BAB II

LANDASAN TEORI

II.1. Sistem Pakar

Sistem pakar merupakan cabang dari *Artifical Intelligence* (AI) yang cukup tua karena sistem ini mulai dikembangkan ada pertengahan 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver* (GPS) yang dikembangkan oleh Newel dan Simon. Sampai saat ini sudah banyak sistem pakar yang dibuat, seperti MYCIN untuk diagnosa penyakit, DENDRAL untuk mengidentifikasi struktur molekul campuran yang tak dikenal, XCON & XSEL untuk membantu konfigurasi sistem komputer besar, SOPHIE untuk analisis sirkuit elektronik, Prospector digunakan di bidang geologi untuk membantu mencari dan menemukan deposit, FOLIO digunakan untuk membantu memberikan keputusan bagi seorang manager dalam stok dan investasi, DELTA dipakai untuk pemeliharaan lokomotif listrik diesel, dan sebagainya.

Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini muncul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *Knowledge assistant* (T. Sutojo, dkk; 2011: 159-160).

Berikut adalah beberapa pengertian sistem pakar :

1. Turban

   “Sistem pakar adalah sebuah sistem yang menggunakan pengetahuan manusia dimana pengetahuan tersebut dimasukkan ke dalam sebuah computer dan kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia”. (T. Sutojo, dkk; 2011: 160).

2. Jackson

   “Sistem pakar adalah program computer yang merepresentasikan dan melakukan penalaran dengan pengetahuan beberapa pakar untuk memecahkan masalah atau memberikan saran”. (T. Sutojo, dkk; 2011: 160).

3. Luger dan Stubblefield

   “Sistem pakar adalah program yang berbasiskan pengetahuan yang menyediakan solusi ‘kualitas pakar’ kepada masalah – masalah dalam bidang (domain) yang spesifik”. (T. Sutojo, dkk; 2011: 160).

II.1.1. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, di antaranya:

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.

2. Membuat seorang yang awam bekerja layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Handal. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem komputer lainmembuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, sistem pakar dapat bekerja dengan informasi yang lengkap. Pengguna dapat merespon dengan: “tidak tahu” atau “tidak yakin” pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawabannya.
10. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula yang bekerja sebagai sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

II.1.2. Kekurangan Sistem Pakar

Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, di antaranya:

1. Biaya yang sangat mahal.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersedian pakar.
3. Sistem pakar tidak 100% bernilai benar.

II.1.3. Ciri – ciri Sistem Pakar

Ciri – ciri dari sistem pakar adalah sebagai berikut:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data – data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan – alasan dengan cara yang dapat dipahami.
4. Bekerja dengan kaidah/rule tertentu.
5. Mudah dimodifikasi.
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna.

II.1.4. Area Permasalahan Aplikasi Sistem Pakar

Biasanya aplikasi sistem pakar menyentuh beberapa area permasalahan berikut:

1. Interprestasi: menghasilkan deskripsi situasi berdasarkan data –data masukan.
2. Prediksi: memperkirakan akibat yang mungkin terjadi dari situasi yang ada.
3. Diagnosis: menyimpulkan suatu keadaan berdasarkan gejala – gejala yang diberikan (*symptoms*).
4. Desain: melakukan perancangan berdasarkan kendala – kendala yang diberikan.
5. Planning: merencanakan tindakan – tindakan yang akan dilakukan.
6. Monitoring: membandingkan hasil pengamatan dengan proses perencanaan.
7. Debugging: menentukan penyelesaian dari suatu kesalahan sistem.
8. Reparasi: melaksanakan rencana perbaikan.
9. *Instruction*: melakuakan instruksi untuk diagnosis, debugging, dan perbaikkan kinerja.
10. Kontrol: melakuakn kontrol terhadap hasil interprestasi, diagnosis, debugging, monitoring, dan perbaikan tingkah laku sistem.

II.1.5. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar meliputi enam hal berikut ini:

II.1.5.1. Kepakaran (*Expertise*)

Kepakaran merupakan suatu pengatahuan yang diperoleh dari pelatihan, membaca dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seorang yang bukan pakar. (T. Sutojo, dkk; 2011: 163).

Kepakaran itu sendiri meliputi pengetahuan tentang:

1. Fakta – fakta tentang bidang permasalahan tertentu.
2. Teori – teori tentang bidang permasalahan tertentu.
3. Aturan – aturan dan prosedur – prosedur menurut bidang permasalahn umumnya.
4. Aturan heuristic yang harus dikerjakan dalam suatu situasi tertentu.
5. Strategi global untuk memecahkan permasalahan.
6. Pengetahuan tentang pengetahuan (*meta knowledge*)

II.1.5.2. Pakar (*Expert*)

Pakar adalah seorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahkan masalah dan memberi nasehat. 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.(T. Sutojo, dkk; 2011: 163).

Jadi seseorang pakar harus mampu melakukan kegiatan – kegiatan berikut:

1. Mengenali dan memformulasikan permasalahan.

2. Memecahkan permasalahan secara cepat dan tepat.
3. Menerangkan pemecahannya.
4. Belajar dari pengalaman.
5. Merestrukturasi pengetahuan.
6. Memecahkan aturan – aturan.
7. Menentukan relevansi.

II.1.5.3. Pemindahan Kepakaran (*Transferring Expertise*)

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.

II.1.5.4. Inferensi (*Inferencing*)

Inferensi adalah sebuah prosedur (program) 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. (T. Sutojo, dkk; 2011: 164).

II.1.5.5. Aturan – aturan (*Rule*)

Kebanyakan software sistem pakar komersial adalah sistem yang berbasis *rule* (*rule-based systems*), yaitu pengetahuan disimpan terutama dalam bentuk *rule*, sebagai prosedur – prosedur pemecahan masalah. (T. Sutojo, dkk; 2011: 165).

II.1.5.6. Kemampuan Menjelaskan (*Explanation Capability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dilakukan dalam subsistem yang disebut subsistem penjelasan (*explanation*). Bagian dari sistem ini mmungkinkan sistem untuk memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi – opersinya.

Karakteristik dan kemampuan yang dimiliki sistem pakar berbeda dengan sistem konvensional. Perbedaan ini ditunjukkan pada tabel II.1.

