BAB II
LANDASAN TEORI

1.1 Penelitian Terdahulu

Salah satu acuan penulis dalam melakukan penelitian untuk memperbanyak teori dan referensi dalam mengkaji penelitian adalah mengacu pada penelitian terdahulu. Banyak penelitian tentang sistem pakar untuk diagnosis penyakit mulut dan gigi, namun menggunakan beberapa kriteria tertentu. Penulis hanya mengangkat penelitian dengan kasus yang berbeda namun metode yang digunakan hampir sama ataupun sebaliknya sebagai referensi dalam memperkaya bahan kajian. Berikut adalah penelitian terdahulu berupa jurnal atau skripsi terkait.

Tabel 2. 1.Penelitian Terdahulu Pertama

Nama Peneliti	Judul Penelitian	Hasil Penelitian
Melinda Sari, Sarjon Defit, Gunaidi Widi Nurcahyo	"Sistem Pakar Deteksi Penyakit pada Anak Menggunakan Metode *Forward Chaining"*	Sistem menerapkan pengetahuan sebanyak 25 gejalan dan 5 jenis penyakit. Nilai akurasi dari 20 sampel pengujian sebesar 90%.
Perbedaan	Jenis diagnosis penyakit dan hasil akurasi	

Sumber : Hasil kajian penulis, 2020.

Pada tabel 2.1 di atas adalah penelitian [34] dengan judul "Sistem Pakar Deteksi Penyakit pada Anak Menggunakan Metode *Forward Chaining*" berhasil melakukan diagnosis dengan nilai akurasi sebesar 90%.

Tabel 2. 2 Penelitian Terdahulu Kedua

Nama Peneliti	Judul Penelitian	Hasil Penelitian

Ahmad Aniq Noor Mutsaqof, Wiharto, dan Esti Suryani.	"Sistem Pakar Untuk Mendiagnosis Penyakit Infeksi Menggunakan *Forward Chaining"*	Pengujian dilakukan terhadap 50 pasien yang menghasilkan 6 kegagalan dalam mendeteksi penyakit infeksi. Nilai akurasi yang diperoleh adalah 88%.
Perbedaan	Jenis penyakit dari sistem pakar dan nilai akurasi	

Sumber : Kajian hasil penulis, 2020.

Pada tabel 2.2 di atas adalah penelitian [9] berjudul "Sistem Pakar Untuk Mendiagnosis Penyakit Infeksi Menggunakan *Forward Chaining"* berhasil melakukan penelitian dengan nilai akurasi 88%.

Tabel 2. 3 Penelitian Terdahulu Ketiga

Nama Peneliti	**Judul Penelitian**	**Hasil Penelitian**
Ilham Roni Yansyah, dan Sumijan	"Sistem Pakar Metode *Forward Chaining* untuk Mengukur Keparahan Penyakit Gigi dan Mulut*"*	Pengujian terhadap 10 data menghasilkan nilai akurasi sebesar 90%. Sistem mempunyai pengetahuan data 27 gejala dan 8 jenis penyakit.
Perbedaan	Nilai akurasi sistem dan pengetahuan	

Sumber : Kajian hasil penulis, 2020.

Pada tabel 2.3 di atas adalah penelitian [35] berjudul “Sistem Pakar Metode *Forward Chaining* untuk Mengukur Keparahan Penyakit Gigi dan Mulut*”* berhasil melakukan diagnosis dengan nilai akurasi sebesar 90%.

1.2 Penyakit Gigi dan Mulut

Kurangnya kesadaran dan pola hidup sehat dalam merawat gigi dan mulut merupakan faktor utama terserangnya berbagai macam masalah gigi dan mulut hingga terjangkitnya suatu penyakit. Kebanyakan orang awam tidak mengetahui penyakit gigi dan mulut, karena kurangnya informasi dan edukasi. Penanganan penyakit gigi dan mulut harus segera ditangani lebih cepat dan benar. Mengetahui jenis penyakit gigi dan mulut sejak awal sangatlah penting. Menurut [2] Penyakit gigi dan mulut yang paling banyak diderita masyarakat adalah penyakit karies gigi dan peradangan gusi. Penyebab utama kedua penyakit tersebut disebabkan oleh kebersihan mulut dan pola makan yang kurang baik.

