BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar

Artificial intelligence (AI) memiliki beberapa domain masalah atau area. Bidang sistem pakar merupakan penyelesaian pendekatan yang sangat berhasil dan bagus untuk permasalahan AI klasik dari pemrograman *intelligent* (cerdas). (Rosnelly, 2012 : 2).

II.1.1. Pengertian Sistem Pakar

Sistem pakar (*expert system*) merupakan solusi AI bagi masalah pemrograman pintar (*intelligent*). 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 hingga membutuhkan keahlian khusus dari manusia.

Dengan kata lain, sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rosnelly, 2012 : 2).

II.1.2. Kelebihan Sistem Pakar

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya (Rosnelly, 2013 : 5-6), seperti :

1. Meningkatkan ketersediaan (*increased availibility*). Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal (*massproduction*).
2. Mengurangi biaya (*reduce cost*). Biaya yang diperlukan untuk menyediakan keahlian per satu orang *user* menjadi berkurang.
3. Mengurangi bahaya (*reduce danger*). Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
4. Permanen (*permanence*). Sistem pakar dan pengetahuan yang terdapat di dalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan dan pengetahuannya hilang saat pakar meninggal dunia.
5. Keahlian multiple (*multiple expertise*). Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat keahlian atau pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan satu orang pakar.
6. Meningkatkan kehandalan (*increased reliability*). Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternatif pendapat dari seorang pakar atau sebagai penengah jika terjadi konflik antara beberapa pakar. Namun hal tersebut tidak berlaku jika sistem dibuat oleh salah seorang

pakar, sehingga akan selalu sama dengan pendapat pakar tersebut kecuali jika sang pakar melakukan kesalahan yang mungkin terjadi pada saat tertekan.

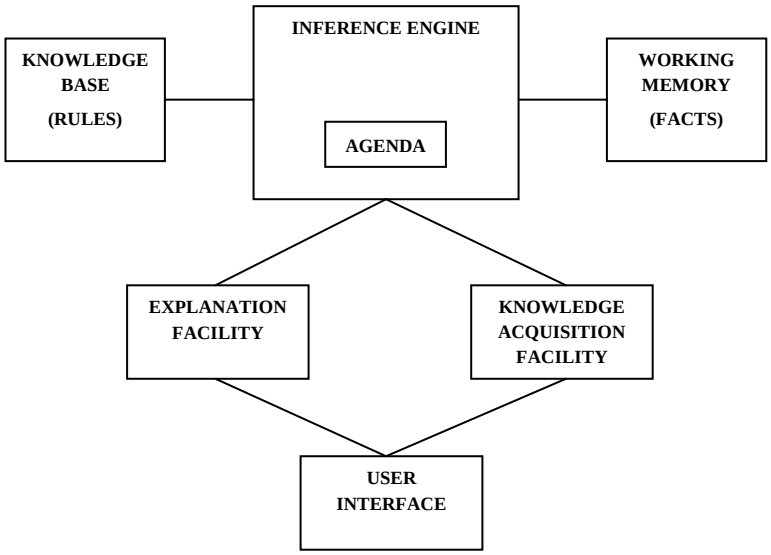
7. Penjelasan (*explanation*). Sistem pakar dapat menjelaskan detail proses penalaran (*reasoning*) yang dilakukan hingga mencapai suatu kesimpulan. Seorang pakar mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melakukannya setiap waktu. Hal ini akan meningkatkan tingkat percayaan bahwa kesimpulan yang dihasilkan adalah benar.
8. Respon yang cepat (*fast response*). Respon yang cepat atau *real time* diperlukan pada beberapa aplikasi. Meskipun bergantung pada *hardware* dan *software* yang digunakan, namun sistem pakar relatif memberikan respon yang lebih cepat dibandingkan seorang pakar.
9. Stabil, tidak emosional dan memberikan respon yang lengkap setiap saat (*steady, unemotional and complete response at all times*). Karakteristik ini diperlukan pada situasi *real time* dan keadaan darurat (*emergency*) ketika seorang pakar mungkin tidak berada pada kondisi puncak disebabkan oleh stres atau kelelahan.
10. Pembimbing pintar (*intelligent tutor*). Sistem pakar dapat berperan sebagai *intelligent tutor* dengan memberikan kesempatan pada *user* untuk menjalankan contoh program dan menjelaskan proses *reasoning* yang dilakukan.
11. Basis data cerdas (*intelligent database*). Sistem pakar dapat digunakan untuk mengakses basis data secara cerdas.

II.1.3. Konsep Umum Sistem Pakar

Sistem pakar umumnya dirancang dengan cara yang berbeda dengan sistem konvensional lain, terutama karena masalah yang dihadapi umumnya tidak memiliki solusi algoritmik dan bergantung pada inferensi untuk mendapatkan solusi yang terbaik yang paling mungkin. Oleh karena itu sistem pakar harus mampu menjelaskan inferensi yang dilakukannya sehingga hasil yang diperoleh dapat diperiksa. *Explanation facility* merupakan bagian terintegrasi dari sebuah sistem pakar. Sebuah *explanation facility* yang detail dirancang untuk memungkinkan *user* mengeksplorasi *rules* melalui tipe pertanyaan "*what if*" yang disebut *hypothetical reasoning* dan bahkan menerjemahkan *natural language* ke dalam *rules* dengan cara induksi dari tabel data. (Rosnelly, 2013 : 8).

