BAB II
LANDASAN TEORI

2.1 Sistem Pakar

Menurut (Triwahyudi, 2013) Sistem pakar (Expert System) merupakan cabang dari kecerdasan buatan (Artificial Intelligence) dan juga merupakan bidang ilmu yang mucul seiring perkembangan ilmu komputer saat ini.

a) Menurut Giarratano dan Riley: Sistem pakar adalah sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar. Sistem ini bekerja untuk mengadopsi pengetahuan manusia ke komputer yang menggabungkan dasar pengetahuan (knowledge base) dengan sistem inferensi untuk menggantikan fungsi seorang pakar dalam menyelesaikan suatu masalah.
b) Menurut Darkin: Merupakan suatu perangkat lunak yang dapat diperbanyak, kemudian dibagikan ke berbagai lokasi maupun tempat yang berbeda-beda untuk dapat digunakan.

Secara umum dikatakan bahwa sistem pakar adalah program komputer yang menirukan penalaran seorang pakar dengan keahlian pada suatu wilayah pengetahuan tertentu. Sistem pakar merupakan program “artificial inteligence” (”kecerdasan buatan” atau AI) yang menggabungkan basis pengetahuan dengan mesin inferensi. Hal ini merupakan bagian perangkat lunak spesialisasi tingkat tinggi atau bahasa pemrograman tingkat tinggi (*High Level Language*), yang berusaha menduplikasi fungsi seorang pakar dalam satu bidang keahlian tertentu. Program ini bertindak sebagai konsultan yang cerdas atau penasihat dalam suatu lingkungan keahlian tertentu, sebagai hasil himpunan pengetahuan yang telah dikumpulkan dari beberapa orang pakar.

Dengan demikian seorang awam sekalipun bisa menggunakan sistem pakar itu untuk memecahkan berbagai persoalan yang ia hadapi dan bagi seorang

ahli, sistem pakar dapat dijadikan alat untuk menunjang aktivitasnya yaitu sebagai sebagai asisten yang berpengalaman.

2.1.1 Struktur Sistem Pakar

Menurut (Triwahyudi, 2013) sistem pakar memiliki dua bagian utama, yaitu :

a. Lingkungan pengembangan (development environment), yaitu bagian yang digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
b. Lingkungan konsultasi (consultation environment), yaitu bagian yang digunakan oleh pengguna yang bukan pakar untuk memperoleh pengetahuan.

Sistem pakar memiliki 3 bagian, yaitu antar muka pengguna, mesin inferensi dan basis pengetahuan.

1. Antar Muka Pengguna Antar muka pengguna adalah perangkat lunak yang menyediakan media komunikasi antara pengguna dengan sistem. Antar muka pengguna memberikan fasilitas informasi dan berbagai keterangan yang bertujuan untuk membantu mengarahkan alur penelusuran masalah sehingga ditemukan sebuah solusi.
2. Mesin Inferensi Mesin Inferensi (Inference Engine), merupakan otak dari Sistem Pakar, juga dikenal sebagai penerjemah aturan (rule interpreter). 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. Kerja mesin inferensi meliputi:
   - Menentukan aturan mana akan dipakai
   - Menyajikan pertanyaan kepada pemakai, ketika diperlukan.
   - Menambahkan jawaban ke dalam memori Sistem Pakar.
   - Menyimpulkan fakta baru dari sebuah aturan.
   - Menambahkan fakta ke dalam memori.

3. Basis Pengetahuan Basis pengetahuan merupakan inti program sistem pakar. Pengetahuan ini merupakan representasi pengetahuan dari seorang pakar.

2.1.2 Keuntungan dan Kelemahan Sistem

Pakar Keuntungan sistem pakar :

a. Memungkinkan orang awam dapat mengerjakan pekerjaan para ahli.
b. Dapat melakukan proses secara berulang secara otomatis.
c. Menyimpan pengetahuan dan keahlian para pakar.
d. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
e. Meningkatkan hasil dan produktivitas, karena sistem pakar dapat bekerja lebih cepat dari manusia.
f. Memiliki kemampuan untuk bekerja dengan informasi yang tidak lengkap dan mengandung ketidakpastian.
g. Memiliki kemampuan untuk mengakses pengetahuan.
h. Meningkatkan kapabilitas dalam penyelesaian masalah.
i. Menghemat waktu dalam pengambilan keputusan.