Ada beberapa jenis penyakit yang menyerang organ gigi dan mulut. Penulis mengemukakan 7 jenis penyakit gigi dalam sistem pakar ini sebagai berikut :

2.2.1 Abrasi Gigi

Definisi : Abrasi merupakan keadaan abnormal dimana ada lapisan gigi yaitu email yang hilang dan terkikis, atau terkadang hingga lapisan yang lebih dari email yaitu dentin. Abrasi gigi disebabkan oleh gaya friksi (gesekan) langsung antara gigi dan objek eksternal [11].

Penyebab : Terjadinya abrasi pada gigi, dapat disebabkan oleh perilaku menyikat gigi, baik itu frekuensi menyikat gigi, jenis sikat gigi yang digunakan, hingga metode atau teknik yang digunakan [11].

2.2.2 Abses Gigi

Definisi : Secara harfiah abses merupakan suatu lobang yang berisi nanah dalam jaringan yang sakit [12].

Penyebab : Faktor lokal yang meliputi Plak (bakteri), kalkulus, Impaksi Makanan, Trauma Oklusi, Faktor Sistemik, dan Faktor Predisposisi [12].

2.2.3 Karies Gigi

Definisi : Karies gigi merupakan penyakit infeksi yang disebabkan oleh demineralisasi email dan dentin yang erat hubungannya dengan konsumsi makanan yang kariogenik. Umumnya anak-anak memasuki usia sekolah mempunyai resiko karies yang tinggi, karena pada usia sekolah ini anak-anak biasanya suka jajan makanan dan minuman sesuai keinginannya. Kebiasaan ini merupakan salah satu faktor penyebab yang multifaktorial [13].

Penyebab : Faktor multifaktorial artinya, karies dapat terjadi bila ada faktor penyebab yang saling berhubungan dan mendukung, yaitu host (saliva dan gigi), mikroorganisme, substrat dan waktu [14].

2.2.4 Gingivitis

Definisi : Gingivitis adalah bentuk penyakit periodontal yang ringan dengan tanda gejala klinis berupa gingiva berwarna merah, membengkak dan mudah berdarah tanpa ditemukan kerusakan tulang alveolar [15].

Penyebab : Terjadinya gingivitis berawal dari plak yang berakumulasi dalam jumlah banyak, inflamasi gingiva ini cenderung dimulai pada daerah papilla interdental dan menyebar pada

leher gigi. Lesi awal akan timbul dalam 2-4 hari dan akan menjadi gingivitis pada waktu 2-3 minggu kemudian [15].

2.2.5 Periodontitis

Definisi : Penyakit periodontal atau biasa disebut periodontitis merupakan kelainan yang sering dijumpai dan terjadi pada manusia dengan faktor resiko yang jelas berperan terhadap gangguan fungsi pengunyahan dan hilangnya gigi geligi [16].

Penyebab : Penyakit periodontal secara umum disebabkan oleh bakteri plak yang terdapat pada permukaan gigi, dimana plak merupakan deposit lunak berupa lapisan tipis *biofilm* yang berisi kumpulan mikroorganisme *patogen* seperti *Porphyromonas gingivalis, Actinobacillus actinomycetemcomitans, Prevotela intermedia,Tannerella forsythia* serta *Fusobacterium nucleatum* [16].

2.2.6 Pulpitis

Definisi : Pulpitis adalah peradangan pada pulpa gigi yang menimbulkan rasa nyeri. Pulpa adalah bagian gigi paling dalam, yang mengandung pembuluh darah dan saraf. Dalam pendiagnosisan penyakit ini, banyak sekali orang yang salah mengartikannya dengan penyakit gigi sensitiF [17].

Penyebab : Pulpitis terjadi ketika lapisan enamel dan dentin pelindung pulpa rusak. Ketika lapisan yang melindungi ini rusak, bakteri dapat masuk dengan mudah dan menyebabkan pembengkakan [17].

2.2.7 Halitosis

Definisi : Halitosis merupakan istilah untuk mendefinisikan bau tidak sedap dari pernafasan. Bau yang tidak sedap diakibatkan oleh bebasnya Volatile Sulfur Compound (VSCs) yang disebabkan oleh aktifitas pembusukan dari mikroorganisme gram negatif [18].

