BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Penelitian Terdahulu ini menjadi salah satu acuan peneliti dalam melakukan penelitian sehingga peneliti dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Dari penelitian terdahulu , peneliti tidak menemukan penelitian dengan judul yang sama seperti judul penelitian peneliti, Namun peneliti mengangkat beberapa penelitian sebagai referensi dalam memperkaya bahan kajian pada penelitian peneliti.

Penelitian mengenai **Penerapan Metode Dempster Shafer Untuk Mendiagnosa Penyakit Ikan Koi** yang telah dilakuan oleh Wirhan Fahrozi, Safrizal, Siti Aisyah (2018) menjelaskan bahwa selain menyediakan informasi tentang penyakit ikan koi sistem tersebut juga menyediakan diagnosa penyakit penyakit ikan koi. Metode penelusuran yang digunakan oleh Wirhan Fahrozi, Safrizal, Siti Aisyah adalah sama-sama menggunakan metode *Dempster Shafer*. Sedangkan *platform* yang digunakan pada sisem tersebut adalah berbasis *web*. Jadi perbedaan dan peningkatan jika dibandingkan dengan penelitian sebelumnya adalah jenis kasus yang diambil, pada penelitian ini penulis mengambil kasus gangguaan *Self Injury*.

Penelitian lain mengenai sistem pakar adalah **Sistem Pakar Untuk Mendiagnosa Penyakit Jantung Menggunakan Metode *Dempster Shafer.*** Sistem pakar ini dikembangkan untuk meningkatkan hasil dari penelitian sebelumnya yang sudah pernah dilakukan. Adapun penelitian sebelumnya yang telah dilakukan oleh Arjon Samuel Sitio (2018) mengenai aplikasi sistem pakar berbasis *dekstop* untuk diagnosa Penyakit Jantung Perbedaan dan peningkatan yang dilakukan dari penelitian sebelumnya adalah *platform* pengembangan sistem. Pada penelitian yang dilakukan oleh Arjon Samuel Sitio platform yang digunakan adalah hanya berbasis *desktop* dan menggunakan bahasa program *VB.Net*, sedangkan pada penelitian ini platform yang digunakan adalah berbasis *web* dan kasus yang diambil berbeda penelitian sebelumnya mengangkat kasus Penyakit Jantung sedang kan penelitian ini mengangkat kasus *Self Injury*, sedangkan metode nya sama yaitu sama-sama menggunakan metode *Dempster Shafer*.

Penelitian lain mengenai sistem pakar adalah **Penerapan Metode Dempster Shafer Untuk Mendiagnosa Penyakit Dari Akibat Bakteri Salmonella.** Sistem pakar ini dikembangkan untuk meningkatkan hasil dari penelitian sebelumnya yang sudah pernah dilakukan. Adapun penelitian sebelumnya yang telah dilakukan oleh Mikha Dayan Sinaga (2016) mengenai aplikasi sistem pakar berbasis dekstop untuk diagnosa Penyakit Dari Akibat Bakteri Salmonella Perbedaan dan peningkatan yang dilakukan dari penelitian sebelumnya adalah *platform* pengembangan sistem. Pada penelitian yang dilakukan oleh Mikha Dayan Sinaga *platform* yang digunakan adalah hanya berbasis *desktop* dan menggunakan bahasa program *VB.Net*, sedangkan pada

penelitian ini *platform* yang digunakan adalah berbasis web dan kasus yang diambil berbeda penelitian sebelumnya mengangkat kasus Penyakit Dari Akibat Bakteri Salmonella sedangkan penelitian ini mengangkat kasus Gangguan *Self Injury*, sedangkan metode nya sama yaitu sama-sama menggunakan metode *Dempster Shafer*.

Penelitian lain mengenai sistem pakar adalah **Sistem Diagnosa Penyakit THT Pada Balita Menggunakan Dempster Shafer.** Sistem pakar ini dikembangkan untuk meningkatkan hasil dari penelitian sebelumnya yang sudah pernah dilakukan. Adapun penelitian sebelumnya yang telah dilakukan oleh Maura Widyaningsih (2018) mengenai aplikasi sistem pakar berbasis *dekstop* untuk diagnosa Penyakit THT Pada Balita Perbedaan dan peningkatan yang dilakukan dari penelitian sebelumnya adalah *platform* pengembangan sistem. Pada penelitian yang dilakukan oleh Maura Widyaningsih *platform* yang digunakan adalah hanya berbasis *desktop* dan menggunakan bahasa program *VB.Net*, sedangkan pada penelitian ini platform yang digunakan adalah berbasis web dan kasus yang diambil berbeda penelitian sebelumnya mengangkat kasus Penyakit THT Pada Balita sedangkan penelitian ini mengangkat kasus Gangguan *Self Injury*, sedangkan metode nya sama yaitu sama-sama menggunakan metode *Dempster Shafer*.