II.1.4. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada gambar II.1. berikut ini :



Gambar II.1. Struktur Sistem Pakar
(Sumber : Rika Rosnelly, 2013 : 13)

Komponen yang terdapat dalam struktur sistem pakar ini adalah *knowledge base* (*rules*), *inference engine*, *working memory*, *explanation facility*, *knowledge acquisition facility*, *user interface*. (Rosnelly, 2013 : 13-15).

1. *Knowledge Base* (Basis Pengetahuan)

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

2. *Inference Engine* (Mesin Inferensi)

   Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* (struktur kontrol) atau *rule interpreter* (dalam sistem pakar berbasis kaidah). Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *processor* pada sistem pakar yang mencocokkan bagian kondisi dari *rule* yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

3. *Working Memory*

   Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau dapat juga dikatakan sebagai global database dari fakta yang digunakan oleh rule-rule yang ada.

4. *Explanation Facility*

   Menyediakan kebenaran dari solusi yang dihasilkan kepada user (*reasoning chain*).

5. *Knowledge Acquisition Facility*

   Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi pemecahan ke program komputer, yang bertujuan untuk memperbaiki atau mengembanglan basis pengetahuan.

6. *User Interface*

   Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu, antar muka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

II.1.5. Karakteristik Sistem Pakar

Menurut Rosnelly (2013 : 20-21), sistem pakar umumnya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

a. **Kinerja sangat baik** (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.

b. **Waktu respon yang baik** (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat

dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata (*real time*).

c. **Dapat diandalkan** (*good reliabilty*). Sistem harus dapat diandalkan dan tidak mudah rusak/*crash*.

d. **Dapat dipahami** (*understandable*). Sistem harus mampu menjelaskan langkah-langkah penalaran yang dilakukannya seperti seorang pakar. Hal ini penting untuk beberapa alasan, yaitu :

   1) Dimungkinkan bahwa sistem pakar berkaitan dengan nyawa manusia atau properti lainnya sehingga harus dapat menjelaskan mengapa dihasilkan suatu kesimpulan tertentu.
   2) Untuk mengkonfirmasi bahwa pengetahuan pakar telah dikumpulkan dengan benar dan digunakan oleh sistem dengan benar pula. Hal ini penting dalam proses debugging pengetahuan yang mungkin salah karena pengetikan atau pemahaman yang salah dari *knowledge engineer*.

e. **Fleksibel** (*flexibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah dan menghapus pengetahuan.

II.2. Tanaman Tomat (*Lycopersicon esculentum* Mill. L.)

Buah tomat (*Lycopersicum esculentum.* Mill. L.) merupakan tanaman sayur yang banyak dikonsumsi oleh masyarakat di seluruh dunia, terutama di Indonesia. Tomat (*Lycopersicon esculentum* Mill. L.) tergolong salah satu komoditas hortikultura bernilai ekonomi tinggi dan masih memerlukan

penanganan serius, terutama dalam hal peningkatan hasilnya dan kualitas buahnya (Catur Wasonowati : 2011).

Kualitas dan hasil produksi tanaman tomat (*Lycopersicon esculentum* Mill. L.) bergantung dari bagaimana petani untuk mengatasi penyakit yang menyerang karena akan berpengaruh terhadap hasil yang diproduksi. Tidak sedikit dari petani yang melakukan kesalahan saat mengatasi permasalahan tersebut. Bahkan, seringkali petani mengalami hambatan dalam melakukan penanganan dengan cepat dan tepat (Indra Dewa Pratama, Muhammad Ilyas : 2016).

II.2.1. Taksonomi dan Morfologi Tanaman Tomat

1. Taksonomi tanaman tomat

Kedudukan tanaman tomat (*Lycopersicon esculentum* Mill. L.) dalam sistematika (tata nama) tumbuhan, diklasifikasikan sebagai berikut :

Divisio : Spermatophyta.

Sub divisio : Angiosperme.

Klas : Dicotyledonae (berkeping dua).

Sub klas : Metachlamidae.

Ordo : Tubiflorae.

Famili : Solanaceae (berbunga seperti terompet).

Genus : Solanum (Lycopersicum).

Spesies : *Lycopersicum esculentum* Mill.