Tabel II.1 Perbandingan antara sistem koonvensional dengan sistem pakar

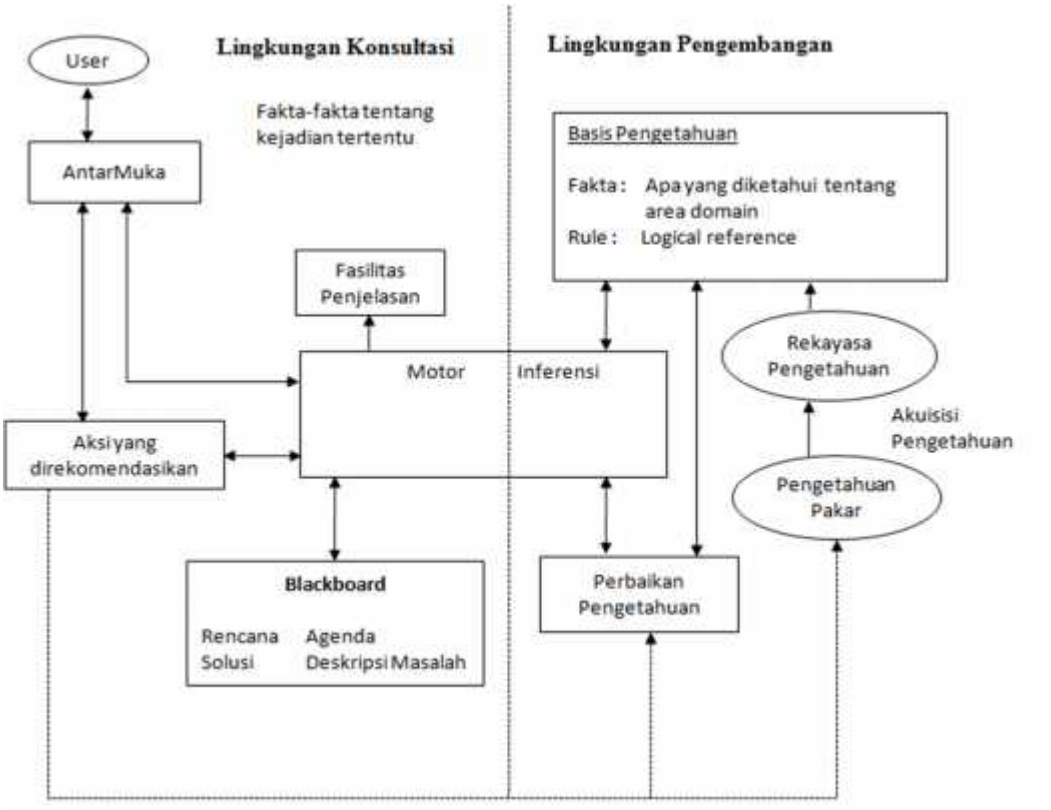
Sistem Konvensional	Sistem Pakar
Informasi dan pemrosesannya biasanya digabungkan dalam suatu program.	Basis pengetahuan dipisahkan secara jelas dengan mekanisme inferensi.
Program tidak membuat kesalahan (yang membuat kesalahan: pemrogram atau pengguna).	Program dapat berbuat kesalahan.
Biasanya tidak menjelaskan mengapa data masukan diperlukan atau bagaimana output dihasilkan.	Penjelasan merupakan bagian terpenting dari semua sistem pakar.
Peruubahan program sangat menyulitkan.	Perubahan dalam aturan – aturan mudah untuk dilakukan.
Sistem hanya bias beroperasi setelah lengkap atau selesai.	Sistem dapat beroperasi hana dengan aturan – aturan yang sedikit (sebagai prototipe awal)

Eksekusi dilakukan langkah demi langkah (algoritmik).	Eksekusi dilakukan dengan menggunakan heuristik dan logika pada seluruh basis pengetahuan.
Perlu informasi lengkap agar bisa beroperasi.	Dapat beroperasi dengan informasi yang tidak lengkap atau mengandung ketidakpastian.
Manipulasi efektif dari basis data yang besar.	Manipulasi efektif dari basis pengetahuan yang besar.
Menggunakan data.	Menggunakan pengetahuan.
Tujuan utama: efisiensi	Tujuan utama: efektivitas.
Mudah berurusan dengan data kuantitatif.	Mudah berurusan dengan data kualitatif.
Menangkap, menambah, dan mendistribusikan akses ke data numerik atau informasi.	Menangkap, menambah, dan mendistribusikan akses ke pertimbangan dan pengetahuan.

(Sumber : T. Sutojo, dkk, 2011; 165 – 166)

II.1.6. Struktur Sistem Pakar

Ada dua bagian penting dari sistem pakar, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen – komponennya dan memperkenalkan pengetahuan kedalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat sistem pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1 menunjukkan komponen – komponen yang penting dalam sebuah sistem pakar.



Gambar II.1 Komponen – komponen dalam sistem pakar
(Sumber : T. Sutojo, dkk, 2011; 167)

Keterangan :

1. Akuisisi Pengetahuan

   subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya kedalam basis pengetahuan dengan format tertentu (dalam bentuk representasi pengetahuan).

2. Basis Pengetahuan (*Knowledge Base*)

Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikna masalah. Basis pengetahuan terdiri dari dua elemen dasar, yaitu fakta dan rule.

3. Mesin Inferensi (*Inference Engine*)

   Mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi atau kesimpulan.

4. Daerah Kerja (*Blackboard*)

   Untuk merekam hasil sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang sedang terjadi, sistem pakar membutuhkan *blackboard*, yaitu area pada memori yang berfungsi sebagai basis data.

5. Antarmuka Pengguna (*User Interface*)

   Digunakan sebagai media komunikasi antara pengguna dan sistem pakar. Komunikasi ini paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik.

6. Subsistem Penjelasan (*Explanation Subsystem/Justifier*)

   Berfungsi memberi penjelasan kepada pengguna, bagaimana suatu kesimpulan dapat diambil.

7. Sistem Perbaikan Pengetahuan (*Knowledge Refining System*)

   Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pengetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuannya sehingga dapat dipakai pada masa mendatang.

8. Pengguna (*User*)

   Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (*training*) dari berbagai permasalahan yang ada.

II.1.7. Tim Pengembangan Sistem Pakar

1. *Domain Expert* adalah pengetahuan dan kemampuan seorang pakar untuk menyelesaikan masalah terbatas pada keahliannya saja. Misalnya seorang pakar penyakit jantung, ia hanya mampu menangani masalah – masalah yang berkaitan dengan penyakit jantung saja. Ia tidak bisa menyelesaikan masalah – masalah ekonomi, politik, hokum, dan lain – lain. Keahlian inilah yang dimasukan dalam sistem pakar.
2. *Knowledge Engineer* (Perekayasa Pengetahuan) adalah orang yang mampu mendesain, membangun, dan menguji sebuah sistem pakar.
3. *Programmer* adalah orang yang membuat program sistem pakar, mengode domain pengetahuan agar dapat dimengerti oleh komputer.