Pada peneliti pertama yang berjudul **Sistem Pakar Diagnosa Gangguan Kepribadian Menggunakan Metode *Dempster Shafer.*** Sistem pakar ini dikembangkan untuk meningkatkan hasil dari penelitian sebelumnya yang sudah pernah dilakukan. Adapun penelitian sebelumnya yang telah dilakukan oleh Dodi

Teguh Yuwono (2019) mengenai aplikasi sistem pakar berbasis *web* untuk diagnosa Gangguan Kepribadian. Kesamaan nya adalah sama – sama penderitanya termasuk gangguan kejiwaan hanya beda gejala yang dialami. Perbedaan dan peningkatan yang dilakukan dari penelitian sebelumnya adalah *platform* pengembangan sistem. Pada penelitian yang dilakukan oleh Dodi Teguh Yuwono *platform* yang digunakan adalah hanya berbasis dekstop, sedangkan pada penelitian ini platform yang digunakan adalah berbasis *online*, sehingga mudah digunakan, dan cara penangan gejala awal yang dialami pasien dapat dilakukan dengan menggunakan obat tradional.

II.2. Landasan Teoritis

II.2.1. Sistem Pakar

Bidang sistem pakar merupakan penyelesaian pendekatan yang sangat berhasil dan bagus untuk permasalahan AI klasik dari pemprograman *intelligent* (cerdas). Sistem Pakar *(expert system)* merupakan solusi AI *(Artificial Intelligence)* bagi permasalahan pemrograman pintar. Profesor Edward Feigenbaum dari Stanford University yang merupakan pionir dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar (*Intelligent Computer Program*) yang memanfaatkan pengetahuan (*knowledge)* dan prosedur inferensi (*inference procedure*) untuk memecahkan masalah yang cukup sulit sehingga membutuhkan keahlian khusus dari manusia. (Rika Rosnelly; 2015: 2).

Secara umum, sistem pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan

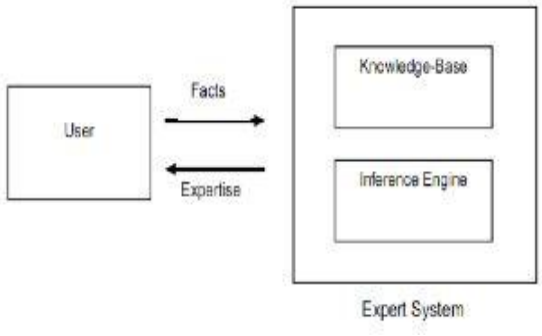
masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awam pun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman.

Untuk pem bangun sistem yang seperti itu maka komponen-komponen dasar yang minimal harus dimiliki adalah sebagai berikut:

1. Antar muka (user interface).
2. Basis pengetahuan (knowledge base).
3. Mesin inferensi (Inference Engine). (Triara Puspitasari, Boko Susilo, Funny Farady Coastera ; 2016).

II.2.1.1. Komponen dasar Sistem Pakar

Konsep dasar sistem pakar berbasis pengetahuan (*knowledge based expert system)*. User memberikan informasi atau fakta kepada sistem dan menerima respon berupa saran ahli (*advice/expertice*). Secara internal, sistem terdiri dari dua komponen utama yaitu basis pengetahuan (*knowladge based*) berisi pengetahuan yang akan digunakan oleh komponen lainnya yaitu mesin inferensi (*inference engine*) untuk menghasilkan kesimpulan sebagai respon terhadap kueri yang dilakukan user

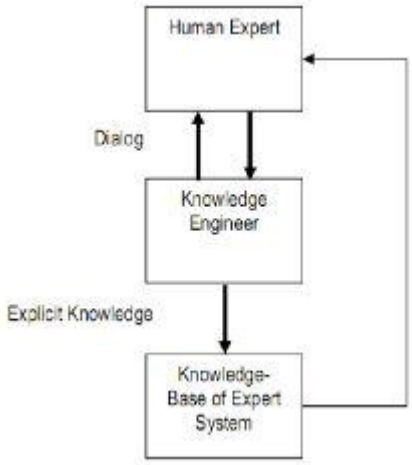


Gambar II.1 : Konsep Dasar Fungsi Sistem Pakar Berbasis Pengetahuan

(Sumber : Rika Rosnelly; 2015: 4)