Tanaman tomat (*Lycopersicon esculentum* Mill. L.) masih satu keluarga dengan kentang (*Solanum tubersoum* L.), terung (*S. melongena* L.), ranti atau

leunca (*S. nigrum* L.), takokak (*S. torvum* sp.). Spesies lain dari kerabat dekat tanaman tomat cukup banyak, diantaranya adalah cherry (*L. cerasiforme*), tomat ranti atau ranggeum (*L. pimpinellifolium*), dan juga *L. peruvianum*, *L. glandulosum*, *L. hirsutum* dan *L. chilense* yang umumnya disebut tomat liar. (Rukmana, 1994 : 19).

2. Morfologi Tanaman Tomat

Tomat merupakan tanaman setahun (*annual*) atau tahunan (*perennial*) yang berumur pendek, tetapi umumnya tumbuh setahun berbentuk perdu. Tinggi tanaman dapat mencapai 2 – 3 meter atau lebih, mempunyai batang lunak dan bulat. Batang tanaman sewaktu masih muda mudah patah, sedangkan setelah tua menjadi keras hampir berkayu dan seluruh permukaan batangnya berbulu halus serta bercabang lebat. Spesifikasi morfologi tanaman tomat (*Lycopersicon esculentum* Mill. L.) yang dapat dilihat secara detail pada tabel II.1. berikut ini :

Tabel II.1. Spesifikasi Morfologi Tanaman Tomat

Spesifikasi	Ciri - ciri
Daun	Daun tomat umumnya lebar-lebar, bersirip dan berbulu, panjangnya antara 20 – 30 cm atau lebih, lebar sekitar 15 – 20 cm dan biasanya tumbuh dekat ujung dahan (cabang). Tangkai daun bulat panjang sekitar 7 – 10 cm dan tebalnya antara 0,3 – 0,5 cm.
Bunga	Bunga tanaman tomat tersusun dalam rangkaian bunga yang jumlah kuntum bunganya beragam antar varietas. Kuntum bunga tomat terdiri atas 5 daun kelopak dan 5 helai mahkota, memiliki bakal buah, kepala putik, tangkai putik serta benang sari.
Buah	Buah tomat umumnya berbentuk bulat atau bulat pipih, oval dengan ukuran panjang 4 – 7 cm, diameter 3 – 8 cm. Struktur buah tomat berada di atas tangkai buah, kulitnya tipis, halus dan bila sudah masak berwarna merah muda, merah dan juga kuning.
Akar	Tanaman tomat memiliki akar tunggang dan akar-akar cabang yang menyebar ke semua arah pada kedalaman hingga 60 – 70 cm.

(Sumber : Rukmana, 1994 : 20).

II.2.2. Penyakit pada Tanaman Tomat

Penyakit pada tanaman tomat (*Lycopersicon esculentum* Mill. L.) juga tergolong banyak dan beragam. Semuanya memberikan kerugian berupa penurunan kuantitas dan kualitas produk. Oleh karena itu, pengendalian penyakit patut diperhatikan. Kebersihan lingkungan penanaman salah satu upaya preventif mencegah penyebaran penyakit.

Umumnya penyakit pada tanaman tomat (*Lycopersicon esculentum* Mill. L.) dibagi menjadi empat kelompok, yaitu penyakit yang disebabkan oleh bakteri, penyakit yang disebabkan oleh jamur/fungi, penyakit yang disebabkan oleh nematoda dan penyakit yang disebabkan oleh virus. Selain itu, ada penyakit yang bersifat fisiologis yang menyebabkan *fruit disorder* (kerusakan pada buah). (Hidayati., Dermawan, 2012 : 75-76).

Menurut Hidayati dan Dermawan, (2012 : 76-86), beberapa penyakit yang sering menyerang pada penanaman tomat (*Lycopersicon esculentum* Mill. L.) dirangkum dalam tabel II.2 :

Tabel II.2. Penyakit pada Tanaman Tomat

No	Nama Penyakit	Nama Latin	Keterangan
Penyakit yang disebabkan oleh bakteri			
1	Layu Bakteri	*Pseudomonas solanacearum*	Organisme penyebabnya adalah bakteri *Pseudomonas solanacearum* E.F Smith.
2	Bercak Bakteri	*Xanthomonas campestris*	Organisme penyebabnya adalah bakteri *Xanthomonas campestris* pv *vesicatoria*.
Penyakit yang disebabkan oleh jamur/fungi			
3	Layu Fusarium	*Fusarium oxysporum*	Organisme penyebabnya adalah jamur *Fusarium oxysporum* f.sp. *lycopersici* (Sacc).

4	Busuk Daun	*Phytophtora infestans*	Organisme penyebabnya adalah jamur *Phytophthora infestans* Mont de Barry.
5	Bercak Kering	*Alternaria solani*	Organisme penyebabnya adalah cendawan *Alternaria solani* ELL & Martin.
Penyakit yang disebabkan oleh virus			
6	TMV	*Tomato Mosaic Virus*	Penularan virus terjadi melalui biji, serangga, manusiadan alat-alat pertanian.
7	CMV	*Cucumber Mosaic Virus*	Peranan vektor kutu daun (*Aphid* sp.) seperti *Aphis gossypii* dan *Myzus persicae*.
Penyakit fisiologis			
8	Busuk Ujung Buah	*Blossom-end rot*	Disebabkan kekurangan unsur hara Calsium (Ca).