Kelemahan sistem pakar :

a. Biaya yang diperlukan untuk membuat, memelihara, dan mengembangkannya sangat mahal.
b. Sulit dikembangkan, hal ini erat kaitannya dengan ketersediaan pakar di bidangnya.
c. Sistem pakar tidak 100% benar karena seseorang yang terlibat dalam pembuatan sistem pakar tidak selalu benar. Oleh karena itu perlu diuji ulang secara teliti sebelum digunakan.
d. Kepakaran sangat sulit diekstrak dari manusia.
e. Pendekatan oleh setiap pakar untuk suatu situasi atau problem bisa berbeda- beda, meskipun sama-sama benar.
f. Sangat sulit bagi seorang pakar untuk mengabstraksi atau menjelaskan langkah mereka dalam menangani masalah.

g. Sistem pakar bekerja baik untuk suatu bidang yang sempit.

h. Istilah yang dipakai oleh pakar dalam mengekspresikan fakta seringkali terbatas dan tidak mudah dimengerti oleh orang lain.

i. Transfer pengetahuan dapat bersifat subyektif dan biasa.

2.2 Metode *Case Based Reasoning* (CBR)

Menurut (Fitria, Lestari, S, And Wawan, D. (2018) Case Based Reasoning adalah metode untuk yang digunakan untuk mengimplementasikan sistem diagnosa komputer kedalam aplikasi didunia nyata. Case Base Reasoning (CBR) juga dapat digunakan untuk menganalisa suatu masalah sesuai dengan kasus yang dihadapi dan untuk selanjutnya mengklasifikasikan kasus tersebut berdasarkan pada pengalaman masa lalu.

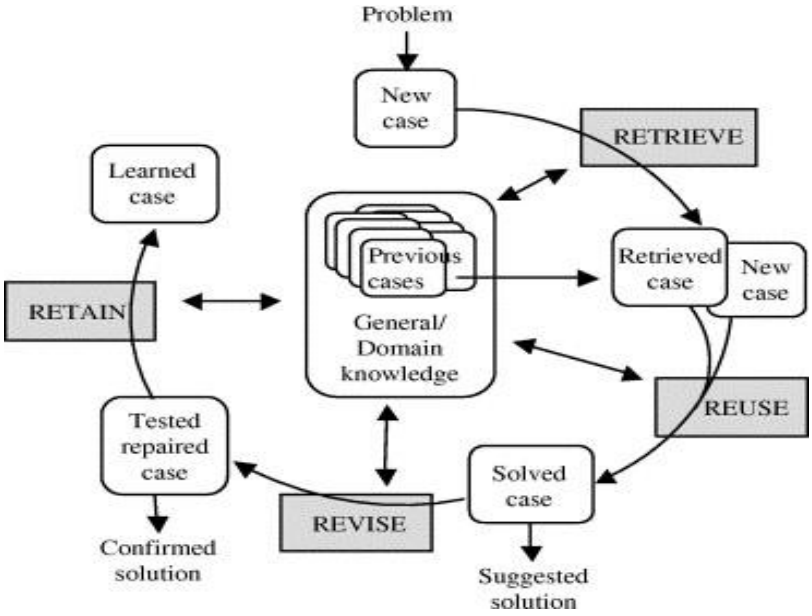
Case Based Reasoning (CBR) menggunakan pendekatan kecerdasan buatan (Artificial Intelligent) yang menitikberatkan pemecahan masalah dengan didasarkan pada knowledege dari kasus-kasus sebelumnya. Apabila ada kasus baru maka akan disimpan pada basis pengetahuan sehingga sistem akan melakukan learning dan knowledge yang dimiliki oleh sistem akan bertambah (Yulmaini And Lestari,. S. 2012).

Tahapan CBR, secara keseluruhan model CBR Cycle dapat digambarkan dengan proses sebagai berikut (Yulmaini, Lestari, S. And Antonio, Y. (2020):

a. *Retrieve*,merupakan proses untuk mendapatkan kembali kasus terdahulu yang serupa dengan kasus yang sedang dihadapi.
b. *Reuse*,merupakan proses untuk menggunakan kembali informasi dan pengetahuan dalam kasus terdahulu untuk menyelesaikan masalah yang dihadapi.
c. *Revise*,merupakan proses memperbaiki solusi yang telah ada sebelumnya.
d. *Retain*,merupakan proses penyimpanan kasus baru dan solusinya

untuk digunakan dalam menyelesaikan kasus berikutnya.