II.2.1.2. Komponen umum Sistem Pakar

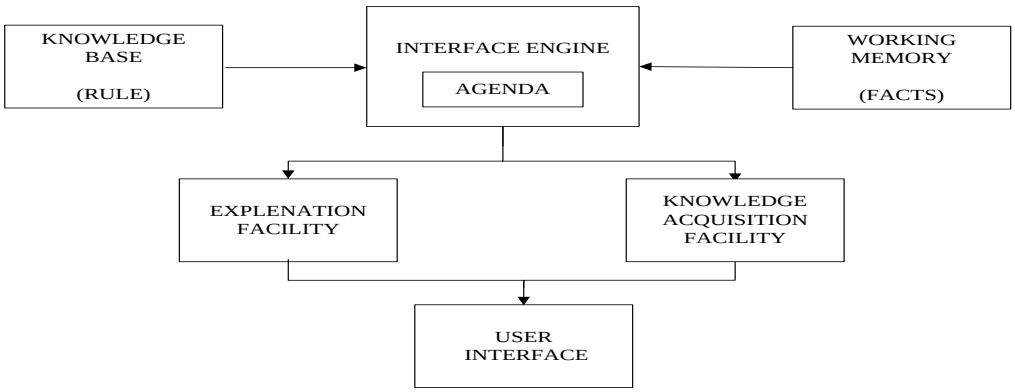
Pengetahuan yang dimiliki sistem pakar di representasikan dalam beberapa cara. Salah satu metode yang paling umum digunakan adalah tipe rules menggunakan format *IF THEN*. Banyak sistem pakar yang dibangun dengan mengespresikan pengetahuan dalam bentuk rules. Bahkan, pendekatan berbasis pengetahuan (*knowledge based approach*) untuk membangun sistem pakar telah mematahkan pendekatan awal yang digunakan pada sekitar tahun 1950-an dan 1960-an yang menggunakan tehnik penalaran(*Reasoning*) yang tidak mengandalkan pengetahuan dari pakar maupun sumber lain dan kodingnya disebut sebagai knowledge engineer. Tahapan pengembangan sistem pakar secara umum tergambar pada gambar II.2 dibawah ini.



Gambar II.2 : Pengembangan Sistem Pakar
(Sumber : Rika Rosnelly; 2015: 7)

II.2.1.3. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada Gambar II.I.



Gambar II.3 : Struktur Sistem Pakar
(Sumber : Rika Rosnelly; 2015: 13)

II.2.1.4. Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu.

Seorang pakar memiliki kemampuan kepakaran, yaitu :

1. Dapat mengenali dan merumuskan suatu masalah.
2. Menyelesaikan masalah dengan cepat dan tepat.
3. Menjelaskan solusi dari suatu masalah.
4. Restrukturisasi pengetahuan.
5. Belajar dari pengalaman.
6. Memahami batas kemampuan. (Rika Rosnelly; 2015: 10).

II.2.1.5. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak manfaat dari sistem pakar, diantaranya :

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti 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. Andal, sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.

8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem computer lain membuat 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 tidak lengkap. (Andi Yulia Muniar, Ashari; 2015: 3).

II.2.1.6. Kekurangan Sistem Pakar

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

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

II.2.2. *Self Injury*

Self Injury adalah perilaku yang mengacu pada penderitaan yang disengaja dan tidak dapat diterima secara sosial, perilaku mencederai diri dilakukan dengan melukai tubuh tanpa niat bunuh diri. Salah satu penyebab individu melakukan mencedera diri adalah pengalaman ketika mereka masih muda, mendapatkan pelecehan seksual di mana ada hubungan antara pengalaman usia dini dan mencederai diri. (Ikromilah Yety Prastuti, Budi Purwoko, Retno Tri Hariastuti, 2019 : 2).

Secara umum Pelaku menyakiti diri sendiri (*self injury*) merupakan upaya untuk mengurangi masalah emosional. Bagi para pelaku lebih baik merasakan sakit

fisik dari pada merasakan sakit psikis atau sakit secara emosionalnya. Pelaku *self Injury* melakukan tindakan menyakiti diri sendiri secara sengaja dengan alasan untuk mengurangi ketegangan, agar merasa lebih tenang dari perasaan yang tidak nyaman akibat dari penolakan yang dirasakan. (Laila faried, IGAA Noviekayati, Sahat Saragih 2018 : 1).

II.2.2.1. Gejala Utama *Self Injury*

1. Memukul diri sendiri
2. menggigit diri sendiri
3. merendahkan diri sendiri
4. Melakukan pembakaran ringan pada anggota tubuh
5. Menggaruk sampai terluka.
6. Menggosok area yang berlebihan untuk membuat luka bakar
7. Memegang benda tajam di tangan
8. Mengenakan baju lengan panjang atau celana panjang, bahkan dalam cuaca panas
9. Biasanya alasan luka karena kecelakaan yang tidak disengaja
10. Kesulitan dalam hubungan interpersonal
11. Perilaku dan ketidakstabilan emosi, impulsivitas dan tidak dapat diprediksi
12. Pernyataan tidak berdaya, putus asa atau tidak berharga (Ikromilah Yety Prastuti, Budi Purwoko, Retno Tri Hariastuti, 2019 : 2).

II.2.3 Metode *Dempster Shafer*

Metode *Dempster-Shafer* pertama kali diperkenalkan oleh *Dempster*, yang melakukan percobaan model ketidakpastian dengan *range probabilities* dari pada sebagai probabilitas tunggal. Kemudian pada tahun 1976 *Shafer* mempublikasikan teori *Dempster* itu pada sebuah buku yang berjudul *Mathematical Theory Of Evident*. *Dempster-Shafer Theory Of Evidence*, menunjukkan suatu cara untuk memberikan bobot kenyakinan sesuai fakta yang dikumpulkan. Pada teori ini dapat membedakan ketidakpastian dan ketidaktahuan. Teori *Dempster-Shafer* adalah representasi, kombinasi dan propogasi ketidakpastian, dimana teori ini memiliki beberapa karakteristik yang secara instutitif sesuai dengan cara berfikir seorang pakar, namun dasar matematika yang kuat.

