalam bentuk peta digital.

Menurut Burrough (dalam Indarto, 2010: 3) Sistem Informasi Geografis (SIG) didefinisikan sebagai suatu alat / media untuk memasukan, menyimpan, mmengambil, memanipulasi, menganalisa, dan menampilkan data-data beratribut Geografis (data spasial) yang berguna untuk mendukung proses pengambilan keputusan dalam perencanaan dan manajemen sumber daya alam, lingkungan transportasi, masalah perkotaan dan administratif.

Menurut oleh Arronoff (dalam Indarto, 2010: 3), yaitu sebuah sistem komputer yang menyediakan empat kemampuan utama untuk menangani data yang telah tergeoreferensi, meliputi: proses pemasukan data, manajemen data (menyimpan dan pemanggilan kembali), manipulasi dan analisis data, dan proses penyajian data (keluaran).

Pada tahun 1999 Murai (dalam Indarto, 2010: 7) mengatakan bahwa sistem informasi Geografis merupakan sistem informasi yang dapat mengelola mengenai

data ruang. Data ruang atau data spasial merupakan data yang dapat merepresentasikan keadaan geografis, sehingga apa yang ada pada kenyataan dapat kita lihat pada sistem informasi tanpa ada perbedaan.

Terdapat beberapa alasan yang mendasari mengapa perlu menggunakan GIS, menurut Indarto (2010) alasan yang mendasarinya adalah :

1. GIS menggunakan data spasial maupun atribut secara terintegrasi
2. GIS dapat memisahkan antara bentuk presentasi dan basis data
3. GIS memiliki kemampuan menguraikan unsur – unsur yang ada dipermukaan bumi ke dalam beberapa *layer* atau *coverage* data spasial.
4. GIS memiliki kemampuan yang sangat baik dalam memvisualisasikan data spasial berikut atributnya.
5. Semua operasi GIS dapat dilakukan interaktif.
6. GIS dengan mudah menghasilkan peta – peta tematik.
7. GIS sangat membantu pekerjaan yang erat kaitanya dengan bidang spasial dan geoinformatika.

Beberapa contoh aplikasi SIG menurut Luthfi (2009, dalam Sistem Informasi Geografi Pengertian dan Pemanfaatannya), yaitu :

1. Pengelolaan Fasilitas: Peta skala besar, *network analysis*, biasanya digunakan untuk pengelolaan fasilitas kota. Contoh aplikasinya adalah penempatan pipa dan kabel bawah tanah, perencanaan fasilitas perawatan, pelayanan jaringan telekomunikasi.
2. Pengelolaan Sumber Daya Alam dan Lingkungan : Untuk tujuan ini pada umumnya digunakan citra satelit, citra Landsat yang digabungkan dengan foto udara, dengan teknik *overlay*. Contoh aplikasinya adalah studi kelayakan untuk tanaman pertanian, pengelolaan hutan dan analisis dampak lingkungan.

3. Bidang Transportasi : Untuk fungsi ini digunakan peta skala besar dan menengah, dan analisis keruangan, terutama untuk manajemen transit perencanaan rute, pengiriman teknisi, analisa pelayanan, penanganan pemasaran dan sebagainya.
4. Jaringan telekomunikasi : SIG digunakan untuk memetakan Sentral, *Main Distribution Poin* (MDF), kabel primer, Rumah Kabel, kabel Sekunder, Daerah Catu Langsung dan seterusnya sampai ke pelanggan. Dengan SIG kerusakan yang terjadi dapat segera diketahui.
5. Sistem Informasi Lahan : Untuk keperluan ini yang digunakan adalah peta kadastral skala besar atau peta persil tanah dan analisis keruangan untuk informasi kadatral pajak.

2.5 *Decision Support Systems* (DSS)

Menurut Turban (2005) dalam Setiabudi S (Hasan, 2002) Sistem Penunjang Keputusan / *Decision Support Systems* (DSS) merupakan sekumpulan prosedur berbasis model untuk data pemrosesan dan penilaian guna membantu para manajer mengambil keputusan.

Menurut Hasan (2002) Konsep Sistem Pendukung Keputusan (SPK) ditandai dengan sistem interaktif berbasis komputer yang membantu pengambilan keputusan memanfaatkan data dan model untuk menyelesaikan masalah yang tidak terstruktur. Pada dasarnya Sistem Pendukung Keputusan (SPK) dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasikan masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam proses pengambilan keputusan, sampai mgevaluasi pemilihan.

Dapat disimpulkan bahwa sistem pendukung keputusan merupakan sebuah sistem yang digunakan untuk membantu menyelesaikan masalah yang tidak terstruktur agar lebih efektif dan dapat memecahkan masalah semi terstruktur.

Menurut Peter G.W Keen dan Scott Morton dalam (Haris, 2011) ada tiga tujuan yang harus dicapai oleh sistem penunjang keputusan, yaitu :

1. Membantu manajer membuat keputusan untuk memecahkan masalah semi terstruktur,
2. Mendukung penilaian manajer buka mencoba untuk menggantikannya,
3. Meningkatkan efektifitas pengambilan keputusan manajer dari pada efisiensinya.

*Multi Atribute Decision Making (*MADM) merupakan metode pengambilan keputusan dalam menetapkan alternatif terbaik dari sejumlah alternatif yang telah ada berdasarkan beberapa kriteria tertentu (Kusumadewi, 2006). Kriteria tersebut merupak ukuran-ukuran, aturan-aturan serta standar yang digunakan dalam pengambilan keputusan. Secara umum MADM menyeleksi alternatif terbaik dari sejumlah alternatif. Menurut Janko dalam Kusumadewi (2006), ada beberapa fitur yang digunakan dalam MADM, yakni :

1. Alternatif, merupakan beberapa obyek berbeda yang memiliki kesempatan sama untuk dipilih oleh pengambil keputusan.
2. Atribut, sering disebut juga kriteria atau karakteristik.
3. Konflik antar kriteria, antar kriteria satu dengan yang lainnya biasanya muncul konflik. Seperti antara keuntungan dengan biaya.
4. Bobot keputusan, pada MADM dicari bobot kepentingan dari setiap kriteria guna menunjukkan kepentingan relatif dari setiap kriteria. W = (w1,w2,w3,....).
5. Matriks keputusan, suatu matriks keputusan X yang berukuran m*n, berisi elemen-elemen x_{ij}, yang merepresentasikan rating dari alternatif A_i (i=1,2,….,m) terhadap kriteria C_j(j=1,2,….,n).

MADM dapat diselesaikan dengan beberapa metode, yakni *Simple Additive Weighting Method*e (SAW), *Weighted Product Methode* (WP), ELECTRE, *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS), *Analytic Hierarchy Process* (AHP).

2.5.1 *Simple Additive Weighting* (SAW)

Menurut Fishburn (dalam Kusumadewi, 2006: 74), Metode SAW sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Rumusan normalisasi matrik terlhat pada persamaan (i).

$$r_{ij} = \begin{cases} \frac{X_{ij}}{Max\ X_{ij}} & (Benefit) \\ \frac{Min\ X_{ij}}{Xij} & (Cost) \end{cases} \quad (1)$$

dimana r_{ij} adalah rating kinerja ternormalisasi dari alternatif A_i pada atribut C_j; i=1,2,...,m dan j=1,2,...,n. Nilai preferensi untuk setiap alternative (V_i) terlihat pada persamaan (ii):

$$V_i = \sum_{j=1}^{n} w_j r_{ij} \quad (2)$$

Nilai V_i yang lebih besar mengindikasikan bahwa alternatif A_i lebih terpilih. (Kusumadewi, 2006).

2.6 *SPATIAL DECISION SUPPORT SYSTEM* (SDSS)

Sistem berbasis komputer interaktif yang dirancang untuk mendukung pengguna dari kelompok pengguna dalam mencapai efektivitas yang lebih tinggi dari pengambilan keputusan saat penyelesaian masalah keputusan spasial (Malczewski,1999)

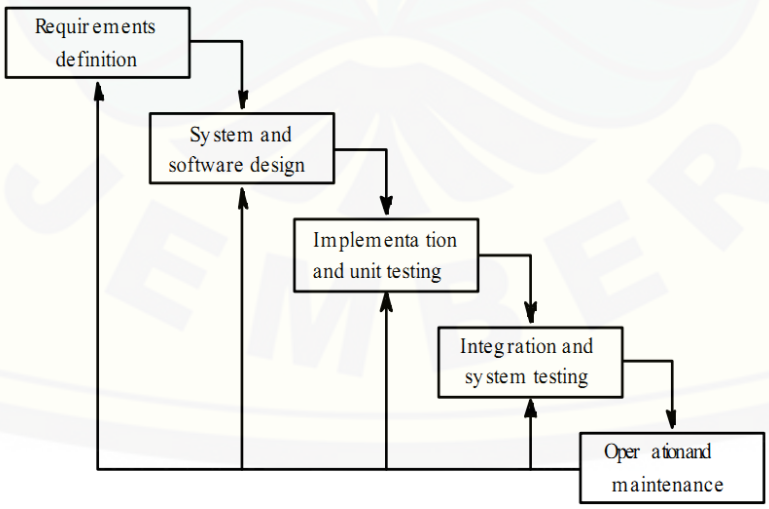
Lingkungan terpadu yang memanfaatkan database yang kedua model spasial dan non-spasial, alat pendukung keputusan seperti sistem pakar, paket statitical, paket

optimasi, dan grafis ditingkatkan untuk menawarkan para pengambil keputusan paradigma baru untuk analisis dan pemecahan masalah (Leipnik, 1993)

Berdasarkan uraian diatas dapat disimpulkan bahwa SDSS adalah sistem komputer yang terintegrasi antara *Decision Support System* (DSS) dengan *Geographic Information Systems* (GIS) untuk menyelesaikan suatu masalah tertentu. *Decision Support System* (DSS) memiliki peranan penting untuk membantu dalam membantu keputusan yang berkaitan dengan masalah spatial maupun non-spatial. Sedangkan *Geographic Information Systems* (GIS) memiliki peranan penting untuk membantu dalam menampilkan informasi hasil dari keputusan yang telah dibuat dalam bentuk peta.

2.7 Model *Waterfall*

Model SDCL air terjun (waterfall) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian, dan tahap pendukung (*support*). Ilustrasi model *waterfall* dapat dilihat pada gambar 2.1.



Gambar 2.1 Ilustrasi Model *Waterfall*

Keterangan dari skema gambar 2.1 adalah :

1. Analisis Kebutuhan

Menganalisis kebutuhan yang akan digunakan dalam pembuatan aplikasi. Meliputi pengumpulan data kebutuhan fungsional dan *non*-fungsional dari aplikasi yang akan kita bangun. Setelah itu, menentukan fungsi dan fasilitas apa saja yang akan dibuat dalam aplikasi.

2. Desain Sistem

Jika proses analisis kebutuhan telah diketahui maka proses selanjutnya adalah pada tahapan desain sistem. Proses pendesainan sistem dari aplikasi yang akan kita bangun yaitu dengan menggunakan *Unified Modeling Language* (UML). Penggunaan UML karena sudah menggunakan konsep *Object Oriented Design* yang tentunya akan sangat memudahkan developer untuk membangun sebuah sistem. Dalam UML ada beberapa diagram yang akan dibuat antara lain:

a. Business Process

Business Proses digunakan untuk menggambarkan inputan data yang dibutuhkan sistem, *output* dari sistem serta tujuan dari pembuatan sistem.

b. Use Case Diagram

Use case adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor.

c. Scenario

Scenario diagram digunakan untuk menjelaskan atau menceritakan fitur atau isi yang ada di *use case* diagram. *Scenario* menjelaskan alur sistem dan keadaan yang akan terjadi ketika terjadi suatu event tertentu.

d. Squence Diagram

Sequence diagram (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, *display*, dan sebagainya berupa pesan.

e. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

f. Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.

g. Entity Relationship Diagram

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

3. *Coding* (Pengkodean)

Setelah proses desain sistem dikerjakan, proses selanjutnya adalah *coding* atau penulisan kode program. Bahasa pemrograman yang dipakai adalah *java* dan *Extensible Markup Language* (XML) dengan *tool* yang digunakan *eclipse* dan menggunakan emulator *genymotion*.

4. Pengujian / Testing

Pengujian wajib dilakukan untuk menguji apakah sistem ini sudah sesuai dengan kebutuhan dari user atau belum. Dan apakah masih ada kesalahan maupun kelemahan terhadap sistem yang kami bangun tersebut. Diharapkan proses pengujian / testing dapat menyempurnakan sistem yang kami buat. Pengujian yang dilakukan melibatkan semua aspek sistem meliputi *hardware*, *software* aplikasi, *environment software*, penempatan aplikasi, dan *user* yang menggunakan aplikasi ini. Pengujian perangkat lunak menggunakan dua metode yakni :

1. *Black Box Testing*

Terfokus pada apakah unit program memenuhu kebutuhan (requirement) yang disebutkan dalam spesifikasi. Pada *black box testing*, cara pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah

hasil dari unit itu sesuai dengan proses bisnis yang diinginkan. Jika ada unit yang tidak sesusai outputnya maka untuk menyelesaikannya, diteruskan pada pengujian yang kedua (Fatta, 2007). Tabel pengujian *black box* dapat dilihat pada tabel 2.1.

Tabel 2.1 Tabel Pengujian *Black Box*

Kelas Uji	Skenario Uji	Hal yang diharapkan	Kesimpulan

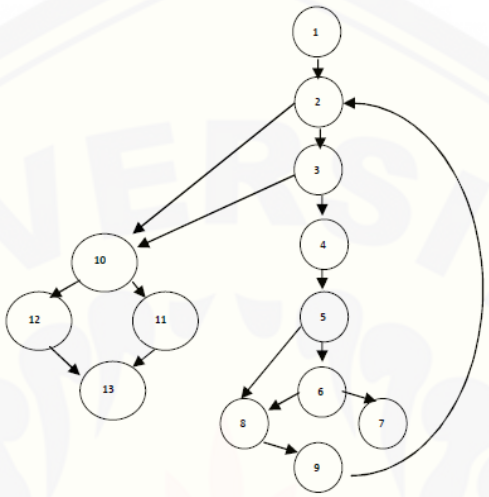
2. *White Box Testing*

White box testing adalah cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada model yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang (Fatta, 2007).

Pengujian *white box* menggunakan metode pengujian berbasis *path.* Pengujian *basis-path* adalah teknik pengujian *white-box* yang diusulkan pertama kali oleh Tom McCabe. Metode basis ini memungkinkan desainer *test case* mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. *Test case* yang dilakukan untuk menggunakan *basis set* tersebut dijamin menggunakan setiap *statement* di dalam program paling tidak sekali selama pengujian. (Beizer, 1990)

Langkah-Iangkah pembuatan *test case* adalah sebagai berikut:

1. Dengan mempergunakan perancangan prosedural atau program sumber sebagai dasar digambarkan diagram alirnya seperti gambar 2.2.



Gambar 2.2 Diagram Alir prosedur rata

2. Menentukan kompleksitas siklomatis untuk diagram alir yang telah dibuat:

$$V(G) = E - N + 2 \quad (3)$$

Keterangan :

V(G) = Kompleksitas Siklomatis.

E = Jumlah Edge

N = Jumlah Node

Hasil perhitungan kompleksitas siklomatis dari diagram alir pada gambar 7 adalah sebagai berikut :

V(G) = 6 region .

V(G) = 17 *edge* - 13 *node* + 2 = 6

V(G) = 5 *node* predikat + 1 = 6

3. Menentukan *path* independen pada diagram alir

Dari hasil perhitungan kompleksitas siklomatis dari diagram alir pada gambar 3.4, terdapat 6 *path* independen yaitu:

path 1 : 1-2-10-11-13

path 2 : 1-2-10-12-13

path 3 : 1-2-3-10-11-13

path 4 : 1-2-3-4-5-8-9-2-..

path 5 : 1-2-3-4-5-6-8-9-2-..

path 6 : 1-2-3-4-5-6-7-8-9-2-...

4. Membuat *test case* yang akan mengerjakan masing-masing *path* pada basis set.

Data yang dipilih harus tepat sehingga setiap kondisi dari *node* predikat dikerjakan semua.

5. Maintenance

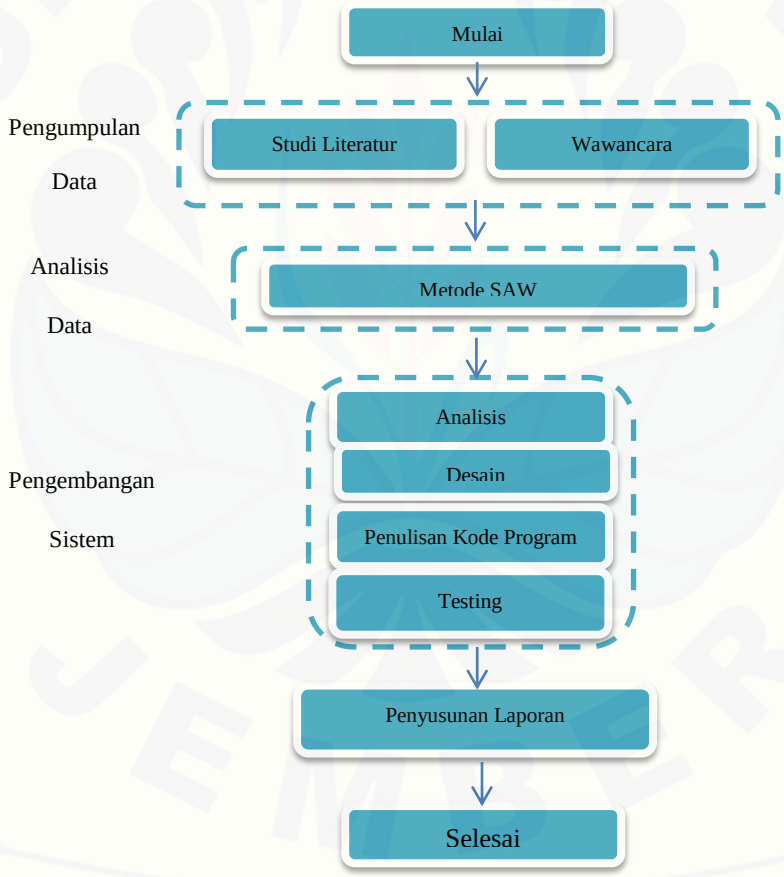
Perawatan diadakan untuk mengatasi masalah pada sistem dilain waktu ketika aplikasi sudah dapat digunakan oleh *user*. Selama *user* menemui *bug* pada aplikasi ini, maka *user* langsung dapat mengkonfirmasi kepada *developer* untuk segera ditangani oleh *developer*.

BAB 3. METODOLOGI PENELITIAN

Pada bab ini dijelaskan tentang metode-metode yang digunakan selama penelitian dilakukan, seperti jenis penelitian, studi literatur, data dan sumber data penelitian, serta tahapan analisis hingga model perancangan sistem.

3.1 Tahapan Penelitian

Penelitian akan dilaksanakan dalam beberapa tahap, tahapan penelitian digambarkan dalam bentuk *flow chart* diagram seperti yang terlihat pada gambar 3.1.



