**BAB II**

**LANDASAN TEORI**

1. **Burung Lovebird**

Lovebird merupakan burung yang memiliki suara merdu dan bulu yang indah. burung ini dikenal sebagai lambing cinta oleh karena itu ia dinamai *Lovebird* dengan arti *burung cinta.*penamaan itu bukan tanpa dasar, itu semua lantaran sepasang burung lovebird yang sudah berjodoh seperti sepasang kekasih. Keduanya berdekatan seolah tidak ingin berpisah jauh dari pasanganya (Handono, 2013).

Menurut redaksi Trubus tahun 2000, jenis burung lovebird ada 9 species. Delapan species berasal dari benua Afrika dan satu species berasal dari pulau madagaskar yang letaknya sebelah timur Afrika Selatan. Tetapi didalam pembahasan kali ini hanya diambil 4 jenis species burung lovebird yang berasal dari benua Afrika saja. Sebagai contoh adalah Agapornis Fischery, Agapornis Personatus, Agapornis Liliane Nyasa, Agapornis Nigrigenis.

Hal utama yang perlu diperhatikan dalam pakan burung lovebird adalah menu yang variatif sehingga nutrisinya kecukupan. Pakan yang bagus selain lengkap nutrisinya seperti protein, karbohidrat, juga lengkap vitaminnya (A, D3, E, B1, B2, B, B12, C, dan K3). Selain itu burung lovebird memerlukan pula nutrisit yang mengandung zat esensial. Sedangkan untuk mineral yang dibutuhkan oleh burung lovebird adalah Calcium, Zinccum, Magnesium, Sodium, Chiorin dan Kalium.

Untuk memulai beternak burung lovebird yang harus diperhatikan adalah memilih calon indukan dengan usia yang relatif masih muda berkisar umur 7 bulan. Burung lovebied sendiri mampu hidup lebih dari 10 hingga 15 tahun lamanya. Beternak burung lovebird sebaiknya dilakukan secara individu supaya kita lebih mudah mengetahui garis keturunan jenis species burung lovebird, karena di Indonesia saat ini galur murni species lovebird sangatlah sulit ditemui dikalangan peternak lokal.

1. **Sistem Pakar**

Sistem pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke computer, agar computer 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 (Kusumadewi, 2003:109).

Sistem pakar pertamakali dikembangkan oleh komunitas AI pada pertengahan tahun 1960. Sistem pakar yang muncul pertama kali adalah *General Purpose Problem Solver* (GPS) yang dikembangkan oleh Newel & Simon (Turban, 1995).

Sistem pakar (*expert system*) merupakan paket perangkat lunak atau paket program computer yang ditujuhkan sebagai penyedia nasihat dan sarana bantu dalam memecahkan masalah dibidang tertentu seperti Sains, Perekayasaan, Matematika, Kedokteran, Pendidikan, dan sebagainya. Sistem pakar merupakan subset dari *Artificial Intelegence* (Arhami, 2005)

1. **Naïve Bayes**
   * 1. **Pengertian Naïve Bayes**

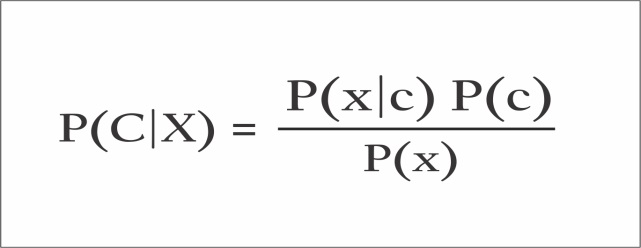
Algoritma Naïve Bayes merupakan sebuah metode klasifikasi menggunakan metode probabilitas dan statistic yang dikemukakan oleh ilmuwan Inggris *Thomas Bayes*. Algoritma Naïve Bayes memprediksi peluang dimasa depan berdasarkan pengalaman dimasa sebelumnyasehingga dikenal sebagai Teorema Bayes. Ciri utama dari Naïve Bayes Classifier ini adalah asumsi yang sangat kuat (naïf) akan independensi dari masing-masing kondisi/ kejadian.