Keempat proses di atas akan terus dilakukan ketika menghadapi kasus baru. Model CBR tersebut dapat disajikan pada Gambar 2.1.



Gambar 2.1 Tahapan CBR

Kelebihan CBR (Yulmaini, Lestari, S. And Antonio, Y. (2020):

1. Pada CBR tidak memerlukan pemahaman bagaimana menyelesaikan masalah sehingga mengurangi dampak penambahan informasi pengetahuan.
2. Pada CBR pengetahuan diperoleh dengan cara mengumpulkan kejadian- kejadian yang telah terjadi dan tidak memerlukan suatu model yang explisit.
3. Pada CBR memiliki kemampuan untuk belajar, cara yang dilakukan yaitu dengan menambahkan kasus baru seiring waktu berjalan dan tanpa perlu menambahkan aturan baru atau mengubah yang sudah ada
4. Kemampuan untuk mendukung justifikasi dengan menawarkan kasus lampau lebih diutamakan.

Kekurangan CBR (Yulmaini, Lestari, S. And Antonio, Y. (2020):

1. Memerlukan perhitungan yang kompleks dan melakukan pengukuran kesamaan antar dua objek agar dapat menyelesaikan masalah secara keseluruhan.
2. Seringkali menghadapi kesulitan dalam menentukan fitur-fitur dari kasus agar bisa dibandingkan.
3. Semakin banyak kasus-kasus tersimpan dalam databased, maka dalam pencarian kasus yang paling mirip dengan kasus baru membutuhkan waktu lebih lama
4. Solusi yang ditawarkan belum tentu yang terbaik, hal ini karena tergantung pada kualitas kasus-kasus sebelumnya yang tersimpan dalam databased. Oleh karena itu tahap Revise menjadi hal yang penting untuk dilakukan sehingga mengurangi tingkat kesalahan.

2.3 Naive Bayes Classifier

Menurut (Rosandy T. 2016) Naive Bayes Clasifier merupakan sebuah media klasifikasi yang berakar pada teorema bayes. Proses pendekatan *Naïve Bayes Classifier* mengasumsikan bahwa ada atau tidaknya suatu fitur pada suatu kelas tidak berhubungan dengan ada atau tidaknya fitur lain di kelas yang sama.

Teori bayes adalah pendekatan statistik fundamental dalam pengenalan pengenalan pola. Pendekatan ini berdasarkan kuantifikasi trade-off antara mengklasifikasikan berbagai keputusan yang dibuat dengan menggunakan probabilitas dan biaya yang disebabkan dalam keputusan tersebut (Artaye, K. 2015)

Pada saat klasifikasi, pendekatan *Bayes* akan menghasilkan tabel kategori yang paling tinggi nilai probabilitasnya yaitu VMAP (Maximum Apriori Probability) dengan atribut inputan G1, G2,… Gn (Wahyudi,S Dan Sofie R. 2014). Pernyataan berikut dapat dituliskan dengan formula bayes sebagai berikut:

$$P\ (ai|vj) = \frac{\mathrm{nc} + \mathrm{mp}}{\mathrm{n} + \mathrm{m}}$$

Dimana :

nc = Jumlah record pada data learning yang v = vj

dan a = aj p = l/ banyaknya jenis class/penyakit

m = jumlah parameter/gejala

n = jumlah record pada data learning yang v=vj /tiap class

Persamaan pada rumus formula bayes diatas dapat diselesaikan

melalui perhitungan sebagai berikut :

a. Menentukan nilai nc untuk setiap class.

b. Menghitung nilai P (ai|vj) dan menghitung nilai P(vj), di mana

c. Menghitung P(ai|vj) untuk tiap v

d. Menentukan hasil klarifikasi yaitu v yang memiliki hasil perkalian terbesar.