4. *Project manager* adalah pemimpin dalam tim pengembangan sistem pakar.
5. *End-User* (biasanya disebut *user* saja) adalah orang yang menggunakan sistem pakar.

II.2. Penyakit Meningitis

Meningitis adalah suatu penyakit infeksi yang menyerang selaput pelapis otak dan sumsum tulang belakang. Penyakit ini pertama kali ditemukan pada tahun 1805 pada saat terjadi wabah di Geneva, Swiss. Setiap tahun kejadian penyakit ini terus meningkat, yag menurut Badan kesehatan Dunia (WHO), diperkirakan ada 223.000 kasus baru pada tahun 2002. Penyebaran penyakit ini sangat cepat sehingga dapat mengakibatkan kejadian endemik (angka kejadiannya selalu ada setiap tahunnya meskipun dalam jumlah yang kecil) dan kejadian epidemik/wabah (tingginya angka kejadian yang sebelumnya tidak ada). Ironisnya, meskipun mendapatkan penanganan yang cepat dengan pengobatan yang maksimal, jumlah kematian akibat meningitis masih sangat tinggi, yakni sebanyak 5% -10% orang meninggal dalam 24 - 48 jam setelah timbul gejala. Selain itu, angka kecacatan yang timbul akibat penyakit ini mencapai 20% dari kasus yang selamat. (dr.J.B.Suharjo B.Cahyono, Sp.PD, dkk; 2010 :136-137)

Gambar II.2 Gambar MRI Bakteri Meningitis
(Sumber : Lorrie Klosterman ; 2007 :15)

Bagian kuning dari gambar MRI menunjukkan bakteri meningitis menginfeksi permukaan otak dan sumsum tulang belakang. Penyebab- penyebab dari meningitis meliputi :

1. Bakteri pionik yang disebabkan oleh bakteri pembentuk pus, terutama meningokokus, pneumokokus, dan basil influenza.
2. Virus yang disebabkan oleh agen-agen virus yang sangat bervariasa
3. Organisme jamur

II.2.1 Anatomi dan fisiologi selaput otak

Secara anatomi menigen menyelimuti otak dan medula spinalis. Jaringan otak dan medula spinalis dilindungi oleh tulang tengkorak dan tulang belakang, serta tiga lapisan jaringan penyambung atau menigen yaitu pia meter, arakhnoid, dan dura meter . Masing- masing merupakan suatu lapisan yang terpisah dan kontinu. Antara

lapisan pia meter dan arakhnoid terdapat penghubung yang disebut trabekula. Dura meter juga disebut pakhimening, sedangkan pia meter dan arakhnoid bersama-sama disebut leptomening. (Arif Muttaqin; 2008 : 13-14).

1. Lapisan Luar (Durameter)

Durameter merupakan tempat yang tidak kenyal yang membungkus otak, sumsum tulang belakang, cairan serebrospinal dan pembuluh darah. Durameter terbagi lagi atas durameter bagian luar yang disebut selaput tulang tengkorak (periosteum) dan durameter bagian dalam (meningeal) meliputi permukaan tengkorak untuk membentuk falks serebrum, tentorium serebelum dan diafragma sella. (Arif Muttaqin; 2008 : 14).

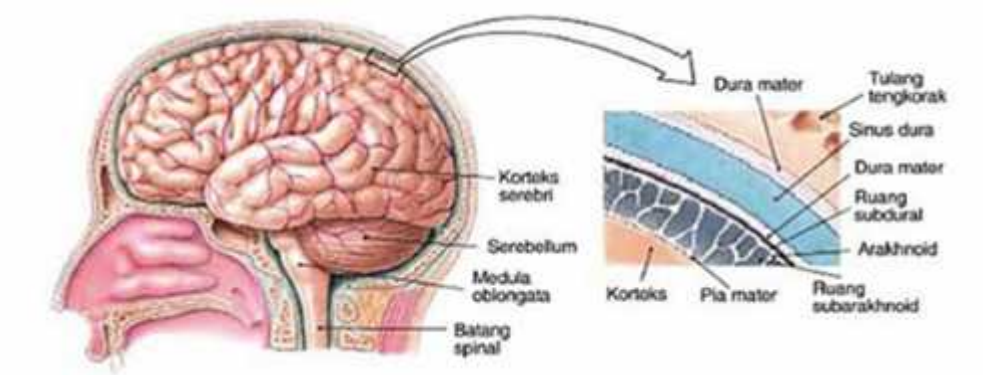
2. Lapisan Tengah (Arakhnoid)

Disebut juga selaput otak, merupakan selaput halus yang memisahkan durameter dengan piameter, membentuk sebuah kantung atau balon berisi cairan otak yang meliputi seluruh susunan saraf pusat. Ruangan diantara durameter dan arakhnoid disebut ruangan subdural yang berisi sedikit cairan jernih menyerupai getah bening. Pada ruangan ini terdapat pembuluh darah arteri dan vena yang menghubungkan sistem otak dengan meningen serta dipenuhi oleh cairan serebrospinal. (Arif Muttaqin; 2008 : 14).

3. Lapisan Dalam (Piameter)

Lapisan piameter merupakan selaput halus yang kaya akan pembuluh darah kecil yang mensuplai darah ke otak dalam jumlah yang banyak. Lapisan

ini melekat erat dengan jaringan otak dan mengikuti gyrus dari otak. Ruangan diantara arakhnoid dan piameter disebut sub arakhnoid. Pada reaksi radang ruangan ini berisi sel radang. Disini mengalir cairan serebrospinalis dari otak ke sumsum tulang belakang. (Arif Muttaqin; 2008 : 15).



Gambar II.3 Gambar hubungan antara otak, tulang tengkorak, dan menigen dilihat dari sisi lateral
(Sumber : Arif Muttaqin; 2008 :15).