Gambar 3.1 *Flowchart* Tahapan Penelitian

3.2 Objek Penelitian

Dalam penelitian ini penulis melakukan wawancara secara langsung pada Apotek Median Farma yang berada di Kecamatan Wuluhan. Wawancara dilakukan untuk pengumpulan data kriteria yang cocok untuk menentukan lokasi pembangunan Apotek.

3.3 Teknik Pengumpulan Data

Proses untuk mendapatkan data yang dibutuhkan untuk membangun aplikasi perencanaan pembangunan apotek menggunakan dua cara, yaitu studi literatur dan wawancara.

4.3.1 Studi Literatur

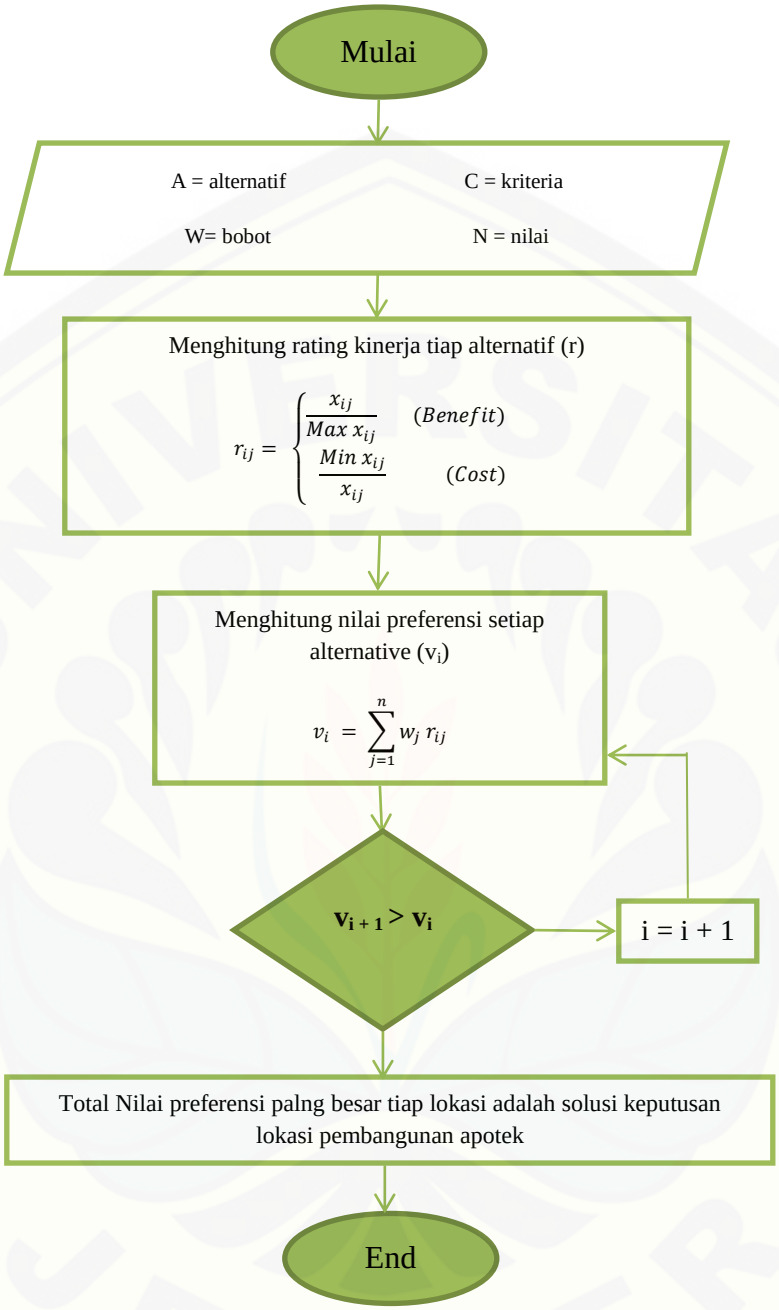
Studi literatur bertujuan untuk menyusun dasar teori yang digunakan dalam penelitian. Sumber yang dapat digunakan sebagai studi literatur antara lain buku, jurnal, karya ilmiah, dan situs web.

4.3.2 Wawancara

Salah satu cara untuk mengumpulkan data adalah melakukan wawancara dengan menanyakan langsung kepada narasumber. Pada penelitian ini penulis melakukan wawancara dengan beberapa pihak terkait untuk memeperoleh data yang dibutuhkan.

3.4 Tahap Analisis

Tahap analisis dilakukan setelah melakukan pengumpulan data kriteria. Data yang diperoleh akan dianalisa dengan metode SAW. Proses pengambilan keputusan dengan menggunakan metode SAW dapat dilihat pada gambar 3.2.



Gambar 3.2 *Flowchart* Metode SAW

Gambar 3.2 adalah gambar *flowchart* metode SAW. Gambar tersebut menjelaskan alur perhitungan metode SAW dan menghasilkan sebuah keputusan pembangunan lokasi apotek yang strategis. Langkah pertama menentukan alternatif atau rencana lokasi pembangunan apotek, kriteria dan bobot yang akan digunakan sebagai acuan atau pertimbangan dalam membangun sebuah apotek. Kemudian memberikan penilaian kriteria pada tiap alternatif. Hasil penilaian dinormalisasi dengan menggunakan perhitungan rating kinerja (r). Perhitungan rating kinerja (r) dengan mengelompokan menjadi 2 yaitu *benefit* atau *cost.* Setehah itu menghitung total nilai preferensi (v) pada tiap alternatif.. Total nilai preferensi (v) yang paling besar adalah keputusan lokasi pembangunan apotek.

3.5 Tahap Pengembangan Sistem

Didalam pembuatan aplikasi sistem penunjang keputusan ini mengikuti tahapan *software development life cycle* (SDLC) *waterfall*. Penggunaan SDLC *waterfall* karena sistem ini tergolong sistem bersekala kecil. Tahapan SDLC dengan metode *waterfall* meliputi tahapan analisis, desain, implementasi, pengujian, dan pemeliharaan.

Setelah tahap pengumpulan data selesai, selanjutnya data akan dianilisis menggunakan metode SAW. Kemudian akan dilanjutkan ke perancangan sistem dengan menggunakan konsep berbasis objek dengan pemodelan *Unified Modelling Language* (UML). Pemodelan UML yang digunakan pada penelitian ini antara lain, *Business Process*, *Usecase Diagram*, *Scenario*, *Sequence Diagram*, *Activity Diagram*, *Class diagram* dan *Entity Relationship Diagram* (ERD). Setelah tahap perancangan selesai, dilanjutkan dengan tahap implementasi menggunakan bahasa pemrograman *Java* dan *Extensible Markup Language* (XML). Hasil perancangan dan implementasi kemudian akan ditesting menggunakan *White Box* dan *Black Box*.

BAB 4. ANALISIS DAN PENGEMBANGAN SISTEM

Bab ini menjelaskan penerapan desain dan pengembangan sistem, menggunakan model pengembangan *waterfall*, meliputi analisis kebutuhan fungsional dan *non*-fungsional sistem, dilanjutkan dengan pembuatan *usecase diagram*, skenario, *activity diagram*, *sequence diagram*, *class diagram dan entity relation diagram* (ERD).

4.1 Pengumpulan Data

Melalui wawancara kepada pemilik apotek median farma, didapatkan data kriteria dalam mendirikan sebuah apotek. Data kriteria yang diperoleh dari hasil wawancara adalah sebagai tabel 4.1.

Tabel 4.1 Tabel data kriteria

No	Kriteria	Bobot	Kategori
1	Kepadatan Penduduk (C1)	90	Benefit
2	Keberadaan Usaha Sejenis (C2)	90	Cost
3	Jarak antar Usaha Sejenis (C3)	85	Benefit
4	Keberadaan Layanan Kesehatan (puskesmas, klinik dan rumah sakit) (C4)	70	Benefit
5	Tingkat Kemudahan Akses	85	Benefit

4.2 Penerapan Metode *Simple Additive Weighting* (SAW)

Metode analisis yang digunakan untuk perangkingan pada sistem ini adalah metode SAW. Untuk lebih jelasnya perhitungan SAW dapat dilihat pada gambar 3.2.

1. Menentukan alternatif atau rencana lokasi pembangunan

Jember merupakan kabupaten yang berada di provinsi Jawa Timur. Jember memiliki 31 kecamatan yaitu seperti yang ada pada tabel 4.2. Pada pembahasan ini alternatif yang dipilih adalah lokasi Ambulu, Balung dan Jenggawah seperti pada tabel 4.3.

Tabel 4.2 Kecamatan di Kabupaten Jember

Kecamatan	
No	Nama
1	Ambulu
2	Ajung
3	Arjasa
4	Balung
5	Bangsalsari
6	Gumukmas
7	Jelbuk
8	Jenggawah
9	Jombang
10	Kalisat
11	Kaliwates
12	Kencong
13	Ledokombo
14	Mayang
15	Mumbulsari
16	Pakusari

Kecamatan	
No	Nama
17	Panti
18	Patrang
19	Puger
20	Rambipuji
21	Semboro
22	Silo
23	Sukorambi
24	Sukowono
25	Sumberbaru
26	Sumberjambe
27	Sumbersari
28	Tanggul
29	Tempurejo
30	Umbulsari
31	Wuluhan

Tabel 4.3 Alternatif

Lokasi	
No	Nama
1	Ambulu
4	Balung
8	Jenggawah

Tabel 4.3 merupakan tabel alternatif. Seseorang yang ingin mendirikan apotek atau cabang apotek memiliki alternatif atau rencana sasaran lokasi yang akan menjadi tempat pendirian apoteknya.

2. Menentukan Data Kriteria

Data kriteria didapat dari hasil wawancara, seperti pada tabel 4.1 meliputi kriteria, bobot dan kategori.

3. Melakukan penilaian

Tabel 4.4 Data Hasil Penilaian

Penilaian			
No	No Lokasi	Kriteria	Nilai
1	1	C1	2
2	1	C2	1
3	1	C3	3
4	1	C4	3
5	1	C5	1
6	4	C1	4
7	4	C2	1
8	4	C3	2

Penilaian			
No	No Lokasi	Kriteria	Nilai
9	4	C4	3
10	4	C5	4
11	8	C1	5
12	8	C2	3
13	8	C3	2
14	8	C4	1
15	8	C5	2

Tabel 4.5 adalah tabel hasil data penilaian. Tabel tersebut berisi nilai tiap kriteria pada tiap lokasi.

4. Membuat tabel matrik (A, C), dimana A adalah alternatif dan C adalah kriteria

Tabel 4.5 Matrik Hasil Penilaian

	Kriteria				
Lokasi	C1	C2	C3	C4	C5
1	2	1	3	3	1
4	1	4	1	2	3
8	5	3	2	1	2

Tabel 4.6 adalah tabel matrik hasil penilaian. Tabel tersebut merupakan tabel data penilaian yang dibentuk matrik dua dimensi. Dibentuk sebuah matrik karena agar lebih mudah dalam membaca dan menerapkan ke sistem yang akan dibuat.

5. Menghitung rating kinerja (r)

Tabel 4.6 Matrik Hasil Rating Kinerja

Lokasi	Kriteria				
	C1	C2	C3	C4	C5
1	0.40	1.00	1.00	1.00	0.33
4	0.20	0.25	0.33	0.67	1.00
8	1.00	0.33	0.67	0.33	0.67

Tabel 4.7 adalah tabel matrik hasil rating kinerja. Pada tahap ini dilakukan perhitungan rating kinerja pada masing-masing nilai. Perhitungan rating kinerja (r) dengan menggolongkan kategori *benefit* atau *cost*. Hasil pada tabel tersebut diperoleh dengan perhitungan berikut :

Matrik Rating Kinerja (C, Lokasi)

C1 adalah *benefit* → $r_{ij} = \frac{x_{ij}}{Max\ x_{ij}}$

(C1, 1) → $r_{c11} = \frac{x_{11}}{Max\ x_{11}} = \frac{2}{5} = 0{,}40$

(C1, 2) → $r_{c12} = \frac{x_{11}}{Max\ x_{11}} = \frac{1}{5} = 0{,}20$

(C1, 3) → $r_{c13} = \frac{x_{11}}{Max\ x_{11}} = \frac{5}{5} = 1{,}00$

C1 adalah *cost* → $r_{ij} = \frac{Min\ x_{ij}}{x_{ij}}$

(C2, 1) → $r_{c21} = \frac{Min\ x_{21}}{x_{21}} = \frac{1}{1} = 1{,}00$

(C2, 2) → $r_{c22} = \frac{Min\ x_{22}}{x_{22}} = \frac{1}{4} = 0{,}25$

(C2, 3) → $r_{c23} = \frac{Min\ x_{23}}{x_{23}} = \frac{1}{3} = 0{,}33$

C1 adalah *benefit* → $r_{ij} = \frac{x_{ij}}{Max\ x_{ij}}$

(C3, 1) → $r_{c31} = \frac{x_{31}}{Max\ x_{31}} = \frac{3}{3} = 1{,}00$

(C3, 2) → $r_{c32} = \frac{x_{32}}{Max\ x_{32}} = \frac{1}{3} = 0{,}33$

(C3, 3) → $r_{c33} = \frac{x_{33}}{Max\ x_{33}} = \frac{2}{3} = 0{,}67$

C1 adalah *benefit* → $r_{ij} = \frac{x_{ij}}{Max\ x_{ij}}$

(C4, 1) → $r_{c41} = \frac{x_{41}}{Max\ x_{41}} = \frac{3}{3} = 1{,}00$

(C4, 2) → $r_{c42} = \frac{x_{42}}{Max\ x_{42}} = \frac{2}{3} = 0{,}67$

(C4, 3) → $r_{c43} = \frac{x_{43}}{Max\ x_{43}} = \frac{1}{3} = 0{,}33$

C1 adalah *benefit* → $r_{ij} = \frac{x_{ij}}{Max\ x_{ij}}$

(C5, 1) → $r_{c51} = \frac{x_{51}}{Max\ x_{51}} = \frac{1}{3} = 0{,}33$

(C5, 2) → $r_{c52} = \frac{x_{52}}{Max\ x_{52}} = \frac{2}{3} = 0{,}67$

(C5, 3) → $r_{c53} = \frac{x_{53}}{Max\ x_{53}} = \frac{3}{3} = 1{,}00$

6. Menghitung total nilai preferensi pada tiap alternatif

Tabel 4.7 Matrik Hasil Nilai Preferensi

Lokasi	Kriteria					Total
	C1	C2	C3	C4	C5	
1	36.00	90.00	85.00	70.00	28.33	309.33
4	18.00	22.50	28.33	46.67	85.00	200.50
8	90.00	30.00	56.67	23.33	56.67	256.67

Tabel 4.8 merupakan tabel matrik hasil nilai preferensi. Hasil nilai-nilai tersebut didapatkan dari hasil matrik rating kinerja yang dikalikan dengan tiap bobot dari kriteria masing-masing. Kemudian dijumlah pada masing-masing alternatif. Hasil pada tabel tersebut diperoleh dengan perhitungan berikut :

Nilai Prefensi $v_i = \sum_{j=1}^{n} w_j \, r_{ij}$

$$v_{c11} = w_{c1} r_1 = 90 \; x \; 0{,}40 = 36{,}00$$
$$v_{c12} = w_{c1} r_2 = 90 \; x \; 0{,}20 = 18{,}00$$
$$v_{c13} = w_{c1} r_3 = 90 \; x \; 1{,}00 = 90{,}00$$
$$v_{c21} = w_{c2} r_1 = 90 \; x \; 1{,}00 = 90{,}00$$
$$v_{c22} = w_{c2} r_2 = 90 \; x \; 0{,}25 = 22{,}50$$
$$v_{c23} = w_{c2} r_3 = 90 \; x \; 0{,}33 = 30{,}00$$
$$v_{c31} = w_{c3} r_1 = 85 \; x \; 1{,}00 = 85{,}00$$
$$v_{c32} = w_{c3} r_2 = 85 \; x \; 0{,}33 = 28{,}33$$
$$v_{c33} = w_{c3} r_3 = 85 \; x \; 0{,}67 = 56{,}67$$
$$v_{c41} = w_{c4} r_1 = 70 \; x \; 1{,}00 = 70{,}00$$
$$v_{c42} = w_{c4} r_2 = 70 \; x \; 0{,}67 = 46{,}67$$
$$v_{c43} = w_{c4} r_3 = 70 \; x \; 0{,}33 = 23{,}33$$
$$v_{c51} = w_{c5} r_1 = 85 \; x \; 0{,}33 = 28{,}33$$
$$v_{c52} = w_{c5} r_2 = 85 \; x \; 1{,}00 = 85{,}00$$
$$v_{c53} = w_{c5} r_3 = 85 \; x \; 0{,}67 = 56{,}67$$

$$\sum_{c=1}^{n} v_{cn1} = v_{c11}v_{c12}v_{c13}v_{c14}v_{c15}$$

$$v_1 = 36{,}00 + 90{,}00 + 85{,}00 + 70{,}00 + 28{,}33 = 309{,}33$$

$$\sum_{c=2}^{n} v_{cn1} = v_{c21}v_{c22}v_{c23}v_{c24}v_{c25}$$

$$v_2 = 18{,}00 + 22{,}50 + 28{,}33 + 46{,}67 + 85{,}00 = 200{,}50$$

$$\sum_{c=3}^{n} v_{cn1} = v_{c31}v_{c32}v_{c34}v_{c35}v_{c35}$$

$$v_3 = 90{,}00 + 30{,}00 + 56{,}67 + 23{,}33 + 56{,}67 = 256{,}67$$

Nilai v_i yang lebih besar mengindikasikan bahwa alternatif A_i lebih terpilih. (Kusumadewi, 2006). Nilai v_1 adalah nilai v terbesar sehingga lokasi yang terpilih sebagai lokasi pembangunan apotek yang strategis adalah lokasi 1 yaitu Kecamatan Ambulu.

4.3 Pengembangan Sistem

Pengembangan sistem yang digunakan oleh penulis adalah pengembangan model *waterfall*. Pengembangan model *waterfall* terdiri dari beberapa tahapan yaitu analisis kebutuhan, desain sistem, penulisan kode program, *testing*, penerapan program dan *maintenance*.

4.3.1 *Statemen of Purpose*

Spatial Decision Support System (SDSS) perencanaan pembangunan apotek ini adalah sebuah sistem berbasis android yang dapat digunakan untuk menunjang keputusan dalam perencanaan penempatan apotek. Pengambilan keputusan dilakukan dengan menggunakan dengan mempertimbangkan kriteria-kriteria dan bobot-bobot kriteria. Kriteria-kriteria dan bobot-bobot kriteria diolah dengan menggunakan metode *Simple Additive* Weighting (SAW) sehingga dapat menghasilkan suatu keputusan yaitu lokasi penempatan apotek yang strategis. Kemudian keputusan yang

dihasilkan dari sistem ini akan ditampilkan pada peta. Dengan adanya system ini, diharapkan dapat membantu dalam perencanaan pembangunan apotek dalam hal menentukan lokasi apotek yang strategis.

