

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam rangka mendapat hasil penelitian yang baik, selain melakukan penelitian langsung, penulis juga melakukan kajian kepustakaan dari hasil penelitian yang telah dilakukan, berikut hasil penelitian yang menjadi acuan:

Albert Yakobus Chandra, Didik Kurniawan, Rahmat Musa (2020) dalam penelitiannya yang berjudul "Perancangan Chatbot Menggunakan Dialogflow Natural Language Processing (Studi Kasus: Sistem Pemesanan pada Coffee Shop)" pada penelitian ini peneliti menemukan beberapa kasus yang sering dialami di lembaga tertentu seperti Perusahaan Mikro seringkali merupakan staf / karyawan dalam memberikan layanan informasi dan transaksi yang dilakukan secara manual kepada pelanggan terkait dengan kegiatan bisnis ini. Siklus ini selalu berulang dari satu pelanggan ke pelanggan lainnya. Dampaknya jika ada kondisi dimana antrian pelanggan yang cukup ramai dari pada beban kerja staf / karyawan akan lebih tinggi dan risiko kesalahan dalam transaksi juga akan tinggi. Perkembangan teknologi informasi dalam kecerdasan buatan pada era industri 4.0 bergerak maju. Salah satunya adalah Machine Learning - Natural Language Processing (NLP) yang merupakan salah satu ilmu yang berfokus pada bagaimana komputer dapat memahami bahasa manusia dan menanggapi. Untuk itu dalam penelitian ini sistem chatbot akan dibangun dalam memberikan informasi dan melakukan transaksi dengan pelanggan. Chatbot ini akan dikembangkan menggunakan alat Dialogflow yang disediakan oleh Google. Chatbot yang dibangun ini diharapkan dapat menjadi alternatif yang dapat diimplementasikan di berbagai bisnis untuk memberikan layanan yang lebih baik bagi pelanggan.

Radhian, Dimar (2019) dalam penelitiannya yang berjudul "Pembangunan Aplikasi Chatbot Sebagai Media Pencarian Informasi Dalam Bidang Peternakan." Pada penelitian ini Dinas Ketahanan Pangan Dan Peternakan Provinsi Jawa Barat merupakan unsur penyelenggara Pemerintah Provinsi Jawa Barat yang melaksanakan urusan pemerintahan bidang pangan dan bidang pertanian. Dinas

Ketahanan Pangan Dan Peternakan Provinsi Jawa Barat dalam misinya berupaya meningkatkan pengetahuan peternak dalam peternakan seperti budidaya ternak. Kurangnya media informasi yang dimiliki, membuat peternak kesulitan mendapatkan informasi dan pengetahuan dalam budidaya ternak. Oleh karena itu dibangunlah aplikasi chatbot. Chatbot merupakan salah satu bentuk aplikasi NLP (Natural Language Processing). Sistem ini digunakan untuk memudahkan pengguna dalam melakukan pencarian informasi yang berisikan informasi pengetahuan dalam budidaya ternak untuk membantu peternak dalam menambah pengetahuan peternak tentang budidaya. Platform yang digunakan untuk mengembangkan chatbot adalah Dialogflow. Platform ini menyediakan layanan NLP (Natural Language Processing) yang membuat chatbot lebih cerdas sehingga dapat memahami maksud dari apa yang ditanyakan oleh pengguna. Hasil penelitian menunjukkan bahwa 72% peternak setuju aplikasi chatbot ini dapat membantu peternak dalam menambah pengetahuan budidaya ternak dan 78% setuju memudahkan peternak dalam mencari informasi budidaya ternak.

Rahmiati, Rahmiati (2019) dalam penelitiannya yang berjudul “Perancangan Sistem Informasi Desa Tegal Sari Kecamatan Satui Menggunakan *Framework Codeigniter*.” penelitian ini dibuat suatu situs web untuk menjawab masalah dalam bentuk media informasi. Desa Tegal Sari adalah salah satu desa di Kabupaten Satui Kalimantan Selatan yang belum memiliki situs web sebagai media informasi. Setiap komunitas yang ingin mengakses informasi di Desa Tegal Sari masih terbatas, untuk mengakses informasi setiap warga negara akan mendapatkan SMS (*Short Message Service*) atau melalui aplikasi Whatsapp oleh masyarakat kabupaten dan prosesnya masih cenderung rumit. Maka dalam penelitian ini dibuat suatu situs web untuk menjawab masalah dalam bentuk media informasi. Aplikasi ini diharapkan mampu menjawab masalah yang ada di Desa Tegal Sari. Aplikasi yang dikembangkan dalam sistem ini menggunakan kerangka kerja Codeigniter (CI). Peneliti memproses informasi dari Desa Tegal Sari menggunakan kerangka Codeigniter (CI) yang mendukung model,

Permana, Eko (2017) dalam penelitiannya yang berjudul “*Perancangan Sistem Informasi Akuntansi Pada Bina Plus Cisarua Dengan Menggunakan PHP Framework Codeigniter Dan MYSQL*” Peneliti dirancang untuk membangun