II.2.2 Jenis – Jenis Penyakit Meningitis

II.2.2.1 Meningitis Virus

Tipe dari meningitis ini sering disebut meningitis aseptis. Tipe ini biasanya disebabkan oleh berbagai jenis penyakit yang disebabkan virus seperti gondok, herpes, simpleks, dan herpes zoster. Eksudat yang biasanya terjadi pada meningitis bakteri tidak terjadi pada meningitis virus dan tidak ditemukan organisme pada kultur cairan otak. Peradangan terjadi pada seluruh korteks serebri dan lapisan otak. Mekanisme atau respons dari jaringan otak terhadap virus bervariasi bergantung pada jenis sel yang terlibat. (Arif Muttaqin ; 2008 : 161)

Virus yang menyebabkan meningitis ada di dalam air liur dan lendir dari seseorang yang terinfeksi. Virus dapat ditularkan ke orang lain melalui bersin, batuk, mencium, atau dengan berbagi minum dan peralatan makan. Nyamuk membawa virus dan dapat menyebabkan meningitis ketika mereka menggigit seseorang yang terinfeksi, meskipun hal ini kurang umum dibandingkan dengan melalui kontak langsung pada manusia. Untungnya, hampir semua orang yang terinfeksi meningitis virus bisa sembuh dalam beberapa minggu, bahkan tanpa obat. Pemulihan ini terjadi karena sistem kekebalan tubuh seseorang yang dapat menghancurkan infeksi virus**.** **(**Lorrie Klosterman ; 2007 :17)

II.2.2.2 Meningitis Bakterial

Meningitis bakterial adalah suatu keadaan ketika menigen atau selaput dari otak mengalami peradangan akibat bakteri. Sampai saat ini bentuk paling signifikan dari meningitis adalah tipe bakterial. Bakterial paling sering dijumpai pada meningitis bakteri akut, yaitu *Neiserria meningitidis* (meningitis meningokokus), *Streptocuccus pneumonia* (pada dewasa), dan *Haemophilus influenzae* (pada anak-anak dan remaja). Ketiga organisme ini menyebabkan sekitar 75% kasus meningitis bakteri. Bentuk penularannya melalui kontak langsung, yang mencakup droplet dan secret dari hidung dan tenggorokan yang membawa kuman (paling sering) atau infeksi dari orang lain. Akibatnya, banyak yang tidak berkembang menjadi infeksi tetapi menjadi pembawa (*carrier).* Insiden tertinggi pada meningitis disebabkan oleh bakteri gram negative yang terjadi pada lansia sama seperti pada seseorang yang menjalani bedah

saraf atau seseorang yang mengalami gangguan respons imun. (Arif Muttaqin, ; 2008 : 161)

Meningitis bakteri dapat mematikan. Bakteri sering mereproduksi lebih cepat daripada virus. Bakteri ini menghasilkan racun yang disebut toksin yang berbahaya bagi tubuh dan dapat menyebabkan banyak kerusakan dalam waktu singkat. Beberapa orang yang terinfeksi meningitis bakteri meninggal dalam waktu gejala pertama mereka seperti sakit kepala parah, demam, leher kaku, kebingungan mental, dan sebuah ruam. Itulah mengapa penting bagi setiap orang untuk mengetahui tanda-tanda meningitis, terutama karena beberapa gejala yang mirip dengan flu. (Lorrie Klosterman ; 2007 :18)

Tabel II.2 Tabel Gejala Flu dan Meningitis Bakterial

Flu atau meningitis bakteri?
Kemungkinan gejala flu: • Demam yang mungkin tinggi • Ringan sampai sedang sakit kepala • Nyeri otot secara keseluruhan • Batuk • Kelelahan • Gejala tingkat off, dari biasanya meningkatkan **Kemungkinan Gejala meningitis** • Sakit kepala parah, dijelaskan oleh banyak pasien yang pernah mereka alami • Sakit leher ekstrim yang membuat sulit bagi seseorang untuk menyentuh dagunya ke dada • Demam dengan variabel yang bisa rendah atau tinggi • Tangan dan kaki dingin selama demam • Sensitivitas ekstrim terhadap cahaya • Kaki dan lengan sangat sakit • Ruam kulit yang semakin memburuk dengan cepat • Mengantuk dan kebingungan mental • Pengetatan otot mendadak disebut kejang

(Sumber : Lorrie Klosterman ; 2007 : 19)

II.3. *Certainty Factor* (Faktor Kepastian)

Teori *Certainty Factor* (CF) diusulkan oleh Shortliffe dan Buchanan pada 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti "mungkin", "kemungkinan besar", "hampir pasti". Untuk mengakomodasi hal ini kita menggunakan *Certainty Factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi (T. Sutojo, dkk; 2011: 194).

Ada dua cara dalam mendapatkan tingkat keyakinan (CF) dari sebuah rule, yaitu :

1. Metode '*Net Belief*' yang diusulkan oleh E.H. Shortliffe dan B.G. Buchanan

CF (Rule) = MB(H, E) – MD(H, E) ……(2.1)

$$MB(H,E) = \begin{cases} 1 & P(H) = 1 \\ \dfrac{\max[P(H|E),P(H)] - P(H)}{\max[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{……(2-2)}$$

$$MD(H,E) = \begin{cases} 1 & P(H) = 1 \\ \dfrac{\min[P(H|E),P(H)] - P(H)}{\max[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{……(2-3)}$$

Di mana:

CF (Rule) = factor kepastian

MB(H,E) = *measure of belief* (ukuran kepercayaan) terhadap hipotesis H, jika diberikan *evidence* E (antara 0 dan 1)

MB(H,E) = *measure of disbelief* (ukuran ketidakpercayaan) terhadap hipotesis H, jika diberikan *evidence* E (antara 0 dan 1)

P(H) = probabilitas kebenaran hipotesis H

P(H|E) = probabilitas bahwa H benar karena fakta E

(Sumber : T. Sutojo, dkk, 2011; 194)

Contoh II.1

Seandainya seorang pakar penyakit kelamin menyatakan bahwa probabilitas seseorang berpenyakit phimosis adalah 0.02. Dari data lapangan menunjukkan bahwa dari 100 orang penderita penyakit phimosis, 40 orang mimiliki gejala kulup berminyak, hitung factor kepastian bahwa phimosis disebabkan oleh adanya kulup berminyak.