4.3.2 Analisis Kebutuhan

Analisis kebutuhan perangkat lunak dalam penelitian ini yaitu dengan cara mengidentifikasi permasalahan yang ada untuk kemudian dicatat dan dijadikan bahan untuk mulai membangun aplikasi perencanaan pembangunan apotek. Analisis kebutuhan yang dilakukan meliputi proses pengumpulan data kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan *fungsional* adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Sedangkan kebutuhan *nonfungsional* adalah kebutuhan yang menitikberatkan pada properti prilaku yang dimiliki oleh sistem. Berikut adalah kebutuhan fungsional dan non-fungsional sistem

a. Kebutuhan fungsional dari sistem yang akan dibangun

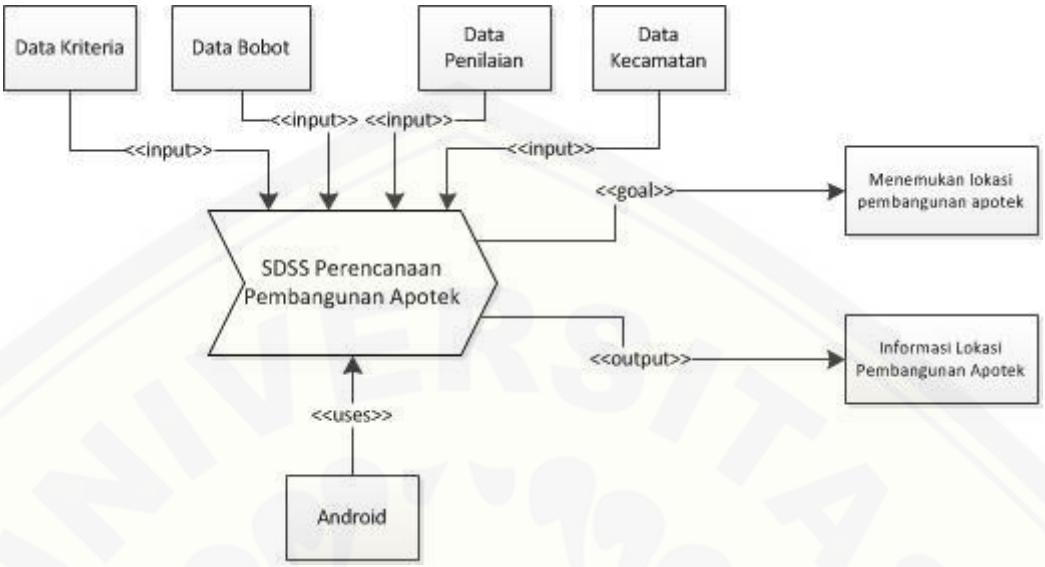
    1. Sistem dapat mengelola data kriteria
    2. Sistem dapat menemukan lokasi pembangunan apotek yang strategis

b. Kebutuhan non-fungsional dari sistem yang akan dibangun

    1. Sistem berbasis android
    2. Sistem terhubung dengan jaringan internet

4.3.3 *Bussiness Process*

Bussines Process digunakan untuk menggambarkan masukan data yang dibutuhkan sistem, *output* dari sistem serta tujuan dari pembuatan sistem. *Bussines Process* sistem dapat dilihat pada gambar 4.1.

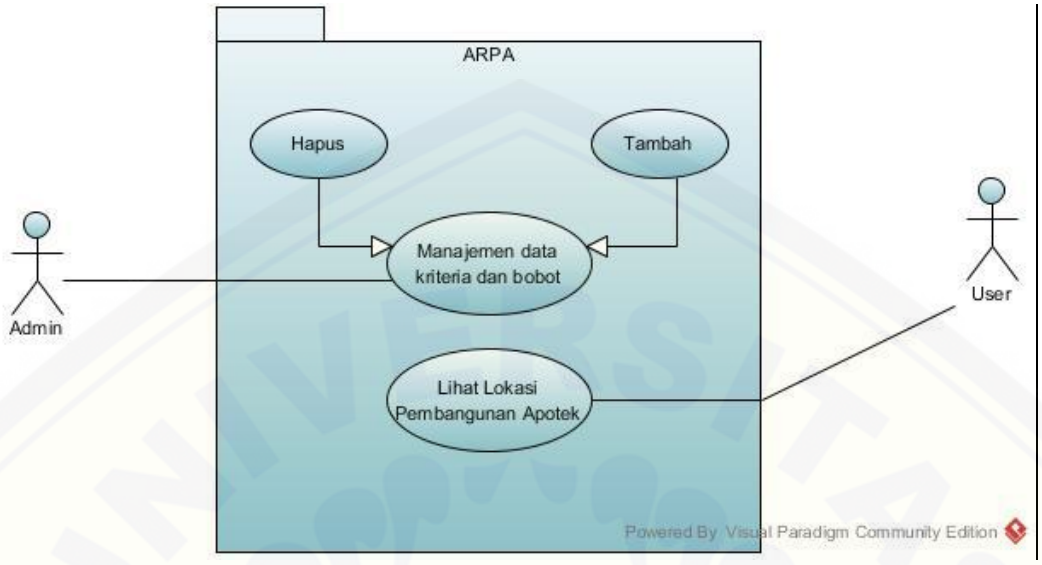


Gambar 4.1 *Bussines Process* Sistem

Berdasarkan gambar 4.1 sistem membutuhkan 4 *input* data dan mengahsilkan 1 *output*. Data yang diperlukan sebagai *input* yaitu data kriteria, data bobot, dan data penilaian. Sedangkan data yang dihasilkan sebagai *output* yaitu informasi lokasi pembangunan apotek. Tujuan atau *goal* dari sistem ini adalah keputusan lokasi pembangunan apotek.

4.3.4 *Usecase Diagram*

Usecase Diagram adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Usecase* sistem dapat diilihat pada gambar 4.2.



Gambar 4.2 *Usecase diagram* sistem

Definisi *usecase* dari *usecase* sistem dapat dilihat pada tabel 4.9. Sedangkan definisi aktor dapat dilihat pada tabel 4.10.

Tabel 4.8 Definisi *Usecase*

No.	Usecase	Deskripsi
1.	Mengelola Data Kriteria	Admin dapat menambah dan menghapus data kriteria.
2.	Lihat Lokasi Pembangunan A potek	User dapat melihat lokasi pembangunan apotek dengan terlebih dahulu menentukan pilihan lokasi sasaran. Kemudian melakukan penilaian terhadap tiap kriteria yang dimiliki kecamatan.

Tabel 4.9 Definisi aktor

No.	Usecase	Deskripsi
1.	Admin	Operator yang mengelola data kriteria penilaian, dan data kecamatan.
2.	User	Operator yang memilih lokasi - lokasi perencanaan pembangunan apotek, menilai tiap kriteria pada tiap kecamatan dan dapat melihat hasil lokasi yang strategis dalam pembangunan apotek

4.3.5 *Scenario*

Scenario menggambarkan alur operasi penggunaan aplikasi perencanaan pembangunan apotek yang meliputi alur utama berserta alur alternatifnya.

Tabel 4.10 *Scenario* Manajemen Kriteria dan Bobot (Tambah data)

ID	**:**	SK-01
Nama *Usecase*	**:**	Manajemen Kriteria dan Bobot (Tambah data)
Aktor	**:**	Admin
Triger	**:**	-
Pra-Kondisi	**:**	Admin login ke sistem
Pasca-Kondisi	**:**	Admin berhasil menambahkan data kriteria

SKENARIO UTAMA	
Actor	***System***
1. Membuka Launcher ARPA	
	2. Menampilkan Splash Screen
	3. Menampilkan Home user

4. Memilih menu Login	
	5. Menampilkan form Login
6. Mengisi form Login	
7. Klik tombol Login	
	8. Menampilkan Home Admin
9. Memilih menu Kriteria	
	10. Menampilkan data kriteria
11. Klik tombol '+'	
	12. Menampilkan form kriteria
13. Mengisi form kriteria	
14. Klik tombol Simpan	
	15. Menampilkan pesan "Data telah tersimpan"
	16. Menampilkan kriteria
SKENARIO ALTERNATIF	
5 Mengisi Form Login	
	6 Jika kosong atau salah, Menampilkan pesan "Username atau password salah"
13. Mengisi form kriteria dan bobot	
	14. Jika kosong, Menampilkan pesan "Tidak Boleh ada yang kosong"

Tabel 4.10 merupakan *scenario* dari *usecase* manajemen kriteria dan bobot (tambah data). *Scenario* manajemen kriteria dan bobot (tambah data) menjelaskan alur proses menambahkan data kriteria. *Scenario* tambah kriteria dibagi menjadi dua bagian yaitu *scenario* utama dan *scenario* alternatif. *Scenario* utama merupakan alur

utama dari proses tambah kriteria. Sedangkan *scenario* alternatif merupakan bagian yang menangani *exception* atau alur alternatif dari proses tambah kriteria. Kondisi setelah *scenario* ini dijalankan adala*h Admin* berhasil menambah data kriteria.

Tabel 4.11 *Scenario* Manajemen Kriteria dan Bobot (Hapus data)

ID	**:**	SK-02
Nama *Usecase*	**:**	Manajemen Kriteria dan Bobot (Hapus data)
Aktor	**:**	Admin
Triger	**:**	-
Pra-Kondisi	**:**	Admin login ke sistem
Pasca-Kondisi	**:**	Admin berhasil menghapus data kriteria

SKENARIO UTAMA	
Actor	***System***
1. Membuka Launcher ARPA	
	2. Menampilkan Splash Screen
	3. Menampilkan Home user
4. Memilih menu Login	
	5. Menampilkan form Login
6. Mengisi form Login	
7. Klik tombol Login	
	8. Menampilkan Home Admin
9. Memilih menu Kriteria	
	10. Menampilkan data kriteria
11. Klik tombol delete pada kriteria yang akan dihapus	
	12. Menampilkan data kriteria

SKENARIO ALTERNATIF	
5. Mengisi Form Login	
	6. Jika kosong atau salah, Menampilkan pesan "Username atau password salah"

Tabel 4.11 merupakan *scenario* dari *usecase* manajemen kriteria dan bobot (hapus data). *Scenario* manajemen kriteria dan bobot (hapus data) menjelaskan alur proses hapus data kriteria. *Scenario* hapus kriteria dibagi menjadi dua bagian yaitu *scenario* utama dan *scenario* alternatif. *Scenario* utama merupakan alur utama dari proses hapus kriteria. Sedangkan *scenario* alternatif merupakan bagian yang menangani *exception* atau alur alternatif dari proses hapus kriteria. Kondisi setelah *scenario* ini dijalankan adalah *Admin* berhasil menghapus data kriteria.

Tabel 4.12 *Scenario* Lihat Lokasi Pembangunan Apotek

ID	**:**	SKN-03
Nama *Usecase*	**:**	Lihat Lokasi Pembangunan Apotek
Aktor	**:**	User
Triger	**:**	-
Pra-Kondisi	**:**	Admin memilih rencana lokasi atau alternative dan melakukan penilaian
Pasca-Kondisi	**:**	Admin berhasil melihat lokasi pembangunan apotek

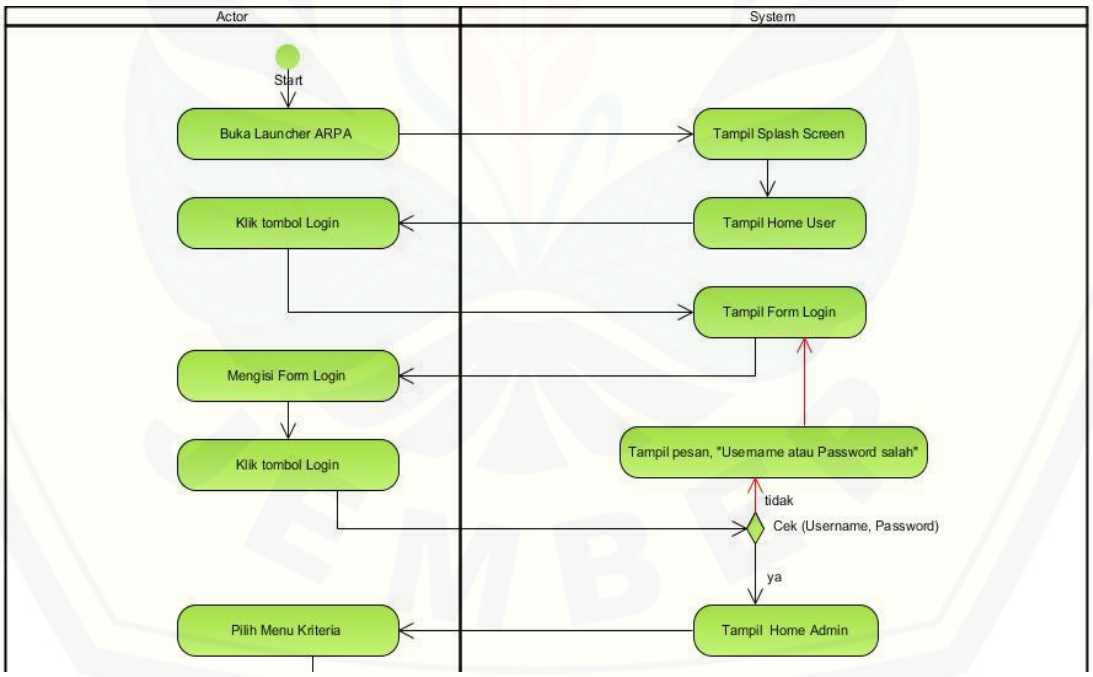
SKENARIO UTAMA	
Actor	***System***
1. Membuka Launcher ARPA	

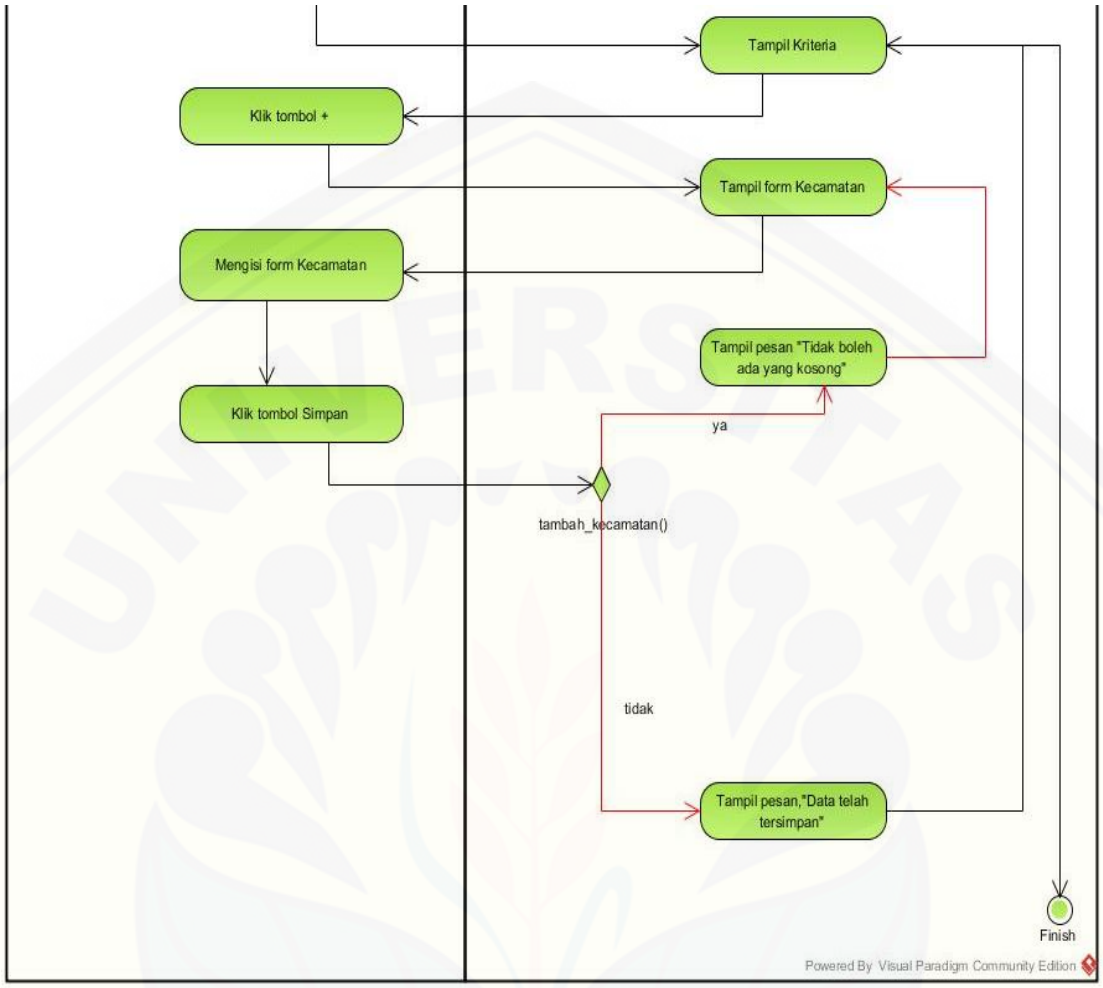
	2. Menampilkan Home user
3. Memilih menu TLP (Temukan Lokasi Pembangunan)	
	4. Menampilkan pilihan kecamatan
5. Memilih lokasi rencana pembangunan apotek	
6. Klik tombol pilih	
	7. Menampilkan lokasi rencana pembangunan apotek
8. Jika klik tombol delete	
	9. Menghapus lokasi rencana pembangunan apotek
	10. Menampilkan lokasi rencana pembangunan apotek
11. Jika klik tombol nilai	
	12. Menampilkan form penilaian lokasi pembangunan apotek
13. Mengisi form penilaian	
14. Klik Tombol Proses	
	15. Menampilkan lokasi pembangunan Apotek
SKENARIO ALTERNATIF	
8. Klik Tombol Proses	
	9. Jika ada yang kosong, Menampilkan pesan “Harap lakukan penilaian terlebih dahulu”

Tabel 4.12 merupakan *scenario* dari *usecase* Lihat Lokasi Pembangunan Apotek. *Scenario* Lihat Lokasi Pembangunan Apotek menjelaskan alur proses menemukan lokasi pembangunan apotek yang strategis. *Scenario* Lihat Lokasi Pembangunan Apotek dibagi menjadi dua bagian yaitu scenario utama dan *scenario* alternatif. *Scenario* utama merupakan alur utama dari proses Lihat Lokasi Pembangunan Apotek. Sedangkan *scenario* alternatif merupakan bagian yang menangani exception atau alur alternatif dari proses Lihat Lokasi Pembangunan Apotek. Kondisi setelah *scenario* ini dijalankan adalah *user* berhasil melihat lokasi pembangunan apotek yang strategis.

