BAB II
LANDASAN TEORI

II.1. Artificial Intelligence

Artificial Intelligence (AI) atau Kecerdasan Buatan merupakan sebuah sistem atau proses yang mampu berfikir seperti manusia, bertindak seperti yang dapat dilakukan manusia, berfikir secara rasional, dan bertindak secara rasional.

Artificial Intelligence (AI) / Kecerdasan Buatan digunakan untuk menjawab problem yang tidak dapat diprediksi dan tidak bersifat algoritmik atau prosedural. dari riset-riset *Artificial Intelligence (AI)* / Kecerdasan Buatan adalah bagaimana membuat sebuah mesin bisa berfikir sama halnya dengan manusia yang bisa berfikir.[1]

II.1.1. Konsep *Artificial Intelligent*

1. **Berfikir Seperti Manusia (*Think Humanly*)**

Berfikir seperti manusia dapat dijelaskan bahwa system yang dilakukan dengan cara intropeksi yaitu penangkapan pemikiran psikologi manusia pada komputer, hal ini sering diujikan dengan neuron ke neuron lainnya atau sel otak dengan sel otak lainnya cara pembelajarannya yaitu melalui experiment-experimen.

2. **Berfikir Rasional (*Think Rasionally*)**

Think rasional merupakan sistem yang disebut penalaran komputasi. Think rasional sulit diterapkan karena ada dua hal yang perlu digaris bawahi. Pertama tidak mudahnya membuat informal knowledge untuk diterapkan menjadi formal knowledge. Kedua, adanya perbedaan yang sangat besar dalam memecahkan masalah dengan menggunakan prinsip dan aksi secara langsung.

3. **Berperilaku Seperti Manusia (*Act Humanly*)**

Acting humanly merupakan system yang melakukan pendekatan dengan menirukan tingkah laku seperti manusia AI (*Act humanly*) mulai diperkenalkan pada tahun 1950, yaitu dengan cara melakukan pengujian melalui *teletype*. *Teletype* yang

merupakan pengujian dengan melakukan tebakan yang dilakukan oleh interrorator untuk menentukan bahwa persoalan yang ada dikerjakan oleh AI atau manusia. Apabila tebakan dari interrogator salah maka komputer dikatakan AI. [1]

II.1.2. *Natural Language Processing*

Natural Language Processing biasa disebut dengan Pemrosesan Bahasa Alami adalah Bidang *Artificial Intelligence* yang berurusan dengan pemahaman bahasa (alami) manusia (bahasa inggris, jerman, Indonesia dan lain-lain). Bahasa alami atau yang disebut dengan natural language adalah suatu bahasa yang diucapkan, ditulis, disyaratkan oleh manusia untuk berkomunikasi umum. Jadi bahasa alami / *natural language* yaitu bahasa yang dimengerti oleh manusia. Bahasa alami pada prinsipnya adalah suatu bentuk representasi dari suatu pesan yang akan dikomunikasikan antar manusia. Representasinya dapat berupa ucapan/ suara (*spoken language*), namun juga bisa dinyatakan berupa tulisan.

Dalam Pengolahan bahasa alami / *Natural Language Processing* terdiri dari bagian-bagian utama, yaitu : parser, sistem representasi pengetahuan dan pengolahan output.

a. Parser.

   Adalah suatu sistem untuk mengambil kalimat input, dan menguraikannya kata per kata serta untuk menentukan jenis kata apa saja yang dapat mengikuti kata tersebut.

b. Sistem Representasi Pengetahuan.

   Adalah Suatu sistem yang menganalisis *output parser* untuk menentukan maknanya.

c. *Output Translator*.