Jawab :

P(Phimosis) = 0.02

P(Phimosis | Kulup Berminyak) = 40/100 = 0,4

MB(H,E) $= \frac{\max[p(H|E),p(H)]-p(H)}{\max[1,0]-p(H)}$

$= \frac{\max[0,4\ 0,02]-0,02}{1-0,02} = \frac{0,4-0,02}{1-0,02} = 0,39$

MD(H,E) $= \frac{\min[p(H|E),p(H)]-p(H)}{\min[1,0]-p(H)}$

$= \frac{\min[0,4\ 0,02]-0,02}{0-0,02} = \frac{0,02-0,02}{-0,02} = 0$

CF = 0,39 – 0 = 0,39

Rule : IF (Gejala = Kulup Berminyak) THEN Penyakit = Phimosis (CF = 0,39)

2. Dengan cara mewawancarai seorang pakar

Nilai CF(Rule) didapat dari interpretasi "term" dari pakar, yang diubah menjadi nilai CF tertentu sesuai table berikut :

Tabel II.3. Uncertain Term CF

Uncertain Term	CF
Definitely not (pasti tidak)	- 1.0
Almost certainly not (hampir pasti tidak)	- 0.8
Probably not (kemungkinan besar tidak)	- 0.6
Maybe not (mungkin tidak)	- 0.4
Unknow (tidak tahu)	- 0.2 to 0.2
Maybe (mungkin)	0.4
Probably (kemungkinan besar)	0.6
Almost certainly (hampir pasti)	0.8
Definitely (pasti)	1.0

(Sumber : T. Sutojo, dkk, 2011; 196)

Contoh II.2 :

Pakar :

Jika batuk dan panas, maka '*hampir dipastikan*' (*Almost certainly*) penyakitnya adalah *influenza*.

Rule : IF (batuk AND panas) THEN penyakit = *influenza* (CF = 0.8)

II.3.1. Kelebihan dan Kekurangan Metode Certainty Factors

Kelebihan metode *Certainty Factors* adalah :

1. Metode ini cocok dipakai dalam sistem pakar yang mengandung ketidakpastian.

2. Dalam sekali proses perhitungan hanya dapat mengolah 2 data saja sehingga keakuratan data dapat terjaga.

Sedangkan kekurangan metode *Certainty Factors* adalah :

3. Pemodelan ketidakpastian yang menggunakan perhitungan metode *certainty factors* biasanya masih diperdebatkan.
4. Untuk data lebih dari 2 buah, harus dilakukan beberapa kali pengolahan data (T. Sutojo, dkk; 2011: 204).

II.4. PHP

PHP atau *Hypertext Preprocessor* merupakan bahasa berbentuk script yang ditempatkan dalam server dan dieksekusi di dalam server untuk selanjutnya ditransfer dan di baca oleh client. Php juga bias disisipkan dalam bahasa HTML. Php Pertama kali diciptakan oleh seorang pria berkewarganegaraan Denmark yang bernama Rasmus Lerdorf pada tahun 1995. Banyak Programmer yang tertarik untuk mengembangkan php karena bersifat Open Source. Pada awal peluncurannya, php hanya dibuat untuk diintegrasikan dengan *Web Server* Apache. Namun sekarang, php juga bias bekerja dengan *Web Server* seperti PWS (*Personal Web Server*), IIS (*Internet Information Server*), dan Xitami. (Andrea Adelheid & Khairil Nst; 2012: 2)

II.5. MySQL

MySQL (*My Structure Query Language*) adalah sebuah perangkat lunak system manajemen basis data SQL (*Database Management System*). MySQL merupakan DBMS yang *multithread, multi-user* yang bersifat gratis dibawah lisensi GNU *General Public Licence* (GPL). MySQL dimiliki dan disponsori oleh sebuah perusahaan Swedia, yaitu MYSQL AB. (Anhar, ST; 2010: 21-22)

Beberapa Kelebihan MySQL, antara lain :

a. MySQL dapat berjalan dengan stabil pada berbagai system operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, dan masih banyak lagi .
b. Bersifat Open Source, MySQL didistribusikan secara open source(gratis), dibawah lisensi GNU General Public Licence (GPL).
c. Bersifat Multiuser, mySQL dapat digunakan oleh beberapa user dalam waktu bersamaan tanpa mengalami masalah.
d. MySQL memiliki kecepatan yang baik dalam menangani query(perintah SQL). Dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
e. Dari segi security atau keamanan data, MySQL memiliki beberapa lapisan security, seperti level subnet mask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password yang terenskrip.

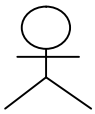
II.6. *Unified Modelling Language* (UML)

Unified Modelling Language adalah bahasa standar yang digunakan untuk menjelaskan dan memvisualisasikan artifak dari proses analisis dan desain berorientasi obyek. UML menyediakan standar pada notasi dan diagrma yang bisa digunakan untuk memodelkan suatu sistem. UML dikembangkan oleh 3 pendekar "berorientasi obyek", yaitu Grady Booch, Jim Rumbaugh dan Ivar Jacobson. UML menjadi bahasa perspektif obyek antara user dengan developer, antara developer dengan developer antara developer analisis dengan developer desain, dan antara developer desain dengan developer pemrograman. (Julius Hermawan :7)

Berikut beberapa notasi dalam UML diantaranya :

1. Actor

   Actor adalah segala sesuatu yang berinteraksi dengan system aplikasi computer. Jadi actor ini bisa berupa orang, perangkat keras, atau mungkin objek lain dalam sistem yang sama. Biasanya yang dilakukan actor adalah memberikan informasi pada system dan\ atau memerintahkan system untuk melakukan sesuatu.



Gambar II.4 *Notasi Actor*
(Sumber : Julius Hermawan :14)

2. Class

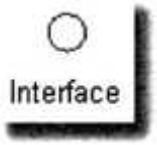
   Class merupakan pembentuk utama dari system berorientasi objek karena

class menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. Class digunakan untuk mengimplementasikan interface.

Gambar II.5 *Notasi* class
(Sumber : Julius Hermawan :14)

3. Interface

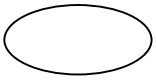
   Interface merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam interface dijabarkan oleh operasi dalam class.



Gambar II.6 *Notasi* interface
(Sumber : Julius Hermawan :15)

4. Use Case

   Use Case menjelaskan urutan kegiatan yang dilakukan actor dan sistem untuk mencapai suatu tujuan tertentu. Walaupun mejelaskan kegiatan namun use case hanya menjelaskan apa yang dilakukan oleh actor dan sistem, bukan bagaimana actor dan sistem melakukan kegiatan tersebut



Gambar II.7 *Notasi* Use Case
(Sumber : Julius Hermawan :16)

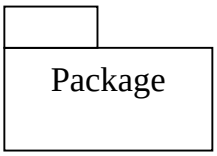
5. Interaction

   Digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

Gambar II.8 *Notasi* Interaction
(Sumber : Julius Hermawan :18)

6. Package

   Package adalah Kontainer atau wadah konseptual yang digunakan untuk mengelompokkan elemen-elemen dari sistem yang sedang dibangun, sehingga bisa dibuat model yang lebih sederhana.tujuannya untuk mempermudah penglihatan (*visibility)* dari model yang sedang dibangun.