4.3.6 Activity diagram

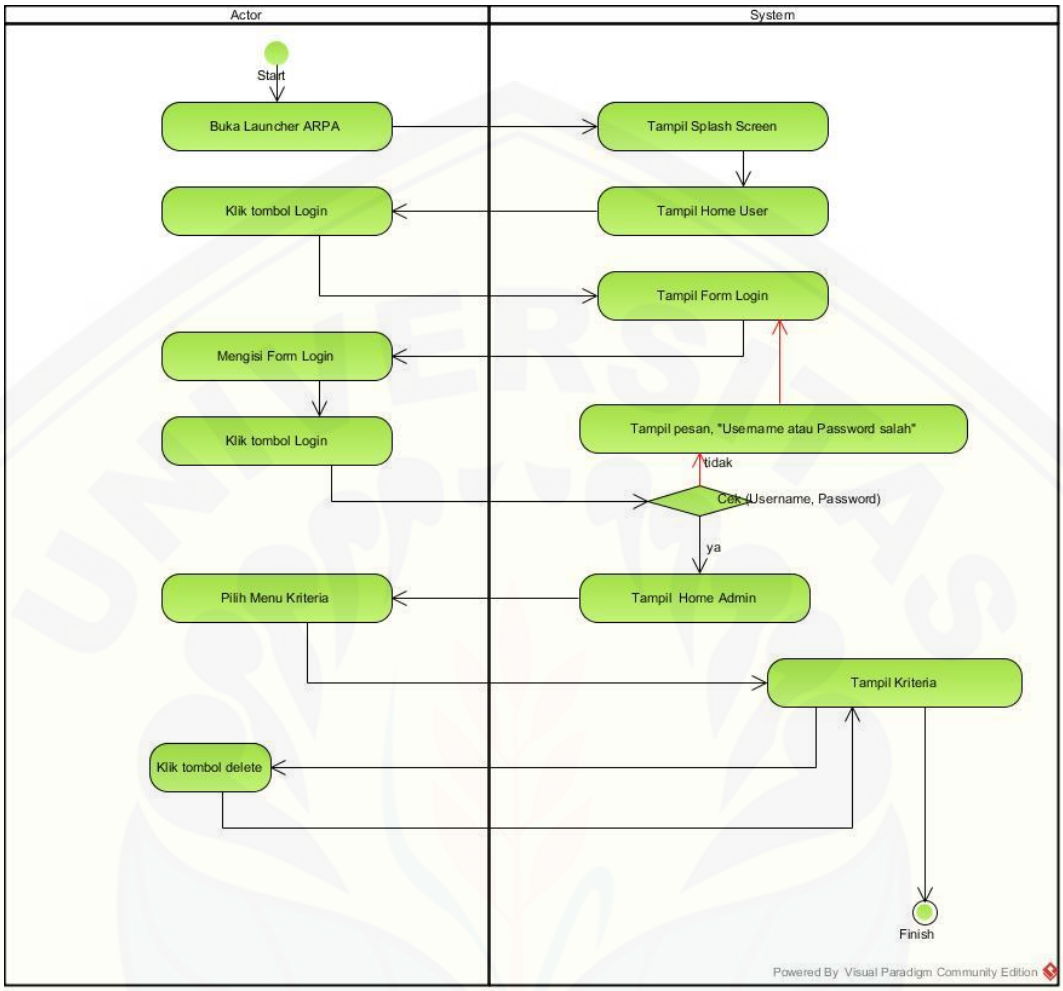
Activity diagram aplikasi perencanaan pembangunan apotek ini berfungsi untuk menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang. *Activity Diagram* sistem adalah sebagai berikut :





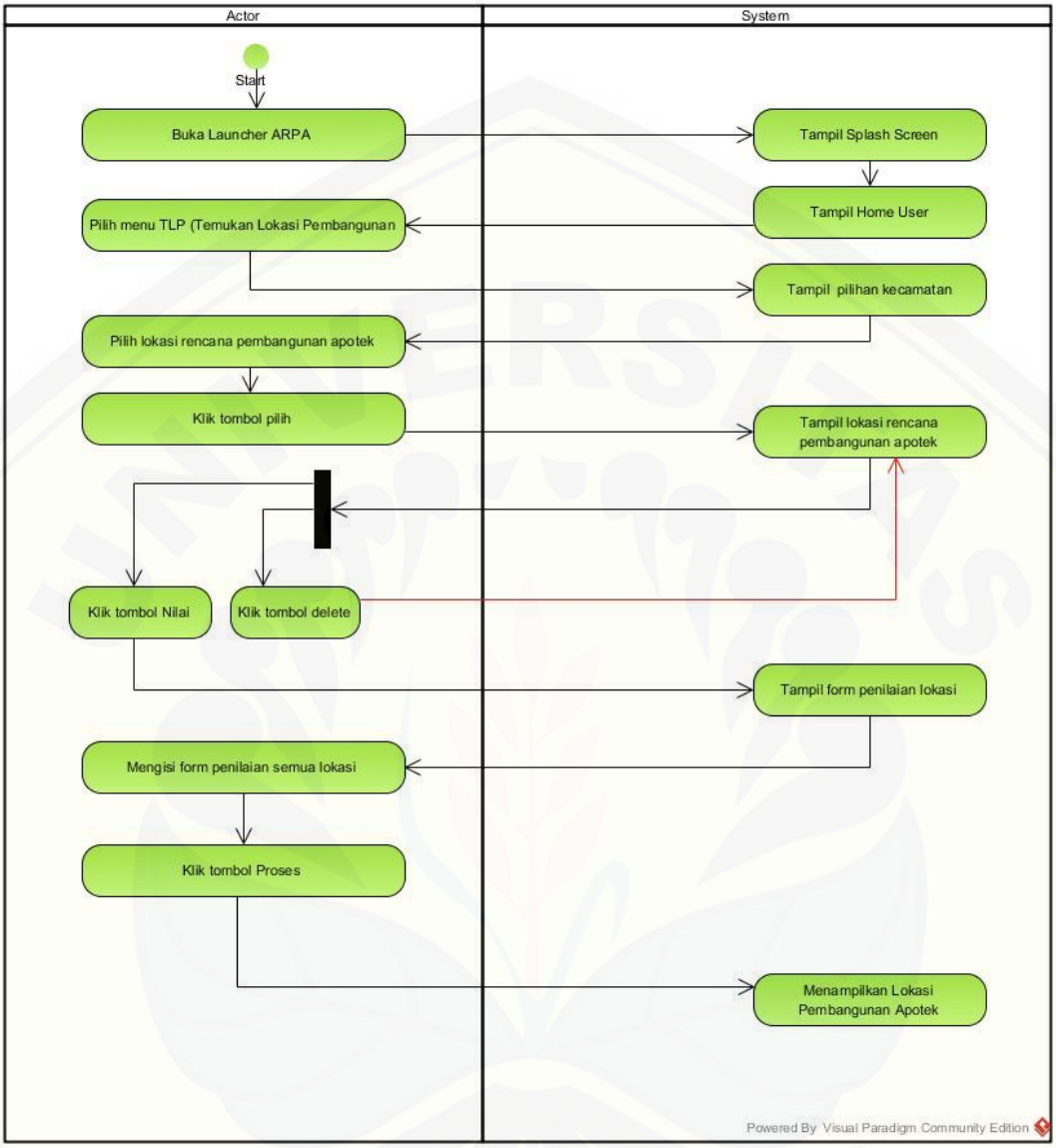
Gambar 4.3 *Activity diagram* Manajemen Kriteria dan Bobot (Tambah data)

Gambar 4.3 menggambarkan alur aktivitas dari proses tambah kriteria. *Activity diagram* tambah kriteria menjelaskan proses mulai dari *Admin* memilih menu kriteria dan menambahkan data kriterianya hingga data kriteria berhasil disimpan ke *database*. Didalam *Activity diagram* ini terdapat dua percabangan *decision* yaitu pada saat memilih tombol "Simpan".



Gambar 4.4 *Activity diagram* Manajemen Kriteria dan Bobot (Hapus data)

Gambar 4.4 menggambarkan alur aktivitas dari proses hapus kriteria. Untuk menghapus kriteria harus melakukan *login* sebagai *admin* terlebih dahulu.



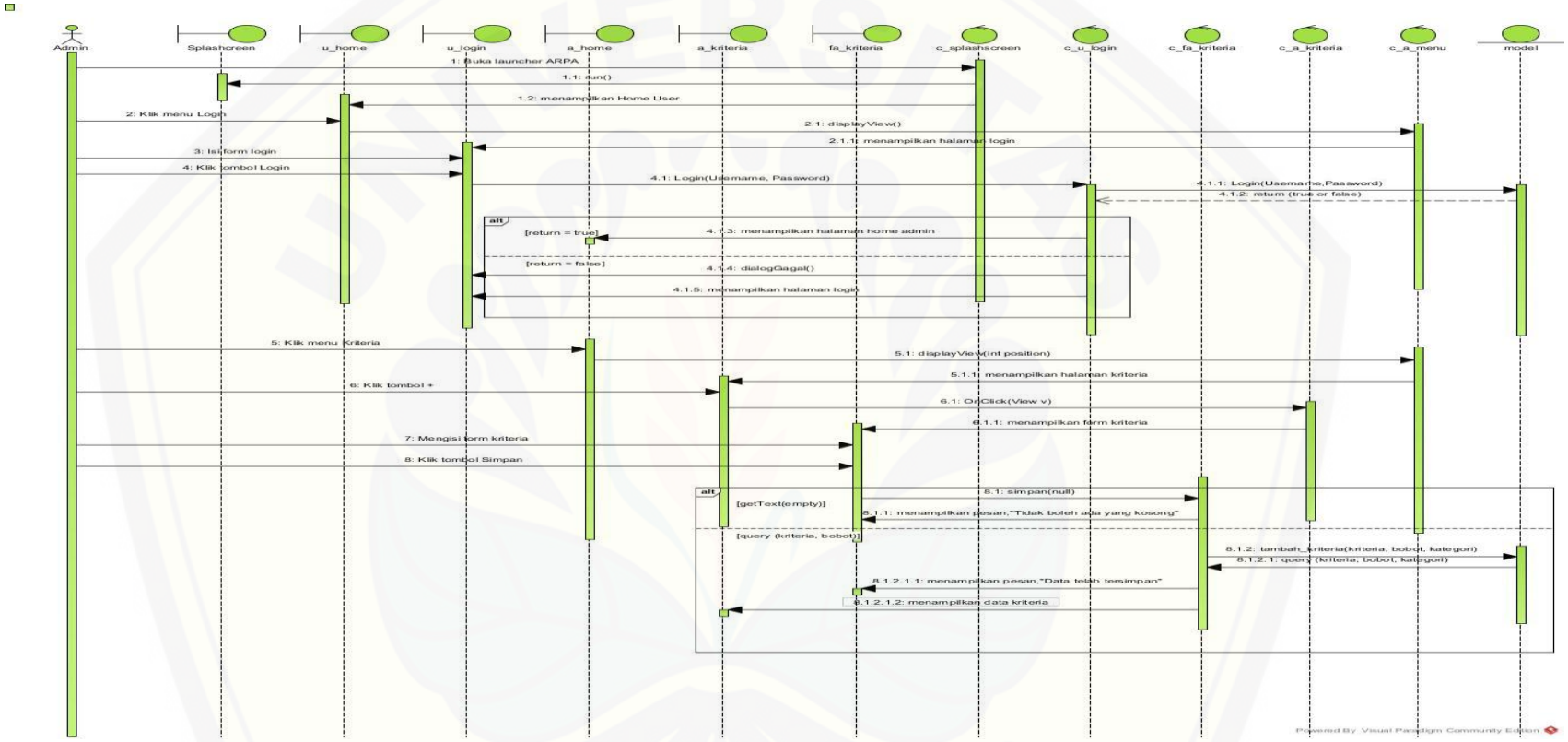
Gambar 4.5 *Activity diagram* Lihat Lokasi Pembangunan Apotek

Gambar 4.5 menggambarkan alur aktivitas dari proses lihat lokasi pembangunan apotek. *User* dapat melihat lokasi pembangunan apotek dengan memilih menu TLP (Temukan Lokasi Pembangunan). Kemudian memilih rencana lokasi pembangunan dan memberikan penilaian terhadap rencana lokasi

pembangunan apotek. Lokasi pembangunan apotek yang strategis akan tampil pada peta.

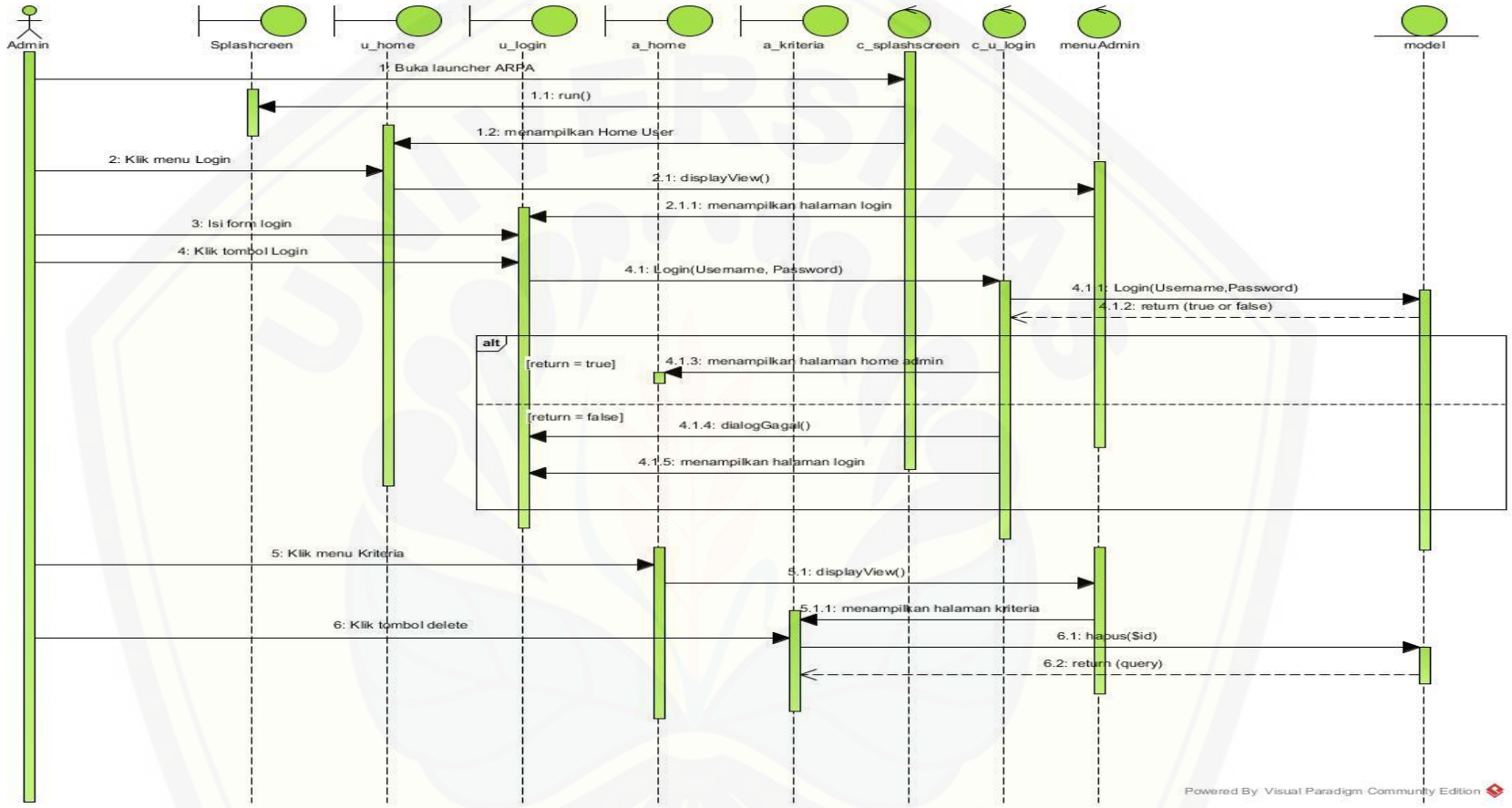
4.3.7 Sequence diagram

Sequence diagram pada aplikasi perencanaan pembangunan apotek ini digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian / *even* untuk menghasilkan *output* tertentu. *Sequence Diagram* diawali dari apa yang memicu aktifitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. *Sequence Diagram* dari sistem adalah sebagai berikut :



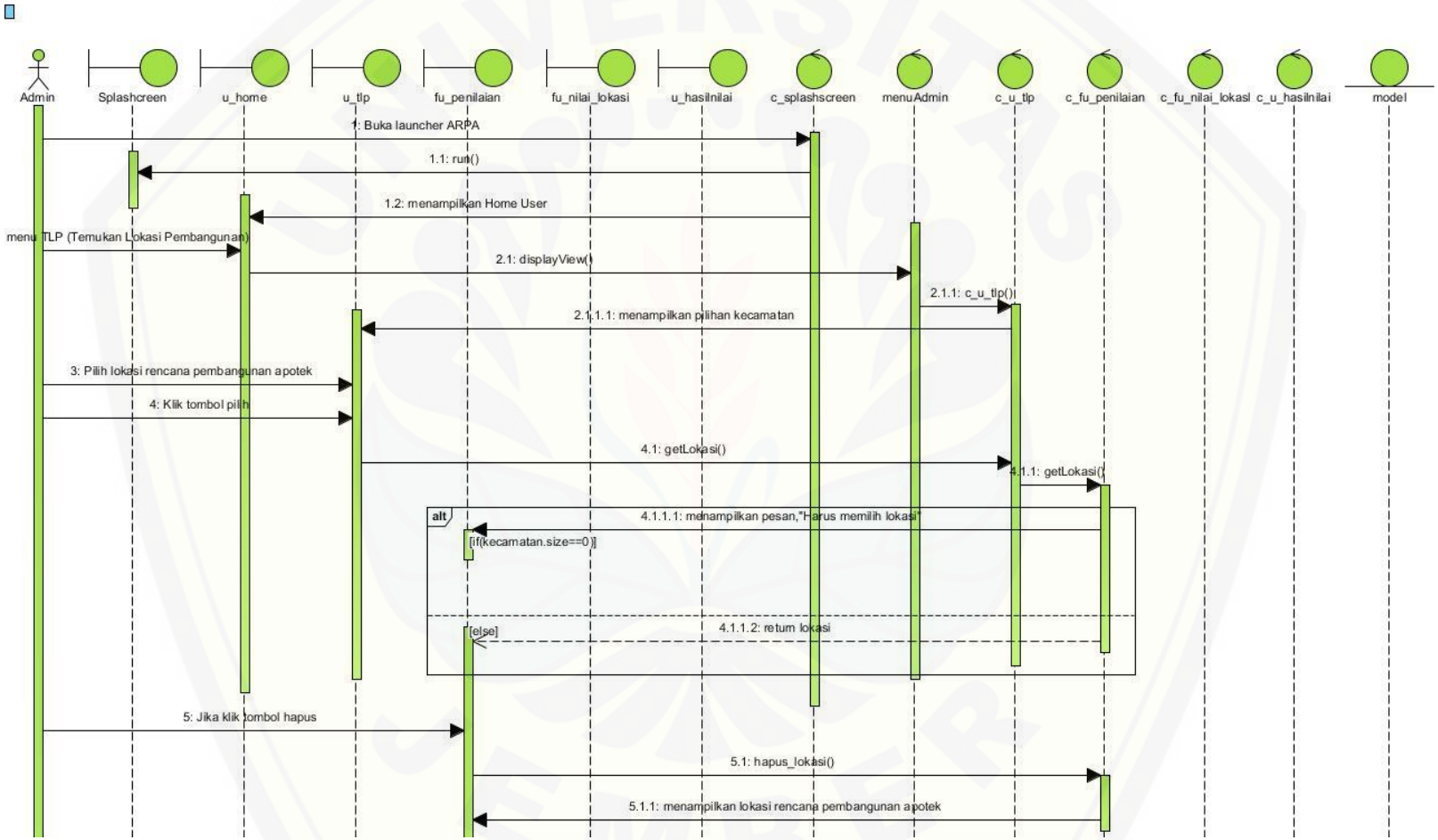
Gambar 4.6 *Sequence diagram* Manajemen Kriteria dan Bobot (Tambah data)