Penyebab : Penyebab halitosis biasanya karena kebersihan mulut yang buruk, karies yang dalam, penyakit periodontal, infeksi rongga mulut, mulut kering, mengonsumsi rokok, ulserasi mukosa, perikoronitis, sisa makanan dalam mulut serta tongue coating [19].

1.3 Sistem Pakar

Menurut [9] mengungkapkan "Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam*"*.

Secara umum sistem pakar ini dirancang untuk melakukan *medical check-up* secara mandiri agar menghasilkan suatu diagnosis awal penyakit mulut dan gigi. Cara kerja sistem pakar ini dimulai dari memasukan informasi data diri, gejala-gejala yang dirasakan dan yang tampak pada gigi dan mulut. Kemudian data-data yang dimasukan tadi akan diproses oleh mesin inferensi berdasarkan basis pengetahuan dari literature pendidikan spesialis kedokteran gigi dan mulut yang diperoleh juga dari metode wawancara dokter spesialis gigi. Nantinya, hasil atau *output* yang diberikan kepada *user* berisi diagnosis penyakit beserta dengan solusi penanganan penyakit yang didiagnosis. Sistem pakar diagnosis ini bekerja menggunakan perhitungan mesin inferensi metode *Forward Chaining* berdasarkan ciri-ciri dan gejala yang dirasakan *user*.

1.3.1 Struktur Sistem Pakar

Secara garis besar sistem pakar disusun berdasarkan dua bagian utama, yaitu:

1. Lingkungan konsultasi *(Consultation environtment)* yaitu lingkungan untuk proses konsultasi *user* dengan sistem pakar.
2. Lingkungan pengembangan *(Development environtment)* yaitu lingkungan *developer* untuk membangun sistem pakar.

1.3.2 Arsitektur Sistem Pakar

Di dalam membangun dan merancang sistem pakar diagnosis penyakit gigi dan mulut ini terdapat beberapa komponen wajib untuk menunjang fungsi kerja sistem pakar ini, yaitu :

1.3.2.1 Tampilan Antarmuka

Tampilan antarmuka adalah bagian visual dari sistem pakar. Tampilan antarmuka atau biasa disebut UI *(User Interface)* menampilkan semua informasi sistem pakar pada layar serta bertujuan untuk meningkatkan *usability* dan UX *(User Experience)*.

1.3.2.2 Mesin Inferensi

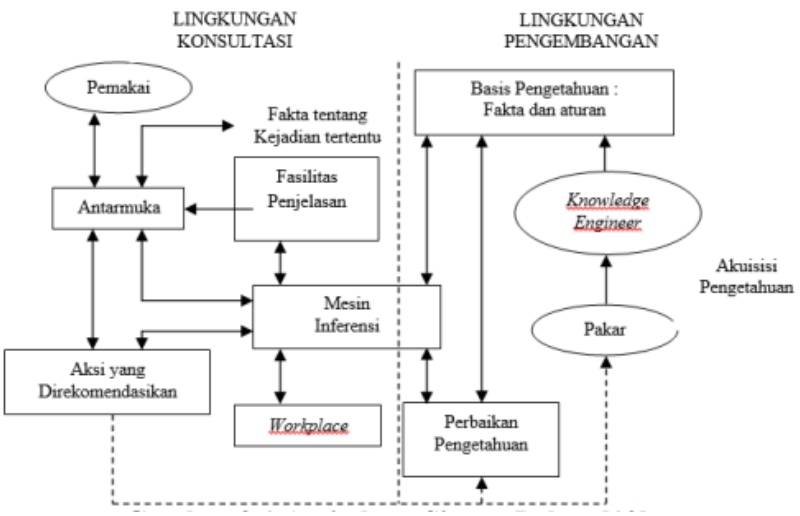
Komponen ini biasa disebut sebagai otaknya sistem pakar, karena mengandung penalaran metodologi yang biasa dipakai oleh seorang pakar untuk menyelesaikan masalah. Penalaran yang dimaksud adalah informasi yang ada di dalam basis pengetahuan dan memori kerja untuk memformulasikan kesimpulan. Metodologi dalam mesin inferensi yang biasa digunakan ada dua macam yaitu metode maju *(forward Chaining)* dan metode mundur *(backward chaining)*.