Untuk menjelaskan teorema *Naive Bayes*, perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang akan dianalisis (Natalius, 2011). Karena itu, *teorema Bayes* di atas dapat disesuaikan sebagai berikut:

$$P\ (V\ |G1, \dots Gn) = \frac{\mathrm{P(V)}\ x\ P(G1, G2, \dots Gn\ |\ V)}{\mathrm{P(G1, G2, \dots Gn)}}$$

Dimana variabel V merepresentasikan kelas penyakit (*Disease*), sementara variabel G1,G2,...Gn merepresentasikan fitur-fitur petunjuk (gejala) yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel dengan fitur (gejala) tertentu dalam kelas V (posterior) adalah peluang munculnya kelas V (sebelum masuknya sampel tersebut (disebut juga prior) dikali dengan peluang munculnya fitur-fitur (gejala- gejala) sampel pada kelas V (disebut juga *likelihood*), dibagi dengan peluang kemunculan fitur-fitur (gejala-gejala) sampel secara global (disebut juga *evidence*).

Menggunakan *teorema* pada persamaan (4) dapat ditulis sebagai berikut :

$$\boldsymbol{Vmap} = \boldsymbol{argmax}vjev\frac{\mathrm{P\ (V)}x\ P(G1, G2, \dots Gn\ |Vj)}{P\ (G1, G2, \dots Gn)}$$

Dimana:

V_{MAP} = Probabilitas (penyakit) tertinggi

P(Vj) = Peluang jenis penyakit kedelai kej (*prior*)

P(G1,G2,…Gn|Vj) = Peluang atribut-atribut *input*an jika diketahui keadaan *Vj P(G1,G2,…Gn)* = Peluang atribut-atribut *input*an

Karena nilai P(G1,G2,…Gn) nilainya konstan untuk semua Vj maka persamaan tersebut dapat ditulis :

$$V_{MAP} = \text{argmaxvj}\epsilon v\ P(Vj) \times P(G1, G2,\ldots Gn|Vj)$$

Untuk perhitungan P(Vj) x P(G1, G2,…Gn|Vj) akan semakin sulit karena jumlah gejala P(Vj) x P(G1, G2,… Gn|Vj) dapat semakin besar.

Perhitungan *Naïve Bayes* adalah sebagai berikut:

$$P\ (Gi\ |Gj) = \frac{\text{nc} + \text{m.p}}{\text{n} + \text{m}}$$

Dimana :

nc = jumlah *record* pada data training yang V = Vj

dan G = Gi p = peluang jenis penyakit (*prior*)

m = jumlah gejala

n = jumlah *record* pada data training yang V = Vj tiap class (penyakit)

Persamaan diatas dapat diselesaikan melalui perhitungan sebagai berikut :

1. Menentukan nilai nc untuk setiap class
2. Menentukan nilai P(Vj) atau nilai prior. Pada penelitian ini nilai prior diperoleh dari penyakit yang dipilih dan tidak dipilih jika dipilih bernilai 1 jika tidak bernilai 0
3. Menghitung nilai P(Gi|Vj)

$$Vmap = ArgmaxVjev\ (Vj)\ N\ I\ P\ (G1\ |Vj)$$

Dimana :

P(Vj) = Peluang jenis penyakit kej nilai *prior* penyakit ke_j

$$P\ (Gi\ |Gj) = \frac{\mathrm{nc} + \mathrm{m.\,p}}{\mathrm{n} + \mathrm{m}}$$

4. Menghitung P(Vj) x P(Gi|Vj) untuk tiap V
5. Menentukan hasil klasifikasi yaitu V yang memiliki hasil perkalian yang terbesar

2.4 *Unified Modelling Language* (UML)

Menurut (Purwati, N., Halimah. And Raharjo, A.2018) *Unified Modeling Language (UML)* adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak, UML menawarkan sebuah standar untukmerancang model sebuah sistem.

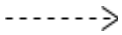
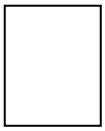
2.4.1 Komponen-Komponen UML

Menurut (Nurjoko 2018) Tujuan pemodelan dalam kerangka pengembangan sistem adalah sebagai sarana analisis, pemahaman, visualisasi, dan komunikasi antar tim pengembang yang beranggotakan beberapa/banyak anggota. Beberapa diagram dalam UML yang akan digunakan dalam membantu pengembangan sistem adalah:

a. Use case Diagram

Menurut (Arfida, S., Amnah., And Wibowo, H. (2018) Use Case diagram merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, Use Case tidak hanya sangat penting pada tahap analisis, tetapi juga sangat penting untuk perancangan, untuk mencari kelas-kelas yang terlibat dalam aplikasi, serta untuk melakukan pengujian.