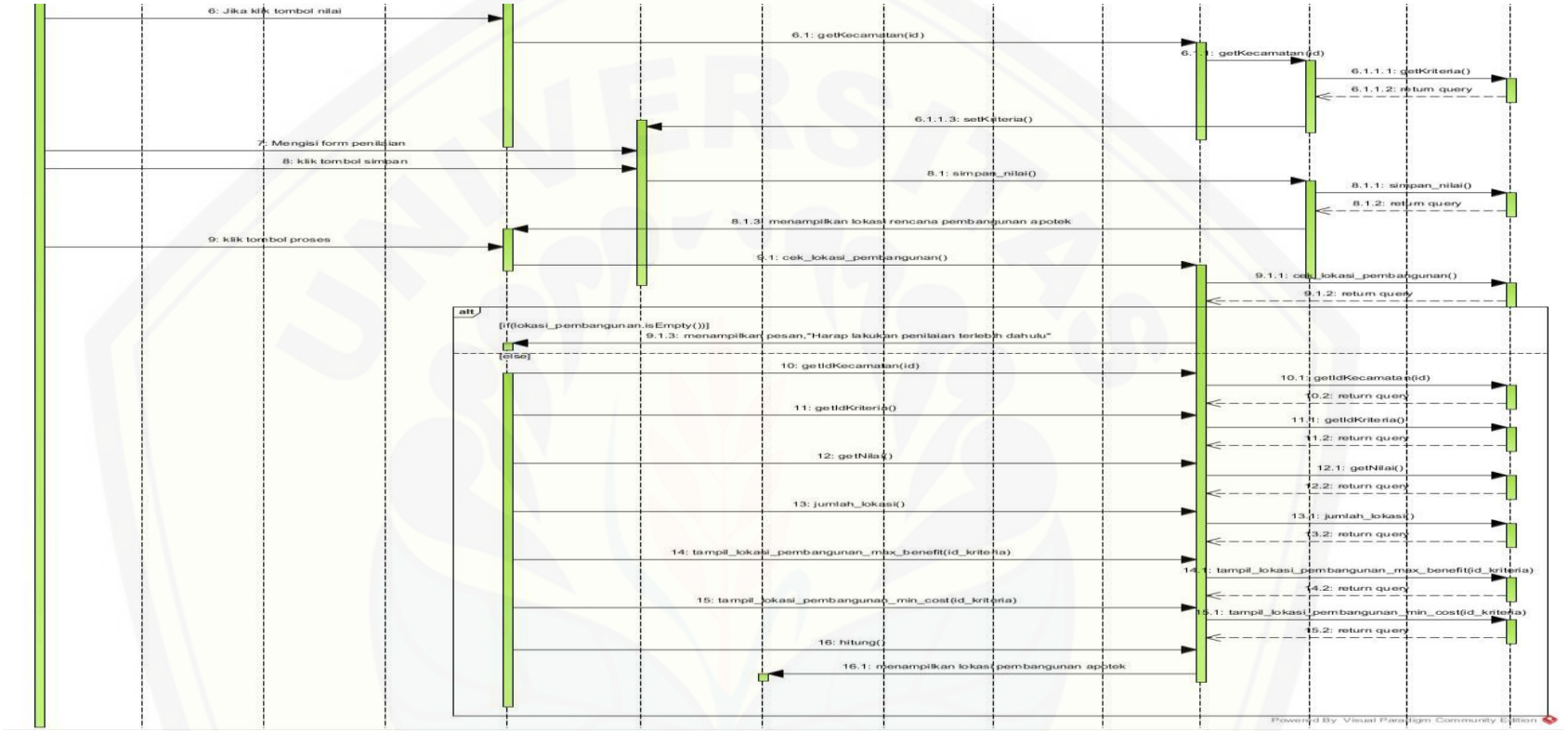
Gambar 4.6 menjelaskan alur *Model View Controller* (MVC) dari proses tambah data kriteria. Pada *Sequence diagram* tambah kriteria digunakan 1 *model class, 6 view class, dan 6 controller class.*



Gambar 4.7 *Sequence diagram* hapus kriteria

Gambar 4.7 menjelaskan alur *Model View Controller* (MVC) dari proses hapus data kriteria. Pada *Sequence diagram* tambah kriteria digunakan 1 *model class, 5 view class, dan 3 controller class.*



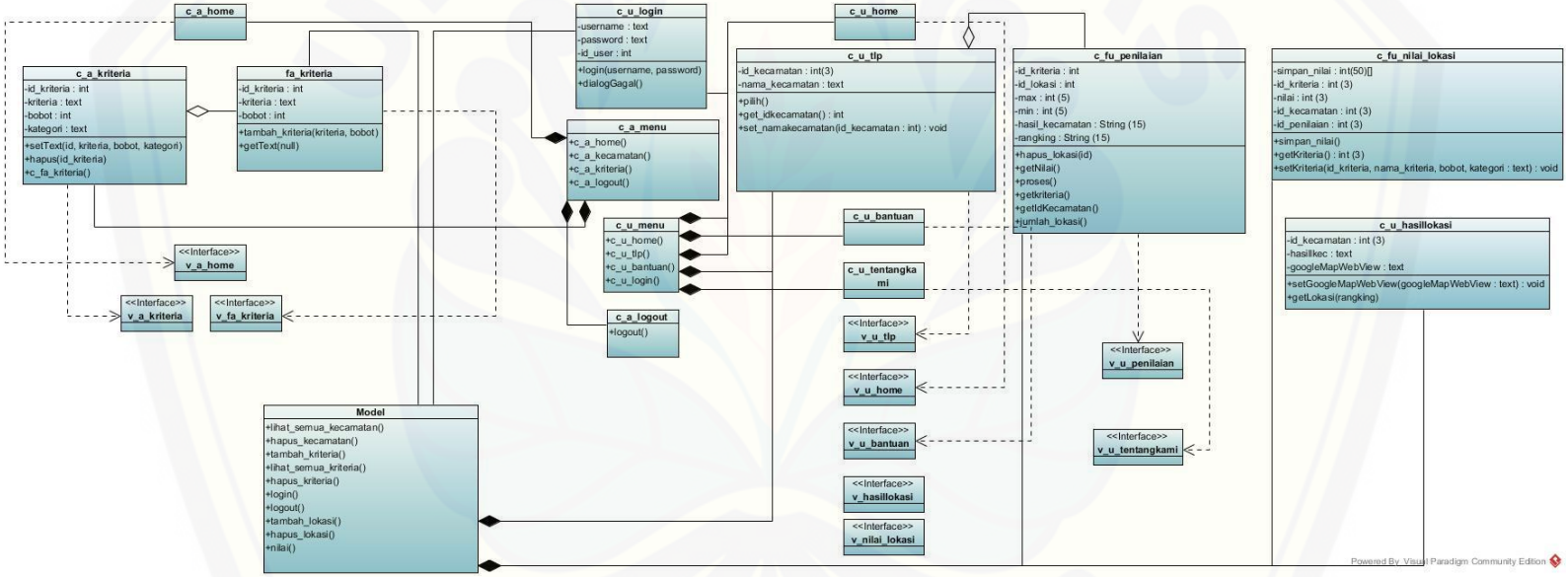


Gambar 4.8 *Sequence diagram* Lihat Lokasi Pembangunan Apotek

Gambar 4.6 menjelaskan alur *Model View Controller* (MVC) dari proses lihat lokasi pembangunan apotek. Pada *Sequence diagram* tambah kriteria digunakan 1 *model class, 6 view class, dan 6 controller class.*

4.3.8 Class Diagram

Class diagram menggambarkan struktur dan penjelasan class, paket, dan objek serta hubungan satu sama lain seperti kompisisi, asosiasi, dan lain-lain. Selain itu class diagram juga menjelaskan hubungan antar class dalam sebuah sistem yang sedang dirancang sehingga bagaimana caranya setiap class saling berkalaborasi untuk mencapai sebuah tujuan. *Class diagram* sistem dapat dilihat pada gambar 4.9.

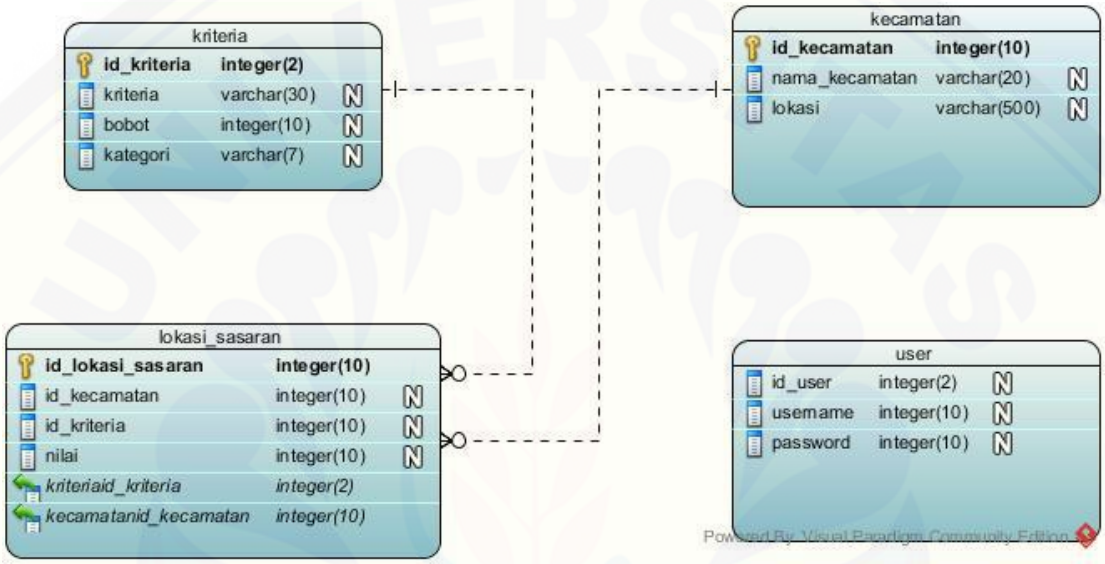


Gambar 4.9 *Class Diagram* Sistem

Gambar 4.9 menjelaskan hubungan antar objek dari *class* yang digunakan dalam sistem. *Class Diagram* sistem terdiri dari 10 *class view*, 14 *class controller*, dan 1 *class model*. Terdapat 4 hubungan pada kelas-kelas yaitu *dependency*, *asosiasi*, *komposisi* dan *agregasi*.

4.3.9 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) pada SDSS perencanaan pembangunan apotek menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. ERD aplikasi ditunjukkan pada gambar 4.10.



Gambar 4.10 *Entity Relationship Diagram* (ERD)

Gambar 4.10 menjelaskan entitas dan hubungan antar entitas dalam sistem. Dalam *ERD* terdapat 4 entitas yaitu kriteria, kecamatan, lokasi_sasaran dan user. Entitas kriteria digunakan untuk kelola kriteria, entitas user digunakan untuk *login* hak akses, entitas kecamatan digunakan untuk menampung data kecamatan dan entitas lokasi_sasaran untuk menampung penilaian lokasi rencana pembangunan apotek.

4.4 Penulisan Kode Program

Tahap penulisan kode program merupakan tahap lanjutan dari desain sistem. Desain sistem dari semua fitur sistem yang telah dibuat menggunakan pemodelan UML akan diimplementasikan kedalam kode program. Penulisan kode program menggunakan bahasa pemrograman *Java* dan *Extensible Markup Language* (XML).

a. Kode Program Manajemen Kriteria Dan Bobot (Tambah Kriteria)

```
51          private void tambah_kriteria() {
52              String kriteria = nikriteria.getText().toString();
53              String bobot = nibobot.getText().toString();
54              int kategori = nikategori.getCheckedRadioButtonId();
55
56              if (kriteria.isEmpty() || bobot.isEmpty()) {
57                  Toast.makeText(getActivity().getBaseContext(),
58                          "Tidak boleh ada yang kosong", Toast.LENGTH_LONG)
59                          .show();
60              }
61
62              else {
63                  if (kategori % 2 == 0) {
64                      int intBobot = Integer.parseInt(bobot);
65                      db.tambah_kriteria(kriteria, intBobot, "Cost");
66                      Toast.makeText(getActivity().getBaseContext(),
67                              "Data telah tersimpan", Toast.LENGTH_LONG)
68                              .show();
69                      Fragment fragment = new c_a_kriteria();
70                      FragmentManager fragmentManager = getFragmentManager();
71                      fragmentManager.beginTransaction()
72                              .replace(R.id.frame_container_admin, fragment)
73                              .addToBackStack(null).commit();
74                  }
75                  else {
76                      int intBobot = Integer.parseInt(bobot);
77                      db.tambah_kriteria(kriteria, intBobot, "Benefit");
78                      Toast.makeText(getActivity().getBaseContext(),
79                              "Data telah tersimpan", Toast.LENGTH_LONG)
80                              .show();
81                      Fragment fragment = new c_a_kriteria();
82                      FragmentManager fragmentManager = getFragmentManager();
83                      fragmentManager.beginTransaction()
84                              .replace(R.id.frame_container_admin, fragment)
85                              .addToBackStack(null).commit();
86                  }
87              }
88          }
```

Gambar 4.11 Kode Program Manajemen Kriteria Dan Bobot (Tambah Data)

Gambar 4.12 merupakan kode program manejemen kriteria dan bobot (tambah data) yang berfungsi agar dapat menambahkan data kriteria. Kode 56-60 adalah kode yang dijalankan ketika form penambahan data kriteria ada yang kosong maka akan menampilkan pesan "Tidak boleh ada yang kosong". Jika form penambahan kriteria sudah terisi semua maka akan dijalankan kode program 62-87 yaitu memasukkan data kriteria ke *database* dan akan menampilkan halaman kriteria.

b. Kode program manajemen kriteria dan bobot (Hapus Kriteria)

```
160     public void hapus_kriteria(int id) {
161
162         db.hapus_kriteria(id);
163         Fragment fragment = new c_a_kriteria();
164         FragmentManager fragmentManager = getFragmentManager();
165         fragmentManager.beginTransaction()
166                 .replace(R.id.frame_container_admin, fragment)
167                 .addToBackStack(null).commit();
168
169     }
```

Gambar 4.12 Manajemen kriteria dan bobot (hapus kriteria)

Gambar 4.13 merupakan kode program manajemen kriteria dan bobot (hapus kriteria) yang berfungsi untuk menghapus data kriteria yang terletak pada kode 162. Kode 163-167 adalah kode program untuk menampilkan halaman kriteria.

c. Kode Program Lihat Pembangunan Apotek

1. Fungsi Tampil_Lokasi_Pembangunan_Max_Benefit()

```
475  public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_max_benefit(
476          String id_kriteria) {
477
478      ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
479      SQLiteDatabase database = this.getWritableDatabase();
480      Cursor cursor = database.rawQuery(
481              "SELECT max(nilai) as max FROM lokasi_pembangunan WHERE id_kriteria='"
482                      + id_kriteria + "'", null);
483
484      if (cursor.moveToFirst()) {
485          do {
486              HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
487              hashMapBenefit.put("max", cursor.getString(0));
488              arrayListkecamatan.add(hashMapBenefit);
489
490          } while (cursor.moveToNext());
491      }
492
493      return arrayListkecamatan;
494  }
```

Gambar 4.13 Kode Program Fungsi Tampil_Lokasi_Max_Benefit()

Fungsi tampil_lokasi_max_benefit() merupakan kode program lihat pembangunan. Kode 475-494 adalah kode program untuk mengambil nilai max pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

2. Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

```
496  public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_min_cost(
497          String id_kriteria) {
498
499      ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
500      SQLiteDatabase database = this.getWritableDatabase();
501      Cursor cursor = database.rawQuery(
502              "SELECT min(nilai) as min FROM lokasi_pembangunan WHERE id_kriteria='"
503                      + id_kriteria + "'", null);
504
505      if (cursor.moveToFirst()) {
506          do {
507
508              HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
509              hashMapBenefit.put("min", cursor.getString(0));
510              arrayListkecamatan.add(hashMapBenefit);
511
512          } while (cursor.moveToNext());
513      }
514
515      return arrayListkecamatan;
516  }
```

Gambar 4.14 Kode Program Fungsi Tampil_Lokasi_Min_Cost()

Fungsi tampil_lokasi_min_cost() merupakan kode program lihat pembangunan apotek yang berfungsi agar *user* dapat melihat lokasi pembangunan apotek yang strategis. Kode 499-515 adalah kode program untuk mengambil nilai min pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

7. Fungsi Hitung()

```
// cek lokasi db
ArrayList<HashMap<String, String>> lokasi_pembangunan = db
        .cek_lokasi_pembangunan();

if (lokasi_pembangunan.isEmpty()) {
    Toast.makeText(getActivity().getBaseContext(),
            "Harap lakukan penilaian terlebih dahulu",
            Toast.LENGTH_LONG).show();
} else {
    System.out.println(rangking + " : Lokasi Terbaik");
    Fragment fragment = new c_u_hasillokasi();
    FragmentManager fragmentManager = getFragmentManager();
    fragmentManager.beginTransaction()
            .replace(R.id.frame_container_user, fragment)
            .addToBackStack(null).commit();

    ArrayList<HashMap<String, String>> arrayJumlahKriteria = db
            .tampil_semua_kriteria();
    ArrayList<HashMap<String, String>> arrayJumlahLokasi = db
            .jumlah_lokasi();
    double alt[][] = new double[arrayJumlahKriteria.size()][arrayJumlahLokasi
            .size()];
    ArrayList<HashMap<String, String>> arrayListKriteria = db
            .tampil_semua_kriteria();
    nilai_max = new ArrayList<String>();
    nilai_min = new ArrayList<String>();
    String simpan_id[] = new String[arrayJumlahLokasi
            .size()];

    if (arrayListKriteria.size() > 0) {
        for (int i = 0; i < arrayListKriteria.size(); i++) {

            HashMap<String, String> hashMapKriteria = arrayListKriteria
                    .get(i);
            String id_kriteria = hashMapKriteria
                    .get("id_kriteria");
            String bobot = hashMapKriteria.get("bobot");
            String kategori = hashMapKriteria
```

```
                .get("kategori");
        ArrayList<HashMap<String, String>> arrayListNilaiMax = db
                .tampil_lokasi_pembangunan_max_benefit(id_kriteria);
        HashMap<String, String> hashMapNilaiMax = arrayListNilaiMax
                .get(0);
        max = hashMapNilaiMax.get("max");
        ArrayList<HashMap<String, String>> arrayListNilaiMin = db
                .tampil_lokasi_pembangunan_min_cost(id_kriteria);

        HashMap<String, String> hashMapNilaiMin = arrayListNilaiMin
                .get(0);
        min = hashMapNilaiMin.get("min");
        ArrayList<HashMap<String, String>> arrayListAlternatif = db
                .tampil_lokasi_pembangunan(id_kriteria);

        if (arrayListAlternatif.size() > 0) {
            for (int j = 0; j < arrayListAlternatif
                    .size(); j++) {

                HashMap<String, String> hashMapNormalisasi = arrayListAlternatif
                        .get(j);
                int nilai = Integer
                        .parseInt(hashMapNormalisasi
                                .get("nilai"));
                String id_kec = hashMapNormalisasi
                        .get("id_kecamatan");
                // nilai preferensi benefit
                if (kategori
                        .equalsIgnoreCase("benefit")) {
                    alt[i][j] = (nilai / Double
                            .parseDouble(max))
                            * Double.parseDouble(bobot);
                }
                // nilai preferensi cost
                else {
                    alt[i][j] = (Double
                            .parseDouble(min) / nilai)
                            * Double.parseDouble(bobot);
```

```
254                         }
255
256                         System.out.println("v = " + " "
257                                 + alt[i][j] + "cell [" + i
258                                 + "][" + j + "]");
259                     }
260                 }
261
262             }
263         }
264         double nilai_total[] = new double[arrayJumlahLokasi
265                 .size()];
266         rangking = simpan_id[0];
267         double temp_total = 0;
268         for (int j = 0; j < arrayJumlahLokasi.size(); j++) {
269             double tempnilai = 0;
270             simpan_id[j] = c_u_tlp.id_kecamatan.get(j);
271             Log.i(simpan_id[j], "ini id kec");
272             for (int k = 0; k < arrayJumlahKriteria.size(); k++) {
273                 //total nilai preferensi tiap lokasi
274                 nilai_total[j] = tempnilai + alt[k][j];
275                 tempnilai = nilai_total[j];
276             }
277             if (nilai_total[j] > temp_total) {
278                 //Lokasi pembangunan apotek
279                 rangking = simpan_id[j];
280                 temp_total = nilai_total[j];
281             }
282
283             System.out.println(nilai_total[j] + " "
284                     + " : nilai alt " + j);
285         }
286         System.out.println(rangking + " "
287                 + " ini adalah hasil akhir");
288     }
289 }
```