sistem informasi akuntansi pengelolaan aset tetap pada asrama bina plus cisarua diharapkan untuk membantu mengurangi masalah yang ada sehingga pengelolaan aset tetap bisa lebih efektif dan efisien. Asrama Bina Plus Cisarua beralamatkan di jalan di Jl. Terusan Kolonel Masturi No. 64 Desa Pasir Halang, RT 001 RW 07, Cisarua, Lembang, Kabupaten Bandung, Jawa Barat 40551 adalah sebuah lembaga dibidang pendidikan. Proses pengelolaan aset tetap pada asrama masih manual dengan menggunakan microsoft excel dan microsoft word, sehingga mengakibatkan sulitnya mengetahui kondisi keseluruhan aset tetap yang mengakibatkan data yang dihasilkan kurang akurat. Jenis penelitian yang peneliti gunakan adalah jenis penelitian akademik, jenis desain penelitian adalah deskriptif analitis, metode penelitian adalah metode survei, teknik pengumpulan data yang peneliti gunakan adalah penelitian lapangan yang terdiri dari wawancara, observasi, dan penelitian kepustakaan, struktur pengembangan sistem yang digunakan adalah waterfall, perancangan sistem informasi yang digunakan adalah diagram sequencediagram , BPMN diagram dan use caseDiagram. Peneliti mengusulkan untuk merancang sistem informasi akuntansi pengelolaan aset tetap pada asrama bina plus cisarua diharapkan untuk membantu mengurangi masalah yang ada sehingga pengelolaan aset tetap bisa lebih efektif dan efisien.

Widyantika, Nuke Brilian (2018) dalam penelitiannya yang berjudul “Perancangan Chatbot Menggunakan *Rivescript* Pada *Website E-Commerce* Sebagai *Virtual Customer Service*.” Penelitian ini yang dikarenakan teknologi sekarang berkembang dan kebutuhan pemilik toko online mlkpancing.com untuk menyelesaikan permintaan layanan pelanggan, oleh karena itu desain sistem dibuat untuk memberikan informasi cepat kepada pelanggan. Penelitian ini bertujuan untuk membangun *chatbot* yang dapat menggantikan peran layanan pelanggan manusia dalam hal memberikan informasi kepada pelanggan menggunakan file yang tersimpan yang berisi informasi stok dalam perdagangan di mlkpancing.com di dalam sistem. Desain dan implementasi perangkat lunak ini menghasilkan prototipe chatbot yang telah dibangun dengan memanfaatkan basis pengetahuan *RiveScript*. *RiveScript* ini membawa input yang dapat diintegrasikan dengan baik oleh *chatbot* sebagai input teks. Selain itu, komunikasi sistem pelanggan dikembangkan. Pemanfaatan chatbot buatan intelijen ditingkatkan ini

dapat menyebabkan kemudahan pelanggan untuk mendapatkan informasi dari database dengan cepat. Sistem chatbot ini lulus kompatibilitas pengujian tanggapan. Hasilnya membuktikan bahwa dari 10 topik pertanyaan, ada 9 tanggapan yang kompatibel dan 1 tidak sesuai. Sementara itu, validitas *respons* dari 10 pertanyaan, ada 8 memberikan informasi valid dan 2 tidak valid. *Chatbot* dengan *RiveScript* dapat membantu pelanggan mendapatkan informasi yang mereka inginkan, dengan kesesuaian jawaban hingga 88%. ada 8 memberikan informasi yang *valid* dan 2 tidak *valid*. *Chatbot* dengan *RiveScript* dapat membantu pelanggan mendapatkan informasi yang mereka inginkan, dengan kesesuaian jawaban hingga 88%. ada 8 memberikan informasi yang valid dan 2 tidak *valid*. *Chatbot* dengan *RiveScript* dapat membantu pelanggan mendapatkan informasi yang mereka inginkan, dengan kesesuaian jawaban hingga 88%.

2.2 Tinjauan Teori

Teori umum pada bab ini menjelaskan tentang penjabaran teori-teori yang di dapat penulis berlandaskan studi pustaka maupun dari buku, yaitu sebagai berikut:

2.2.1 Pengertian Perancangan

Perancangan adalah tahapan perancangan (*design*) memiliki tujuan untuk mendesain sistem baru yang dpat menyelesaikan masalah-masalah yang dihadapi dari pemilihan sistem terbaik. (bin Ladjamudin, 2005).

Perancangan adalah proses pengembangan spesifikasi sitem baru berdasarkan hasil rekomendasi analisis sistem. (Kusrini dkk, 2007).

Perancangan adalah penggambaran, perancangan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi, perencanaan sistem dapat dirancang dalam bentuk bagan alur sistem (*system flowchart*), yang merupakan alat bentuk grafik yang dapat digunakan untuk menunjukan urutan-urutan proses dari sistem. (Syifaun Nafisah, 2003).

2.2.2 Pengertian Aplikasi

Aplikasi adalah suatu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktifitas seperti perniagaan, pelayanan, masyarakat, periklanan atau semua proses yang dilakukan manusia. (Henry, 2004).

Aplikasi adalah program yang memiliki aktifitas pemrosesan perintah yang diperlukan untuk melaksanakan permintaan pengguna dengan tujuan tertentu. (Supriyanto, 2005).

2.2.3 Pengertian Perancangan Aplikasi

Konsep Merancang Multimedia merupakan aplikasi multimedia yang akan dibuat. Untuk dapat merancang konsep dalam membuat aplikasi multimedia dibutuhkan kreatifitas. Kreatifitas adalah kemampuan untuk menyajikan gagasan atau ide baru. Sedangkan inovasi merupakan aplikasi dari gagasan atau ide baru tersebut. Untuk menciptakan ide yang orisinil tidaklah mudah, maka dapat digunakan beberapa teknik untuk menciptakan ide, yaitu penyesuaian (adaptasi), multimedia yang telah ada dianggap belum sesuai dengan lingkungan yang dituju.