Tabel 2.1 Simbol *Use Case*

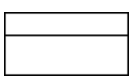
NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya Komputasi

a) *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan *object* beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Tabel 2.2. Simbol Class Diagram

b)

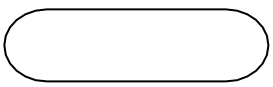
NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.

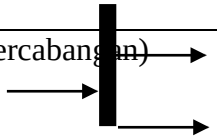
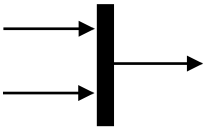
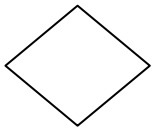
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

b. Activity Diagram

1) *Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.
2) *Activity diagram* merupakan *state* diagram khusus, yang sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan perilaku internal sebuah sistem dan interaksi antar subsistem, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Tabel 2.3 Simbol *Activity* Diagram

No.	Simbol	Keterangan
1	*Start State* 	*Start state* adalah sebuah kondisi awal sebuah *object* sebelum ada perubahan keadaan. Start state digambarkan dengan sebuah lingkaran solid.
2.	*End State* 	*End state* adalah menggambarkan ketika objek berhenti memberi respon terhadap sebuah event. *End state* digambarkan dengan lingkaran solid di dalam sebuah lingkaran kosong.
3.	*State/Activities* 	*State* atau *activities* menggambarkan kondisi sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya.

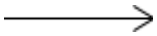
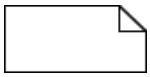
4.	*Fork* (Percabangan) 	*Fork* atau percabangan merupakan pemisalah beberapa aliran konkuren dari suatu aliran tunggal.
5.	*Join* (Penggabungan) 	*Join* atau penggabungan merupakan pengabungan beberapa aliran konkuren dalam aliran tunggal.
6.	*Decision* 	*Decision* merupakan suatu logika aliran konkuren yang mempunyai dua cabang aliran konkuren.

c. *Sequence Diagram*

Sequence diagram secara grafis menggambarkan bagaimana objek berinteraksi antara satu sama lain melalui pesan pada sebuah *use case* atau operasi.

Tabel 2.4 Simbol *Sequence Diagram*

NO	**GAMBAR**	**NAMA**	**KETERANGAN**
1		*State*	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2		*Initial Pseudo State*	Bagaimana objek dibentuk atau Diawali
3		*Final State*	Bagaimana objek dibentuk dan Dihancurkan

4		*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
5		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
6.		*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2.5 Database

Database adalah kumpulan file-file yang mempunyai kaitan antara satu file dangan file yang lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu. (Nugroho, 2011).

Istilah-istilah yang digunakan dalam basis data:

1) *File* : merupakan kumpulan dari atribut *record-record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda-beda dalam data *value*-nya.
2) *Record* : merupakan kumpulan dari elemen-elemen yang saling berhubungan atau berkaitan menginformasikan tentang *entry* secara lengkap.
3) *Field* : merupakan sekumpulan tanda-tanda yang berbentuk kesatuan tersendiri, merupakan bagian terkecil dari *record* dan bentuknya unik dijadikan *field* kunci yang dapat mewakili *record*-nya.
4) *Entity* : merupakan tempat kejadian atau konsep yang informasikan direkam.

2.6 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database,* terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis (Nugroho,2011).

2.7 Metode Pengembangan Perangkat Lunak.

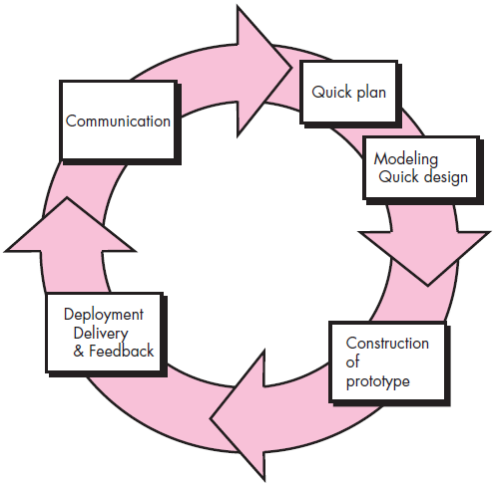
Menurut (Pressman R, S. 2012) Metode Prototype adalah proses pembuatan model sederhana software yang mengijinkan pengguna memiliki gambaran dasar tentang program serta melakukan pengujian awal. Prototype memberikan fasilitas bagi pengembang dan pemakai untuk saling berinteraksi selama proses pembuatan, sehingga pengembang dapat dengan mudah memodelkan perangkat lunak yang akan di buat.