   Adalah Suatu terjemahan yang merepresentasikan sistem pengetahuan serta melakukan langkah- langkah yang berupa jawaban atas bahasa alami. Output translator merupakan Output khusus yang sesuai dengan program komputer lainnya. [2]

II.1.2.1. *Pattern Matching*

Pattern Matching merupakan algoritma yang digunakan untuk mencari suatu teks maupun memvalidasi suatu teks. Pada zaman ini telah banyak berkembang berbagai jenis algoritma *patten matching*. Dalam ilmu komputer, pencocokan pola adalah tindakan memeriksa urutan tertentu token untuk kehadiran konstituen dari beberapa pola. Berbeda dengan pengenalan pola, pertandingan biasanya memiliki tepatnya. Pola umumnya memiliki bentuk baik urutan maupun struktur pohon. Penggunaan pencocokan pola termasuk keluaran lokasi (jika ada) dari pola dalam urutan tanda, *output* beberapa komponen pola yang cocok, dan untuk mengganti pencocokan pola dengan beberapa urutan tanda lain (yaitu, pencarian dan mengganti). Pola urutan (misalnya, string teks) sering digambarkan menggunakan ekspresi reguler dan cocok menggunakan teknik seperti *backtracking*.

Menulis ulang istilah dan bahasa grafik menulis ulang bergantung pada pencocokan pola untuk cara mendasar program mengevaluasi ke hasilnya.
Diberikan: [3]

1. teks (*text*), yaitu (*long*) *string* yang panjangnya *n* karakter
2. *pattern*, yaitu *string* dengan panjang *m* karakter ($m < n$) yang akan dicari di dalam teks.

II.1.2.2. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh *Donald E. Knuth* pada tahun 1967 dan James H. Morris bersama *Vaughan R. Pratt* pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977.Jika kita melihat algoritma *Brute Force* lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian.

Perhitungan penggeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokan pada saat pattern sejajar dengan teks[i..i + n – 1], kita bisa menganggap ketidakcocokan pertama terjadi di antara teks[I + j] dan pattern [j],dengan 0 < j < n.berarti, teks [i..i + j – 1] = pattern [0..j – 1] dan a = teks [I + j]tidak sama dengan b = pattern[j].Ketika kita menggeser, sangat beralasan bila ada sebuah awalan udari pattern akan sama dengan sebagian akhiran udari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan utersebut sejajar dengan akhiran dari u.dengan kata lain, pencocokan string akan berjalan secara efesien bial kita mempunyai table yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokan di karakter ke- jdari pattern. Tabel itu harus memuat next[j]yang merupakan posisi karakter pattern[j].

setelah digeser, sehingga kita bisa menggeser pattern sebesar j –next[j]relative terhadap teks. Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan string: [7]

1. Algoritma Knuth-Morris-Pratt muali mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per-karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    a) Karakter di pattern dan di teks yang dibandingkan tidak cocok(mismatch).
    b) Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
    c) Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

II.2. Media Pembelajaran

Secara umum media pembelajaran adalah alat bantu dalam proses belajar mengajar. Sesuatu apa pun yang dapat dipergunakan untuk merangsang pikiran, perhatian, perasaan, dan kemampuan atau ketrampilan pebelajar tersebut sehingga dapat mendorong terjadinya proses belajar atau kegiatan pembelajaran. Batasan dari

media pembelajaran ini cukup luas dan mendalam dengan mencakup pengertian sumber, manusia dan lingkungan setra metode yang dimanfaatkan dari tujuan pembelajaran atau pelatihan tersebut.

Singkatnya pengertian media pembelajaran adalah suatu alat sebagai perantara untuk pemahaman makna dari materi yang disampaikan oleh pendidik atau guru baik berupa media cetak atau pun elektronik dan media pembelajaran ini juga sebagai alat untuk memperlancar dari penerapan komponen-komponen dari sistem pembelajaran tersebut, sehingga proses pembelajaran dapat bertahan lama dan efektif, suasana belajar pun menjadi menyenangkan. [4]

II.3. Perangkat Lunak

Perangkat Lunak (*software*) merupakan suatu program yang dibuat oleh pembuat program untuk menjalankan perangkat keras komputer. Perangkat Lunak adalah program yang berisi kumpulan instruksi untuk melakukan proses pengolahan data. *Software* sebagai penghubung antara manusia sebagai pengguna dengan perangkat keras komputer, berfungsi menerjemahkan bahasa manusia ke dalam bahasa mesin sehingga perangkat keras komputer memahami keinginan pengguna dan menjalankan instruksi yang diberikan dan selanjutnya memberikan hasil yang diinginkan oleh manusia tersebut.