Merancang konsep analisis sistem bekerjasama dengan pemakai, mungkin juga bekerjasama dengan professional komunikasi seperti prosedur, sutradara, penulis naskah, editor elektronik terlibat dalam merancang konsep yang menentukan keseluruhan pesan dan membuat aliran (urutan) pada aplikasi multimedia yang akan dibuat. Untuk dapat merancang konsep dalam membuat aplikasi multimedia dibutuhkan kreatifitas. Kreatifitas adalah kemampuan untuk menyajikan gagasan atau ide baru. Sedangkan inovasi merupakan aplikasi dari gagasan atau ide baru tersebut. Untuk menciptakan ide yang orisinil tidaklah mudah, maka dapat digunakan beberapa teknik untuk menciptakan ide, yaitu penyesuaian (adaptasi).

2.2.4 Pengertian Sistem

Sistem berasal dari bahasa Latin (*Systema*) dan bahasa Yunani (*Sustena*) adalah suatu kesatuan yang terdiri dari komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi untuk mencapai suatu tujuan. Pada prinsipnya, setiap sistem selalu terdiri atas empat elemen:

1. Objek yang dapat berupa bagian, elemen, ataupun *variable*. Ia dapat benda fisik, abstrak, ataupun keduanya sekaligus tergantung kepada sifat sistem tersebut.
2. Atribut yang menentukan kualitas atau sifat kepemilikan sistem dan objeknya.
3. Hubungan internal di antara objek-objek di dalamnya.
4. Lingkungan tempat di mana sistem berada.

2.2.4.1 Karakteristik Sstem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu mempunyai komponen-komponen, batas sistem, lingkungan luar sistem, penghubung, masukan, keluaran, pengolah, dan sasaran atau tujuan. (Ladjamudin, 2013)

Adapun penjelasan dari masing-masing karakteristik sistem menurut tersebut adalah sebagai berikut:

a. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

b. Batasan Sistem

Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan dan menunjukkan ruang lingkup dari sistem tersebut.

c. Lingkungan Luar Sistem

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan juga merugikan.

d. Penghubung Sistem

Penghubung sistem merupakan media yang menghubungkan antara satu subsistem dengan subsistem yang lainnya. Melalui penghubung ini kemungkinan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

e. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan dan masukan sinyal maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat berjalan. Sinyal input adalah energi yang diproses untuk mendapatkan keluaran dari sistem.

f. Keluaran Sistem

Keluaran sistem adalah energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran dapat merupakan masukan untuk subsistem yang lain.

g. Pengolahan Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan merubah masukan menjadi keluaran.

h. Sasaran Sistem

Suatu sistem mempunyai tujuan atau sasaran, kalau sistem tidak mempunyai sasaran maka sistem tidak akan ada. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya. Sasaran sangat berpengaruh pada masukan dan keluaran yang dihasilkan.

2.2.5 Pengertian Informasi

Informasi merupakan hasil pengolahan data dari satu atau berbagai sumber yang kemudian diolah, sehingga menghasilkan nilai, arti, dan manfaat. Jadi dapat disimpulkan bahwa informasi adalah data yang telah diolah menjadi sebuah bentuk yang lebih berguna dan bermanfaat bagi yang menerimanya (Pratama E. A., 2014)

Definisi sistem menurut Mulyadi (2016:5), Sistem adalah “suatu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan pokok perusahaan”. Berdasar kan pengertian diatas dapat disimpulkan bahwa

sistem adalah kumpulan dari komponen-komponen yang saling berkaitan satu dengan yang lain untuk mencapai tujuan dalam melaksanakan suatu kegiatan pokok perusahaan.

2.2.5.1 Komponen Sistem Informasi

Komponen sistem informasi terdiri dari:

a. Blok Masukan

Blok masukan (*input*) ini mewakili data yang masuk kedalam sistem informasi. Termasuk dalam blok masukan ini adalah metode-metode dan media untuk menangkap data yang akan dimasukkan.

b. Blok Model

Blok model ini terdiri dari kombinasi antara prosedur, logika dan model matematika yang akan memanipulasi data inputan agar menghasilkan keluaran (*output*) seperti yang diinginkan.

c. Blok Keluaran

Blok keluaran (*output*) ini berupa informasi yang berkualitas serta dokumentasi yang bermanfaat untuk semua pemakai sistem.

d. Blok Teknologi

Blok teknologi adalah kotak alat (*tool-box*) dalam sistem informasi yang digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan output serta membantu pengendalian sistem secara keseluruhan. Teknologi ini terdiri dari 3 bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*).

e. Blok Basis Data

Basis data (*database*) merupakan kumpulan data yang saling terhubung, tersimpan dalam perangkat keras komputer. Yang mana diperlukan perangkat lunak untuk memanipulasinya agar menjadi informasi yang berkualitas.

f. Blok Kendali

Blok kendali adalah suatu perancangan yang diterapkan untuk mencegah terjadinya kesalahan-kesalahan, kegagalan sistem atau yang lainnya agar dapat segera teratasi.

2.2.6 Pengertian Web

Web adalah (Arief 2011a:7) salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, animasi, video) didalamnya yang menggunakan protocol HTTP (*Hypertext Transfer Protoco*) dan untuk mengaksesnya menggunakan perangkat lunak yang disebut broser, semua dokumen web ditampilkan oleh browser dengan cara diterjemahkan. Beberapa jenis browser yang populer saat ini diantaranya adalah Internet Explorer yang diproduksi oleh Microsoft, Mozilla Firefox, Opera, dan Safari.

