SISTEM PAKAR DIAGNOSA PENYAKIT PADA ULAR DENGAN METODE *FORWARD CHAINING* BERBASIS ANDROID

(Skripsi)

Oleh

Tegar Ageng Khresna Ferrial

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

2019

ABSTRAK

SISTEM PAKAR DIAGNOSA PENYAKIT PADA ULAR DENGAN METODE *FORWARD CHAINING* BERBASIS ANDROID

Oleh

TEGAR AGENG KHRESNA FERRIAL

Penyakit ular merupakan sesuatu yang sangat tidak diinginkan oleh pemelihara ular khususnya mereka yang menangkarkan ular dalam jumlah yang banyak karena dapat menyebabkan kecacatan maupun kematian pada ular. Pakar ular yang sedikit dan kurangnya informasi cara penanggulangan penyakit dapat mengakibatkan banyak kerugian, oleh karna itulah dibutuhkan sistem pakar untuk mendiagnosa penyakit pada ular. Sistem pakar ini dibuat dengan tujuan untuk membantu pemelihara awam maupun peternak ular dalam mendiagnosa penyakit ular. Sistem pakar yang dibangun dalam penelitian ini menggunakan metode *forward chaining* dengan basis *android.* Sistem ini dapat mendiagnosa 12 jenis penyakit dengan 49 gejala penyakit pada ular. Hasil pengujian diagnosis yang dilakukan oleh sistem telah sesuai dengan diagnosis manual yang dilakukan oleh pakar dengan persentase akurasi sebesar 94%.

Kata kunci : *Forward Chaining*, Penyakit Ular, Sistem Pakar, Android

ABSTRACT

DISEASE DIAGNOSIS EXPERT SYSTEM ON SNAKES WITH FORWARD CHAINING METHOD BASED ON ANDROID

By

TEGAR AGENG KHRESNA FERRIAL

Snake disease is something that is very unwanted by snake keepers, especially those who breed snakes in large quantities because they can cause disability or death in snakes. The very small number of snake experts and lack of information on how to deal with snake disease can cause many disadvantages, therefore an expert system is needed to diagnose snake disease. This expert system was created with the aim of helping lay keepers and snake breeders to diagnose snake disease. The Expert System built in this study uses the forward chaining method with an Android base. This system can diagnose 12 types of diseases with 49 symptoms of snake disease. The results of diagnostic tests performed by the system are in accordance with the manual diagnosis carried out by experts with a percentage of accuracy of 94%.

Keywords : Forward Chaining, Snake Disease, Expert System, Android

SISTEM PAKAR DIAGNOSA PENYAKIT PADA ULAR DENGAN METODE *FORWARD CHAINING* BERBASIS ANDROID

Oleh

TEGAR AGENG KHRESNA FERRIAL

Skripsi

Sebagai Salah Satu Syarat untuk Memperoleh Gelar

SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

2019

Judul Skripsi	:	**SISTEM PAKAR DIAGNOSA PENYAKIT PADA ULAR DENGAN METODE *FORWARD CHAINING* BERBASIS ANDROID**
Nama Mahasiswa	:	**Tegar Ageng Khresna Ferrrial**
NPM	:	1317051063
Program Studi	:	S1 Ilmu Komputer
Jurusan	:	Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Pembimbing I	Pembimbing II
Rico Andrian, S.Si., M.Kom.	**Bambang Hermanto, S.Kom., M.Cs.**
NIP 19750627 200501 1 001	NIP 19790912 200812 1 002

2. Ketua Jurusan Ilmu Komputer

Dr. Ir. Kurnia Muludi, M.S.Sc.
NIP 19640616 198902 1 001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Rico Andrian, S.Si., M.Kom.** .....

   Sekertaris : **Bambang Hermanto, S.Kom., M.Cs.** .....

   Penguji
   Bukan Pembimbing : **Dr. Ir. Kurnia Muludi, M.S.Sc.** .....

2. PLT Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Prof. Sutopo Hadi, M.Sc., Ph.D.
NIP 19710415 199512 1 001

Tanggal Lulus Ujian Skripsi : 31 Januari 2019

PERNYATAAN SKRIPSI MAHASISWA

Yang bertanda tangan dibawah ini

Nama	: Tegar Ageng Khresna Ferrial
Nomor Pokok Mahasiswa	: 1317051063
Program Studi	: S1 Ilmu Komputer

Dengan ini menyatakan bahwa benar ini adalah penelitian saya sendiri. Sepengetahuan saya, pembahasan materi dalam laporan penelitian ini belum pernah dipublikasikan atau ditulis oleh orang lain atau dipergunakan dan diterima sebagai persyaratan penyelesaian studi.

Bandarlampung, 31 Januari 2019
Yang Menyatakan

Tegar Ageng Khresna Ferrial
NPM 1317051063

RIWAYAT HIDUP

Penulis dilahirkan di TanjungKarang, Bandar Lampung pada tanggal 19 November 1996, sebagai anak pertama dari empat bersaudara dengan Ayah bernama R. Ferry Indarmawan dan Ibu bernama Lia Mutiara Sari.

Penulis memiliki tiga orang adik bernama Vidia Anugerah Ayu Tavia Ferrial, Nindya Ramadhanti Indah Tantia Ferrial dan Aurora Chantika Chaturisa Ferrial. Penulis menyelesaikan Taman Kanak-Kanak (TK) pada tahun 2002 di TK Kartika 26 Bandar Lampung, Sekolah Dasar (SD) Negeri Percontohan Kebun Bunga 4 Banjarmasin pada tahun 2008, Sekolah Menengah Pertama (SMP) Negeri 6 Banjarmasin pada tahun 2010, dan Sekolah Menengah Atas (SMA) Negeri 2 Bandar Lampung pada tahun 2013.

Pada tahun 2013, penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur SBMPTN. Pada bulan Januari-Februari 2016, penulis melakukan kerja praktik di DISKOMINFO Provinsi Lampung. Pada bulan Juli-Agustus 2016, penulis melakukan Kuliah Kerja Nyata selama 40 hari di Desa Kediri Kabupaten Gading Rejo.

Selama menjadi mahasiswa penulis aktif dalam Organisasi Himpunan Mahasiswa Jurusan Ilmu Komputer (Himakom) Universitas Lampung dengan menjabat sebagai Anggota Biro Kewirausahaan pada tahun 2013-2014 dan Kepala Biro Kewirausahaan pada tahun 2014-2015.

MOTO

“No Guts No Glory”

(Tegar A.KFerrial)

"Kejarlah kesuksesan dengan caramu sendiri, selagi itu baik dan halal lakukanlah. Sukseslah dengan tidak berpatokan pada hasil orang lain agar kamu selalu bersyukur."

(Tegar A.KFerrial)

“Jadikanlah sabar dan sholat sebagai penolongmu, Sesungguhnya Allah bersama orang-orang yang sabar".(Q.S Al-Baqarah: 153)

PERSEMBAHAN

Bismillahirrohmanirrohim, dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang, segala puji hanya milik Allah SWT dan atas nikmat yang tak terhingga. Sholawat serta salam selalu tercurah kepada Rasulullah Muhammad SAW. Karya ini saya persembahkan sebagai bukti cinta kasih kepada :

Orangtua

Kedua Orangtuaku tercinta dan tersayang, KBP R. Ferry Indarmawan, S.H., S.IK dan Lia Mutiara Sari terimakasih atas segenap cinta kasih, motivasi, doa dan bantuan yang luar biasa dan sangat berpengaruh untuk saya dalam menghantarkan saya hingga menuju keberhasilan ini. Semoga saya selalu bisa membahagiakan dan bias menjadi kebanggaan untuk kalian semua.

Saudara-saudaraku

Terimakasih telah membantu dan memberi motivasi serta memberi semangat untuk kesuksesanku

Almamater Tercinta

Universitas Lampung Program Studi S1 Ilmu Komputer

SANWACANA

Bismillahirrohmanirrohim,

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala nikmat dan karunia-Nya skripsi ini dapat diselesaikan sebagai salah satu syarat dalam meraih gelar Sarjana Komputer pada Program Studi S1 Ilmu Komputer, Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam dengan judul skripsi "Sistem Pakar Diagnosa Penyakit pada Ular dengan Metode *Forward Chaining* Berbasis Android".

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari peranan dan bantuan dari berbagai pihak. Untuk itu penulis mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Rico Andrian, S.Si., M.Kom. selaku pembimbing I dan sekaligus Pembimbing Akademik, terimakasih atas kesabarannya dalam membimbing serta memotivasi penulis.
2. Bambang Hermanto, S.Kom., M.Cs. selaku pembimbing II, terimakasih atas kesabarannya dalam membimbing serta memotivasi penulis.
3. Dr. Ir. Kurnia Muludi, M.S.Sc. yang telah berkenan sebagai pembahas serta membantu dalam menyelesaikan skripsi ini.
4. Dr. Ir. Kurnia Muludi, M.S.Sc. selaku Ketua Jurusan Ilmu Komputer.

5. Prof. Warsito, S.Si., DEA, Ph.D. sebagai Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
6. Didik Kurniawan, S.Si., MT. sebagai Sekertaris Jurusan Ilmu Komputer.
7. Bapak dan Ibu Dosen Jurusan Ilmu Komputer terimakasih atas segenap ilmu yang bermanfaat kepada penulis selama melaksanakan perkuliahan di Program Studi S1 Ilmu Komputer.
8. Ade Nora A.Md. sebagai Staff Administrasi di Jurusan Ilmu Komputer yang telah membantu segala urusan administrasi selama kuliah.
9. Kedua Orangtuaku KBP R. Ferry Indarmawan, S.H., S.IK. dan Lia Mutiara Sari, Adik-adikku Vidia Anugerah Ayu Tavia Ferrial, Nindya Ramadhanti Indah Tantia Ferrial dan Aurora Chantika Chaturisa Ferrial terimakasih atas segenap cinta kasih, motivasi, bantuan yang luar biasa yang sangat berpengaruh untuk saya. Keluarga besarku yang tak bisa disebutkan satu per satu namanya terimakasih atas doa dan dukungan kalian.
10. Seseorang yang akan menjadi pendamping hidupku kelak Nona Diana Ardinur S.Pd, terimakasih atas semua cinta kasih sayang, doa, motivasi, semangat serta dukungannya untuk mengerjakan skripsi ini.
11. Terimakasih untuk teman seperjuangan selama kuliah Realaxe's Abdurrahman Adeli, Alika Nera G.J, Dyah Ayu S, M. Apriansyah Setiawan, Randika Gumilar S, Rizki Hafizh Nur, Shafinna Azzahra, Ully Novianty yang sudah membantu banyak hal selama kuliah dan semoga untuk selamanya.

12. Terimakasih untuk Genk PU Abdi Gusti R, Asep Fathurrahman, Maulidi Saputra, Raditya Rike N, Ibnu Sabil, Valerian Pratama.
13. Terimakasih untuk teman-teman Prodi S1 Ilmu Komputer angkatan 2013 yang telah bersama-sama dalam menempuh kuliah selama ini.

Penulis menyadari bahwa skripsi ini jauh dari sempurna, akan tetapi banyak harapan semoga skripsi ini dapat berguna dan memberikan manfaat bagi kita yang membaca, Aamiin.

Bandar Lampung, 31 Januari 2019

Penulis,

Tegar Ageng Khresna Ferrial

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar **Halaman**

DAFTAR TABEL

Tabel **Halaman**

DAFTAR LAMPIRAN

Lampiran **Halaman**

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ular merupakan salah satu jenis *reptile* yang paling banyak dipelihara masyarakat baik hanya untuk koleksi sampai peternakan besar yang memiliki varian dan kuantitas yang banyak. Ular merupakan salah satu jenis *reptile* yang memiliki perawatan yang cenderung mudah dan memiliki banyak sekali varian jenis dan warna, serta tidak mudah terkena penyakit jika dirawat dengan baik dan benar. Ular juga dalam beberapa kasus dapat terkena penyakit karena kurangnya pengetahuan pemelihara (*keeper*) dalam perawatannya. Seorang *keeper* tak jarang memiliki ular yang mengidap suatu penyakit, baik itu penyakit yang ringan sampai penyakit yang bisa membahayakan nyawa ular itu sendiri.

Pengetahuan akan penanganan penyakit ular selama ini hanya didapatkan dari diskusi dengan anggota komunitas yang memiliki pengalaman akan suatu penyakit tertentu dan tidak melalui seorang pakar yang memang mengetahui secara jelas akan penyakit ular tersebut, sehingga hasil diskusinya pun tidak memiliki keakuratan data yang tinggi. Data penyakit yang telah didapat dari para pakar kemudian akan digunakan dalam pengembangan sistem pakar penyakit ular ini. Penelitian inipun mendapatkan respon yang baik dari para pakar karena sistem

ini akan membantu para pemelihara dalam mendiagnosa penyakit yang diderita ular.