(Sumber : Hidayati., Dermawan, 2012 : 75-86).

II.2.3. Budidaya Tanaman Tomat

1. Syarat-syarat tumbuh

Dalam budidaya tanamat tomat (*Lycopersicon esculentum* Mill. L.) ada beberapa syarat-syarat yang harus dipenuhi agar tanaman tomat (*Lycopersicon esculentum* Mill. L.) dapat tumbuh dengan baik dan dapat menghasilkan buah berkualitas tinggi. Menurut Rukmana (1994 : 33-34), adapun syarat-syarat tumbuh tersebut adalah sebagai berikut :

a. Iklim

Tanaman tomat (*Lycopersicon esculentum* Mill. L.) cocok dengan iklim kering dengan memerlukan sinar matahari minimal 8 jam per hari dan curah hujan pada kisaran 750 – 1.250 mm per tahun.

b. Tanah

Keadaan tanah harus subur, gembur dan banyak mengandung zat-zat organis (humus), sirkulasi udara dan tata air dalam tanah baik, serta memiliki pH tanah antara 5 – 7.

c. Daerah

Tanah cocok dengan daerah pegunungan, bisa hidup juga di dataran rendah, tetapi hasilnya kurang. Tomat bisa tumbuh dengan baik dan optimal pada ketinggian 1000-1250 m dari permukaan laut (dpl), sedangkan suhu udara yang cocok dengan temperatur siang hari $\pm 24^{o}$C dan malam hari berkisar antara 15-20^{o}C.

2. Kultur teknik budidaya

Dalam usaha tani atau agribisnis, budidaya merupakan salah satu faktor penentu keberhasilannya. Sedikit saja kesalahan dalam teknik budidaya akan berakibat fatal. Kesalahan tersebut dapat mengakibatkan berkurangnya hasil yang diperoleh, baik dari segi kuantitas maupun kualitas, yang merupakan karakter unggulan serta nilai keistimewaan genetik dari suatu tanaman yang diusahakan. Menurut Hidayati dan Dermawan (2012 : 26-48;56-63), beberapa hal yang perlu dipersiapkan dan dilakukan dalam budidaya tanaman tomat (*Lycopersicon esculentum* Mill. L.) diantaranya :

a. Pemilihan benih dan penyemaian

Hampir semua varietas yang digunakan untuk usaha budidaya tanaman tomat (*Lycopersicon esculentum* Mill. L.) adalah varietas hibrida. Hal ini terjadi karena keunggulan yang terdapat pada varietas hibrida sangat jelas

dibandingkan dengan kultivar lokal, baik dari segi produktivitas maupun ketahanan terhadap penyakit. Setelah dilakukan pemilihan benih, maka selanjutnya adalah melakukan penyemaian. Umumnya ada dua cara penyemaian tomat yang dilakukan oleh petani sebagai berikut.

1) Penyemaian dalam bedeng semai (*seedbed*) dengan cara ditebar merata dalam bedengan dan dilakukan pindah bibit ke bumbungan setelah 7 hari.
2) Penyemaian langsung dalam bumbungan yang terbuat dari daun pisang, plastik polibag ukuran diameter 3 cm dengan tinggi 6 cm atau dalam *seeding tray*.

b. Pembibitan

Pada fase awal, yang harus diperhatikan adalah kondisi media semai yang tidak boleh terlalu lembap atau terlalu kering. Media yang terlalu basah akan menyebabkan benih busuk, sedangkan media yang terlalu kering akan menyebabkan gagalnya perkecambahan. Hal ini berarti bahwa penyiraman harus dikontrol atau dipantau secara ketat. Pembukaan persemaian harus dilakukan segera setelah benih berkecambah agar tidak terjadi etiolasi. Agar diperoleh bibit yang kokoh, segera setelah dan benar pertama berkembang, bibit dirangsang dengan digoyang-goyang secara manual atau menggunakan alat setiap pagi dan sore hari sampai bibit siap dipindahkan ke lapang.

c. Persiapan lahan penanaman

Persiapan lahan di kebun dilakukan dengan cara pengolahan tanah. Tahap-tahap pengolahan tanah dilakukan dengan cara sebagai berikut :