Teori *Dempster Shafer* merupakan teori yang membahas tentang penanganan dalam berbagai suatu kemungkinan yang dapat dikombinasikan pada suatu fakta tertentu. Dari DST atau *Dempster Shafer Theory* tersebut mempunyai permasalahan yaitu konflik yang dapat dipersatukan dengan berbagai informasi yang ada.kumpulan informasi yang saling berbeda dapat diberikan symbol sebagai q atau *theta*[16].

Sebuah teori yang menjelaskan tentang teori matematika untuk melakukan pembuktian yang didasari oleh fungsi kepercayaan dan juga logika yang masuk akal, yang digunakan untuk menyatukan beberapa bukti untuk menghasilkan kemungkinan dari suatu peritiwa. Pengembang dari teori ini adalah Arthur P. Dempster dan Glenn Shafer. Pada umumnya teori *Dempster-Shafer*

Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval: [*Belief*,*Plausibility*]. *Belief* (Bel) adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian. *Plausibility* (Pls) akan mengurangi tingkat kepastian dari *evidence. Plausibility* bernilai 0 sampai 1. Jika yakin akan X', maka dapat dikatakan bahwa Bel(X') = 1, sehingga rumus di atas nilai dari Pls(X) = 0. Menurut Giarratano dan Riley fungsi *Belief* dapat diformulasikan dan ditunjukkan pada persamaan (1):

$$Bel\ (X) = \sum_{Y \subseteq X} m(Y) \qquad (1)$$

Dan *Plausibility* dinotasikan pada persamaan (2):

$$Pls\ (X) = 1 - Bel\ (X) = 1 - \sum_{Y \subseteq X} m(X) \qquad (2)$$

dimana :

Bel (X) = *Belief* (X)

Pls (X) = *Plausibility* (X)

m (X) = *mass function* dari (X)

m (Y) = *mass function* dari (Y)

Teori *Dempster-Shafer* menyatakan adanya *frame of discrement* yang dinotasikan dengan simbol (Θ). *frame of discrement* merupakan semesta pembicaraan dari sekumpulan hipotesis sehingga sering disebut dengan *environment* yang ditunjukkan pada persamaan:

Θ = { θ1, θ2, … θN}

Dimana :

Θ = *frame of discrement atau environment*

θ1,…,θN = element/ unsur bagian dalam *environment*

Environment mengandung elemen-elemen yang menggambarkan kemungkinan sebagai jawaban, dan hanya ada satu yang akan sesuai dengan jawaban yang dibutuhkan. Kemungkinan ini dalam teori *Dempster-Shafer* disebut dengan *power set* dan dinotasikan dengan P (Θ), setiap elemen dalam *power set* ini memiliki

nilai interval antara 0 sampai 1.

m : P (Θ) [0,1] □

Sehingga dapat dirumuskan pada persamaan:

Dengan :

P (Θ) = *power set*

m (X) = *mass function* (X)

Mass function (m) dalam teori *Dempster-shafer* adalah tingkat kepercayaan dari suatu *evidence* (gejala), sering disebut dengan *evidence measure* sehingga dinotasikan dengan (m). Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua *evidence* secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2*n*. Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai :

m{θ} = 1,0

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densi tasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitas nya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu ditunjukkan pada persamaan:

$$m3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X).\, m_2(Y)}{1 - \sum_{X \cap Y = \emptyset} m1(X).\, m2(Y)} \quad (5)$$

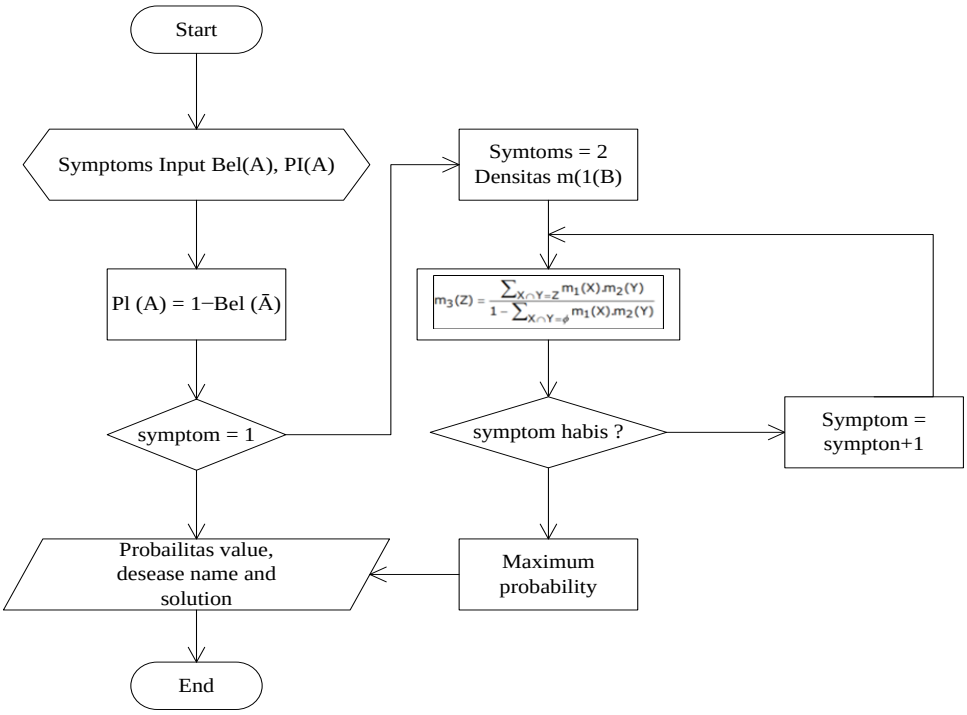
dimana :