Naïve Bayes Classifier bekerja sangat baik dibandingkan dengan model Classifier lainnya. Hal ini dibuktikan pada jurnal *Xhemali, Daniela, Chris J. Hinde, and Roger G. Stone*. “Naïve Bayes vs Decision Trees vs Neural network in the classification of training web pages.” (2009), mengatakan bahwa “*Naïve Bayes Classifier memiliki tingkat akurasi yang lebih baik disbanding model classifier lainnya*”.

* + 1. **Teorema Bayes (Dalil Naïve Bayes)**

Bayes merupakan teknik prediksi berbasis probabilitas sederhana yang berdasar pada penerapan Teorema Bayes (atau aturan bayes) dengan asumsi independensi (ketidak tergantungan) yang kuat (naïf). Dengan kata lain, naïve bayes, model yang digunakan adalah “model fitur independen”

Dalam Bayes, maksud independensi yang kuat pada fitur adalah bahwa fitur pada sebuah data tidak berkaitan dengan ada atau tidaknya fitur lain dalam data yang sama. Prediksi bayes didasarkan pada teorema bayes dengan formula umum sebagai berikut :



**Gambar 2.1 Formula Teorema Bayes**

Penjelasan dari gambar formula teorema bayes adalah sebagai berikut :

|  |  |
| --- | --- |
| Parameter | Keterangan |
| P ( C | X ) | Probabilitas hipotesis berdasarkan kondisi (hipotesis C terjadi jika diberikan bukti (evidence) X terjadi |
| P ( x | c ) | Probabilitas berdasarkan kondisi pada hipotesis (x terjadi akan mempengaruhi hipotesis c). |
| P ( c ) | Probabilitas bukti (prior probability), terjadi tanpa memandang bukti apapun. |
| P ( x ) | Probabilitas c, terjadi tanpa memandang hipotesis/bukti yang lain |
| x | Data dengan clas yang belum diketahui. |
| c | Bukti data (merupakan class spesifik). |

Ide dasar dari aturan Bayes adalah bahwa hasil dari hipotesis atau periswa **C** dapat diperkirakan berdasarkan pada beberapa bukti **X** yang diamati. Ada beberapa hal penting dari aturan bayes tersebut yaitu :

1. Sebuah probabilitas awal/prior **C** atau **P(C)** adalah probabilitas dari suatu bukti sebelum bukti diamati.
2. Sebuah probabilitas akhir **C** atau **P(C|X)** adalah probabilitas dari suatu bukti setelah bukti diamati.
   * 1. **Naïve Bayes Untuk Klasifikasi**

Kaitan antara Naïve Bayes dengan klasifikasi, kolerasi hipotesis dan bukti klasifikasi adalah bahwa hipotesis dalam teorema Bayes merupakan label kelas yang menjadi target pemetaan dalam klasifikasi, sedangkan bukti merupakan fitur-fitur yang menjadikan masukan dalam model klasifikasi. Jika **X** adalah vector mesukan yang berisi fitur dan **Y** adalah label kelas, Naïve Bayes dituliskan dengan **P(X|Y)**. Notasi tersebut berarti probabilitas label kelas **Y** didapatkan setelah fitur-fitur **X** diamati. Notasi ini disebut juga probabilitas akhir (*posterior probability*) untuk **Y**, sedangkan **P(Y)** disebut probabilitas awal (*prior probability*) **Y**. Selama proses pelatihan harus dilakukan pembelajaran probabilitas akhir **P(Y|X)** pada model untuk setiap kombinasi **X** dan **Y** berdasarkan informasi yang didapatdari data latih. Dengan membangun model tersebut, memaksimalkan nilai **P(X|Y)** yang didapatkan.

Formulasi Naïve Bayes untuk klasifikasi adalah :

**P(Y|X)** adalah probabilitas data dengan vector **X** pada kelas **Y.** adalah probabilitas independen kelas **Y** dari semua fitur dalam vector **X**. Nilai **P(X)** selalu tetap sehingga dalam perhitungan prediksi nantinya kita tinggal menghitung bagian dengan memilih yang terbesar sebagai kelas yang terpilih sebagai hasil prediksi. Sementara probabilitas independen tersebut merupakan pengaruh semua fitur data terhadap setiap kelas **Y**, yang dinotasikan dengan formula teorema Bayes.