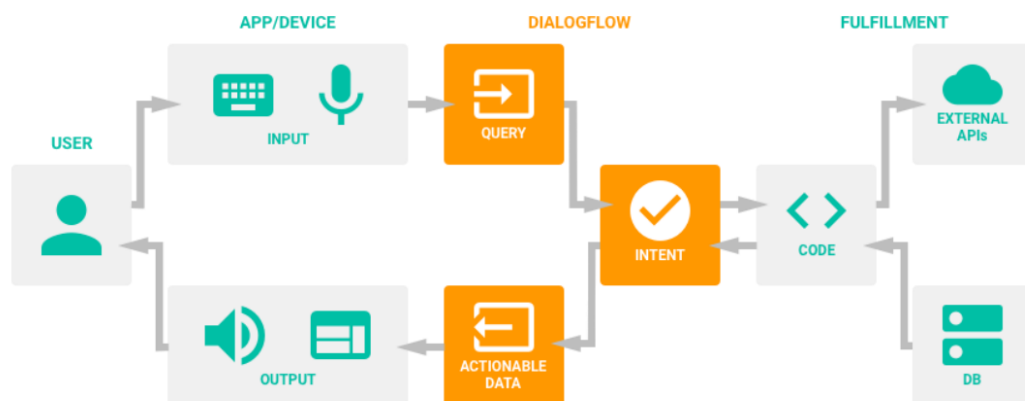
2.2.7 Google Dialogflow

Google Dialogflow adalah sebuah platform yang berbasis pengolahan bahasa alami untuk para pengembang perangkat lunak membuat sebuah sistem chatbot. Google Dialogflow sebelum diakuisi oleh perusahaan Google pada September 2016 bernama Api.ai. Api.ai awalnya dimiliki oleh perusahaan Speaktoit, sebuah perusahaan yang dikenal dengan aplikasi virtual buddy berbasis sistem operasi Android, iOS, dan windows phone yaitu Assistant. Perusahaan Google membeli Api.ai milik Speaktoit, karena platform tersebut menyediakan alat - alat untuk mengembangkan aplikasi - aplikasi pada Google *Virtual Assistant*. Pada 10 Oktober 2017 Api.ai resmi berganti nama menjadi Dialogflow. Berikut adalah fitur – fitur yang ditawarkan pada Google Dialogflow:

1. Dialogflow Agents

Agents mempunyai definisi yang sama seperti *Natural Language Understanding* (NLU). *Agents* dapat digunakan oleh sistem yang pengembang punya untuk dapat mentransformasi permintantaan bahasa alami dari pengguna sistem menjadi sebuah *actionable* data. Transformasi terjadi ketika masukan dari pengguna cocok dengan salah satu intent yang terdapat pada *agent* itu sendiri. *Intent* adalah komponen di *agent* yang

belum ditetapkan atau sudah ditetapkan oleh pengembang yang memproses permintaan – permintaan pengguna. *Agent* dapat didesain untuk mengatur sebuah alur percakapan dengan bantuan *context*, *intent priorities*, *slot filling*, *responsibilities*, dan *fulfillment* dengan perantara *webhook*. Fitur untuk mengatur pembelajaran mesin yang digunakan untuk sistem pengembang. Dalam menu ini terdapat fitur *match mode*, yaitu sebuah pengaturan yang menentukan algoritma yang harusnya digunakan untuk pembelajaran mesinnya. Algoritma yang telah ditentukan akan digunakan pada semua *intent*. Terdapat 2 algoritma yang disediakan oleh Dialogflow agent untuk pembelajaran mesinnya, yaitu: *Rule-based Grammar Matching (Hybrid)* dan *Machine Learning Matching*. Pada Gambar di bawah ini adalah alur kerja Dialogflow Agent yang merupakan alur kerja *agent* pada Google Dialogflow



Gambar 2. 1 Alur Kerja Dialogflow Agent

Sumber: <https://dialogflow.com/docs/images/overview/agents/overview.png>

2. Dialogflow Intent

Intent adalah sebuah representasi sebuah dialog antara apa yang pengguna katakan atau masukan dan apa tindakan atau balasan yang diberikan oleh sistem. *Intent* memiliki beberapa 9 bagian di dalamnya, yaitu: *Context*, *Events*, *Training Phrase*, *Action and Parameters* dan *Responses*.

3. Dialogflow *Entities*

Entity pada Dialogflow adalah sebuah alat yang berfungsi untuk mengekstrak parameter values dari masukan pengguna. *Entity* yang digunakan pada agent tertentu akan bergantung pada parameter *values*, karena *entity* akan memberikan output sesuai dengan fungsi parameter yang ada pada *intent*. Terdapat 3 tipe *entity* pada Dialogflow, yaitu: *System Entities*, *Developer Entities* dan *User Entities*.

4. Dialogflow *Tools*

Dialogflow *tools* merupakan sekumpulan alat – alat untuk membantu pengembangan sebuah *chatbot* pada Dialogflow. Dialogflow *tools* mempunyai 4 alat - alat, yaitu: *Fulfillment*, *Intergrations*, *Training*, *History*, *Analytics* dan *Prebuilt Agents*.

2.3 Aplikasi Pendukung

Berikut ini adalah beberapa aplikasi pendukung yang penulis gunakan dalam membangun sebuah Aplikasi dalam tugas akhir ini adalah sebagai berikut:

2.3.1 XAMPP

XAMPP adalah (Choliviana, Triyono, & Sukadi, 2012) paket PHP dan MySQL berbasis *open source*, yang dapat digunakan sebagai *tool* pembantu pengembangan aplikasi berbasis PHP.

Manfaat Xampp sebagai perangkat lunak bebas yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsi XAMPP sendiri adalah sebagai *server* yang berdiri sendiri (*localhost*). XAMPP merupakan perangkat yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan menginstall XAMPP maka tidak perlu lagi melakukan instalasi dan konfigurasi *web server* Apache, PHP dan MySQL secara manual. XAMPP akan menginstalasi dan mengkonfigurasikannya secara otomatis.