m3(Z) = *mass function* dari *evidence* (Z)

m1(X) = *mass function* dari *evidence* (X), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

m2(Y) = *mass function* dari *evidence* (Y), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut. (Mikha Dayan Sinaga, Nita Sari Br. Sembiring, 2016: 2)

Berikut adalah alur metode *DempsterShafer*:



Gambar II.2. Alur Metode *DempsterShafer*
(Sumber : Maura Widyaningsih, 2018 : 5)

II.2.4. *PHP*

PHP adalah bahasa pemrograman script server-side yang didesain untuk pengembangan web. Selain itu, *PHP* juga bias digunakan sebagai bahasa pemrograman pemrograman umum. *PHP* di kembangkan pada tahun 1995 oleh *Rasmus Lerdorf*, dan sekarang dikelola oleh The *PHP* Group. Situs situs resmi *PHP* beralamat di *http://www.php.net*. *PHP* disebut bahasa bahasa pemrograman server side karena *PHP* diproses pada komputer server. Hal ini berbeda dibandingkan dengan bahasa pemrograman client-side seperti JavaScript yang diproses pada web browser (client). (Hendra Agusvianto, 2017 : 2).

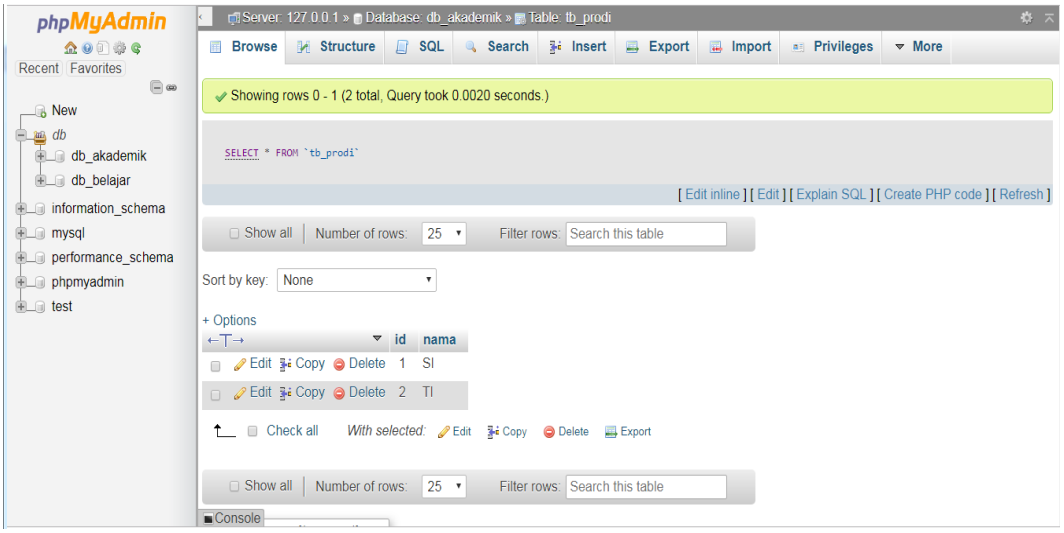
Pada saat pertama kali dijalankan*Sublime Text,* akan menampilkan sebuah jendela *Sublime Text*pada gambar :

Gambar II.3. *Sublime Text*

II.2.5. *MySQL*

MySQL adalah sebuah implementasi dari sistem sistemmanajemen yang pada basis basis data yang relasional *(RDBMS)* yangdidistribusikan secara gratis. Setiap setiap pengguna dapat secara bebas menggunakan *MySQL,* namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam basis basis dan data data yang telah ada sebelumnya; *SQL (Structured Query Language)*. *SQL* adalah sebuah inti konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data data, yang memungkinkan pada pengoperasian data dikerjakan dengan mudah secara otomatis. (Hendra Agusvianto, 2017 : 2).