1.3.2.3 Basis Pengetahuan

Berisi pengetahuan, formulasi dan penyelesaian masalah. Komponen basis pengetahuan disusun oleh dua elemen yaitu fakta dan aturan. Fakta adalah informasi objek dalam permasalahan, sedangkan aturan adalah bagaimana cara memperoleh fakta baru dari fakta yang telah diketahui.

1.3.2.4 Memori Kerja

Sebagai tempat menyimpan fakta-fakta hasil dari proses konsultasi untuk kemudian diproses oleh mesin inferensi berdasarkan basis pengetahuan untuk menentukan suatu keputusan.



Gambar 2. 1.Sistem Pakar [9]

Kelebihan dan Kelemahan Sistem Pakar

Sebagai sebuah sistem yang bekerja dengan cara mengadopsi kemampuan pengetahuan manusia ke dalam sebuah perangkat komputer dengan tujuan untuk menyelesaikan permasalahan layaknya seorang pakar pastinya mempunyai beberapa kelebihan dan kelemahan dalam proses pengambilan suatu keputusannya. Berikut adalah uraian kelebihan dan kelemahan dari suatu sistem pakar.

Penelitian [20] mengungkapkan bahwa sistem pakar mempunyai kelebihan dan kekurangan, yaitu :

1. Membantu orang awam untuk menyelesaikan masalah 'tanpa' bantuan para pakar.
2. Meningkatkan kualitas dan produktivitas.
3. Mampu beroperasi dalam lingkungan yang berbahaya.
4. Memiliki kemampuan untuk mengakses pengetahuan dan keahlian para ahli baik yang biasa maupun yang langka.
5. Sebagai asisten para ahli sehingga meringankan pekerjaan para ahli.
6. Memiliki reabilitas.
7. Dapat menghemat waktu dalam pengambilan keputusan.

Adapun kekurangan dari sistem pakar adalah :

1. Tidak ada jaminan bahwa sistem pakar memuat 100% kepakaran yang diperlukan.
2. Pengembangan sistem pakar tergantung ada tidaknya pakar di bidangnya sehingga pengembangannya dapat terkendala.
3. Biaya untuk mendesain, mengimplementasikan dan memeliharanya dapat sangat mahal tergantung seberapa lengkap dan kemampuannya.

1.3.4 *Forward Chaining*

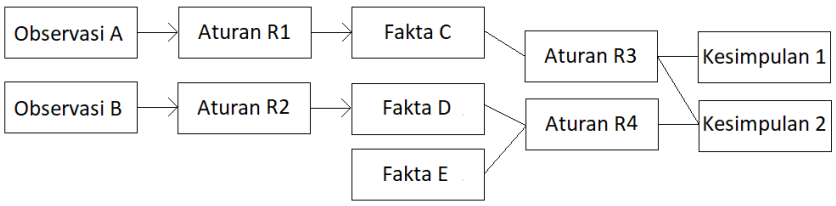
Perancangan sistem pakar ini dibangun menggunakan metode penalaran-penalaran yang dimulai dari fakta terlebih dahulu untuk menguji kebenaran

hipotesis yang disebut metode *Forward Chaining*. Mudahnya, metode *Forward Chaining* ini adalah suatu strategi pencarian dari sekumpulan data dan fakta dicari kesimpulan menjadi suatu solusi dari permasalahan.

Contoh sederhana dari metode *Forward Chaining* ini adalah pernyataan sebagai berikut:

"Jika cuaca mendung dan udara lembab, maka hari ini akan hujan." Berdasarkan pernyataan tadi, metode *Forward Chaining* juga biasa disebut metode *IF-THEN*.

"Forward Chaining merupakan strategi pencarian yang memulai proses pencarian dari sekumpulan data atau fakta, dari data-data tersebut dicari suatu kesimpulan yang menjadi solusi dari permasalahan yang dihadapi. Mesin inferensi mencari kaidah-kaidah dalam basis pengetahuan yang premisnya sesuai dengan data-data tersebut, kemudian dari kaidah-kaidah tersebut diperoleh suatu kesimpulan. *Forward Chaining* memulai proses pencarian dengan data sehingga strategi ini disebut juga data driven. Contoh proses *Forward Chaining* berbentuk kaidah produksi : *IF* kondisi 1 *AND* kondisi 2 *AND* kondisi 3 *THEN* kesimpulan" [9].



Gambar 2. 2 *Forward Chaining* [9].