Penelitian tentang sistem pakar penyakit hewan sebelumnya sudah pernah dilakukan, yaitu pada penelitian tentang Sistem Pakar Diagnosa Penyakit pada Hewan Peliharaan dan Sistem Pakar Diagnosa Penyakit Sapi. Penelitian tentang Sistem Pakar Penyakit Pada Hewan Peliharaan bertujuan untuk memberikan kemudahan bagi masyarakat dalam mendiagnosa dan menangani penyakit pada hewan peliharaannya, serta membangun aplikasi berbasis web agar informasi yang disediakan dapat diakses secara luas melalui internet. Metode yang digunakan pada penelitian ini adalah metode *forward chaining* karena data-data yang didapat berupa gejala lebih mudah diaplikasikan dalam bentuk *rule-rule* yang sesuai dengan sistem pakar. Data-data berupa gejala penyakit akan digunakan sebagai pilihan gejala-gejala yang nantinya pengguna akan diminta oleh sistem untuk memilih gejala-gejala sesuai dengan yang dialami hewan peliharaannya, barulah didapat hasil diagnosa berupa nama penyakit. Hasil dari penelitian ini adalah administrator dapat menambah dan mengedit basis pengetahuan maupun data dokter yang ada di sistem. Pengguna dapat melakukan konsultasi dengan cara memilih satu gejala khas dan memilih pilihan gejala-gejala lainnya sesuai dengan gejala yang terlihat pada hewan peliharaannya. Hasil pengujian dan nilai keakuratan yang didapat sebesar 86,667%, terbukti bahwa sistem pakar diagnosa penyakit hewan peliharaan dapat mendiagnosa penyakit anjing dan kucing berdasarkan gejala-gejala yang dialami, serta dapat memberikan pengetahuan tambahan kepada pemilik hewan tentang penyakit hewan dan pencegahannya. Sistem memiliki kekurangan yakni sistem tidak bisa

menyelesaikan masalah penyakit yang memiliki gejala pendukung yang juga merupakan gejala khas atau gejala yang pasti ada pada penyakit lain maupun sebaliknya. Aplikasi sistem pakar diagnosa penyakit hewan ini hanya dapat menghasilkan satu diagnosa (Pramudita, 2014).

Penelitian selanjutnya tentang Sistem Pakar Diagnosa Penyakit Sapi yang bertujuan untuk membuat suatu sistem pakar yang mampu memberikan diagnosis yang akurat akan penyakit yang mungkin diderita pada ternak sapi beserta cara pengobatannya, serta memberikan kemudahan akses pada pengguna (peternak) dalam menggunakan sistem ini. Penelitian ini menggunakan metode *Forward Chaining* dan *Certainty Factor*. *Forward Chaining* merupakan suatu penalaran yang dimulai dari fakta-fakta untuk mendapatkan suatu kesimpulan (*Concusion*) dari fakta tersebut, sedangkan *Certainty Factor* bertujuan untuk memprediksi nilai ketidakpastian penyakit sapi, melalui penalaran atas gejala-gejala yang dialami oleh sapi, dan dilengkapi juga dengan saran-saran atau informasi yang diperlukan sehubungan dengan hasil prediksi diagnosa tersebut. Sistem ini dikembangkan dengan basis *Android* agar peternak bisa langsung mengirim data-data sapi ke *server* sehingga konsumen dapat melihat kriteria sesuai dengan penyakit yang diderita. Penelitian inimemberikan hasil bahwa sistem pakar ini telah mencakup berbagai aspek penyakit sapi. *Rule-rule* yang telah dibuat juga sudah sesuai dengan sistem pakar, serta sistem pakar ini juga telah dirancang untuk dapat digunakan dengan mudah oleh *user* karena sistem ini bersifat *user friendly*. Basis pengetahuan yang digunakan dalam sistem pakar ini adalah metode *forward chaining* dan *certainty factor* (Sibagariang, 2015).

Pengembangan sistem untuk diagnosa penyakit pada ular masih sangat jarang dan minimnya ahli *reptile* khususnya ular untuk menanggulangi penyakit yang dapat menyerang ular ketika ular dirawat dengan kurang benar maka penyakit tersebut tidak dapat langsung ditangani dan dapat mengakibatkan penyakit yang lebih buruk lagi bahkan mungkin kematian pada ular karena kurangnya ahli pada ular tersebut. Penelitian ini diperlukan oleh para pemelihara dan penghobi ular karena dapat membantu mereka terutama para pemelihara awam yang baru memelihara ular agar dapat mendiagnosa dan melakukan penanganan dengan segera pada ular mereka yang terkena penyakit.

Sistem pakar tersebut bekerja seperti performa seorang pakar dalam mengambil kesimpulan ataupun keputusan berdasarkan kondisi tertentu. Pakar dalam sistem pakar yang dibangun adalah dokter hewan dan kondisi yang digunakan untuk mengambil penghitungan kesimpulan dari gejala dan kategori peresiko yang diderita ular. Proses pembangunan sistem pakar ini akan menggunakan metode *forward chaining*.

Implementasi sistem ini adalah memanfaatkan dari suatu peralatan *mobile* seperti *smartphone, tablet pc*, dan lainnya yang sekarang ini sangatlah beragam. Selain sebagai media komunikasi dalam bentuk panggilan suara ataupun pesan singkat, dalam perkembangannya merupakan media yang mampu dilengkapi dengan berbagai program aplikasi tambahan untuk kemudahan pengguna. Bentuk pemanfaatan dari teknologi *smartphone* tersebut salah satunya adalah tentang pelayanan kesehatan dalam bentuk diagnosa penyakit. Diagnosis penyakit yang umum dilakukan untuk membantu pengguna dalam penanganan dan deteksi dini akan penyakit tersebut, sehingga pelayanan kesehatan dapat lebih cepat dilakukan.

Pembangunan aplikasi sistem pakar ini menggunakan *Android Application* dengan implementasinya dapat berupa perangkat *smartphone* yang tidak memberatkan memori, maka aplikasi ini dapat dibuat. Aplikasi ini merupakan pengembangan dari *Artificial Intelligence* yaitu sistem pakar dengan menggunakan metode *forward chaining* yang mampu menunjukkan ukuran kepastian terhadap suatu fakta atau aturan tentang penyakit pada ular.

Masyarakat membutuhkan sebuah aplikasi yang mudah diakses dan memberikan sebuah diagnosa penyakit yang diderita oleh seekor ular dan memberikan solusi akan pengobatan dan perawatan akan ular yang sakit tersebut. Masyarakat selama ini hanya mengira-ngira penyakit apa yang diderita ular peliharaannya dan masih belum memiliki sumber yang jelas akan jenis dan penanganan penyakit yang diderita ular mereka.

Penelitian ini akan mengembangkan suatu sistem pakar tentang diagnosa penyakit ular menggunakan metode *forward chaining* berbasis *android*. Sistem pakar diagnosa penyakit ular ini diimplementasikan di *platform android*.

1.2 Rumusan Masalah

Rumusan masalah yang dibahas dalam penulisan penelitian ini adalah bagaimana melakukan diagnosa penyakit pada ular berbasis *android* yang mudah diaplikasikan.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem yang dibangun menyediakan informasi yang berkaitan dengan penyakit pada ular.
2. Penelitian ini hanya membahas tentang 12 jenis penyakit pada ular dengan 49 gejala penyakit.
3. Metode penalaran yang digunakan yaitu *forward chaining*.

1.4 TujuanPenelitian

Tujuan penelitian pengembangan Sistem Pakar Diagnosa Penyakit Pada Ular dengan Menggunakan Metode *Forward Chaining* Berbasis *Android* adalah :

1. Membuat *software* Diagnosa Penyakit Ular yang dapat diinstal untuk *Smartphone* atau *Tablet Android*.
2. Membuat *software* aplikasi Diagnosa Penyakit Ular menggunakan Sistem Pakar dengan menerapkan metode *Forward Chaining* berbasis *Android*.
3. Mengimplementasikan sistem yang *user friendly*, sehingga sistem pakar ini dapat membantu memberikan informasi dan solusi dengan mudah.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Memberikan kemudahan bagi masyarakat untuk mendiagnosa penyakit pada ular.
2. Memberikan informasi kepada pengguna aplikasi atau*user* tentang penyakit ular.

BAB II
TINJAUAN PUSTAKA

2.1 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* yang menyertakan *middleware* (virtual *machine*) dan sejumlah aplikasi utama. Android merupakan modifikasi dari kernel Linux (Andry, 2011). Sistem operasi ini dikembangkan oleh sebuah perusahaan bernama Android, Inc. Android Inc inilah yang menjadi awal dari munculnya *Android*. Android Inc adalah sebuah perusahaan *start-up* kecil yang berlokasi di Palo Alto, California, Amerika Serikat yang didirikan oleh Andy Rubin bersama Rich Miner, Nick Sears, dan Chris White. Perusahaan tersebut diakuisisi oleh Google dan para pendirinya bergabung ke Google pada bulan Juli 2005. Andy Rubin sendiri kemudian diangkat menjadi Wakil Presiden divisi *Mobile* dari Google.

Tujuan pembuatan sistem operasi ini adalah untuk menyediakan *platform* yang terbuka, yang memudahkan orang mengakses internet menggunakan telepon seluler. Android juga dirancang untuk memudahkan pengembang membuat aplikasi dengan batasan yang minim sehingga kreativitas pengembang menjadi lebih berkembang (Andry, 2011).

Android tidak memiliki ketentuan yang tetap dalam konfigurasi *Software* dan *Hardware* sebagai *Open Source* dan bebas dalam memodifikasi. Fitur- fitur yang didapat dalam Android antara lain (Lee, 2011):

1. *Storage* - Menggunakan SQLite, database yang ringan, untuk sebuah penyimpanan data.
2. *Connectivity* - Mendukung GSM/EDGE, IDEN, CDMA, EV-DO, UMTS.
3. *Bluetooth* (termasuk A2DP dan AVRCP), WiFi, LTE, dan WiMax.
4. *Messaging* – Mendukung SMS dan MMS
5. *Web Browser* – Berbasiskan open-*source* WebKit, bersama mesin
6. *Chrome's V8 JavaScript*
7. *Media support* – Termasuk mendukung untuk beberapa media berikut :H.263, H.264 (dalam bentuk 3GP or MP4), MPEG-4 SP, AMR, AMRWB (dalam bentuk 3GP), AAC, HE-AAC (dalam bentuk MP4 atau 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, GIF, dan BMP.
8. *Hardware support* – Sensor akselerasi, Kamera, Kompas Digital, Sensor Kedekatan, GPS.
9. *Multi-touch* – Mendukung *multi-touch screens*
10. *Multi-tasking* – Mendukung aplikasi *multi-tasking*
11. *Flash-support* – Android 2.3 mendukung *Flash* 10.1
12. *Tethering* – Mendukung pembagian dari koneksi Internet sebagai *wired* atau *wireless hotspot*
13. *Play store* – Layanan konten digital milik Google yang melingkupi toko daring untuk produk-produk seperti musik atau lagu, buku, aplikasi, permainan, ataupun pemutar media.

14. Lingkungan pengembangan yang kaya, termasuk *emulator*, peralatan *debugging*, dan *plugin* untuk Eclipse IDE.

2.1.1 Arsitektur Android

Arsitektur Android secara garis besar dapat dijelaskan sebagai berikut:

a. *Application and Widgets*

Application and widgets adalah *layer* dimana berhubungan dengan aplikasi dan biasanya *download* aplikasi kemudian melakukan instalasi dan menjalankan aplikasi tersebut, di *layer* inilah terdapat seperti aplikasi inti termasuk klien *email*, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Aplikasi ditulis menggunakan bahasa pemrograman Java.

b. *Application Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau *member* kemampuan kepada pengembangan untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur alarm, dan menambahkan tambahan seperti status *notification*s dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan, sehingga bisa kita simpulkan *Application Frameworks* ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan atau pembuatan aplikasi yang akan dijalankan di sistem operasi *Android* karena pada

layer inilah aplikasi dapat dirancang dan dibuat, seperti *content providers* yang berupa sms dan panggilan telepon.

Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

1. Views.
2. Content Provider.
3. Resource Manager.
4. Notification Manager.
5. Activity.

c. *Libraries*

Libraries adalah *layer* dimana fitur-fitur android berada biasanya para pembuat aplikasi kebanyakan mengakses *library* untuk menjalankan aplikasinya berjalan diatas kernel, *layer* ini meliputi berbagai library C atau C++ inti seperti Libc dan SSL, serta:

1) Libraries media untuk pemutar media audio dan video.

2) Libraries untuk manajemen tampilan.

3) Libraries Graphics mencakup SGL dan OpenGL untuk grafis 2D dan 3D.

4) Libraries SQLite untuk dukungan database.

5) Libraries SSL dan *WebKit* terintegrasi dengan web *browser* dan security.

6) Libraries Live Webcore mencakup modern web *browser* dengan engine embedded web view.

7) *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's.

d. *Android Run Time*

Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. *Android Run Time* dibagi menjadi dua bagian yaitu:

1. *Core Libraries* : aplikasi android dibangun dalam bahasa Java, sementara *Dalvik* sebagai virtual mesin bukan *Java Virtual Machine*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java atau C yang dihandle oleh Core *Libraries.*
2. *Dalvik Virtual Machine* : Mesin *virtual* yang berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien dimana merupakan pengembangan yang mampu membuat linux kernel untuk *threading* dan manajemen tingkat rendah.

e. Linux kernel

Linux kernel adalah *layer* dimana inti dari operating sistem dari Android itu sendiri, berisi file-file sistem yang mengatur sistem *processing memory*, *resources*, *drivers*, dan sistem-sistem operating Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel *release* 2.6 (Nazruddin, 2012).

2.2 Sistem Pakar

Sistem pakar sebagai sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut (Martin dan Oxman, 1988) dalam (Pratama, 2015).

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud di sini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam. Sistem pakar dipandang berhasil ketika mampu mengambil keputusan seperti yang dilakukan oleh pakar aslinya baik dari sisi proses pengambilan keputusannya maupun hasil keputusan yang diperoleh (Kusrini, 2008).