Gambar II.9 *Notasi Package*
(Sumber : Julius Hermawan :19)

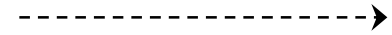
7. Note

   Note digunakan untuk memberikan keterangan dan komentar tambahan dari suatu elemen sehingga bias langsung terlampir dalam model. Note bias ditempelkan ke semua notasi yang lain.

Gambar II.10 *Notasi Note*
(Sumber : Julius Hermawan :19)

8. Dependency

   Depedency merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain.



Gambar II.11 *Notasi Dependency*
(Sumber : Julius Hermawan :20)

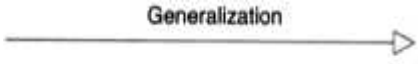
9. Association

   Association menggambarkan navigasi antar class (*Navigation*), beberapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*Multiolicity antar class*), dan apakah suatu class menjadi bagian dari class lainnya(Aggregation).

Gambar II.12 *Notasi* Association
(Sumber : Julius Hermawan :21)

10. Generalization

    Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.



Gambar II.13 *Notasi Generalization*
(Sumber : Julius Hermawan :22)

11. Realization

    Realizzation menunjukkan hubungan bahwa elemen yang ada di bagian

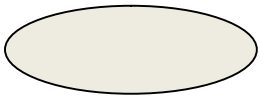
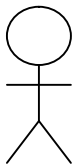
tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.

Gambar II.14 *Notasi Realization*
(Sumber : Julius Hermawan :22)

II.6.1 Use Case Diagram

Menggambarkan fungsionalitas dari sebuah sistem (apa fungsinya), yang mempresentasikan sebuah interaksi antara actor dan sistem (sebuah pekerjaan). Misalnya menambah data/membuat laporan.Aktor alaha sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

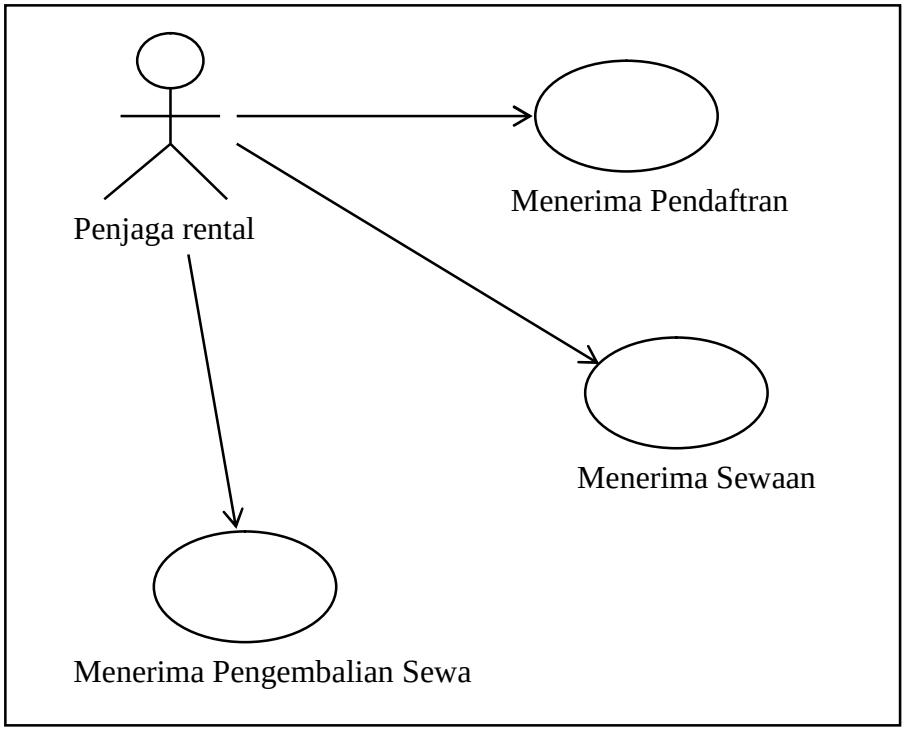
Tabel II.4 Simbol – simbol yang digunakan dalam *Use Case Diagram*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit unit yang bertukar pesan antar unit dengan actor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *Use case*.
	*Actor* atau Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas – tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor beriteraksi dengan *use case*, tetapi tidak memiliki kontrol terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang

	meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
<<*include*>>	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case lain*, contohnya adalah pemanggilan sebuah fungsi program.
<<*extend*>>	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

***(*Sumber : Windu Gata ; 2013 : 4)**

Berikut contoh bentuk *use case diagram :*

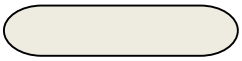
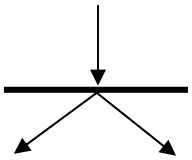
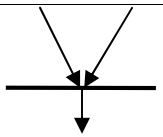
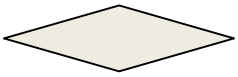
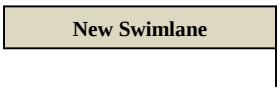


Gambar II.15 Contoh *Use Case Diagram*
(Sumber : Julius Hermawan :22)