Perangkat lunak terdiri dari kode-kode yang dibuat atau ditulis dalam bahasa pemrograman tertentu. Banyak bahasa pemrograman yang dapat dijadikan alat bantu untuk merancang sebuah perangkat lunak antara lain bahasa PASCAL, C, C++, BASIC, Java, HTML, PHP, JavaScript, Python, dan bahkan tersedia *tool* yang berbasis grafis seperti Visual Basic, Visual C++, Borland Delphi, dan masih banyak lagi *tool* dan bahasa pemrograman lainnya. [5]

II.4. Object Oriented Programming (OOP)

Object Oriented Programming adalah sebuah pendekatan untuk pengembangan suatu software dimana struktur software tersebut di dasarkan kepada

interaksi object dalam penyelesaiaan suatu proses. Interaksi tersebut mengambil form dari pesan-pesan dan mngirmkannya kembali antara object tersebut. Object akan merespon pesan tersebut menjadi sebuah tindakan.

Object Oriented terdiri dari object yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, users dari software program dilibatkan dari logika proses untuk menyelesaikan tugas. Adapun karakteristik yang dimiliki OOP yaitu :

a. Object

   Sebuah struktur yang menggabungkan data dan prosedur untuk bekerja bersama-sama

b. Abstraksi

   Ketika membangun object dalam aplikasi OOP, adalah penting untuk menggabungkan konsep abstraksi ini. Jika membangun aplikasi shipping, maka harus membangun object produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan haus dibuang.

c. Encapsulation

   Sebuah proses dimana tidak ada akses langsung ke data yang di berikan, bahkan hidden. Jika ingin mendapat data, haris berinteraksi dengan object yang bertanggung jawab atas data tersebut.

d. Polymorphism

   Kemampuan 2 buah objek yang berbeda untuk merespon permintaan yang sama dalam suatu cara yang unik. Dalam OOP, di terapkan polymorphism melalui proses yang di sebut *overloading*.

e. Inheritance

   Pengguna inheritance dalam OOP untuk mengklarifikasikan object dalam program sesuai karakteristik umum dan fungsi.[6]

II.5. Metode Analisis Yang digunakan

II.5.1. *Unified Model Language*

Kegunaan diagram pada pemodelan adalah untuk formulisasi ekspresi model objek secara koheren, presisi dan mudah dirumuskan. Pemodelan berorientasi objek memerlukan kakas untuk mengekspresikan model. UML menyediakan sejumlah diagran untuk mengekspresikan pemodelan berorientasi objek yang dilakukan.

1. Use case diagram

Diagram use-case digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram use-case menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksi –interaksinya dengan dunia luar. Dengan cara ini, diagram use-case menggantikan diagram konteks pada pendekatan konvensional.

Pemodelan ini biasa dilakukan lewat proses berulang interaksi antara pengembang dan pemakai untuk memperoleh spesifikasi kebutuhan yang sama-sama disepakati.

Pemodelan use-case diciptakan oleh Ivar Jacobson.

Tujuan utama pemodelan use-case adalah:

a. Memutuskan dan mendeskripsikan kebutuhan-kebutuhan fungsional sistem.
b. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan,sehingga model use case digunakan di seluruh proses pengembangan untuk komunikasi dan menyediakan basis untuk pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada use-case.
c. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem.Menguji apakah sistem telah memberikan fungsionalitas yang diminta.

d. Menyediakan kemampuan melacak kebutuhan fingsionalitas menjadi kelas-kelas dan operasi-operasi aktual di sistem.Untuk menyederhanakan perubahan dan ekstensi ke sistem dengan mengubah model use-case dan kemudian melacak use-case yang dipengaruhi ke perancangan dan implementasi sistem.