Pada saat pertama kali dijalankan *MySQL*, akan menampilkan sebuah jendela *MySQL* pada gambar :



Gambar II.4. Tampilan Awal *MySQL*

II.3. Basis Data (*Database*)

Database adalah sekumpulan tabel-tabel yang saling berelasi, relasi tersebut bisa ditunjukkan dengan kunci dari tiap tabel yang ada.Satu *database* menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instansi.*Database* mempunyai kegunaan dalam mengatasi penyusunan dan penyimpanan data, maka seringkali masalah yang dihadapi adalah:

1. Redundansi dan Inkonsistensi data
2. Kesulitan dalam pengaksesan data
3. Isolasi data untuk standarisasi
4. *Multi user*
5. Keamanan data

6. Integritas data

7. Kebebasan data (Urva dan Siregar, 2015 :95).

II.4. *UML* (*Unified Modelling Language*)

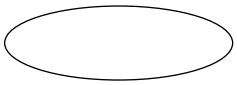
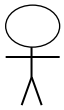
Unified Modeling Language (*UML*) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.*UML* merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

Dalam membangun perancangan sistem dengan alat bantu perancangan *Unified Modeling Language* (*UML*) ada beberapa tahapan yang akan dilakukan, yaitu sebagai berikut :

1. *Use case* Diagram

Use Case Diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Adapun simbol *Use Case Diagram* yang dapat di lihat pada Tabel II.2. :

Tabel II.2. Simbol *Use Case diagram*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	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 control 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 mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber :Urva ; 2015)

Beberapa fungsi dan kegunaan dari *UML* yaitu (Mulyani, 2016 : 244) :

1. *Visualizing,* yaitu sebagai alat komunikasi konseptual model antara tim pengembang sistem (sistem analis dengan programmer)
2. *Specifying,* yaitu sebagai *tools* yang digunakan untuk memodelkan sistem secara tepat dan jelas.

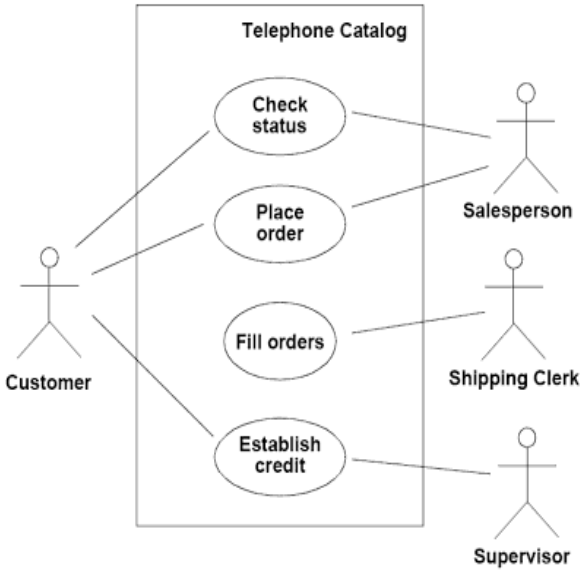
3. *Constructing,* yaitu *UML* sebagai bahasa gratis mampu melakukan *mapping* dan konseptual model kedalam bahasa pemrograman.
4. *Documenting,* yaitu *UML* digunakan sebagai *tools* untuk melakkukan dokumentasi teknis sebuah sistem.

Diagram-diagram yang terdapat dalam *UML* sangat banyak, berikut ini beberapa diagram yang sering digunakan dalam pengembangan sistem yaitu :

II.1. Use Case Model

Use case model merupakan kumpulan diagram dan text yang saling bekeria sama untuk mendokumentasikan bagaimana *user* (aktor) berinteraksi dengan sistem. *Use case model* terdiri dari beberapa diagram :

1) *Use case diagram* yaitu diagram yang menggambarkan dan merepresentasikan aktor, *use cases* dan *dependencies* suatu proyek dimana tujuan dan diagram ini adalah untuk menjelaskan konsep hubungan antara sistem dengan dunia luar. *Use case diagram* dapat dilihat pada gambar II.2 seperti berikut :



Gambar II.5 Use Case Diagram

(Sumber : Mulyani ; 2016)

II.2. Class Diagram

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

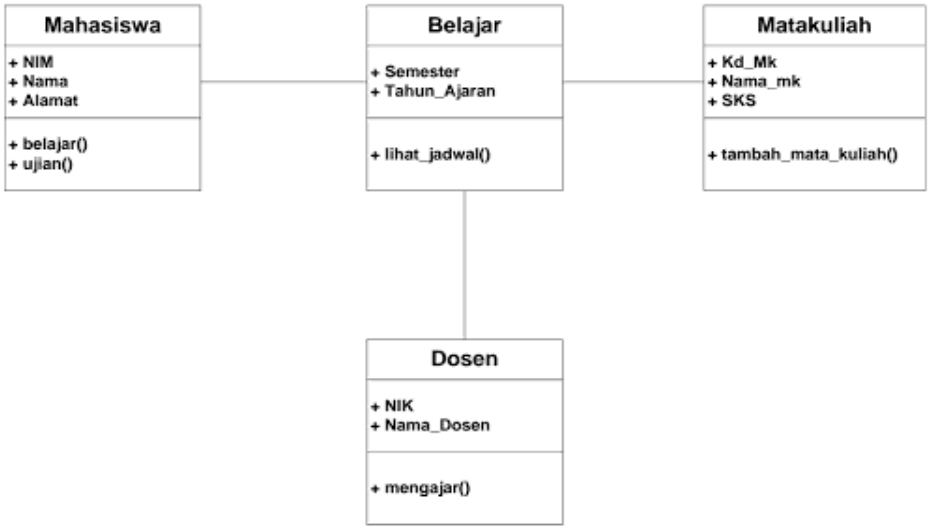
Class Diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility,* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti. Adapun simbol-simbol *Multipcity class diagram* yang dapat di lihat pada Tabel II.3. :