II.6.2 Activity Diagram

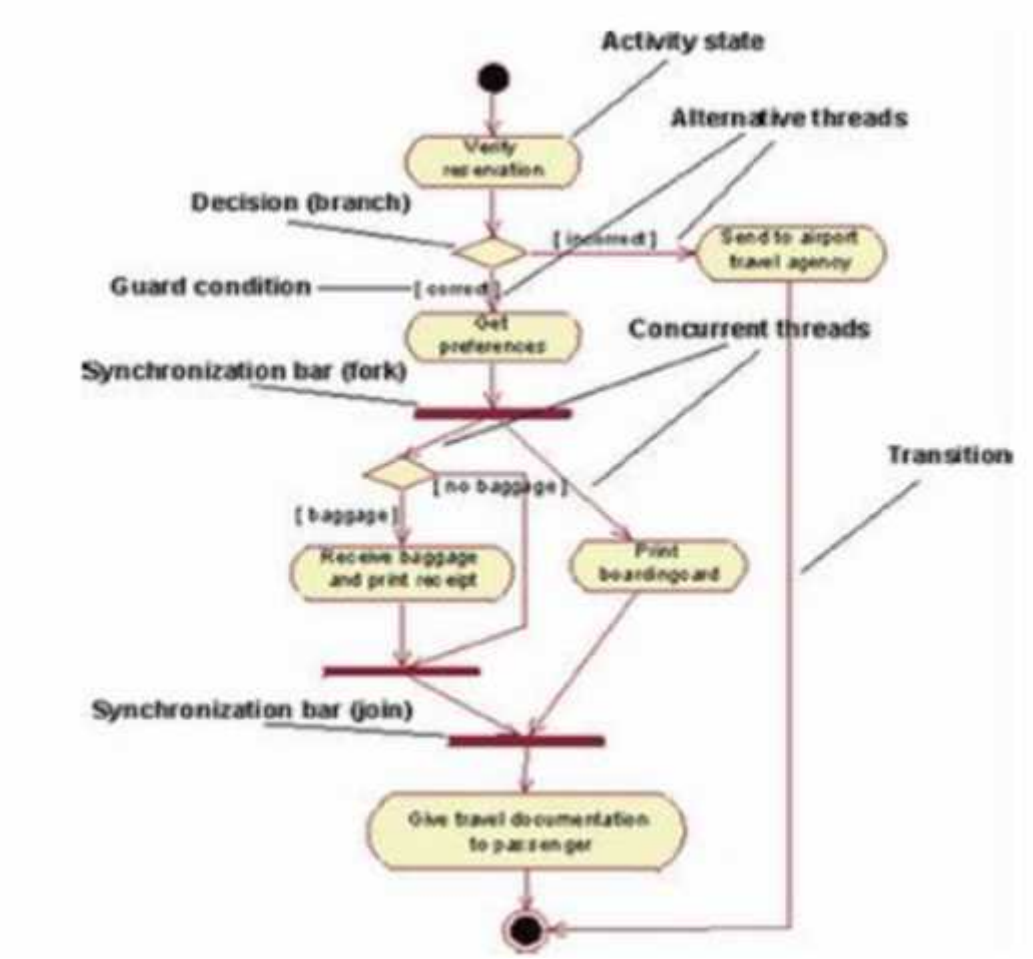
Activity diagram menggambarkan aktifitas – aktifitas objek, state, transisi state dan event. Dengan kata lain kehiatan diagram alur kerja menggambarkan prilaku sistem untuk aktivitas .

Tabel II.5 Simbol – simbol yang digunakan dalam *Activity Diagram*

Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas
	*End Point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join* (penggabungan) atau *Rake*, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*atau *false*.
	*Swimlane*, pembagian activity diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

Berikut contoh bentuk *Activity diagram* :



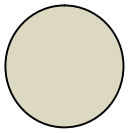
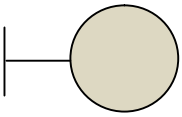
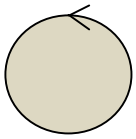
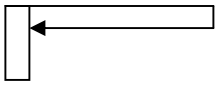
Gambar II.16 Contoh *Activity diagram*
(Sumber : Jurnal Informatika Mulawarman Vol 6 No.1 ; 2011 : 4)

II.6.3 Diagram Urutan (Sequence Diagram)

Diagram ini menggambarkan interaksi antarobjek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagiannya) berupa *message* yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertical (waktu0 dan dimensi horizontal (objek – objek yang terkait). Biasa digunakan untuk menggambarkan scenario atau rangkaian langkah-langkah yang dilakukan sebagai

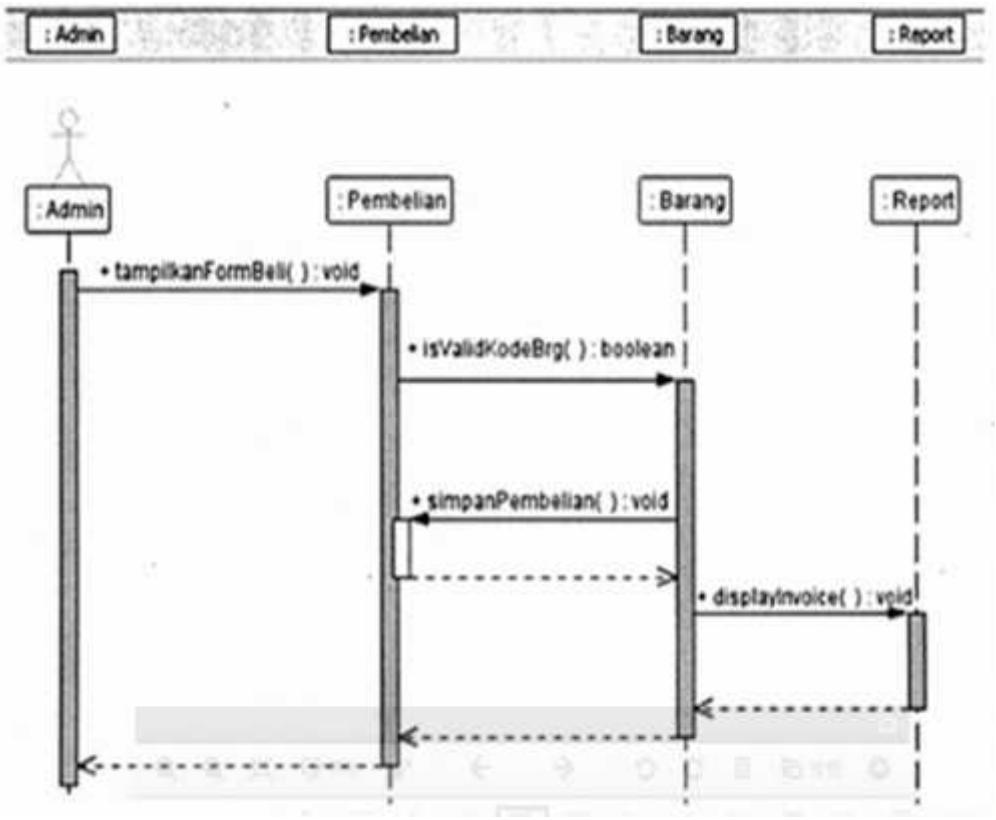
respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari sebuah aktivitas tertentu, kemudian berproses mengikuti urutan tertentu, yang bias terlihat melalui *message* antarobjek.