1.3.4.1 Mekanisme Metode *Forward Chaining*

Pada dasarnya, metode *Forward Chaining* ini menggunakan fakta yang diperoleh dari *user* dengan cara mencocokannya dengan hipotesa untuk mencari suatu konklusi. Dalam hal ini, konklusi yang dimaksud adalah diagnosis jenis penyakit yang kemungkinan dialami *user*. Perlu

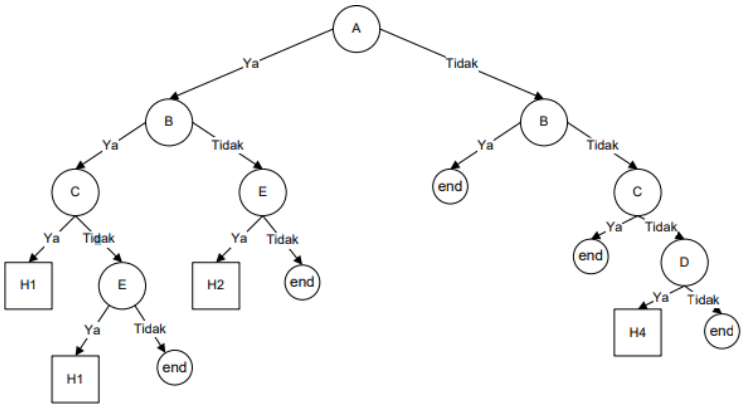
adanya suatu *knowledge base* yang berisi aturan atau *rule* untuk mengatur hubungan antara premis-premis dan konklusi.

1.3.4.2 Representasi Pengetahuan

Dalam penyelesaian memudahkan menerjemahkan pengetahuan (*knowledge base*) ke sistem, perlu adanya representasi pengetahuan. Menurut [9] ada beberapa jenis representasi pengetahuan, yaitu :

1. Pohon Keputusan

Pohon keputusan atau yang biasa disebut (*Decision Tree)* merupakan bentuk representasi pengetahuan untuk memodelkan persoalan yang terdiri dari serangkaian keputusan yang mengarah ke solusi. Pohon keputusan dapat dilihat pada gambar 2.3 di bawah ini.



Gambar 2. 3 Pohon Keputusan [9].

2. Probabilitas Keakuratan *Forward Chaining*

Untuk menghitung nilai probabilitas keakuratan *Forward Chaining* adalah dengan membandingkannya diagnosis metode dengan diagnosis pakar secara langsung. Berikut, rumus proporsi system pakar menurut [21] (2.1):

$$p = \frac{n(A)}{n(S)}\ x\ 100\% \qquad (2.\ 1)$$

Keterangan :

p : proporsi

n (A) : banyaknya gejala yang terdeteksi pada penyakit

A n (S) : banyaknya gejala yang dimiliki penyakit A

Menurut [22] dalam penelitian jurnalnya, probabilitas nilai keakuratan *Forward Chaining* studi kasus penyakit autisme pada anak dapat dihitung dengan data tabel 2.4 berikut :

Tabel 2. 4 Keakuratan *Forward Chaining* [22].

Kasus	Diagnosis Pakar	Diagnosis Sistem	Nilai Keakuratan
1.	Hiperaktif (ADHD)	Hiperaktif (ADHD)	1
2.	Tuna Grahita	Default	0
3.	Autis infatil	Autis infatil	1
4.	Gangguan Kelainan Syaraf otak sebelah kiri	Default	0
5.	Autis infatil	Autis infatil	1
6.	Sindrom Asperger	Hiperaktif (ADHD)	0
7.	Autis infatil	Autis infatil	1
8.	Hiperaktif (ADHD)	Hiperaktif (ADHD)	1
9.	Sindrom Asperger	Sindrom Asperger	1
10.	Autis infatil	Autis infatil	1
11.	Sindrom Asperger	Sindrom Asperger	1

Dari studi kasus data di atas, dapat dihitung suatu nilai probabilitas keakuratan metode *Forward Chaining* ini (2.2), yaitu :

$$P11(akurat) = \frac{8}{11} x100\% = 72{,}73\% \quad (2.2)$$

$$P11(tidak\ akurat) = \frac{3}{11} x100\% = 27{,}27\%$$

1.4 *Tools* Pembangun Sistem Pakar

1.4.1 *XAMPP*

XAMPP adalah perangkat lunak yang penamaannya diambil dari akronim kata *Apache, MySQL / MariaDB, PHP*, dan *Perl*. Imbuhan huruf "X" berasal dari istilah *cross platform* sebagai simbol bahwa aplikasi ini bisa dijalankan di empat sistem operasi berbeda, seperti *OS Linux*, *OS Windows*, *Mac OS*, dan juga *Solaris*.