Metode ini cocok digunakan untuk mengembangkan sebuah perangkat lunak yang dikembangkan kembali. Metode ini dimulai dengan pengumpulan kebutuhan pengguna. Kemudian membuat sebuah rancangan kilat yang selanjutnya akan dievaluasi kembali sebelum di produksi secara benar. Prototype bukanlah merupakan sesuatu yang lengkap, tetapi sesuatu yang harus dievaluasi dan dimodifikasi kembali. Segala perubahan dapat terjadi

pada saat prototype dibuat untuk memenuhi kebutuhan pengguna dan saat yang sama memungkinkan pengembangan untuk lebih memahami kebutuhan pengguna secara baik.

Berikut adalah tahapan dalam metode prototype :

1. Komunikasi (*Communication*) dan pengumpulan data awal, yaitu komunikasi dengan klien dan user untuk menentukan kebutuhan.
2. Perencanaan cepat (*Quick Plan*), yaitu pembuatan perencaan analisis terhadap kebutuhan pengguna.
3. Pemodelan perancangan cepat (*Modeling Quick Design*), yaitu membuat rancangan desain program.
4. Pembentukan prototype (*Construction of prototype*), yaitu pembuatan aplikasi berdasarkan dari pemodelan desain yang telah dibuat.
5. Penyerahan sistem dan umpan balik (*Development Delevery and Feedback*), yaitu memproduksi perangkat secara benar sehingga dapat digunakan oleh pengguna.



Gambar 2.2 Diagram Prototype

Penjelasan dari gambar 2.2 diatas adalah sebagai berikut :

a) Tahap pertama adalah *communication* dan pengumpulan data awal yaitu tahap suatu perencanaan yang di lakukan, mulai dari menciptakan dan melaksanakan proses untuk memastikan bahwa perencanaan tersebut berkualitas tinggi, terpercaya, efisiensi biaya.
b) Tahap kedua adalah *quick plan* yaitu analisis terhadap kebutuhan pengguna.
c) Tahap ketiga adalah *modelling quick* design yaitu pembuatan desain secara umum untuk selanjutnya dikembangkan kembali.
d) Tahap keempat adalah *construction of prototype* adalah pembuatan perangkat prototype termasuk pengujian dan penyempurnaan.
e) Tahap kelima adalah *deployment delivery and feedback* adalah tahap penyerahan sistem ke pengguna dan umpan balik.

2.8 Website

2.8.1 Pengertian Website

Menurut (Arkhiansyah, Y. 2018) *Website* adalah kumpulan halaman-halaman. Yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan

yang saling terkait, yang masing masing dihubungkan dengan jaringan-jaringan halaman.

2.8.2 Web Server

Suatu aplikasi yang jalan pada suatu situs web dan bertanggung jawab untuk merespon permintaan file dari *web browser*. Dalam melakukan permintaan suatu halaman pada suatu situs *web browser* melakukan koneksi ke suatu server dengan protokol HTTP. Server akan menanggapi koneksi tersebut dengan mengirimkan isi file yang diminta dan memutuskan koneksi tersebut. *Web Browser* kemudian memformat informasi yang didapat dari server. Pada bagian *server, browser* yang berbeda dapat melakukan koneksi pada server yang sama untuk memperoleh informasi yang sama. Dalam memberikan halaman yang diminta web server dapat melakukan kerja sama dengan server lain seperti Aplikasi Server.

2.8.3 Web Browser

Software yang dijalankan pada komputer pengguna (*client*) yang meminta informasi dari server web dan menampilkannya sesuai dengan file data itu sendiri. Tugas utama dari *web browser* adalah mendapatkan dokumen dari web.