Model formula teorema Bayes yang selanjutnya akan digunakan dalam proses klasifikasi. Untuk klasifikasi dengan data kontinyu digunakan rumus Densitas Gauss sebagai berikut :

Keterangan :

P = Peluang

Xi = Attribut ke i

xi = Nilai Attribut ke i

Y = Kelas yang dicari

yj = Sub kelas Y yang dicari

u = Mean, menyatakan rata-rata dari seluruh attribute

= Deviasi standart, menyatakan varian dari seluruh attribute.

Parameter bias didapat dari mean sampel **Xi (x)** dari semua data latih yang menjadi milik kelas , sedangkan dapat diperkirakan dari varian sampel data latih.

* + 1. **Karakteristik Naïve Bayes**

Klasifikasi dengan Naïve Bayes bekerja berdasarkan teori probabilitas yang memandang semua fitur dari data sebagai bukti dalam probabilitas. Hal ini memberikan karakteristik Naïve Bayes sebagai berikut :

1. Metode Naïve Bayes bekerja teguh (*robust*) terhadap data-data terisolasi yang biasanya merupakan data dengan karakteristik berbeda (*outliner*). Naïve Bayes juga bias menangani nilai attribute yang salah dengan mengabaikan data latih selama proses pembangunan model dan prediksi.

2. Tangguh menghadapi attribute yang tidak relevan.

3. Attribut yang mempunyai korelasi bias mendegradasi kinerja klasifikasi Naïve Bayes karena asumsi independensi attribute tersebut sudah tidak ada.

* + 1. **Kelebihan dan kekurangan Naïve Bayes**

Teori Bayesian menurut Grainner (1998), mempunyai beberapa kelebihan yaitu :

1. Mudah untuk dipahami.
2. Hanya memerlukan pengkodean yang sederhana.
3. Menangani kuantitatif dan data diskrit.
4. Kokoh untuk titik noise yang diisolaso, misalkan titik yang dirata-ratakan ketika mengestimasi peluang bersyarat data.
5. Lebih cepat dalam perhitungan.
6. Hanya memerlukan sejumlah data kecil pelatihan untuk mengestimasi parameter (rata-rata dan variable) yang dibutuhkan untuk klasifikasi.
7. Cepat dan efisiensi ruang.
8. Kokoh terhadap attribute yang tidak relevan.

Teori probabilitas Bayesian juga memiliki kekurang yaitu :

1. Satu probabilitas tidak bisa mengukur seberapa dalam tingkat keakuratanya. Dengan kata lain, kurang bukti untuk membuktikan kebenaran jawaban yang dihasilkan dari teori probabilitas Bayesian.
2. Banyak celah untuk mengurangi keefektifan probabilitas.
3. Mengasumsikan variable bebas.
4. **Unified Modeling Language (UML)**

UML (Unified Modeling Language) adalah keluarga notasi grafis yang didukung oleh meta model tunggal, yang membantu pendeskripsian dan desain system perangkat lunak, khususnya system yang dibangun menggunakan pemrograman berorientasi objek. (Fowler, 2005)

Menurut Windu Gata, Grace (2013:4),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.

Menurut Nugroho (2009:4), UML (Unified Modeling Language) adalah Metodologi kolaborasi antara metoda-metoda Booch, OMT (Object Modeling Technique), serta OOSE (Object Oriented Software Enggineering) dan beberapa metoda lainnya, merupakan metodologi yang paling sering digunakan saat ini untuk analisa dan perancangan sistem dengan metodologi berorientasi objek mengadaptasi maraknya penggunaan bahasa “pemrograman berorientasi objek” (OOP).

1. **Use Case Diagram**

Use Case merupakan sebuah teknik yang digunakan dalam pengembangan sebuah software atau sistem informasi untuk menangkap kebutuhan fungsional dari sistem yang bersangkutan, Use Case menjelaskan interaksi yang terjadi antara ‘aktor’ — inisiator dari interaksi sistem itu sendiri dengan sistem yang ada, sebuah Use Case direpresentasikan dengan urutan langkah yang sederhana.