Keuntungan penggunaan *XAMPP* adalah pengganti peran *web hosting* dimana file-file pembangun *web* disimpan didalam *hosting* lokal (*localhost*) agar bisa ditampilkan di *browser*. Bagian-bagian penting di dalam *XAMPP* yaitu :

1. *Htdocs* adalah folder atau direktori untuk menyimpan file-file pembangun *web* seperti *HTML, PHP*, dan file lainnya.
2. *PhpMyAdmin* adalah suatu *server* di komputer untuk manajemen basisdata *MySQL*.
3. *Control Panel* adalah panel untuk mengontrol *start* atau *stop* layanan *service XAMPP*.

1.4.2 *HTML*

Dalam membangun suatu halaman *web* dibutuhkan bahasa *markup* untuk menampilkan berbagai informasi di dalamnya. Bahasa *markup* yang digunakan adalah *HTML* karena kemudahan dalam penggunaannya.

"*HTML* (*Hypertext Markup Language*) adalah bahasa dasar untuk *web scripting* bersifat *client side* yang memungkinkan untuk menampilkan informasi dalam bentuk teks, grafik, serta multimedia dan juga untuk menghubungkan antartampilan *web page* (*hyperlink*)" [23].

1.4.3 *PHP*

Pembangunan *web* tidak hanya menggunakan bahasa *markup HTML* yang bekerja secara client di sisi *server*, tetapi juga butuh disisipkannya suatu bahasa pemorgraman script *PHP* secara *server-side* untuk membantu fungsi kerja bahasa *HTML*. *PHP* adalah singkatan dari *Hypertext Prepocessor*. Kelebihan mendasar dari *PHP* antara lain mendapatkan data dari *form*, menghasilkan tampilan

dinamis, *cookies* dan kemampuan integrasinya dalam mendukung banyak *database*.

PHP adalah *script* yang membuat dokumen *HTML* secara *on the fly*, maksudnya dokumen *HTML* yang dihasilkan dari suatu aplikasi bukan dokumen *HTML* yang dibuat dengan menggunakan *editor* teks atau *editor HTML* [23].

1.4.4 *MySQL*

Komponen penting lainnya dalam pembangungan sistem pakar adalah Basisdata. Secara sederhana basisdata atau *database* merupakan suatu gudang data elektronis. Didalamnya kita dapat menyimpan, mengakses, mengelola, dan mengupdate data. Menurut [24] *MySQL* (*My Structured SE Language*) adalah: " Suatu sistem basisdata *relation* atau RDBMS (*Relational Database Managemnt System*) yang mampu bekerja secara cepat dan mudah digunakan *MySQL* juga merupakan program pengakses *database* yang bersifat jaringan, sehingga sapat digunakan untuk aplikasi *multiuser*. *MySQL* didistribusikan gratis di bawah lisensi GPL (*General Public License*).

Didalam *MySQL* terdapat suatu bahasa perintah atau *query* yaitu SQL yang merupakan singkatan dari *Structured Query Language*. Terdapat 3 klausa struktur dasar dari ekspresi *SQL*:

a. *Select* digunakan untuk mendaftar semua atribut yang diinginkan sebagai hasil suatu *query.*
b. *From* ini mencatat semua relasi yang di"*scan*" dalam evaluasi suatu *query*.
c. *Where* terdiri dari sebuah predikat yang menyangkut atribut dari relasi yang muncul dalam klausa *from.*

Ekspresi dasar modifikasi data ada 3, yaitu :

a. *Delete* untuk menghapus data.
b. *Insert* untuk memasukan data, dan
c. *Update* untuk mengganti data.