2.3.2 PHP

PHP adalah (Arief 2011c:43) Bahasa *server-side –scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang dinamis. Karena PHP merupakan *server-side-scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di server kemudian hasilnya akan dikirimkan ke *browser* dengan format HTML

PHP singkatan dari PHP *Hypertext Preprocessor* yang merupakan sebuah bahasa *scripting* yang terpasang pada HTML. Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang *web* menulis halaman *web* dinamik dengan cepat. Hubungan PHP dengan HTML, halaman *web* biasanya disusun dari kode-kode HTML yang disimpan dalam sebuah file berekstensi .html. File HTML ini dikirimkan oleh *server* ke *browser*, kemudian *browser* menerjemahkan kode-kode tersebut sehingga menghasilkan suatu tampilan yang indah. Lain halnya dengan program PHP, Program ini harus diterjemahkan oleh *web-server* sehingga menghasilkan kode html yang dikirim ke *browser* agar dapat ditampilkan.

2.3.3 MySQL

MySQL adalah (Arief 2011:152) salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengolahan datanya”.

MySQL dikembangkan oleh perusahaan swedia bernama MySQL AB yang pada saat ini bernama Tcx DataKonsult AB sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak tahun 1979. Awalnya Tcx merupakan perusahaan pengembang *software* dan konsultan *database*, dan saat ini MySQL sudah diambil alih oleh Oracle Corp.

Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat *open source* (tidak berbayar). MySQL merupakan *database* yang pertama

kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi *web* yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP.

2.3.4 Codeigniter

CodeIgniter adalah Kerangka Pengembangan Aplikasi - *toolkit* - untuk orang yang membangun situs *web* menggunakan PHP. Tujuannya adalah untuk memungkinkan Anda mengembangkan proyek lebih cepat daripada yang Anda bisa jika Anda menulis kode dari awal, dengan menyediakan serangkaian perpustakaan yang kaya untuk tugas-tugas yang biasanya dibutuhkan, serta antarmuka yang sederhana dan struktur logis untuk mengakses perpustakaan ini. CodeIgniter memungkinkan Anda secara kreatif fokus pada proyek Anda dengan meminimalkan jumlah kode yang diperlukan untuk tugas yang diberikan.

Menurut Hakim (2010:3) CodeIgniter adalah sebuah *framework* PHP yang dapat membantu mempercepat *developer* dalam pengembangan aplikasi *web* berbasis PHP dibandingkan jika menulis semua kode program dari awal.

Jadi CodeIgniter adalah sebuah *framework* buatan Rick Ellis yang digunakan untuk mempermudah pada *developer* dalam mengembangkan suatu aplikasi *web*.

2.3.5 Bootstrap

Bootstrap sebuah *library framework* CSS yang dibuat khusus untuk bagian pengembang *front-end* website. Menurut Alatas (2013:2) “bootstrap merupakan *framework* ataupun *tools* untuk membuat aplikasi *web* ataupun situs *web responsive* secara tepat, mudah dan gratis”. Lain pula menurut Riyanto (2014:18)

2.3.6 Sublime Text

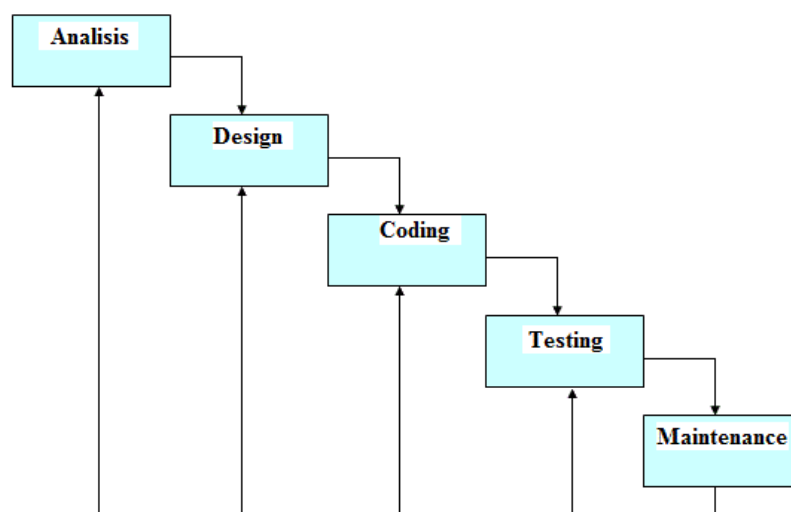
Sublime text salah satu kode editor yang biasa digunakan oleh para programmer untuk membuat suatu program.

Menurut Supono dan Putratama (2016:14) “*Sublime text* merupakan perangkat lunak *text editor* yang digunakan untuk membuat atau meng-edit suatu aplikasi. *Sublime text* mempunyai fitur plugin tambahan yang memudahkan programmer”.

Selain itu, menurut Faridi (2015:3) menjelaskan bahwa “*Sublime Text 3* adalah editor berbasis python, sebuah teks editor yang elegan, kaya akan fitur, *cross platform*, mudah dan simple yang cukup terkenal di kalangan developer (pengembang), penulis dan desainer”.

2.4 Pengertian Metode Waterfall

Metode *Waterfall* adalah suatu metodologi pengembangan perangkat lunak yang mengusulkan pendekatan kepada perangkat lunak sistematis dan sekuensial yang mulai pada tingkat kemajuan sistem pada seluruh analisis, design, kode, pengujian dan pemeliharaan. Model pengembangan sistem yang digunakan adalah *Waterfall Strategy*. Strategi ini mengisyaratkan ‘penyelesaian’ tiap proses satu per satu (Whitten et al, 2004), Model ini melakukan pendekatan secara sistematis dan urut mulai dari level kebutuhan sistem lalu menuju ke tahap analisis, desain, coding, *testing* atau *verification*, dan *maintenance*. Sebagai contoh tahap desain harus menunggu selesainya tahap sebelumnya yaitu tahap requirement. Berikut gambar model waterfall.