Perilaku sistem adalah bagaimana sistem beraksi dan bereaksi. Perilaku ini merupakan aktifitas sistem yang bisa dilihat dari luar dan bisa diuji.Perilaku sistem ini dicapture di dalam *Use Case. Use Case* sendiri mendeskripsikan sistem, lingkungan sistem, serta hubungan antara sistem dengan lingkungannya. Manfaat dari adanya Use Case sendiri adalah :

1. Digunakan untuk berkomunikasi dengan *end user* dan domain *expert.*
2. Memastikan pemahaman yang tepat tentang *requirement* / kebutuhan sistem.
3. Digunakan untuk mengidentifikasi siapa yang berinteraksi dengan sistem dan apa yang harus dilakukan sistem.
4. *Interface* yang harus dimiliki sistem.
5. Digunakan untuk ferifikasi.

Sedangkan dari segi karakteristik, karakteristik dari *Use Case* adalah sebagai berikut :

1. *Use cases* adalah interaksi atau dialog antara sistem dan actor, termasuk pertukaran pesan dan tindakan yang dilakukan oleh sistem.
2. *Use cases* diprakarsai oleh actor dan mungkin melibatkan peran actor lain. Use cases harus menyediakan nilai minimal kepada satu actor.
3. *Use cases* bisa memiliki perluasan yang mendefinisikan tindakan khusus dalam interaksi atau use case lain mungkin disisipkan.
4. *Use case* class memiliki objek *use case* yang disebut skenario. Skenario menyatakan urutan pesan dan tindakan tunggal.

Untuk mendukung dalam pembuatan *Use Case*, berikut adalah penjelasan mengenai komponen yang terdapat pada *Use Case* :

**Tabel 2.1** Tabel Simbol Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Actor* | Menggambarkan sesorang yang beriteraksi dengan sistem, di mana hanya bisa menginputkan informasi dan menerima indormasi dari sistem dan tidak memegang kendali pada use case. Dan biasa actor di gambarkan dengan stickman. |
| 2 |  | *Use case* | Gambaran fungsional sistem yang akan di buat, agar pengguna lebih mengerti penggunaan system. |
| 3 |  | *Association* | Penghubung diatara elemen yang menghubungkan antara *Actor*  dan *Use Case*. |
| 4 |  | *Generalization* | Sebuah elemen yang menjadi spesialisasi dari elemen yang lain. |
| 5 |  | *Dependency* | Sebuah elemen yang bergantung beberapa cara terhadap elemen lainnya. |
| 6 |  | *Include* | Menspesifikasikan bahwa *usecase* adalah sumber secara *eksplisit.* |
| 7 |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas prilaku dari *use case*  pada suatu titik yang diberikan. |
| 8 |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 9 |  | *Collaboration* | Interaksi dari aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah-jumlah elemennya. |
| 10 |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan menunjukan suatu sumber daya komputasi. |

1. **Activity Diagram**

Activity diagram adalah teknik untuk menggambarkan logika procedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram hampir sama dengan sebuah aliran data, akan tetapi perbedaan prinsip antara activity diagram lebih mendukung kepada behavior parallel. (Rijal, 2010)

Sukamto dan Shalahuddin (2013:161), “Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak”.

Activity diagram, merupakan diagram yang menggambarkan tentang aktifitas pada suatu sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah apa saja yang terjadi dalam proses kerja sistem yang dibuat. Sehingga dengan menggunakan activity diagram dapat memudahkan dalam proses pembuatan sistem.

**Tabel 2.2** Tabel Simbol Activity Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Actifity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| 2 |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3 |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| 4 |  | *Actifity Final*  *Node* | Bagaimana objek dibentuk dan dihancurkan . |
| 5 |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |

1. **Sequence Diagram**

Sequence diagram merupakan penjabaran dari *behavior* sebuah scenario tunggal seperti menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini didalam *use case*. (Rijal; 2010)

**Tabel 2.3** Tabel Simbol Sequence Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *LifeLine* | Objek entity, antarmuka yang saling berinteraksi. |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |

1. **Class Diagram**

Sukamto dan Shalahuddin (2013:141), “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.