2.2.1 Ciri-Ciri Sistem Pakar

Ciri-ciri sistem pakar adalah:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran-penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Padat penebaran optimal.
4. Dirancang untuk dapat dikembangkan secara bertahap.
5. Pengetahuan dan mekanisme penalaran (inference) jelas terpisah.
6. Keluaranya bersifat anjuran.
7. Sistem dapat mengaktifkan kaidah secara searah yang sesuai dituntun oleh dialog dengan user.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai dituntun oleh dialog dengan *user* (Sutojo, 2011) dalam (Pratama, 2015).

2.2.2 Konsep Dasar Sistem Pakar

Pengetahuan dari suatu sistem pakar mungkin dapat direpresentasikan dalam sejumlah cara. Metode yang paling umum untuk merepresentasikan pengetahuan adalah dalam bentuk tipe aturan (*rule*) IF…THEN (Jika…maka).

Konsep dasar dari suatu sistem pakar mengandung beberapa unsur atau elemen, yaitu (Kusrini, 2006):

2.2.2.1 Kepakaran

Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan, membaca, dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan pakar.

2.2.2.2 Pakar

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasihat. Seorang pakar harus mampu menjelaskan dan mempelajari hal-hal baru yang berkaitan dengan topik permasalahan, jika perlu harus mampu menyusun kembali pengetahuan-pengetahuan yang didapatkan, dan dapat memecahkan aturan-aturan serta menentukan relevansi kepakarannya.

2.2.2.3 Pengalihan Kepakaran

Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian ditransfer kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu:

1. Akuisisi pengetahuan (dari pakar atau sumber lain)
2. Representasi pengetahuan (pada komputer)
3. Inferensi pengetahuan
4. Pemindahan pengetahuan ke pengguna

2.2.2.4 Inferensi

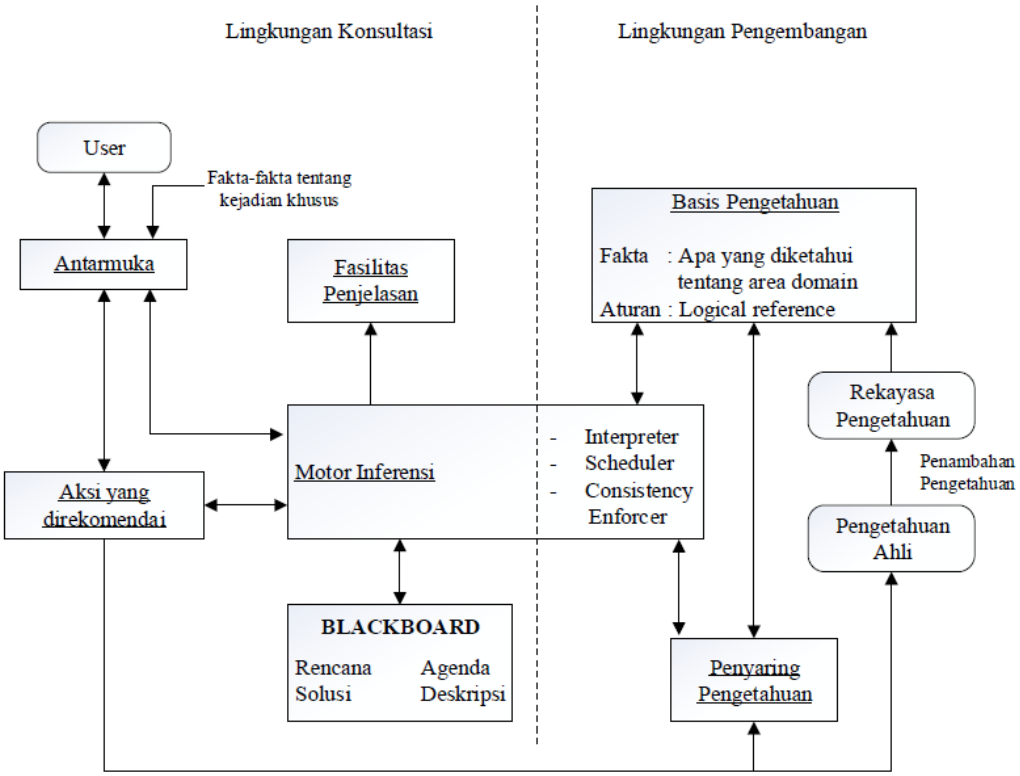
Inferensi adalah sebuah prosedur yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencakup prosedur-prosedur mengenai pemecahan masalah. Semua pengetahuan yang dimiliki oleh seorang pakar disimpan pada basis pengetahuan oleh sistem pakar. Tugas mesin inferensi adalah mengambil kesimpulan berdasarkan basis pengetahuan yang dimilikinya.

2.2.2.5 Aturan

Aplikasi sistem pakar komersial kebanyakan merupakan sistem berbasis *rule* (*rule-based system*), yaitu pengetahuan disimpan terutama dalam bentuk *rule*, sebagai prosedur-prosedur pemecahan masalah.

2.2.3 Struktur Sistem Pakar

Sistem pakar disusun oleh dua bagian utama, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*) (Kusrini,2006). Lingkungan pengembangan sistem pakar digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar. Komponen- komponen sistem pakar dalam kedua bagian tersebut dapat dilihat dalam Gambar 1.



Gambar 1. Struktur Sistem Pakar (Kusrini, 2006).

Hasil pemrosesan yang dilakukan oleh mesin inferensi dari sudut pandang pengguna bukan pakar berupa aksi yang direkomendasikan oleh sistem pakar disertai dengan fasilitas-fasilitas penjelasan yang dibutuhkan pengguna.

Komponen yang terdapat dalam sistem pakar, yaitu *design interface*, basis pengetahuan, akuisisi pengetahuan, mesin inferensi, *workplace*, fasilitas penjelasan, dan perbaikan pengaturan.

Komponen-komponen yang terdapat dalam sistem pakar antara lain, sebagai berikut:

1. Antarmuka Pengguna (*User Interface*)

   Antarmuka pengguna merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Dialog antara program dan pemakai terjadi pada bagian ini, yang memungkinkan sistem pakar menerima instruksi dan informasi (*input*) dari pemakai, juga memberikan informasi (*output*) kepada pemakai.

2. Basis Pengetahuan

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

3. Mesin Inferensi

   Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi adalah program komputer yang memberikan metodologi untuk penalaran

tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace*, dan untuk memformulasikan kesimpulan (Arhami, 2005).

4. Akuisisi Pengetahuan

Akuisisi pengetahuan adalah akumulasi, transfer, dan transformasi keahlian pemecahan masalah dari pakar atau sumber pengetahuan terdokumentasi ke program komputer, untuk membangun atau memperluas basis pengetahuan. *Knowledge engineer* dalam tahap ini berusaha menyerap pengetahuan untuk selanjutnya ditransfer ke dalam basis pengetahuan. Pengetahuan diperoleh dari pakar, dilengkapi dengan buku, basis data, laporan penelitian, dan pengalaman pemakai.

5. *Blackboard* (Tempat Kerja)

Blackboard adalah area kerja memori tempat pendeskripsian masalah yang diberikan oleh data input, digunakan juga untuk perekaman hipotesis dan keputusan sementara. Tiga tipe keputusan dapat direkam dalam *blackboard*, yaitu rencana (bagaimana mengatasi persoalan), agenda (tindakan potensial sebelum eksekusi), dan solusi (hipotesis kandidat dan arah tindakan alternatif yang telah dihasilkan sistem sampai dengan saat ini) (Kusrini, 2006).

6. Fasilitas Penjelasan Sistem

Fasilitas penjelasan sistem merupakan bagian dari sistem pakar yang memberikan penjelasan tentang bagaimana program dijalankan, apa yang harus dijelaskan kepada pemakai tentang suatu masalah, memberikan rekomendasi

kepada pemakai, mengakomodasi kesalahan pemakai dan menjelaskan bagaimana suatu masalah terjadi.

7. Perbaikan Pengetahuan

Pakar manusia memiliki sistem perbaikan pengetahuan, yakni mereka dapat menganalisis pengetahuannya sendiri dan kegunaannya, belajar darinya, dan meningkatkannya untuk konsultasi mendatang.

2.2.4 Basis Pengetahuan

Basis pengetahuan berisi pengetahuan-pengetahuan dalam menyelesaikan masalah, tentu di dalam domain tertentu. Ada dua bentuk pendekatan basis pengetahuan yang umum digunakan, yaitu:

1. Penalaran berbasis aturan (*Rule-Based Reasoning*)

Pengetahuan direpresentasikan dengan menggunakan aturan berbentuk :IF-THEN. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan si pakar dapat menyelesaikan masalah tersebut secara berurutan. Bentuk itu juga digunakan apabila dibutuhkan penjelasan tentang langkah-langkah pencapaian solusi.

2. Penalaran berbasis kasus (*Case-Based Reasoning*)

Basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan diturunkan suatu solusi untuk keadaan yang sekarang. Bentuk ini akan digunakan apabila *user* menginginkan untuk tahu lebih banyak lagi pada kasus-kasus yang hampir sama. Bentuk ini juga digunakan apabila kita telah

memiliki sejumlah situasi atau kasus tertentu dalam basis pengetahuan (Kusumadewi, 2003: 105).

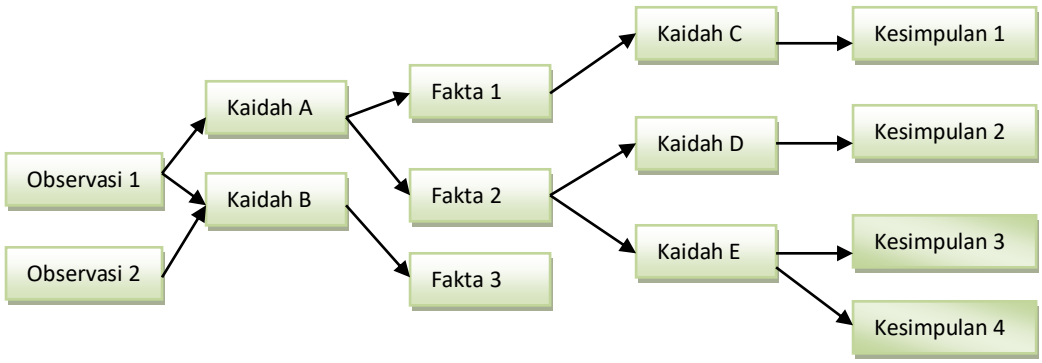
2.2.5 Mekanisme Inferensi

Mekanisme inferensi adalah bagian dari sistem pakar yang melakukan penalaran dengan menggunakan isi daftar aturan berdasarkan aturan dan pola tertentu. Mekanisme inferensi menguji aturan satu demi satu sampai kondisi aturan tersebut benar selama proses konsultasi antar sistem dan pemakai. Fungsi motor inferensi merupakan pembuktian hipotesis, bila hipotesis sudah dimasukkan ke dalam sistem pakar, maka motor inferensi pertama-tama mengecek apakah hipotesis sudah ada dalam basis data atau belum, jika sudah ada, maka hipotesis dianggap sebagai fakta yang sudah dibuktikan, sehingga operasi tidak perlu dilanjutkan. Ada dua metode inferensi yang paling penting dalam sistem pakar yaitu Runut Maju (*Forward Chaining*) dan Runut Mundur (*Backward Chaining*) dan yang akan digunakan dalam penelitian ini yaitu metode Runut Maju (*Forward Chaining*).

2.2.5.1 Runut Maju (*Forward Chaining*)

Runut maju berarti menggunakan himpunan aturan kondisi-aksi. Data pada metode ini digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan. Proses menambahkan data ke memori kerja, proses diulang sampai ditemukan suatu hasil (Wilson, 1998) dalam (Pratama, 2015). Metode inferensi runut maju cocok digunakan untuk menangani masalah pengendalian dan peramalan. Pelacakan ke depan adalah pendekatan yang dimotori data (*data-driven*). Pendekatan dalam hal ini dimulai dengan pelacakan dari informasi

masukan, dan selanjutnya mencoba menggambarkan kesimpulan. Pelacakan ke depan mencari fakta yang sesuai dengan bagian IF dari aturan IF-THEN (Arhami, 2005). Proses *forward chaining* disajikan pada Gambar 2.



Gambar 2. Proses *Forward Chaining* (Arhami, 2005)

2.2.5.2 Metode *Forward Chaining*

Metode *forward chaining* merupakan teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian IF dari rules IF-THEN. Bila ada fakta yang cocok dengan bagian IF, maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian THEN) ditambahkan ke dalam *database*. Setiap kali pencocokan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi (Sutojo, 2011).

Contoh:

Misalkan diketahui sistem pakar menggunakan 5 buah *rule* berikut.