Gambar 2. 2 Metode *Waterfall*


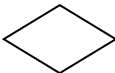
2.5 Teori Perancangan Basis Data

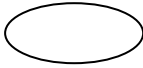


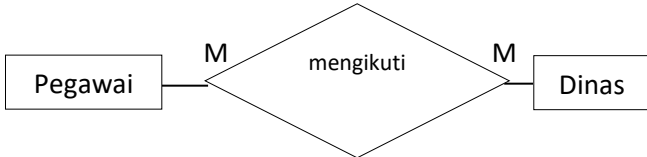
Menurut (Kristanto, 2012), Basis data adalah Suatu sistem penyusunan dan pengelolaan *record-record* dengan menggunakan komputer dengan tujuan untuk menyimpan serta memelihara data operasional lengkap suatu perusahaan atau organisasi, sehingga mampu menyediakan informasi yang optimal yang diperlukan maka untuk proses pengambilan keputusan.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol dan mengakses basis data dengan cara yang praktis dan efisien. DBMS juga dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda.

2.5.1 ERD (*Entity Relationship Diagram*)

ERD atau *Entity Relationship Diagram* merupakan suatu model jaringan yang menggunakan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara entitas yang memiliki jumlah atribut dengan entitas yang lain dalam suatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data (database). Model data ini juga akan membantu pada saat melakukan analisis dan perancangan basis data, karena model data ini akan menunjukkan bermacam-macam data yang dibutuhkan dan hubungan antara data. ERD juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang akan digunakan untuk memodelkan struktur data serta hubungan antara data (Membara, 2014).

Notasi	Keterangan
Entitas 	Entitas adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
Relasi 	Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.

Atribut		Atribut yaitu berfungsi sebagai mendeskripsikan karakter entitas (atribut yang berfungsi sebagai key diberi garis bawah).
Garis		Garis penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.
		One to Many adalah perbandingan antara entity pertama dengan entity kedua berbanding satu berbanding banyak.
		Many to Many yaitu perbandingan antara entity pertama dengan entity kedua berbanding banyak berbanding banyak.

Tabel 2. 1 ERD (*Entity Relationship Diagram*)

2.5.2 LRS (*Logical Record Structure*)

LRS (*Logical Record Structure*) adalah representasi dari struktur *record* pada table-table yang terbentuk dari hasil antar himpunan entitas. LRS dibentuk dengan nomor dari tipe *record*. LRS terdiri dari link-link diantara tipe *record*. *Link* ini menunjukkan arah dari satu tipe *record* lainnya. (Riyanto, 2005).

2.6 *Unified Modelling Language (UML)*

Menurut (Pandawa, 2016) UML (*Unified Modelling Language*) adalah bahasa garis untuk mendokumentasikan menspesifikasikan, dan membangun system perangkat lunak UML berorientasi objek menerapkan banyak level abstraksi, tidak tergantung pada Bahasa dan teknologi, pemanduan beberapa notasi diberagam metodologi, usaha bersama dari banyak pihak. Standar UML dikelola oleh OMG (*Object Management Group*).




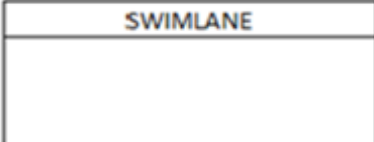


Sedangkan menurut (Akil, 2018, p. 3), *Unified Model Language (UML)* adalah bahasa pemodelan visual yang digunakan untuk mendefinisikan, memvisualisasikan, membangun dan mendokumentasikan rancangan dari suatu sistem perangkat lunak.


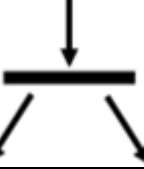


Pada UML terdiri dari beberapa macam diagram berikut penjelasan masing – masing diagram yang penulis gunakan dalam mendesain sistem:

2.6.1 Activity Diagram

Activity Diagram ialah sebuah diagram yang digunakan untuk menggambarkan jalur aktivitas pada sistem yang sedang dirancang. *Activity diagram* juga digunakan untuk menggambarkan *decision* yang mungkin terjadi pada saat sistem dirancang serta digunakan untuk menggambarkan sebuah proses parallel yang mungkin terjadi pada saat eksekusi.

Menurut (Anatasia, 2018, p. 49), *Activity Diagram* yaitu aktivitas yang menggambarkan urutan kegiatan atau urutan aktivitas dari sebuah sistem. Tujuan dibuatnya *activity diagram* adalah untuk memudahkan dalam memahami proses bisnis sistem. Adapun simbol – simbol *activity diagram* dapat dilihat pada tabel berikut:


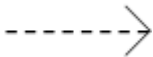
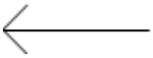
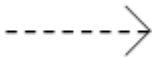
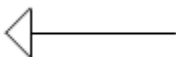
No.	Simbol	Keterangan
1		<i>Start</i> merupakan status awal sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2		<i>Activity</i> merupakan aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		<i>Decision</i> merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari 1.
4		<i>Swimlane</i> digunakan untuk memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
5		<i>Finish</i> merupakan status akhir sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6		<i>Asosiasi</i> penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.






7		<i>Join</i> digunakan untuk menunjukkan kegiatan yang digabungkan.
8		<i>Fork</i> digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel.
9		<i>Miracle Activities</i> digunakan untuk masukan dan keluaran, dipakai pada waktu <i>start point</i> .
10		<i>Black Hole Activities</i> digunakan untuk masukan dan keluaran.