* 1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
  2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (view) Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case (controller)* Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian use case, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data (model) Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

**Tabel 2.4** Tabel Simbol Class Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 2 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |

1. **Basis Data (Database)**

Basis data berasal dari 2 kata yaitu basis yang berarti tempat penyimpanan atau gudang sebagai tempat berkumpul atau menetap. Sedangkan data merupakan fakta yang diperoleh. Jadi basis data bisa dikatakan sebagai kumpulan data yang saling terkoneksi dan tersimpan secara berkelompok untuk dapat digunakan atau dimanfaatkan dengan cepat dan mudah untuk memenuhi suatu kebutuhan dan disimpan pada suatu media penyimpanan elektronik (Fathansyah,2012:2).

Basis data (Database) adalah sebuah kumpulan dari *files* yang saling berkaitan. (Whitten & Bently, 2007).Teknologi *database* memiliki kelebihan dalam cara penyimpanan data dengan format yang fleksibel. Hal ini memungkinkan dikarenakan *database* terpisah dari sistem informasi dan aplikasi yang menggunakannya. *Database* juga dapat berkembang sesuai dengan perubahan kebutuhan organisasi maupun perkembangan lainnya.

Whitten juga menjelaskan bahwa komponen pembentuk atau penunjang dari *Database* adalah sebagai berikut :

1. Field

Field adalah bagian paling kecil dari data yang memiliki arti atau makna didalam sebuah fileatau *database*.

1. Records

Records adalah sebuah kumpulan dari field-field yang tersusun dan terikat berdasarkan format tertentu yang sudah ditentukan sebelumnya.

1. File atau Tables

File dalam konteks database dalah himpunan semua kejadian yang terjadi pada struktur record. Tables adalah *database* yang mempunyai relasi setara dengan sebuah files.

1. **Istilah Umum Basis Data**

Berikut beberapa istilah dalam basis data yang perlu diketahui diantaranya sebagai berikut :

1. **Entity**

Entityadalah objek dalam enterprise yang akan disajikan di dalam database. Contoh : supplier, gudang, pegawai, part, dll.

1. **Atribut (elemen data).**

Setiap entity memiliki atribut-atribut yang mencirikan entity tersebut. Contoh: entity pegawai terdiri atas atribut : kode pegawai, alamat, dll.

1. **Relasi**

*Relasi* adalah suatu asosiasi hubungan antara dua atau lebih *entity*. Contoh : mahasiswa mengambil mata kuliah, pasien dirawat dokter, dan lain-lain. Dalam hubungan antar *entity* dikenal ada dua kunci data penghubung, yaitu :

1. **Primary Key (Kunci Utama)**

Satu attribute field atau satu set attribute yang mengidentifikasi secara unik suatu kejadian yang spesifik pada entity.

1. **Foreign Key (Kunci Tamu)**

Satu attribute atau satu set attribute yang melengkapi suatu hubungan yang menunjukkan ke entity induknya. Kunci tamu berada pada entity anak.

Setiap proyek dalam sistem database dapat dipresentasikan dengan menggunakan beberapaentity dan setiap entity dapat terdiri dari beberapa elemen data (atribut). Dalam database model relasi dikenal beberapa tipe relasi antara entity dan relasi antara elemen data. Tipe relasi tersebut yaitu:

1. **Relasi satu ke satu (*One to One*)**

Menggambarkan hubungan satu ke satu, yaitu satu record pada entitypertama berhubungan dengan satu record pada entity kedua atau sebaliknya. Misalnya setiap pasien yang dirawat akan menempati satu tempat tidur untuk satu periode perawatan.

1. **Relasi satu ke banyak (*One to Many*)**

Menggambarkan hubungan satu ke banyak, yaitu satu record pada entity pertama berhubungan dengan banyak record pada entity kedua atau sebaliknya. Misalnya antara file data karyawan dengan file pendapatan hariannya. Seorang karyawan memepunya beberapa record data pendapatan hariannya, yang keseluruhannya menunjukkan gaji karyawan tersebut pada periode tertentu.