1.4.5 *Visual Studio Code*

Visual Studio Code (*VS Code*) ini adalah sebuah teks *editor* ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi *Linux, Mac*, dan *Windows*. Teks *editor* ini secara langsung mendukung bahasa pemrograman *JavaScript, Typescript,* dan *Node.js*, serta bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang *via marketplace Visual Studio Code* seperti *C++, C#, Python, Go,* dan *Java.*

Dengan banyaknya fitur yang ditawarkan di *Visual Studio Code* memberikan kemudahan dalam mengedit *code.* Keuntungan lainnya adalah performanya yang handal hanya memakai memori yang sedikit, hal ini memandakan tidak perlu spesifikasi yang tinggi untuk dapat mengoperasikan *software Visual Studio Code* ini.

1.4.6 *StarUML*

Dalam perancangan sistem pakar ini, diperlukan suatu metode atau langkah-langkah sistematis dari awal hingga akhir. Dimana metode yang dipakai dapat mendokumentasikan, menspesifikasikan langkah demi langkah pembangunan sistem pakar ini. Penulis menggunakan suatu *software* untuk menyelesaikan metode tersebut menggunakan *StarUML*.

Menurut [25] *Unified Modeling Language* (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membanngun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.

Ada beberapa komponen *diagram* yang dipakai penulis di dalam *starUML* dalam perancangan sistem pakar ini adalah sebagai berikut:

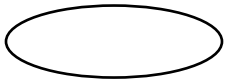
1. ***Use Case Diagram***

   Sebuah *Use Case Diagram* menyatakan visualisasi interaksi yang terjadi antara pengguna (aktor) dengan sistem. *Diagram* ini bisa menjadi

gambaran yang bagus untuk menjelaskan konteks dari sebuah sistem sehingga terlihat jelas batasan dari sistem [26]

Di dalam *use case diagram* terdapat beberapa simbol-simbol menurut [26] yaitu :

Tabel 2. 5 Simbol *Use Case Diagram*

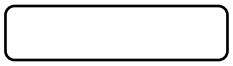
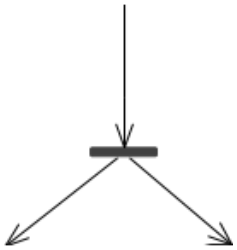
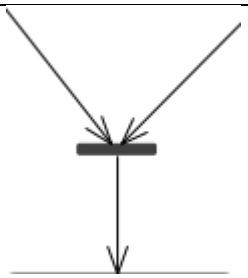
Simbol	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target 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 berinteraksi

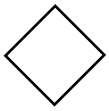
	dengan *Use Case*, tetapi tidak memiliki kontrol terhadap *use case*
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasika n siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasika n data.
	Asosiasi antara aktor, dan *use case* yang 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

2. *Activity Diagram*

Menurut [26] *Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* yaitu:

Tabel 2. 6 Tabel 1.Simbol *Activity Diagram*

Simbol	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities*, menggambar kan suatu proses/kegiat an bisnis
	*Fork*/percaban gan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi

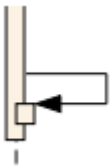
	*Decision Points*, menggambar kan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity diagram* untuk menunjukkan siapa melakukan apa

3. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *Sequence Diagram* [26] yaitu:

Tabel 2. 7 Simbol *Sequence Diagram*

Simbol	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface*s atau interaksi antara

	satu atau lebih aktor dengan sistem, seperti tampilan form entry 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
	Message, simbol mengirim pesan antar *class*
	*Recursive*, menggambarka n pengiriman pesan yang dikirim untuk dirinya sendiri
	*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*

4. *Class Diagram*

*Class Diagram m*erupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality* [26] yang dapat dilihat pada tabel berikut.

Tabel 2. 8 Multiplicity *Class Diagram*

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 maksimal 4

1.4.7 *Confusion Matrix*

Confusion matrix menurut [27] dapat diartikan sebagai suatu alat yang memiliki fungsi untuk melakukan analisis apakah *class*ifier tersebut baik dalam mengenali tuple dari kelas yang berbeda. Nilai dari *True Positive* dan *True-Negative* memberikan informasi ketika *classifier* dalam melakukan klasifikasi data bernilai benar, sedangkan *False Positive* dan *False-Negative* memberikan informasi ketika *classifier* salah dalam melakukan klasifikasi data. Tabel *confusion matrix* dapat dilihat pada tabel 2.9 berikut.

Tabel 2. 9 Confusion *Matrix* menampilkan total *positive* dan *negative tuple* [27].