Gambar 4.15 Kode Program Fungsi Hitung()

Gambar 4.15. Kode 496-516 adalah kode program untuk mengambil nilai min pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut juga berada pada kelas model. Selanjutnya digunakan kelas c_fu_penilaian(). Kode 179-185 adalah kode untuk mengecek apakah *user* sudah melakukan penilaian. Jika *user* belum melakukan penilaian pada lokasi maka akan menampilkan pesan "Harus melakukan penilaian terlebih dahulu". Jika *user* telah melakukan penilaian, maka kode program yang dijalankan adalah kode 187-289. Kode 217-229 adalah kode program untuk mengambil nilai max dan min tiap kriteria

yang telah dihitung pada kelas model. Kode 243-253 adalah kode program untuk menghitung nilai preferensi *benefit* dan *cost*. Kode 274 adalah kode program untuk menjumlahkan nilai preferensi pada tiap lokasi. Kemudian pada kode 279-281, lokasi yang memiliki total nilai preferensi paling besar dijadikan solusi pembangunan apotek yang strategis.

4.5 Pengujian Sistem

Pada penelitian ini penulis menggunakan dua metode pengujian sistem yaitu *Black Box Testing* dan *White Box Testing*. Berikut adalah hasil pengujian sistem :

4.5.1 *White Box Testing*

Pengujian *white box* testing terdiri dari listing program, diagram alir, *cyclomatic complexcity*, jalur program independen dan *test case*. Pada tahap ini fitur yang diuji adalah sebagai berikut:

1. Pengujian Manajemen Kriteria dan Bobot (Tambah Kriteria)

a. Kode Program Manajemen Kriteria Dan Bobot (Tambah Kriteria)

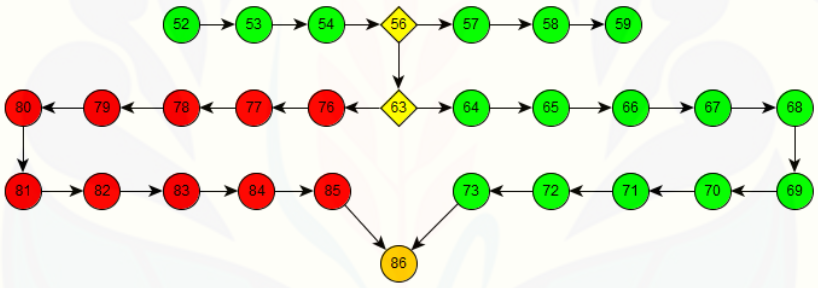
```
51          private void tambah_kriteria() {
52              String kriteria = nikriteria.getText().toString();
53              String bobot = nibobot.getText().toString();
54              int kategori = nikategori.getCheckedRadioButtonId();
55
56              if (kriteria.isEmpty() || bobot.isEmpty()) {
57                  Toast.makeText(getActivity().getBaseContext(),
58                          "Tidak boleh ada yang kosong", Toast.LENGTH_LONG)
59                          .show();
60              }
61
62              else {
63                  if (kategori % 2 == 0) {
64                      int intBobot = Integer.parseInt(bobot);
65                      db.tambah_kriteria(kriteria, intBobot, "Cost");
66                      Toast.makeText(getActivity().getBaseContext(),
67                              "Data telah tersimpan", Toast.LENGTH_LONG)
68                              .show();
69                      Fragment fragment = new c_a_kriteria();
70                      FragmentManager fragmentManager = getFragmentManager();
71                      fragmentManager.beginTransaction()
72                              .replace(R.id.frame_container_admin, fragment)
73                              .addToBackStack(null).commit();
74                  }
```

```
75                  else {
76                      int intBobot = Integer.parseInt(bobot);
77                      db.tambah_kriteria(kriteria, intBobot, "Benefit");
78                      Toast.makeText(getActivity().getBaseContext(),
79                              "Data telah tersimpan", Toast.LENGTH_LONG)
80                              .show();
81                      Fragment fragment = new c_a_kriteria();
82                      FragmentManager fragmentManager = getFragmentManager();
83                      fragmentManager.beginTransaction()
84                              .replace(R.id.frame_container_admin, fragment)
85                              .addToBackStack(null).commit();
86                  }
87              }
88          }
```

Gambar 4.16 Kode Program Fungsi Tambah Kriteria

Gambar 4.16 adalah kode program tambah fungsi kriteria. Kode program tersebut terdapat *exception code* yaitu ketika form tambah kriteria ada yang kosong maka menampilkan pesan,"Tidak boleh ada yang kosong".

b. Diagram Alir Manajemen Kriteria dan Bobot (Tambah Kriteria)



Gambar 4.17 Diagram alir fitur Tambah Kriteria

Gambar 4.17 menjelaskan alur jalannya program berdasarkan kode program pada gambar 4.16. Kode program diubah kedalam alir untuk dicari nilai kompleksitas siklomatiknya. Dari nilai kompleksitas siklomatik akan diketahui jalur dari alur jalannya sistem setiap *function*.

c. Perhitungan *Cyclomatic Complexity* Manajemen Kriteria dan Bobot (Tambah Kriteria)

Perhitungan diagram alir pada tambah kriteria menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

function tambah_kriteria() : V(G) = E – N + 2 = 29 – 28 + 2 = 3

d. Pengujian Jalur Manajemen Kriteria dan Bobot (Tambah Kriteria)

Pengujian jalur jalur program berdasarkan perhitungan *Cyclomatic Complexity* adalah sebagai berikut :

function tambah_kriteria() :

Jalur 1 : 52-53-54-56-57-58-59

Jalur 2 : 52-53-54-56-64-65-66-67-68-69-70-71-72-72-73-86

Jalur 3 : 52-53-54-56-61-76-77-778-79-80-81-82-83-84-85-86

e. *Test Case* manajemen kriteria dan bobot (tambah kriteria)

Tabel 4.13 *Test Case* Manajemen Kriteria dan Bobot (Tambah Kriteria)

Test Case function tambah_kriteria()	
Jalur 1	
Test Case	Jika form ada yang kosong atau belum terisi
Target yang diharapkan	Menampilkan pesan,"Tidak boleh ada yang kosong"
Hasil pengujian	Benar
Path/Jalur	52-53-54-56-57-58-59
Jalur 2	
Test Case	Jika kategori *cost*
Target yang diharapkan	Menyimpan data kriteria dengan kategori *cost*
Hasil pengujian	Benar
Path/Jalur	52-53-54-56-64-65-66-67-68-69-70-71-72-72-73-86
Jalur 3	
Test Case	Jika kategori benefit
Target yang diharapkan	Menyimpan data kriteria dengan kategori *benefit*

Hasil pengujian	Benar
Path/Jalur	52-53-54-56-61-76-77-778-79-80-81-82-83-84-85-86

2. Pengujian Manajemen Kriteria dan Bobot (Hapus Kriteria)

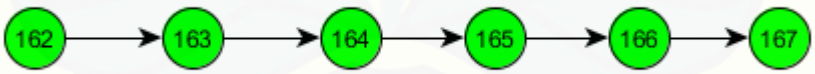
a. Kode Program Manajemen Kriteria dan Bobot (Hapus Kriteria)

```
160    public void hapus_kriteria(int id) {
161
162        db.hapus_kriteria(id);
163        Fragment fragment = new c_a_kriteria();
164        FragmentManager fragmentManager = getFragmentManager();
165        fragmentManager.beginTransaction()
166                .replace(R.id.frame_container_admin, fragment)
167                .addToBackStack(null).commit();
168
169    }
```

Gambar 4.18 Kode Program Manajemen Kriteria dan Bobot (Hapus Kriteria)

Gambar 4.18 merupakan kode program hapus kriteria. Ketika kode tersebut berjalan maka akan menghapuskan data kriteria dari *database*. Setelah itu akan menampilkan data kriteria.

b. Diagram alir manajemen kriteria dan bobot (Hapus Kriteria)



Gambar 4.19 Diagram Alir Manajemen Hapus Kriteria

Gambar 4.19 menjelaskan alur jalannya program berdasarkan kode program pada gambar 4.18. Kode program diubah kedalam alir untuk dicari nilai kompleksitas siklomatiknya. Dari nilai kompleksitas siklomatik akan diketahui jalur dari alur jalannya sistem setiap *function*.

c. Perhitungan *Cyclomatic Complexity* Manajemen Kriteria dan Bobot (Hapus Kriteria)

Perhitungan diagram alir menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

function hapus_kriteria() : V(G) = E – N + 2 = 5 – 6 + 2 = 1

d. Pengujian Jalur Manajemen Kriteria dan Bobot (Hapus Kriteria)

Pengujian jalur program berdasarkan perhitungan *Cyclomatic Complexity* adalah sebagai berikut :

function tambah_kriteria() :

Jalur 1 : 162-163-164-165-166-167

e. *Test Case* Manajemen Kriteria dan Bobot (Hapus Kriteria)

Tabel 4.14 *Test Case* Manajemen Kriteria dan Bobot (Hapus Kriteria)

Test Case function hapus_kriteria()	
Jalur 1	
Test Case	Hapus kriteria
Target yang diharapkan	Menghapus data kriteria dari database
Hasil pengujian	Benar
Path/Jalur	162-163-164-165-166-167

3. Pengujian Lihat Lokasi Pembangunan Apotek

1. Fungsi fungsi tampil_lokasi_pembangunan_max_benefit()

a. Kode program fungsi tampil_lokasi_pembangunan_max_benefit()

```
475  public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_max_benefit(
476          String id_kriteria) {
477
478      ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
479      SQLiteDatabase database = this.getWritableDatabase();
480      Cursor cursor = database.rawQuery(
481              "SELECT max(nilai) as max FROM lokasi_pembangunan WHERE id_kriteria='"
482                      + id_kriteria + "'", null);
```

```
483
484         if (cursor.moveToFirst()) {
485             do {
486                 HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
487                 hashMapBenefit.put("max", cursor.getString(0));
488                 arrayListkecamatan.add(hashMapBenefit);
489
490             } while (cursor.moveToNext());
491         }
492
493         return arrayListkecamatan;
494     }
```

Gambar 4.20 Kode Program Fungsi Tampil_Lokasi_Pembangunan_Max_Benefit()

Gambar 4.20 Fungsi tampil_lokasi_max_benefit() merupakan kode program lihat pembangunan. Kode 475-494 adalah kode program untuk mengambil nilai max pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

b. Diagram Alir Kode Program

Gambar 4.21 Diagram Alir Tampil_Lokasi_Pembangunan_Max_Benefit()

Gambar 4.21 menjelaskan alur jalannya program berdasarkan kode program pada gambar 4.20. Kode program diubah kedalam alir untuk dicari nilai kompleksitas siklomatiknya.

c. Perhitungan *Cyclomatic Complexity* Kode Program

Perhitungan diagram alir menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

$$V(G) = E - N + 2 = 13 - 14 + 2 = 1$$

d. Pengujian Jalur Fungsi Tampil_Lokasi_Pembangunan_Max_Benefit()

Pengujian jalur jalur program berdasarkan perhitungan *Cyclomatic Complexity* adalah sebagai berikut :

Jalur 1 : 478-479-480-481-482-483-484-485-486-487-488-490-491-493-494

e. *Test Case* fungsi tampil_lokasi_pembangunan_max_benefit()

Tabel 4.15 *Test Case* fungsi tampil_lokasi_pembangunan_max_benefit()

Test Case function hapus_kriteria()	
Jalur 1	
Test Case	Fungsi tampil_lokasi_pembangunan_max_benefit()
Target yang diharapkan	Menghitung nilai max
Hasil pengujian	Benar
Path/Jalur	478-479-480-481-482-483-484-485-486-487-488-490-491-493-494

2. Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

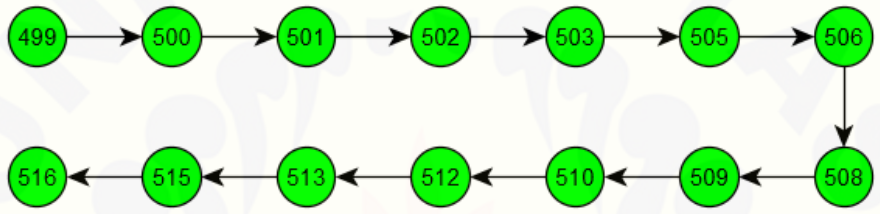
a. Kode Program Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

```
496    public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_min_cost(
497            String id_kriteria) {
498
499        ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
500        SQLiteDatabase database = this.getWritableDatabase();
501        Cursor cursor = database.rawQuery(
502                "SELECT min(nilai) as min FROM lokasi_pembangunan WHERE id_kriteria='"
503                        + id_kriteria + "'", null);
504
505        if (cursor.moveToFirst()) {
506            do {
507
508                HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
509                hashMapBenefit.put("min", cursor.getString(0));
510                arrayListkecamatan.add(hashMapBenefit);
511
512            } while (cursor.moveToNext());
513        }
514
515        return arrayListkecamatan;
516    }
```

Gambar 4.22 Kode Program Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

Gambar 4.22 Fungsi tampil_lokasi_min_cost() merupakan kode program lihat pembangunan apotek yang berfungsi agar *user* dapat melihat lokasi pembangunan apotek yang strategis. Kode 499-515 adalah kode program untuk mengambil nilai min pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

b. Diagram Alir Kode Program



Gambar 4.23 Diagram Alir Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

Gambar 4.23 menjelaskan alur jalannya program berdasarkan kode program pada gambar 4.22. Kode program diubah kedalam alir untuk dicari nilai kompleksitas siklomatiknya. Dari nilai kompleksitas siklomatik akan diketahui jalur dari alur jalannya sistem setiap *function*.

c. Perhitungan *Cyclomatic Complexity* Kode Program

Perhitungan diagram alir menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

$$V(G) = E - N + 2 = 13 - 14 + 2 = 1$$

d. Pengujian jalur fungsi tampil_lokasi_pembangunan_min_cost()

Pengujian jalur jalur berdasarkan perhitungan *Cyclomatic Complexity* adalah sebagai berikut :

Jalur 1 : 499-500-501-502-503-505-506-508-509-510-512-513-515-516

e. *Test Case* fungsi tampil_lokasi_pembangunan_max_benefit()

Tabel 4.16 *Test Case* manajemen kriteria

Test Case function hapus_kriteria()	
Jalur 1	
Test Case	Fungsi tampil_lokasi_pembangunan_min_cost()
Target yang diharapkan	Menghitung nilai min
Hasil pengujian	Benar
Path/Jalur	499-500-501-502-503-505-506-508-509-510-512-513-515-516

3. Fungsi Hitung()

a. Kode Program Fungsi Hitung()

```
178             // cek lokasi db
179             ArrayList<HashMap<String, String>> lokasi_pembangunan = db
180                     .cek_lokasi_pembangunan();
181
182             if (lokasi_pembangunan.isEmpty()) {
183                 Toast.makeText(getActivity().getBaseContext(),
184                         "Harap lakukan penilaian terlebih dahulu",
185                         Toast.LENGTH_LONG).show();
186             } else {
187                 System.out.println(rangking + " : Lokasi Terbaik");
188                 Fragment fragment = new c_u_hasillokasi();
189                 FragmentManager fragmentManager = getFragmentManager();
190                 fragmentManager.beginTransaction()
191                         .replace(R.id.frame_container_user, fragment)
192                         .addToBackStack(null).commit();
193
```

```
194                 ArrayList<HashMap<String, String>> arrayJumlahKriteria = db
195                         .tampil_semua_kriteria();
196                 ArrayList<HashMap<String, String>> arrayJumlahLokasi = db
197                         .jumlah_lokasi();
198                 double alt[][] = new double[arrayJumlahKriteria.size()][arrayJumlahLokasi
199                         .size()];
200                 ArrayList<HashMap<String, String>> arrayListKriteria = db
201                         .tampil_semua_kriteria();
202                 nilai_max = new ArrayList<String>();
203                 nilai_min = new ArrayList<String>();
204                 String simpan_id[] = new String[arrayJumlahLokasi
205                         .size()];
206
207                 if (arrayListKriteria.size() > 0) {
208                     for (int i = 0; i < arrayListKriteria.size(); i++) {
209
210                         HashMap<String, String> hashMapKriteria = arrayListKriteria
211                                 .get(i);
212                         String id_kriteria = hashMapKriteria
213                                 .get("id_kriteria");
214                         String bobot = hashMapKriteria.get("bobot");
215                         String kategori = hashMapKriteria
216                                 .get("kategori");
217                         ArrayList<HashMap<String, String>> arrayListNilaiMax = db
218                                 .tampil_lokasi_pembangunan_max_benefit(id_kriteria);
219                         HashMap<String, String> hashMapNilaiMax = arrayListNilaiMax
220                                 .get(0);
221                         max = hashMapNilaiMax.get("max");
222                         ArrayList<HashMap<String, String>> arrayListNilaiMin = db
223                                 .tampil_lokasi_pembangunan_min_cost(id_kriteria);
224
225                         HashMap<String, String> hashMapNilaiMin = arrayListNilaiMin
226                                 .get(0);
227                         min = hashMapNilaiMin.get("min");
228                         ArrayList<HashMap<String, String>> arrayListAlternatif = db
229                                 .tampil_lokasi_pembangunan(id_kriteria);
230
```

```
231                 if (arrayListAlternatif.size() > 0) {
232                     for (int j = 0; j < arrayListAlternatif
233                             .size(); j++) {
234
235                         HashMap<String, String> hashMapNormalisasi = arrayListAlternatif
236                                 .get(j);
237                         int nilai = Integer
238                                 .parseInt(hashMapNormalisasi
239                                         .get("nilai"));
240                         String id_kec = hashMapNormalisasi
241                                 .get("id_kecamatan");
242                         // nilai preferensi benefit
243                         if (kategori
244                                 .equalsIgnoreCase("benefit")) {
245                             alt[i][j] = (nilai / Double
246                                     .parseDouble(max))
247                                     * Double.parseDouble(bobot);
248                         }
249                         // nilai preferensi cost
250                         else {
251                             alt[i][j] = (Double
252                                     .parseDouble(min) / nilai)
253                                     * Double.parseDouble(bobot);
```

```
254                         }
255
256                         System.out.println("v = " + " "
257                                 + alt[i][j] + "cell [" + i
258                                 + "][" + j + "]");
259                     }
260                 }
261
262             }
263         }
264         double nilai_total[] = new double[arrayJumlahLokasi
265                 .size()];
266         rangking = simpan_id[0];
267         double temp_total = 0;
```