R1 : IF (Y AND D) THEN Z

R2 : IF (X AND B AND E) THEN Y

R3 : IF A THEN X

R4 : IF C THEN L

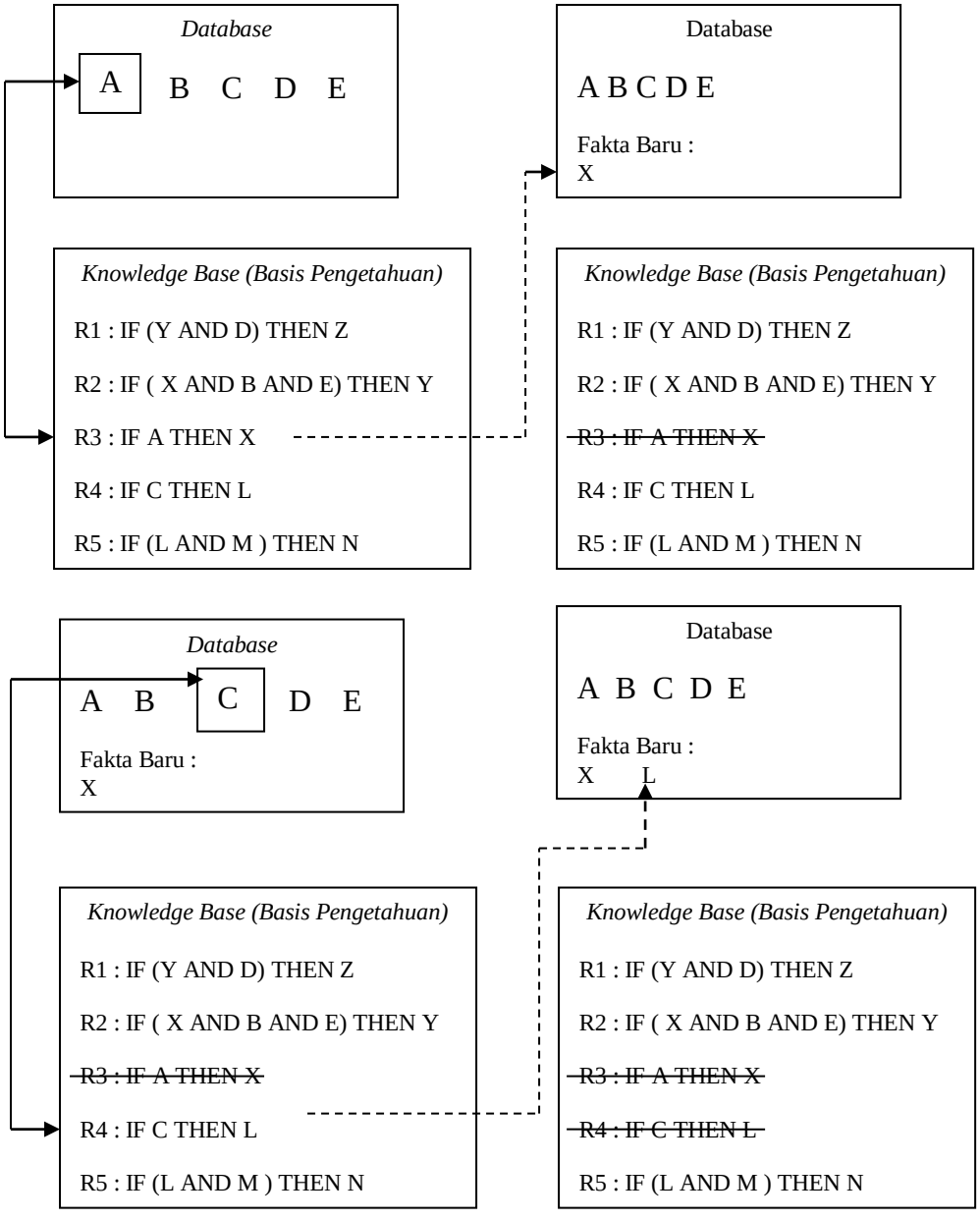
R5 : IF (L AND M) THEN N

Fakta-fakta : A, B, C, D, dan E bernilai benar

Tujuan : menentukan apakah Z bernilai benar atau salah

Iterasi ke-1

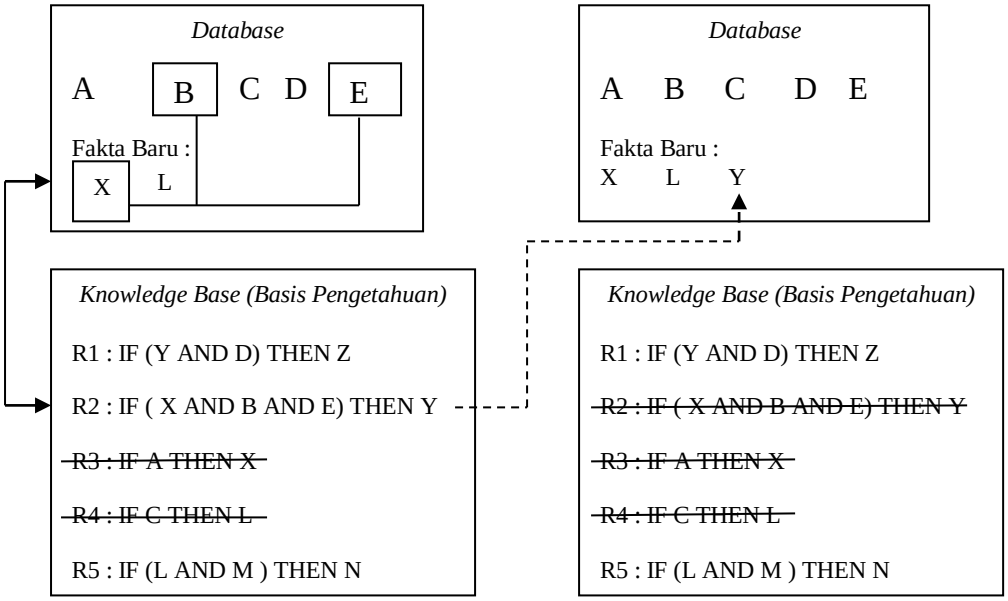
Iterasi ke-1 dapat dilihat pada Gambar 3.



Gambar 3. Iterasi ke-1

Iterasi ke-2

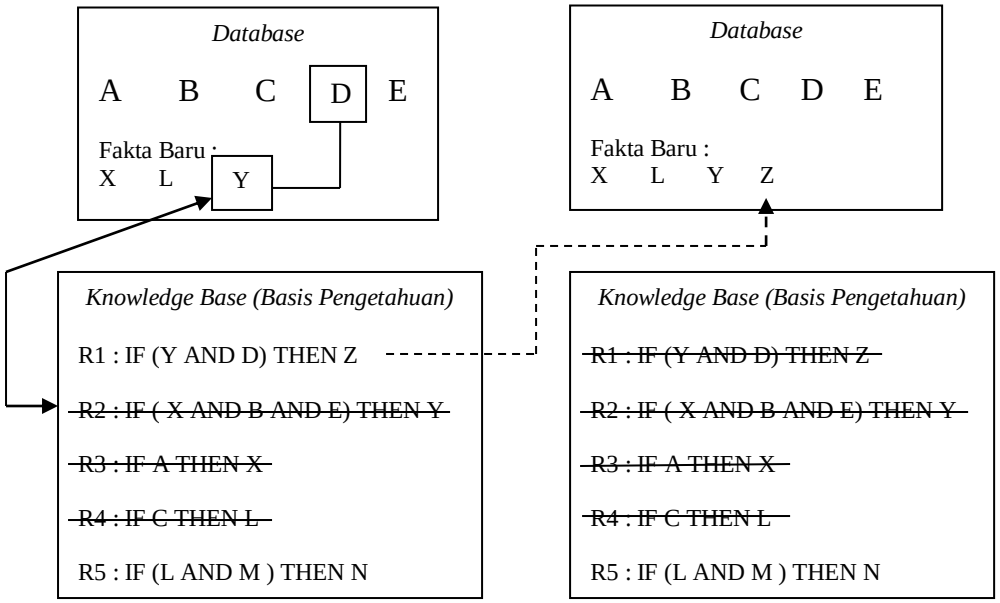
Iterasi ke-2 dapat dilihat pada Gambar 4.



Gambar 4. Iterasi ke-2

Iterasi ke-3

Iterasi ke-3 dapat dilihat pada Gambar 5.



Gambar 5. Iterasi ke-3

Sampai di sini proses dihentikan karena sudah tidak ada lagi *rule* yang bisa dieksekusi. Hasil pencarian adalah Z bernilai benar (lihat *database* di bagian fakta baru).

2.3 Metodologi Pengembangan Sistem

Metodologi yang akan digunakan pada proses pengembangan Sistem Pakar Diagnosa Penyakit Ular ini yaitu menggunakan metode *Unified Modelling Language* (UML) dalam proses desain sistemnya.

2.3.1 UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan *design* ke dalam empat tahapan iterative, yaitu: identifikasi kelas-kelas dan obyek-obyek, identifikasi semantik dari hubungan obyek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur pemodelan *entity-relationship*.

Tahapan utama dalam metodologi ini adalah analisis, *design* sistem, *design* obyek, dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OOP. Metode OOSE dari Jacobson lebih memberi penekanan pada *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement*, analisis, *design*, implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Design UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya (Nugroho, 2010).

UML dideskripsikan oleh beberapa diagram, yaitu:

1. *Use Case* Diagram

Use case Diagram digunakan untuk menggambarkan sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* diagram lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. Sebuah *use case* diagram merepresentasikan sebuah interaksi antara aktor dengan sistem yang akan dikembangkan (Fowler, 2004).

Komponen-komponen dalam *use case* diagram (Fowler, 2004):

a. Aktor

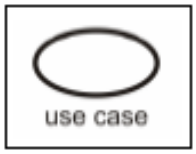
Aktor pada dasarnya bukanlah bagian dari *use case* diagram, namun untuk dapat terciptanya suatu *use case* diagram diperlukan aktor, dimana aktor tersebut

mempresentasikan seseorang atau sesuatu (seperti perangkat atau sistem lain) yang berinteraksi dengan sistem yang dibuat. Sebuah aktor mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima dan memberi informasi pada sistem. Aktor hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Aktor digambarkan dengan *stick pan* seperti yang terdapat pada Gambar 6.

Gambar 6. Contoh Aktor (Fowler, 2004).

b. Use Case

Gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti kegunaan sistem yang akan dibangun. Bentuk *use case* dapat terlihat pada Gambar 7.



Gambar 7. *Use Case* (Fowler, 2004).

Ada beberapa relasi yang terdapat pada *use case diagram*:

1. *Association*, menghubungkan *link* antar *element*.
2. *Generalization*, disebut juga pewarisan (*inheritance*), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.

3. *Dependency*, sebuah elemen bergantung dalam beberapa cara ke elemen lainnya.
4. *Aggregation*, bentuk *association* dimana sebuah elemen berisi elemen lainnya.

Tipe relasi yang mungkin terjadi pada *use case diagram*:

1. <<*include*>>, yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.
2. <<*extends*>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan peringatan.
3. <<*communicates*>>, merupakan pilihan selama asosiasi hanya tipe *relationship* yang dibolehkan antara aktor dan *use case*.

2. *Activity Diagram*

Activity Diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat digunakan untuk aktifitas lainnya (Fowler, 2004).

Tabel Notasi *Activity Diagram* diilustrasikan pada Tabel 1.

Tabel 1. Notasi *Activity Diagram* (Meildy, 2014).

Simbol	Keterangan
	Titik Awal

Lanjutan Tabel 1. Notasi *Activity Diagram* (Meildy, 2014).

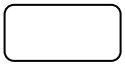
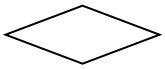
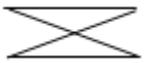
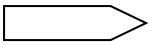
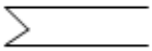
Simbol	Keterangan
	Titik Akhir
	*Activity*
	Pilihan untuk mengambil keputusan
	Fork; Digunakan untuk menunjukan kegiatan yang dilakukan secara pararel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	Rake; Menunjukan adanya dekomposisi
	Tanda waktu
	Tanda Pengiriman
	Tanda Penerimaan
	Aliran akhir (*Flow Final*)

Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Pembuatan *activity diagram* pada awal pemodelan proses dapat membantu memahami keseluruhan proses. *Activity diagram* juga digunakan untuk menggambarkan interaksi antara beberapa *use case* (Fowler, 2004).

3. *Class Diagram*

Class adalah sebuah spesifikasi yang akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class Diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain (Fowler, 2004).

Class memiliki tiga area pokok:

1. Nama (*Class Name*)
2. Atribut
3. Metode (*Operations*)

Class pada UML digambarkan dengan segi empat yang dibagi beberapa bagian. Bagian atas merupakan nama dari *class*. Bagian yang tengah merupakan struktur dari *class* (atribut) dan bagian bawah merupakan sifat dari *class* (metode/operasi).

Atribut dan metode dapat memiliki salah satu sifat berikut:

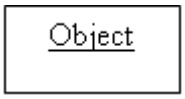
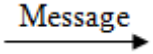
1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan *class* lain yang mewarisinya.
3. *Public*, dapat dipanggil oleh *class* lain (Fowler, 2004).

