BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

A. Konsep Dasar Sistem Informasi

Menurut Yakub (2012:1) "sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, terkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk tujuan tertentu".

Menurut Rosa dan Shalahuddin (2013:4) "Rekayasa Perangkat Lunak (*Software Engineering*) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin". Perangkat lunak banyak dibuat dan pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keenggan pemakai perangkat lunak atau (*User*) untuk mengubah cara kerja dari manual ke otomatis, atau ketidakmampuan user menggunakan komputer. Oleh karena itu, rekayasa perangkat lunak dibutuhkan agar perangkat lunak yang dibuat tidak hanya menjadi perangkat lunak yang tidak terpakai.

Menurut Rosa dan Shalahuddin (2013:28) "Model Air Terjun (Waterfall) sering juga disebut model Sekuensial linier (*Sequential Linear*) atau alur hidup klasik (*Classic Life Cycle*)". Model air terjun menyediakan pendekatan alur hidup perangkat

lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung (*support*).

B. Konsep Dasar Web

Menurut Yuhefizar (2008:161) "mengemukakan mengemukakan bahwa *website* adalah kumpulan dari halaman *web* yang terdapat pada suatu domain, yang terdiri dari dua atau lebih halaman *web".*

C. *Unified Modelling Language (UML)*

Menurut Rosa dan Shalahuddin (2013:133) "*UML (Unified Modelling Language)* adalah salah satu standar bahasa yang banyak digunakan di dunia industry untuk mendefinisikan *requirement,* membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

Menurut Rosa dan Shalahuddin (2013:140) *UML (Unified Modelling Language)* dapat dikelompokan atas:

1. *Use Case Diagram*

Menurut Rosa dan Shalahuddin (2013:155) "*Use Case* merupakan pemodelan untuk tingkah laku (*behavior*) sistem informasi yang akan dibuat". *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

2. *Activity Diagram*

Menurut Rosa dan Shalahuddin (2013:161) "*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak". *Activity Diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar *transisi* di *trigger* oleh selesainya *state* sebelumnya (*internal processing).* Oleh karana itu *Activity Diagram* tidak menggambarkan *behavior internal* sebuah sebuah sistem dan interaksi antar subsistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari *level* atas secara umum.

3. *Component Diagram*

Menurut Rosa dan Shalahuddin (2013:148) "*Component Diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem". Sebuah komponen bisa mengakses services yang ada pada komponen lain. Komponen ini menyediakan services tersebut disebut export interface sedangkan yang mengaksesnya disebut import interface. *Component Diagram* menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk ketergantungan (*dependency)* diantara komponen.

D. *Entity Relationship Diagram (ERD)*

Menurut Yakub (2012:60) *" Relationship Diagram (ERD)* merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak". *ERD* juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas yang lain dalam suatu sistem yang terintegrasi. Komponen-komponen yang terdapat didalam *Entity Relationship Model,* yaitu:

1. **Entiti** merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari entity biasanya digambarkan dengan persegi panjang.
2. **Atribut** setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasi isi elemen satu dengan yang lain.
3. **Relasi** hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.
4. ***Indicator Type***
    a. *Indicator Type Associative Object*

        Berfungsi sebagai suatu objek dan suatu *relationship.*

    b. *Indicator Type Supertipe*

        Terdiri dari suatu object dan satu sub kategori atau lebih yang dihubungkan dengan satu *relationship* yang tidak bermakna.

5. ***Cardinatlity Ratio atau Mapping Cardinality,*** *Cardinality Ratio* adalah menjelesakan hubungan batasan jumlah keterlambatan satu *entity* lainnya atau banyaknya *entity* yang bersesuain dengan *entity* yang lain melalui *relationship.*
6. ***Derajat Relationship***

   Derajat Relationship menyatakan jumlah *entity* yang berpatisipasi didalam suatu *relationship*
7. ***Participation Constraint***

   Menjelaskan apakah keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain.

E. LRS *(Logical Record Structure)*

Menurut Frieyadie (2007:13) "*LRS (Logical Record Structure)* merupakan hasil pemodelan Entity Relationship (ER) beserta atributnya sehingga bisa terlihat hubungan-hubungan antar entitas".

Menurut Frieyadie (2007:14) dalam pembuatan *LRS* terdapat tiga hal yang dapat mempengaruhi, yaitu:

1. Jika tingkat hubungan (*Cardinality)* satu pada satu (*one-to-one),* maka digabungkan dengan entitas yang lebih kuat (*strong entity),* atau digabungkan dengan entitas yang memiliki atribut yang lebih sedikit.

2. Jika tingkat hubungan (*Cardinality)* satu pada banyak (*one-to-many),* maka hubungan relasi atau digabungkan dengan entitas yang tingkat hubungannya banyak.
3. Jika tingkat hubungan (*Cardinality)* banyak pada banyak (*one-to-many),* maka hubungan relasi tidak akan digabungkan dengan entitas manapun, melainkan menjadi sebuah LRS.

F. Pemrograman Terstruktur

Menurut Rosa dan Shalahuddin (2013:67) "Pemrograman terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer". Modul-modul (pembagian program) biasanya dibuat dengan mengelompokan fungsi-fungsi dan prosedur-prosedur yang diperlukan sebuah proses tertentu.