2.9 Black Box Testing

Menurut (Muhammad, F,A.,Karnila, S. And Kurniawan H. 2018) *Black Box Testing* merupakan pengujian terhadap fungsi program dengan cara menemukan kesalahan pada program. Pengujian dilakukan dengan cara melakukan validasi kebenaran input dan output yang sesuai. Kategori-kategori kesalahan yang diuji oleh *black box testing* adalah fungsi-fungsi yang salah atau hilang, kesalahan interface, kesalahan dalam struktur data atau akses database eksternal, kesalahan performa, kesalahan inisialisasi dan terminasi.

Berikut adalah langkah-langkah dari proses *black box testing*:

1. Menganalisa kebutuhan spesifikasi dari perangkat lunak.
2. Pemilihan jenis input yang memungkinkan menghasilkan output benar serta jenis input yang memungkinkan output salah pada perangkat lunak yang sedang diuji. Menentukan output untuk suatu jenis input.
3. Pengujian dilakukan dengan input-input yang telah benar-benar diseleksi.
4. Perbandingan output yang dihasilkan dengan output yang diharapkan.
5. Menentukan fungsionalitas yang seharusnya pada perangkat lunak yang sedang diuji.

2.10 Penelitian Terkait

Tabel. 2.5 Penelitian Terkait

No	Penulis	Judul (Tahun)	Penelitian
1	Wahyudi Setiawan dan Sofie Ratnasari	Sistem Pakar Diagnosis Penyakit Mata Menggunakan Naive Bayes Classifier (2014)	Penelitian ini menghasilkan sistem inferensi dengan perhitungan Naive Bayes Classifier.
2	Fitria, Sri L, Dan Wawan D.	Metode Case Based Reasoning (CBR) Pada Sistem Diagnosa Penyakit Kulit(2018)	Penelitian ini menggunakan Metode Case Based Reasoning(CBR), dimana CBR merupakan salah satu metode pendekatan berbasis pengetahuan untuk mempelajari dan memecahkan berdasarkan masalah pengalaman pada masa lalu yang disimpan didalam sistem yang dinamakan basis kasus.
3.	Yulmaini, dan Lestari, S.	Implementasi Case Base Reasoning (Cbr) Untuk Mengidentifikasi Penyakit Kulit Pada Bayi (2012).	Penelitian ini adalah Pemanfaatan Penalaran bayi.

4.	Triwahyudi, S.	Rancang Bangun Aplikasi Sistem Pakar Untuk Mendeteksi Penyakit Infeksi Saluran Kemih Dengan Menggunakan Metode Dempster Shafer (2013)	Penelitian ini menghasilkan diagnose yang lebih tepat dan mempunyai kepastian yang lebih kuat tanpa adanya perubahan apapun penambahan dan pengetahuannya.

Perbedaan penelitian ini dengan penelitian sebelumnya adalah:

1. Wahyudi S dan Ratnasari S (2014) membahas penyakit mata menggunakan metode Naive Bayes Classifier dan tidak membuat aplikasi sistem pakar, sedangkan dalam penelitian ini membahas penyakit organ pencernaan menggunakan metode Naive Bayes Classifier. Penelitian ini sebagai acuan untuk diambil data rumus dan pehitungan . Naive Bayes Classifier.

2. Fitria, Sri L, Dan Wawan D (2018) Membahas diagnosa penyakit kulit menggunakan Case Based Reasoning adanya sistem pendiagnosa penyakit kulit yang dibangun dapat memberikan acuan untuk langkah pembangunan sistem yang lebih baik kedepannyadan, tidak membuat aplikasi sistem pakar, sedangkan dalam penelitian ini membahas penyakit organ pencernaan menggunakan metode case based reasoning dan metode naïve bayes classifier untuk perhitungan dan ada system pakarnya.

3. Yulmaini, dan Lestari, S. (2012) Implementasi Case Base Reasoning (Cbr) Untuk Mengidentifikasi Penyakit Kulit Pada Bayi. Pemanfaatan penalaran berbasis kasus (pbk) untuk domain kesehatan khususnya mengidentifikasi penyakut kulit pada bayi. Tidak membuat aplikasi sistem pakar.

4. Triwahyudi, S. (2013) Rancang bangun aplikasi sistem pakar untuk mendeteksi penyakit infeksi saluran kemih dengan menggunakan metode dempster shafer dan menghasilkan diagnosa yang lebih tepat dan mempunyai kepastian yang lebih kuat, sedangkan dalam penelitian ini menggunakan metode naïve bayes classifier sebagai acuan untuk diambil data rumus dan pehitungan.