Tabel 2. 2 Simbol Activity Diagram

2.6.2 Use Case Diagram

Menurut (Anatasia, 2018, p. 45), *Use Case Diagram* adalah diagram yang menggambarkan interaksi antara pengguna (*actor*) dengan sistem informasi yang akan dibuat. *Use case* menggambarkan siapa saja actor yang terlibat dan fungsi apa saja yang dapat digunakan *actor* pada sistem informasi tersebut. *Use Case Diagram* terdiri dari *Use Case*, *Actor*, *Relationship*, *System Boundary* atau batas sistem (opsional). Adapun simbol – simbol *Use Case Diagram* dapat dilihat pada tabel berikut:


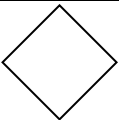


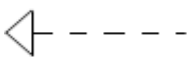
No	Gambar	Nama	Keterangan
1		Actor	Menspesifikasi antara himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case.
2		Depedency	Hubungan perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
3		<i>Generalization</i>	Hubungan dimana objek anak(<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
4		Include	Menspesifikasikan atau bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
5		<i>Extend</i>	Menspesifikasikan atau bahwa <i>use case</i> target memperluas perilaku dari

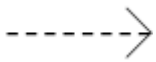

			<i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa sajakah yang menghubungkan antara objek satu dengan objek lainnya.
7		<i>System</i>	Menspesifikasi paket yang menampilkan sistem secara terbatas.
8		<i>Use case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemen.
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 3 Simbol *Use Case* Diagram

2.6.3 Class Diagram

Menurut (Anatasia, 2018, p. 56), *Class diagram* menggambarkan struktur dan deskripsi *class*, *package*, dan objek serta hubungan satu sama lain seperti *inheritance*, *association*, dan lain – lain. Dibawah ini adalah gambaran dari simbol - simbol *class* diagram:


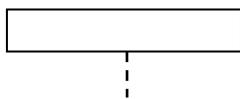
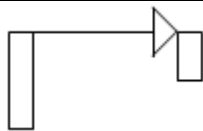
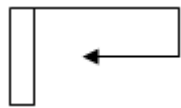
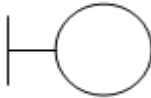

No	Gambar	Nama	Keterangan
1		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagai perilaku dan struktur data dari objek yang ada diatasnya objek induk (<i>ancestor</i>).
2		<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		<i>Collaboration</i>	Deskripsi dari urutan aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.

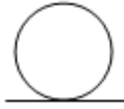
6		<i>Dependency</i>	Hubungan yang dimana ada perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek yang lainnya.

Tabel 2. 4 Simbol *Class Diagram*

2.6.4 *Sequence Diagram*

Menurut (Anatasia, 2018, p. 51), *Sequence Diagram* menggambarkan interaksi antar objek dalam sistem. *Sequence Diagram* digunakan untuk menggambarkan scenario pada use case. Jumlah *Sequence Diagram* harus sama dengan jumlah use case. Adapun simbol – simbol *Sequence Diagram* dapat dilihat pada gambar dibawah ini:

No	Gambar	Nama	Keterangan
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Life Line</i>	Objek <i>entity</i> , antarmuka yang saling berinteraksi.
3		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
4		<i>Message to Self</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
5		<i>Boundary</i>	Untuk menunjukkan adanya <i>interfaces requirement</i> , bukan untuk menunjukkan bagaimana <i>interface</i> yang akan di implementasikan.
6		<i>Controller</i>	Sebuah control adalah meniru suatu kelas yang menggambarkan suatu

			pengendalian manajer atau kesatuan. Suatu control mengorganisir dan menjadwalkan aktivitas.
7		Entity	Suatu entity adalah tempat atau ketekunan mekanisme yang menangkap pengetahuan dan informasi di dalam suatu sistem.

Tabel 2. 5 Simbol *Sequence Diagram*

2.7 Notasi UML

Notasi UML dturunkan dari tiga notasi yang sudah ada sebelumnya yaitu Grady Booch OOD (*Object Oriented Design*), Jim Rumbough OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*). Notasi UML yang digunakan sekarang merupakan penggabungan dari tiga notasi tersebut, yang terdiri dari: (Nurhasanah, 2014)

1. Aktor (*Actor*)

Dalam pemodelan sistem dengan UML, actor adalah seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan. *Actor* berada diluar lingkup sistem atau perangkat lunak yang sedang kita kembangkan, bersifat eksternal.

Secara prinsip dapat kita kenali 3 jenis actor untuk hampir semua sistem atau perangkat lunak yang kita kembangkan: para pengguna sistem, perangkat lunak yang berinteraksi dengan sistem, dan perangkat lunak yang kita kembangkan serta waktu. Jadi aktor ini bisa berupa orang, perangkat keras atau mungkin juga objek lain dalam sistem yang sama. Biasanya yang dilakukan oleh aktor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu.

Kelas seperti juga objek, adalah sesuatu yang membungkus (encapsulate) informasi dan perilaku dalam dirinya. Dalam pengembangan system tradisional, kita mengadakan pendekatan dengan cara memisahkan informasi-informasi pada sisi basis data dan perilaku yang mengaksesnya di sisi aplikasi pemasup atau pengakses. Pendekatan berorientasi objek berbeda, yaitu menggabungkan potongan-potongan informasi dengan perilaku yang akan mengaksesnya dalam apa yang dinamakan kelas.

2. *Usecase*

Usecase adalah peringkat tertinggi dari fungsional yang dimiliki system. Dengan kata lain, *usecase* menggambarkan bagaimana seseorang akan menggunakan sistem. *Usecase* menjelaskan suatu urutan kegiatan yang dilakukan oleh aktor dan sistem untuk mencapai tujuan tertentu walaupun menjelaskan kegiatan namun usecase hanya menjelaskan apa yang dilakukan oleh aktor dan sistem, bukan bagaimana aktor dan sistem melakukan kegiatan tersebut.