1) Bersihkan lahan terlebih dahulu dari sisa-sisa tanaman atau perakaran dari pertanaman sebelumnya. Selain itu, batu-batu sisa bangunan, kaleng, plastik dan sampah lain harus disingkirkan dari areal penanaman.
2) Bajak atau cangkul tanah sedalam 30-40 cm, lalu bentuk bedengan selebar 110-120 cm, tinggi 40-50 cm dan lebar parit 60-70 cm. Adapun panjang bedengan tergantung dari luasan lahan yang ada dan kemampuan tenaga kerja untuk memeliharanya.
3) Buat parit di sekeliling lahan kebun tomat dengan lebar 70 cm dan kedalaman 70 cm.
4) Setelah bedengan terbentu, pupuk bedengan dengan pupuk kandang yang telah matang sebanyak 1,0-1,5 kg/tanaman. Pupuk dasar diberikan masing-masing SP-36 dan KCI dengan dosis 15-20 g per tanaman, baik dengan cara ditabur atau pada lubang tanam.
5) Pada tanah yang pH-nya rendah, lakukan pengapuran sebanyak 100-125 g/tanaman bersamaan dengan pemberian pupuk kandang.
6) Aduk pupuk kandang dan kapur pertanian dengan tanah bedengan secara merata sambil dibalik. Bedengan dibiarkan selama 1-2 minggu.

d. Teknik penanaman

Penanaman tomat (*Lycopersicon esculentum* Mill. L.) dilahan dapat dilakukan dengan dua teknik, yaitu tanpa menggunakan mulsa dan dengan menggunakan mulsa. Secara umum, kedua teknik penanaman ini relatif sama. Perbedaannya, teknik menggunakan mulsa cenderung lebih praktis dibandingkan dengan teknik tanpa menggunakan mulsa.

e. Pemeliharaan

Setelah penanaman, kegiatan selanjutnya adalah pemeliharan. Adapun kegiatan pemeliharan tanaman tomat umumnya meliputi sebagai berikut :

1) Pengairan (penyiraman)

Pada fase awal pertumbuhan atau saat tanaman tomat (*Lycopersicon esculentum* Mill. L.) masih menyesuaikan diri (adaptasi) terhadap lingkungannya, penyiraman perlu dilakukan secara rutin setiap hari. Penyiraman sebaiknya dilakukan pagi dan sore hari.

2) Pemasangan ajir (turus)

Pertumbuhan tanaman tomat (*Lycopersicon esculentum* Mill. L.) sangat cepat, baik tipe *determinate* maupun *indeterminate*. Tanaman *determinate* memiliki cabang samping yang banyak sehingga tanaman perlu ditopang agar tidak rebah. Sebaliknya, untuk tanaman *indeterminate* memerlukan rambatan yang tinggi karena pertumbuhan ke atas sangat cepat. Oleh karena itu, perlu dilakukan pemasangan ajir untuk menopang tanaman.

3) Perempelan tunas di bawah V

Penentuan bagian V pada tanaman tomat (*Lycopersicon esculentum* Mill. L.), baik *determinate* maupun *indeterminate*, dilakukan dengan memilih dua cabang yang membentuk huruf V dan paling kokoh. Setelah bagian V ditetapkan maka semua tunas yang ada di bagian bawah V harus dibuang agar tanaman bisa memanjat atau tumbuh secara optimal.

4) Pemupukan tambahan (susulan)

Tanaman tomat (*Lycopersicon esculentum* Mill. L.) yang sudah dipupuk pada saat akan dipasang mulsa plastik perlu dberikan pupuk tambahan agar pertumbuhannya prima. Jenis pupuk yang digunakan pada fase pertumbuhan vegetatif aktif (daun dan tunas) adalah pupuk daun, pupuk NPK, pupuk urea dan pupuk KCI dengan dosis yang dianjurkan pada masing-masing pupuk.

II.3. Metode Teorema Bayes

Metode Bayes adalah pendekatan secara statistik untuk menghitung trade off di antara keputusan yang berbeda-beda, dengan menggunakan probabilitas dan nilai yang menyertai suatu pengambilan keputusan tersebut (Agustina, 2014). Metode Bayes digunakan untuk menghitung ketidakpastian data menjadi data yang pasti dengan menyertakan persentasenya. Teorema Bayes lebih banyak diterapkan pada hal-hal yang berkenaan dengan diagnosis secara statistik yang berhubungan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan. Adapun Formula Bayes dapat dinyatakan sebagai berikut. (Vonny Pawaka).

Probabilitas bayes merupakan salah satu cara untuk mengatasi ketidakpastian data dengan formula bayes yang dinyatakan dengan (Jusniwati : 2013) :

$$P(H|E) = \frac{P(E|H)*P(H)}{P(E)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Dimana :

P(H | E) = Probabilitas hipotesa H benar jika diberikan *evidence* E.

P(E | H) = Probabilitas munculnya *evidence* E, jika diketahui hipotesa H benar.

P(H) = Probabilitas hipotesa H (menurut hasil sebelumnya) tanpa memandang *evidence* apapun.

P(E) = Probabilitas *evidence* E.

Secara umum, teorema bayes dengan E kejadian dan hipotesis H dapat dituliskan dalam bentuk :

$$P(Hi|E) = \frac{P(E \cap H)}{\sum p(E \cap Hi)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$= \frac{P(E|Hi).\ P(Hi)}{\sum (P(E|Hi).\ P(Hi)}$$

$$= \frac{P(E|Hi)\ .\ P(Hi)}{P(E)}$$

atau

$$P(H_i|E) = \frac{P(E|H_i)\ .\ P(H_i)}{\sum_{K=1}^{n} P(E|H_k)\ .\ P(H_k)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

I.4. Model Perancangan

Model perancangan digunakan oleh penulis untuk memberikan gambaran mengenai konsep perancangan sistem yang akan diimplementasikan.