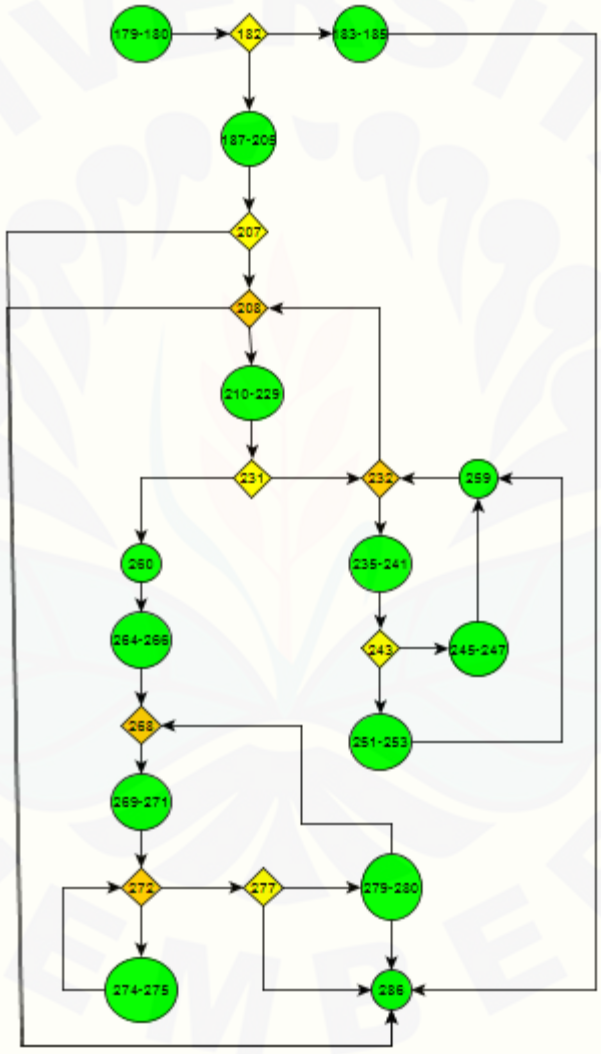
```
268                 for (int j = 0; j < arrayJumlahLokasi.size(); j++) {
269                     double tempnilai = 0;
270                     simpan_id[j] = c_u_tlp.id_kecamatan.get(j);
271                     Log.i(simpan_id[j], "ini id kec");
272                     for (int k = 0; k < arrayJumlahKriteria.size(); k++) {
273                         //total nilai preferensi tiap lokasi
274                         nilai_total[j] = tempnilai + alt[k][j];
275                         tempnilai = nilai_total[j];
276                     }
277                     if (nilai_total[j] > temp_total) {
278                         //Lokasi pembangunan apotek
279                         rangking = simpan_id[j];
280                         temp_total = nilai_total[j];
281                     }
282 
283                     System.out.println(nilai_total[j] + " "
284                             + " : nilai alt " + j);
285                 }
286                 System.out.println(rangking + " "
287                         + " ini adalah hasil akhir");
288             }
289         }
```

Gambar 4.24 Kode Program Fungsi Hitung()

Gambar 4.24. Kode 496-516 adalah kode program untuk mengambil nilai min pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut juga berada pada kelas model. Selanjutnya digunakan kelas c_fu_penilaian(). Kode 179-185 adalah kode untuk mengecek apakah *user* sudah melakukan penilaian. Jika *user* belum melakukan penilaian pada lokasi maka akan menampilkan pesan "Harus melakukan penilaian terlebih dahulu". Jika *user* telah melakukan penilaian, maka kode program yang dijalankan adalah kode 187-289. Kode 217-229 adalah kode program untuk mengambil nilai max dan min tiap kriteria

yang telah dihitung pada kelas model. Kode 243-253 adalah kode program untuk menghitung nilai preferensi *benefit* dan *cost*. Kode 274 adalah kode program untuk menjumlahkan nilai preferensi pada tiap lokasi. Kemudian pada kode 279-281, lokasi yang memiliki total nilai preferensi paling besar dijadikan solusi pembangunan apotek yang strategis.

b. Diagram Alir Fungsi Hitung()



Gambar 4.25 Diagram Alir Fungsi Hitung()

Gambar 4.25 menjelaskan alur jalannya program berdasarkan kode program pada gambar 4.24. Kode program diubah kedalam alir untuk dicari nilai kompleksitas siklomatiknya. Dari nilai kompleksitas siklomatik akan diketahui jalur dari alur jalannya sistem setiap *function*.

c. Perhitungan *Cyclomatic Complexity* Kode Program Fungsi Hitung()

Perhitungan diagram alir pada fungsi hitung() menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

$$V(G) = E – N + 2 = 27 – 23 + 2 = 6$$

d. Pengujian jalur Lihat Lokasi Pembangunan Apotek

Pengujian jalur program berdasarkan perhitungan *Cyclomatic Complexity* adalah sebagai berikut ::

Jalur 1 : (179-180) - 182 - (183-185) – 286

Jalur 2 : (179-180) - (187-205) - 207 – 260 – (264 – 266) – 268 - (269 – 271) – 272 – (274-275) – 272 – 277 - 286

Jalur 3 : (179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 232 – (235-241) – 243 – (245 – 247) – 259 – 208 – 260 – (264 – 266) – 268 - (269 – 271) – 272 – (274-275) – 272 – 277 – (279 – 280) -286

Jalur 4 : (179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 232 – (235-241) – 243 – (251-253) – 259 – 208 – 260 – (264 – 266) – 268 - (269 – 271) – 272 – (274-275) – 272 – 277 – (279 – 280) -286

Jalur 5 : (179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 260 – (264 – 266) – 208 – (209 – 271) – 272 – (274 – 275) - 277 – 286

Jalur 6 : (179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 260 – (264 – 266) – 208 – (209 – 271) – 272 – (274 – 275) - 277 – (279 – 280) – 286

e. *Test Case* Fungsi hitung()

Tabel 4.17 *Test Case* fungsi hitung()

Test Case function hitung()	
Jalur 1	
Test Case	Penanganan jika belum dilakukan penilaian
Target yang diharapkan	Menampilkan pesan "Harap melakukan penilaian terlebih dahulu"
Hasil pengujian	Benar
Path/Jalur	(179-180) - 182 - (183-185) – 286
Jalur 2	
Test Case	Jika tidak ada data kriteria
Target yang diharapkan	Perhitungan tidak dilanjutkan
Hasil pengujian	Benar
Path/Jalur	(179-180) - (187-205) - 207 – 286
Jalur 3	
Test Case	Menghitung nilai preferensi *Benefit*
Target yang diharapkan	Diperoleh nilai preferensi *benefit*
Hasil pengujian	Benar
Path/Jalur	(179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 232 – (235-241) – 243 – (245 – 247) – 259 – 208 – 260 – (264 – 266) – 268 - (269 – 271) – 272 – (274-275) – 272 – 277 – (279 – 280) -286
Jalur 4	
Test Case	Menghitung nilai preferensi *cost*
Target yang diharapkan	Diperoleh nilai preferensi *cost*
Hasil pengujian	Benar

Path/Jalur	(179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 232 – (235-241) – 243 – (251-253) – 259 – 208 – 260 – (264 – 266) – 268 - (269 – 271) – 272 – (274-275) – 272 – 277 – (279 – 280) -286
Jalur 5	
Test Case	Menghitung total nilai preferensi tiap alternatif
Target yang diharapkan	Diperoleh total nilai preferensi tiap alternatif
Hasil pengujian	Benar
Path/Jalur	(179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 260 – (264 – 266) – 208 – (209 – 271) – 272 – (274 – 275) - 277 – 286
Jalur 6	
Test Case	Mencari lokasi yang memiliki total nilai preferensi terbesar
Target yang diharapkan	Diperoleh lokasi yang memiliki total nilai preferensi terbesar
Hasil pengujian	Benar
Path/Jalur	(179-180) - (187-205) - 207 – 208 – (210-229) – 231 – 260 – (264 – 266) – 208 – (209 – 271) – 272 – (274 – 275) - 277 – (279 – 280) - 286

4.5.2 *Black Box Testing*

Pengujian *Black Box* menitikberatkan pada fungsionalitas sistem. Pengujian ini tidak melihat kinerja *internal* dari sistem, jadi hanya berfokus pada kinerja sistem sesuai dengan spesifikasi dan kebututuhan yang dianalisis pada bab perancangan. Pengujian *Black Box* dilakukan oleh *owner* apotek median farma. Hasil pengujian *Black Box* dapat dilihat pada Tabel 4.18.

Tabel 4.18 Pengujian *Black Box*

No.	Fitur	Action	Hasil yang Diharapkan	Kesimpulan
1.	*Login*	1. *Input username* dan *password* yang benar, kemudian klik *login*. 2. *Login* sebagai *admin* *Username* : admin *Password* : 1	1. *Login* sukses 2. Masuk ke sistem sebagai a*dmin*	[√] Berhasil [ ] Gagal
		1. *username* atau *password* salah 2. Field *username* atau *password* kosong	1. Menampilkan pesan *username* atau *password* salah 2. Menampilkan notifikasi tidak boleh ada yang kosong	[√] Berhasil [ ] Gagal
2.	*Logout*	1. Klik menu *logout*	1. Keluar dari hak akses yang dimiliki dan kembali ke halaman home *user*.	[√] Berhasil [ ] Gagal
3.	Tambah kriteria	1. Mengisi form tambah kriteria, kemudian klik simpan.	1. Menyimpan kriteria ke *database*. Kemudian menampilkan data kriteria yang telah di tambah.	[√] Berhasil [ ] Gagal
		2. Salah satu field yang harus diisi	2. Menampilkan notifikasi tidak boleh ada yang	[√] Berhasil [ ] Gagal

			dikosongi		kosong	
4.	Hapus kriteria	1.	Klik tombol delete pada salah satu kriteria yang akan dihapus	1.	Menghapus kriteria dari *database* dan menampilkan halaman kriteria	[√] Berhasil [ ] Gagal
5.	Pilih rencana lokasi pembangunan apotek	1.	Memilih rencana lokasi pembangunan apotek kemudian klik tombol pilih	1.	Menampilkan rencana lokasi pembangunan apotek yang telah dipilih	[√] Berhasil [ ] Gagal
		2.	Tidak memilih rencana lokasi pembangunan apotek kemdian klik tombol pilih	2.	Menampilkan pesan harus memilih lokasi	[√] Berhasil [ ] Gagal
6.	Penilaian lokasi	1.	Menilai lokasi kemudian klik tombol proses	1.	Menampilkan lokasi pembangunan apotek yang strategis	[√] Berhasil [ ] Gagal
		2.	Belum menilai lokasi kemudian klik proses	[√] Berhasil [ ] Gagal		

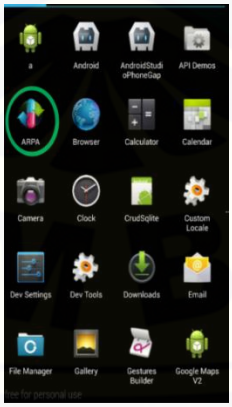
BAB 5. HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil pembuatan sistem dan juga pembahasannya. Penelitian ini menghasilkan sebuah sistem perencanaan pembangunan apotek yang dapat mencari lokasi strategis dalam membangun apotek. Pembahasan bertujuan untuk menjelaskan bagaimana penelitian ini menjawab perumusan masalah serta tujuan dan manfaat dari sistem ini.

5.1 SDSS Perencanaan Pembangunan Apotek

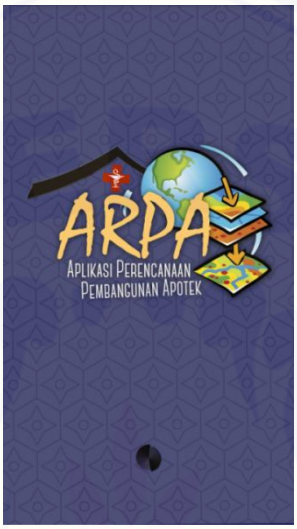
SDSS Perencanaan Pembangunan Apotek dibangun untuk dapat menemukan lokasi yang strategis dalam pembangunan sebuah apotek. Untuk menemukan lokasi yang strategis, diperlukan kriteria-kriteria yang dijadikan sebagai acuan dalam penilaian untuk menentukan lokasi pembangunan apotek yang strategis. Kriteria-kriteria tersebut dapat ditambahkan melalui menu kriteria pada *admin*. Setelah itu memilih lokasi yang akan dijadikan sasaran dan memberikan penilaian terhadap lokasi-lokasi tersebut. Kemudian lokasi pembangunan yang strategis akan tampil pada peta. Langkah-langkah menggunakan sistem adalah sebagai berikut :

1. Buka Launcher ARPA (Aplikasi Perencanaan Pembangunan Apotek) seperti pada gambar 5.1.



Gambar 5.1 Launcher ARPA

2. Menampilkan *Splash Screen* dengan durasi waktu 3 detik kemudian akan muncul tampilan *home user* seperti gambar 5.2.

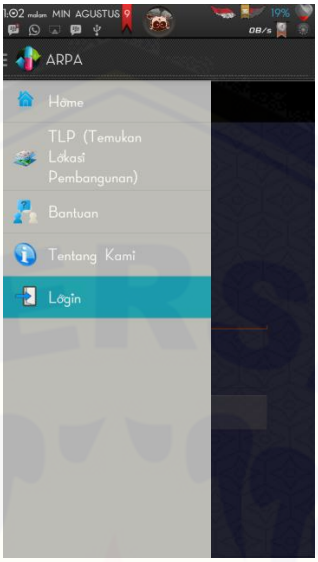


Gambar 5.2 Tampilan *Splash Screen*

3. Menampilkan *Home User* seperti gambar 5.3.

Gambar 5.3 Tampilan *Home User*

4. Pilih Menu Login seperti gambar 5.4.

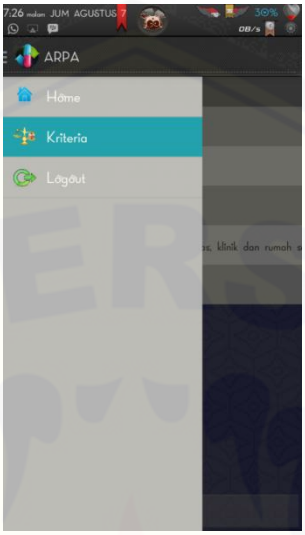


Gambar 5.4 Tampilan Menu *User*

5. Melakukan Login sebagai admin (username = a, password = 1) pada form login seperti pada gambar 5.5.

Gambar 5.5 Tampilan Halaman Login

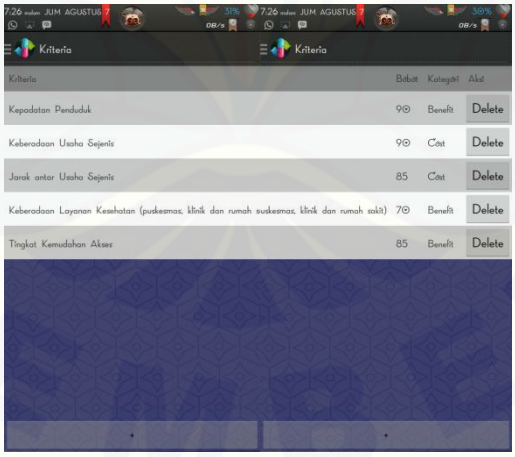
6. Masuk pada Menu Kriteria seperti pada gambar 5.6.



Gambar 5.6 Tampilan Menu *Admin*

7. Tampil Halaman Kriteria

Tekan tombol "+" pada halaman kriteria jika ingin menambahkan kriteria dan tekan tombol *delete* pada kriteria yang ingin dihapus seperti pada gambar 5.7.



Gambar 5.7 Tampilan Halaman Kriteria *Scroll* Kiri dan *Scroll* Kanan

8. Tekan Tombol “+” maka akan tampil Halaman Form Tambah Kriteria

Gambar 5.8 Tampil Halaman Tambah Kriteria

Gambar 5.8 adalah Halaman form tambah kriteria. Isi Form Kriteria sesuai dengan kriteria yang menjadi acuan dalam pembangunan apotek sesuai kebutuhan dan keinginan. Kemudian tekan tombol simpan.

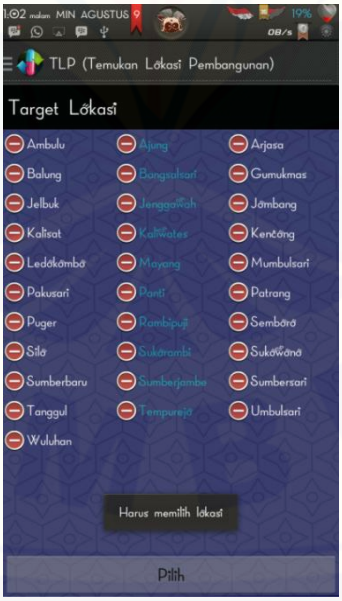
9. Pilih Menu Logout dari hak akses *admin* seperti pada gambar 5.9.

Gambar 5.9 Tampil Logout

10. Pilih Menu TLP (Temukan Lokasi Pembangunan) seperti pada gambar 5.10.

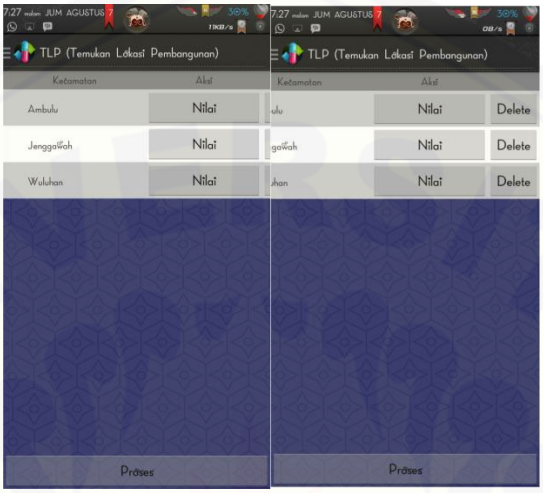
Gambar 5.10 Menu *User* TLP (Temukan Lokasi Pembangunan)

11. Pilih lokasi yang dijadikan sebagai rencana lokasi pembangunan apotek pada halaman target lokasi seperti pada gambar 5.11.