4. *Sequence Diagram*

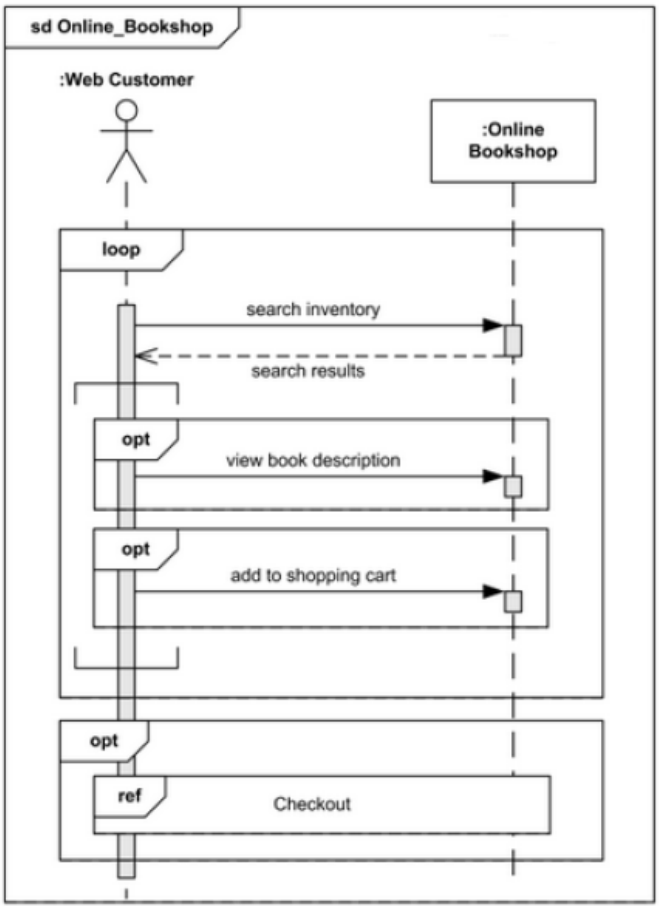
Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik tertentu dalam eksekusi sistem (Fowler, 2014).

Notasi *Sequence Diagram* disajikan pada Tabel 2.

Tabel 2. Notasi *Sequence Diagram* (Meildy, 2014).

Simbol	Nama	Keterangan
	*Object*	*Object* merupakan *instance* dari sebuah *class* dan dituliskan secara *horizontal*. Digambarkan sebagai sebuah *class* (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.
	*Actor*	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *actor* sama dengan simbol pada *actor Use Case* Diagram.
	Lifeline	*Lifeline* mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus *vertical* yang ditarik dari sebuah obyek.
	*Activation*	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.
	*Message*	*Message*, digambarkan dengan anak panah *horizontal* antara *Activation*. *Message* mengindikasikan komunikasi antara object-object.

Contoh dari *sequence diagram* dapat dilihat pada Gambar 8.



Gambar 8. Contoh *Sequence Diagram* (uml-diagrams.org, 2014)

2.3.2 Keunggulan UML

UML secara umum diterapkan dalam pengembangan sistem/perangkat lunak berorientasi obyek sebab metodologi UML ini umumnya memiliki keunggulan-keunggulan sebagai berikut (Nugroho, 2010):

a. *Uniformity*

Pengembang cukup menggunakan satu metodologi dari tahap analisis hingga perancangan dengan metodologi UML, hal ini tidak bisa dilakukan dalam metodologi pengembangan terstruktur. UML juga memungkinkan kita merancang

komponen antarmuka pengguna (*user interface*) secara integrasi bersama dengan perancangan perangkat lunak sekaligus dengan perancangan basis data.

b. *Understandability*

Kode yang dihasilkan dapat diorganisasi ke dalam kelas-kelas yang berhubungan dengan masalah sesungguhnya sehingga lebih mudah dipahami siapapun juga dengan metodologi ini.

c. *Stability*

Kode program yang dihasilkan relatif stabil ssepanjang waktu sebab sangat mendekati permasalahan sesungguhnya di lapangan.

d. *Reusability*

Penggunaan ulang kode dimungkinkan dengan metodologi berorientasi objek, sehingga pada gilirannya akan sangat mempercepat waktu pengembangan perangkat lunak.

2.4 Teknik Pengujian Perangkat Lunak

Pendekatan kasus uji terbagi menjadi dua macam yaitu *white-box* dan *black-box*. Pendekatan *white-box* adalah pengujian untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya (Jiang, 2012). Jalur logika perangkat lunak akan dites dengan menyediakan kasus uji yang akan mengerjakan kumpulan kondisi dan pengulangan secara spesifik, sehingga melalui penggunaan metode ini akan dapat memperoleh kasus uji yang menjamin bahwa semua jalur independen pada suatu model telah digunakan minimal satu kali, penggunaan keputusan logis pada sisi benar dan salah, pengeksekusian semua *loop* dalam

batasan dan batas operasional perekayasa, serta penggunaan struktur data internal guna menjamin validitasnya (Pressman, 2010).

Pendekatan *black-box* merupakan pendekatan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan (Jiang, 2012). Kasus uji ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya. Teknik pengujian ini berfokus pada domain informasi dari perangkat lunak, yaitu melakukan kasus uji dengan mempartisi domain *input* dan *output* program. Metode *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian ini berusaha menemukan kesalahan dalam kategori fungsi-fungsi yang tidak benar atau hilang, kesalahan *interface*, kesalahan dalam struktur data atau akses basis data eksternal, kesalahan kinerja, dan inisialisasi dan kesalahan terminal (Pressman, 2010).

2.4.1 *Equivalence Partitioning*

Equivalence Partitioning (EP) merupakan metode *black box testing* yang membagi domain masukan dari program kedalam kelas-kelas sehingga *test case* dapat diperoleh. *Equivalence Partitioning* berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk *Equivalence Partitioning* berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat

berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi *Boolean* (Pressman, 2001).

2.4.2 Probabilitas Klasik

Probabilitas merupakan suatu cara kuantitatif yang berhubungan dengan ketidakpastian yang telah ada (Arhami, 2005:135). Probabilitas klasik disebut juga *a priori probability* karena berhubungan dengan suatu permainan (*games*) atau sistem. Seperti yang telah disebutkan sebelumnya, istilah *a priori* berarti "sebelum" (Arhami, 2005:138). Probabilitas ini dianggap sebagai suatu jenis permainan seperti pelemparan dadu, permainan kartu, dan pelemparan koin.

Rumus umum untuk probabilitas klasik di definisikan sebagai peluang *P(A)* dengan *n* adalah banyaknya kejadian, *n(A)* merupakan banyaknya hasil mendapatkan *A*. Frekuensi *relative* terjadinya *A* adalah $\frac{n(A)}{n}$ maka (Arhami, 2005:138) :

$P(A) = \frac{n(A)}{n}$ …….....(Persamaan 1)

Probabilitas klasik ini digunakan untuk mendapatkan peluang kemungkinan penyakit, sehingga untuk menghitung persentase penyakit adalah :

Persentase *(A) = P(A) x 100%*.....(Persamaan 2)

Tabel persentase penyakit disajikan pada Tabel 3.

Tabel 3. Kemungkinan Penyakit

Kondisi	Persentase
Pasti Tidak	<10%
Tidak Tahu	10-19%
Hampir Mungkin	20 – 39%
Mungkin	40 – 59%
Kemungkinan Besar	60 - 79%
Hampir Pasti	80 – 99%
Pasti	100%

2.4.3 Skala *Likert*

Sikap dapat diukur dengan metode *rating* yang dijumlahkan (*Method of Summated Ratings*). Metode ini merupakan metode penskalaan pernyataan sikap yang menggunakan distribusi respons sebagai dasar penentuan nilai skalanya. Nilai skala setiap pernyataan tidak ditentukan oleh derajat *favourable* nya masing-masing akan tetapi ditentukan oleh distribusi *respons* setuju dan tidak setuju dari sekelompok responden yang bertindak sebagai kelompok uji coba (*pilot study*) (Azwar, 2011).

Skala Likert, yaitu skala yang berisi lima tingkat preferensi jawaban dengan pilihan sebagai berikut: 1 = sangat tidak setuju; 2 = tidak setuju; 3 = ragu–ragu atau netral; 4 = setuju; 5 = sangat setuju. Penentuan kategori interval tinggi, sedang, atau rendah selanjutnya menggunakan rumus sebagai berikut :

$$I = \frac{100\%}{K} \text{ (Persamaan 3)}$$

Keterangan :

I = Interval;

K = Kategori jawaban

Kriteria penilaian = % Total skor tertinggi – Interval (I)

2.4.4 Genymotion

Pengujian aplikasi juga dapat dilakukan dengan menggunakan *emulator*. *Emulator* tersebut adalah Genymotion. Genymotion adalah tambahan baru untuk pengujian android karena memiliki fitur yang sama dengan perangkat *smartphone*. Genymotion dapat dijalankan di Android Studio ataupun Eclipse (Tomar, 2016).

2.5 Ular

Ular termasuk dalam Kelas *Reptilia*, merupakan hewan yang hidupnya melata, dan kebanyakan hidup di *terestrial*, dan termasuk dalam Ordo *Squamata* karena tubuhnya dikelilingi oleh sisik. Ular masuk dalam kategori *Ophidia* (*Serpentes*) atau hewan tidak berkaki dalam penggolongan Sub-ordo.

Penggolongan dalam Familia ular dibagi menjadi 3 kelompok, yaitu :

1. Elaphidar (ular beracun yang tersebar di lima Benua, kecuali Benua Eropa). Contoh ular yang terkenal dari Familia ini adalah Cobra yang banyak tersebar di Asia Tenggara, di Benua Afrika (Mamba), di Benua Amerika (*Coral* Snake).
2. *Hydrophidae* (ular beracun yang hidup di perairan laut, tersebar di Asia Selatan dan Amerika tengah) ciri khusus yang dimiliki oleh ular ini adalah bentuk ekor yang menyerupai dayung. Ular ini memiliki panjang mulai dari 3 kaki (1 meter) seperti (*Laticauda laticauda*) hingga panjang 10 kaki (3 meter) seperti (*Hydrophis spiralis*). Ular laut ini jarang menyerang manusia, karena mereka memangsa ikan.
3. Boidae (ular tidak beracun, tersebar di dunia lama dan baru seperti Afrika, India, Indochina, dan Australia). Contoh dari familia ini adalah Reticulatus atau Python pada umumnya. Boa dan Anaconda (Amerika Tengah dan Selatan), dengan kata lain Familia ini merupakan kelompok Famili ular-ular terbesar di dunia (Kurniati, 2009).

2.5.1 Penyakit Ular

Penyakit pada ular dapat disebabkan baik oleh bakteri, virus, parasit, maupun fungi (Cooper dan Jackson, 1981). Beberapa penyakit tersebut diantaranya adalah septisemia yang disebabkan oleh bakteri dan virus. Berbagai agen penyebab sepsis antara lain bakteri *Pseudomonas spp; Paramyxovirus; tuberculosis* akibat infeksi bakteri *Mycobacterium thamnopeos; leptospirosis* akibat infeksi bakteri *Leptospira sp;* tumor pada ular akibat infeksi *Oncornavirus.* Serta *Inclusion Body Disease (IBD)* dengan agen penyebabnya *Retrovirus* (Kaplan dan Jereb, 1995).

Penyakit pada ular akibat infeksi parasit diantaranya adalah *amoebiasis* akibat infeksi *Entamoeba Invadens, siklospirosis* akibat infeksi *Cyclospironinae, cryptosporidiosis* akibat infeksi *Cryptosporidium sp., Hemogregariniasis* disebabkan oleh parasit darah *Haemogregarine* dan *nematodiasis* dengan famili *ascaridae* disebabkan oleh genus *Ophidascaris, cestodiosis* disebabkan oleh *Bothridium sp., Kalicephalus* pada famili *strongilidae* dan *rhabdias* dengan famili *lungworm* serta adanya penyakit *trematodiasis* yang ada pada famili *spirorchidae* disebabkan oleh *Spirorchis elegans, Styphiloroda* dalam famili *styphiloroda* dan *dasymetra, lechriorchis, zeugorchis, ochestosoma* serta *stomatrema* pada famili *renifers*. Penyakit pada ular juga dapat diakibatkan dari infeksi fungi atau *mikosis* yaitu *mikosis traktus alimentaris* akibat infeksi *Metharizium anisopliae* dan *Paecilomyces lilasinus* dan *mikosis traktus respirasi* akibat infeksi *Metharizium anispliae* serta *aktinomikosis* akibat infeksi *Actinomycete* (Jones dalam Cooper dan Jackson, 1981; Fryc dalam Kirk, 1983).

BAB III
METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan di Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilakukan pada bulan Oktober 2016 sampai dengan selesai.