II.4.1. Normalisasi

Istilah normalisasi berasal dari E. F. Codd, salah seorang perintis teknologi basis data. Selain dipakai sebagai metodologi tersendiri untuk menciptakan struktur tabel (relasi) dalam basis data (dengan tujuan untuk mengurangi kemubaziran data), normalisasi terkadang hanya dipakai sebagai perangkat verifikasi terhadap tabel-tabel yang dihasilkan oleh metodologi lain. Normalisasi memberikan panduan yang sangat membantu bagi pengembang untuk mencegah penciptaan struktur tabel yang kurang fleksibel atau mengurangi ketidakefisienan (Kadir, 2003 : 65).

1. Anomaly

Anomali adalah proses pada basis data yang memberikan efek samping yang tidak diharapkan (misalnya menyebabkan ketidakkonsistenan data atau membuat sesuatu data menjadi hilang ketika data lain dihapus). Macam anomali ada tiga buah, yaitu sebagai berikut :

a. Anomali peremajaan

Anomali ini terjadi bila terjadi pengubahan pada sejumlah data yang mubazir, tetapi tidak seluruhnya diubah.

b. Anomali penghapusan

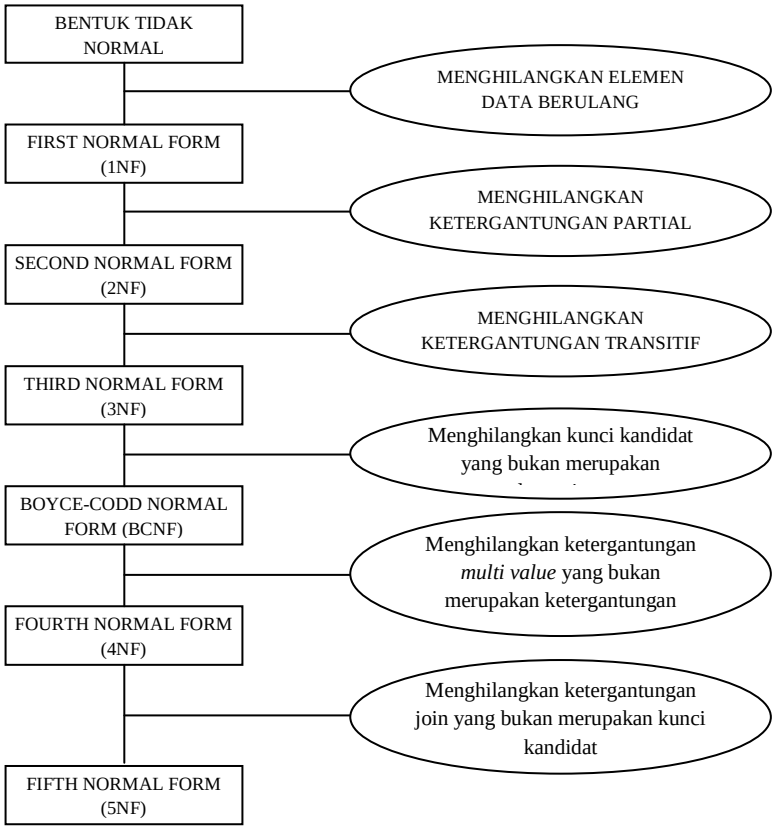
Anomali penyisipan terjadi jika pada saat penambahan hendak dilakukan ternyata ada elemen data yang masih kosong dan elemen data tersebut justru menjadi kunci.

c. Anomali penyisipan

Anomali penghapus terjadi sekirannya sesuatu baris (tupel) yang tak terpakai dihapus dan sebagai akibatnya terdapat data lain yang hilang. (Kadir, 2003 : 65-68).

2. Langkah-langkah pembentukan normalisasi

Langkah-langkah pembentukan normalisasi dapat dilihat pada gambar II.2. berikut ini beserta penjelasannya.



Gambar II.2. Langkah-langkah pembentukan normalisasi
(Sumber : Linda Marlinda, 2004 : 121)

Keterangan gambar di atas dapat dijelaskan sebagai berikut (Marlinda, 2004 : 122-125) :

a. Bentuk tidak normal (*Unnormalized Form*)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan saat menginput.

b. Bentuk normal kesatu (1 NF/*First Normal Form*)

Suatu relasi 1 NF jika dan hanya jika sifat dan setiap relasi atributnya bersifat atomik. Atom adalah zat terkecil yang masih memiliki sifat induknya, bila dipecah lagi maka ia tidak memiliki sifat induknya.

c. Bentuk normal kedua (2 NF/*Second Normal Form*)

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu. Atribut bukan kunci haruslah bergantung secara fungsi pada *primary key*. Jadi, untuk membentuk normal kedua haruslah sudah ditemukan kunci-kunci field. Kunci field haruslah unik dan dapat mewakili atribut lain yang menjadi anggotanya.

d. Bentuk normal ketiga (3 NF/*Third Normal Form*)

Untuk menjadi bentuk normal ketiga maka relasi haruslah dalam bentuk normal kedua dan semua atribut bukan primer tidak punya hubungan yang transitif. Dengan kata lain, setiap atribut bukan kunci haruslah bergantung hanya pada *primary key* dan pada *primary key* secara menyeluruh.