		Predicted *class*		
		yes	no	Total
Actual *class*	yes	TP	FN	P
	no	FP	TN	N
	Total	P’	N’	P+N

1.4.8 *Black-box Testing*

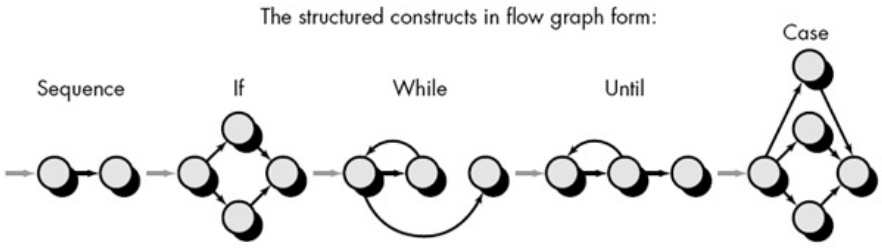
Setelah dilakukan pengukuran akurasi, selanjutnya dibutuhkan sebuah aktifitas untuk mengevaluasi kebenaran yaitu berupa pengujian kepada sistem yaitu *black-box testing*. Menurut [28] *black-box testing* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

1.4.9 *White-box Testing*

Pengujian perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan [28]. Beberapa tahapan *whitebox testing* antara lain :

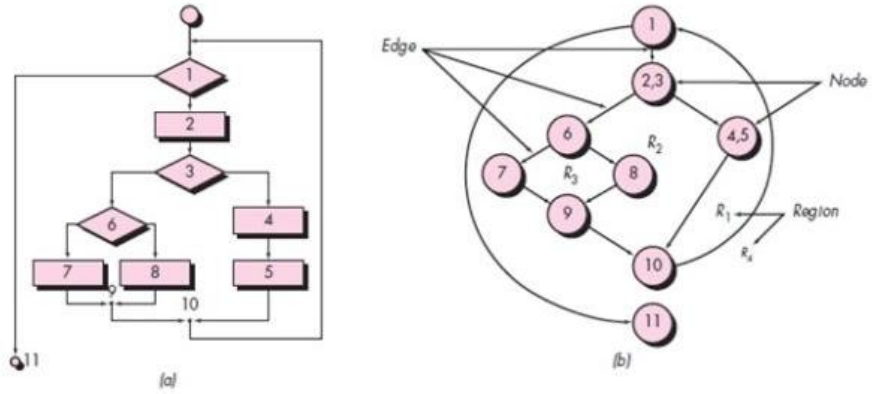
1. *Control Flow Testing*

   Notasi sederhana untuk merepresentasikan *control flow*. Notasi struktur kontrol pada *flowgraph* dapat dilihat pada gambar 2.4 di bawah ini.



Gambar 2. 4 Notasi *flowgraph* [31].

Sebelum mengilustrasikan pemakaian notasi *flowgraph* perlu rancangan prosedural *flowchart* kemudian memetakannya ke dalam bentuk *flowgraph*. Ilustrasi konversi *flowchart* ke *flowgraph* dapat dilihat pada gambar 2.5 di bawah ini.



Gambar 2. 5 Konversi *flowchart* ke *flowgraph* [31].

Pada gambar 2.5 di atas terdapat beberapa notasi. Notasi lingkaran yang mewakili *statement* khusus atau tidak disebut *node.* Notasi garis panah yang menggambarak aliran kontrol disebut *edge* atau *link.* Area yang dibatasi *edge* dan *node* disebut *region.*

2. Basis *Path Testing*

Basis *path* memungkinkan perancang *test case* untuk membuat pengukuran kompleksitas logikal dari rancangan prosedural dan menggunakan pengukuran ini sebagai panduan untuk mendefinisikan himpunan basis dari jalur eksekusi [30]. *Cyclomatic complexity* adalah *metric* piranti lunak yang menyediakan ukuran kuantitatif dari kompleksitas logikal program [32]. Perhitungan *Cylcomatic Complexity* dapat dibuat menggunakan rumus $V(G)$ (2.3) di bawah ini.

$$V(G) = E - N + 2 \; atau \; V(G) = P + 1 \qquad (2.3)$$

Keterangan :

$V(G)$ = *Cyclomatic complexity*

E = Jumlah *edge*

N = Jumlah *node*

P = Jumlah *predicate note*