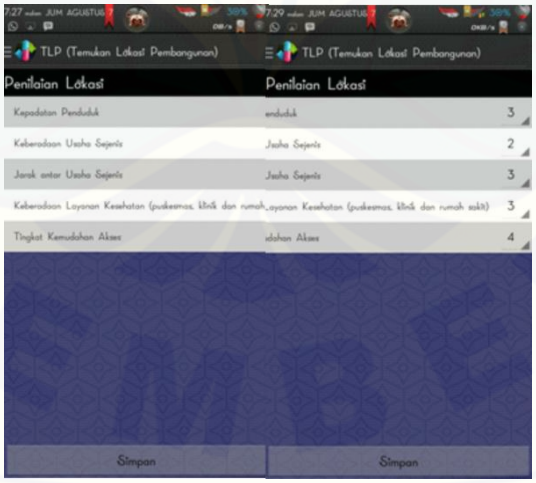
Gambar 5.11 Halaman TLP (Temukan Lokasi Pembangunan)

12. Tampil halaman rencana lokasi pembangunan. Tekan tombol nilai untuk melakukan penilaian atau tekan tombol hapus untuk menghapus rencana lokasi pembangunan apotek yang tidak diinginkan pada halaman seperti gambar 5.12.



Gambar 5.12 Halaman Rencana Lokasi Pembangunan *Scroll* Kiri dan *Scroll* Kanan

13. Tampil Halaman Penilaian. Lakukan penilaian dengan mengisi form penilaian dan tekan tombol simpan seperti gambar 5.13.



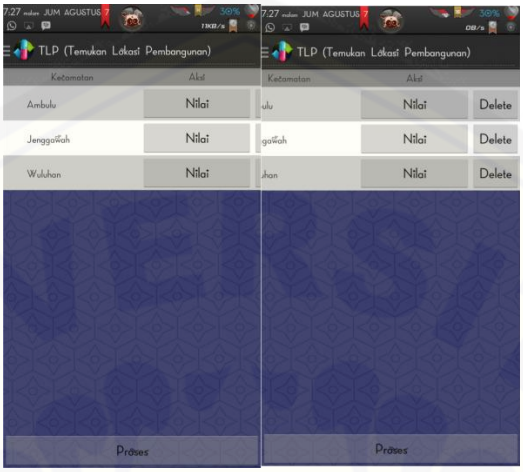
Gambar 5.13 Halaman Penilaian *Scroll* Kiri dan *Scroll* Kanan

Gambar 5.13 merupakan form penilaian lokasi rencana penempatan apotek. Penilaian dapat dilakukan dengan pemilihan nilai 1 sampai 5. Nilai 1 adalah nilai terendah dan nilai 5 adalah nilai yang tertinggi. Kepadatan penduduk adalah perbandingan antara jumlah penduduk dengan luas wilayah yang dihuni (Ida Bagoes Mantra, 2007). Ukuran yang biasa digunakan adalah jumlah penduduk setiap satu Km^2. Keberadaan sejenis adalah keberadaan apotek dalam suatu kecamatan. Jarak antar usaha sejenis adalah jarak antar usaha apotek. Keberadaan Layanan Kesehatan adalah keberadaan layanan kesehatan seperti puskesmas, rumah sakit atau klinik dalam suatu kecamatan. Tingkat kemudahan akses adalah banyaknya jalur yang dapat digunakan untuk menuju lokasi pembangunan apotek. Keterangan penilaian dapat dilihat pada tabel 1.

Table 1 Keterangan Penilaian

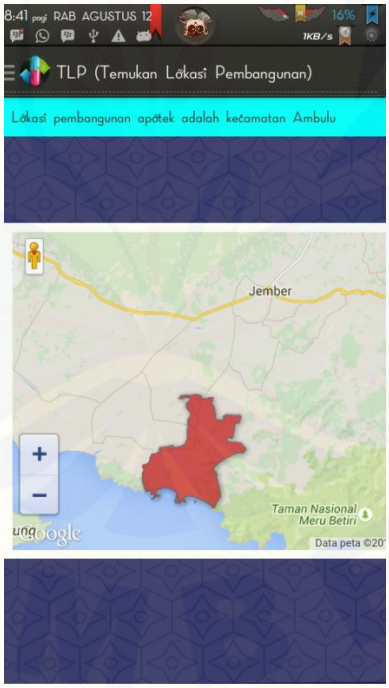
	Keterangan Nilai				
Kriteria	1	2	3	4	5
Kepadatan Penduduk	<100 jiwa / km^2	101-200 jiwa / km^2	201-300 jiwa / km^2	301-400 jiwa / km^2	401-500 jiwa / km^2
Keberadaan Usaha Sejenis	1 apotek	2 apotek	3 apotek	4 apotek	>5 apotek
Jarak antar Usaha Sejenis	1 km	2 km	3 km	4 km	>5 km
Keberadaan Layanan Kesehatan (puskesmas, klinik dan rumah sakit)	1 layanan kesehatan	2 layanan kesehatan	3 layanan kesehatan	4 layanan kesehatan	5 layanan kesehatan
Tingkat Kemudahan Akses	1 jalur	2 jalur	3 jalur	4 jalur	>5 jalur

14. Tampil Halaman Lokasi seperti pada gambar 5.14. Lakukan penilaian pada semua lokasi. Kemudian tekan tombol proses.



Gambar 5.14 Halaman Rencana Lokasi Pembangunan *Scroll* Kiri dan *Scroll* Kanan

15. Tampil Lokasi Pembangunan Apotek yang Strategis seperti pada gambar 5.15.



Gambar 5.15 Tampil Lokasi Pembangunan Apotek yang Strategis

5.2 Hasil Implementasi SAW pada Sistem

```
475  public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_max_benefit(
476          String id_kriteria) {
477
478      ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
479      SQLiteDatabase database = this.getWritableDatabase();
480      Cursor cursor = database.rawQuery(
481              "SELECT max(nilai) as max FROM lokasi_pembangunan WHERE id_kriteria='"
482                      + id_kriteria + "'", null);
483
484      if (cursor.moveToFirst()) {
485          do {
486              HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
487              hashMapBenefit.put("max", cursor.getString(0));
488              arrayListkecamatan.add(hashMapBenefit);
489
490          } while (cursor.moveToNext());
491      }
492
493      return arrayListkecamatan;
494  }
```

Gambar 5.16 Kode Program Fungsi Tampil_Lokasi_Pembangunan_Max_Benefit()

Gambar 5.16 Kode tersebut fungsi tampil_lokasi_max_benefit() merupakan kode program lihat pembangunan. Kode 475-494 adalah kode program untuk mengambil nilai max pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

```
496    public ArrayList<HashMap<String, String>> tampil_lokasi_pembangunan_min_cost(
497            String id_kriteria) {
498
499        ArrayList<HashMap<String, String>> arrayListkecamatan = new ArrayList<HashMap<String, String>>();
500        SQLiteDatabase database = this.getWritableDatabase();
501        Cursor cursor = database.rawQuery(
502                "SELECT min(nilai) as min FROM lokasi_pembangunan WHERE id_kriteria='"
503                        + id_kriteria + "'", null);
504
505        if (cursor.moveToFirst()) {
506            do {
507
508                HashMap<String, String> hashMapBenefit = new HashMap<String, String>();
509                hashMapBenefit.put("min", cursor.getString(0));
510                arrayListkecamatan.add(hashMapBenefit);
511
512            } while (cursor.moveToNext());
513        }
514
515        return arrayListkecamatan;
516    }
```

Gambar 5.17 Kode Program Fungsi Tampil_Lokasi_Pembangunan_Min_Cost()

Gambar 5.17 Kode tersebut fungsi tampil_lokasi_min_cost() merupakan kode program lihat pembangunan. Kode 499-515 adalah kode program untuk mengambil nilai max pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut berada pada kelas model.

```
178             // cek lokasi db
179             ArrayList<HashMap<String, String>> lokasi_pembangunan = db
180                     .cek_lokasi_pembangunan();
181
182             if (lokasi_pembangunan.isEmpty()) {
183                 Toast.makeText(getActivity().getBaseContext(),
184                         "Harap lakukan penilaian terlebih dahulu",
185                         Toast.LENGTH_LONG).show();
186             } else {
187                 System.out.println(rangking + " : Lokasi Terbaik");
188                 Fragment fragment = new c_u_hasillokasi();
189                 FragmentManager fragmentManager = getFragmentManager();
190                 fragmentManager.beginTransaction()
191                         .replace(R.id.frame_container_user, fragment)
192                         .addToBackStack(null).commit();
193
194                 ArrayList<HashMap<String, String>> arrayJumlahKriteria = db
195                         .tampil_semua_kriteria();
196                 ArrayList<HashMap<String, String>> arrayJumlahLokasi = db
197                         .jumlah_lokasi();
198                 double alt[][] = new double[arrayJumlahKriteria.size()][arrayJumlahLokasi
199                         .size()];
200                 ArrayList<HashMap<String, String>> arrayListKriteria = db
201                         .tampil_semua_kriteria();
202                 nilai_max = new ArrayList<String>();
203                 nilai_min = new ArrayList<String>();
204                 String simpan_id[] = new String[arrayJumlahLokasi
205                         .size()];
206
207                 if (arrayListKriteria.size() > 0) {
208                     for (int i = 0; i < arrayListKriteria.size(); i++) {
209
210                         HashMap<String, String> hashMapKriteria = arrayListKriteria
211                                 .get(i);
212                         String id_kriteria = hashMapKriteria
213                                 .get("id_kriteria");
214                         String bobot = hashMapKriteria.get("bobot");
215                         String kategori = hashMapKriteria
```

```
                .get("kategori");
        ArrayList<HashMap<String, String>> arrayListNilaiMax = db
                .tampil_lokasi_pembangunan_max_benefit(id_kriteria);
        HashMap<String, String> hashMapNilaiMax = arrayListNilaiMax
                .get(0);
        max = hashMapNilaiMax.get("max");
        ArrayList<HashMap<String, String>> arrayListNilaiMin = db
                .tampil_lokasi_pembangunan_min_cost(id_kriteria);

        HashMap<String, String> hashMapNilaiMin = arrayListNilaiMin
                .get(0);
        min = hashMapNilaiMin.get("min");
        ArrayList<HashMap<String, String>> arrayListAlternatif = db
                .tampil_lokasi_pembangunan(id_kriteria);

        if (arrayListAlternatif.size() > 0) {
            for (int j = 0; j < arrayListAlternatif
                    .size(); j++) {

                HashMap<String, String> hashMapNormalisasi = arrayListAlternatif
                        .get(j);
                int nilai = Integer
                        .parseInt(hashMapNormalisasi
                                .get("nilai"));
                String id_kec = hashMapNormalisasi
                        .get("id_kecamatan");
                // nilai preferensi benefit
                if (kategori
                        .equalsIgnoreCase("benefit")) {
                    alt[i][j] = (nilai / Double
                            .parseDouble(max))
                            * Double.parseDouble(bobot);
                }
                // nilai preferensi cost
                else {
                    alt[i][j] = (Double
                            .parseDouble(min) / nilai)
                            * Double.parseDouble(bobot);
```

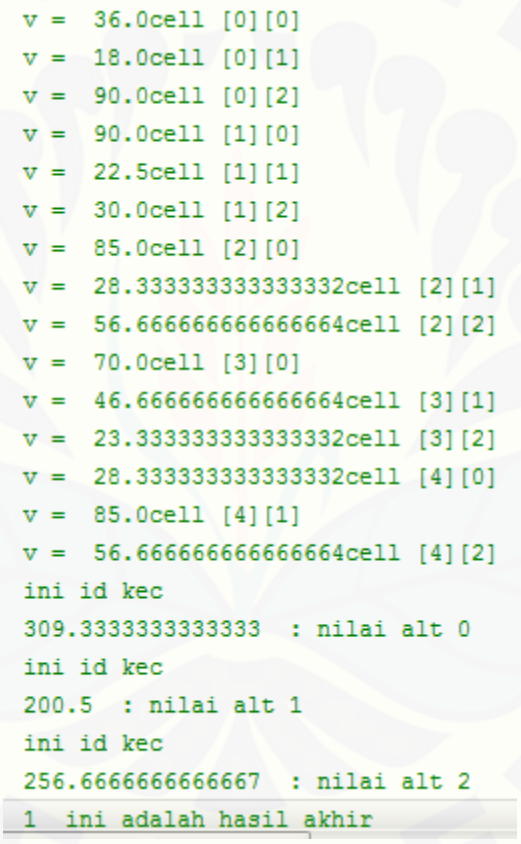
```
254                         }
255
256                         System.out.println("v = " + " "
257                                 + alt[i][j] + "cell [" + i
258                                 + "][" + j + "]");
259                     }
260                 }
261
262             }
263         }
264         double nilai_total[] = new double[arrayJumlahLokasi
265                 .size()];
266         rangking = simpan_id[0];
267         double temp_total = 0;
268         for (int j = 0; j < arrayJumlahLokasi.size(); j++) {
269             double tempnilai = 0;
270             simpan_id[j] = c_u_tlp.id_kecamatan.get(j);
271             Log.i(simpan_id[j], "ini id kec");
272             for (int k = 0; k < arrayJumlahKriteria.size(); k++) {
273                 //total nilai preferensi tiap lokasi
274                 nilai_total[j] = tempnilai + alt[k][j];
275                 tempnilai = nilai_total[j];
276             }
277             if (nilai_total[j] > temp_total) {
278                 //Lokasi pembangunan apotek
279                 rangking = simpan_id[j];
280                 temp_total = nilai_total[j];
281             }
282
283             System.out.println(nilai_total[j] + " "
284                     + " : nilai alt " + j);
285         }
286         System.out.println(rangking + " "
287                 + " ini adalah hasil akhir");
288     }
289 }
```

Gambar 5.18 Kode Program Fungsi Hitung()

Gambar 5.18. Kode 496-516 adalah kode program untuk mengambil nilai min pada tiap kriteria yang kemudian digunakan untuk menghitung rating kinerja (r). Kode program tersebut juga berada pada kelas model. Selanjutnya digunakan kelas c_fu_penilaian(). Kode 179-185 adalah kode untuk mengecek apakah *user* sudah melakukan penilaian. Jika *user* belum melakukan penilaian pada lokasi maka akan menampilkan pesan "Harus melakukan penilaian terlebih dahulu". Jika *user* telah melakukan penilaian, maka kode program yang dijalankan adalah kode 187-289.

Kode 217-229 adalah kode program untuk mengambil nilai max dan min tiap kriteria yang telah dihitung pada kelas model. Kode 243-253 adalah kode program untuk menghitung nilai preferensi *benefit* dan *cost*. Kode 274 adalah kode program untuk menjumlahkan nilai preferensi pada tiap lokasi. Kemudian pada kode 279-281, lokasi yang memiliki total nilai preferensi paling besar dijadikan solusi pembangunan apotek yang strategis.

Hasil Keluaran Hitungan pada *Eclipse*



Gambar 5.19 Hasil Keluaran Perhitungan

Gambar 5.19 adalah hasil keluaran dari perhitungan program. Hasil tersebut telah sama dengan perhitungan manual yaitu seperti pada tabel 4.7. Jadi implementasi SAW pada sistem telah benar.

BAB 6. PENUTUP

Pada bab ini merupakan bagian akhir di dalam penulisan skripsi, berisi tentang kesimpulan dan saran. Kesimpulan yang ditulis merupakan kesimpulan dari hasil penelitian yang telah dilakukan dan saran lanjutan untuk dilakukan pada penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan dari penelitian yang telah dilakukan adalah sebagai berikut:

a. Sistem dapat membantu dalam mengambil keputusan lokasi pembangunan apotek dengan fitur pengolahan kriteria yang ada pada *admin*. Data kriteria tersebut digunakan oleh *user* dalam menilai lokasi-lokasi yang dijadikan sebagai alternatif lokasi pembangunan. Kemudian data penilaian *user* tersebut diolah menggunakan metode *Simple Additive Weighting* (SAW) dan menghasilkan keputusan lokasi pembangunan apotek yang strategis.
b. Sistem dapat menghasilkan keputusan dengan menerapkan metode *Simple Additive Weighting* (SAW). Keputusan didapat dari hasil nilai-nilai dari *rating* kinerja yang diolah menjadi nilai-nilai preferensi. Total nilai preferensi tiap alternatif yang terbaik adalah hasil keputusan lokasi penempatan apotek. Pada pembahasan, lokasi yang menjadi hasil keputusan adalah lokasi yang memiliki nomor 1 karena memiliki nilai preferensi paling tinggi yaitu 309,33.

6.2 Saran

a. Menerapkan sistem ini pada wilayah lain yang ada di luar Kabupaten Jember.
b. Menerapkan metode perangkingan lain agar dapat diciptakan perbandingan antar dua metode.
c. Pengembangan lebih lanjut untuk penelitian ini dapat dilakukan dengan membangun sistem ini dalam *platform* lain seperti iOS, *windows phone* dan lain sebagainnya.

DAFTAR PUSTAKA

Al-Bahra. (2006). *Rekayasa Perangkat Lunak.* Yogyakarta: Graha Ilmu.

Beizer, B. (1990). *Software Testing Techniques.* New York: Van Nostrand Reinhold.

Hasan, I. (2002). *Pokok-pokok Pengambilan Keputusan.* Jakarta: Ghalia Indonesia.

Hikmat, M. M. (2011). *Metode Penelitian dalam Prespektif Ilmu Komunikasi dan Sastra.* Yogyakarta: Graha Ilmu.

Huda, A. A. (2013). *Live Coding! 9 Aplikasi Android Buatan Sendiri.* Yogyakarta: ANDI.

Indarto. (2009). *Buku Ajar Dasar-Dasar Sistem Informasi Geografis.* Yogyakarta: JEMBER UNIVERSITY PRESS.

Irwansyah, E. (2013). *Sistem Informasi Geografi : Prinsip Dasar dan Penbembangan Aplikasi.* Yogyakarta: Digibooks.

Kasman, A. D. (2013). *Kolaborasi Dahsyat Android Dengan Php & Mysql.* Yogyakarta: Lokomedia.

Marimin. (2004). *Teknik dan Aplikasi Pengambilan Keputusan Kriteria Majemuk.* Jakarta: Grasindo.

Riyanto. (2010). *Membuat Sendiri Aplikasi Mobile Platform Java Me, Blackberry & Android.* Yogyakarta: ANDI.

Sri Kusumadewi, d. (2006). *Fuzzy Mult-Attribute Decision Making (Fuzzy MADM).* Yogyakarta: Graha Ilmu.

Sugiyono. (2012). *Model Peta Digital Rawan Sambaran Petir dengan Menggunakan Metode SAW (Simple Additive Weighting) Studi Kasus Propinsi Lampung*. *Telematika mkom*, Vol. 4 No. 1.

Suprianto, D., & Agustina, R. (2013). *Pemrograman Aplikasi Android*. Yogyakarta: MediaKom.

Syamsuni. (2006). *Farmasetika Dasar dan Hitungan Farmasi*. Jakarta: Buku Kedokteran EGC.