II.4.2. *Unified Modelling Language* (UML)

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek (Rosa A.S., M. Shalahuddin, 2014 : 137-138*)*

Berikut ini adalah beberapa diagram UML (Rosa A.S., M. Shalahuddin, 2014 : 146-167*)* :

1. *Use Case Diagram*

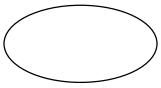
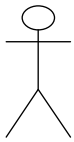
Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (behaviour) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

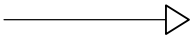
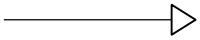
Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
- *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut ini adalah simbol-simbol yang ada pada *use case diagram* :

Tabel II.3. Simbol-simbol *Use Case Diagram*

Simbol	Keterangan
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor / *Actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor
Asosiasi / *Association*	Komuniakasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Generalisasi / *Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Ekstensi / *Extends* «extends» 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan
Menggunakan / *Include / Uses* «uses» 	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini

(Sumber : Rosa A.S., M. Shalahuddin, 2014 : 155-158)

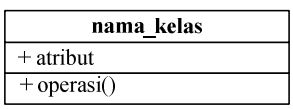
2. *Class Diagram*

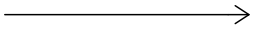
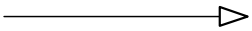
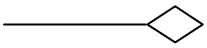
Diagram kelas atau *Class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Berikut ini adalah simbol-simbol yang ada pada *class diagram* :

Tabel II.4. Simbol-Simbol *Class Diagram* (Diagram Kelas)

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek

Asosiasi / *association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *directed asosiaciation* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi / *Generalization* 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / *dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas
Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua bagian (*whole-part*)

(Sumber : Rosa A.S., M. Shalahuddin, 2014 : 146-147)

3. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

- Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
- Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

- Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
- Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut ini adalah simbol-simbol yang ada pada *activity diagram* :

Tabel II.5. Simbol-simbol *Activity Diagram* (Diagram Aktivitas)

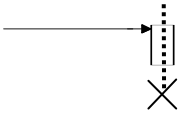
Simbol	Deskripsi
Status Awal 	Status awal aktivitas sistem sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan / *Decision* 	*Decision*, atau pilihan untuk mengambil keputusan.
Penggabungan / *Join*	Arah tanda panah alur proses.
Status Akhir 	Titik akhir atau akhir dari aktivitas.

(Sumber : Rosa A.S., M. Shalahuddin, 2014 : 162-163)

4. *Sequence Diagram*

Diagram sekuen (*Sequence Diagram*) menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Berikut ini adalah simbol-simbol yang ada pada *sequence diagram*:

Tabel II.6. Simbol-simbol *Sequence Diagram* (Diagram Sekuen)

Simbol	Deskripsi
Aktor / *Actor*  Atau nama aktor Tanpa waktu aktif	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek
Objek nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya
Pesan tipe create << create >>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call 1 : nama_metode() 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri
Pesan tipe send 1 : masukan	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang di akhiri, sebaiknya jika ada *create* maka ada *destroy*

(Sumber : Rosa A.S., M. Shalahuddin, 2014 : 165-167)

II.5. Bahasa Pemrograman dan *Software* yang Digunakan

Bahasa pemrograman dalam bentuk pengkodingan dan perangkat lunak atau *Software* yang digunakan untuk mendukung pembuatan aplikasi sistem pakar identifikasi penyakit pada tanaman tomat.

II.5.1. *PHP Hypertext Preprocessor* (PHP)

Pada tahun 1994 seorang programmer bernama Rasmus Lerdorf awalnya membuat sebuah halaman website pribadi, tujuannya adalah untuk mempertahankan halaman *website* pribadi tersebut sekaligus membangun halaman *web* yang dinamis. PHP pada awalnya diperkenalkan sebagai singkatan dari *Personal Home Page*. PHP pertama ditulis menggunakan bahasa Perl (*Perl Script*), kemudian ditulis ulang menggunakan bahasa pemrograman C CGI-BIN (*Common Gateway Interface-Binary*) yang ditujukan untuk mengembangkan halaman *website* yang mendukung formulir dan penyimpanan data. Pada tahun 1995 *PHP Tool 1.0* dirilis untuk umum, kemudian pengembangannya dilanjutkan oleh Andi Gutmans dan Zeev Suraski. Perusahaan bernama Zend kemudian melanjutkan pengembangan PHP dan merilis PHP versi terakhir pada saat ini. (Sibero, 2013 : 49).