Tabel II.3. ***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 maksimum 4

(Sumber : Urva ; 2015)

Class diagram adalah diagram yang digunakan untuk merepresentasikan kelas, komponen-komponen kelas dan hubungan anatara masing-masing kelas. Selain itu class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. *UML* menggunakan istilah fitur sebagai istilah umum yang meliputi property dan operasi sebuah kelas. *Class diagram* dapat dilihat pada gambar II.5 seperti berikut :



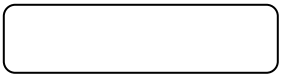
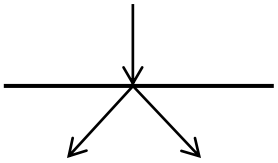
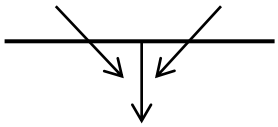
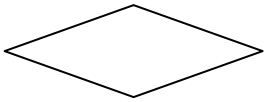
Gambar II.6 Class Diagram

(Sumber : Mulyani ; 2016)

II.3. Activity Diagram

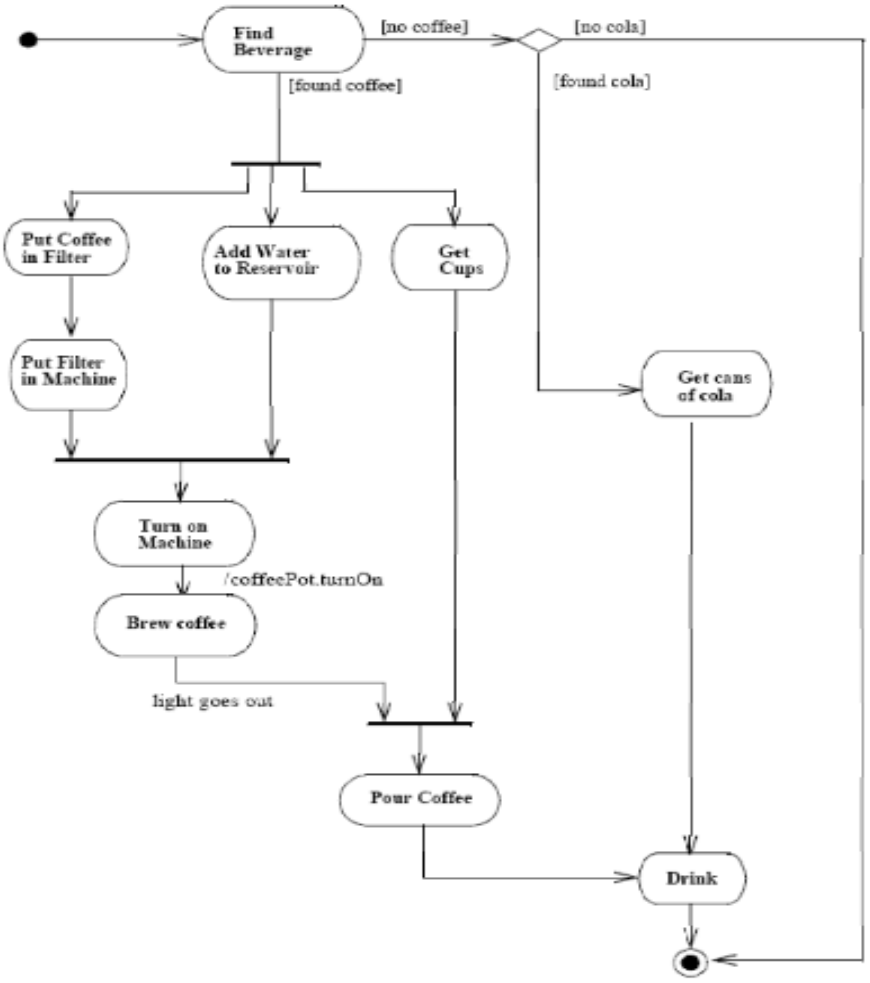
Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Adapun simbol-simbol *activity diagram* yang dapat di lihat pada Tabel II.4.