G. PHP

Menurut Hidayatullah dan Kawistara (2015:231) "*PHP (Hypertext Preprocessor)* adalah suatu bahasa *scripting* khususnya digunakan untuk *web development"*. Karena sifatnya yang server side scripting, maka untuk menjalankan *PHP* harus menggunakan *web server*. PHP juga dapat diintegrasikan dengan HTML, JavaScript, JQuery, Ajax. Namun pada umumnya PHP lebih banyak digunakan bersamaan dengan *file* bertipe HTML.

H. HTML

Menurut Hidayatullah dan Kawistara (2015:13) "*Hypertext Markup Language* (HTML) adalah bahasa standar yang digunakan untuk menampilkan halaman *web"*. Yang bisa dilakukan dengan HTML yaitu:

1. Mengatur tampilan dari halaman web dan isinya.
2. Membuat tabel dalam halaman web.
3. Mempublikasikan halaman web secara online.
4. Membuat form yang bisa digunakan untuk menangani registrasi dan transaksi via web.
5. Menambahkan objek-objek seperti citra, audio, video, animasi, java applet dalam halaman web.
6. Menampilkan area gambar (*canvas)* di browser.

I. MySQL

Anhar (2010:21) "MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS dari sekian banyak DBMS, seperti Oracle, MS SQL, Postagre SQL, dan lain-lain". MySQL merupakan DBMS yang *multithread, multi-user*, yang sifatnya gratis dibawah lisensi GNU General Public Licence (GPL)".

J. CSS

Menurut Andi (2008:100) "*CSS (Cascade Style Sheet)* adalah sebuah fitur yang diperkenalkan sejak HTML versi 4.0 dan berfungsi untuk menangani masalah tampilan pada HTML seperti jenis, ukuran, dan warna font, posisi teks, batas tulisan atau margin, warna background". Hal penting yang perlu diperhatikan adalah cara meletakkan CSS dan juga bahasa berbasis web lain untuk memudahkan manajemen file, editing, dan maintenance.

K. *White Box Testing*

Menurut Rizki (2011:261) "*White Box Testing* secara umum merupakan jenis *testing* yang lebih berkonsentrasi terhadap "isi" dari perangkat lunak itu sendiri. Jenis ini lebih banyak berkonsentrasi pada s*ource code* dari perangkat lunak yang dibuat.

L. *Black Box Testing*

Menurut Rizki (2011:264) " *Black Box Testing* adalah tipe testing yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya". Sehingga para *tester* memandang perangkat lunak seperti layaknya sebuah "kotak hitam" yang tidak penting dilihat isinya, tetapi cukup dikenai proses *testing* dibagian luar.

2.2. Penelitian Terkait

Dengan pembahasan atau permasalahan yang berbeda-beda dan aplikasi yang berbeda-beda. Berikut ini beberapa bentuk sistem IT *Helpdesk* yang pernah dibuat oleh beberapa sumber:

> Menurut Tarmuji (2008:146), Langkah-langkah yang perlu dilakukan oleh instansi dalam implementasi tim *Helpdesknya* antara lain: Membentuk tim khusus untuk menganalisa organisasinya terkait kebutuhan implementasi *Helpdesk* tersebut, Membentuk organisasi *Helpdesk* sesuai kebutuhan, Memilih *workflow Helpdesk* yang disesuaikan dengan kondisi instansi, Memilih *framework* yang tepat untuk mendasari jalanya implementasi *Helpdesk* di instansi terkait, Menuangkan aturan-aturan dasar (*standar operation procedur-SOP).*

> Menurut Purwanto (2011:A-101), Bagi orang awam teknologi informasi merupakan sebuah hal yang asing dan tidak mereka pelajari dalam ruang lingkup pekerjaan mereka. Oleh karena itu sudah merupakan kewajiban dari bagian teknologi informasi ini untuk membantu agar penerapan teknologi informasi yang ada diperusahaan mereka dapat membantu pekerjaan mereka bukan malah memperlambat. Oleh karena itu dibutuhkan sebuah media untuk dapat membantu para pegawai agar dapat mengatasi masalah-masalah yang berasal dari teknologi informasi baik termasuk masalah perangkat keras komputer maupun perangkat lunak sehingga sedikit banyak para pegawai tersebut dapat menyelesaikan sendiri masalah-masalah kecil yang muncul dan disisi lain akan meringankan tugas dari pegawai bagian teknologi informasi. Pengembangan sistem Help Desk Troubleshooting Hardware dan Software yang akan dikembangkan ini menggunakan methodologi zeand frame work, dirasakan pengatasan masalah dewasa ini masih tidak maksimal karena tidak adanya interaksi antara user dan bagian IT padahal sistem ini sejatinya dibuat hal tersebut menjadi key word dalam pengentasan permasalahan sesuai dengan methode yang akan dikembangkan dan sebagai salah satu sarana agar antara bagian IT bisa meningkatkan pelayanan terhadap user tanpa user merasa ditinggalkan merupakan alasan pengembangan dengan menggunakan methodologi tersebut.