3.2 Perangkat Penelitian

Alat pendukung yang digunakan dalam penelitian ini adalah sebagai berikut:

A. Perangkat Keras

- *Notebook* dengan spesifikasi :

  Processor : Intel® Core™ i5-3230M CPU @2.6GHz

  RAM : 4096 MB

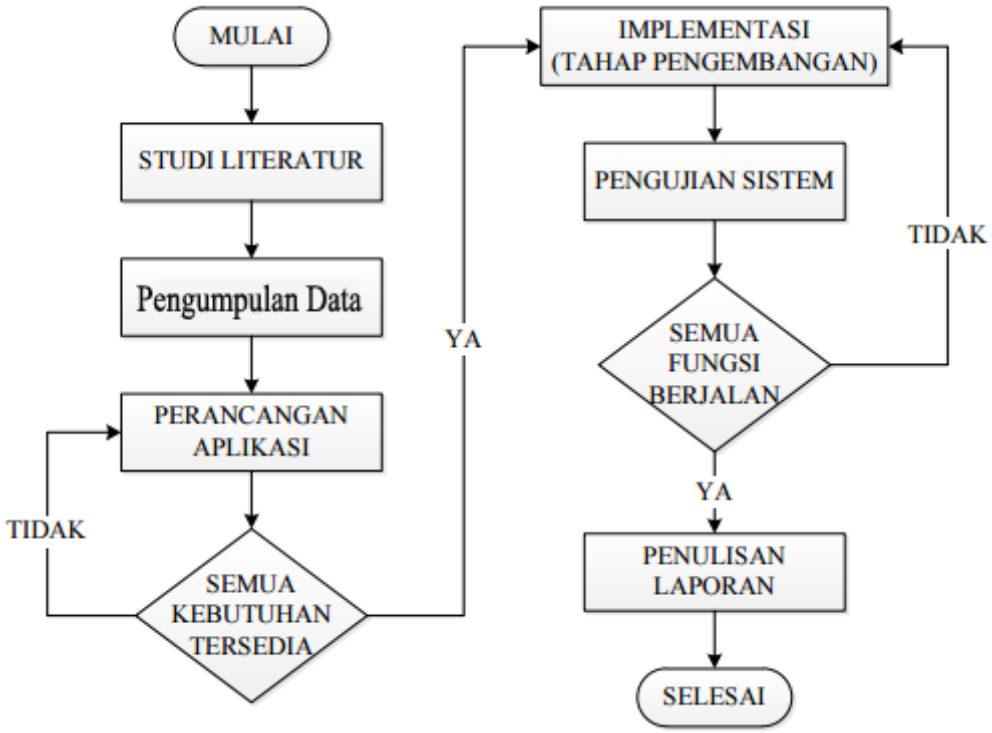
  Harddisk : 500 GB

B. Perangkat Lunak

- Sistem Operasi Windows 7 64 Bit
- *Text Editor* Notepad++
- CorelDRAW X7
- Android Studio

3.3 Tahapan Penelitian

Tahapan penelitian yaitu tahapan yang akan dilakukan peneliti untuk mempermudah dalam melakukan penelitian. Desain penelitian sistem pakar penyakit ular dengan metode *forward chaining* digambarkan pada Gambar 9.



Gambar 9. Diagram Alir Penelitian

3.3.1 Studi Literatur

Studi literatur dilakukan dengan cara mempelajari aspek-aspek yang berkaitan dengan penelitian ini, diantaranya adalah mencari jenis-jenis penyakit ular, gejala-gejala dari penyakit tersebut dan solusinya, mempelajari metode *forward chaining*. Data-data yang digunakan dalam studi literatur didapat dengan cara mengumpulkan jurnal, mengumpulkan data dari ahli dan buku yang berkaitan dengan topik.

3.3.2 Pengumpulan Data

Tahapan ini mengumpulkan data-data terkait penyakit ular. Data yang dikumpulkan berasal dari hasil konsultasi dengan pakar dan literatur-literatur tentang penyakit ular. Data-data yang dikumpulkan disusun menjadi basis aturan yang akan digunakan dalam sistem pakar.

3.3.3 Perancangan Sistem

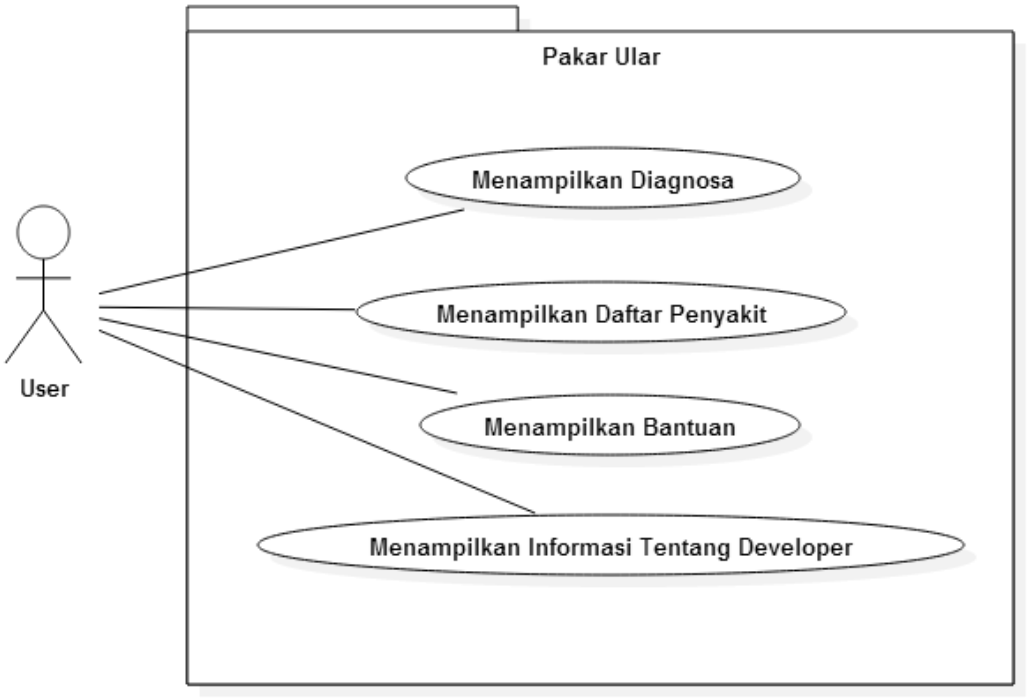
Perancangan sistem adalah tahap setelah analisis dari siklus pengembangan sistem. Perancangan sistem disini berupa penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Perancangan sistem menentukan bagaimana suatu sistem akan menyelesaikan apa yang mesti diselesaikan. Tahap ini termasuk mengkonfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem sehingga setelah dilakukan instalasi akan benar-benar memuaskan rancang bangun yang telah ditetapkan pada akhir tahap analisis sistem.

3.3.3.1 Perancangan UML (*Unified Modelling Language*)

Pemodelan (*modeling*) adalah tahap merancang perangkat lunak sebelum melakukan tahap pembuatan program (*coding*). Perancangan sistem pada penelitian ini dilakukan dengan memodelkan permasalahan dalam bentuk diagram-diagram UML sebagai berikut.

1. *Use Case Diagram*

Use case Diagram dibawah ini menggambarkan sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case diagram* ini lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. Pengguna (*user*) pada aplikasi ini dapat melakukan 4 (empat) interaksi antara lain Daftar Penyakit, Diagnosa, Bantuan dan Informasi Tentang *Developer* aplikasi. *Use case diagram* aplikasi Sistem Pakar Diagnosa Penyakit Ular dapat dilihat pada Gambar 10.



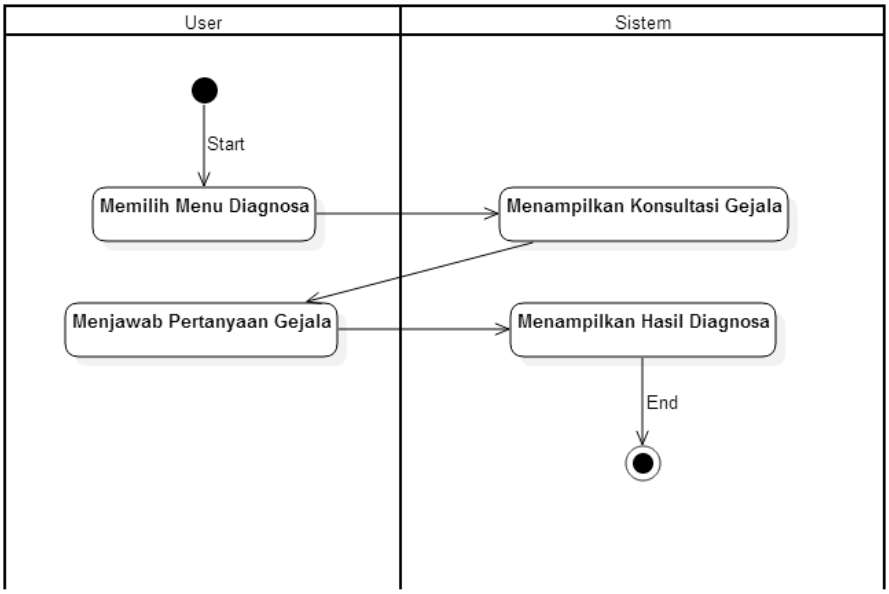
Gambar 10. *Use Case Diagram*

2. *Activity Diagram*

Activity diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam satu operasi sehingga dapat juga untuk aktivitas lainnya. Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Aplikasi Sistem Pakar Diagnosa Penyakit Ular ini memiliki 4 (empat) *activity diagram* yaitu sebagai berikut:

a. *Activity Diagram* Diagnosa

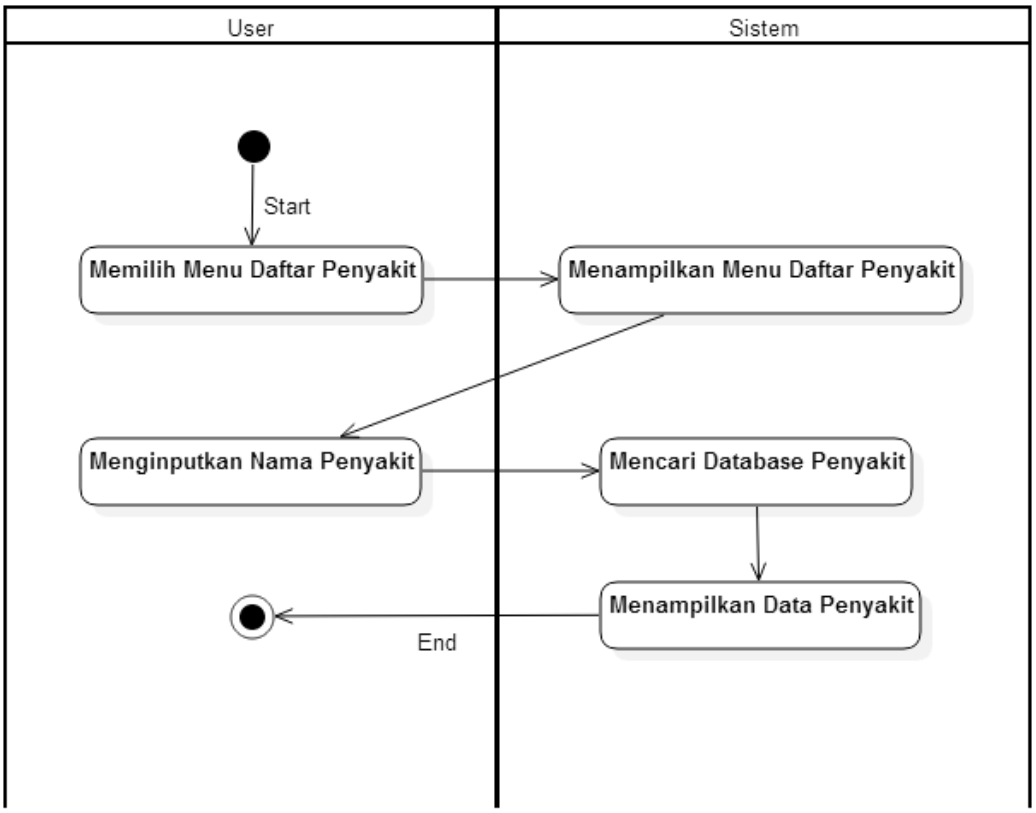
Activity diagram Diagnosa dimulai dengan pengguna memilih menu Diagnosa, kemudian sistem akan menampilkan kolom konsultasi gejala dan pengguna menjawab pertanyaan gejala penyakit ular. Sistem mencari *database* penyakit ular dan menampilkan hasil diagnosa. *Activity diagram* Diagnosa disajikan pada Gambar 11.



Gambar 11. *Activity Diagram* Diagnosa

b. *Activity Diagram* Daftar Penyakit

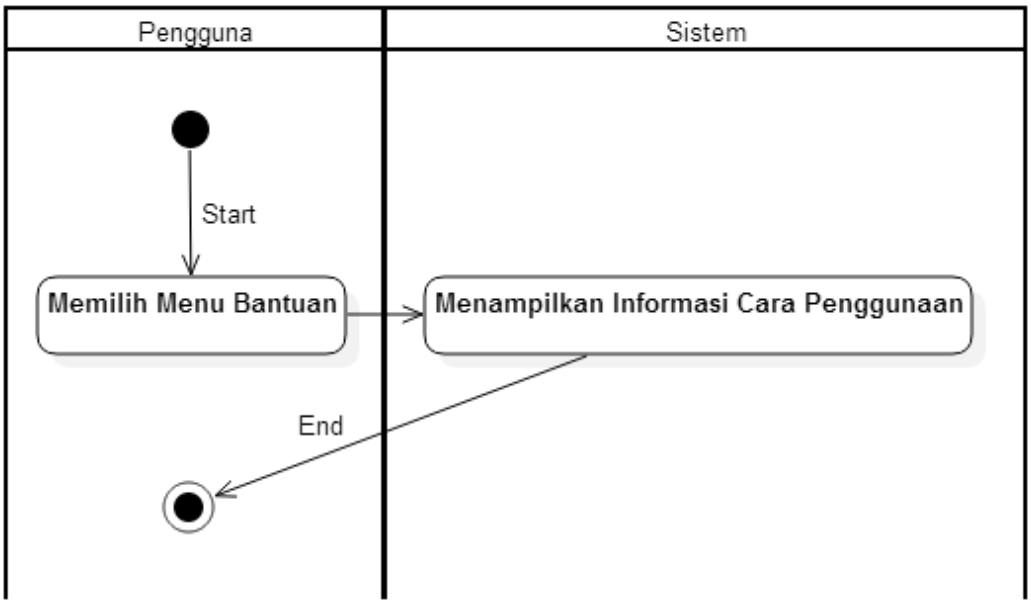
Activity diagram Daftar Penyakit dimulai dengan pengguna memilih menu "Daftar Penyakit", kemudian sistem akan menampilkan daftar penyakit ular. Pengguna dapat mengetikan nama Penyakit yang akan dicari, jika sudah klik tombol "cari". *Activity diagram* Daftar Penyakit dilihat pada Gambar 12.