A. Pengertian PHP

Menurut Sibero (2013 : 49), PHP adalah pemrograman *interpreter* yaitu proses penerjemahan baris kode sumber menjadi kode mesin yang dimengerti komputer secara langsung pada saat baris kode dijalankan. PHP disebut sebagai pemrograman *Server Side Programming*, hal ini dikarenakan seluruh prosesnya

dijalankan pada *server*. PHP adalah suatu bahasa dengan hak cipta terbuka atau yang juga dikenal dengan istilah *Open Source*, yaitu pengguna dapat mengembangkan kode-kode fungsi PHP sesuai dengan kebutuhannya.

B. Penulisan Kode PHP

Pemrograman PHP dapat ditulis dalam dua bentuk yaitu penulisan baris kode PHP pada *file* tunggal dan penulisan kode PHP pada halaman html (*embedded*). Kedua cara penulisan tersebut tidak memiliki perbedaan, hanya menjadi kebiasaan gaya penulisan dari programmer. (Sibero, 2013 : 49-50). Berikut ini adalah contoh penulisan kode program PHP :

Contoh 1 dari penulisan baris kode PHP pada *file* tunggal.

singlefile_php.php

```
<?php
        echo “<html>”;
        echo “<head>”;
        echo “<title>Contoh PHP</title>”;
        echo “</head>”;
        echo “<body>”;
        echo “<p>Di bawah ini adalah tulisan dari PHP</p>”;
        echo “Teks dari PHP”;
        echo “</html>”;         ?>
```

II.5.2. *Cascading Style Sheet* (CSS)

Cascading Style Sheet dikembangkan untuk menata gaya pengaturan halaman *web*. Pada awalnya CSS dikembangkan pada SGML pada tahun 1970 dan terus dikembangkan hingga saat ini CSS telah mendukung banyak bahasa *Markup* seperti : HTML, XHTML, XML, SVG (*Scalable Vector Graphics*) dan Mozilla XUL (*XML User interface Language*). Mengacu dari arti bahasa, *Cascading Style Sheet* memiliki arti Gaya Menata Halaman Bertingkat, yang

berarti setiap satu elemen yang telah diformat dan memiliki anak dan telah diformat, maka anak dari elemen tersebut secara otomatis mengikuti format elemen induknya. (Sibero, 2013 : 112-113).

Cascading Style Sheet terdiri dari *Selector*, Properti dan Nilai. Seperti halnya HTML, PHP dan bahasa pemrograman lainnya, CSS juga memiliki aturan penulisan. Berikut di bawah ini aturan penulisan CSS :

```
span // span => Selector
{
   font-weight: bold; // font-weight => properti, bold => nilai
   color: red; // color => properti, red => nilai
}
```

II.5.3. *MySQL*

MySQL atau dibaca "*My Sekuel*" dengan adalah suatu RDBMS (*Relational Database Management System*) yaitu aplikasi sistem yang menjalankan fungsi pengolahan data. *MySQL* pertama kali dikembangkan oleh *MySQL* AB yang kemudian diakuisisi Sun Microsystem dan terakhir dikelola oleh Oracle Coorporation. (Sibero, 2013 : 97).

Menurut Nugroho, *MySQL* adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multi user* serta menggunakan perintah standar *SQL* (*Structred Query Language*). *MySQL* memiliki dua bentuk lisensi, yaitu *FreeSoftware* dan *Shareware*. *MySQL* yang biasa kita gunakan adalah *MySQL FreeSoftware* yang berada di bawah Lisensi GNU/GPL (*General Public Licence*).

Jadi, berdasarkan definisi dari kedua ahli tersebut dapat disimpulkan bahwa *MySQL* adalah aplikasi *database server* yang menjalankan fungsi pengolahan data dengan menggunakan perintah standar *SQL*.

II.5.4. *Adobe Dreamweaver*

Adobe Dreamweaver adalah suatu produk *Web Developer* yang dikembangkan oleh *Adobe Systems Inc.*, sebelumnya produk *Dreamweaver* dikembangkan oleh *Macromedia Inc.* yang kemudian sampai dengan saat ini pengembangannya diteruskan oleh *Adobe Systems Inc*. Setelah diambil alih oleh *Adobe Systems Inc., Dreamweaver* dikembangkan dan dirilis dengan kode nama *Creative Suite (CS).*

Ruang Kerja atau *Workspace* adalah bagian keseluruhan tampilan dari *Adobe Dreamweaver*. Ruang kerja *Dreamweaver* terdiri dari *Welcome Screen, Menu, Insert bar, Document Window, CSS Panel, Aplication Panel, Tag Inspector, Property Inspector, Result Panel* dan *Files Panel*. Masing-masing dari komponen tersebut memiliki fungsi dan aturan. (Sibero, 2013 : 384).