Elemen diagram use-case adalah:

a) Aktor

Aktor adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain.Aktor adalah sesuatu atau seseorang yang berinteraksi dengan sistem, yaitu siapa atau apa yang menggunakan sistem.Yang dimaksud dengan berinteraksi adalah aktor mengirim atau menerima pesan ke atau dari sistem, atau mempertukarkan informasi dengan sistem. Dalam perspektif aktor, use-case melakukan sesuatu yang berharga bagi aktor.

Aktor adalah tipe (kelas), bukan instan. Aktor merepresentasikan peran bukan pemakai individu dari sistem.Aktor mempunyai nama. Nama yang dipilih seharusnya menyatakan peran aktor.

b) Use-Case

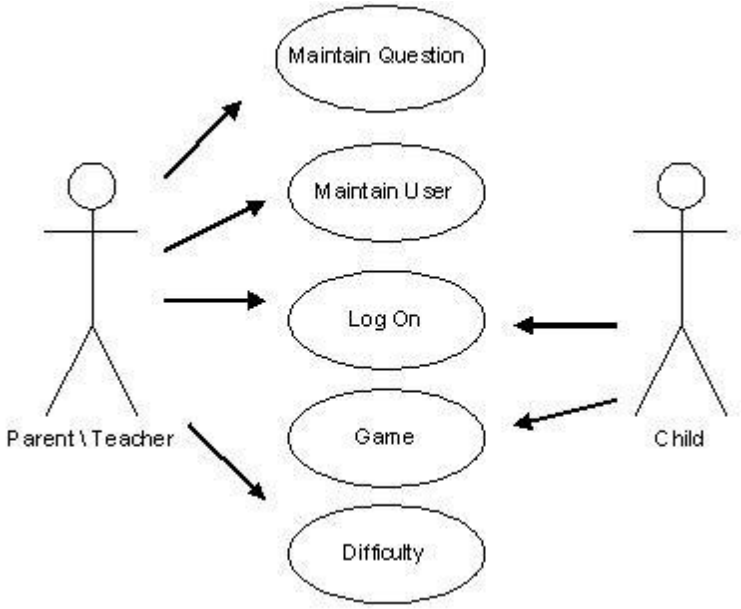
Use-case adalah cara spesifik penggunaan sistem oleh aktor.Ciri-ciri dari use-case adalah:

1. Pola perilaku yang harus dipenuhi oleh sistem.
2. Sekuen transaksi terhubung yang dilakukan aktor dan sistem.
3. Memberikan sesuatu yang berharga bagi aktor.

c) Hubungan antar use-case.

Keterhubungan antar use-case dengan user-case lain berupa generalisasi antar use-case, yaitu:

1. Include, perilaku use-case merupakan bagian dari use-case yang lain.
2. Extend, perilaku use-case memperluas perilaku use-case yang lain.