Gambar 12. *Activity Diagram* Data Penyakit

c. *Activity Diagram* Bantuan

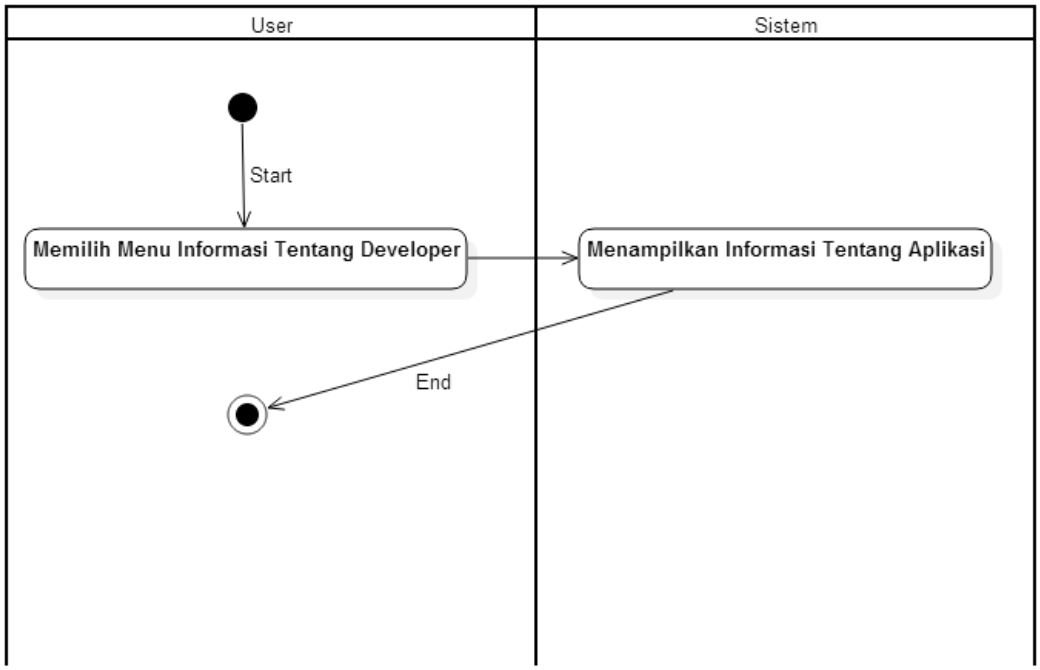
Activity diagram bantuan dimulai dengan pengguna memilih menu “Bantuan”, kemudian sistem akan menampilkan informasi yang berkaitan dengan cara penggunaan aplikasi Sistem Pakar Diagnosa Penyakit Ular. *Activity diagram* bantuan disajikan pada Gambar 13.



Gambar 13. *Activity* Diagram Bantuan

d. *Activity Diagram* Informasi Tentang *Developer*

Activity diagram informasi tentang *developer* aplikasi dimulai dengan pengguna memilih menu “Informasi Tentang Developer”, kemudian sistem menampilkan informasi mengenai aplikasi Sistem Pakar Diagnosa Penyakit Ular. *Activity diagram* informasi tentang *developer* dapat dilihat pada Gambar 14.



Gambar 14. *Activity Diagram* Informasi Tentang *Developer*

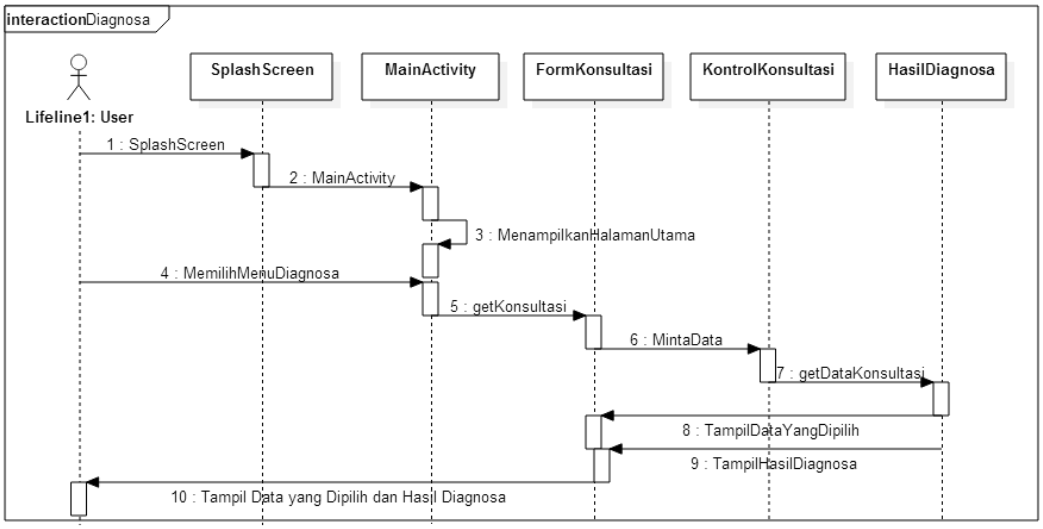
3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. *Sequence diagram* berguna untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik tertentu dalam eksekusi sistem. *Sequence diagram* pada aplikasi Sistem Pakar Diagnosa Penyakit Ular ini terdapat 4 (empat) macam yaitu sebagai berikut:

a. *Sequence Diagram* Diagnosa

Menu “Diagnosa” dapat diakses pengguna dengan cara memilih aplikasi Sistem Pakar Diagnosa Penyakit Ular, kemudian otomatis akan menuju halaman *splash screen*, selanjutnya akan muncul menu utama aplikasi, dan pengguna dapat

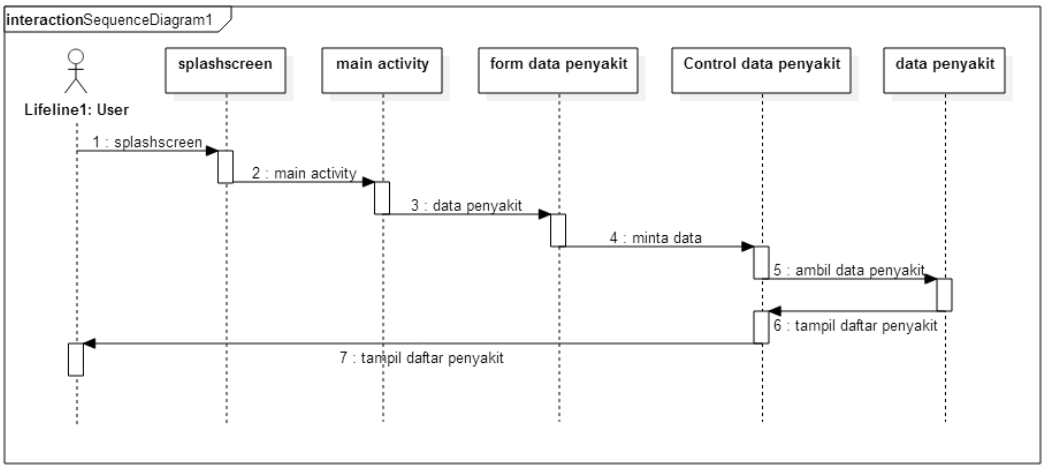
memilih menu “Diagnosa”, maka sistem akan menampilkan pertanyaan konsultasi gejala penyakit ular, lalu menjawab pertanyaan gejala penyakit ular dan sistem akan memproses *database* kemudian menampilkan hasil diagnosa penyakit, gejala dan solusi. *Sequence diagram* Diagnosa dapat dilihat pada Gambar 15.



Gambar 15. *Sequence Diagram* Diagnosa

b. *Sequence Diagram* Daftar Penyakit

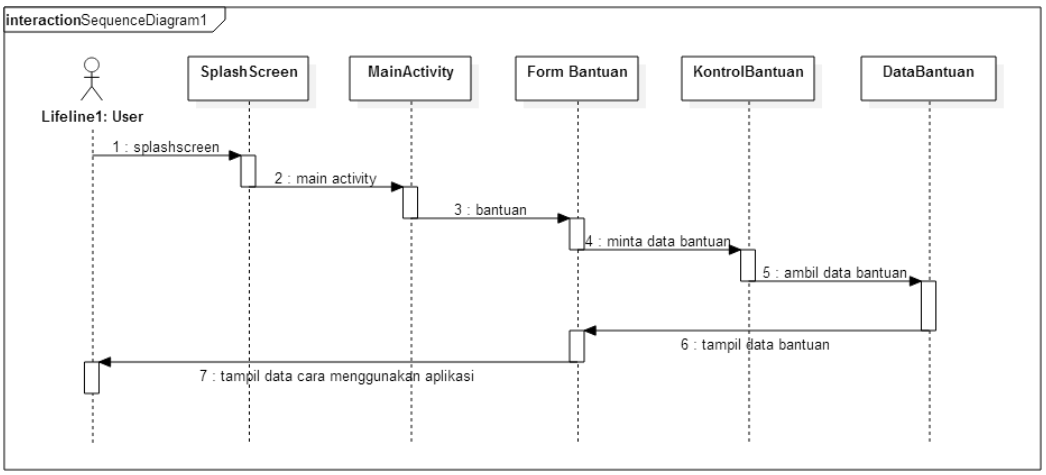
Menu “Daftar Penyakit” dapat diakses pengguna dengan cara memilih aplikasi Sistem Pakar Diagnosa Penyakit Ular, kemudian otomatis akan menuju halaman *splash screen*, selanjutnya akan muncul menu utama aplikasi, dan pengguna dapat memilih menu “Daftar Penyakit”, maka sistem akan menampilkan daftar penyakit ular. Pengguna kemudian dapat memilih nama penyakit yang diinginkan untuk mendapatkan informasi terkait penyakit tersebut. *Sequence diagram* Daftar Penyakit dapat dilihat pada Gambar 16.



Gambar 16. *Sequence Diagram* Daftar Penyakit

c. *Sequence Diagram* Bantuan

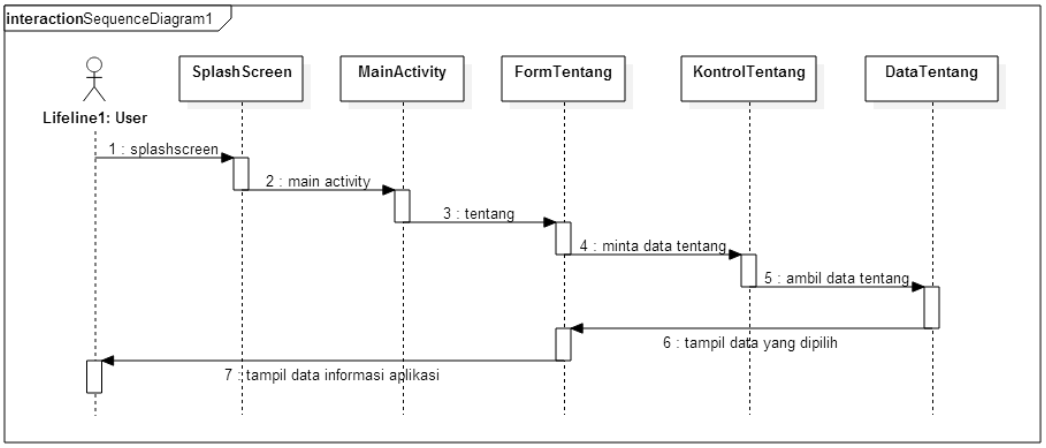
Pengguna dapat memilih menu "Bantuan" ketika sudah berada di menu utama aplikasi untuk mengetahui informasi mengenai cara penggunaan aplikasi, maka sistem akan menampilkan informasi tersebut. *Sequence diagram* bantuan disajikan pada Gambar 17.



Gambar 17. *Sequence Diagram* Bantuan

d. *Sequence Diagram* Informasi Tentang *Developer*

Pengguna dapat memilih menu "Informasi Tentang *Developer*" ketika sudah berada di menu utama aplikasi untuk mengetahui informasi mengenai aplikasi dan *developer*-nya, maka sistem akan menampilkan informasi tentang aplikasi Sistem Pakar Diagnosa Penyakit Ular. *Sequence diagram* informasi tentang *developer* aplikasi disajikan pada Gambar 18.