Tabel II.6 Simbol – simbol yang digunakan dalam *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas – entitas yang membentuk gambaran awal sistem yang menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi interface atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan form cetak.
	*Control Class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek, *control objek* mengkoordinir pesan antara *boundary* dengan entitas
	Message, simbol mengirim pesan antara *class.*
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
	Lifeline, garis titik – titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata ; 2013 : 7)

Berikut contoh bentuk *Sequence Diagram* :



Gambar II.17 *Contoh Sequence Diagram*
(Sumber : Miftkhul Huda & Bunafit Komputer ; 2010 : 143)

II.6.4 Class Diagram (Diagram Kelas)

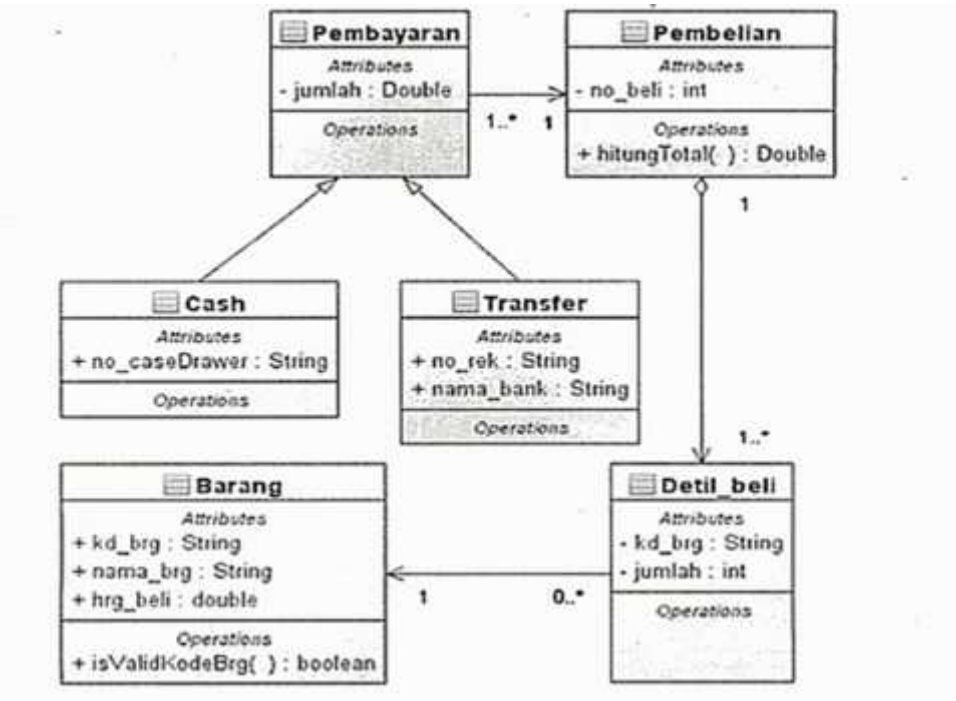
Class adalah sebuah spesifikasi objek, yang memiliki atribut/property dan layanan/fungsional (metode.fungsi).*Class diagram* menggambarkan struktur dan deskripsi kelas, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi dan lain-lain. Kelas memiliki tiga hal pokok : Nama (dan *stereotype*), Atribut, dan Metode.Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti.

Tabel II.7 Hubungan antar kelas

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh: 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 9)

Berikut contoh bentuk *Class Diagram* :



Gambar II.18 Contoh *Class Diagram*
(Sumber : Miftkhul Huda & Bunafit Komputer ; 2010: 143)

II.6.5 Normalisasi Data

Normalisasi data adalah proses di mana table-tabel pada database dites dalam hal kesalingtergantungan diantara field – field pada sebuah table. Misalnya jika pada sebuah table terdapat ketergantungan terhadap lebih dari satu field dalam table

tersebut, maka table tersebut harus di pecah menjadi banyak table. Banyaknya table tergantung dari seberapa banyak ketergantungannya. Tiap table hanya boleh memiliki sebuah fueld kunci yang menjadi ketergantungan dari field lainnya dalam table tersebut. (Wahana Komputer ; 2010: 32).

II.6.5.1 Bentuk Normal Tahap Pertama (1st Normal Form)

Sebuah table disebut 1NF jika :

a. Tidak ada baris yang duplikat dalam table tersebut.
b. Masing – masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah table berarti tabeel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.(Kusrini, M.Kom;2007: 41)

Tabel II.8 Tabel Bentuk Tidak Normal

Nim
Nama
Kode_Jur
Nama_Jur
Kepala_Jur
Kode_MK
Nama_MK
Nilai_MK

(Sumber : Ema Utami dan Sukrisno ; 2011 : 77)

Tabel II.9 Tabel Bentuk First Normal Form (1NF)

Nim*
Nama
Kode_Jur
Nama_Jur
Kepala_Jur

Mahasiswa

Nim**
Kode_MK
Nama_MK
Nilai_MK

Nilai

(Sumber : Ema Utami dan Sukrisno ; 2011 : 77)

II.6.5.2 Bentuk Normal Tahap Kedua (2 nd Normal Form)

Bentuk Normal Kedua (2NF) terpenuhi jika pada sebuah table semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh. Sebuah table dikatakan tidak memenuhi 2NF, jika ketergantungannya hanya bersifat persial (hanya tergantung pada sebagian dari primary key). (Kusrini, M.Kom ; 2007 : 41)

Berikut Contoh Bentuk Second Normal Form (2NF)

Tabel II.10 Bentuk Tabel Second Normal Form (2NF)

Nim*
Nama
Kode_Jur
Nama_Jur
Kepala_Jur

Mahasiswa

Kode_MK*
Nama_MK

Nilai

Nim**
Kode_MK**
Nilai_MK

Mata Kuliah

(Sumber : Ema Utami dan Sukrisno ; 2011 : 77)

II.6.5.3 Bentuk Normal Tahap Ketiga (3rd Normal Form)

Sebuah table dikatakan memenuhi bentuk normal ketiga (#NF), jika untuk setiap ketergantungan fungsional dengan notasi X-> A, dimana A mewakili semua atribut tunggal di dalam table yang tidak ada di dalam X, maka :

a. X haruslah superkey pada table tersebut
b. Atau A merupakan bagian dari primary key pada table tersebut. (Kusrini, M.Kom ; 2007 : 42)

Berikut Contoh Bentuk Third Normal Form (3NF)

Tabel II.11 Bentuk Tabel Third Normal Form (3NF)

Nim*
Nama
Kode_Jur

Mahasiswa

Kode_Jur*
Nama_Jur
Kepala_Jur

Jurusan

Kode_MK*
Nilai_MK

Mata Kuliah

Nama**
Kode_MK*
Nilai_MK

Nilai

(Sumber : Ema Utami dan Sukrisno ; 2011 : 78)