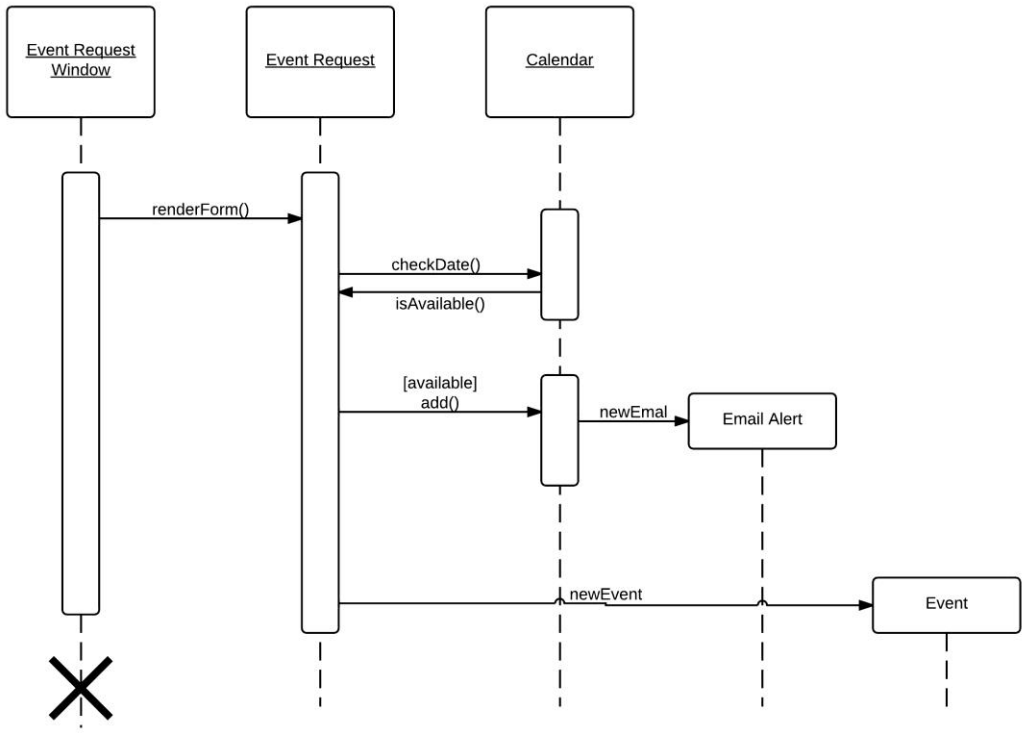
Gambar Error! No text of specified style in document..**6.1. Contoh *Use Case***

2. *Sequence Diagram*

Diagram sekuen mendeskripsikan komunikasi di antara objek-objek , meliputi pesan-pesan yang ada dan urutan pesan tersebut muncul.

Diagram sekuen digunakan untuk:

1. Overview perilaku sistem.
2. Menunjukkan objek-objek yang diperlukan.
3. Mendokumentasikan skenario dari suatu diagram use case
4. Memeriksa jalur-jalur pengaksesan

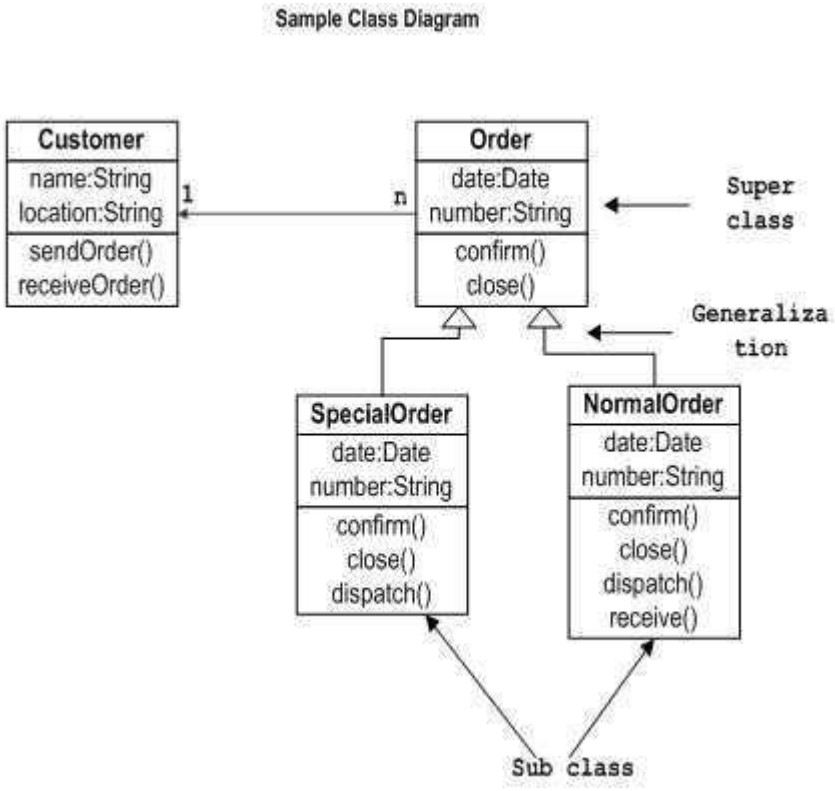


Gambar Error! No text of specified style in document.**.6.2.**

Contoh *Sequence Diagram*

3. *Class Diagram*

Diagram kelas merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek.Pemodelan kelas merupakan pemodelan paling utama di pendekatan berorientasi objek.Pemodelan kelas menunjukkan kelas-kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi-operasi di kelas-kelas.[7]