Keunggulan dari cara memandang sistem sebagai kumpulan usecase adalah kemampuannya untuk memisahkan implementasi sistem dari alasan mengapa sistem harus ada. Ia akan membantu kita untuk berfokus pada apa yang paling penting, yaitu menentukan apa yang dibutuhkan serta apa harapan pengguna terhadap sistem atau perangkat lunak yang sedang dikembangkan.

3. *Interaction*

Interaction digunakan untuk menunjukan baik aliran pesan atau informasi antara objek. Biasanya interection ini dilengkapi juga dengan teks bernama operationsignature yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

4. *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam *interface* dijabarkan dalam operasi dalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam objek.

5. *Package*

Package adalah *container* atau wadah konseptual yang digunakan untuk mengelompokkan elemen-elemen dari suatu sistem yang sedang dibangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya

adalah untuk mempermudah penglihatan (*visibility*) dari suatu model yang sedang dibangun.

6. *Note*

Note dibangun untuk memberikan keterangan dan komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa ditempelkan ke semua elemen notasi yang lain.

7. *Dependency*

Merupakan relasi yang menunjukkan bahwa perubahan pada suatu elemen memberikan pengaruh pada elemen yang lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa ada tanda panah. Terdapat dua *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada di garis tanpa panah) memicu eksekusi bagian dari elemen yang lain (yang ada di garis dengan panah), misalnya untuk notasi A B operasi yang ada di class A memicu dieksekusinya operasi yang berada di class B.

Extend menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disiapkan kedalam elemen yang ada di garis dengan panah, misalnya untuk notasi A B suatu fungsi dari *usecase* A bisa disisipkan ke dalam *usecase* B atau dengan kata lain A optimal untuk B. Kedua *stereotype* ini di representasikan dengan menambahkan *text include* atau *extend* di notasi *dependency*.

8. *Association*

Association menggambarkan navigasi antara *class* (*navigation*), berapa banyak objek lain yang bisa berhubungan dengan satu objek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).

Navigation di lambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukkan bahwa dengan mengetahui salah satu class bisa didapatkan dari informasi lainnya.

Sementara dengan *unidirectional navigation* hanya dengan mengetahui *class* di ujung garis *association* tanpa panah kita bisa mendapatkan informasi dari *class* di ujung dengan panah, tetapi tidak sebaliknya.

Pada penelitian kali ini penulis menggunakan visual paradigm untuk merancang aplikasi. Visual paradigm adalah perangkat lunak yang digunakan untuk perancangan aplikasi dengan tools UML.

2.8 Pengujian Perangkat Lunak

Pengujian adalah aktivitas yang digunakan untuk dapat melakukan evaluasi suatu atribut atau kemampuan dari program atau sistem dan menentukan apakah telah memenuhi kebutuhan atau hasil yang diharapkan. (Nidhra and Dondetti, 2012)

Menurut Munawar (2018, p. 38), Pengetesan *software* adalah kegiatan yang sangat penting sebelum diimplementasikan. Beberapa organisasi besar, biasanya memiliki bagian khusus untuk pengetesan *software* sebelum diimplementasikan. Namun tidak semua organisasi memiliki bagian khusus untuk pengetesan *software*. Oleh karena itu, beberapa organisasi melibatkan analisis dalam pengetesan, karena analisis lebih memahami kebutuhan organisasi sehingga bisa mengatur kinerja *software* dari sisi kebutuhan fungsional dan non-fungsional.

Beberapa metode yang biasa digunakan untuk melakukan pengujian antara lain *Black Box* dan *White Box Testing*.

2.8.1 Black Box Testing

Black Box Testing merupakan sebuah metode yang digunakan untuk menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah input diterima dengan benar dan output yang dihasilkan telah sesuai dengan yang diharapkan. (Nidhra and Dondetti, 2012).

Pengujian perangkat lunak mempunyai beberapa level, untuk pengujian menggunakan metode *Black Box*, terdapat enam level yaitu *Integration*, *Functional*, *System*, *Acceptance*, *Beta*, dan *Regression*.

Salah satu dari pengujian *Black Box* yang dapat dilakukan oleh seorang penguji independen adalah *Functional Testing*. Basis uji dari *Functional Testing* ini adalah pada spesifikasi dari komponen perangkat lunak yang diuji.

Fokus dari pengujian menggunakan metode *Black Box* adalah pada pengujian fungsionalitas dan output dihasilkan aplikasi. Pengujian *Black Box* didesain untuk mengungkap kesalahan pada persyaratan fungsional dengan mengabaikan mekanisme internal atau komponen dari suatu program. *Functional Testing* memastikan bahwa semua kebutuhan – kebutuhan telah dipenuhi dalam sistem aplikasi. Dengan demikian fungsinya adalah tugas – tugas yang didesain untuk dilaksanakan sistem. *Functional Testing* berkonsentrasi pada hasil dari proses bukan bagaimana proses terjadi

Black Box Testing cenderung untuk menemukan hal-hal berikut:

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (*interface errors*).
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi (*performance errors*).
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan *valid*?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada *input-input* tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metode atau teknik untuk melaksanakan *Black Box Testing*, antara lain:

1. *Equivalence Partitioning*
2. *Boundary Value Analysis/Limit Testing*
3. *Comparison Testing*
4. *Sample Testing*
5. *Robustness Testing*

6. *Behavior Testing*
7. *Requirement Testing*
8. *Performance Testing*
9. Uji Ketahanan (*Endurance Testing*)
10. Uji Sebab-Akibat (*Cause-Effect Relationship Testing*)