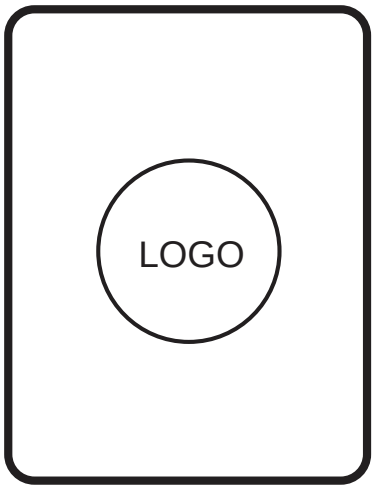
Gambar 18. *Sequence Diagram* Informasi Tentang *Developer*

3.3.3.2 Perancangan Antarmuka

Perancangan antarmuka merupakan proses penggambaran bagaimana sebuah tampilan (*interface*) sistem dibentuk. Aplikasi Sistem Pakar Diagnosa Penyakit Ular dirancang dengan tampilan yang *user friendly*, sehingga diharapkan dapat mempermudah pengguna dalam menggunakan aplikasi ini. Aplikasi ini memiliki beberapa *layout* atau *form* antara lain :

1. Desain Antarmuka *Splash Screen*

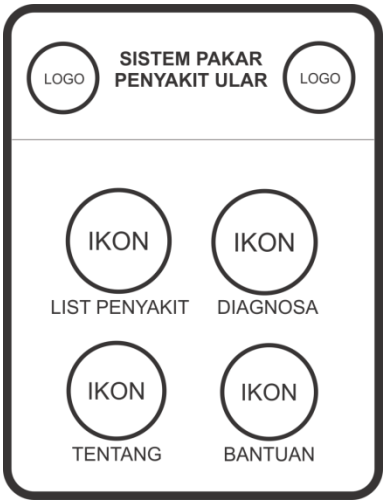
Splash Screen adalah *form* yang ditampilkan diawal ketika aplikasi/program dijalankan. Aplikasi Sistem Pakar Diagnosa Penyakit Ular menggunakan *splash screen* yang muncul sepersekian detik pada saat pertama membuka aplikasi Sistem Pakar Diagnosa Penyakit Ular. *Splash screen* di sini dimaksudkan sebagai estetika untuk menujukan identitas aplikasi saja, tanpa fungsi lainnya. Perancangan antarmuka *splash screen* aplikasi dapat dilihat pada Gambar 19.



Gambar 19. Desain Antarmuka *Splash Screen*

2. Desain Antarmuka Utama

Menu utama berisikan menu-menu pilihan yang dapat digunakan oleh pengguna. Menu yang terdapat pada menu utama antara lain : menu Diagnosa, menu Daftar Penyakit, menu Bantuan dan menu Informasi Tentang *Developer* aplikasi. Perancangan antarmuka utama aplikasi dapat dilihat pada Gambar 20.



Gambar 20. Desain Antarmuka Utama

3. Desain Antarmuka Diagnosa

Tampilan ketika pengguna memilih menu “Diagnosa”, sistem akan memberikan pertanyaaan yang kemudian dijawab oleh *user* dan disimpan untuk diproses. Data digunakan untuk menampilkan pertanyaan yang terkait dengan jawaban yang diberikan pengguna dan menyimpan aturan analisa kemungkinan penyakit yang diderita ular. Pengguna tinggal menekan tombol “YA” atau “TIDAK” untuk menjawab beberapa pertanyaan yang diajukan oleh sistem. Perancangan antarmuka Diagnosa dapat dilihat pada Gambar 21.

Gambar 21. Desain Antarmuka Diagnosa

Menu Diagnosa memiliki sub menu Hasil Diagnosa Penyakit, dengan rincian sebagai berikut :

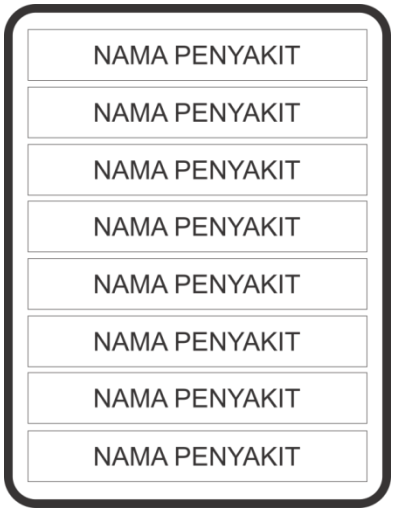
- Desain Antarmuka Sub Menu Hasil Diagnosa Penyakit

Sub menu ini menampilkan hasil diagnosa penyakit yang diderita dan memberikan informasi tentang penyakit tersebut. Pengguna dapat melihat apa penyakit yang diderita ular dan penjelasan secara singkat tentang penyakit tersebut. Perancangan antarmuka Sub Menu Hasil Diagnosa Penyakit dapat dilihat pada Gambar 22.

Gambar 22. Desain Antarmuka Sub Menu Hasil Diagnosa Penyakit

4. Desain Antarmuka Daftar Penyakit

Pengguna dapat langsung melihat daftar nama penyakit ular dan mencari jenis penyakit ular, dengan menekan tombol “Daftar Penyakit” kemudian memilih nama penyakit yang ada pada daftar penyakit. Menu *Detail* Penyakit muncul jika pengguna memilih salah satu nama penyakit yang ada pada daftar penyakit. Perancangan antarmuka Daftar Penyakit dan *Detail* Penyakit dapat dilihat pada Gambar 23 dan 24.

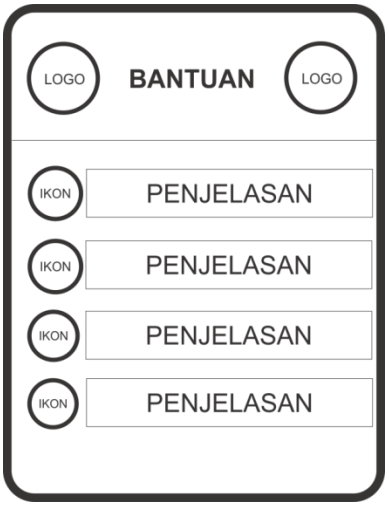


Gambar 23. Desain Antarmuka Daftar Penyakit

Gambar 24. Desain Antarmuka *Detail* Penyakit

5. Desain Antarmuka Bantuan

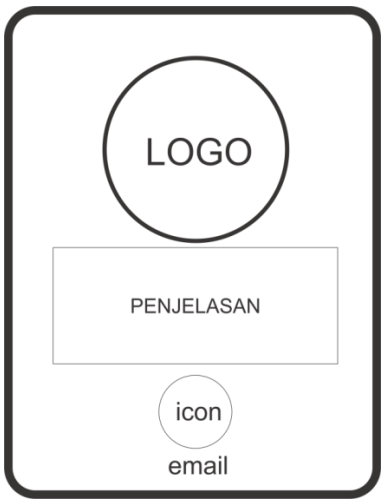
Pengguna dapat melihat informasi mengenai cara penggunaan aplikasi Sistem Pakar Diagnosa Penyakit Ular melalui menu “Bantuan”. Perancangan antarmuka Bantuan dapat dilihat pada Gambar 25.



Gambar 25. Desain Antarmuka Bantuan

6. Desain Antarmuka Informasi Tentang *Developer*

Menu "Informasi Tentang *Developer*" menampilkan informasi mengenai aplikasi dan *developer* Sistem Pakar Diagnosa Penyakit Ular. Menu ini memiliki 2 (dua) tombol yang dapat pengguna gunakan untuk memberikan kitik dan saran melalui email *developer*, dan memberikan *rating* atau penilaian melalui *play store*. Perancangan antarmuka Informasi Tentang *Developer* dapat dilihat pada Gambar 26.



Gambar 26. Desain Antarmuka Informasi Tentang *Developer*

3.4 Metode Pengujian Sistem

Pengujian sistem dimaksudkan untuk menguji semua elemen–elemen perangkat lunak yang dibuat apakah sudah sesuai dengan yang diharapkan. Pendekatan kasus uji dalam penelitian ini adalah pengujian *black box* dengan metode *Equivalence Partitioning* (EP). Pengujian ini dilakukan dengan membagi domain masukan dari program ke dalam kelas-kelas sehingga *test case* dapat diperoleh. EP berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. EP didasarkan pada premis masukan dan keluaran dari suatu komponen yang dipartisi ke dalam kelas-kelas, menurut spesifikasi dari komponen tersebut, yang akan diperlakukan sama (ekuivalen) oleh komponen tersebut. Pengujian ini harus diyakinkan bahwa masukan yang sama akan menghasilkan respon yang sama pula. Metode EP digunakan pada pengujian aplikasi Sistem Pakar Penyakit Ular ini karena metode ini dapat digunakan untuk mencari kesalahan pada fungsi yang diberikan ke aplikasi dan dapat mengetahui kesalahan pada *interface* aplikasi sehingga dapat mengurangi masalah terhadap nilai masukan. Rancangan daftar pengujian disajikan pada Tabel 4.

Tabel 4. Daftar Pengujian

No.	Kelas Uji	Daftar Pengujian	Skenario Uji	Hasil yang Diharapkan
1	*User Interface*	Pengujian pada *icon* Pakar Ular	Klik *icon* Pakar Ular pada perangkat android *user*	Menampilkan tampilan *splash screen*
		Pengujian pada menu utama Pakar Ular	Klik tombol menu Diagnosa	Menampilkan tampilan menu Diagnosa

Lanjutan Tabel 4. Daftar Pengujian

No.	Kelas Uji	Daftar Pengujian	Skenario Uji	Hasil yang Diharapkan
1	*User Interface*	Pengujian pada menu utama Pakar Ular	Klik tombol menu Daftar Penyakit	Menampilkan tampilan menu Daftar Penyakit
			Klik tombol menu Informasi Tentang *Developer*	Menampilkan tampilan menu Tentang
			Klik tombol menu Bantuan	Menampilkan tampilan menu Bantuan
2	Fungsi Menu Diagnosa	Pengujian pada Menu Diagnosa	Pengujian konsultasi Penyakit Ular	Menampilkan pertanyaan konsultasi penyakit yang disediakan system
			Pengujian Hasil Diagnosa Penyakit Ular	Menampilkan hasil diagnosa penyakit yang diderita ular
3	Fungsi Menu Daftar Penyakit	Pengujian pada Menu Daftar Penyakit	Klik *list* penyakit	Menampilkan *detail* penyakit yang dipilih
4	Fungsi Menu Informasi Tentang *Developer*	Pengujian pada Menu Informasi Tentang *Developer*	Klik *icon e-mail*	Menampilkan halaman pesan baru untuk pengembang
5	Fungsi Menu Bantuan	Pengujian pada Menu Bantuan	Klik Bantuan	Menampilkan halaman bantuan

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang didapat berdasarkan hasil penelitian yang telah dilakukan adalah sebagai berikut :

1. Aplikasi Pakar Ular telah berhasil dibangun sebagai sarana penunjang untuk pemelihara ataupun peternak ular dalam mendiagnosa penyakit ular.
2. Sistem pakar yang dibangun dapat memberikan hasil diagnosa berdasarkan fakta-fakta yang telah diberikan.
3. Aplikasi dapat membantu pemelihara ataupun peternak dalam mendiagnosa penyakit ular layaknya seorang pakar.
4. Hasil pengujian fungsional menunjukkan bahwa sistem pakar yang dibangun telah berjalan sesuai dengan yang diharapkan.
5. Penilaian penggunaan aplikasi melalui pengisian kuisioner menunjukkan bahwa aplikasi Pakar Ular memperoleh presentase penilaian rata-rata sebesar 86,14% dan termasuk dalam indeks kategori penilaian “Sangat Baik”.
6. Hasil diagnosis yang dilakukan oleh sistem telah sesuai dengan diagnosis manual yang dilakukan oleh pakar dengan persentase akurasi sebesar 94%.

5.2. Saran

Saran yang diberikan setelah dilakukan penelitian ini untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Penambahan data-data penanganan penyakit pada ular dan pengembangan proses diagnosa menggunakan gambar.
2. Aplikasi ini nantinya dapat dikembangkan sehingga kompatibel pada *platform* selain android, seperti iOS, dan Windows Phone.

DAFTAR PUSTAKA

Andry. 2011. *Android A sampai Z*. PC plus: Jakarta.

Arhami, M. 2005. *Konsep Dasar Sistem Pakar*. Yogyakarta: Andi.

Azwar, S. 2011. *Sikap dan Perilaku. Dalam: Sikap Manusia Teori dan Pengukurannya 2nd ed.* Yogyakarta: Pustaka Pelajar.

Fowler, Martin. 2004. *UML Distilled Panduan Singkat Bahasa pemodelan Objek Standar, Edisi 3*. Andi Publishing, Yogyakarta.

Jiang, F., Y. Lu. 2012. Software testing model selection research based on yin-yang testing theory. In: *IEEE Proceeding of International Conference on Computer Science and Information Processing (CISP)*, pp. 590-594.

Kusrini. 2008. *Aplikasi Sistem Pakar*. Yogyakarta: Andi.

Kusumadewi, S. 2003. *Artificial Intelegence (Teknik dan Aplikasinya)*. Yogyakarta: Graha Ilmu.

Lee, W. M. 2011. *Beginning Android Application Development*. Wiley Publishing, Inc.

Meildy, Bayu. 2014. *Daftar Simbol*. [*Online*]. Tersedia : http://elib.unikom.ac.id/download.php?id=83238. Diakses pada tanggal 26 November 2016.

Nazruddin, Safaat H. 2012. *(Edisi Revisi) Pemograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Informatika, Bandung.

Nugroho. 2010. *Rekayasa perangkat lunak menggunakan UML dan Java* Yogyakarta: Andi.

Pressman, Roger S. 2001. *Software Engineering A Practitioner's Approach Fifth Edition*. McGraw-Hill Companies, Inc, New York.

Pressman, R.S. 2010. *Software Engineering: A Practitioner's Approach, 7th Edition*. McGraw-Hill, New York.

Sutojo, T., Edy, M., dan Vincent, S. 2011. Kecerdasan Buatan. Yogyakarta: Andi

Tomar, Aditya. 2016. *Flash Alert Notification System Using Android*. IJARCE Volume 5, Issue 9.

Uml-diagrams.org. 2014. *The Unified Modeling Language*. [*Online*] Tersedia: http://www.uml-diagrams.org/. Diakses pada 25 November 2016.