Gambar Error! No text of specified style in document.**.6.3. Contoh *Class Diagram***

II.6.5.1. Simbol *Flowchart*

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah.Flowchart program merupakan langkah-langkah (instruksi-instruksi) program yang menceritakan kejadian suatu proses satu dengan proses lainya dalam suatu program secara mendetail yang di wakilkan dalam bentuk simbol atau bagan.[8]

II.7. Pengembangan Perangkat Lunak

II.7.1. Python

Python merupakan salah satu bahasa pemrogramman yang memadukan antara kesederhanaan dan kekuatan (*powerfu*l). Anda akan terkejut betapa mudahnya untuk

berkonsentrasi pada penyelesaian pada suatu permasalahan dari pada malah memikirkan syntax dan struktur dari bahasa pemrograman itu sendiri.

Python adalah bahasa pemrograman yang mudah dipelajari dan *powerful*. Python memiliki struktur data tingkat-tinggi yang efisien dan merupakan pendekatan yang sederhana tetapi efektif pada pemrograman yang berorientasi pada objek (*Object-oriented Programming*). *Syntax* elegan dan *dynamic typing* yang dimiliki oleh python, bersama dengan *interpreted nature* dari Python, menjadikannya bahasa pemrograman yang ideal untuk melakukan '*scripting'* dan pengembangan aplikasi yang pesat dalam banyak area pada kebanyakan platform.[9]

II.7.2. Database

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur, dan juga batasan-batasan data yang akan disimpan. Basis data merupakan aspek yang sangat penting dalam sistem informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghidari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga update yang rumit.

Proses memasukkan dan mengambil data ke dan dari media penyimpanan data memerlukan perangkat lunak yang disebut dengan sistem manajemen basis data (database management system | DBMS). DBMS merupakan sistem perangkat lunak yang memungkinkan user untuk memelihara, mengontrol, dan mengakses data secara praktis dan efisien. Dengan kata lain semua akses ke basis data akan ditangani oleh DBMS. Ada beberapa fungsi yang harus ditangani DBMS yaitu mengolah pendefinisian data, dapat menangani permintaan pemakai untuk mengakses data, memeriksa sekuriti dan integriti data yang didefinisikan oleh DBA (Database Administrator), menangani kegagalan dalam pengaksesan data yang disebabkan oleh

kerusakan sistem maupun disk, dan menangani unjuk kerja semua fungsi secara efisien.

Tujuan utama dari DBMS adalah untuk memberikan tinjauan abstrak data kepada pengguna. Jadi sistem menyembunyikan informasi tentang bagaimana data disimpan, dipelihara, dan tetap dapat diambil (akses) secara efisien. Pertimbangan efisien di sini adalah bagaimana merancang struktur data yang kompleks tetapi masih tetap bisa digunakan oleh pengguna awam tanpa mengetahui kompleksitas strukturnya. [10]

II.8. Pengujian Program

II.8.1. *BlackBox Testing*

BlackBox Testing merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

Ciri-Ciri Black Box Testing

1. *Blackbox testing* berfokus pada kebutuhan fungsional pada software, berdasarkan pada spesifikasi kebutuhan dari software.
2. *Blackbox testing* bukan teknik alternatif daripada white box testing. Lebih daripada itu, ia merupakan pendekatan pelengkap dalam mencakup error.
3. *Blackbox testing* melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. juga disebut sebagai *behavioral testing, specification-based testing, input/output testing* atau *functional testing.*

II.8.2. WhiteBox Testing

Metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian white-box). pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar

spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan output yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu. Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan.Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit testing juga.[11]