Tabel II.4. Simbol *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 parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Urva; 2015)

Jika sebelumnya Anda sudah pernah menggunakan *flowchart*, maka Anda akan mudah dalam memahami dan mempelajari *activity diagram*, karena *diagram* ini sangat mirip sekali dengan *fiowchart*, perbedaanya adalah *activity diagram* memiliki kemampuan untuk melakukan percabangan aktifitas, selain itu *activity* diagram juga memungkinkan pemisahan aktifitas antar aktor. *Activity diagram* adalah diagram *UML* yang digunakan untuk menggambarkan alur aktifitas dari satu proses. *Activity diagram* memungkinkan siapapun yang melakukan proses untuk memilih urutan dalam melakukannya, dengan kata lain diagram hanya menyebutkan aturan-aturan rangkaian dasar yang harus kita ikuti. Hal ini penting untuk pemodelan bisnis karena proses-proses sering muncul secara paralel. Ini juga berguna pada algoritma yang bersamaan, dimana urutan-urutan independen dapat melakukan hal-hal secara paralel. *Acitvity diagram* dapat dilihat pada gambar II.6 seperti berikut :



Gambar II.7 Activity Diagram

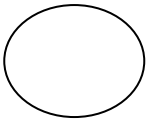
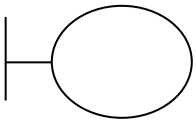
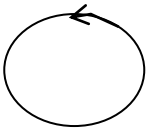
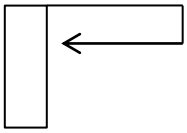
(Sumber : Mulyani ; 2016)

II.4.Sequence Diagram

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Adapun simbol *Sequence Diagram* yang dapat di lihat pada Tabel II.5. :

(Urva, 2015 : 2 – 4).

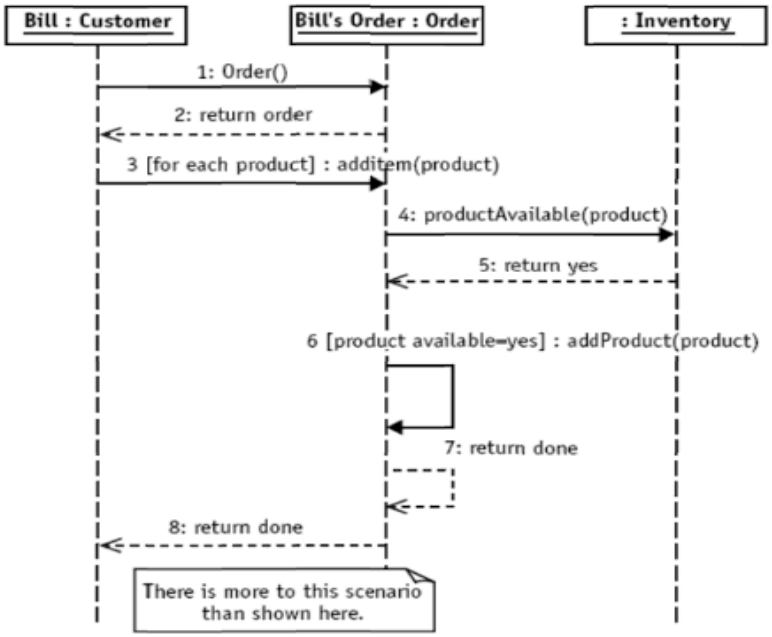
Tabel II.5. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass,* 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* 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.
	*Message,* simbol mengirim pesan antar *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 aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Urva; 2015)

Sequence diagram adalah diagram yang menggambarkan interaksi antar objek. *Sequence diagram* secara khusus menjabarkan *behavior* sebuah skenario

tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek ini dalam sebuah *use case*. *Sequence diagram* dapat dilihat pada gambar II.7 seperti berikut :



Gambar II.8 Sequence Diagram

(Sumber : Mulyani ; 2016)

II.5. Normalisasi

Normalisasi merupakan peralatan yang digunakan untuk melakukan proses pengelompokkan data menjadikan tabel-tabel yang menunjukkan entitas dan relasinya. Secara umum proses normalisasi dibagi menjadi tiga tahap, yaitu tahap tidak normal, normalisasi tahap 1, normalisasi tahap 2, normalisasi tahap 3. Pada tahap ketiga biasanya sudah akan diperoleh tabel yang optimal.

1. Bentuk Tidak Normal (*Unnormalized Form*) Pada tahap ini, semua data yang ada direkam tanpa format tertentu. Data bisa jadi mengalami duplikasi.
2. Bentuk Normal Tahap 1 (*1st Unnormalized Form*) Pada tahap ini, dibentuk tabel-tabel yang menampung data yang ada dikelompokkan berdasarkan suatu karakteristik tertentu.
3. Bentuk Normal Tahap 2 (*2st Unnormalized Form*) Pada tahap ini, dilakukan penentuan *field* kunci dari masing-masing tabel. Kunci tersebut harus unik dan mewakili tabel. Bentuk normal tahap 2 (2NF) terpenuhi jika pada sebuah tabel, semua atribut selain *primary key* memiliki ketergantungan fungsional pada *primary key* yang utuh. (Lukman ; 2012 : 219-220)

II.6. Kamus Data

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data sistem analis dapat mendefinisikan data yang mengalir pada sistem dengan lengkap. Kamus data dibuat berdasarkan arus data yang dibuat pada *data flow diagram*. Arus yang ada di DFD (*Data Flow Diagram*) bersifat global dan hanya menunjukkan nama arus datanya saja. Kamus data atau *data dictionary* harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. (Drs. Hermansyah Sembiring, M.Kom, dkk ; 2013 :