1. **Relasi banyak ke banyak (*Many to Many*)**

Menggambarkan hubungan banyak ke banyak, yaitu lebih dari satu record pada entity pertama berhubungan dengan lebih dari satu record pada entity kedua atau sebaliknya. Dibutuhkan associativeentity atau entity perantara yang berisi field kunci dari masing-masing entity.

1. **MySQL**

MySQL adalah bagian dari perangkat lunak RDBMS untuk mengolah database dengan cepat, juga dapat menyimpan data dalam jumlah besar, sehingga bisa diakses banyak pengguna, sehingga dapat melakukan proses secara teratur dan bersamaan.

Adapun beberapa alasan mengapa menggunakan MySQL sebagai server database untuk aplikasi yang akan dikembangkan ini (Raharjo,2011) :

1. Fleksibel

MySQL dapat digunakan sebagai pengembangan aplikasi dekstop maupun website dengan menggunakan teknologi yang bervariasi.

1. Performa Tinggi

MySQL memiliki mesin *query* dengan performa yang tinggi, dengan begitu proses transaksional dapat dilakukan dengan cara cepat.

1. Gratis

MySQL dapat digunakan secara gratis dan juga ada yang bersifat komersial. Bersifat komersial biasanya telah ditambahi dengan kemampuan spesifik dan mendapat pelayanan dari *technicial support.*

1. Proteksi data yang handal

MySQL menyediakan mekanisme yang *powerfull* untuk menangani tentang keamanan data dengan menyediakan fasilitas manajemen *user*,enkripsi data, dll.

1. Komunitas Luas

Pengguna MySQL sangatlah banyak dan memilik komunitas yang luas. Hal ini dapat berguna jika terjadi suatu permasalahan dalam pengolahan data menggunakan MySQL. Dengan mengikuti komunitas tertentu, maka dapat menanyakan atau mendiskusikan permasalahan tersebut melalui forum.

1. **PHP**

PHP adalah bahasa scripting yang hanya berjalan pada server dan menyatu dengan HTML untuk membuat konten halaman web yang dapat berubah-ubah. Karena PHP merupakan server-side-scripting maka penulisan dan perintah-perintah PHP dieksekusi di server kemudian hasilnya akan dikirimkan ke browser dengan format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak terdeteksi oleh browser sehingga halaman web lebih aman. PHP dibuat untuk membuat halaman web dinamis, yaitu halaman web yang dapat merangkai suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data di halaman web. (Arief , 2011d:43)

1. **CodeIgniter (CI)**

CodeIgniter merupakan aplikasi open sourceyang berupa frameworkdengan model MVC (Model, View, Controller) untuk membangun website dinamis. CodeIgniter (CI) adalah framework pengembangan aplikasi (Application Development framework) dengan menggunakan PHP, suatu kerangka untuk bekerja atau membuat program dengan menggunakan PHP yang lebih sistematis (Septian, 2011).

1. **Model**

Model adalah yang bertanggung jawab terhadap operasi databse,baik itu crate, read, update, atau delete*.* Ia berupa fungsi-fungsi operasional *database* yang dapat dipanggil oleh controller

1. **View**

Viewadalah yang menangani tampilan. Bagian inilah yang bertugas untuk mempresentasikan data kepada user. Ia berbebtuk struktur HTML yang berisikan variable data yang di kirimkan oleh controller.

1. **Controller**

Controller adalah bagian ynag mengatur hubungan antara Model dan View. Ia adalah otak dari kinerja aplikasi. Terdiri dari fungsi-fungsi yang bersifat operasional dan logika. Saat request yang masuk, akan menangani dan memprosesnya untuk kemudian di tampilkan dalam View.

1. **Sublime Text**

Sublime text adalah teks editor berbasis Python, sebuah teks editor yang elegan, kaya fitur, cross platform, mudah dan simpel yang cukup terkenal di kalangan developer (pengembang), penulis dan desainer. Sampai saat ini sublime text sudah mencapai versi 3 yang rilis secara resmi pada tanngal 29 Januari 2013. Aplikasi ini biasa digunakan programmer untuk menyunting source code program yang sedang kerjakan. (Farid Miftah, 2015)